

UiO : **Department of Informatics**
University of Oslo

Implementation of a digital Player Monitoring System: pmSys

Cong Nguyen Nguyen
Master's Thesis Spring 2015



Implementation of a digital Player Monitoring System: pmSys

Cong Nguyen Nguyen

Abstract

Today football matches are often settled by the smallest margin possible. Teams will therefore try to find ways of gaining even the smallest of advantages over their opponent. Success and failure is often determined by how a team can keep their players healthy and rejuvenated throughout the course of a long season, which includes training matches, league matches and, for the top teams, tournament matches in Europe. Although it is impossible to predict whether a player will break a leg, injuries relating to overuse can be detected, and in many cases avoided. If we monitor a player daily, and build a strong foundation of data, it is possible to track the players condition so that we can maximize his performance, avoid fatigue and overtraining, while also prevent the player for getting injured as a result of overuse.

In this thesis, we will discuss ways to design and develop a tailor-made system that will work as a supplement for trainers and coaches so that they can monitor their players. By giving football teams a better alternative for monitoring their players, rather than using a pen and paper, or even improving the already existing applications and systems, we hope to provide them with the edge that is often needed to succeed.

The system is today deployed to the Norwegian national football team and several Norwegian Top Division teams. It is used daily by the coaches to monitor and supervise their players.

Contents

1	Introduction	1
1.1	Background	1
1.2	Problem Definition	2
1.3	Limitations	2
1.4	Research Method	2
1.5	Main Contributions	3
1.6	Outline	3
2	Background and Related Work	5
2.1	Sports related background	5
2.1.1	Rating of Perceived Exertion	5
2.1.2	Wellness	6
2.1.3	Injury	6
2.2	Participatory Sensing	7
2.3	Open mHealth	8
2.4	Ohmage	9
2.5	DHIS 2	10
2.6	Milan Lab	11
2.7	Summary	12
3	PmSys	13
3.1	Overview	13
3.2	Functional requirements	13
3.2.1	Collect data	13
3.2.2	Present data	14
3.2.3	Analysis of data	14
3.2.4	Tools for assistance	14
3.2.5	Cross-platform support	14
3.3	Non-functional requirements	15
3.3.1	Usability	15
3.3.2	Scalability	15
3.3.3	Availability	15
3.3.4	Privacy and Security	15
3.3.5	Performance	15
3.4	Development methodology	15
3.5	PmSys: version 1	16
3.5.1	Ohmage mobile applications	17
3.5.2	Ohmage Web FrontEnd	17
3.5.3	Ohmage back-end	17
3.6	Summary	17

4	Ohmage back-end	19
4.1	First version of back-end	19
4.2	Ohmage web API	20
4.3	User	21
4.3.1	Role	22
4.4	Class	22
4.5	Campaign	22
4.6	Configuration	25
4.7	Summary	26
5	pmSys-app	27
5.1	Related work: Ohmage mobile applications	27
5.1.1	Login	27
5.1.2	Campaign (Projects) and Surveys	28
5.1.3	Reporting	29
5.1.4	Queue	30
5.1.5	Notifications	31
5.1.6	Feedback	32
5.1.7	Summary	33
5.2	pmSys-app features	33
5.3	System Design	33
5.3.1	Single Page Application	33
5.3.2	Hybrid mobile application	35
5.3.3	Model-View-Controller (MVC) model	37
5.3.4	AngularJS	37
5.3.5	Design summary	40
5.4	Implemented modules	40
5.4.1	Login	40
5.4.2	Reporting	41
5.4.3	Visualization	44
5.4.4	Notification	46
5.4.5	File system and Offline-mode	48
5.4.6	Miscellaneous	49
5.5	Deployment	50
5.5.1	Apples App Store - iOS	50
5.5.2	Google Play - Android	50
5.6	Evaluation	50
5.7	Summary	55
6	pmSys-trainer	57
6.1	Related work: Ohmage Web FrontEnd	57
6.1.1	Presentation	57
6.1.2	Survey response read	58
6.1.3	Visualization module	59
6.1.4	Manual work	62
6.1.5	Tools for assistance	62
6.1.6	Ohmage Web Summary	62
6.2	pmSys-push module	62
6.2.1	Features for pmSys-push	63
6.2.2	Apple Push Notification	63
6.2.3	Google Cloud Messaging	63
6.2.4	Tokens	64
6.2.5	Cron	65
6.3	pmSys-trainer features	65

6.4	System design	65
6.4.1	NodeJS with Express	65
6.4.2	Redis	68
6.4.3	Bootstrap	68
6.4.4	NPM	69
6.5	Implementation	69
6.5.1	Login	69
6.5.2	Push messaging	69
6.5.3	Survey responses	71
6.5.4	Visualization	72
6.6	Evaluation	74
6.7	Summary	75
7	Automatic data analysis	77
7.1	Features	77
7.2	Background	77
7.2.1	Rating of Perceived Exertion	77
7.2.2	Wellness	78
7.2.3	Baselines	78
7.3	Implementation	78
7.3.1	Analysis of wellness	80
7.3.2	Analysis of training load	80
7.3.3	Player profile	81
7.4	Discussion	82
7.5	Summary	83
8	Conclusion	85
8.1	Summary	85
8.2	Main Contributions	86
8.3	Future work	86
8.3.1	PmSys-app	86
8.3.2	PmSys-trainer	86
8.3.3	Ohmage back-end	87
A	Accessing the source code	89
A.1	pmSys-app	89
A.2	pmSys-trainer	89
A.3	pmSys-push	89
B	User Surveys	91
B.1	pmSys vs. Ohmage	91
B.2	Rating of pmSys	95
C	Surveys in pmSys	99
C.1	RPE Survey	99
C.2	Wellness survey	102
C.3	Injury survey	105

List of Figures

2.1	RPE scale [9]	5
2.2	Wellness questionnaire [10]	6
2.3	Injury questionnaire: key questions [11]	6
2.4	Injury questionnaire: follow-up questions [11]	7
2.5	Common architecture for PS applications [13]	8
2.6	Stovepipe architecture (left) and Open mHealth's open architecture (right) [16]	9
2.7	Ohmage platform architecture [17]	10
2.8	Illustration of the health information cycle [18]	11
3.1	The SCRUM process [24]	16
3.2	Ohmage-based system: pmSys version 1	16
3.3	New version of pmSys: Abstract architecture	18
4.1	Abstract Architecture: pmSys back-end early phase	19
4.2	Example of a HTTP-request to the Ohmage back-end	21
4.3	Example response from Ohmage back-end in JSON-format	21
4.4	Example error response from Ohmage back-end in JSON-format	21
4.5	Example of a XML-configuration for a campaign	24
4.6	Optimal configuration	26
4.7	Used configuration	26
5.1	Login: Ohmage Android and Ohmage MWF	28
5.2	Login: Issues with custom server on Ohmage MWF	28
5.3	Survey: Empty list	29
5.4	Campaign: Add campaign and surveys in Ohmage MWF	29
5.5	Reporting: Presentation of questionnaires in Ohmage MWF	30
5.6	Offline reporting: Queue of reports in Ohmage MWF	31
5.7	Notifications: Local reminders in Ohmage MWF	31
5.8	Feedback: Response history in Ohmage Android	32
5.9	Classic (left) vs Ajax (right) web application model [34]	34
5.10	Classic web synchronous communication [34]	35
5.11	AJAX web asynchronous communication [34]	35
5.12	MVC model	37
5.13	PmSys file structure	38
5.14	AngularJS routing example	39
5.15	AngularUI states example	39
5.16	AngularUI routing with states	39
5.17	Illustration of nested view in pmSys	40
5.18	pmSys: Login module	41
5.19	pmSys: Reporting module	42
5.20	pmSys: Example of questionnaire	42
5.21	Example: Survey response in a JSON-object	43
5.22	Example of a XML-configuration in JSON-format	43
5.23	Visualization: Ohmage response data format	45

5.24	Example: Array of data for plot	45
5.25	Example: String to date parsing	45
5.26	Visualization: Example of graphs in pmSys-app	46
5.27	Notifications: Local notifications	47
5.28	Notifications: Remote push-notification	47
5.29	Illustration of the bypass module in pmSys	48
5.30	Benchmark: Full process from home screen to a completed survey (RPE questionnaire) .	51
5.31	Benchmark: Only the reporting process (RPE questionnaire)	51
5.32	Benchmark: Only reporting process (Injury questionnaire)	51
5.33	Userstudy: Usability	53
5.34	Userstudy: Design	53
5.35	Userstudy: Navigation	54
5.36	Userstudy: Presentation	54
5.37	Userstudy: Rating of pmSys	55
6.1	Ohmage Web: Dashboard	58
6.2	Ohmage Web: Survey response read example	58
6.3	Ohmage Web: Visualization module	59
6.4	Ohmage Web: Total response count graph	60
6.5	Ohmage Web: Single variable graph	60
6.6	Ohmage Web: Multiple variable graph	61
6.7	pmSys-push API endpoints	63
6.8	pmSys-push architecture	64
6.9	PmSys-trainer architecture	66
6.10	The flow of a request through a Node web application [53]	66
6.11	Code example of Express route handler	67
6.12	The flow of a request with Express [53]	67
6.13	Main view: layout.html	67
6.14	Sub view: index.html	68
6.15	Sub view: about.html	68
6.16	Example of Express view templates	68
6.17	pmSys-trainer: Login	69
6.18	pmSys-trainer: Listing of players who have reported	70
6.19	pmSys-trainer: Manual push messaging to selected players	70
6.20	pmSys-trainer: Adding a automated push message	70
6.21	pmSys-trainer: Listing of automated push messages	70
6.22	pmSys-trainer: Survey response read	71
6.23	Statistics of responses	71
6.24	Data format in C3.js	72
6.25	pmSys-trainer: Example of team visualization of RPE	73
6.26	pmSys-trainer: Wellness and injury visualization	73
6.27	Player visualization	74
7.1	Automatic data analysis	79
7.2	Abstract Architecture: pmSys automatic analysis module	79
7.3	Analysis: Strain	80
7.4	Player profile	81

List of Tables

4.1	Ohmage web-APIs which are used in pmSys	20
4.2	Class roles	22
4.3	Campaign roles	23
4.4	Prompt-types used in pmSys	23
4.5	Survey response read: Access rules	24
4.6	Configuration: Users	25
4.7	Configuration: Classes	25
5.1	OS and their programming language	36
5.2	Response statistics	52

Acknowledgement

I want to thank my supervisor Pål Halvorsen who has greatly helped me by providing guidance, ideas, discussion and feedback throughout this thesis. Your knowledge, commitment and positiveness is truly inspiring, and I am grateful for the opportunity to work on this exiting project.

Furthermore, I wish to express my sincere thanks to Thuc Hoang and Kennet Vuong for their cooperation in this long journey. Their commitment and hard working attitude is what made this thesis possible, and I personally want thank the both of you for the support, motivation and encouragements during this thesis.

I also wish to thank the Norwegian School of Sports Sciences, and especially Håvard Wiig and Thor Einar Andersen for giving me the opportunity to combine my thesis with your PhD research. I am thankful to them for sharing their expertise in sports medicine, as well as providing us with input and discussion throughout the whole thesis. I also want to thank everyone that have been testing and using this system and providing us with great feedback.

I take this opportunity to thank Håvard Johansen, Dag Johansen, Svein Arne Pettersen from University of Tromsø, as well as Carsten Grizwodz and Håkon Kvale Stensland from Simula Research Lab for providing input and great insight.

Finally, I would like to thank my family and friends for their support and encouraging words. I am also grateful for my girlfriend, Elin Vyvy Tran who supported me through this venture; you motivated and encouraged me in moments when I needed it.

Oslo, May 18, 2015
Cong Nguyen Nguyen

Chapter 1

Introduction

1.1 Background

In football today, there is an increased focus on fitness and physical prowess. Pure talent in football in terms of abilities like technique, passing, creativity and so on, is an essential part of the game, but the physical aspects of the game are equally important. We will not see a player that is talented, but has poor physical abilities dominate the game today. When we look at the game nowadays, it is important to be able to run fast, be in good shape, and strong at the same time. Cristiano Ronaldo is a prime example of a player that has the physical tools, as well as the necessary skills with the ball in order to succeed. To manifest his success, we can mention that he has won the highest ranked individual award a football player can achieve, the FIFA Player of The Year Award 2014 [1]. In order to acquire those mentioned abilities, it will require many hours of practice, but there is a fine line between practicing a lot and overtraining.

Overtraining appears to be caused by too much high intensity training, combined with little time for the body to recover, and is often the cause for **Overuse Injuries**. An overuse injury is a microtraumatic damage to a bone, muscle or tendon that has been subjected to repetitive stress without sufficient time to heal or undergo the natural reparative process [2]. With the increased amount of training, comes an increased need for coaches to monitor their players because of the risk of gaining an injury from overtraining or overuse. The **Norwegian Top Division (Tippeliga)** teams, have previously monitored their team by using pen and paper. This format is very tedious, and it becomes a tough task to keep track of all the players over a longer period. Going through hundreds of reports manually for over twenty players is a daunting task for most coaches. In recent years, some teams have started to use digitalized survey systems to make it easier for the players to give their daily reports to the coaches. The system collects and stores the data in a centralized server, and the data is easily accessible for the coach at any time. The digitalized solutions are a much better option then pen and paper from a practical perspective, but the existing systems are not tailor-made for football teams.

Technology today is advanced enough that we can introduce informatics to enhance these reporting systems to create a complete monitoring system. The majority of the Norwegian population have access to ICT-equipment (Information- and Communication technology) [3], making a digitalized reporting system viable. A system that replaces pen and paper with mobile devices will make it easier for the players to submit reports. With a dedicated and tailor-made system towards football teams, both coaches and players will participate in the monitoring process and benefit from it. With the right tools and features, this system can help the coaches supervise the activities of their players to keep them fresh and healthy for a whole season, avoiding overtraining and the eventually overuse injuries.

Norwegian School of Sports Sciences (NiH) is conducting a PhD study to investigate the usefulness of objective and subjective tools for monitoring training load and fatigue in professional football [4]. Their ambition is to provide better recovery and periodization strategies to enhance physical performance, avoid overtraining, and prevent overuse injuries. We have directly collaborated with NiH throughout this master thesis, with the objective of providing them with the digitalized system needed to monitor the training load and fatigue through self-report surveys.

1.2 Problem Definition

Monitoring is defined as "observing and checking the progress or quality of something over a period of time" [5]. *In this thesis, we investigate the question on how to develop and design a digitalized monitoring system that includes collecting, storing, presenting and analysis of data. How can we create a tailor-made system to assist football teams in monitoring the training load and fatigue of their players by providing meaningful information?*

We will research and develop a **Player Monitoring System (pmSys)** for football teams, in order to provide them with meaningful information, which will allow them to observe the progress of the players. The main objective for pmSys is to ease the monitoring process for both coaches and players. For the players, we create a new reporting tool that is less tedious than pen and paper. We also address deficiencies that existing digital reporting systems has, in order to improve them in pmSys. Particular focus will be on optimizing the reporting process, as well as include motivational features to encourage players to participate in the monitoring.

Our focus for the coaches is to provide an interface that is helpful for the purpose of monitoring players. The system must therefore provide the coaches with features to monitor, but also help the coaches to assist and supervise their players. The interface is vital for the system, as it presents the coach with an overall understanding of the players fitness, thus providing the coach the opportunity to adjust the training regime to fit the players better.

Lastly, we evaluate pmSys in terms of performance and usability. Evaluation of the performance will be performed through various benchmarks, while the evaluation of usability will be done through an end-user feedback. The results from the evaluation will give us an indication on the usefulness of the new system.

1.3 Limitations

PmSys is a system that is accessible for all athletes, but for this thesis, the focus group is limited to football players. We had the opportunity to collaborate with both the Norwegian National Football Team and several Tippeliga teams. In return, this meant that we could develop and test our system towards actual users, and gain feedback from professional players that would use it every day.

The system is using an existing back-end solution, Ohmage back-end. We will not go in depth on the Ohmage system, but we will only explain why and how we have used it. Ohmage is a big open source project and is used worldwide. Therefore, by going in depth on this system itself would be a thesis alone. In this thesis, we point out the flaws that the Ohmage framework has as a whole when used for the purpose of monitoring football players. We can then focus on how we can improve these flaws, and make a more complete system for our purpose. The security and privacy challenges of developing this system are not highlighted in this thesis. We will briefly explain how some of the issues are solved, but we will not go in depth. We assume that the existing Ohmage back-end will provide our system with the necessary security measures that are needed.

Furthermore, pmSys is currently limited to self-report subjective data. The system has yet to implement third-party devices that can passively collect objective data like heart rate, number of kilometers ran, hours of sleep and so on. We believe this part of the system has a lot of potential, and it is in fact a part of Thuc Hoang's master thesis. However, for this particular thesis, this area has not been implemented yet.

1.4 Research Method

In this thesis, we follow the design methodology described by ACM Task Force in *Computing as a Discipline* [6]. This method involves the design, implementation and evaluation of ways to develop the system. The new system has been used on real football teams and players, and it has been in use for several months. This has allowed us to make improvements and changes according to the feedback from the users, and we were able to fix problems we encountered upon deployment. By using it in a real scenario, we were allowed to verify, to a greater degree, the effectiveness of our solutions, and how well

it has replaced the previous solutions of reporting, but also how well the system functions as a monitoring system.

1.5 Main Contributions

The main contributions of pmSys are the deployment of the system for several Tippeliga teams. This includes following teams: Rosenborg Ballklubb, Strømsgodset Idrettslag, Viking Fotballklubb, Sarpsborg 08 Fotballforening and Tromsø Idrettslag. All the teams have participated in NiH's PhD study, and have used it during the 2013/2014 season of the Tippeliga. Some teams have agreed to participate in the 2014/2015 season as well. The **G16 Lørenskog Idrettsforening (LIF)** football team is also using pmSys. They are our main test group, because we have been able to have direct contact with the players. With them, we have gained direct feedback, instead of going through a third-party.

We have also had the opportunity to deploy pmSys to the Norwegian National Team. The players of the national team have used pmSys when they have gathered to practice and play matches for Norway. Through our collaboration with the National Team, the system has gotten some media attention as well. The system, specifically the application was described as an application that will secure success in the UEFA-Euro tournament [7] for Norway.

The deployment of pmSys consists of a tailor-made mobile application for reporting and a web-portal for the coaching staff. The mobile application supports two of the major platforms for mobile devices: iOS and Android. We have also managed to deploy the application to the app-stores on both platforms, making it a highly available application. For players that do not have mobile devices with a supported platform, we have released a web solution that works on any modern web browser. The application has been downloaded a total of 195 times since it was released (last checked: 8th May 2015). The web-portal for the coaches have also been developed and rolled out in the later stages of pmSys. The web-portal features several tools to assist the team in monitoring their players, such as: team visualizations, player visualization, push-messaging, automatic analysis and statistics. We have also developed a standalone push-server that supports push-messages to both iOS and Android mobile devices.

Through the development of pmSys, we have managed to create a monitoring system with high usability for all the users. Compared to previous used systems, we have greatly increased the activity of the players. The first four months after the release of pmSys, we registered over 2500 reports, compared to the 500 registered reports for the whole season last year. By having a system that reminds the players to report, as well as motivational features and an optimized user interface for reporting, we have created an application that are better suited than the old reporting tools. The web-portal is also been seen being used a lot more frequently than the existing analysis application. We have also implemented automatic analysis, that gives the coaches useful, but easy to understand feedback of the players.

1.6 Outline

We begin this thesis by giving a presentation of related background and work that has directly influenced the development of pmSys in Chapter 2. Secondly, we introduce pmSys in Chapter 3, and give an in-depth requirement analysis of the proposed system. The explanation on the purpose of the system, and the requirement set for the development process is presented. In Chapter 4, we introduce the Ohmage back-end, which is the main building block of pmSys. We present how we have configured the back-end to match the requirements we have set. Chapter 5 will be about the pmSys mobile application. This chapter will cover the whole development process of the application, from an idea to a finished prototype that is now available in Apple's App Store and Google Play. In Chapter 6, we present the pmSys-trainer web-portal. We will look at how this web-portal works in conjunction with the pmSys mobile application to give the coaches the necessary information about the players. In both Chapters 5 and 6, a comparison between Ohmage and PmSys will be done, followed by an evaluation of pmSys as a monitoring system. Chapter 7 is a specialization chapter where we will research the opportunities on how the system can perform automatic analyzes of the collected data. Finally, in Chapter 8, we conclude the thesis by summarizing our work and findings, as well as discussing future work for the system.

Chapter 2

Background and Related Work

In this chapter, we will present the sports related background, which is directly connected to the data we are collecting in pmSys. Secondly, we will look into systems and architectures that revolve around the collecting data. Lastly, we give a brief explanation as for why they do not fit as a monitoring system, but also how they have influenced the implementation of the new system.

2.1 Sports related background

In this section, we will give a brief introduction about the data we are collecting in our system. In this thesis we work from an informatics point of view and explain and illustrate the development of a digitalized monitoring system. However, from a sports perspective, we do not have the knowledge on what data we should collect, nor do we know what parameters to set in order to collect relevant data to monitor football players. In order to create pmSys, we have collaborated with NiH, as mentioned in the introduction (See Section 1.1). Through their PhD [4] about monitoring load and fatigue, they have defined a set of surveys that pmSys is using to collect relevant data.

2.1.1 Rating of Perceived Exertion

The **Rating of Perceived Exertion (RPE)** survey measures the player's perceived intensity of a training session. The data collected can be used by researchers in medical studies, but also as a description of the intensity of the training sessions for the training staff. The RPE scale (See Figure 2.1) rates the perceived degree of exertion, and it is done through a self-analysis of the physical impact on the player during the workout. This report is a subjective measurement and the players will give a single number representing the entire training sessions intensity [8].

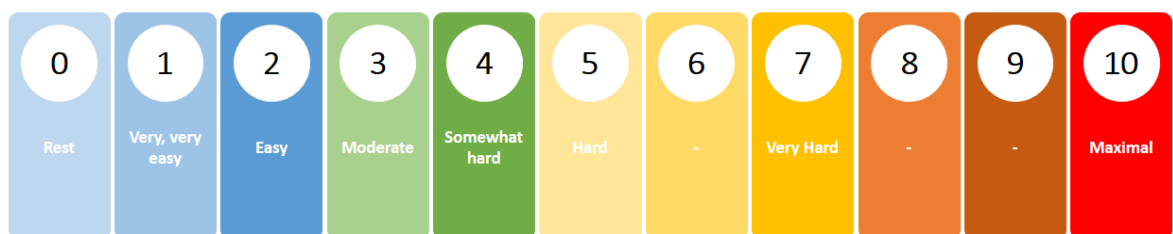


Figure 2.1: RPE scale [9]

The report is done directly after the training session, and the player subjectively rates the level of exertion. The level of exertion is often multiplied with the duration of the session to quantify the training load on the player. The coaches can also register their expected exertion to compare their expected values to their player's experienced exertion.

2.1.2 Wellness

In the survey about wellness, the players record a self-assessment of their well-being. The survey gives the team's staff an indication of how well each player is feeling, based on the following parameters: fatigue, stress, sleep quality, mood, and muscle soreness. This assessment is done once a day in the morning, and the player rates their well-being on a scale from 1-5 (See Figure 2.2).

	5	4	3	2	1	Record Score
FATIGUE	Very fresh	Fresh	Normal	More tired than normal	Always tired	
SLEEP QUALITY	Very restful	Good	Difficulty falling asleep	Restless sleep	Insomnia	
GENERAL MUSCLE SORENESS	Feeling great	Feeling good	Normal	Increase in soreness/tightness	Very sore	
STRESS LEVELS	Very relaxed	Relaxed	Normal	Feeling stressed	Highly stressed	
MOOD	Very positive mood	A generally good mood	Less interested in others &/or activities than usual	Snappiness at teammates, family and co-workers	Highly annoyed/irritable/down	

Figure 2.2: Wellness questionnaire [10]

Through this survey, we can discover that inappropriate training loads after a match can negatively impact the player in terms of fatigue and muscle soreness. This survey is a simple and cost-effective monitoring tool for the coaches [10], and can help them adapt their training sessions to avoid overtraining and fatigue on their players.

2.1.3 Injury

In this survey, the players report any eventual health problems they have experienced during the past week. The survey has four main questions that are based on the Oslo Sports Trauma Research Center ¹ questionnaire on health problems (See Figure 2.3) [11].

Question 1

Have you had any difficulties participating in normal training and competition due to injury, illness or other health problems during the past week?

- ☐ Full participation without health problems
- ☐ Full participation, but with injury/illness
- ☐ Reduced participation due to injury/illness
- ☐ Cannot participate due to injury/illness

Question 2

To what extent have you reduced your training volume due to injury, illness or other health problems during the past week?

- ☐ No reduction
- ☐ To a minor extent
- ☐ To a moderate extent
- ☐ To a major extent
- ☐ Cannot participate at all

Question 3

To what extent has injury, illness or other health problems affected your performance during the past week?

- ☐ No effect
- ☐ To a minor extent
- ☐ To a moderate extent
- ☐ To a major extent
- ☐ Cannot participate at all

Question 4

To what extent have you experienced symptoms/health complaints during the past week?

- ☐ No symptoms/health complaints
- ☐ To a mild extent
- ☐ To a moderate extent
- ☐ To a severe extent

Figure 2.3: Injury questionnaire: key questions [11]

¹www.ostrc.no

Depending on the response from the questionnaire, the player is given a severity score of their injury or illness. Question 1 and 4 with four alternatives can give 0, 8, 17, 25 points depending on the severity of the injury. Question 2-3 with five alternatives can give 0, 6, 13, 19 or 25 points. By adding up the score of each question, the player will be given a severity score between 0-100 that will determine how severe the injury or illness is [11], where 100 is very severe. When the player is done answering the four key questions, there are follow-up questions to obtain further details of their injury/illness problem (See Figure 2.4). This survey is used to gather information about injuries and illnesses, but it can also be used in conjunction with RPE and Wellness measurements to have a better change of finding the cause of the injury.

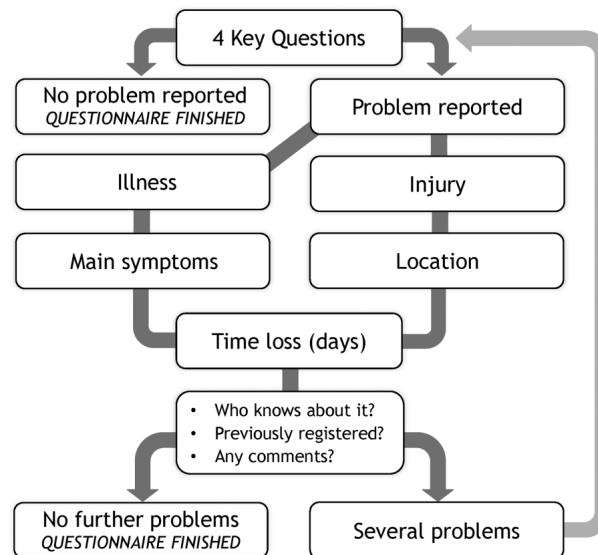


Figure 2.4: Injury questionnaire: follow-up questions [11]

2.2 Participatory Sensing

Participatory Sensing (PS) is a distributed data collection approach that many existing systems are following. Over the years, the number of people carrying mobile phones has increased all over the world. In 2014, 81 percent of the Norwegian population owned a smartphone [12]. These devices are increasingly capable of capturing, classifying and transmitting several types of data, either interactively or autonomously. PS will use the deployed mobile phones to form interactive, participatory sensor networks that enable users to gather, analyze and share local knowledge. Built-in microphones and cameras can record environmental data. Cell-tower localization and GPS can provide location and time synchronized data. Wireless radios and internal processing power enables human interaction with both local data and remote servers.

An architecture for PS, as shown in Figure 2.5, can enhance and systematize data collection by increasing the quantity, quality and credibility of the gathered data. Increased data quantity is achieved by having mobile applications on the smart phones, which can increase the reach and coverage of the system. Enabling users to be able to transmit data real-time, before the environment changes, can increase data quality. By tagging data points with location and time through localization services on smartphones, we can increase the credibility of the information. PS exists in several fields like health, where it can assist individuals and their care providers in monitoring their health. For emergency situations, it can be used by citizens or emergency response teams to rapidly gather information about any given situation. In education, it can be used as an instructional tool for teaching about data collection [14].

For pmSys, it is a approach that we will incorporate in our system. Collecting data and submitting reports through mobile devices not only makes it easier for the football players, but also increases the quality, quantity and credibility of the collected data.

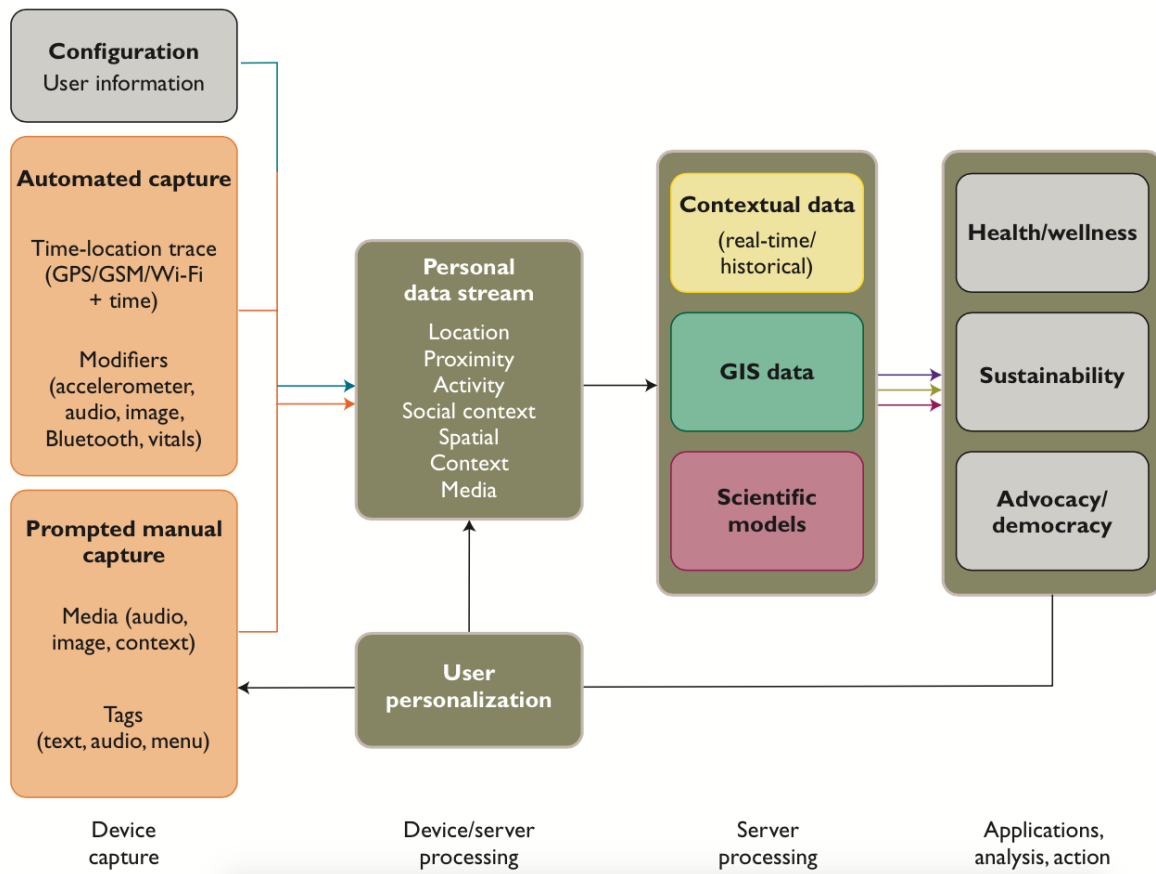


Figure 2.5: Common architecture for PS applications [13]

2.3 Open mHealth

Mobile Health (mHealth) is a practice of medicine and public health supported by mobile devices. It follows the PS approach introduced in the previous section (See Section 2.2) for collecting data, but the architecture is tailored towards public health care. Chronic diseases, like heart disease, stroke, cancer, chronic respiratory diseases and diabetes represent 60 percent of deaths across the world [15]. The traditional model of episodic care in clinics and hospitals are suboptimal for improving these disease outcomes. By presenting mobile communication devices in conjunction with internet and social media, it will present opportunities to enhance disease prevention and management by extending the reach and coverage of traditional care. This approach is known as mHealth [16].

mHealth makes it possible for patients to collect and share relevant data at any time, and not just when they visit a clinic or the hospital. This will allow for more rapid convergence to optimal treatment. The challenge for mHealth is the amount of closed application with its own data format, management and analysis. This approach is known as "stovepipe" or "siloeed" (See Figure 2.6), and limits the potential mHealth has.

Open mHealth revolves around building an architecture that has shared data standards. An open architecture (See Figure 2.6) will fully realize mHealth's potential because the process of collecting, storing and presenting the data is following a standard. The standards can be implemented in systems for clinics and hospitals, enabling them to access all types of data, instead of being limited to certain closed systems.

The main objective for this approach is to make an open platform in order to encourage code reuse, but to also have a standardization of data. This approach is something pmSys has been heavily influenced by, and we are actually building our system as a new layer on top of an existing and extensible PS-platform.

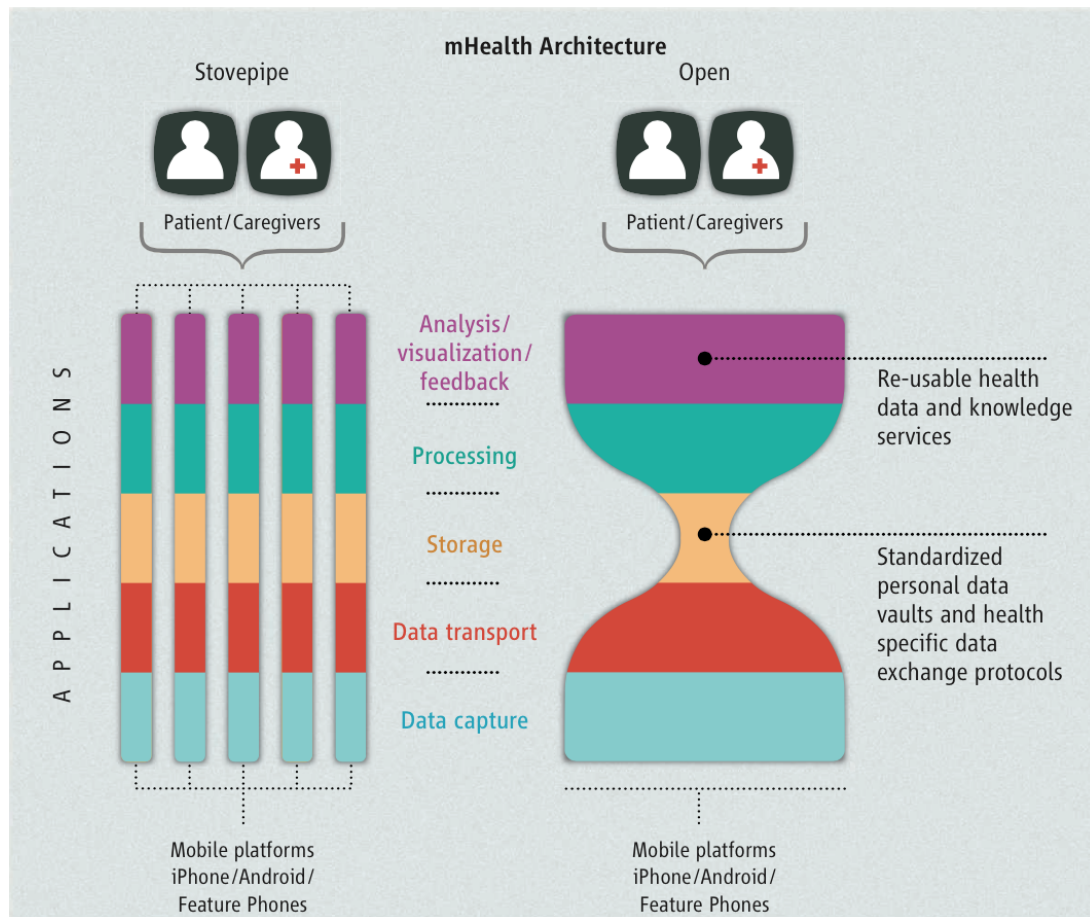


Figure 2.6: Stovepipe architecture (left) and Open mHealth's open architecture (right) [16]

2.4 Ohmage

Ohmage is a concrete example of the approaches presented in PS and Open mHealth. Ohmage is a modular and extensible open-source PS platform that collects, stores, analyzes and presents data from both prompted self-report and continuous data streams. Ohmage is used as a general, reusable and extensible platform to help design and iterate on PS deployments. Instead of having many "siloed" PS-applications, Ohmage offers an effective general PS platform for collection, management, analysis and visualization of a diverse types of data. The "siloed"-approach discouraged code reuse, caused high development and maintenance overhead and did not scale well [17]. The mentioned drawbacks led to the development of an open platform, Ohmage.

As a general PS system, Ohmage was actually specifically designed to allow easy configuration of systematic data collections in which individuals captured self-report and passive data using smartphones. This is done through the various applications from the Ohmage platform (See Figure 2.7), like Ohmage MWF. As a reusable and extensible platform, various mobile sensing applications have already been integrated with Ohmage to automatically collect data. It achieves its modularity and extensibility through a rich set of backward-compatible **Application Programming Interface (API)**, which provides unified data access across all the applications that have been built on the Ohmage back-end.

In addition to the data collecting applications, Ohmage also has web-based administration tools and several features for analysis and visualization of data. All of these mentioned applications are built on the Ohmage back-end through the extensible web-API. Because Ohmage has an open architecture, and with the flexibility the web-API provides, it allows for components and application to be built independently. An Ohmage-based application with different purposes can be implemented in parallel, but still follow the shared generic standards that Ohmage has defined. By using Ohmage as a building block to develop PS applications, it will save the developers a lot of time and resources, which will allow them to focus on other objectives instead of the technology and other challenges in PS development. As a PS

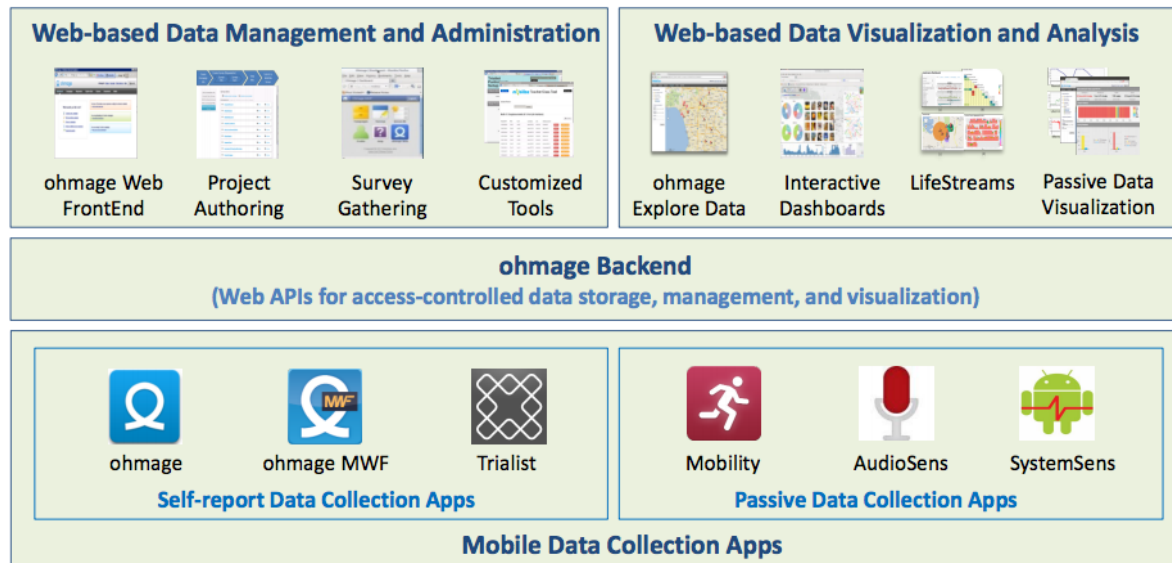


Figure 2.7: Ohmage platform architecture [17]

platform, Ohmage offers all the features and tools necessary to collect data, and with the modularity and extensibility it offers through the API, we can build tailor-made PS-applications for monitoring football players.

2.5 DHIS 2

District Health Information Software (DHIS), version 2² is an open source tool for collection, validation, analysis and presentation of aggregated statistical data, tailored for health information. The purpose of DHIS2 is to provide a comprehensive **Health Information Software (HIS)** solution for the reporting and analysis needs of health information at any level.

A HIS can be better described through the information cycle described Figure 2.8. The cycle shows different components, stages and processes where the data is collected, checked for quality, processed, analyzed and used. DHIS2 main objective is to have a digital framework that supports all the stages of the information cycle, which includes [18]:

- Collecting data
- Running quality checks
- Data access at multiple levels
- Reporting
- Analysis
- Enable comparison of the data across time and space
- See trends

As a PS-application, DHIS 2 supports data collection through mobile devices and is tailor-made for poor countries that have limited cellular capacity. It has a Java mobile client that supports submission of forms using GRPS or SMS, and is the most common DHIS Mobile client. In areas where the mobile data capacity is good, they have a more advanced web browser based client. This client has support for a greater amount of platforms, because it only requires a web browser. The challenge with the web-based

²www.dhis2.org

The Information Cycle

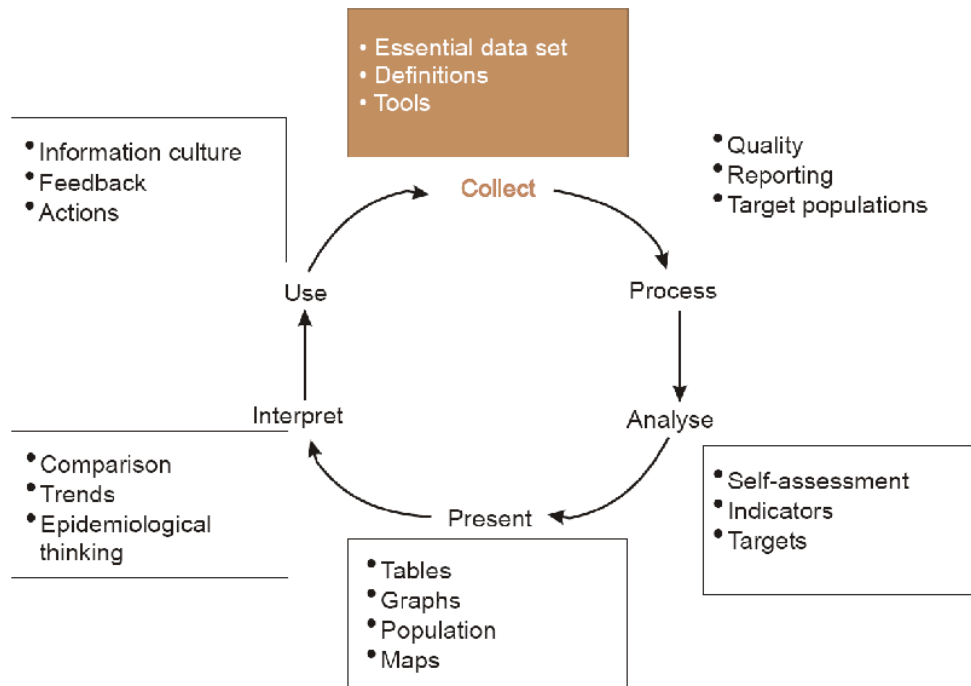


Figure 2.8: Illustration of the health information cycle [18]

solution is that it requires newer and more advanced phones, which can be hard to acquire in poorer countries. Lastly, DHIS 2 also supports regular SMS. The features are limited compared to the other solutions, but it is the most widely available technology on mobile phones and can be used in most circumstances.

DHIS 2 is a specialized data collecting system for health. It aims for usage in poorer countries where they cannot afford more advanced smartphones. This will lead to less sophisticated mobile application, and fewer features that are needed for pmSys. DHIS 2 also offers a web-API that is very similar to the Ohmage web-API, providing access to the DHIS 2 data model. With the API, it is possible for developers to create web applications based on the DHIS 2 back-end, and integrate it to the DHIS 2 main application. The drawback is that the custom-build applications cannot be developed as standalone applications, and can only be developed as an extension in a closed environment within the DHIS 2. With the restrictions, it will not allow us to develop a system that includes all the features that are needed for pmSys.

2.6 Milan Lab

MilanLab is the high tech interdisciplinary scientific research center of the football club AC Milan, and was founded by Jean-Pierre Meersseman together with Bruno Demichelis. It has been operating since 2002 and aims to provide the best possible management of the individual's well-being and health. Professional football players endure a lot of physical stress over the course of the football season, both in the intense competitive phase, and in the training and recovery phases as well. The main mission of MilanLab is to optimize the results of the team through a system of information technology. This in return makes it possible to examine the physical status of a player at any given moment. By monitoring the players, it makes it possible to see what physical effects have on the body, which flags up the actions that will be required to ensure maximum performance [19].

Several top-level players have praised the concepts introduced in MilanLab. Proving that through monitoring and right training, it is possible to avoid injuries and get the player in the best form possible. Under MilanLab supervision, several stars have managed to play in Serie A even after passing the age of 35. David Beckham, Clarence Seedorf, Paolo Maldini and Gennaro Gattuso are some well-known

players that have played at a high level despite their age [20].

2.7 Summary

In this chapter, we have presented sports related background information, which is directly related to the data pmSys will collect and use for monitoring. The data is the foundation for our work in reaching the main goal of being able to monitor the players. We have also introduced architectures and existing systems that have influenced our system. PmSys is following the PS-approach, using mobile devices to collect data from the players. We are also incorporated the open architecture approach presented by Open mHealth, by building our system on an open extensible platform, Ohmage. We have also drawn the conclusion that the existing systems with their included tools and applications are either too generic (Ohmage), or too specialized in their field (DHIS 2). The other issue with DHIS 2, is the closed environment, that will not allow us to develop a system with all the necessary tools and features.

MilanLabs have shown that through monitoring it is possible to create a system that can help teams keep their players fresh and healthy. Players that were way past their peak, could still play at a high level. With pmSys, we do not expect to achieve the same results as AC Milan, but if we can recreate some of the elements to assist football teams monitor their players to achieve some of the results over time, that will be an achievement for us.

In the next chapter (Chapter 3), we will explain the requirements and architecture of pmSys. We will also present how we have taken elements of the related architectures and existing systems into our design and development process. Particularly focus will be on how we are going to build an Ohmage-based system.

Chapter 3

PmSys

In this chapter, we outline the requirements of pmSys. The requirement analysis has been done in close collaboration with NiH. First, we give an overview of the capabilities of the system, and how related architectures is being used for pmSys. We will then present the functional and non-functional requirements for pmSys based on the problem definition in Section 1.2. The first version pmSys was based on Ohmage platform, including the applications that follow with it. We will briefly present the applications and explain how they were used.

3.1 Overview

The surveys defined in Section 2.1, will be implemented in the system and presented as questionnaires for the players in order to collect relevant data. PmSys will therefore be a system for collecting, storing, analyzing and presenting data that are based on the questionnaires. The system will be an Ohmage-based system comprised of three main components that each serves as an essential part of pmSys. The first component is based on the PS approach introduced in Section 2.2, that includes the usage of mobile phones to collect data. This approach will enhance the quality, quantity and credibility of the data. This aspect of the system is very important because the questionnaires require the participants to submit the reports at the right time, but also over a longer period [8]. With mobile devices, the player can submit reports anytime as long as they have access to the device, thus increasing the reach of the system and the quantity of data. The quality of data is ensured by enabling the player to answer the questionnaires real-time. Credibility is obtained by being able to timestamp the reports, but also link a report to the specific player that submitted it. This component will be a mobile application that will be installed on the mobile devices. The second component of the system is the back-end that will store the submitted reports from the players through their mobile devices. The last main component is the dedicated tool for the coaches, where the reports are presented in form of analysis and graphics to ease the process of monitoring the players. This tool will be developed as a responsive and an interactive web-application.

The majority of the users will mostly be people without technical background. With varying degree of technical knowledge, all the features in pmSys have to be easy to use. That means that the **User Interface (UI)** must be clear and simple without unnecessary distractions. The system must also include features and tools to motivate and encourage the players to report.

3.2 Functional requirements

3.2.1 Collect data

The collected data will be in form of responses from the players based on the implemented questionnaires. This process will require an interface that will present the questions for the players, and a back-end solution with a database to store the submitted reports. For this process, we will design and develop a mobile application. The mobile application must be developed with the focus on optimizing the reporting process by minimizing the steps needed to complete a questionnaire. The system will

require the player to answer submit reports every day, and with a cumbersome UI it can become a time-consuming process that may discourage some players.

3.2.2 Present data

The presentation of the data from the reports of the players is crucial for a monitor system. How the data is presented will have a big effect on the usefulness of the system. This will be valid for both the coach and the players. The raw data must be processed and presented in a way that will allow the users to do as little manual work on the data as possible, and at the same time be useful. The presentation of the data is also directly related to the process of collecting data. The system must have features that will motivate the players to participate in the monitoring process over the course of a long season. The presentation of the data for players in form of visualizations to give them feedback on their development can give them a sense of being more involved. This can be a motivational factor that will encourage them to keep reporting, as they feel they get something in return for their participation. The importance here is to limit what to present. If the system does not present anything, most users will be discouraged to use the system. However, if the system presents too much unnecessary information, they might find it cumbersome and confusing.

3.2.3 Analysis of data

Analysis of the data must be presented to the coach in a way that will allow the coach to monitor the players. The importance here is to highlight the important data from the questionnaires, because that can help the coach make the proper decision in regards to the players training regimen. The analysis of the data can be in form of text or visualizations. Visualizations of the data over a certain period will show the coaches how the players are progressing or regressing. The visualization should contain data from the whole team to see how the team performs as a whole, but the system must also allow for a more detailed inspection of the individual players as well. An important aspect is the ability to compare the reports from different questionnaires up against each other. A viable scenario is that the coach wants to analyze what effect a tough training session has on the players wellness the day after, and based on the analysis he can increase or reduce the intensity of the next training session. This is one of the many analyses the coach can make from the presented data. Therefore, in order for the coach to take advantage of the collected data, it has to be presented in a way that makes sense for them from a football perspective.

3.2.4 Tools for assistance

It is very important to keep a high level of quality on the data, but the quantity is equally important in order to get the correct analysis of the players. If we take a real scenario - for example if the player forgets to report a training session right after it is completed. The perceived exertion might be inaccurate because it is reported late, and the player has been able to rest, compared to immediately after the session. Another plausible scenario is when a player forgets to report a tough training session. When the player then reports that he is very sore, that is a abnormal value that cannot be explained. The system will present that there has not been a session, but the player is still very sore. Both scenarios can lead to abnormal values that can be hard to understand for the coach. To avoid the unwanted scenarios the system must provide the coaches and the players with assistance. For the coaches, they can get a automatically generated list with the players who have made a report, and those who have not. They must also have a communication channel in combination with this list to help remind the player to make a report. The system must provide the players with local reminders on the smartphones, but also remote notifications that can be sent from either a coach or an automaton.

3.2.5 Cross-platform support

With the diversity of different platforms available for the users, the system must support the majority of them, and at least the most popular of them. For mobile devices, we have several platforms, but the most used is iOS and Android. PmSys will aim to support those two in the first release, with the opportunity

to expand to more platforms in future work. The dedicated web-portal for the coaches will support all platforms that have access to a modern web-browser. The importance of having cross-platform support is to be able to reach most of the actual users.

3.3 Non-functional requirements

PmSys will also have non-functional requirements that have to be taken into account. The following sections outlines the main non-functional requirements set for pmSys.

3.3.1 Usability

As mentioned in Section 3.1, the varying degree of technical competence requires the system to be user-friendly. Most of the football players have non-technical backgrounds. This will require the system to have an intuitive layout, simple interactions that will result in a system with high usability. The challenge here is to make a system with optimized solutions, but still maintain high usability.

3.3.2 Scalability

The back-end of the system must support concurrent requests from many users that wants to retrieve, update and send data to the server. The storage systems must also be scalable in order to store the data that will be sent every day.

3.3.3 Availability

The system must be available for the players at any time in order to ensure data quality and quantity. It must have a solution to allow the player to report even without internet. The system must also have a solution for players who are not running a supported platform on their mobile device.

3.3.4 Privacy and Security

Privacy and security is important in this system. Since pmSys collects personal data from the players, it is very important to secure and protect the privacy of the players. Ministry of Justice and Public Security in Norway has strict rules for processing of personal data, and everyone that wants to process this type of data have to apply and follow a plethora of rules [21]. In pmSys, all the players are anonymous, and the data stored in the system cannot be identified to an individual. Only persons that have applied and have been granted permissions from the Ministry of Justice and Public Security can identify the individuals.

3.3.5 Performance

Performance is an important aspect for the usability. A slow UI can have a negative effect on the user experience. Every module of the system must have good performance, and be fast and responsive for the users. Teams expect the system to work well out of the box.

3.4 Development methodology

For the development process of pmSys, we have followed elements of the agile development practice SCRUM. In SCRUM, we split the project up in smaller modules called user stories. These stories combined, is called the product backlog, which are features that need to be implemented. This is a part of something called the "pregame", where we plan and design the features we want to include in the system. We then have to prioritize which stories we want to include for the next release of the system. The stories included here are called the release backlog. Then we have to estimate how long each story will take to implement, in order to set a release date. When all the planning is completed, we enter something called "game", which consists of sprints. Sprints are short-duration milestones, which allow us to tackle a manageable amount of work, and get it ready for release. The sprints consist of

development, reviewing and adjusting the product. The last part is called "closure", where we prepare for the release of a version of the product. We will run tests and then deploy it [22]. The explained process is also illustrated in Figure 3.1. To have a shared interface to place the stories and the backlog, we have used Trello, that is a online web-based planning board. It features a friendly UI where we can drag and drop the stories to the appropriate state [23].

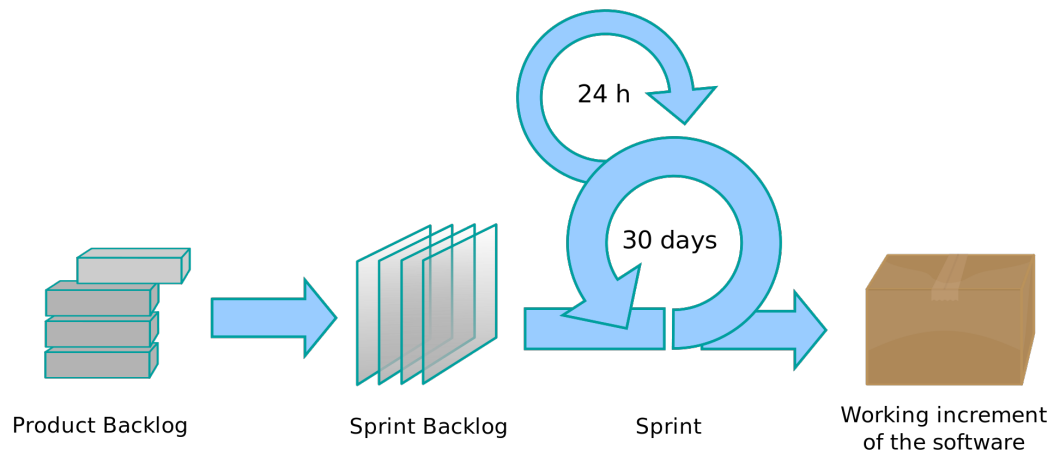


Figure 3.1: The SCRUM process [24]

3.5 PmSys: version 1

In the early stages of pmSys, we used the Ohmage platform and the applications that it offered for collecting, storing and presenting data. As a data collecting system, Ohmage is great and offers many features to get going. It is also a system that has been used by **Tromsø Idrettslag (TIL)** in collaboration with **University of Tromsø (UiT)** to collect data from their players, and later on it was used in the first version of pmSys as well. In the next sections, we will present the platform together with the applications used in the early days of pmSys in collaboration with NiH. In Figure 3.2 we have highlighted the used applications from the Ohmage platform.

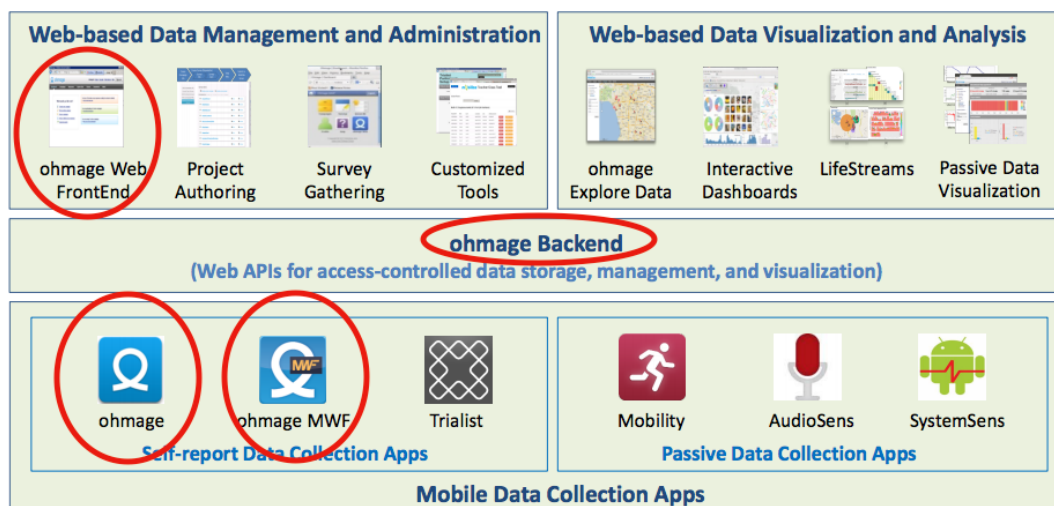


Figure 3.2: Ohmage-based system: pmSys version 1

3.5.1 Ohmage mobile applications

Ohmage has two mobile applications for collecting self-reported data. The Ohmage **Mobile Web Framework (MWF)** is a cross-platform mobile application and is currently available for both Android and iOS users. Self-reported data refers to data that is captured by a prompted experience sampling method where the participants are asked to record the experiences and observations in the form of surveys [25]. Ohmage MWF supports a rich set of prompt types, notably single or multiple choice questions where the user can send in response in several types: number, free-text, time-stamp and many more. The Ohmage MWF application is developed with HTML5, JavaScript and the UCLA-developed MWF technologies ¹ for dynamic and responsive page rendering and data cache for offline operations. PhoneGap library ² is used as the middleware to access the native features on the device such as GPS, camera, etc. While the iOS version of the Ohmage MWF has been updated with the latest features, the Android version has not. For this reason, the Android version of the Ohmage MWF was not used.

Fortunately, Ohmage released another application for Android users, a native application also called Ohmage. The Ohmage Android application has all the features that are included in the MWF application. It is also up to date, and supports the addition of mobility. Mobility is a application for capturing passive data [17], but this application has been taken off Google Play, so this feature is no longer available.

3.5.2 Ohmage Web FrontEnd

The Ohmage Web FrontEnd is the administration tool for the Ohmage system. It is the main project, data and user management portal for the Ohmage back-end. System administrators can use the front-end to create and manage user accounts and groups, configure surveys and inspect the collected data. Other users who are privileged can use it as an analysis tool to monitor incoming data reported through the mobile applications. A restricted user, can also use the front-end, but is restricted to their own data. This portal was used both as a administration management tool, but also as an analysis tool to export and view response data.

3.5.3 Ohmage back-end

The Ohmage back-end is the backbone of an Ohmage-based system. It provides secure communication, authentication, account management, access control, data storage and management. All of these features are reached through a rich extensible API. These APIs makes it able to perform operations like **create, read, update and delete (CRUD)** to all the Ohmage objects on the system. All the CRUD operations are available for all applications through an web APIs, that will allow applications to do CRUD operations directly to the back-end. With the web-APIs, any mobile or web application that has the intent of collection data, can be created. Both the Ohmage mobile application and the Ohmage Web FrontEnd are built on the Ohmage back-end.

3.6 Summary

In this chapter, we have presented pmSys. We have carried out a requirement analysis for what is required for the player monitoring system, and discovered a set of functional and non-functional requirements. We have also presented the first version of pmSys, which revolved around the usage of the Ohmage framework. Through the usage of the Ohmage applications, we discovered several deficiencies that did not match our requirements for pmSys. The deficiencies are mainly on the front-end solutions: Ohmage MWF, Ohmage Android and Ohmage Web FrontEnd. The experienced drawbacks led to the development of new applications, pmSys mobile application (pmSys-app) and the pmSys web-portal (pmSys-trainer).

As presented in Section 2.4, Ohmage is a modular, extensible PS framework, which can serve as a great building block for new systems that revolves around collecting data with mobile devices. In this thesis, we have chosen to use the Ohmage back-end, since it features all the necessary solutions on the back-end for us to build upon. With the Web-API the Ohmage back-end provides, we have developed

¹<http://mwf.ucla.edu/index.php>

²<http://phonegap.com/>

new front-end applications that are better suited towards football players. In Figure 3.3 we present the abstract architecture for the new pmSys system.

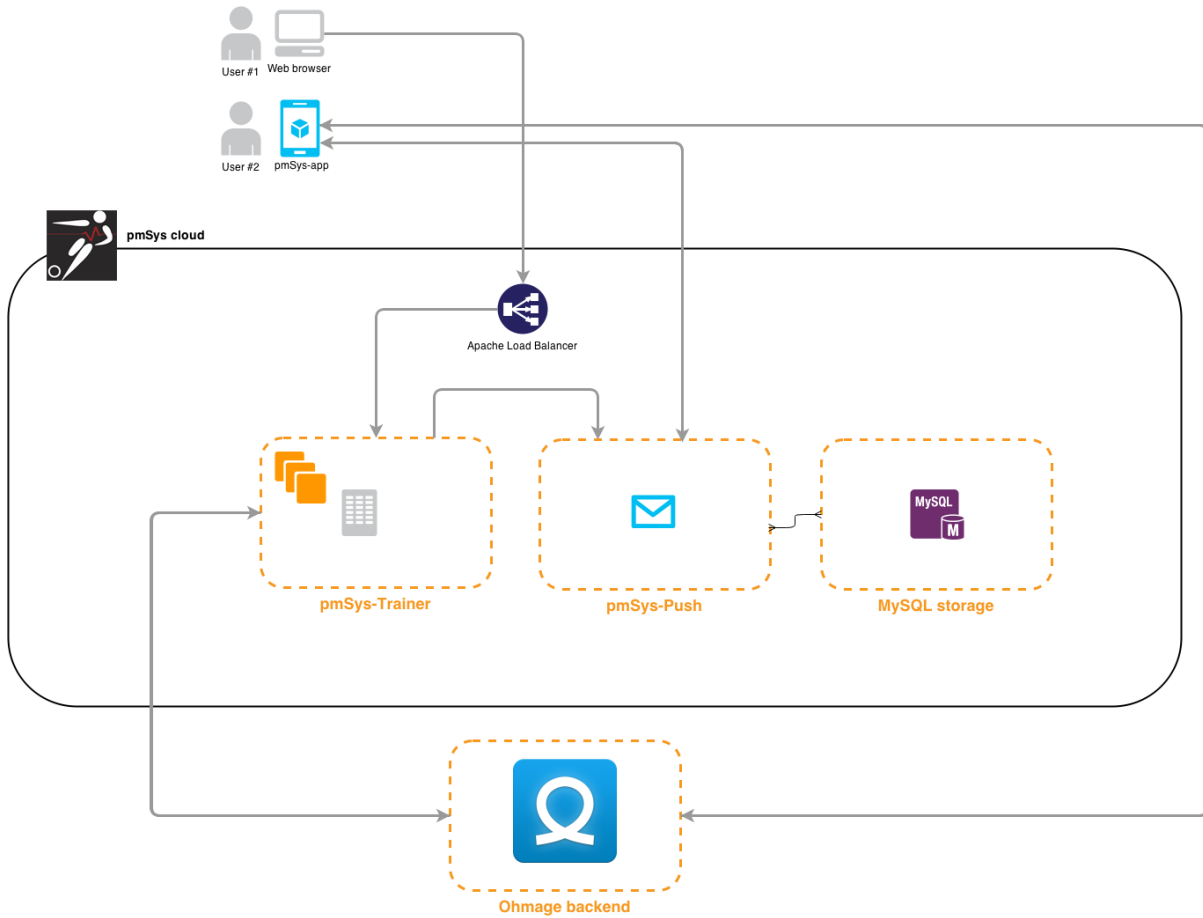


Figure 3.3: New version of pmSys: Abstract architecture

In the next few chapters, we will go in depth on how we have developed an Ohmage-based system, which eventually became pmSys. In Chapter 4, we present how we have configured the Ohmage backend for pmSys. Following that chapter, in Chapter 5, we will introduce the pmSys-app by presenting the development process, and all the design and implementation choices we have made. In Chapter 6, we present the pmSys-trainer web-portal. The two last chapters will compare the existing Ohmage applications to the new pmSys applications. Through this comparison, we can evaluate how pmSys is functioning as a monitoring system, and how well it have replaced pen and paper, as well as existing systems.

Chapter 4

Ohmage back-end

In this chapter, we present the Ohmage back-end. This back-end solution comes with a web-APIs that will be the foundation for our new system pmSys. With this back-end, we were able to design and develop two new tailor-made front-end solutions for monitoring football players. We will first present the original idea for the pmSys back-end, and explain why we ended up using an existing back-end instead. We will then present how we have configured the back-end to meet our requirements for pmSys.

4.1 First version of back-end

Before we were aware of Ohmage, we actually started by designing a complete new back-end for our project pmSys. The envisioned back-end had to be able to ensure secure communication, proper authentication, account management, access control, data storage and management. The back-end also had to have a communication channel through an API in order to communicate with web applications and mobile applications. Our proposed design for the back-end, is shown in Figure 4.1.

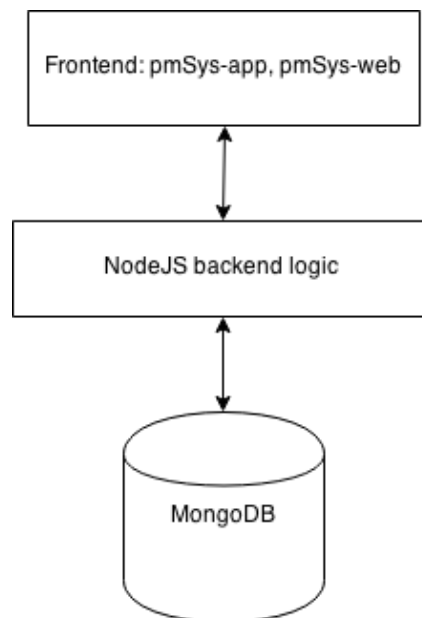


Figure 4.1: Abstract Architecture: pmSys back-end early phase

nban

- NodeJS is a platform build on Chrome's JavaScript runtime. It as an asynchronous event driven framework, and can handle many connections concurrently. This is in contrast to todays practice where concurrency is achieved when OS threads is deployed. Thread-based networking is relatively inefficient and difficult to use. But because almost no function in Node directly performs

any I/O operations, the processes will never block. Therefore, we can develop a system without worries of deadlocks, also making it easier to develop a scalable system [26].

- MongoDB is one of the most widespread NoSQL databases. It is an open-source, document-oriented database. Compared to the more traditional relational databases, MongoDB claims that NoSQL databases are more scalable and provide better performance. It is also built to deal with the limitations of SQL databases, particularly scalability, replication and unstructured data storage [27].

As a part of our planned thesis, we wanted to compare this modern type of networking application with the more traditional back-end solutions (i.e Apache Tomcat and MySQL databases) in terms of performance, scalability and reliability. During our master programme, we had a course about open-source frameworks and app-development. We really got into the open source environment and wanted to explore the possibilities of these new development platforms. But as we progressed in just creating a simple "proof of concept" back-end with those two frameworks, we noticed that creating such an advanced system was very complicated. Creating a proper back-end that met all the requirements was challenging, and there were many aspects of the development process that we have little to no experience with.

We were then introduced to Ohmage framework, which offered us developers all the features and tools for collecting data. TIL have in collaboration with UiT already been using this system as their reporting system for earlier seasons before the pmSys project was started. With all the advantages of building an Ohmage-based system that have been mentioned in the previous chapters, we decided to use the Ohmage back-end. This will give us a head start by having a back-end ready, but also because we can avoid all the challenges and complications that arises when trying to build our own back-end.

4.2 Ohmage web API

The Ohmage web-API supports CRUD operations towards the back-end. This means that all the Ohmage objects are accessible and can be manipulated through the web. In pmSys, we will have two front-end solutions that will use the Ohmage back-end: one mobile application for the players and a web-portal for the coaching staff. The main operations towards the back-end that is used in pmSys is shown in Table 4.1 [28].

Entity	Short Description	URI
Campaign Manipulation	CRUD operations on campaigns.	/campaign
Class Manipulation	CRUD operations on classes.	/class
Survey Manipulation	CRUD operations on surveys.	/survey
User Authentication	Login to Ohmage and logout.	/user
User Manipulation	CRUD operations on the user	/user

Table 4.1: Ohmage web-APIs which are used in pmSys

The Web-API provides a navigable interface to the Ohmage data model. The developer can for instance access different data elements depending on the hyperlink that had been provided. The data is accessed through a **Uniform Resource Locator (URL)**, using a well-known protocol, **Hypertext Transfer Protocol (HTTP)**. This is the main advantage of Ohmage back-end, where developers can build third-party application using the Ohmage platform, without actually knowing the specifics of the Ohmage technology or design constrains. All data returned from the Ohmage API is retrieved in a well-known format, **JavaScript Object Notation (JSON)**. To perform an operation towards the back-end, the API uses HTTP-methods to map the CRUD operations to HTTP-requests, giving the developers direct access to the Ohmage data model when requesting a resource at a given URL. All of the Ohmage available operations are reached by a POST-request, which is the HTTP-method used to request a resource at the URL when you expect to do something with the entity. Depending on what resource we want to get from the Ohmage back-end we specify a POST-request.

```

POST /app/user/auth HTTP/1.1
Host: dev.ohmage.org
User-Agent: Mozilla/5.0 (Linux; U; Android 1.0; en-us; ...) ...
Content-Length: byte-length-of-content
Content-Type: application/x-www-form-urlencoded

user=user&password=password&client=ohmage-android

```

Figure 4.2: Example of a HTTP-request to the Ohmage back-end

In the example in Figure 4.2, the POST-request is sent to the URL that contains the authentication module of Ohmage. The parameters we send with this request, is the username and the password of a user. The API will always require that a client is specified. The client can be a web-browser or a mobile client. The parameters will vary depending on what resource you are requesting. If the POST-request manages to reach a valid back-end, the back-end will then respond with a JSON object, containing either the requested object (See Figure 4.3) or an error message (See Figure 4.4) explaining the error [29].

```

{
  "result" : "success",
  "hashed_password": "42..."
}

```

Figure 4.3: Example response from Ohmage back-end in JSON-format

```

{
  "errors": [
    {
      "code": 0000,
      "text": "explanatory text"
    },
    ...
  ]
}

```

Figure 4.4: Example error response from Ohmage back-end in JSON-format

In Table 4.1, we highlighted the modules that are being used in pmSys. In the next sections, we will explain how these entities function individually, but also in conjunction with each other. Later on, we also present how we have configured the Ohmage back-end to suit our purposes and requirements in pmSys.

4.3 User

A user is defined as an individual who can authenticate with the system and perform certain actions. The actions that the user can perform are determined by their role within the system. To create a new user in the system, you need to have an account with the admin role. When the Ohmage back-end is deployed, an default admin account is created to manage the system [30].

When a new user is created, the following properties must be set:

- Username - the length of this string has to be between 3 and 25 characters and must contain alphanumeric characters.

- Password - the length of the password must be between 8 and 16. It requires; at least one lowercase character, and at least one digit. According to Ohmage Wiki it must contain special letters, but our usage of the Ohmage back-end has not required it so far.

4.3.1 Role

A user is given a role depending on what actions they have to perform towards the system. If we exclude the admin role, which can perform every action available on the system, a new user can perform different actions depending on what roles they are given in a class and campaign.

4.4 Class

A class is a named group of users. An account with the admin role is required in order to create a class and add user to the class. When creating a new class following properties must be set [30]:

- Class URN - a **Uniform Resource Name (URN)** that is unique.
- Class name - a descriptive name of the class.
- Description (optional) - some form of description for the class, if the name is not descriptive enough.

In a class, a user is given a class role. The two class roles are named privileged and restricted. What actions the roles within a class can perform is shown in Table 4.2.

Operation	Privileged	Restricted
Read the Class properties	anytime	anytime
Read list of logins for the Class	anytime	anytime
Read detailed information of the class (Login, Roles, Personal info)	anytime	never
Read List of campaigns that the class is associated with	anytime	anytime
Add and remove users from the class	anytime	never
Update class information, change users role	anytime	never

Table 4.2: Class roles

4.5 Campaign

A campaign consists of an XML configuration, which will define surveys and the prompts within the surveys. The XML-configuration must also include a unique campaign URN, and a campaign name. Users that are given the permission to create campaigns can become authors, define and create campaigns. This is limited in pmSys on purpose, where we only allow for admin to create campaigns. This is because we have been given a set of surveys that is the same for all teams. Because of this, we do not want to give unnecessary permissions for the users.

When a class is added to a campaign, their role within the campaign is based on their class role. Privileged users become **supervisors**, while restricted users become **analysts**. These campaign roles will control the visibility of the data and what actions the user can do in the campaign. In Table 4.3, we show which actions the roles can perform within the campaign.

Furthermore, a campaign has two states, which is running or stopped. When it is active and running, users may upload their survey responses. If it has stopped, no surveys can be uploaded. This is used when a campaign is unused, but the response data have to be kept. This is because when a campaign is removed, the response data is deleted with it. The campaign is also configured to be public or private. In public state, everyone can read all the responses, while in private state it is limited.

Operation	Supervisor	Analyst
Read campaign properties (XML, name, description, state)	anytime	anytime
Read what classes that participate in the campaign	anytime	anytime
Read user roles of others on the campaign	anytime	only shows the author
Update running state	anytime	never
Update privacy state	anytime	never
Add/remove class, supervisor, analyst to the campaign	anytime	never
Delete campaign	anytime	never

Table 4.3: Campaign roles

XML-configuration

The XML-configuration is the base of a campaign. In this section, we will show a XML-configuration and explain how to configure one. In the example XML-file (See Figure 4.5), we will define a simple campaign with a singular survey, together with some prompts. Prompts are essentially questions of a survey, but in Ohmage they call it for prompts. Ohmage supports several prompt-types in the XML-configuration, and in Table 4.4 we show the types that are used in pmSys.

Type name	Description	Format
Number	A whole or decimal number.	A numeric value.
Single Choice	A single choice answer from the user	A key for the selected answer.
Timestamp	Moment in time (date included) by a user.	An ISO-8601 timestamp [31].

Table 4.4: Prompt-types used in pmSys

In addition to the used prompt-types for pmSys, Ohmage also several other prompt-types, but these are not necessary for pmSys at the current state. Ohmage can also configure prompts with the following types:

- Audio
- Multiple choice
- Multiple choice custom
- Photo
- Remote activity
- Single choice custom
- Text
- Video

To further explain the XML-file (See Figure 4.5), we explain the most important parameters. To define a survey we need a survey-ID. This ID is to identify a certain survey within the campaign, so when a client requests the responses for a certain survey, the system will refer to the ID. Along with an ID, a title and a description is required. The title and description is the information that will be displayed in the UI of a user. To define the questions in a survey, Ohmage is using prompts. Prompts are defined the same way as the surveys, with the requirement of an ID. Along with the ID, a string with the question, a prompt-type and the alternatives for the prompt is needed. A big advantage with Ohmage, is that we can request response data through prompt-IDs. This way, we can carefully select which prompts we want response data from, in order to highlight them for the users in our applications. The XML-file can also configure conditions, which will allow the user to skip questions based the alternative chosen on the prompt [32]. When an XML-file is completed, it is uploaded through the Ohmage Web Frontend.

```

<?xml version="1.0" encoding="UTF-8"?>
<campaign>
  <!-- an unique campaign URN -->
  <campaignUrn>urn:campaign:demo:eng:srpecoach</campaignUrn>
  <!-- name of the campaign -->
  <campaignName>Surveys for coaches-demo-eng</campaignName>
  <!-- under this block, one or several surveys can be defined -->
  <surveys>
    <survey> <!-- start of a survey -->
      <id>plansrpe</id> <!-- ID of the survey -->
      <title>Planned srPE</title>
      <description>Plan srPE</description>
      <submitText>Survey is done. Thank you!</submitText>
      <showSummary>true</showSummary>
      <editSummary>false</editSummary>
      <summaryText>Results</summaryText>
      <anytime>true</anytime>
      <contentList>
        <prompt> <!-- prompt -->
          <id>srpeType</id>
          <displayLabel>Session</displayLabel>
          <promptText>Type of session</promptText>
          <promptType>single_choice</promptType>
          <!-- alternatives for the prompt/question -->
          <properties>
            <property>
              <key>0</key>
              <label>Football session</label>
              <value>0</value>
            </property>
            <property>
              <key>1</key>
              <label>Endurance training</label>
              <value>1</value>
            </property>
          </properties>
        </prompt>
      </contentList>
    </survey>
  </surveys>
</campaign>

```

Figure 4.5: Example of a XML-configuration for a campaign

Access rules

An important part of the system is being able to read the survey responses stored in the system. The access rules that Ohmage has defined for reading responses is listed in Table 4.5

Operation	Supervisor	Analyst
Read own private or shared survey responses	anytime	anytime
Read someone else's shared survey responses	anytime	Only if the campaign is shared
Read someone else's private survey responses	anytime	never

Table 4.5: Survey response read: Access rules

4.6 Configuration

In the previous sections, we have highlighted the main features in Ohmage that we are using in pmSys. In order to match the requirements for pmSys, we have configured Ohmage to suit our purposes. First off, we have a specific way of creating the users. This includes a python script that runs directly on the Ohmage database. This is done in collaboration with our host in UiT, that is hosting the Ohmage back-end that we are using. Since pmSys will register and store personal data from the user, all of the stored data has to be anonymous. The username stored in the database must not be identifiable with any football player. In order to match this requirement, all the usernames are randomly generated through the Python script. The format of the usernames includes an acronym of the team, followed by a randomly generated string of letters and numbers. The password will also be randomly generated together with the username, but the user will have the opportunity to change to a desired password. Only the staff, who have been permitted to process personal data will have access to the mapping between the random generated username and the specific person in real life. We then assign the players and staff to their roles (See Figure 4.6). It is important that the players only can read their own data, while it is important to let the coach be able to read all the players response data.

User	Class role	Campaign role	Example username
Player	Restricted	Analyst	uio.ad123l
Staff (coach, medical, etc.)	Privileged	Supervisor	uio.ads12l

Table 4.6: Configuration: Users

We treat classes as a football team in pmSys. Each team will get two classes created, one for the players and one for the staff. Each team will get a unique URN and name. The URN consists of what entity it is, in this case a class. Then, it is followed up by the acronym of the team name, an identifier to separate players and coaches and a version number of the class. See Table 4.7 for examples.

Team	Players	Staff
University of Oslo	urn:class:uio:players:1	urn:class:uio:coaches:1
Tromsø IL	urn:class:til:players:1	urn:class:til:coaches:1
Norwegian Nationalteam	urn:class:nor:players:1	urn:class:nor:coaches:1

Table 4.7: Configuration: Classes

In theory, we only need two XML-configurations of surveys: One campaign for the player surveys and one for the coach surveys. The player classes can be added to the player campaign, and the coach classes will be added to both the player and coach campaigns. The coach needs to be added to the player campaign in order to read their data. This was thought to be the optimal configuration (See Figure 4.6), since all the teams will use the same surveys that have been implemented in pmSys.

This configuration was implemented, because we initially thought that Ohmage separated data through classes. That meant that teams can only access data their own, and cannot read another team's data. Ohmage unfortunately separates data through campaigns. The problem with the architecture described in Figure 4.6, is that every privileged user that is participating in the campaign will get access to all the response data. In a real life scenario, a coach can in theory access another team's response data. In order to solve this major issue, we had to configure a XML-configuration for each team (see Figure 4.7).

This architecture ensures that the data for one team is only available for that chosen team. Players

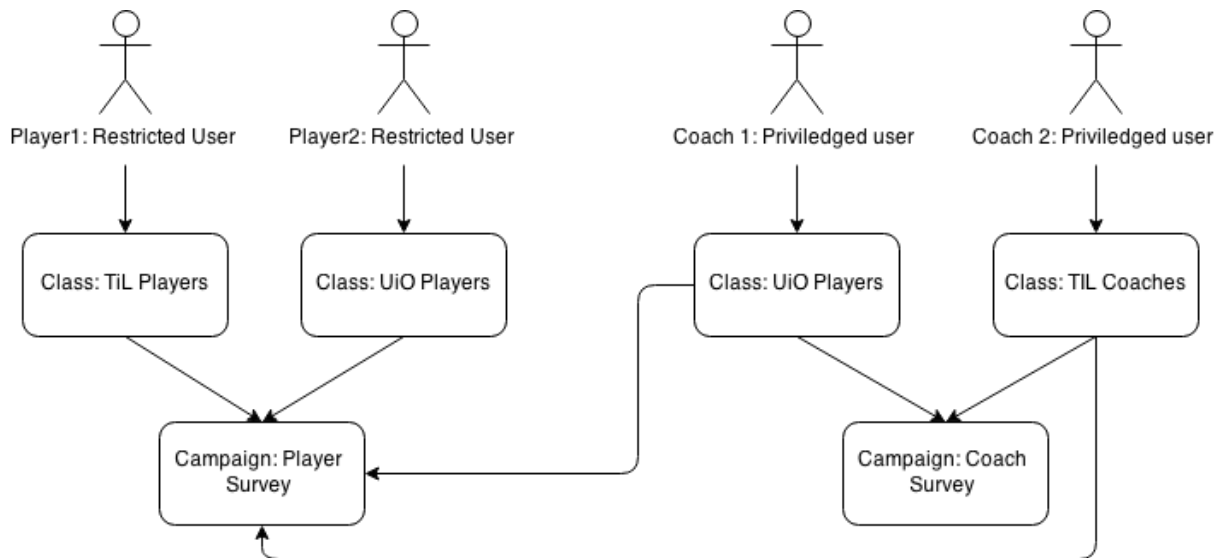


Figure 4.6: Optimal configuration

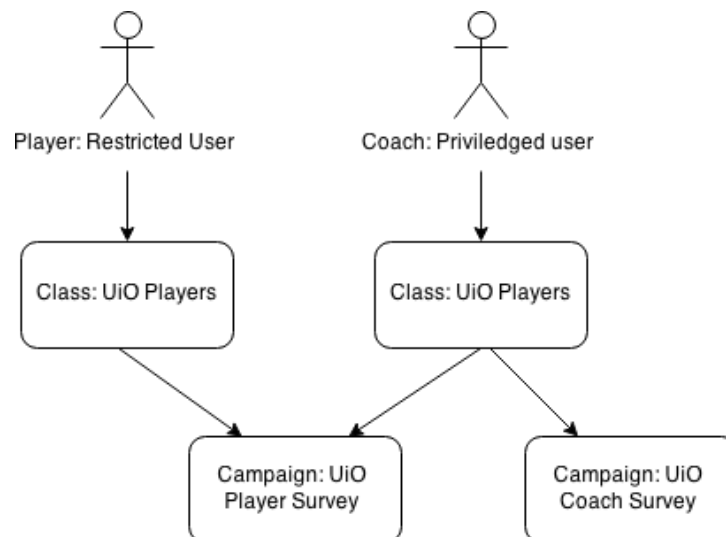


Figure 4.7: Used configuration

will automatically have their data separated because restricted users only have access to their own data, and the coaches can only access data from the campaigns they are a part of. We have chosen to configure all campaigns to have the privacy state as **private**. This is to ensure that the separation of the response data between the players within a team is held, but also the full separation between the teams.

4.7 Summary

Ohmage back-end of the Ohmage platform is a concrete example that will allow us to follow the PS-approach introduced in Section 2.2. It provides secure communication, proper authentication, account management, access control, data storage and management for all types of PS-deployments. Through the web-API, we can develop applications that can communicate through the web, and access the Ohmage model without knowing how Ohmage is built up. For pmSys, it has saved us a lot of time, moving away from the complexity on making a proper back-end for data collection. Thus, allowing us to focus on creating a system that is tailor-made for monitoring football players. In the next chapter, we will present the pmSys-app, which is the new mobile application for capturing self-report data.

Chapter 5

pmSys-app

In the requirement analysis in Chapter 3, we pointed out that the Ohmage mobile applications had several drawbacks when used specifically for football players. We will in this chapter, present the issues that have been discovered, that led to the development of a new mobile application for capturing self-report data, pmSys-app. We will present the design and the frameworks used for the development of this application. Then finish the chapter by evaluating the pmSys-app up against the Ohmage application through various benchmarks and end-user studies.

5.1 Related work: Ohmage mobile applications

In this section, we evaluate the Ohmage Android application and the Ohmage MWF application. We present sections of the applications that have been unsatisfying when used specifically for football players. The presented information has been done in collaboration with NiH and the participating Toppeliga teams that have been using the first version of pmSys. The drawbacks are in the form of technical issues we have discovered, or usability issues discovered by the players. The iOS users have used the Ohmage MWF application. Ohmage has two mobile applications for Android users, Ohmage MWF and the Ohmage Android. In this evaluation, we focus on the Ohmage Android, because the Ohmage MWF is not up to date for Android devices.

5.1.1 Login

In Figure 5.1, we present the login sections of the applications. The main issue with the MWF version for Android (See Figure 5.1(a)), is that it does not support other servers besides the Ohmage default servers. Because of this, the players cannot login to our version of the Ohmage back-end. This problem practically made the Ohmage MWF unavailable for Android users. The Ohmage Android application (See Figure 5.1(c)) supports custom servers besides the default Ohmage servers. This gives the players an option to use the pmSys's instance of the Ohmage back-end, allowing them to access the questionnaires that we have configured for them. This is the main reason that we had to use the Ohmage Android over the Ohmage MWF. While the Ohmage Android has a well-functioning login section, the MWF version for iOS users has issues. The iOS version supports custom server as shown in Figure 5.1(b). The support of custom servers is crucial, since we are using our own version of the Ohmage back-end.

If the user is able to enter to correct URL to access custom server on the first attempt, the module is perfectly fine in Ohmage MWF (See Figure 5.2(a)). The issues arise when the URL is entered incorrectly when trying to access a custom server. If the user then attempts to rewrite the URL correctly, the application still interprets the URL as a faulty one. In Figure 5.2(b) we have purposely written a faulty URL. The application as it should, interpret it as a faulty URL and give the user feedback. The issue is when the user corrects the URL, it will still be interpreted as a faulty URL, as shown in Figure 5.2(c). The URLs in Figure 5.2(a) and Figure 5.2(c) are completely the same, but the application will still interpret one of them as a faulty URL. The only known solution we have discovered is to select a default server, reselect the custom server option, re-enter the URL and then retry. Since the addition of custom servers is an essential part of pmSys, this is a major drawback that must be fixed.

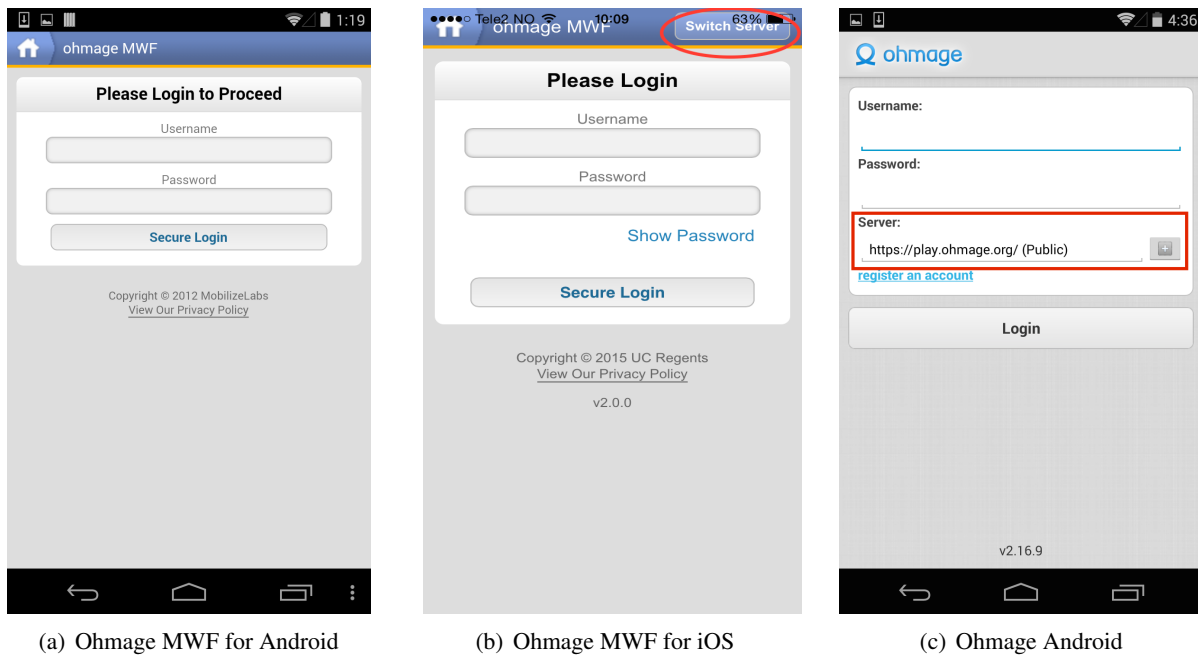


Figure 5.1: Login: Ohmage Android and Ohmage MWF

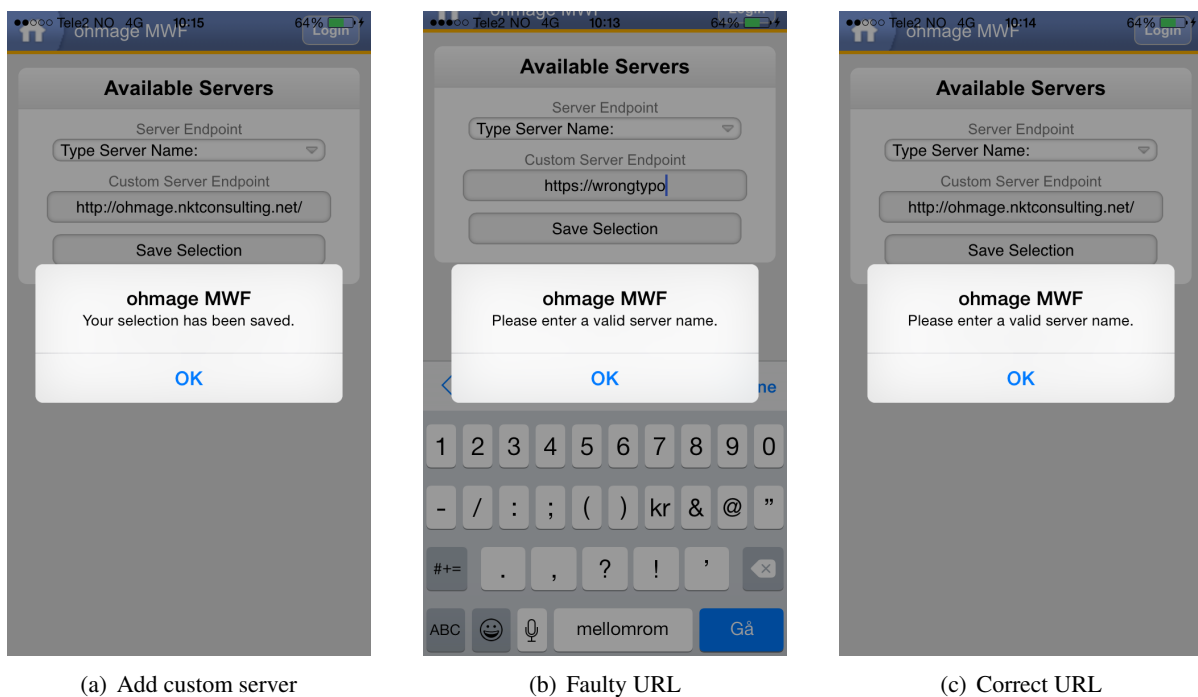
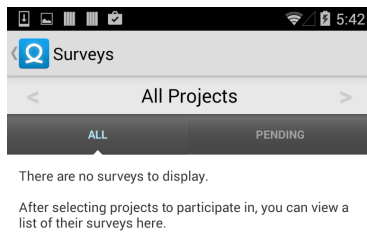


Figure 5.2: Login: Issues with custom server on Ohmage MWF

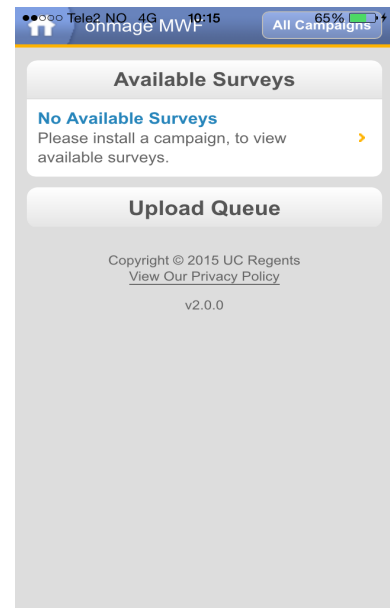
5.1.2 Campaign (Projects) and Surveys

This section consists of campaigns and surveys. The Ohmage Android application has named a campaign, a project instead, but both are practically the same. The first time a player uses the application, it can be confusing. When the player has successfully logged in, they will be greeted with a dashboard that acts as the main menu. From here all the features within the applications is presented. The most important feature of pmSys is the capability to collect data, and are done through questionnaires presented in the survey section. The problem is that no surveys are available for the user, as shown in Figure 5.3. To access the surveys, the user is required to access to campaigns (iOS) or projects (Android) to add the surveys. In Figure 5.4, we show the process of adding survey on the MWF version, but this

process is the same for both applications. This is an unnecessary process, but is required to be able to start reporting. This can result in confusion that may lead to a troublesome start for new users.



(a) Ohmage Android

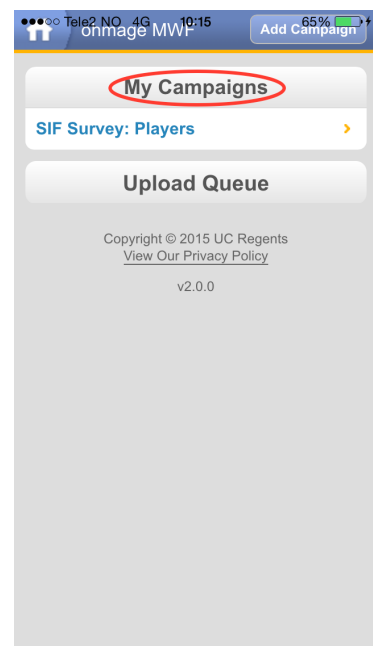


(b) Ohmage MWF

Figure 5.3: Survey: Empty list



(a) List of available campaigns/projects



(b) Added campaign

Figure 5.4: Campaign: Add campaign and surveys in Ohmage MWF

5.1.3 Reporting

In the requirement analysis in Chapter 3, we highlighted that the process of reporting must be a user-friendly and fast process. The focus on the reporting section was to optimize the process and minimizing the steps needed to finish a report. We will demonstrate that both the Ohmage applications have UI solutions that are time consuming, and require more user interactions then necessary. For the

demonstration, we will use the Ohmage MWF, since the reporting process is completely the same on both applications.

In Figure 5.5, we present the process of answering the questionnaires. To complete a question based on the prompt-type "single choice" (See Figure 5.5(a)), the user has to select an alternative and press "next" to complete it. For the prompt-type "number" (See Figure 5.5(b)), the player can only increase or decrease the value by one for each click. The other alternative is to write the value manually, but this is not presented in an intuitive way. If we assume that a default value for training length is set to 90 minutes, the player is required to click 30 times if they just had a training session, which lasted 60 minutes. At the beginning, this will not be an issue because the players will still be unfamiliar with the questionnaires. But pmSys is a monitoring system that requires the player to answer the same questionnaires every day over a long period. Over the course of the season, it is safe to assume that the players will start memorizing the majority of the questions, and will therefore be able to submit reports fairly quickly. With the existing UI, the extra clicks required will be a hindrance for how fast they can finish a report.

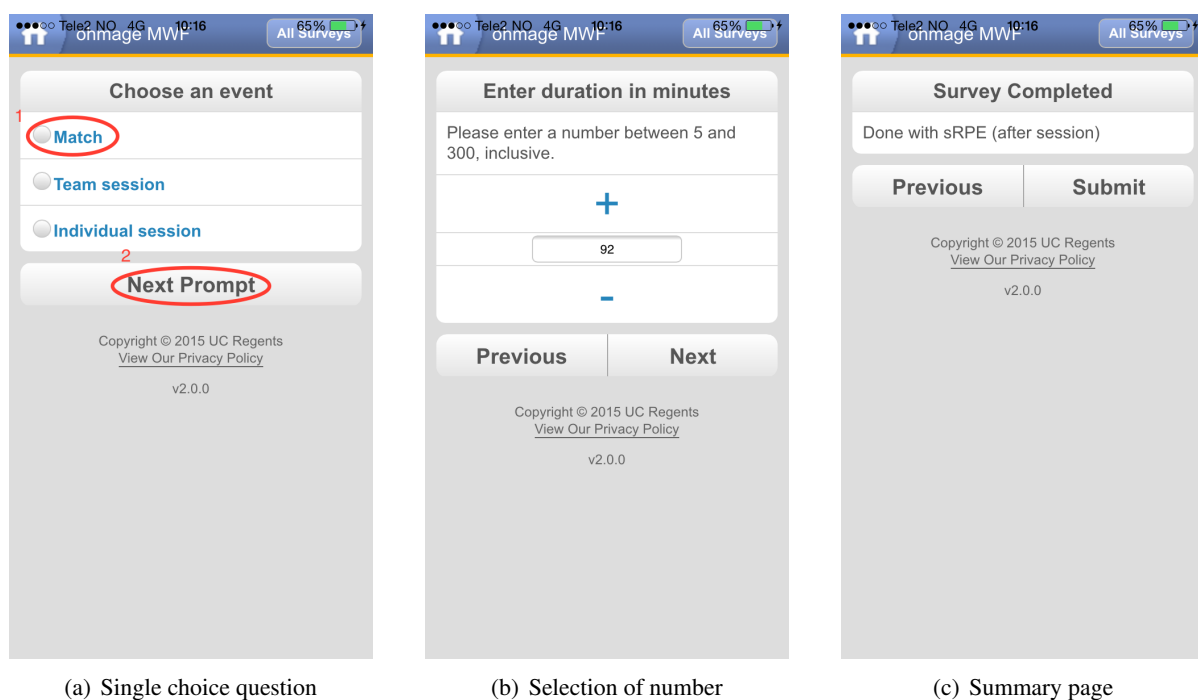


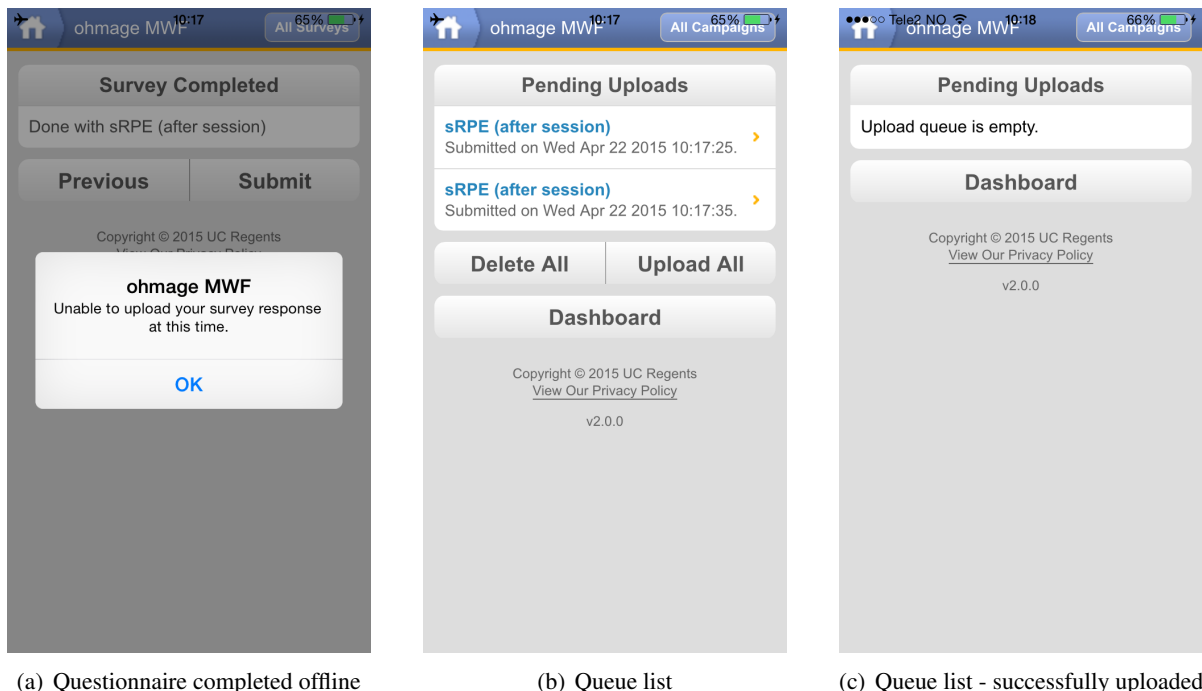
Figure 5.5: Reporting: Presentation of questionnaires in Ohmage MWF

When the player is finished answering the questionnaire and is ready to submit the report, a summary page is shown to indicate that the questionnaire is done (See Figure 5.5(c)). A big drawback is that the summary page does not summarize the questionnaire, which had been completed. The application does not provide the user with an interface to validate the report, and the only way to check the report is to manually go back through all the questions. This can lower in the quality of the response data or be more time consuming. Firstly, the player might not bother to manually check the answers after a completed report, that may lead to an inaccurate report. Secondly, since there is no summary page for the actual report, the player will not have an easy way to validate the report, and are forced to go back and check manually.

5.1.4 Queue

The Ohmage applications both support offline reporting. This is important to maintain the availability of the system, and allow the players to report even if they do not have access to Wi-Fi or have cellular coverage. The main drawback with this section for both applications is that the offline feature is not an automated process. We will show an example from Ohmage MWF (See Figure 5.6), but the process is the same for both applications.

When a player completes a questionnaire without access to internet, the application will give a



(a) Questionnaire completed offline

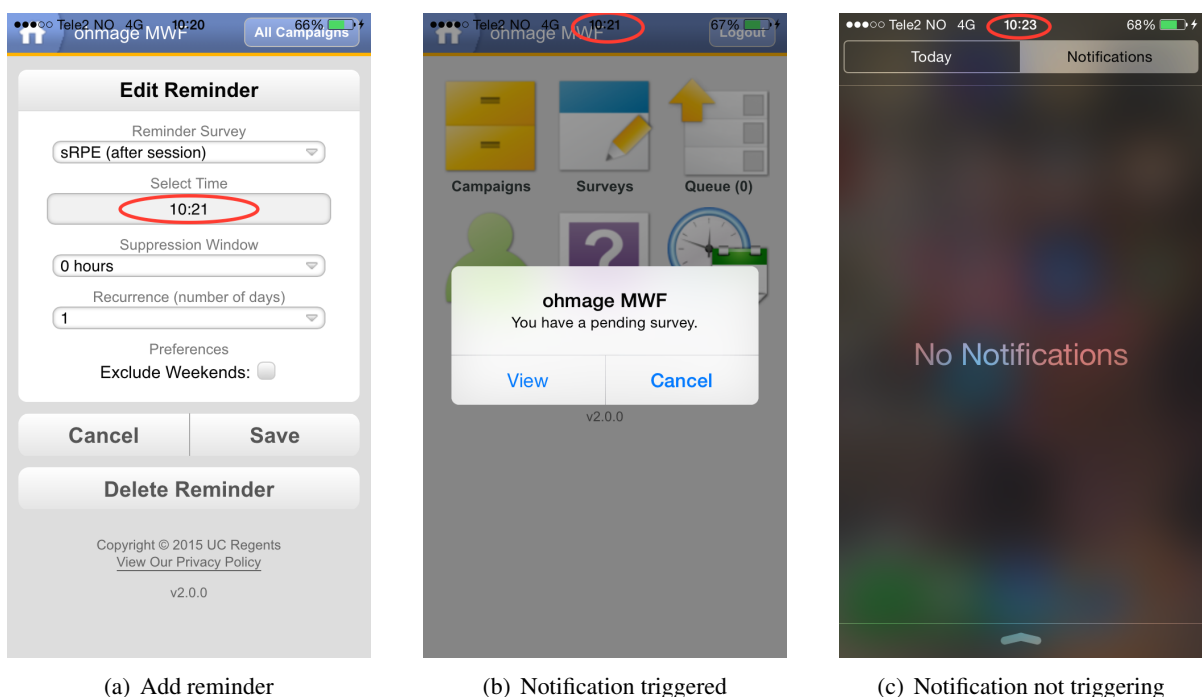
(b) Queue list

(c) Queue list - successfully uploaded

Figure 5.6: Offline reporting: Queue of reports in Ohmage MWF

feedback that the response cannot be uploaded to the server (See Figure 5.6(a)). The reports that are completed offline is therefore placed in a queue, waiting to be uploaded (See Figure 5.6(b)). The main drawback of this feature is that the process is not automated. The player is required to remember that they have reports in the queue, and then manually give the application permission to upload the reports. If the upload is successful, the list is emptied, as shown in Figure 5.6(c).

5.1.5 Notifications



(a) Add reminder

(b) Notification triggered

(c) Notification not triggering

Figure 5.7: Notifications: Local reminders in Ohmage MWF

The Ohmage applications features reminders in form of local notifications, and require the players to manually setup the reminders they want. This is ideal for players who have many individual training sessions with nobody to remind them to make a report. The Ohmage Android application has fully functional local notification module, which works with the application in the foreground, but also when the application is in the background. The only minor issue, is that the reminder is not highlighted by the dashboard, but instead hidden inside the survey module and can be hard to find. The Ohmage MWF has a functional local notification module that works when the application is running in the foreground. In Figure 5.7(a), we display a planned reminder that will trigger at 10:21, and as expected, the application triggers an event that shows that we have a report to make (See Figure 5.7(b)).

The main issue is when Ohmage is closed and running in the background. We initiate a new reminder that is supposed to trigger at 10:23, and close the application to let it run in the background. Unfortunately, the reminder will not trigger an event and remind the player to report, because the application is running in the background, as shown in Figure 5.7(c). This is a big drawback, especially if we look at this scenario from the football players perspective. How often will they actually have the application open? It would probably only be when they are actually reporting. Besides reporting, they will most likely not be using the application. If the reminders do not trigger when it is ran in the background, the feature is useless.

5.1.6 Feedback

In order for pmSys to succeed, we need the players to report regularly and make them aware of the importance of the reporting. In addition to have tools to help them remember to report at the right time, it is also necessary to let the players participate in the monitoring aspect of the system by giving the feedback on their development. The Ohmage Android features a response history (See Figure 5.8), which the players can use to see their past reports. While this is indeed a welcomed tool by the players, looking at raw data from one day at a time does not provide them information about their development and trends. The Ohmage MWF application on the other hand does not provide them with information, and because of this, the application is a pure data collecting tool. Without proper feedback that will motivate the players, it may discourage some players from reporting. With the lack of data, the monitoring and analysis of the player becomes inaccurate.

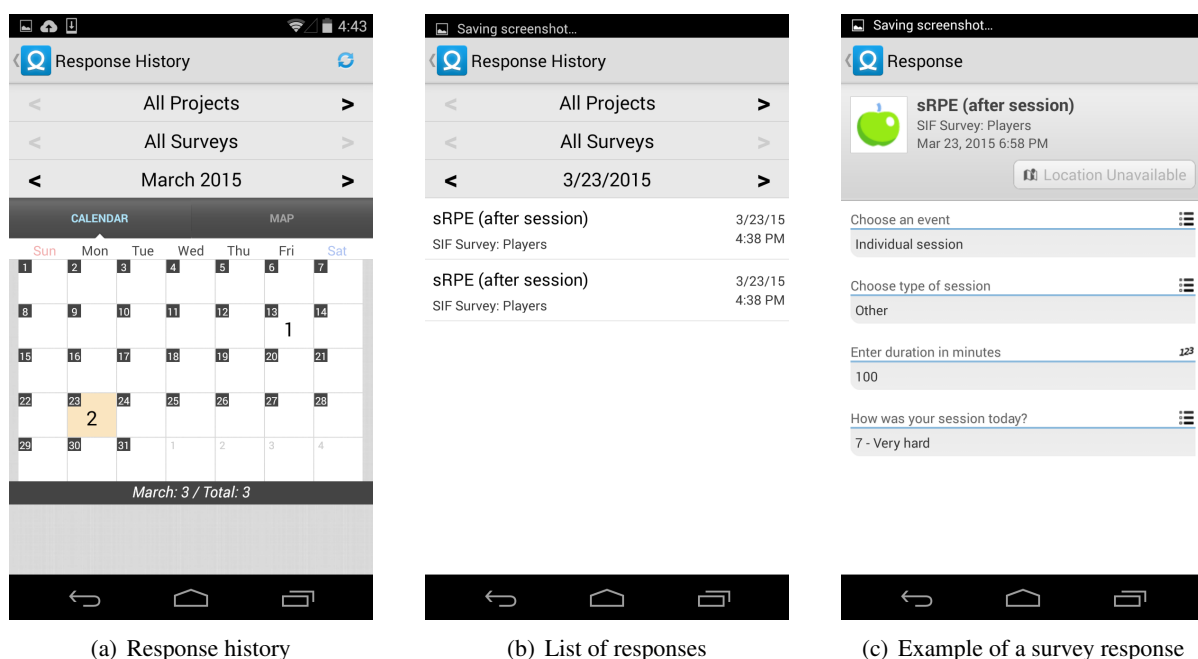


Figure 5.8: Feedback: Response history in Ohmage Android

5.1.7 Summary

The mobile applications from Ohmage that were used during the first version of pmSys covered many of the requirements. The Ohmage Android application was an especially well functioning application, and all the features that was included worked as intended. But both have clearly not been developed with focus on optimizing the reporting process by reducing the amount of steps necessary to complete a report. The Ohmage MWF for the iOS users shares the same issues as the Ohmage Android application, but in addition to those issues, it also had many bugs, and we discovered several technical issues. Overall, the Ohmage application's main objective is to capture data, while pmSys main goal is to monitor football players. Our focus group has requirements that the Ohmage applications cannot meet. It lacks the presentation of the data for the players in a way that allows them to participate in the monitoring. It also lacks proper tools to help remind them to report in order to ensure quality and quantity of the reports.

5.2 pmSys-app features

Based on the experiences with the Ohmage MWF and Ohmage Android in our first version of pmSys, we wanted to build on the Ohmage platform, but also improve the drawbacks that were presented in the previous section. Features that are included in the pmSys-app (30. march 2015) are improvements of the existing Ohmage applications, but some of the implementations are also new features that is included based on feedback.

- Presentation of questionnaires.
- Optimized and fast reporting.
- Supports local notifications. Players can set up their own reminders locally, that will work without access to internet.
- Supports remote notifications. Players will receive reminders as push-messages.
- Feedback in form of visualizations.
- Offline reporting. An automated process for queuing and uploading of reports that are completed without access to internet.
- Cross-platform support. PmSys-app currently supports two major mobile platforms: iOS and Android.

5.3 System Design

PmSys-app is a mobile-application that consists of more then just one development technology. First, it is a **Single Page Application (SPA)**, but in addition to that it is also a Hybrid mobile application. We will explain both approaches and give our explanation for using these approaches to build the mobile application. We will also present the frameworks and tools that have been used to develop and deploy pmSys-app.

5.3.1 Single Page Application

Before SPA was introduced, the classic web applications were based on a multi-page interface model. In other words, for every request the user made the entire interface had to be refreshed. Due to new emerged technologies, dubbed AJAX (Asynchronous JavaScript and XML), SPA is a possibility. With SPA the web application is composed of an individual page that can be updated independently on each of the user actions. This means that the web application does not need to be reloaded like the classical multi-page web application. This will in return, increase level of interactivity, responsiveness and user satisfaction of the application [33].

AJAX

AJAX itself is not a technology, it consists of several technologies that when combined can create more powerful web applications. We will not go in depth on each of the technologies used, but instead we will give a brief explanation on what it does [34]:

- XHTML (Extensible HTML) and CSS for standards-based presentation of the view.
- Document Object Model (DOM) for dynamic interaction and display.
- XML (Extensible Markup Language) and XSLT (EXtensible Stylesheet Language) for data interchange and manipulation.
- XMLHttpRequest to asynchronously retrieve data.
- JavaScript to bind it all together

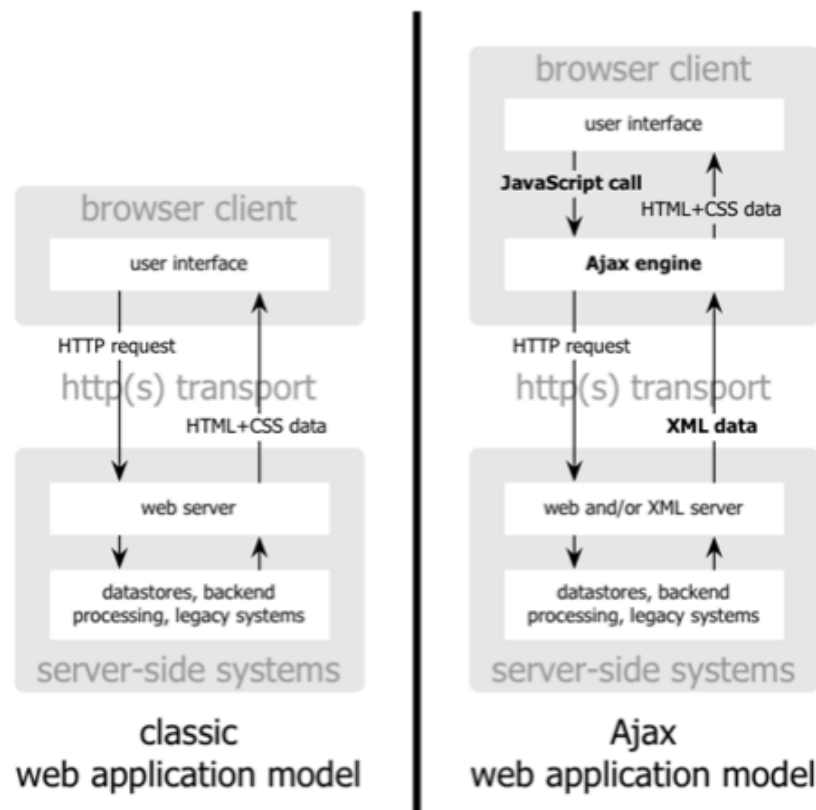


Figure 5.9: Classic (left) vs Ajax (right) web application model [34]

In Figure 5.9 we compare the traditional web application (left) and the Ajax model (right). In the traditional web application, when a user performs an action, the server will respond to the user interaction and reply with the correct content that the user requested. While the server is working, the user must wait for the server (See Figure 5.10). In this multi-page model, for every event, the user has to wait for even longer. This was previously the drawback of web application compared to the more powerful desktop applications that have superior performance and responsiveness. With AJAX, the start and stop interaction is eliminated in the web by introducing an intermediary - the AJAX engine (See Figure 5.9). This is a layer between the user and the server that is loaded at the start of the session. Instead of loading the webpage, this engine is loaded and is responsible for rendering both the UI and communicating with the server on the user's behalf. This allows the user to communicate with the UI in the application, independently of the communication with the server (See Figure 5.11). This asynchronous communication between client and server allows for the application to be built within a

single page [34]. Every event from the user that normally generated an HTTP request back to server, is instead taken form as a JavaScript call to the engine instead. Any response to the user that does not need a request towards the server such as simple data validation, editing data in memory, and navigation will be handled by the AJAX engine alone. If a request to the server is needed, such as submitting data, loading additional interface code or retrieving new data, the engine will asynchronously, usually using XML to send requests without stalling the interaction of the application [34].

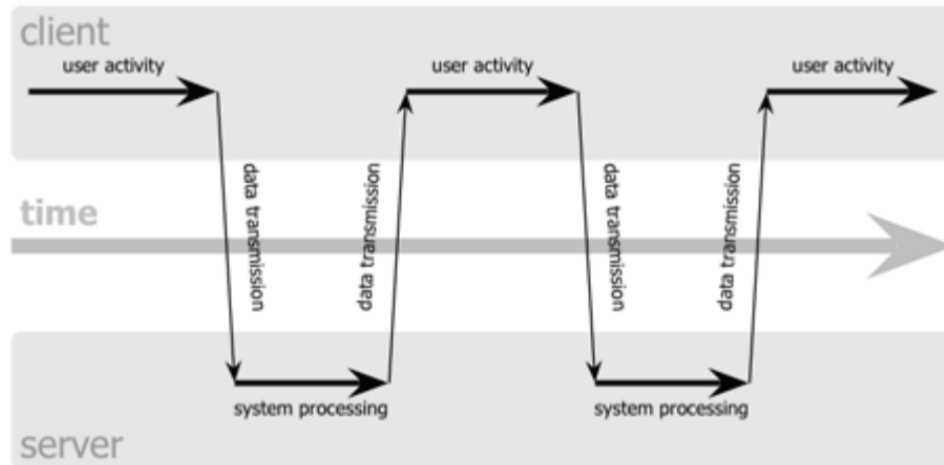


Figure 5.10: Classic web synchronous communication [34]

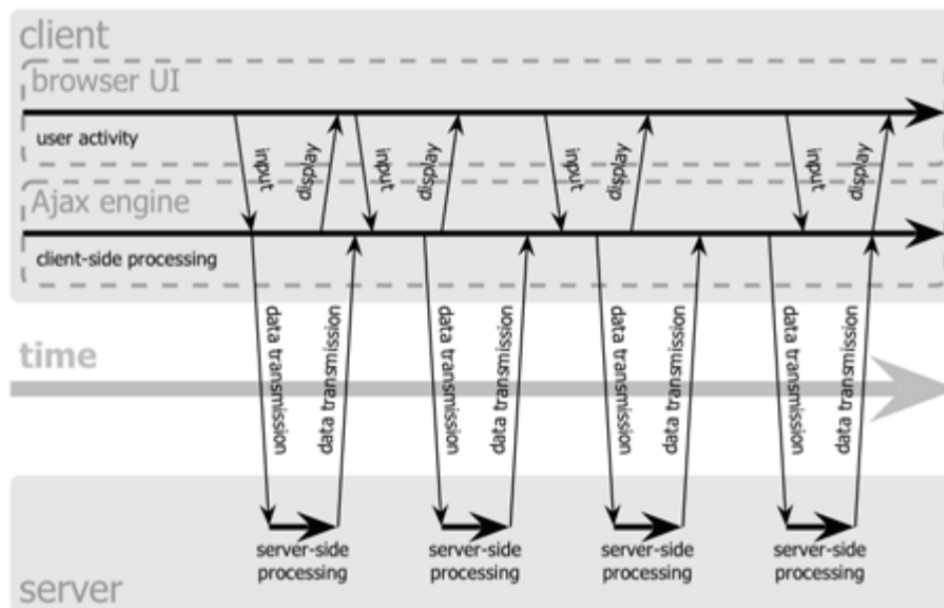


Figure 5.11: AJAX web asynchronous communication [34]

5.3.2 Hybrid mobile application

The hybrid mobile application approach is a way to develop simple mobile applications using web technologies like HTML, CSS and JavaScript. These applications can be developed without the knowledge of the programming languages on the Operating Systems (See Table 5.1).

A web-application is an application built on the same browser based technologies. The modern mobile devices consist of powerful browsers that support the advanced capabilities of the modern versions of these technologies: HTML5, CSS3 and advanced JavaScript. Through the new technology, we have moved from static webpages to powerful, rich browser-based applications. The drawback of

Mobile OS type	Skill set required
Apple iOS	C, Objective C
Google Android	Java
RIM Blackberry	Java
Symbian	C, C++, Python, HTML/CSS/JS
Windows Mobile	.NET
Windows 7 Phone	.NET

Table 5.1: OS and their programming language

developing a web-application is that the developers cannot access the device APIs. The device APIs allows the application to interact directly with the hardware or other features that the phone can offer [35].

The Native applications have binary executable files that are downloaded directly to the device and stored locally. Once the application is downloaded and installed, the user can launch it like any other service the device offers. A native application interfaces directly with the operating system without any container or intermediary, and can therefore access all the APIs that the **Operating System (OS)** offer. The drawback is that the developers have to write the source code and create additional resources (images, audio, custom OS-specific files). Then using tools provided by the OS vendor, the source code is compiled to create an executable that runs on that specific OS [35].

Hybrid applications combines these two approaches. The developers write significant portions of the code in cross-platform web technologies, while still maintaining direct access to the native APIs when it is needed. How this works, is that the native portion of the hybrid application uses the OS APIs to create a embedded HTML rendering engine, that serves as the bridge between the browser and all of the device APIs. This enables a hybrid application to access all the features that modern mobile devices can offer, where a web-application cannot.

For pmSys, we needed to access several device APIs in order to be able to develop the required features for our system. The issues is that we do not have the knowledge, or time to create native applications for the different OSes that is required to cover the availability for majority of the users. By developing a hybrid application, we can have a singular development environment that we have experience with, but also be able to implement the features required in pmSys that you normally cannot do with regular mobile web-application. By using the hybrid approach, we can enjoy the best of both worlds. On the one hand, with the native bridge we can take full advantage of all the different features and capabilities of the modern mobile device. On the other hand, the majority of the code is written in well known web technologies that can be shared between the different mobile platforms. This in return will shorten the development process, but also make it easier to maintain and introduce new features. In the next sections, we will present the frameworks that have been used to develop a hybrid mobile application.

Cordova

Cordova is the native container to our application. It is a platform for building native looking mobile application using web technologies such as HTML5, CSS3 and JavaScript. With Cordova, pmSys has been deployed to iOS and Android without writing any native code. The framework allows us to access the native mobile functions with a set of APIs by using JavaScript, while still developing the application using web technologies [36].

Ionic

Ionic is a framework that contains mobile-optimized modules for the UI. The main advantages with Ionic are that it is native-focused even though it is built like a web-application. With Ionic, we have a set of components, gestures and tools for building highly interactive mobile application [37]. Due to the lack of design background and UI-expertise, we chose this framework for implementing the UI.

5.3.3 Model-View-Controller (MVC) model

PmSys-app is a complex application with thousands of lines with code. Therefore, it is very helpful to utilize a design pattern for organizing our code. By using a design pattern, it will result in a application that is easier to comprehend, maintain and extend. MVC is a design framework that revolves around dividing a application into model, view and controller. The benefits of this pattern are code reuse in all the layers, and makes the software easier to scale and maintain [38]. The design pattern is adapted by most web applications as the base architecture or the framework [39]. Since pmSys mainly is a web application, this framework suits our purposes perfectly.

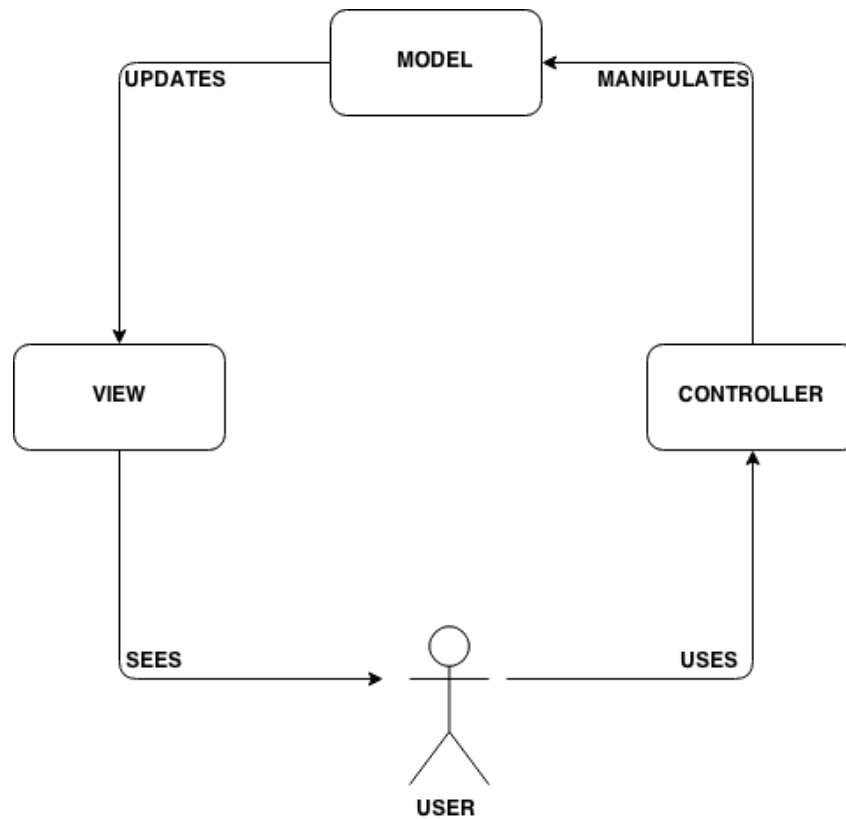


Figure 5.12: MVC model

The model component of this architecture in Figure 5.12 represents the data. It can also contain logic to update the controller if the data changes and it can also access a database for stored data. The next component is the view, which essentially is the UI and what the user can see. Everything the user does in the view is sent to the third component, the controller. The controller routes the request from the user to the correct data in the model. When the model returns the requested data to the controller, the data is then represented to the user through a specified view, depending on what data is returned.

5.3.4 AngularJS

AngularJS is a powerful MVW (Model-view-whatever) framework for web development that allows us to create a SPA. The main purpose of the framework is to bind the view to the model, so that whatever is updated in the view is also updated in the model, and visa versa. PmSys is using AngularJS to implement the design pattern MVC into our code base. In pmSys, we have defined the model as services. A service will provide the application access to data by pulling data from the Ohmage server.

Structure

With AngularJS we have followed a module-based approach, that means that we are dividing the application into smaller modules. Each feature in the application is therefore a module where they each

have a service, view and controller. In Figure 5.13(a), we show what the file system of the application looks like, where each folder is equivalent to a module. In each of the modules, we have defined a service, view and controller (Figure 5.13(b)). We also take advantage of one of the core features of AngularJS [40], thus allowing us to reuse services that are common between the different modules. By injecting a needed service to a module, we do not need to write the same code in each module.

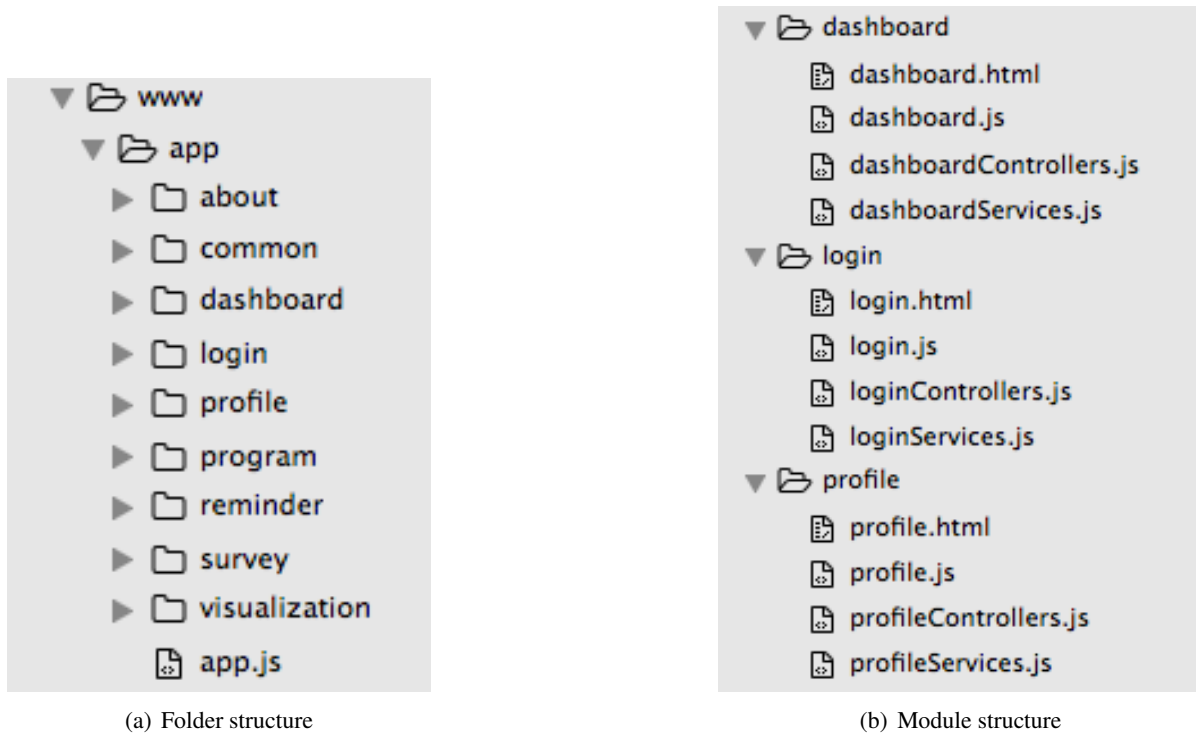


Figure 5.13: PmSys file structure

Routing and States

PmSys uses a core component from AngularJS, the routing module. The routing will tell the application where to find the content based on the user interaction, and is what allows for navigation within the application. The native AngularJS routing module that follows with AngularJS, is called `ngRoute` [41]. In Figure 5.14 we show a simple example application on how AngularJS routing module works. Depending on the URL, the application will provide the correct controllers and views for the requested content. But as applications expands with introduction of new features and increased complexity, the URLs will become more complicated, and the sheer number of URLs is also increased. To avoid being tied to URLs for routing in our application, we will use another routing module from the AngularUI team called AngularUI Router [42]. By using this module, we can change parts of the view even if the URL does not change. It provides a different approach than the native, `ngRoute` module, and changes the view in the application based on the **states** of the application instead of just the route URL. In Figure 5.15, we show an example on the routing of the visualization module in `pmSys-app`, that is based on states.

To create a link with the routing module, we use something called "ui-sref" (See Figure 5.16). The actual hyperlink is then generated and will point to the certain state that has been defined. With the AngularUI routing module, we can take advantage of nested views. If we now assume that we have accessed the visualization page through the first button in Figure 5.16, we can create nested views by using dot denotation. If we click the second button in Figure 5.16, we will access the `visualization.srpe` state. Note that the application has to be in the parent view in order to access the child view through this short notation. If the client wants to go directly into a child view, the complete state name must be used. By taking advantage of this feature of nested views, we can change parts of the view, instead of loading a completely new view for each URL. In Figure 5.17, we show what part of the application that has its view changed based on the state.

```

var app = angular.module('app', ['ngRoute']);
// configure our routes
app.config(function($routeProvider) {
  $routeProvider
    // route for the home page
    .when('/', {
      templateUrl : 'pages/home.html',
      controller : 'mainController'
    })
    // route for the about page
    .when('/about', {
      templateUrl : 'pages/about.html',
      controller : 'aboutController'
    })
});
// create the controller and inject Angular's $scope
app.controller('mainController', function($scope) {
  // create a message to display in our view
  $scope.message = 'Main.';
});
app.controller('aboutController', function($scope) {
  $scope.message = 'About.';
});

```

Figure 5.14: AngularJS routing example

```

.state('visualization', {
  url: '/visualization',
  templateUrl: 'app/visualization/visualization.html',
  controller: 'VisualizationController'
})
.state('visualization.srpe', {
  parent: 'visualization',
  params: ['urn'],
  templateUrl: 'app/visualization/visualizationSrpe.html',
  controller: 'VisualizationSrpeController'
})
.state('visualization.srpe.rpe', {
  parent: 'visualization.srpe',
  params: ['urn'],
  templateUrl: 'app/visualization/visualizationSrpeRpe.html',
  controller: 'VisualizationSrpeRpeController'
})

```

Figure 5.15: AngularUI states example

```

<!-- route to visualization -->
<button type="button" ui-sref="visualization">Go to visualization!</button>
<!-- route to visualization.srpe -->
<button type="button" ui-sref=".srpe">See srpe data!</button>
<!-- route to visualization.srpe.rpe -->
<button type="button" ui-sref=".rpe">See srpe data!</button>

```

Figure 5.16: AngularUI routing with states

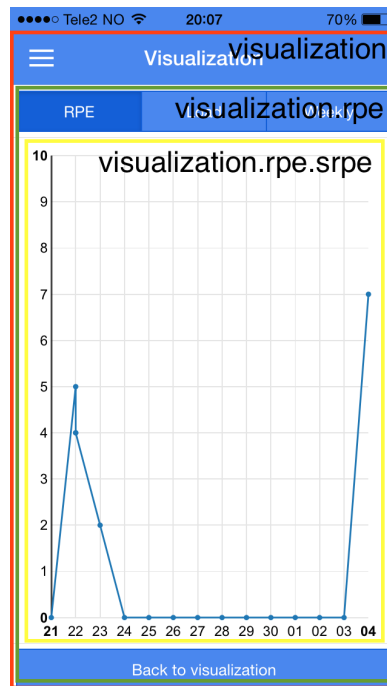


Figure 5.17: Illustration of nested view in pmSys

5.3.5 Design summary

With the hybrid-approach, pmSys will be able to support cross-platform deployment. Another advantage for this approach is that we have good knowledge in the web-technologies that are used to develop such an application. We are essentially building a Single Page Web Application for mobile devices, and by doing so we can create an application with a native feel since the interaction between user and application is more fluid than the traditional start and wait approach in multi-page applications. And by having a application container to access the device APIs, we can create a mobile application that has all the features a native application has. With AngularJS, we bind it all together to form the application.

5.4 Implemented modules

The pmSys-app is split up into smaller modules. Every module has its own feature and serves different purposes in the application. We will present the most important modules in-depth, while the miscellaneous features are mentioned.

5.4.1 Login

In Figure 5.18(a), we present the login section of pmSys-app, which is the module for authentication of the user. The user provides their username and password, and is authenticated through the Ohmage back-end. In pmSys-app we have implemented our version of the Ohmage server as the default server. Custom servers are also supported, and users can easily connect to other servers if needed. Usually this option is not necessary, but if the pmSys production servers go down, the user can manually connect to a another server. The user has to choose the "custom server" selection and write the URL of the specific server, as shown in Figure 5.18(b). Ohmage supports two types of authentication, namely stateless and stateful authentication. The authentication process is done through a POST-request to the Ohmage back-end with the login credentials the user has provided.

Stateless Authentication

Stateless authentication allows for a one-time authentication process. PmSys-app uses this authentication, because it will not require the user to authenticate him or herself every time they use the application.

(a) Login section

(b) Custom server selection

Figure 5.18: pmSys: Login module

When sending the POST-request to the server, the server responds with a hashed password that can be used later, to perform other CRUD-operations towards the server. The stateless authentication has one limitation; the hashed password cannot perform CRUD-operations to the entity "class" (See Section 4.4) on the Ohmage back-end.

Stateful Authentication

Stateful authentication generates a one-time token, and the default duration of this token is 15 minutes. This authentication-type is therefore not viable for pmSys-app, because it will require the user to refresh the token when it expires. If the user has been idle for more than 15 minutes, re-authentication is required. Stateful authentication is only used once during the first-time login to obtain the authentication token. This token is needed once, in order to request information about the "class" the logged in user is a part of. This process is done as the user authenticates.

5.4.2 Reporting

The report module is the reporting tool for the users in the pmSys-app. When a user has logged in successfully, they will be greeted with a list of campaigns. The application routes the user directly to the report module because reporting is the main objective for the players when using the application. As mentioned earlier, we want to develop the application with focus on optimizing the reporting process, and by avoiding the number of user interactions, the player can report quickly and focus on playing football.

A survey is defined in a campaign on the Ohmage back-end. The campaign is created through an XML-configuration shown in Section 4.5). This configuration is uploaded to the back-end, and can be accessed through the web-APIs of Ohmage. PmSys-app requests all the campaigns that are available for the user through a POST HTTP-request. The majority of the users will only have one campaign on their list, since they only belong to one team. In the future though, we might see players participating in two campaigns, one campaign for their team and some will also participate in the campaign for the National Team. Depending on which campaigns the player is reporting for, he or she can select the appropriate campaign and get a list of available surveys that is defined in the campaign. In Figure 5.19, the campaigns are listed for the user, and when a campaign is selected, a list of available surveys is shown.

The same XML-configuration that is uploaded to the server is returned when the application requests

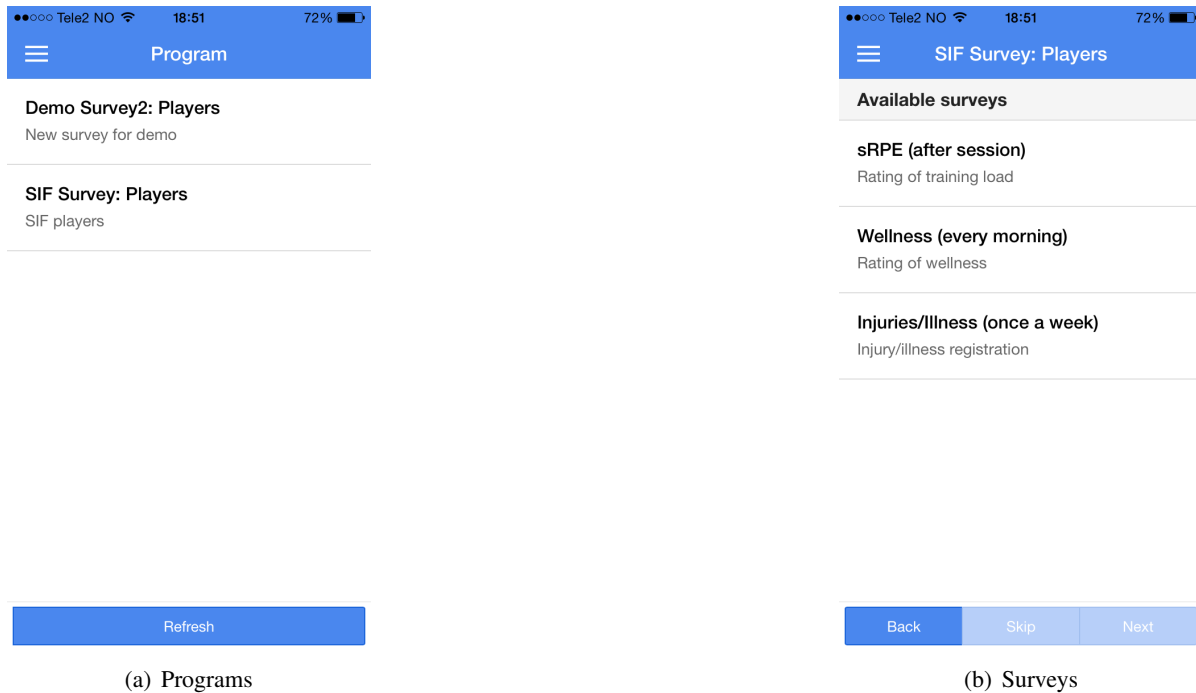


Figure 5.19: pmSys: Reporting module

the server for the campaigns. This XML-configuration is then parsed to scope out the necessary data before it is rendered for the UI-view for the player. The parsing of the XML-file is done through a framework called X2JS [43], which converts XML to JavaScript objects. This allows the application to work with the surveys as JSON objects. In Figure 5.22 an example of a JSON-object that have been parsed from the XML-file is shown. Based on the JSON-object, the surveys is parsed and presented as a questionnaire for the player in the UI. In Figure 5.20, parts of the questionnaire are shown. PmSys-app supports the following prompt-types from the Ohmage model: Number, timestamp and single choice.

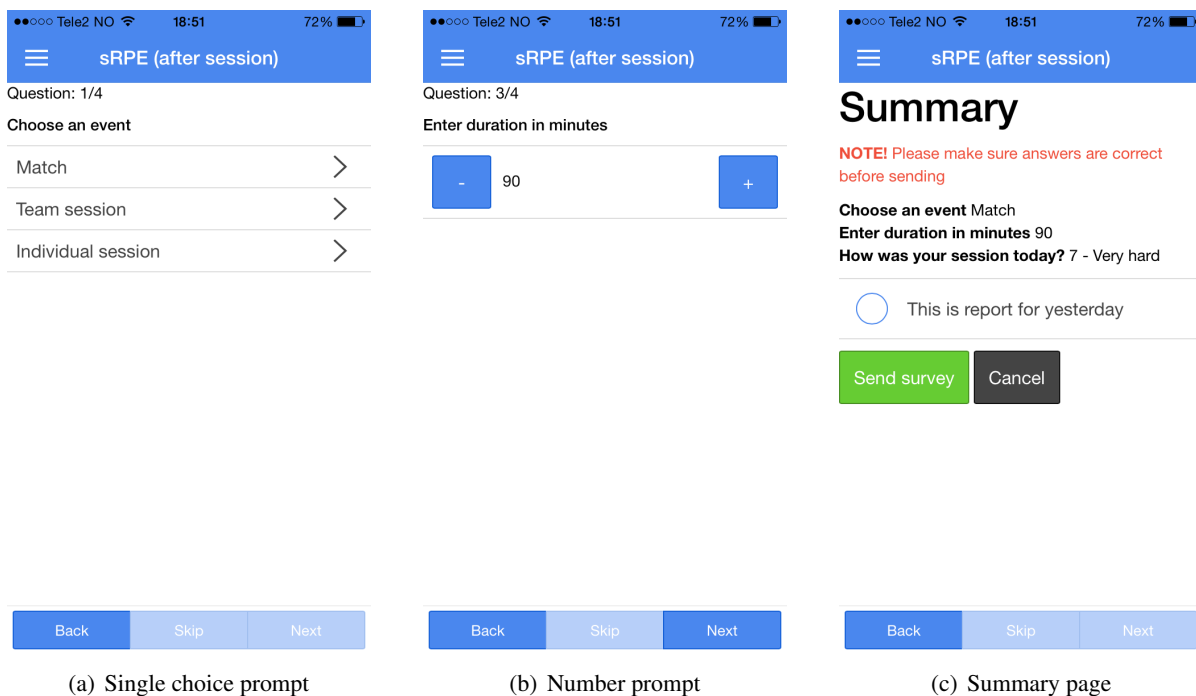


Figure 5.20: pmSys: Example of questionnaire

As the player answer the questionnaire, the application creates a dynamic JSON-object with a table

of the responses from the user. This allows the user to change the responses before submission. The application will automatically look for the right prompt-id and overwrite the old response. When the player submits the response, the application completes the JSON-file in the required Ohmage format, and sends the report as a POST HTTP-request to the Ohmage server. In Figure 5.21, we show a short example of the response data from a player.

```
{
  "prompt_id": "srpeLength",
  "value": 90
},
{
  "prompt_id": "srpeWorkload",
  "value": 1
}
```

Figure 5.21: Example: Survey response in a JSON-object

```
{
  "campaign": {
    "campaignUrn": "urn:campaign:demo:eng:srpecoach",
    "campaignName": "Surveys for coaches-demo-eng",
    "surveys": {
      "survey": {
        "id": "plansrpe",
        "title": "Planned srPE",
        "description": "Plan srPE",
        "submitText": "Survey is done. Thank you!",
        "showSummary": "true",
        "editSummary": "false",
        "summaryText": "Results",
        "anytime": "true",
        "contentList": {
          "prompt": {
            "id": "srpeType",
            "displayLabel": "Session",
            "promptText": "Type of session",
            "promptType": "single_choice",
            "properties": {
              "property": [
                {
                  "key": "0",
                  "label": "Football session",
                  "value": "0"
                },
                {
                  "key": "1",
                  "label": "Endurance training",
                  "value": "1"
                }
              ]
            }
          }
        }
      }
    }
  }
}
```

Figure 5.22: Example of a XML-configuration in JSON-format

To enable the players to go through the reporting process as fast as possible, pmSys has a UI that is fast and responsive. The questionnaires are set to be answered over the course of a season, and it is a repetitive process. Therefore, the players will most likely start to memorize the surveys and be able to

answer them very quickly as long as the application allows them to. When going through a questionnaire as shown in Figure 5.20, clicking once on the chosen alternative completes single choice prompts. The prompt types, number and timestamp requires the user to select the right value before advancing to the next question, but very few questions are of this prompt-type. The prompt type "number" is also optimized by allowing the user to increment and decrement the value by a set interval. We have also included a summary page. This will help the player look through their selected answers before submitting, increasing the quality of the data (See Figure 5.20(c)). We have also added a requested feature that allows the user to submit reports that have been forgotten. If the user has marked that the report is for yesterday, as shown in Figure 5.20(c)), that counts as a report for the day before.

5.4.3 Visualization

Visualization is the motivational feature that pmSys has. It provides the players with visualization, which will give them feedback over a specific time windows instead of just raw response data. With the graphs, the players can easily follow their development and trends to see how they are progressing over a period of time. The visualization module consist of three implementation phases: Request, parse and presentation.

We have implemented a variation of graphs for each of the surveys that is implemented in the system. To create the plots we have used a JavaScript visualization library, NVD3. NVD3 is a library that consists of re-usable charts [44] that is built on the low-level JavaScript library D3.js (Data Driven Documents) ¹. D3.js can create very complicated and sophisticated visualizations, but the learning curve is extremely high. For pmSys, the plots are less complicated, and we can use NVD3s reusable charts to create the needed visualizations.

Request data

In order to request the data from a player, the application perform POST HTTP-request to the Ohmage back-end. Before the request is done, we specify a query that contains parameters that will return a JSON-object containing the data we have specified. In the following list, we explain the most important parameters:

- Campaign URN - the campaign URN of the campaign the player is participating in.
- Column list - a list of what data to acquire from the server. Ohmage supports a variety of data, but pmSys will only gather the username, timestamp and the response data for the visualizations
- Start and end date - specify a time window for the responses.
- Prompt list - this list specifies what prompts that should be included in the response data. This list is a comma-separated list of prompt-ids.

The Ohmage server will automatically get the response data according to the parameters sent, and return it as a JSON-object. With access to the raw-data of the responses, the application has to parse the data to a fitting format for the visualization.

Data parsing

The raw data returned by the server is in a format that is not supported by the visualization library, NVD3. Ohmage will for each prompt create an array that stores the responses in an array. The timestamps and usernames is also managed by an array. That means that all the arrays are connected, and the timestamp in index 0, corresponds to the response in index 0 of all the prompts arrays, as well as the first element in the username array. Figure 5.23, illustrates the connection between the arrays and the elements in the array.

NVD3 is not able to interpret multiple arrays, and require us to parse the raw data into a supported format. In Figure 5.4.3, we show example data for a graph. Each graph contains an array of data-points.

¹www.d3js.org

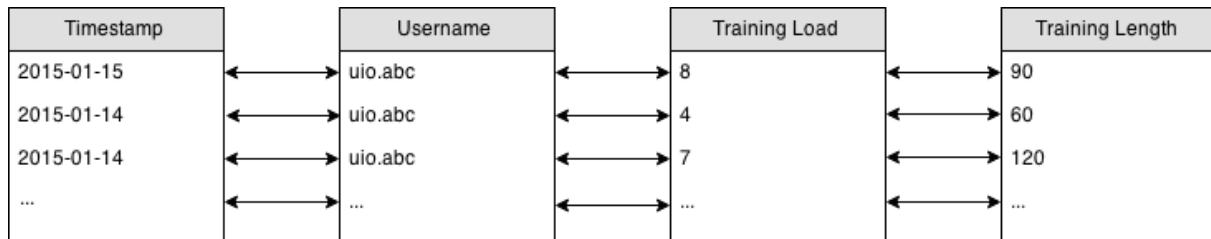


Figure 5.23: Visualization: Ohmage response data format

Each data-point is defined through a 2-dimensional array that will specify the x- and y- value of the data-point.

```
dataPoints = [ [timestamp, response], [timestamp2, response2],
               [timestamp3, response3], ... ]
```

Figure 5.24: Example: Array of data for plot

Another important parsing implementation is the date parsing. Ohmage is returning the timestamps in string format, but NVD3 requires the dates to be formatted as JavaScript date objects. Therefore, before we can push the timestamps into the data-point array, we have to convert the string over to a Date object. In Figure 5.4.3, we show a short code sample for date parsing. First, we have to manipulate the original Ohmage timestamp string to only contain the date. In the application, we only display the date of the report, therefore the time is not needed. With a library, called moment.js [45] we parse the string to a Date object by sending two parameters: the string we want to parse and the format (YYYY-MM-DD).

```
dateString = "2015-01-01 18:29:30"; //this is the string format Ohmage has
    on timestamps
dateString.substring(0, 10); // this returns a string that contains
    2015-01-01

//using moment.js for easy parsing to JavaScript Date
moment.format(dateString, "YYYY-MM-DD"); //this is now a date object with
    date 2014-01-01
```

Figure 5.25: Example: String to date parsing

Data plotting

To plot the graph, NVD3 requires a container in the HTML view as a placeholder. The SVG figure is then rendered through JavaScript code and sent into the view. NVD3 supports several features on the plots that we can decide to include in the plot:

- Tooltip - when the user hovers over the data points, it will display detailed information.
- Legend - allows the user to show or hide data-sets from the plot.
- Axis manipulation - What to show on the X- and Y-axis.
- Type - NVD3 supports a variety of charts: Line, bar, scatter, combination charts and many more.

The features can be configured through parameters, thus making all the different plots easily configurable. We have implemented graphs based on each questionnaire, and the graphs that are featured in the pmSys-app are the same as the graphs the coaches will use to analyze their players

data. The main difference is that the players only get to see their own data. In Figure 5.26, we show examples of the graphs the player will use to be able to follow their own development, and in this way participate in monitoring, rather than just submitting reports with nothing in return. The RPE graph (See Figure 5.26(a)) will show the player how high exertion and training load they have went through the last month. The wellness graph in Figure 5.26(b) is displaying the wellness reports, both the latest, and over the last month. Lastly, the application will show a graph (See Figure 5.26(c)), indicating what injuries that player has had.

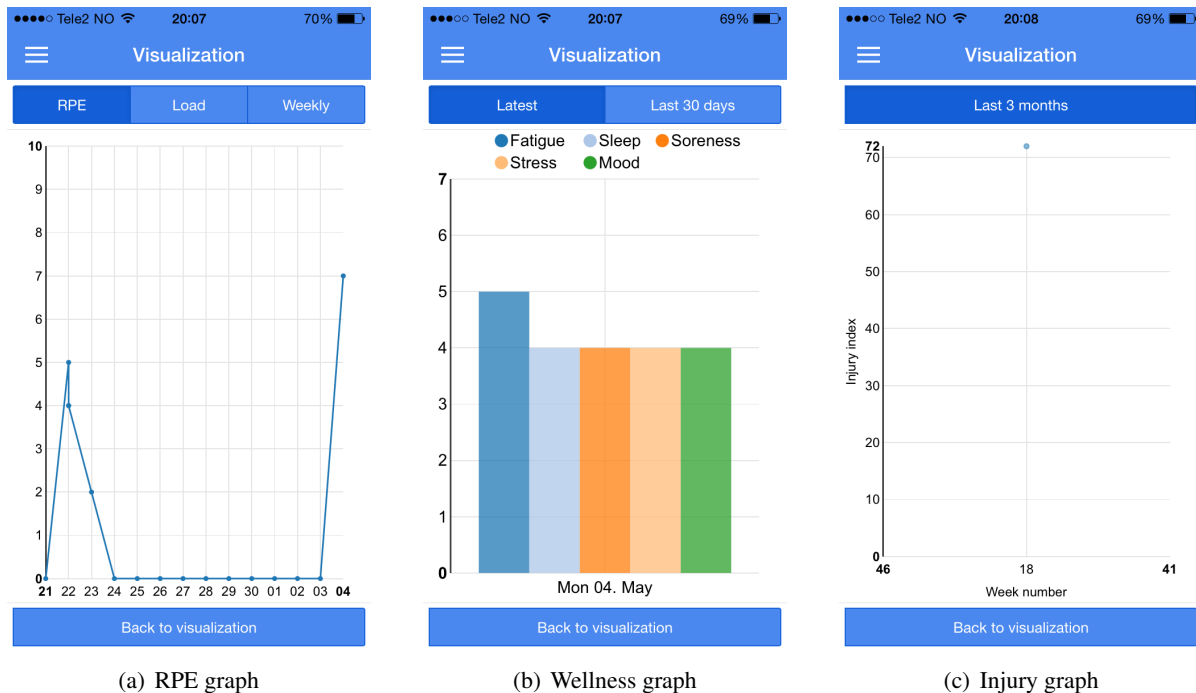


Figure 5.26: Visualization: Example of graphs in pmSys-app

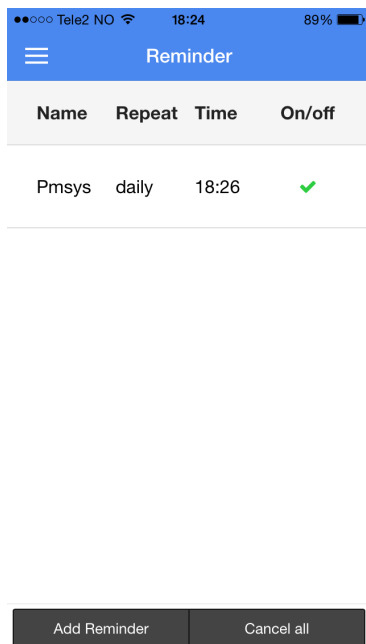
5.4.4 Notification

PmSys-app has two types of notifications, local and remote. The two provides the players notifications directly on the mobile device, but serve different purposes.

Local notification

Local notifications are important for players who have individual practice sessions in addition to the team sessions. PmSys's local notifications is based on the Cordova Local-Notification Plugin, which allows the application to access the device API which will trigger events on the phone depending on what event it is [46]. For pmSys's reminder, it is time-based events, which will allow the application to create scheduled notifications on the device.

To create a reminder on pmSys, the player must choose a title, the message to appear and the time of when to trigger the event. The module also supports repetition, and the player can choose between daily and weekly. The notification message appears for the players in the form of either displaying a banner, badging the icon of the application or playing a sound. The event will trigger even if the application is not running in the foreground, thus giving the player a reminder even if they are not using pmSys. Figure 5.27 shows an example of a local reminder that is going to trigger every day. The main challenge in this module is to store the notifications. The application has two containers that have to be synchronized. First, we have to store the reminder in the JavaScript container when the application is used, but the notifications must also be saved to file for storage, in case the application is closed, so it can be loaded up again when the application is restarted.



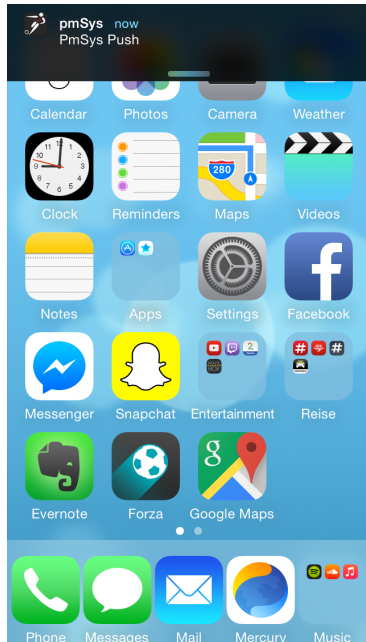
(a) List of reminders



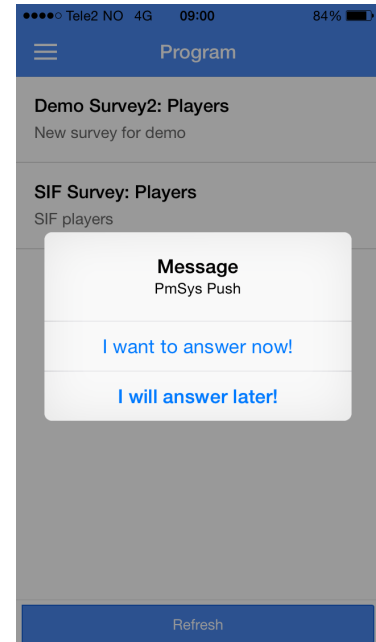
(b) Reminder triggered

Figure 5.27: Notifications: Local notifications

Remote notification



(a) Push notification



(b) Push notification in app

Figure 5.28: Notifications: Remote push-notification

The remote notification is important for reminders that involve the whole team. In pmSys-app, the remote notifications are received as push messages. These push messages are perfect for reminders for team sessions or other events that are repeatable. The difference from local notifications, are that remote notifications do not require the players to set up anything by themselves. They will receive push message sent by the coach, or an automated push-message that have been created by the coach. In order for the players to receive push message, they have to allow pmSys-app to send them push-message. If they

do, the moment they login with the application, they are automatically subscribed to the push-service, pmSys-push. PmSys-push is a system that allows the staff to send push-messages to their players on both platforms, Android and iOS. In Figure 5.28, we show an example of received push-message sent from the pmSys-trainer web-portal.

5.4.5 File system and Offline-mode

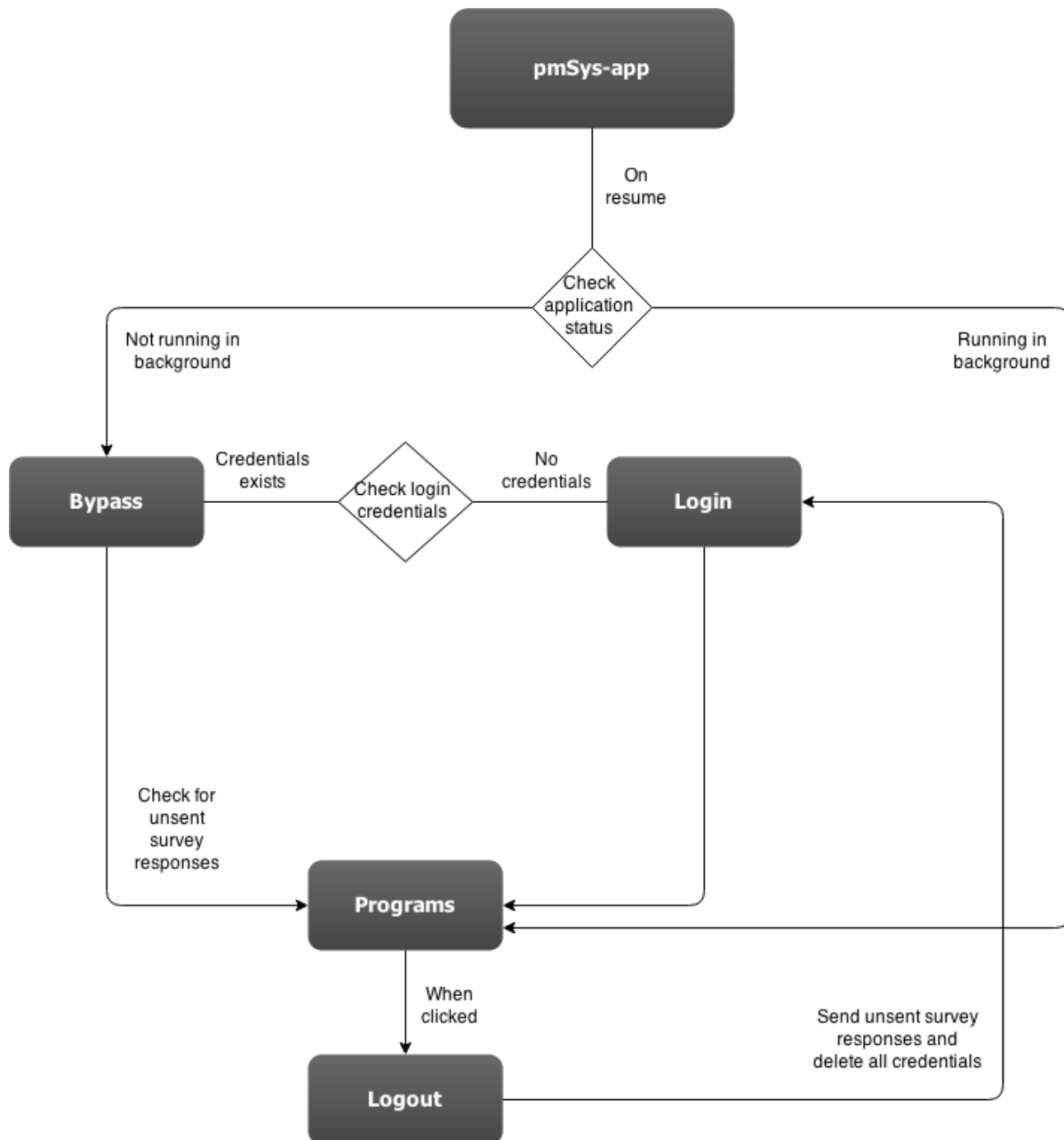


Figure 5.29: Illustration of the bypass module in pmSys

The file system module that comes with Cordova is essential for pmSys-app. In web-applications, it is possible to store everything in JavaScript containers, and in a hybrid application that is a possibility as well. The drawback is that it is normally stored in the memory. Every OS and device has a limited amount of memory, and when the system detects that it is about to run out of memory, it automatically shuts down applications that are running in the background. This will delete everything we have stored in the memory, including everything in the JavaScript containers. This creates the need for a storage place that is independent of the memory. Cordova gives us access to the file system of the device through JavaScript to create folders and files that belong to pmSys-app only. PmSys-app will create files to store

the necessary data to be able to function offline. This will allow the application to work without cellular networks or Wi-Fi, but the stored data is also necessary in case the application is shut down by the OS, or manually by the user.

The first operation the application does, is to create a folder that corresponds to the username of the player. A file is created and stored in that folder when the user is logging in, storing the necessary login information from the user. The information is of course validated with the information on the Ohmage server, and the application will only store login information once the player is authenticated. We also use the file system to store surveys, thus allowing the players to be able to report whenever they want without cellular networks or Wi-Fi available. The completed reports are also stored if the system detects that there is no available network, and will automatically upload all the reports once it detects that there is an available network available again. The automation of the survey upload is triggered every time the application is reopened.

The final feature regarding the file system is being able to bypass the login module once the player has logged in. As long as the application is not killed, the user is authenticated and will not be asked to log in again. As mentioned earlier, when the application is closed and killed by the OS, all the information stored in the memory is also gone. But with the stored login information in the file system, we can use that information and tell the application that a user has already logged in, and bypass the login. With this solution, the player will never be logged out unless they do it own their own. The moment they logout, the application un-authenticates the player and removes all the files that corresponds to that user from the file system. In Figure 5.29, we present the bypass solution in pmSys. Every time the user reopens the pmSys-app, it will check the status of the application. If it is still in memory and running in the background, the application will reopen and run as normal. If pmSys-app is killed by the OS and not in the background anymore, the application will automatically check the file system for available user credentials. If none are present, it will require the user to authenticate. If it finds the user credentials in the file system, it will bypass the login module and use the stored information.

5.4.6 Miscellaneous

Navigation

The user interface of pmSys consists of two main navigation paths. We have the side-menu that gives access to all the modules and features the application offer. We also have a dashboard that presents the same modules and features. To navigate in pmSys, the user will mainly use the side-menu, but the dashboard can also be used.

Dashboard

The dashboard module is the original startpage of pmSys-app where all the features and tools are presented to the user. The latest version of pmSys though, has the reporting module as the startpage. The reason for this is to allow the players to skip the dashboard, and go directly to the questionnaires to start reporting. This solution is a part of the focus on optimization of the reporting process. By allowing players to skip an interaction, they can in theory complete a report much quicker.

Glossary

The glossary is a URL to a webpage that contains the glossary for the pmSys surveys. PmSys is an application in English, but many players have Norwegian as their native language. Therefore, we have a glossary that gives detailed information of the implemented questionnaires in both English and Norwegian.

Profile

The profile module is a presentation of the player. It will provide some information about the logged in user, as well as whichever server they are logged into. It also provides the user the opportunity to change their password. The user will have to re-authenticate if they decide to change their password.

5.5 Deployment

Deploying the application on the app-market to make it available for the users was a long and complicated process. The challenge here is to synchronize the releases on the different platforms, because they have different requirements for publishing and updating the application. PmSys is currently available for iOS and Android.

5.5.1 Apples App Store - iOS

The deployment routine for iOS is a tedious process for first time developers. They have many detailed guidelines that developers have to follow [47]. For pmSys-app we needed to know how long the deployment process would take, to synchronize with the deployment for Android, but also plan a release plan for further versions of pmSys-app. Apple requires all applications to be signed with a valid Apple Developer account (100\$ per year), which will be held accountable for the application. Deployment of pmSys-app for the first time was, September 22th, 2014. It took Apple 14 days to review and approve our application. To our surprise, it was approved on our first attempt, and pmSys-app was available on App Store October 6th, 2014. From the experience we decided to have a window of two weeks from the day we submitted, to the day we announce a new release for the users. Because of this, every release had to be tested thoroughly before sending it in for review by Apple.

5.5.2 Google Play - Android

Google Play has a fully automated system for reviewing new mobile applications. It will carry out a system check on how the application code behaves in the OS, and does not require a time consuming review process that Apple has. This is an issue with Google Play, because they accept nearly all kinds of applications without actually looking into what the application provides. Google Play is easier for developers to deploy and update their application, but it also allows for many bogus applications. The time Google spent from we submitted it, and for it to be reviewed was around 15 minutes. However, it took 4 hours before we started to see it listed on Google Play.

5.6 Evaluation

In this section, we will evaluate the pmSys mobile application. As described in the problem definition, pmSys aims to replace pen and paper, as well as create a reporting tool tailored towards football players. We have looked into existing applications that have been used, and discovered a set of drawbacks that did not meet the requirements to ensure the quantity and quality of the data. With pmSys-app we have taken what a typical football player will encounter on a regular day basis into consideration, and developed a application that meets those requirements.

The majority of the time, pmSys-app is used to submit reports. Because of this, pmSys-app has been developed with the focus on optimizing the reporting process of the application. We have reduced the number of necessary user interactions to complete a report, resulting in a decrease of time used by several seconds compared to Ohmage. Looking at the results from the benchmarks, we see results in favor of pmSys on the time consumed on the reporting process. These benchmarks also showed us a bottleneck in pmSys-app, which is the loading screen. Because of our implementation of a 3-second delay when the application is opened, the results are fairly even when we compare the time it takes from the home-screen to a completed report (See Figure 5.30). When we exclude all other interactions, and focus only on the reporting module for both applications, pmSys edges out Ohmage by a good margin, both on a short questionnaire (See Figure 5.31), and on a longer questionnaire (See Figure 5.32).

Another important feature in pmSys-app reporting module is the addition of a summary page. In the Ohmage applications, the user had to either go back and check their answers manually, increasing the time used to complete a report. Alternatively, they can ignore the possibility of a wrong selection of an alternative in the survey, thus decreasing the data quality. With the summary page in pmSys-app the players can validate their answers before submitting, saving time and increasing the data quality.

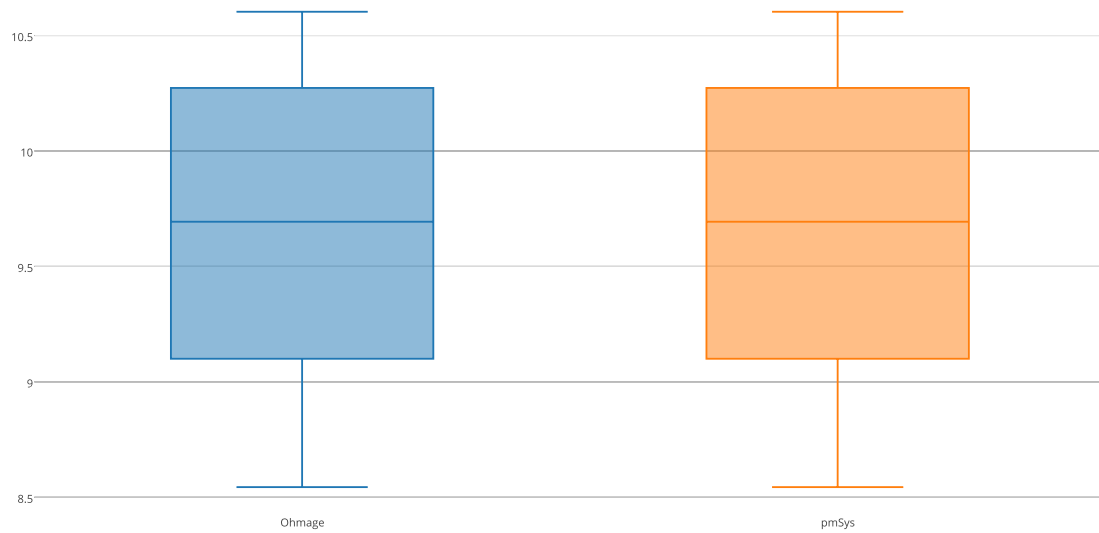


Figure 5.30: Benchmark: Full process from home screen to a completed survey (RPE questionnaire)

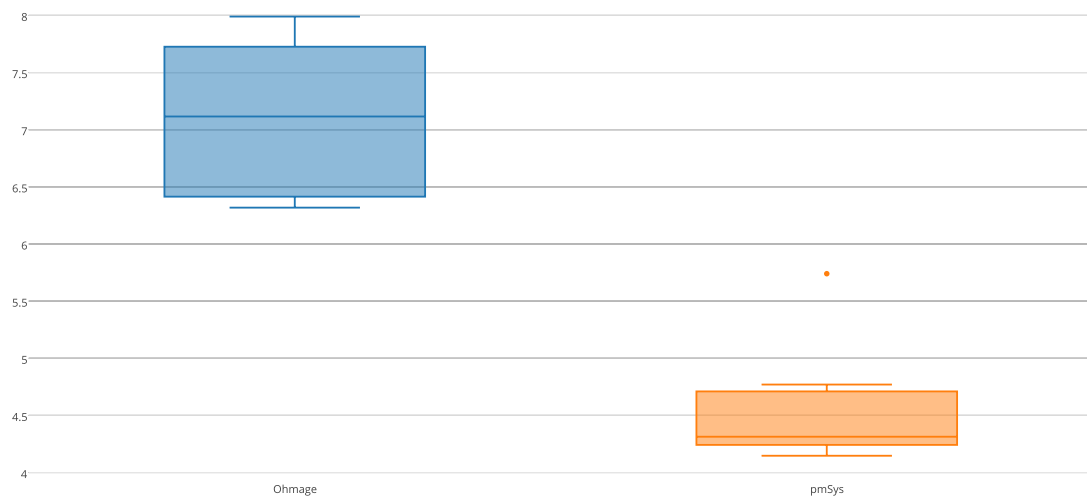


Figure 5.31: Benchmark: Only the reporting process (RPE questionnaire)

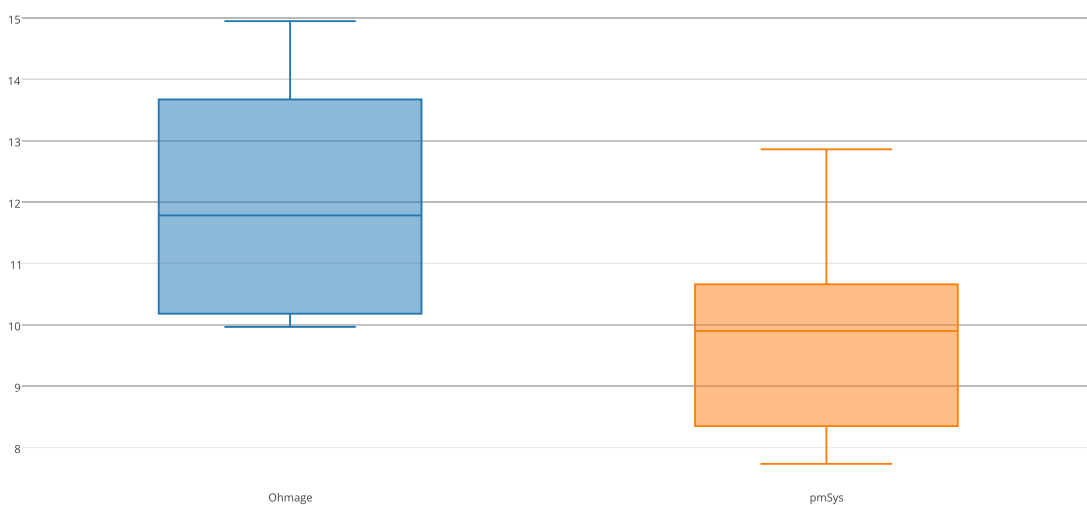


Figure 5.32: Benchmark: Only reporting process (Injury questionnaire)

A major issue with the existing systems is the lack of push-messages. Both Ohmage applications had local notifications in form of reminders, and in pmSys-app this is included as well. But based on feedback we have received, the main problem is that nobody is using this feature. This applies for both Ohmage and pmSys. This results in a very low response rate from the majority of the players, affecting the quantity of the data and creating many holes in the response data. With the lack of data, the quality of the analysis of the data is also flawed. With pmSys, we have included the support for push-messages through our own push system, pmSys-push. This can in theory replace the local reminders and can allow players to circumvent the need to configure reminders. The only drawback is that the application is dependent on internet to receive the push message. With pmSys-push, the players can receive individual messages from their coach, or a collective message for the whole team. This has increased the activity of the players, and helped them remember to submit their reports at the right time and at the right frequency as well. We have compared the response rate of last season with Ohmage, against the four first months with pmSys. The difference in activity is astonishing. In the 2014 season, we had three teams from Tippeliga that used the first version of pmSys with the Ohmage mobile applications. The numbers were fairly low in the first season, and only a total of 467 responses were registered for all the players. There were only 37 players across three teams who even bothered to participate in the reporting process. This gave us only 12.6 reports per player over the course of a whole season. Such a shortfall in responses is insufficient in order to properly monitor a player. On the other hand, with pmSys-app, we have registered a staggering 2593 responses from 27 players from only one team, and we are only halfway through 2015. We believe that push-messages are a driving factor in increasing the activity of the players, but it alone is not responsible for these impressive numbers. The improvements inside the application compared to Ohmage are also a major factor when evaluating pmSys-app.

Year	Number of players	Registered reponses in total	Average response per player	Mobile client
2014	37	467	12.6	Ohmage
2015	27	2593	96	

Table 5.2: Response statistics

To evaluate the usability of the application, we conducted a user-study on users who had no experience with pmSys. In this study, we had 27 independent users from University of Oslo, Oslo and Akershus University College of Applied Sciences and Norwegian Business School, but also some users outside who were not a students as well. We managed to gather a wide variety of people who had little to no experience with mobile development, and asked them to test Ohmage and pmSys. They then rated both applications, based on the usability on a scale from 1-5:

1. Poor
2. Ok
3. Good
4. Very good
5. Excellent

Based on the responses from the test-group, we can tell that pmSys is the general more popular application based on usability. We have presented the results in box-plots (See Figure 5.33, 5.34, 5.35 and 5.36), illustrating the response distribution of all the participants of the study. Overall, pmSys was the preferred application across the different usability aspects.

The test-group also had some positive comments about pmSys-app:

- *"Surveys of pmSys was better contructed. Easier to click and advance to the next question. Plus they had summary page. Nice with graphs and tracking."*
- *"PmSys was a easier and more detailed system to use. It looked modernized with easier features that makes it easier to use the application."*

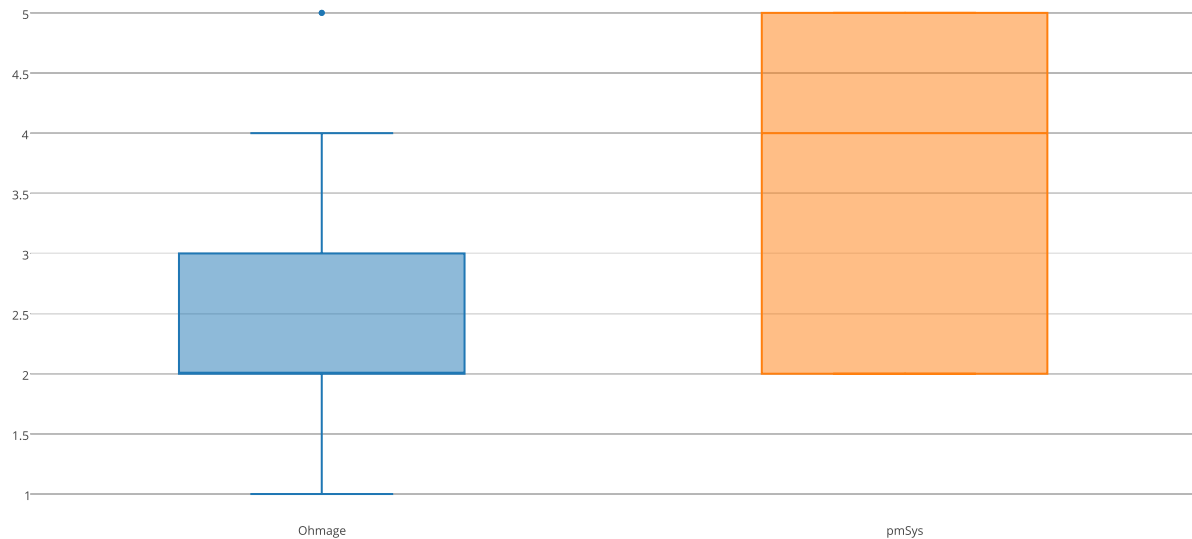


Figure 5.33: Userstudy: Usability

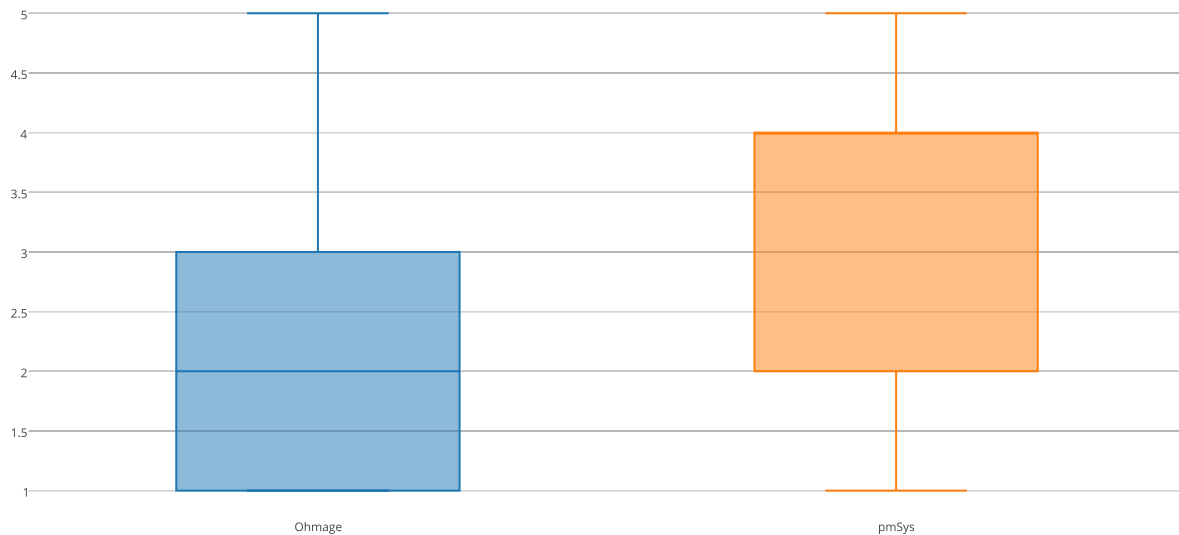


Figure 5.34: Userstudy: Design

- *"PmSys offers more functions, which is positive compared to Ohmage."*
- *"PmSys seemed more innovative then Ohmage."*
- *"I liked the usability of pmSys, which i belive is a important aspect of an application like this."*

Based on the comments and results, pmSys made a good first impression on the users compared to Ohmage. PmSys performed well as an intuitive application with high usability for users that have no past experience with pmSys. In addition, the new features in pmSys-app also left a good impression for the test-group. However, not every comment was positive, and we also received some comments that will help us improve the application:

- *"Give the axes (x,y) a name so the users do not get confused"*
- *"Ohmage looked better with the pictures, pmSys looked too simple"*
- *"Some explanation on what the values provides"*

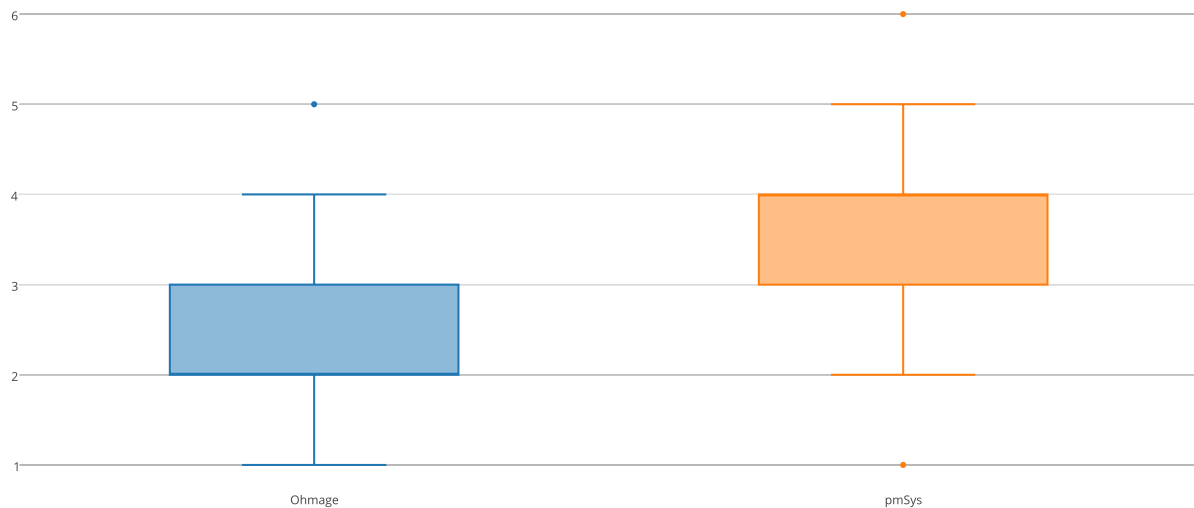


Figure 5.35: Userstudy: Navigation

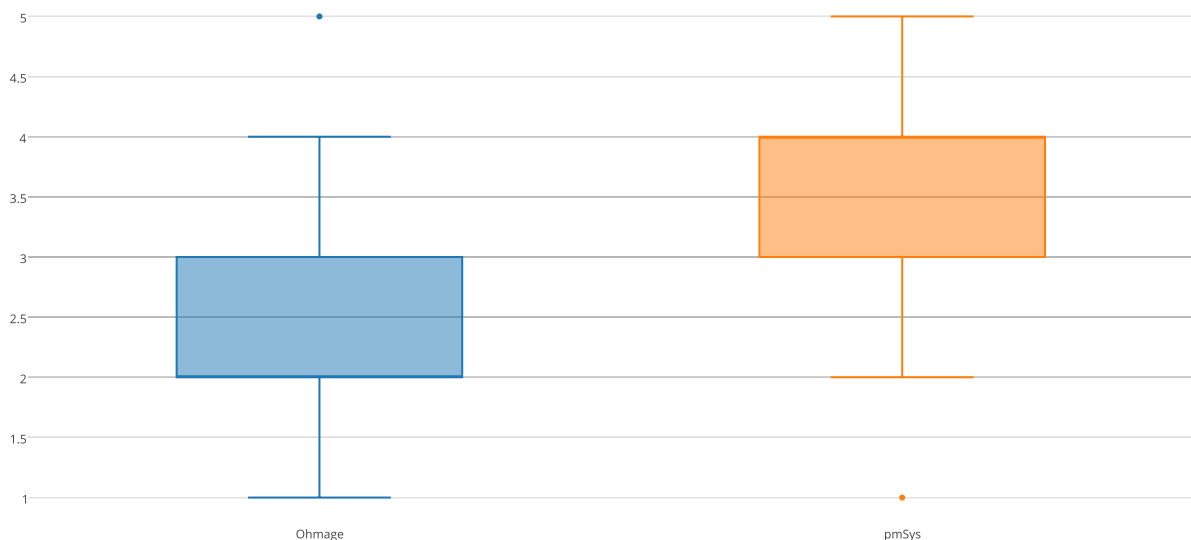


Figure 5.36: Userstudy: Presentation

The comment on the lack of pictures is based on the fact that in pmSys, the users is directly routed to the reporting module and skipping the dashboard. For a new user this can be confusing, since they will be met by a list that shows all the available campaigns. The default startpage is indeed the dashboard, but we have to take into consideration that we are creating a system that is tailor-made for football players who are going to report every day for a whole season, and the main functionality of the application is to report. The feedback on the graphs is something that we also can discuss. If we add too many details on the plot, the space where the actual graph is rendered becomes smaller. On smaller devices like mobile phones, we have limited space. We have to carefully choose what to display for the users, and we have chosen to focus on the actual graph and rather give the players an explanation through the glossary.

The last user-study we conducted was to gather feedback on how pmSys has functioned as a reporting tool for the football players that have been using pmSys so far. This will tell us, how well the pmSys-app is perceived among the actual users of the application. We asked the players to rate our system based on several parameters shown in Figure 5.37 from a scale 1-5.

Based on the results, the average rating was between good to very good. This is positive results, and we believe that the improvements we have implemented, together with the new features have come a long way to motivate and encourage the players to participate in the monitoring process. It is important to emphasize that there are still areas that pmSys can improve on, but we think that pmSys-app is on the

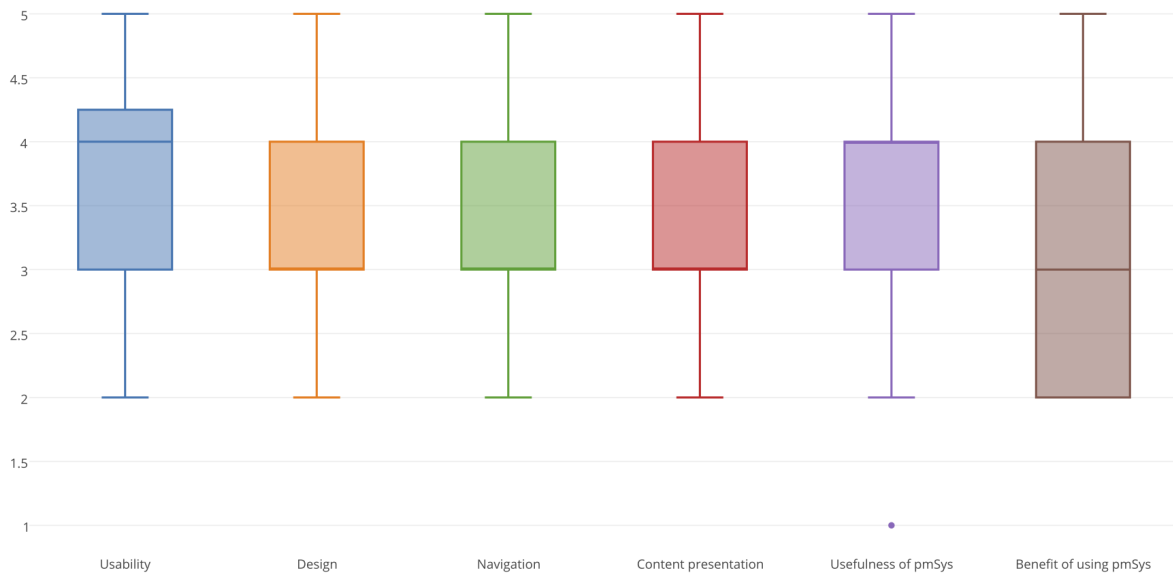


Figure 5.37: Userstudy: Rating of pmSys

right track. The comments from actual users of the application give us this confirmation:

- *"Very nice app. Saves me alot of time."*
- *"PmSys works phenomenally for me, and i am very satisfied with the positive impact it has had on me."*

5.7 Summary

In this chapter we have presented the data collecting tool for our system, pmSys-app. We have introduced the two mobile development approaches, the hybrid- and the SPA-approach, which have been used in the development of the mobile application. With our limited programming knowledge in the native languages of the mobile OS, and the limited time frame we had in the master thesis we believe that we made the best choice based on the circumstances we were in. The Hybrid-approach allowed us to support cross-platform deployment, but still be able to write the majority of the source code in well-known web standards as HTML, CSS and JavaScript. With the usage of different frameworks, we deployed the application as full worthy native applications for iOS and Android in their respective app stores. With SPA we gave our hybrid application a native feel, making it competitive against native applications.

With pmSys, the cumbersome format of pen and paper can be replaced. We also believe that the application is a big improvement from the existing Ohmage application when used specifically for football players with the purpose of monitoring them. We have been able to increase the activity of the football players through reminders and motivational features. The improvements and the new features presented in pmSys-app shows that pmSys is moving in the right direction, and the presented user-studies and benchmarks can confirm that.

Chapter 6

pmSys-trainer

In this chapter, we present the pmSys-trainer web-portal. The web-portal is a web application that will present the collected data in a way that will make sense for the coaching staff of a football team. In the surveys presented in Chapter 2, we have a strong foundation for collecting data in order to monitor the players. But in order to get useful information, the data has to be presented in a way that can assist the coaches and the medical staff in analyzing the data in an easy way, thus being able to monitor their players. The web-portal will also include tools to give the coach the opportunity to assist and supervise their players in the reporting process. We will first present the Ohmage Web FrontEnd that has been used in the first version of pmSys. We then present the technology and frameworks used to develop this web-portal. We will also present the push-service, which is an individual system, but is integrated in the web-portal so that coaches can send push-messages to their players.

6.1 Related work: Ohmage Web FrontEnd

For pmSys to be viable for a team, both the players and the coach need tools that are tailor-made for them. For players, we have evaluated the Ohmage mobile applications and developed the new pmSys-app, which was better suited for them. There are several limitations on a mobile device compared to a desktop device. With the mobile device you get a small screen that will limit the amount of information that can be viewed. In addition, there is also a battery and power limitations. To give a coach the proper tools with all the essential features to fully monitor their players, there is a need for a more powerful web application on a full-screen desktop.

In the first version of pmSys, we used the **Ohmage Web FrontEnd (Ohmage Web)** as both an administration and analysis tool. Ohmage Web is a web-application, and the main advantage is that it is platform independent and will work on all modern web-browsers. The application gives admins an user-friendly tool through an UI, to access the Ohmage model to execute different operations on the Ohmage objects. As an administrative front-end, it provided us, the developers with all the necessary features to configure the Ohmage back-end to suit pmSys. We will therefore evaluate Ohmage Web, strictly as a analysis tool for the purpose of monitoring football players, and present the drawbacks that lead to the need for a tailor-made web-portal for the coaches. This process has been done in collaboration with NiH, since they have used this tool for their PhD research on monitoring load and fatigue [4].

6.1.1 Presentation

The Ohmage Web is detailed and presents many features, but not all of them are needed by a coach. For a player monitoring system, we do not want the staff of the team to do the administrative operations such as: creating users and classes, define campaigns and surveys, configuring permissions and so on. It can make it complicated for the team, when they are presented with too many unnecessary features. That means that features that are not contributing to the monitoring aspect of the system should be excluded. In Figure 6.1, we present the dashboard that a coach will see when they login. As explained earlier, the front-end offers many tools and features. But if we look at it from a coach's perspective, few of these tools and features are relevant for them to actually monitor their players, assist them on the reporting process

or perform any analyses. The only features that are worth mentioning are the ability to read responses and explore the data through visualizations. But as we go into these modules, we will demonstrate why they do not meet the requirements for pmSys. We start by the ability to read responses.

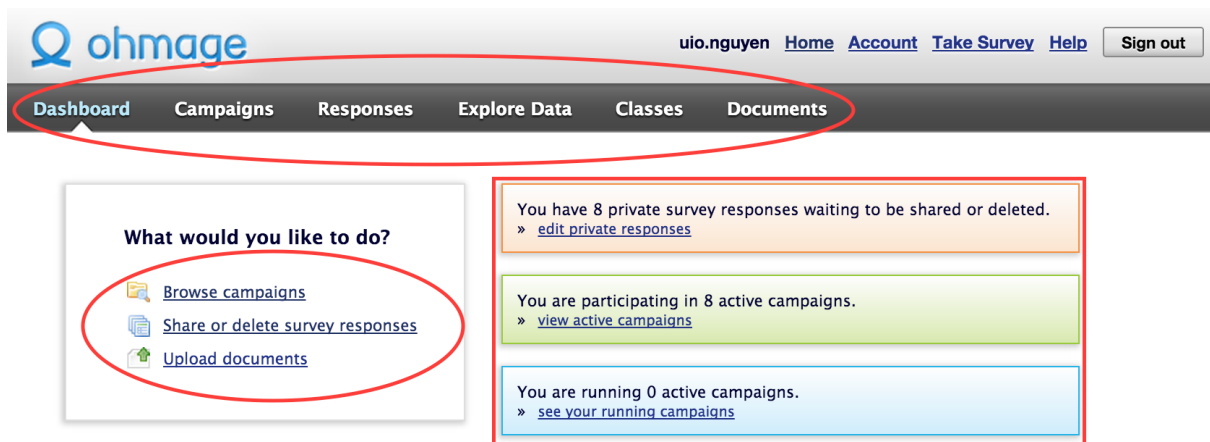


Figure 6.1: Ohmage Web: Dashboard

6.1.2 Survey response read

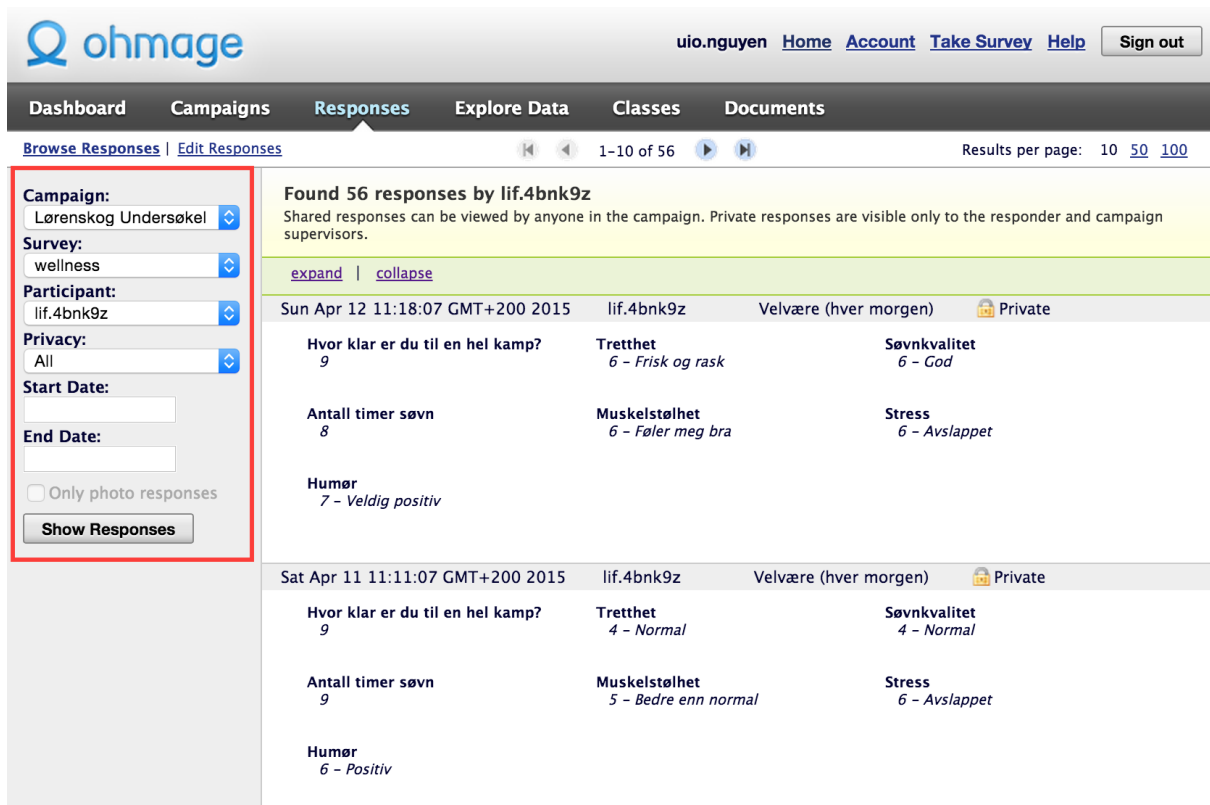


Figure 6.2: Ohmage Web: Survey response read example

In Figure 6.2 we introduce the response read module from Ohmage Web. This module will let the coaches read the responses from his players based on selected parameters. Besides looking at the raw

response data, Ohmage do not have any additional analysis or calculations on the data. For the coach to obtain relevant information for his whole team in order to monitor them, the system must offer more than just a presentation of raw response data.

6.1.3 Visualization module

The visualization module of Ohmage Web offers a diverse set of different plots and graphs. In Figure 6.3, we present the module, with the different plots it supports.

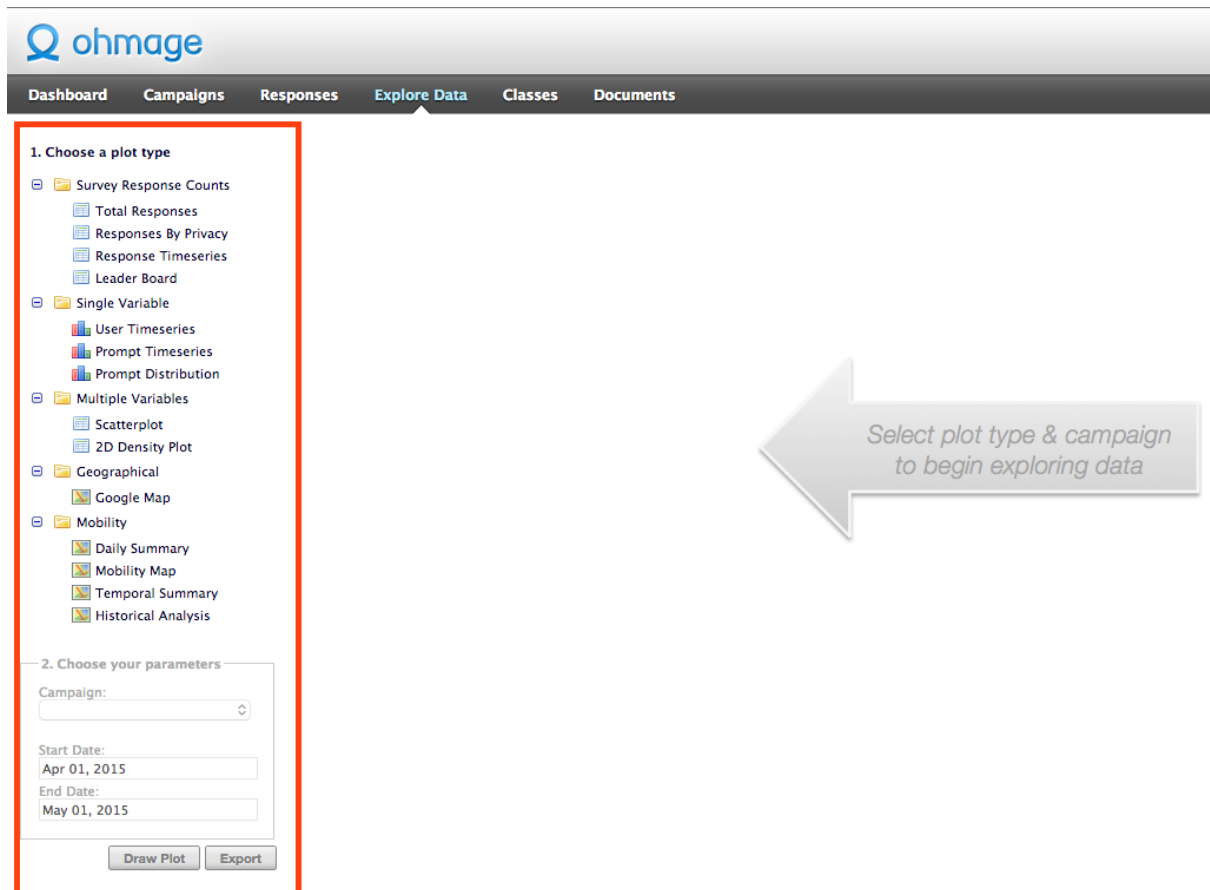


Figure 6.3: Ohmage Web: Visualization module

Survey Response Counts

This plot type will create graphs based on the response counts of a selected campaign (See Figure 6.4). The coach must select a specific time window, and the module can plot the following graphs:

- Total response count of the whole team for the selected campaign.
- Responses by privacy. Graph based on the privacy state, but since all the responses in pmSys is configured to have the privacy state "private", this graph will not provide anything useful.
- Response timeseries. Basically the same as total response count plot, but each survey has an individual graph showing the response count.
- Leader board will provide a list with numbers on the total responses, for each player.

These graphs are a good way to check the activity of the players and see which players are dedicated and makes an effort in the monitoring process, and spot those who are not.

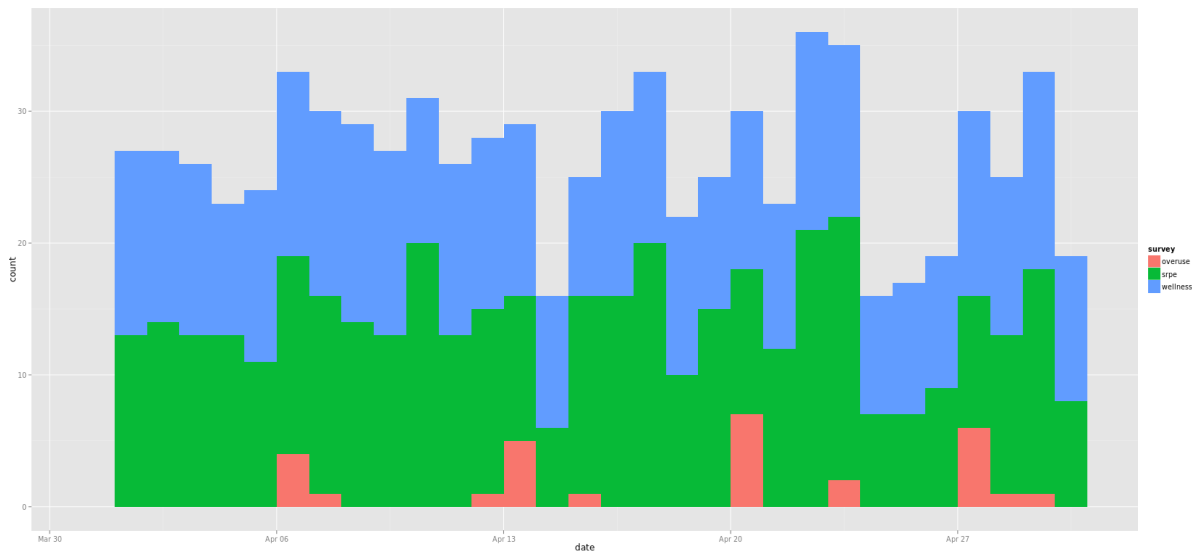


Figure 6.4: Ohmage Web: Total response count graph

Single Variable

This plot type is a more specific plot. The graph created is based on a single variable that is selected by the coach, (See Figure 6.5). The coach can select a single prompt, from any of the surveys to generate graphs. Following graphs can be presented for the coach:

- User timeseries for a selected player.
- Prompt timeseries for all the players.
- Prompt distribution.

In these visualizations, the coach can select either an individual player or the whole team to generate statistics of the responses. It provides the coach with statistics on which alternative in the surveys are selected the most and which are chosen the least.

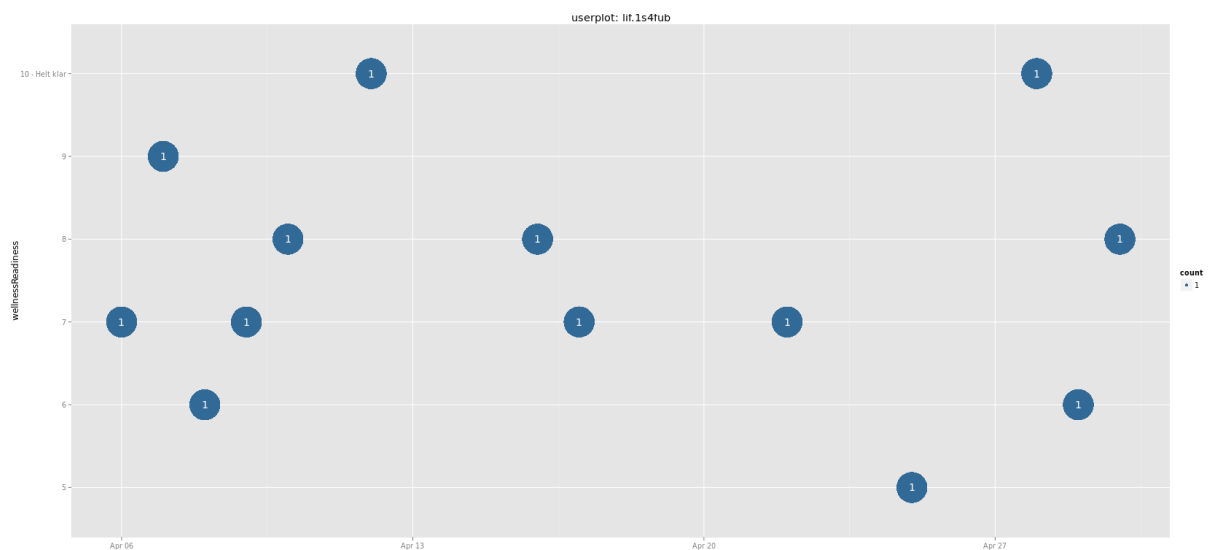


Figure 6.5: Ohmage Web: Single variable graph

Multiple variable

Ohmage has named the plot type for multiple variable, but it will only supports two variables. In this plot, the coach has the option of selecting the two prompt-id to create either a scatter-plot or a 2-dimensional Density plot, to compare the responses from the two prompts. In Figure 6.6, we show an example of a multiple variable plot.

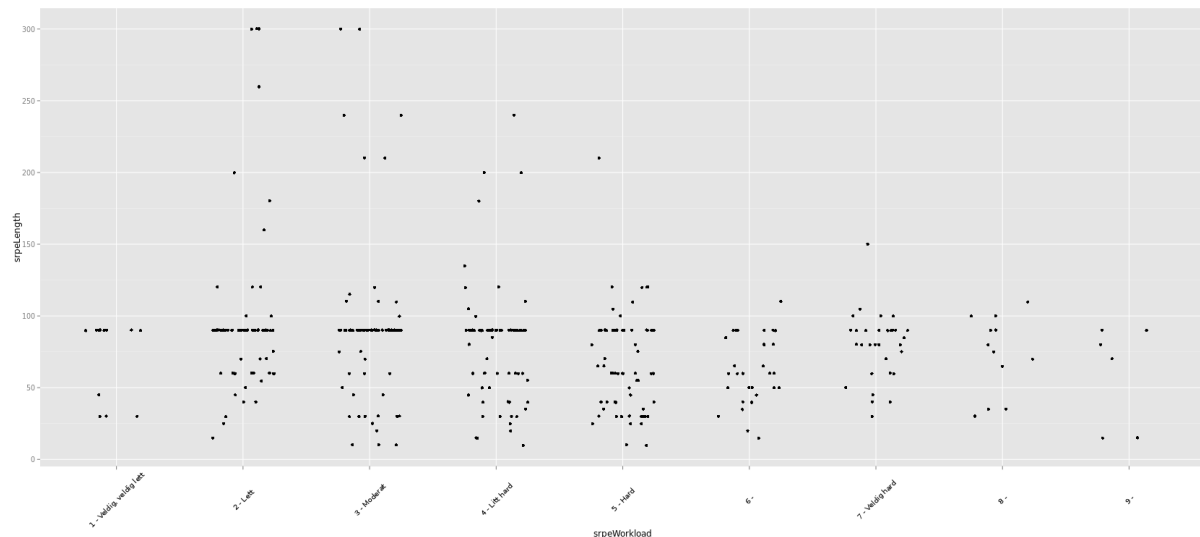


Figure 6.6: Ohmage Web: Multiple variable graph

Others

In addition to the mentioned plots, Ohmage also has the feature of integrating Google Maps and Mobility. But since pmSys does not register the GPS-location of the reports, this feature will not be useful. Mobility is a passive data collecting application, but in Section 3.5, we have already mentioned that this application is out of the market and unavailable.

Visualization module evaluation

The visualization module from Ohmage Web is a tool that brings us one step closer to the requirement set for pmSys. It offers a diverse set of different plots that certainly provides the coach with useful information. The main issue with the visualization module from Ohmage, is the focus on statistical plots. These plots can be useful, but for the monitoring aspect, providing only statistical data will not benefit the coaches in terms of monitoring the training load and fatigue. Ohmage does not have any graphs that will provide an overall view on how all the players are responding to matches or training sessions. There may be several questions that the coach will want to be able to analyze, but Ohmage is not capable of this. For example, after a hard training session, how many have reported high levels of fatigue, or that they are very sore? Ohmage is not capable of providing such information through their graphs. There are also limitations on doing further calculations and analysis on the data before plotting it, and the graphs are also limited to only 1 or 2 variables. Based on feedback from NiH, they moved away from this module because it did not provide them with the needed features. They had to manually export the raw data to compute further analysis of the data, and used third-party software to create the graphs and plots in order to obtain relevant information for the purpose of monitoring the football player. The Ohmage Web graphs are also static pictures, and are not interactive with the user, and will not allow the coach to highlight certain areas for more detailed information.

6.1.4 Manual work

Ohmage requires the user to manually perform many features that can normally be automated. By this we mean that certain features and user interactions can be automatized for a better user experience. This way, the coach can focus on the objective of monitoring the players. The response data from the players must be analyzed to bring more concrete answers than just raw numbers. In order to perform further analysis of the data, Ohmage supports exporting of the response data into the CSV-format¹. This exportable data can then be used in a third-party software like Microsoft Excel for some easy statistical computations, or R for an even more advanced analysis of the data.

This is a process that NiH have been doing in order to analyze the response data for their research. To provide the coaches with the data, they have to manually send the reports and analysis through e-mail. This is a very time consuming and cumbersome way of doing it, and it will not allow the coach to get the analysis in real-time. Another drawback is that the visualization module requires a lot of manual selection of parameters in order to render the plots. For generic use, this is an advantage since it then can be used with any survey that is configured in Ohmage. For pmSys we have defined a set of surveys that are the same for all the teams and players. This allows us to predefine which data to use in the plots and automatically visualize the data without requiring any unnecessary user interactions.

6.1.5 Tools for assistance

In order for pmSys to get full participation from the players, the coaches have to be active, help and motivate the players to submit their daily reports. Ohmage does not have any module for communication between player and coach. If a player feels that he is getting something in return from the reporting, it will motivate them to keep on reporting. A small message from the coach can be encouraging, but the motivational aspect is something Ohmage lacks. Furthermore, Ohmage does not have a dedicated tool to check the reporting status of all the players. It is a necessity for the coaches to be able to supervise their players in order to get the best result for both parties.

6.1.6 Ohmage Web Summary

As an administration tool, the Ohmage front end is powerful and offers us an easy GUI to execute operations towards the Ohmage back-end. But as a monitoring tool for the coaches, it does not meet any of the requirements set for pmSys. The presentation of the data is crucial, and presenting raw data from the responses will not provide the coach with the sufficient information that they need to make a proper decision. Because of all the missing features, Ohmage front end was never presented to the coaches as an analysis tool. It was mainly NiH that used it to manually export the data to analyze the it. These data was then presented to the coach on an infrequent basis. This led to the need for a tailored and dedicated application for the coaches where they could get live analysis presented when they needed it.

6.2 pmSys-push module

In this section, we present the middleware that creates a one-way communication channel between the coach and the players, pmSys-push. It is developed as a standalone system on the Node.js platform [26], but is integrated in the pmSys-trainer to provide the coaches with a user-friendly UI. Since we are developing a system for both the iOS and Android platform, we had to build a push module to support both platforms at once. The other alternative is using existing providers, like Amazon. It exist a plethora of push-providers with different prices and features. The common trait all these providers have, are that they have their own Software Development Kit (SDK), which have to be integrated into the application. The main drawback is that they only support native applications. With pmSys, we have support for all platforms since we provide a REST-API (See Figure 6.7) that is easy to use for all developers.

¹CSV - Comma-Separated Values http://en.wikipedia.org/wiki/Comma-separated_values

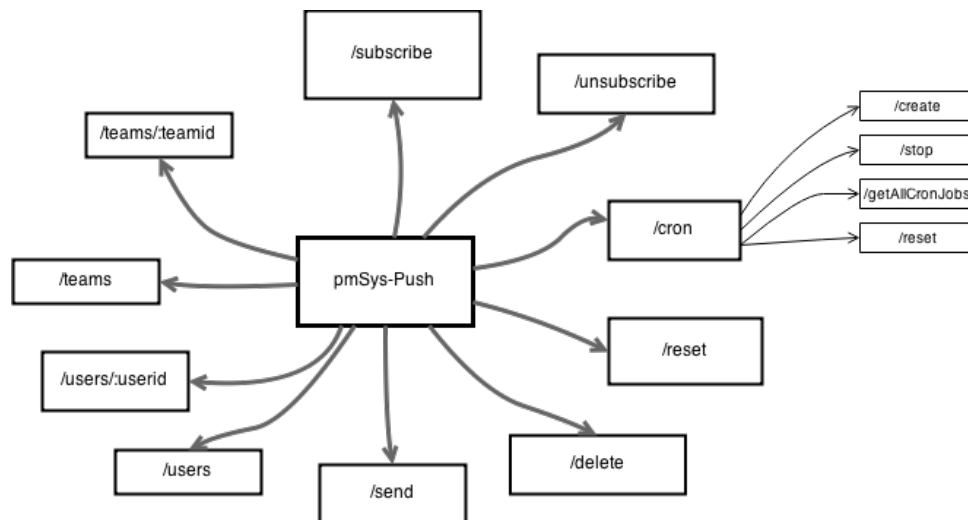


Figure 6.7: pmSys-push API endpoints

6.2.1 Features for pmSys-push

One of the main issues for the football teams during the monitoring process was that players forgot to report. Ohmage have reminders, but this had to be configured by the users manually. Another issue was that the reminders in the Ohmage MWF application did not function properly. The only alternative for the coaching staff to give the players reminders was through direct communication, sending SMS or call their players. With questionnaires that require them to report regularly and at the right time, it is important to have tools that can remind them automatically. To provide the coach with a tool to assist them in reminding the players, we have included the following features:

- Push messages for two platforms: iOS and Android
- Manual push messages. Select player(s) and message to send anytime. Useful for events that is sporadic.
- Automatic push messages. Coach can select date, time and occurrence for when push messages must be sent. Useful events that is repeated.
- Cold-boot. Cron-jobs for automatic push messages is stored and saved, and will not be removed even if the servers crash.

6.2.2 Apple Push Notification

Apple Push Notification (APN) service is the push service from Apple. It provides the developers the opportunity to implement push messages to their mobile application made for the operative system iOS. APN generates a push token for each device registered. APN has a throughput on 9.000 notifications per second, and for all iOS versions below 8.0 the allowed size of each message is 256 bytes. IOS 8.0+, the limit is increased to 2048 bytes [48].

To be able to send push messages to all the devices, the push server has to get a SSL Certificate issued by Apple. The certificates are limited to a single application based on the application's bundle ID. Apple has two development environments for a push-server, development and production. When the developers are testing the server, it will use the development environment and token will be generated based on that environment. The moment the application is uploaded to the app-store the server automatically switches to the production and generates tokens in production environment.

6.2.3 Google Cloud Messaging

Google Cloud Messaging (GCM) is Google's push messaging system that provides developers to be able to send messages to Android devices, as well as allowing the devices to send messages back to

the server [49]. In pmSys, we will only take advantage of the possibility to send push messages to the players. GCM, like APN will generate unique tokens for each registered device. GCM will also store a sender's key, that allows a push server with the key to push messages to the application. Important note is that the keys are unique generated to the device, and not the application. The senders key is registered to the applications bundle ID when a project is created through the Google Developer Console. Each project will be given a unique key.

6.2.4 Tokens

The tokens provided by APN or GCM is stored in a MySQL database. The main idea of using a MySQL database over any other solution is to be able to merge the existing Ohmage MySQL database with the pmSys-push database. In doing so, we will only have to maintain one database where the data is stored and secured in one place. In the database we store essential data to send push-messages:

- Username of the player
- Team the player belongs to
- The device token
- Platform
- Domain

In Figure 6.8, we illustrate the communication between pmSys-push and APN/GCM, and how the server handles the device tokens that are provided.

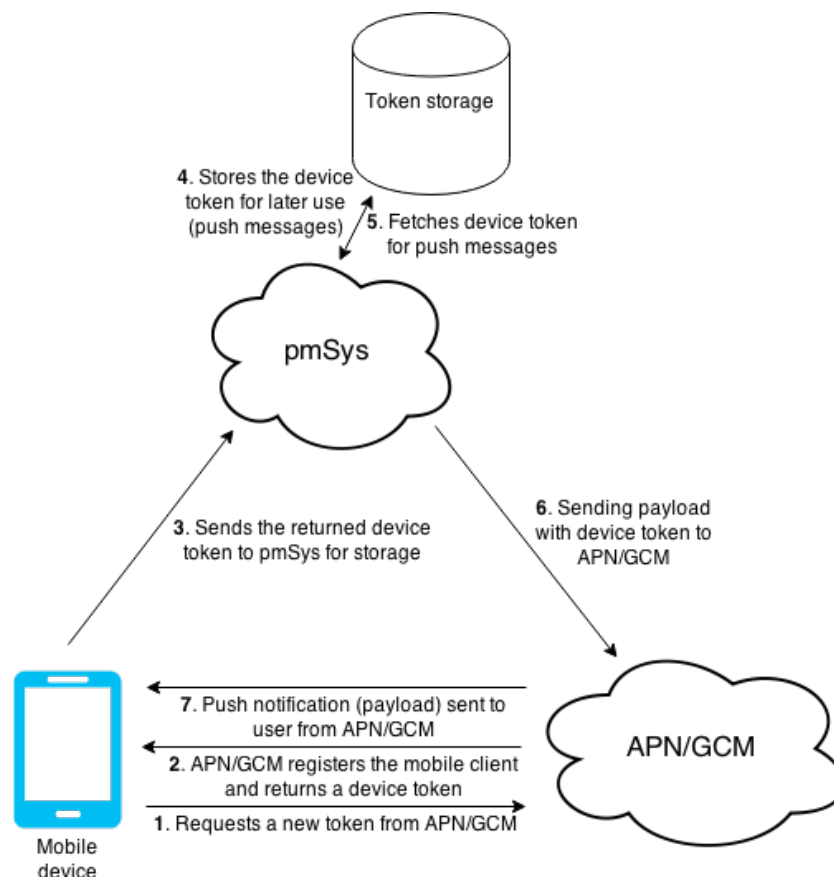


Figure 6.8: pmSys-push architecture

6.2.5 Cron

Cron is a system program that is running in the background, and is used to execute desired task at designated times [50]. In UNIX operating systems, the cron will be used to run certain shell commands at a specific time. For Node, we have used a framework called node-cron [51], which will have the same behavior as a cron in UNIX, only that it will execute JavaScript code in Node instead. Because the cron jobs are stored in memory, it will be deleted if the server shuts down. To mitigate this effect, we have implemented our own cold-boot function that stores all information about the cron jobs to file. If the server requires a reboot, the cron jobs will be recreated from file once the server has restarted.

In pmSys-push, the cron will be set to execute certain JavaScript functions that will send out push messages. This example code will show how to start a cron job:

```
cronTime: '00 00 00 * * *',
onTick: function() {
    //execute the job
    sendPush(target, message)
},
start: train_evaluation
});
```

6.3 pmSys-trainer features

The features that are implemented in pmSys-trainer is based on the feedback from NiH and the Norwegian National team, but also from the experience with Ohmage Web as an analysis tool for player monitoring. With pmSys-trainer web-portal, we have included features that will function as an analysis tool, but also tools that the coaches can use to help and assist their players in the reporting process.

- Survey response for each player
- Detailed visualization both individually and for the team.
- Statistics for reports
- List of players who have reported
- Communication with players individually through push-messages
- Automated push-messages for all players

6.4 System design

The system design of pmSys-trainer is built upon the idea introduced in Chapter 4 of building a own back-end with Node. As we started with Node we also became aware of a framework known as Express [52], that allowed us to build web-application on the Node platform. But as we changed out the NodeJS back end with the Ohmage server back-end, we kept the Node back-end to build a front-end solution on with Express that became the pmSys-trainer. In Figure 6.9, we show the architecture of pmSys-trainer.

6.4.1 NodeJS with Express

Node was introduced in Chapter 4, as an alternative to the traditional server solutions. Express can be seen as a light layer on top of the Node web server, and will make it more pleasant to develop web application on the Node platform. Express simplifies the Node API and adds helpful features for building web applications [53]. If we look at the Node web server, the communication between a client and the server without Express will look like in Figure 6.10.

The JavaScript function that handles incoming HTTP requests is called a request handler. This function takes the request, figures out what to do and responds. Node.js integrated HTTP server handles

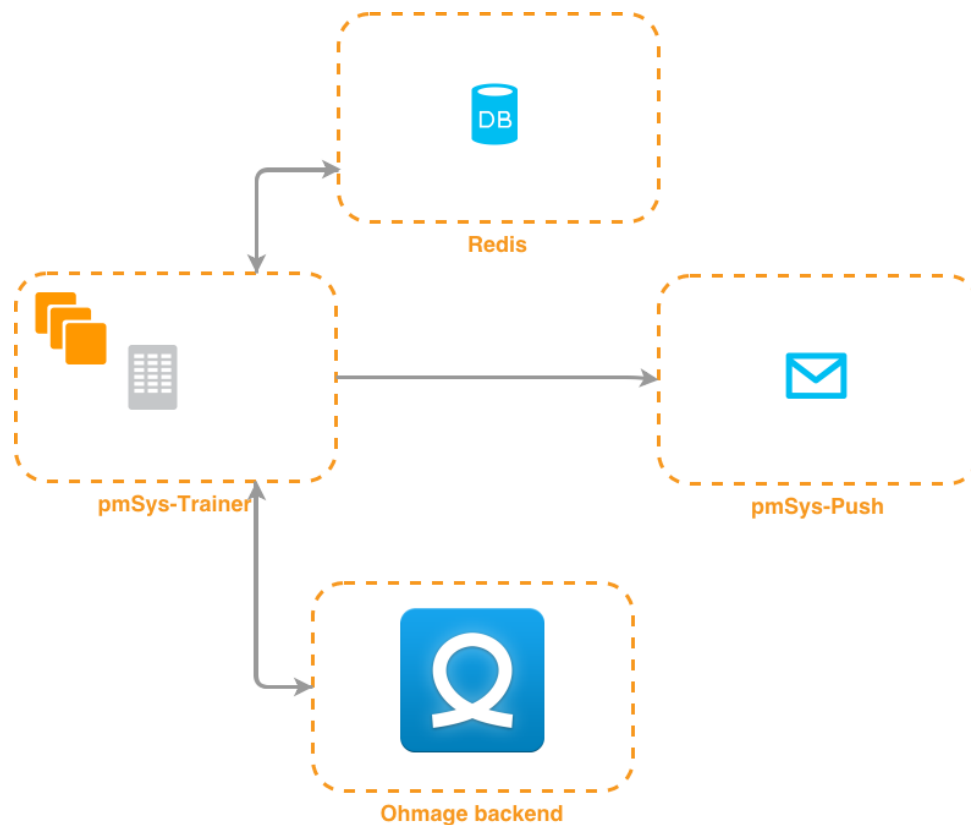


Figure 6.9: PmSys-trainer architecture

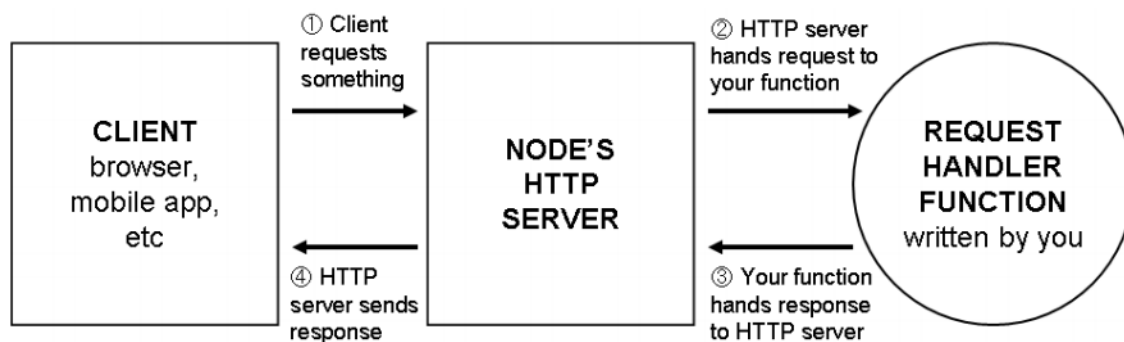


Figure 6.10: The flow of a request through a Node web application [53]

the connection between the client and the function so we do not need to handle tricky network protocols, allowing the developer to focus on implementing the handler. In JavaScript code, the handler is simply a function that takes in two arguments: an object that represents the request, and the object that represents the response. In our web-portal the request handler is used to check the URL that the client is requesting. When a client request the homepage, the handler responds by rendering the index.html in the browser. If the URL is not defined it should respond with a 404 error page. This is an easy concept, but with the native Node API it is complex. Luckily, with Express a lot of this complexity is abstracted away. An example is sending a JPEG file will be 45 lines in raw Node code, while in Express this is done in one line [53]. In Figure 6.11, we show an example on how to write a function that handles the request of the homepage, and the handler responds with the rendering of a index.html page in just a few lines of code.

If we look at Figure 6.12, Express adds another layer to the request flow. Before the request is sent to the handler, it is handed to Express that adds additional features to the request and response arguments.

In addition to routing with Express, we also take advantage of the template engine in order for the application to respond to requests with view templates. In order to use the template engine in Express, a

```
router.get('/', function(req, res) {
  res.render('index.html');
});
```

Figure 6.11: Code example of Express route handler

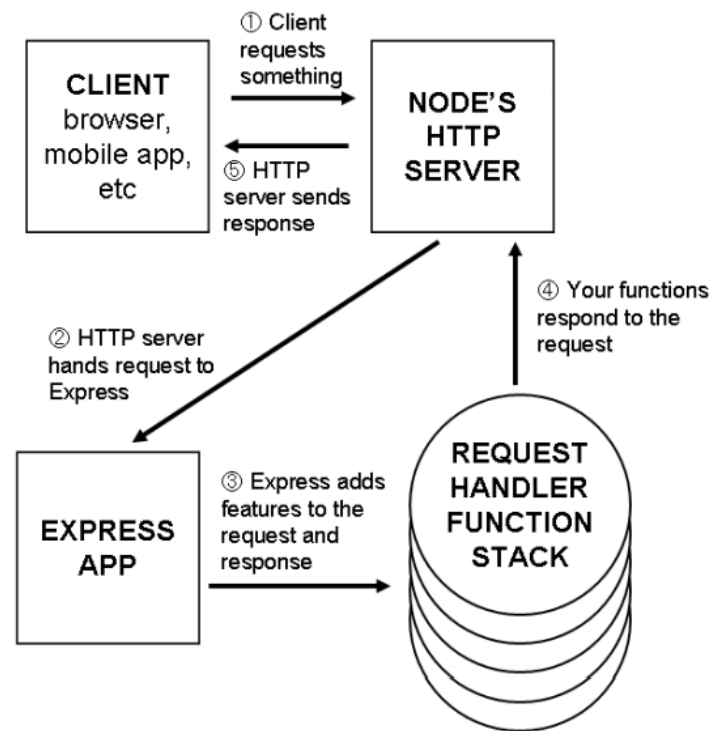


Figure 6.12: The flow of a request with Express [53]

view engine is needed. We decided to use Nunjucks because the syntax is similar to regular HTML, with extra functionalities that is easy to learn. These templates are the views that are rendered in Figure 6.11. The template engine also allows for inheritance of views, where we can define a main-view with many sub-views. With this, only the content that are written inside a block is changed as the user loads new views. In Figure 6.13, we have created a main-view. Then we have two sub-views that extends the main-view (See Figure 6.14 and Figure 6.15). The only content that is changed when a user navigates from the `index.html` to `about.html` is the code written inside the block. In Figure 6.16, we show how it will look in the browser.

```
<html>
  <body>
    {% block head %}

    {% endblock %}
  </body>
</html>
```

Figure 6.13: Main view: layout.html

```
{% extends "layout.html" %}
{% block body %}
  <h1>This is a subview</h1>
  <p>Hello!</p>
{% endblock %}
```

Figure 6.14: Sub view: index.html

```
{% extends "layout.html" %}
{% block body %}
  <h1>This is a subview</h1>
  <p>About us, we are pmSys</p>
{% endblock %}
```

Figure 6.15: Sub view: about.html

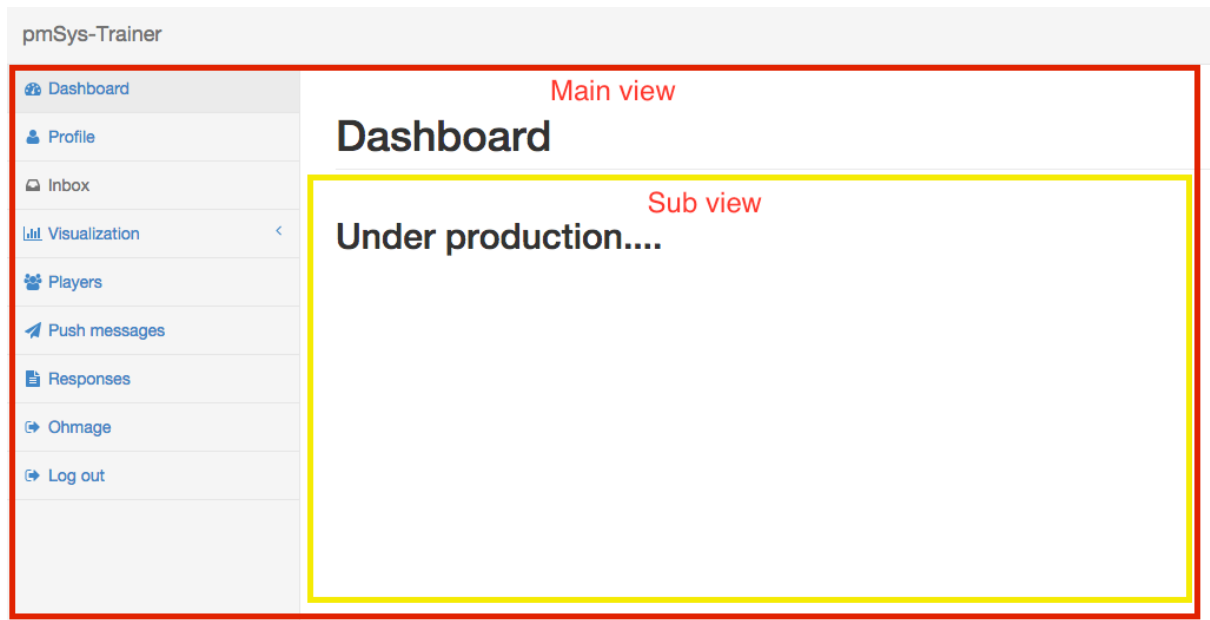


Figure 6.16: Example of Express view templates

6.4.2 Redis

Redis is an open source key-value data structure server. It is a NoSQL database that are much faster than other databases. NoSQL databases like MongoDB is a disk-based document store, while Redis uses RAM to access data much faster. Redis can store strings, hashes, lists, sets, sorted sets, bitmaps and hyperloglogs [54]. It can support a large number of concurrent clients, which make it scalable. It also cover the weaknesses of using RAM by having build-in persistence (snapshotting or journaling to disk), making it a real database instead of a temporary cache. As long as your server have enough RAM to support the amount of traffic, it is great. The pitfall in using Redis is that storing in memory is expensive compared to disk-based storing.

6.4.3 Bootstrap

Bootstrap is a popular HTML, CSS and JS framework for web developing. It was created by Twitter and has become one of the most popular front-end frameworks and open source projects in the world [55]. It easily and efficiently scales the web application to be used on all sorts of devices. It comes with integrated HTML and CSS components. PmSys-trainer uses a template called SB-Admin that is build

on the Bootstrap framework. It offers all the UI-components that we need, and because of the lack of design background, we decided to use this framework for the UI.

6.4.4 NPM

NPM is a package manager for Node [56]. It exist over 100.000 packages that is usable with Node. With NPM we can create a JSON-file where we can define the name, version and packages for our web application, simplifying the addition of different frameworks to our development.

6.5 Implementation

In this section, we present the implemented features based on the feedback. We will present the most important features in-depth.

6.5.1 Login

To authenticate a user, we send the username and password that they provide as a HTTP-request towards the Ohmage back-end. If the authentication is successful, the server responds with a hashed password that will be stored in the redis-server together with the session data. The login of pmSys-trainer also allows the coaches to select what Ohmage instance they will login to (See Figure 6.17). When the user is authenticated through the Ohmage back end, the server will responds with a hashed password that is stored in the redis-server as a session storage. This makes it possible for the coaches to use the web-portal without signing in each time.

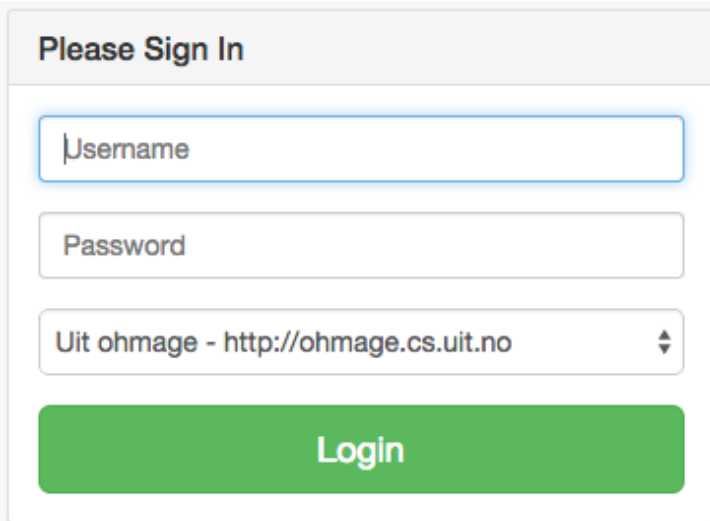
The image shows a web form for logging in. At the top, there is a header 'Please Sign In' in a bold, dark font. Below the header, there are three input fields stacked vertically. The first field is labeled 'Username' and has a light blue border. The second field is labeled 'Password' and has a light gray border. The third field is a dropdown menu with the text 'Uit ohmage - http://ohmage.cs.uit.no' and a small upward and downward arrow icon on the right. Below these three fields is a large, solid green button with the word 'Login' in white, bold, sans-serif font.

Figure 6.17: pmSys-trainer: Login

6.5.2 Push messaging

This module is the front-end UI to the pmSys-push system. Through the UI the staff of the team can send individual messages to either the whole team or selected players. They can also setup automated push-messages that can be configured by the coach. A list of players who have submitted reports is also included (See Figure 6.18). With the list, the coach can see which players that have forgot to make a report and send a push message to them, as shown in Figure 6.19.

This module also features automated push. The coach can define push messages to be automatically sent from the pmSys-push server, as shown in Figure 6.20. A list of automated push messages is shown in Figure 6.21 and the coach can delete and create new ones at any time.

Players	Select all	Wellness	sRPE
	<input type="checkbox"/>		
lif.p3e5ki	<input type="checkbox"/>		
lif.lctn5n	<input type="checkbox"/>	✓	
lif.ionk2m	<input type="checkbox"/>		
lif.ebylrr	<input type="checkbox"/>	✓	
lif.p9nzh	<input type="checkbox"/>		
lif.4bnk9z	<input type="checkbox"/>	✓	

Figure 6.18: pmSys-trainer: Listing of players who have reported

Type in message to send:

Please report RPE!

Send

Figure 6.19: pmSys-trainer: Manual push messaging to selected players

Automated push

Name

Name for push

Start time

Press to select

Choose days to run on:

☐ Sunday ☐ Monday ☐ Tuesday ☐ Wednesday ☐ Thursday ☐ Friday ☐ Saturday

Message

Write pushmessage

Close Add push

Figure 6.20: pmSys-trainer: Adding a automated push message

Automated pushmessages

+ Add

Name	Time	Repeat	Message	Created by	Remove
Skade	08:00	Monday	Husk å rapportere skade / sykdom i dag	lif.paal	Delete
VELVÆRE - hverdag	07:30	Monday,Tuesday,Wednesday,Thursday,Friday	Husk velvære rapportering	lif.paal	Delete
VELVÆRE - helg	09:00	Sunday,Saturday	Husk velvære rapportering	lif.paal	Delete
sRPE	21:00	Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday	Har du hatt noen treninger i dag som du ikke har rapportert..!?	lif.paal	Delete

Figure 6.21: pmSys-trainer: Listing of automated push messages

6.5.3 Survey responses

This module allows the coach to look at the response data from each player. Since the coach is defined as a privileged user on the Ohmage back end, they can access all the responses of the campaign they participate in. The coach will specify a date, and the application will show a list of players that have reported that day (Figure 6.22(a)). The web application will make a POST HTTP-request to the Ohmage back-end and request for the response data for the selected player. The back end will respond with a JSON-object containing the response, and will be parsed before it is presented to the coach in a readable format (Figure 6.22(b)).

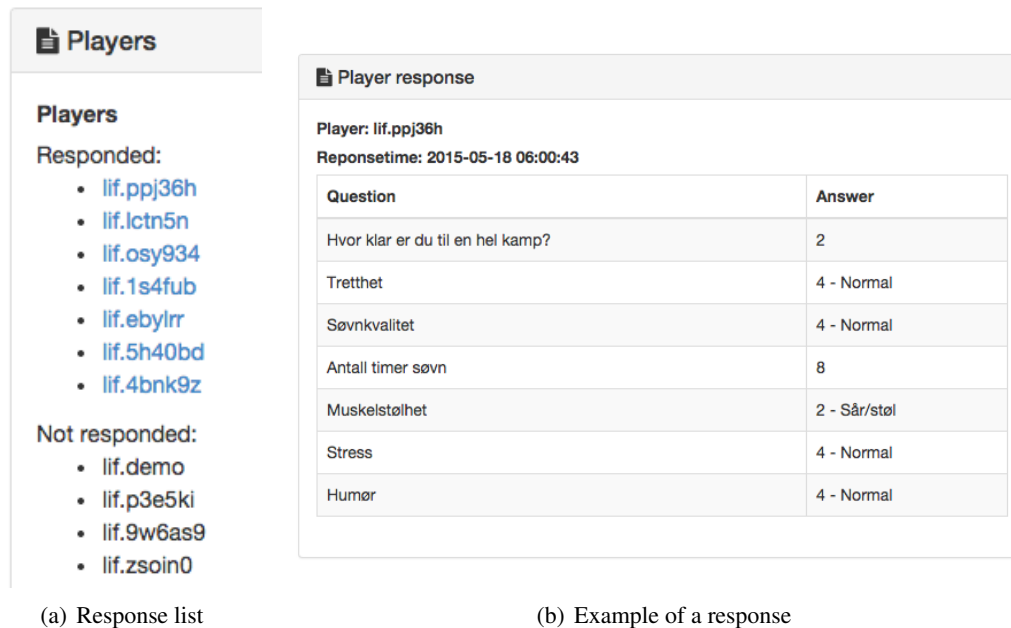


Figure 6.22: pmSys-trainer: Survey response read

In addition to looking at the response data of the reports for each individual player, we have also implemented overall statistics on the response rate of the team. As shown in Figure 6.23 we present the coach with a visualization of how many responses that have been registered the last two weeks. This is to monitor the activity of the players.

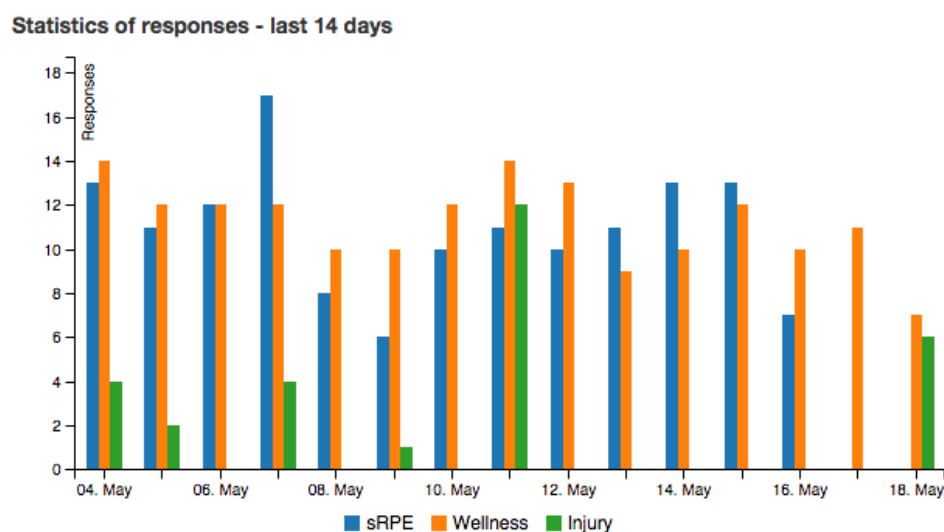


Figure 6.23: Statistics of responses

6.5.4 Visualization

The visualization module is a tool to provide the coach with graphs of the reports from the players. The visualization module consists of two core components, the team and the player visualization. For both the components, they follow the same steps that was presented in Section 5.4.3 for visualizations: gathering, parsing and plotting of data. The only difference is that we have swapped out the NVD3 library with a library called C3.js. C3 offers the same reusable charts, but features more customizations. We can easily make custom graphs, and combine the different charts to create the plots we need.

The format of the data is also slightly different from NVD3. Each dataset on the graph is represented with two array. As shown in Figure 6.24, we have two arrays: one representing the response data from the player and one that is the timestamp of the response. We then have to create an object defining what are the x- and y- values of the dataset, and for our plots, the timestamps is the x-values, while the response data represents the y-values. An advantage with C3 over NVD3, is that we do not need to create JavaScript date objects of the dates because the library will take care of that, but the string has to be in a specific format: YYYY-MM-DD.

```
//array of dates
dates = [date1, date2, date3, ....];
//array of reponses
response_value = [response1, response2, response3, ....]

//bind the arrays together
bind = {
  // y-value : x-value
  'response_value' : 'dates'
}
```

Figure 6.24: Data format in C3.js

A major problem we encountered was that our web application did all the data manipulation and parsing on the client-side. When the coach requested a visualization of the team, we sent the response data to the client-side. The web browser then took care of the parsing and plotting of the data through JavaScript. This was to minimize the load on our server, and worked well on small data sets. But as the players had reported over a longer period, the amount of data had grown too big, crashing the application when it tried to parse the large dataset through the web browser. Because of this issue, we had to resolve to server-side data parsing.

Team visualization

Team visualizations main purpose is to give the coach an overall view of all their players, and see how they are developing and progressing. The coach can also compare players to see how differently they react to the training sessions, as well as look at the reports from the three questionnaires in conjunction with each other. The application will present three graphs, representing the three questionnaires that have been implemented in the system. The first plot shown in Figure 6.25, presents the perceived exertion of the team sessions. Each player is presented as a line, and the coaches expected exertion is presented as a bar. The system will also automatically calculate the team average rating of the exertion that is also presented. We only include team sessions, because this graph compares the player perceived exertion to the coaches expected exertion. If we include the individual sessions and matches in the equation, the total exertion will be higher making the comparison to the coach expected exertion for a team sessions wrong. The graphs are also interactive, where the coach can hover over the data-points to highlight detailed information. All the graphs also have support for a zooming function, where the coach can highlight a certain time frame, as marked with the red box in Figure 6.25. The module also presents a wellness graph that provides the coach with an overall view of the players well-being (See Figure 6.26). Lastly, a injury graph is presented to provide the coach with information on the injury status of the team (See Figure 6.26).

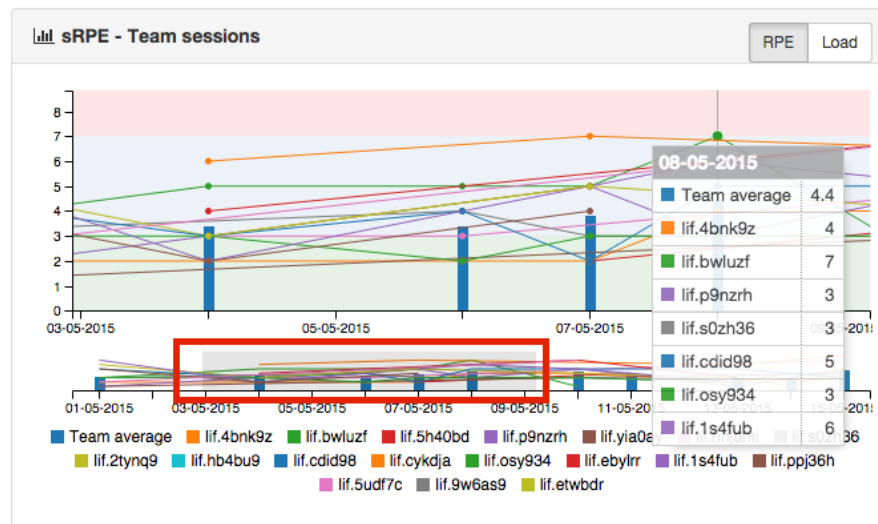


Figure 6.25: pmSys-trainer: Example of team visualization of RPE

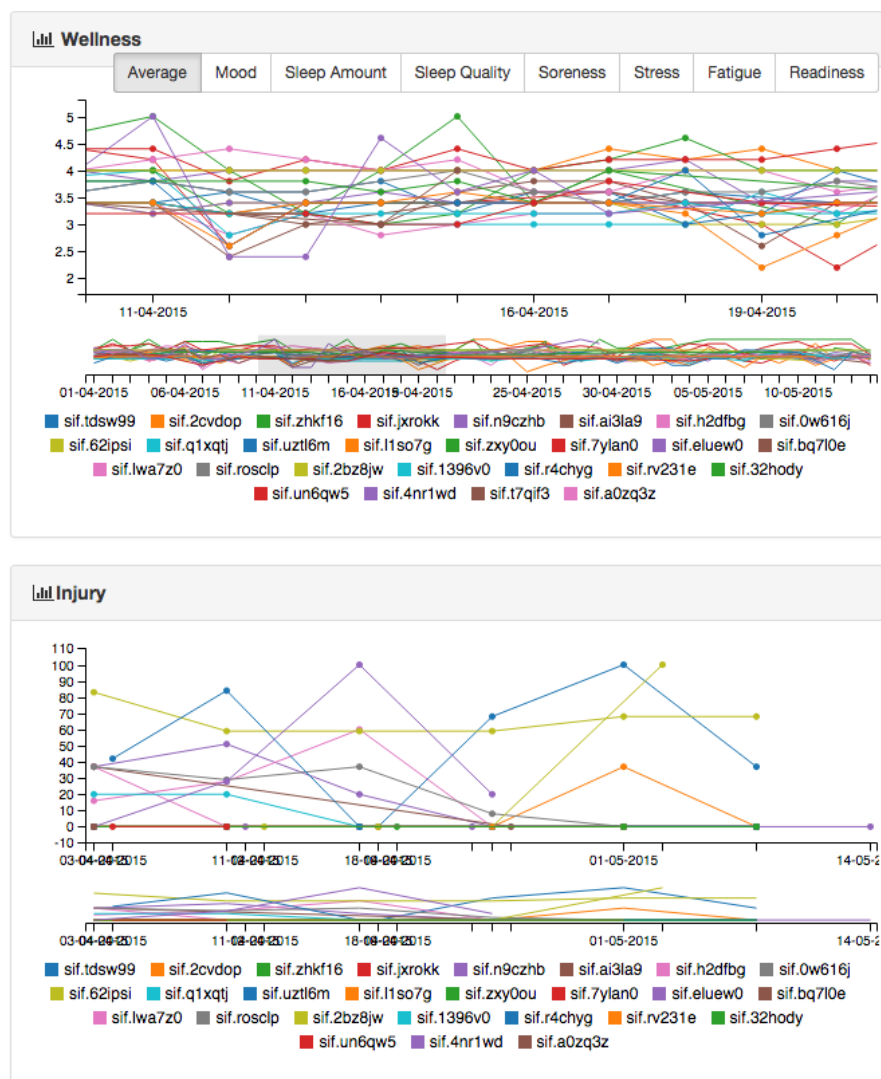


Figure 6.26: pmSys-trainer: Wellness and injury visualization

Player visualization

Player visualization gives a detailed assessment of the players physical status based on the reports. The same graphs in the team visualization are also presented in the player visualization. The difference is

that the coach will only be presented with individual data (See Figure 6.27). This will often be used in conjunction with the team visualization. If the coach detects abnormal values from certain players on the team graphs, they might want to get a more detailed analysis of the player to figure out the reason.

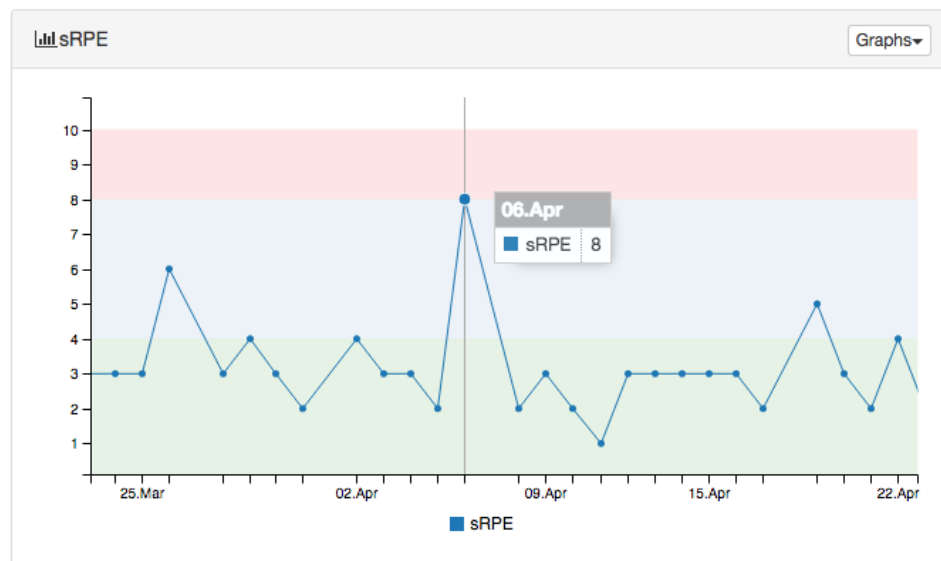


Figure 6.27: Player visualization

6.6 Evaluation

In this section, we will evaluate pmSys-trainer as an analysis tool for the coaches to assist them in monitoring their players. PmSys-trainer is an essential part of pmSys, because it will involve the coach in a much higher degree compared to Ohmage Web. The reason for this, is that the Ohmage Web is not a tailor-made application for the coaches and had no features that were useful for the purpose of assisting or monitoring the players. None of the teams that are participating in NiH's PhD study has used the Ohmage Web, while pmSys-trainer is already deployed for both the National Team and the Tippeliga teams. The web-portal is being used frequently and we have received very positive feedback from NiH and all the coaches.

With pmSys-trainer, the coaches will only be required to authenticate themselves once when they start the application. PmSys-trainer uses the same authentication method as the mobile application, stateless authentication. Since we store the hashed password within the session cookie, the coach will not be disconnected from the server if they go idle. Ohmage Web in comparison will require re-authentication if the user is idle. This is due to the usage of stateful authentication. Further on, the UI of pmSys-trainer highlight only the features that are useful for the coach, this is due to the fact that we have tailor-made this application for the purpose of monitoring players, and have received feedback on what features that is useful. Although the web application is mainly used as a desktop version, the frameworks we have used for the development are responsive and supports mobile devices, as well as tablets.

The new visualization module is arguably the most useful feature for pmSys-trainer. The most important implementation is the fact that we have visualizations that are based on the implemented questionnaires, that will give the coach an overall view of the team's fitness in terms of training load, wellness and injuries. We also have graphs and statistics to provide the trainer with information on the activity of the players. With the help of these features, it can assist the coach to make a proper decision for the players. Compared to Ohmage Web, pmSys-trainer provides relevant graphs that are interactive and provides more useful information.

With the integration of pmSys-push server to the web-portal, the coaches can send push-messages directly to the players mobile devices. They can set up automatic push-messages that directly will assist the players to remember to report. They can also send individual push-messages to each player to give them feedback on their development or give them encouraging words to motivate them. Together with

visualization module, they can analyze the players and have a direct communication channel to give them feedback instantly.

Lastly, all information that is presented in the web-portal is live data. The moment a player submits a report, it will appear in the portal. Compared to the previous ways to analyzing data, pmSys-trainer is more user friendly, more useful and is accessible at any time. It also relieves a lot of pressure of NiH, that previously had to manually analyze the data and send teams the reports. This is now implemented and integrated directly into the web-portal, saving a lot of time and manual work.

There is still room for a lot of improvements, and the potential is big. We have several requested features that we will include in the next versions of the web-portal. We also have many ideas for improvements that we will present in future work in our conclusion.

6.7 Summary

In this chapter, we presented the monitoring tool for the coaches, pmSys-trainer. We began the chapter by going through Ohmage Web as an analysis tool for the coaches to monitor their players. Based on the evaluation, we concluded that the Ohmage Web had to be kept as an administrative tool for pmSys. But as a tool to assist the coaches in monitoring the players, it lacked many features that were set for pmSys.

This led to the design and development of the new and dedicated web-portal that is tailor-made for coaches. The main purpose of the new system is monitoring, as well as assisting and supervising the players of a team. We have also introduced the standalone pmSys-push system that are integrated with the pmSys-trainer web-portal that makes it possible for the coaches to have a direct communication channel to the players. By introducing team visualizations, the coach can obtain an overall view of the fitness of all the players based on how high training load they have experienced, their wellness, and the injury status of each player. They can also obtain a more detailed analysis through an individual player visualization. Through the graphs, they have an easy way of inspecting and monitoring the players general fitness on a daily basis.

PmSys-trainer also provides the coach with tools that can help them assist the players in the reporting process. Lists of the forgotten reports, and statistics on the response rate of the players provide the coaches a measurement of the activity from the players. Together with the push-module, they can instantly send a message to remind the players to report if they see someone that have forgot. They can also setup automatic push-messages which can be sent at their desired time for all the players. The opportunities here are many. Based on the feedback from the Norwegian National Team, NiH and the Tippeliga teams, the pmSys-trainer web-portal has become a useful tool and has eased the process of monitoring compared to previous systems.

Chapter 7

Automatic data analysis

In this chapter, we look at the possibilities around automated analysis of the reports from the players. In the requirement specification, we wanted the system to minimize the amount of manual work for the coaches. PmSys-trainer is already providing the coach with detailed team-visualizations, as well as more detailed player visualization. By inspecting these visualizations, the coach can analyze the data and supervise the players based on what is shown in these graphs. The analysis through the examination of the data in the visualizations will allow the coaches to take decisions that can benefit the players.

We will in this chapter, present how we can provide the system with the reports and let it do the analysis automatically for the coaches. We will first introduce the background on how the reports can be further analyzed to create answers that are more concrete for the coach. Based on the introduced background, we try to implement statistical calculations to provide the coach with more advanced analysis. The main objective of this module is to provide the coach with an in-depth analysis that that is easy to understand and informative. We will also create a more detailed player profile page that will include the automatic analysis.

7.1 Features

- Analysis of the player wellness. The system will give a warning if a registered response below a certain baseline.
- Analysis of the amount of training load. Little variation in the training load will give warning, as well as a high exertion the day before.
- Analysis of injury. System will give feedback if the player has a registered injury.
- New player-profile page, that replaces the old player visualization.

7.2 Background

In Section 2.1, we introduced the sports related background for this system, that is directly connected to the data we are collecting in pmSys. In order to implement the new module that will automatically analyze the reports from the players, we will present the theory on how we can use the collected data for further analysis on the player.

7.2.1 Rating of Perceived Exertion

The RPE is used to monitor the training sessions of a player. With this, we can quantify the total **Training Load (TL)** the session have impacted the player, by using this formula:

$$\text{training duration (minutes)} * \text{training exertion (1-10 scale)} = \text{Training load (AU (Arbitrary units))} \quad (7.1)$$

In order to do further analysis of the data, simple calculations of "monotony" and "strain" can be made from the TLs. Training monotony is a measure of a day-to-day training variability that has been found to be related to the onset of over training when monotonous training is combined with high TLs. It has been suggested that training with low monotony may prevent injury, illness and improve the performance for the players [57]. Monotony is calculated from the average daily TL, divided by the **Standard Deviation (SD)** of the daily TL calculated over a week. If we were to calculate the monotony of today, we have to include the six previous days as well (7 days in total). The formula for calculating training monotony:

$$\text{mean daily TL} / \text{standard deviation} = \text{training monotony} \quad (7.2)$$

The training strain is calculated from the total training load over the week, multiplied by the calculated training monotony. This method is useful for monitoring players when they are undertaking high TLs. It is the training strain that can be associated with incidences of illness and poor performances [57]. High strain will appear when the player is performing many sessions with high TLs in succession, leaving not enough time for the player to recover between the sessions. The formula for calculating training strain:

$$\text{total weekly TL} * \text{monotony} = \text{training strain} \quad (7.3)$$

7.2.2 Wellness

The wellness survey is used to monitor the general wellbeing of the player. In order to analyze the wellness of a player over time, we will calculate the **Standard Difference score (z-score)** of the latest report. The reason to use the z-score is that every individual player is different. Instead of defining a baseline that will apply for all the players, we will rather analyze the player individually to get the best result. The z-score is a ranking of a value in a dataset. It is calculated by subtracting the specified value from the mean, and dividing it by the SD. Assume we have a dataset, the formula is as following:

$$\text{z-score} = (\text{score} - \text{mean}) / \text{SD} \quad (7.4)$$

If a z-score is 0.0, it means it is equivalent to mean. A z-score of 1.0 is approximately 1 SD above the mean, and 2.0 is 2 SD above the mean and so on. Conversely, negative z-scores will take into account the rankings below the mean [58]. In the analysis implemented in pmSys, we will focus on the negative z-score. This is used to detect if a player is reporting his wellness to be way below the average.

7.2.3 Baselines

The most challenging task is to define the baselines on the values that are achieved through the various calculations done with the reports. This is an area that has to be researched more in order to define the concrete baselines for all the calculations. Together with NiH, we have defined the a set of baselines that is used in this module and will present them in the implementation section (See Section 7.3). It is important to mention that these baselines are still being researched by NiH, and will probably be adjusted as they make progress on the PhD study about monitoring training load and fatigue.

7.3 Implementation

The module will be an extension of the player visualization module in the pmSys-trainer web-portal. In addition to the old player visualization, the web-portal will present the automatic analysis of the player. With the parsed data from the visualization module, we send relevant data to the new analysis module that will execute the statistical computations that will respond with concrete values on the status of the player (See Figure 7.2). The analysis will be in form of a warning system that consist of three levels:

1. Green light. Everything is fine.
2. Yellow light. A warning. This player must be monitored.

3. Red light. Danger. Precautions must be taken.

The analysis extension will use the Node back-end that the pmSys-trainer is built on to perform the calculations. We have also integrated a JavaScript library called Simple Statistics to simplify the development. It is a JavaScript implementation that offers descriptive, regression and inference statistics [59]. Because this module is currently an extension of the player visualization module, we can use the data that was requested for the visualizations. The raw data from the back-end is then parsed and sent to the analysis module. The module will execute the analysis and return with a value. These values are then sent to the front-end to be evaluated with JavaScript, then presented to the coach in the UI, as shown in Figure 7.1.

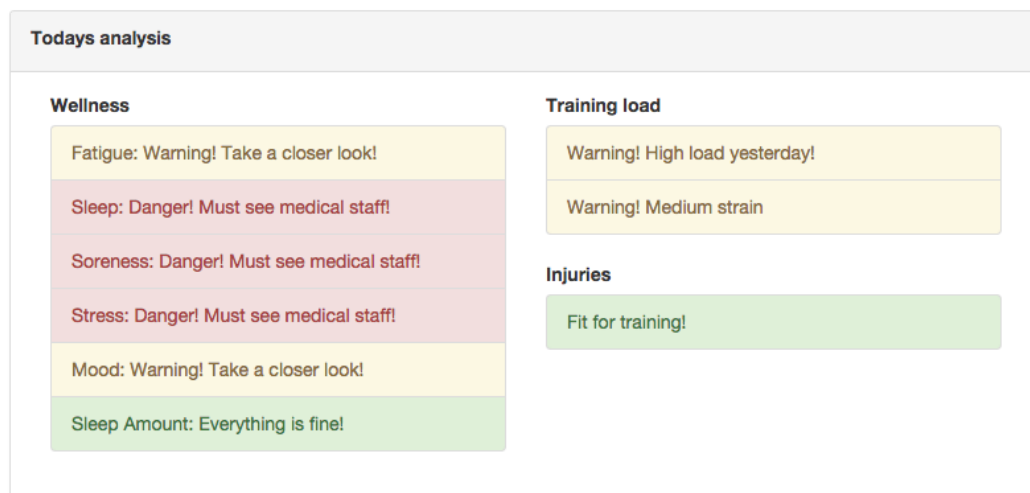


Figure 7.1: Automatic data analysis

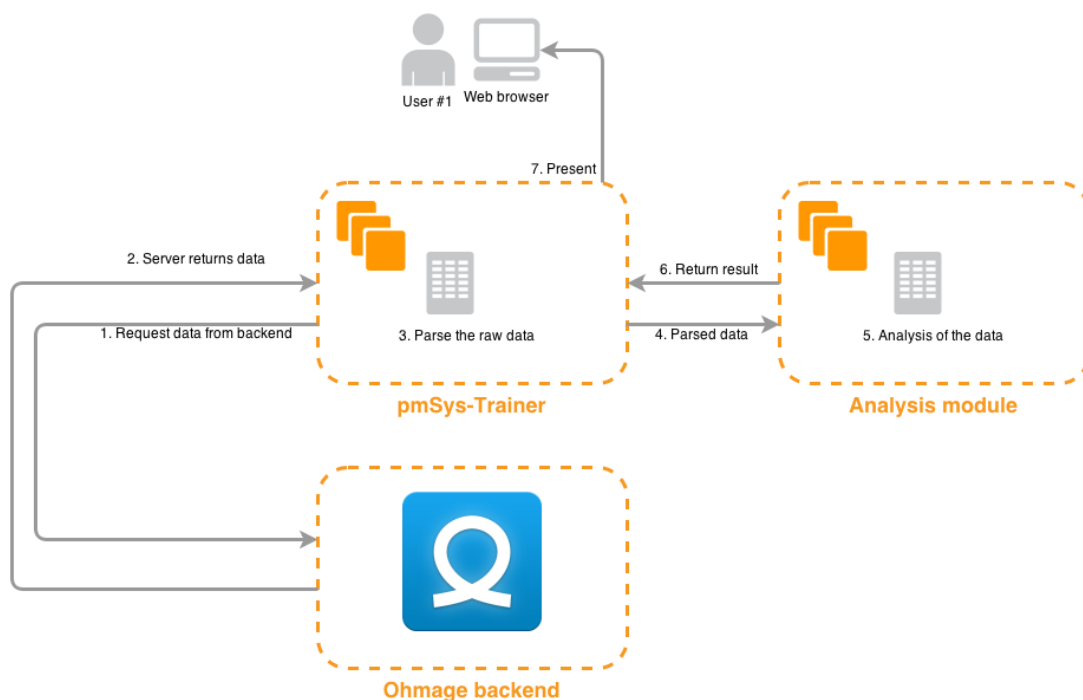


Figure 7.2: Abstract Architecture: pmSys automatic analysis module

7.3.1 Analysis of wellness

To analyze the wellness through z-scoring, the module takes in a JSON-object containing the wellness response data from the player. We will then calculate the mean and SD of the data sets of the following wellness parameters: fatigue, sleep quality, soreness, stress and mood. Based on the mean and SD, we then calculate the z-score from the latest wellness report. The evaluation of the results is based on these baselines:

- Z-score of 0 and higher = Green light.
- Z-score between 0 to -2 = Yellow light.
- Z-score -2 and lower = Red light.

For sleep amount, we do not calculate the z-score. Instead the following baseline is set: 7 hours or more is good (green), 4-6 hours is decent (yellow), while 4 or less is very poor (red).

7.3.2 Analysis of training load

In this analysis, we will look of two types of training loads. The first is an implementation of the analysis of yesterdays total load for the individual player. The system will search for the total training load for the player from the day before, and based on that give the coach a feedback. The other one is based on the training strain the player have experienced. We will provide the pseudo-code for calculation of strain (See Figure 7.3.2), since it is a bit more complex. In Section 5.4.3, we explained the format of the parsed data: all the tables indexes is corresponding, which means that the "timestamp" in index 1 corresponds to the "loadData" in index 1. So in the code, when it finds a matching date, it will take the corresponding "load" value and add it to the total load.

```
timestamps = [dates]
loadData = [loads]

newTable = []

startDate = Date today;
endDate = 1 week ago;

While(startDate > End)
    for(each element in timestamps)
        if(element is equal to startDate)
            add corresponding load from loadData

        subtract a day from the startDate

//new table now contains the last weeks data of loads
mean = find mean(newTable)
SD = findSD(newTable)

strain = mean*SD
return strain;
```

Figure 7.3: Analysis: Strain

When the calculations are done in the analysis, it is returned to the player visualization module, and sent to the front-end for evaluation. The evaluation is based on the following baselines for yesterdays load:

- Less then 400 = Green light.

- 400-800 = Yellow light.
- Above 800 = Red light.

And for the strain we have defined temporary baselines, since this has not been decided for sure yet:

- Less than 1000 = Green light.
- 1000-2000 = Yellow light.
- Above 2000 = Red light.

7.3.3 Player profile

As a part of the implementation of the automatic analysis module on the back-end. We have also revamped the player visualization module on the front-end, and have included more detailed graphs together with the automatic analysis of the player. As we keep evolving this part of the system, the player visualization module is becoming more of a player profile, rather than just purely visualizations. In Figure 7.4, we show an overview of the new profile page. The module will now provide the coach with information about the player's readiness, training load, weekly training load, wellness, injuries and an analysis:

- Readiness: How ready the player is for a match on a scale from 1-10.
- Training load: The player's training load for the last 30 days.
- Weekly training load: New graph with the integration of the new calculations of: Weekly TL, training monotony and training strain.
- Wellness: The player's wellness the last 30 days.
- Injury: Registered injuries
- Analysis: An analysis of the player's fitness.

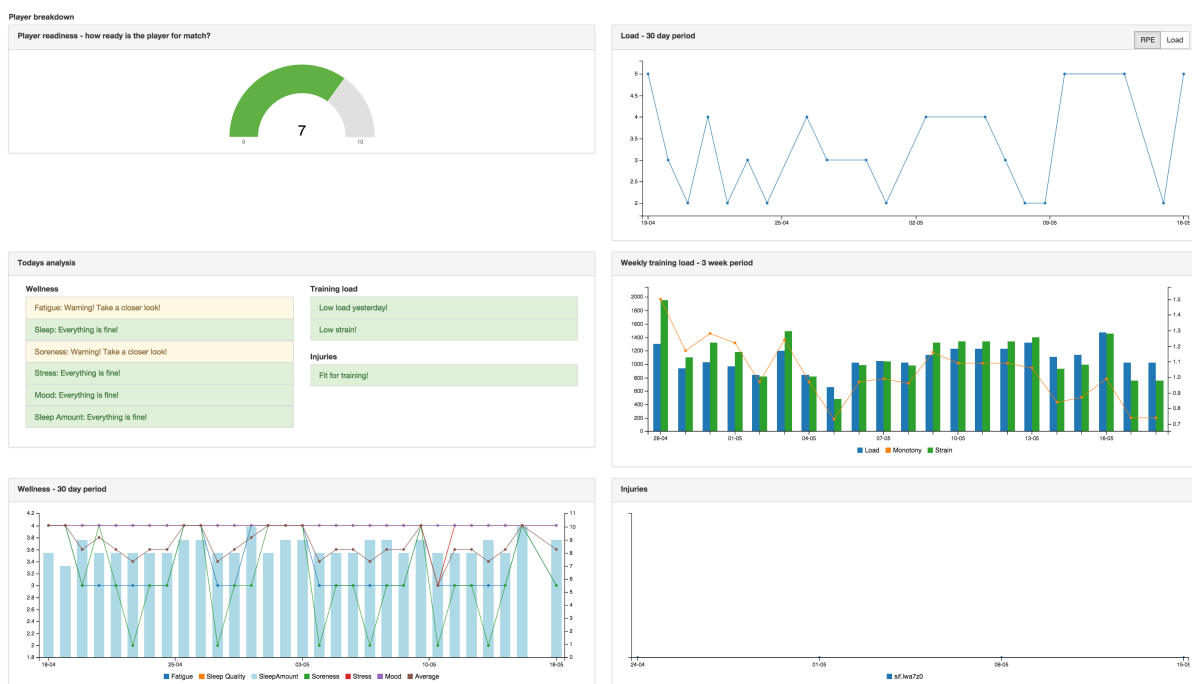


Figure 7.4: Player profile

7.4 Discussion

In this section, we will discuss the usefulness of the new module, and the potential it has for future work of pmSys. First off, the automatic analysis will give the coach feedback based on the reports of an individual player. Compared to before, where the coach had to analyze the graphs to spot abnormalities in the reports, the system will now detect certain values and give a warning based on how critical it is. The implemented analysis is just a small portion of the potential this module has, but it is already useful in terms of providing information that is easy to understand, as well as being useful. The analysis of wellness with z-score provides an individual analysis of each player on how they are feeling, and can detect wellness reports that are lower than normal. The analysis of the training load will help the coach adjust the training, since successive training sessions with high load has lead to incidences of illness and poor performances [57].

The short-term plans for this module is that we want to extend it to other platforms as well. Currently it is limited to the pmSys-trainer web-portal, but in the future, we can create this module as an web-API that can take in data-sets of the players before it returns an analysis. With such a web-API, we can implement the automatic data analysis for other platforms, including the pmSys-app for the players. There are still many useful statistical computations that can be used to analyze the data, that we can continue to implement in future work to provide even more information for the coaches. Over time, we hope to create a player profile that will analyze each player's tendencies. At the moment, NiH has set some average baselines that are implemented in our system. Although these baselines are based on the research they have done, it is not 100 percent accurate for all players. This brings us to the ideas around machine learning.

Machine learning uses statistics, the same way we have used statistics to implement the simple analysis of each player. The statistic is used to determine the probability of an event to happen. In pmSys's case, we want to calculate what type of training will cause high training loads, or can negatively impact the wellness of a player? How high training loads will lead to illness and poor performance? How many high load training sessions in succession is harmful? By introducing machine learning, the system can analyze each player individually, and create a unique player profile that is based solely on the collected data from the specific player. How are the opportunities around machine learning for pmSys? According to the book, *Machine Learning In Action* [60], there are certain steps to follow in order to create a machine learning application:

1. Collect data
2. Prepare input data
3. Analyze the data
4. Train an algorithm
5. Test the algorithm
6. Deploy and use it

PmSys is currently at the third step of a long process of developing a machine learning application. NiH is still researching and analyzing player reports to understand in under which circumstances does it lead to overtraining and overuse injuries. Before we can introduce machine learning we have to model the problem and understand the problem around overtraining and overuse injuries. The actual machine learning takes place when we have in some extent, modeled the problem, and come with an algorithm that the machine will use for the actual learning. The learning process is the process of feeding it with known results and values that come from research and analysis. This process is also called supervised learning. The next step is to continue the learning process by testing it for unknown values. The system will try to predict the outcome based on the previous learning. PmSys, are still far away from being able to accomplish such a feat, but as we learn more about the player reported data, this can become a possibility.

The data that pmSys is collecting and used for analysis is currently limited to subjective data from the players, as explained in Section 1.3. But, the implementation of objective data into pmSys is currently being researched and is a part of our long-term plan for pmSys. With the introduction of objective data, we can create even more sophisticated and more accurate analysis. Known usage of third-party devices and systems that are being used by NiH and the Tippeliga teams are the HurLabs jumping board [61], FitBit [62] wristband activity tracker and ZXY Sports Tracking system [63]. Each of them are measuring different types of objective data. We will not go in-depth on the specifications and the usage of the devices, but we want to point out the many opportunities there is for integration of objective data in conjunction with the already collected subjective data through pmSys. The main challenge when introducing another source of data, like objective data, is being able to extract the relevant data for the analysis, as well as use all the data in conjunction with each other. How can the subjective data we measure with the pmSys, be used together with objective data from third-party devices?

Another problem that arises when we combine the many sources of possible data are that the sets starts to become complex and very big. The problem with big data-sets are that we cannot compute the data in the traditional ways, because it is too time consuming. We cannot expect a coach to wait for hours while the system computes an analysis. The analysis of the big data-sets are often called **big data**, and the main objective of big data is to manage immense amount of data quickly. To handle the data, big data will take advantage parallelism. This parallelism is achieved through big computing clusters, that are a large collection of hardware, including conventional processors and are connected by Ethernet cables and inexpensive switches. The issue with big data sets is not a problem with pmSys yet, but as the system evolves with the possible introduction of new sources for data, this is field that we have to be aware of.

7.5 Summary

In this chapter, we introduced a little module that realizes a small portion of the big potential the automatic data analysis module has. This module allows the coaches to circumvent the need to check the graphs every time a simple analysis is needed for the individual player. By providing the coach with easy and useful information through automatic analysis, we have given the coach another feature that have eased the monitoring process even further. We have then presented our ideas and future work for this module, and shown the immense potential it has. First, we have presented the short-term idea of extending the module to be used with pmSys-app by introducing a web-API. We also plan to implement additional computations, to provide the coach with even more information. Then, we have presented the long-term plans for the analysis module, which includes objective data and machine learning together with big data.

Chapter 8

Conclusion

8.1 Summary

In the Section 1.2, we presented the problem definition as: *In this thesis, we investigate the question on how to develop and design a digitalized monitoring system that includes collecting, storing, analysis and presentation of data. How can we create a tailor-made system to assist football teams in monitoring the training load and fatigue of their players by providing meaningful information?* We also went into the details of the problem definition, by explaining that the main objective of the system is to ease the monitoring process for both coaches and players, by replacing the cumbersome pen and paper format, as well as improving existing system that have been used.

In order to collect the appropriate data to monitor of football players, we have collaborated with NiH. NiH is currently conducting a PhD study to investigate the usefulness of objective and subjective tools for monitoring training load and fatigue in professional football [4]. As a part of this collaboration, we developed a tailor-made system for football teams to collect self-report subjective data from the players. The focus of this new system is being able to collect data through questionnaires, asking the players about their perceived training load, wellness and injuries. Through the collected data, we can monitor the fitness of the players, and provide teams with a tool that can help them provide better recovery and periodization strategies to enhance physical performance, avoid fatigue and overtraining, and be able to prevent overuse injuries.

Based on the problem definition, we have designed and implemented a player monitoring system, called pmSys. PmSys consists of three main components. First, a server back-end that will take care of everything such as secure communication, proper authentication, account management, access control, data storage and management. We have then developed a new mobile application, called pmSys-app. By taking into account what requirements a football player is demanding in a monitoring system, we have developed a new mobile application, which allows for fast reporting. In addition to the reporting implementation, we have also implemented motivational features, as well as tools to assist them in the reporting process. By introducing these additional features to the reporting tool, the application has managed to encourage the players to participate in the team's monitoring process.

We have also created a custom-built web-portal for the coaches, named pmSys-trainer. It is a dedicated tool for monitoring, but it also includes features to assist and supervise the players during the monitoring process. This application is vital since it is the component of the system that realizes the monitoring. By presenting visualizations and analysis, the coaches is now capable of taking decision that will benefit the players in terms of enhancing physical performance, avoiding fatigue and overtraining, and possibly prevent overuse injuries. We have also eased the process of monitoring a team by automatizing many of the features, and removing the unnecessary tools that was confusing in previous applications.

To evaluate pmSys as a monitoring system, we have performed benchmarks and conducted user-studies up against existing systems, and presented these results. Finally, we have looked at the opportunities of the system being able to automatically analyze the reports from the players. In return, the system can be capable of creating detailed and accurate player profiles that is based on each players data.

8.2 Main Contributions

In this thesis, we have shown the implementation of a monitoring system, that have been tailor-made for monitoring football players. The system is deployed for the Norwegian National Team, several Tippeliga teams and G16 Lørenskog Idrettsforening football team. All the teams are using it frequently, and we have received great feedback from all the participating teams.

We have designed and developed a mobile application that is suited for a football player, and shown that the solutions that we have implemented in the application have given good results in terms of increased activity and interest for the monitoring process. With our own push server, pmSys-push, the coach can send push-messages for the team, which have resulted in reports that are made more frequently and at the right time. This have increased the data quality and quantity, which again have led to more accurate analysis of the player. By introducing visualizations as feedback for the players, we have given them motivation to participate. They will get a sense of being more involved rather than just delivering reports to the coach with nothing in return. We have also been able to optimize the reporting process for the players. With pmSys-app, the time consumed for completing a report is overall lower when compared to existing solutions. This will allow the player to use less time on reporting, and focus on playing football.

We have also designed and developed a dedicated monitoring and analysis tool for the coaches in the web-portal, pmSys-trainer. The web-portal has given the coach a user-friendly tool to monitor, analyze, assist and supervise the players. The web-portal features tools like visualizations, push-messaging to the players, automatic analysis and statistics. These features are giving the coaches a proper tool for monitoring the players compared to the existing applications. By circumventing the need for third-party software, and implementing the analysis directly into the web-portal, the coaches can receive feedback from the players instantly as they submit a report, allowing for the coach to adjust immediately.

8.3 Future work

Since the deployment of pmSys for the teams, we have received a lot of feedback on requested features and improvements for the system. We will in the next sections list the requested features, as well as ideas for implementation and experiments that we could see be done in the future.

8.3.1 PmSys-app

Extend platform support The mobile application already support two of the major mobile platforms in iOS and Android. With Cordova, the possibility of extending the support to other platforms is viable. We have already looked into Windows Phone, since we had several players that were using that platform on the various teams, and hope to be able to provide pmSys-app to them as well.

Reporting feedback A feature that will automatically tell the player that the report has been done, thus avoiding duplicate reports. This can be implemented as a counter, that will show the player how many times the player have answered the different questionnaires.

Chat A requested feature by both the coaches and players. Currently, with push-messages, pmSys has an one-way communication channel. With a chat system, the players can respond and discuss with the coaches in real-time.

Language selector The majority of the users speak Norwegian as their native language. But since pmSys have international players that are using the system, we must support English as well. PmSys is implemented in English. By providing a language selector, the player can select a language they feel comfortable with.

8.3.2 PmSys-trainer

Machine learning This opportunities around this field has been discussed in Section 7.4. With machine learning we can get a detailed individual analysis on each player. We can train the system to recognize certain values that have been discovered through research and analysis.

Third-party devices (objective data) PmSys is currently a self-report subjective data collection system. We base our analysis on these data. With the introduction to new sources of data in form of objective data through various devices, we can get even more data about the players. The challenge is to extract relevant data in order to increase the quality of the analysis.

Introduction of Big Data If pmSys evolves into a major system with several sources of objective and subjective data, the data-sets will eventually become too large. Traditional ways of computing data will therefore not be viable anymore, and we must introduce the Big-data approach.

User mapping In pmSys, all the users are anonymous. Only those who have been granted permission to process personal data have the mapping between the randomly generated username and the actual football player. This mapping is in form of local PDF-file that the coach have to manually use and translate the usernames over to real names, which is time consuming and cumbersome. We want to integrate this mapping in to the web-portal, so they coach can circumvent the manual mapping.

Register questionnaire Only the pmSys-app is capable of registering reports from the questionnaires. This means that if the coach were to register an expected RPE for a team session, they would have use the mobile application. By introducing the reporting module on the web-portal, the coach can easier plan, analyze and register new expected RPE's for the players.

Web-portal for players In the current version, pmSys-trainer is a dedicated web-portal for the coaches. In the future, we can open this portal for the players, providing them more features than on the mobile application.

8.3.3 Ohmage back-end

Third-party devices (objective data) An alternative for the integration of new sources of data, is to use our existing Ohmage back-end for it. The commodity of collecting objective data is explained in Section 8.3.2.

Update campaign A major drawback on the campaigns and survey module of Ohmage, is that once it is configured and used, we cannot edit it. So for instance, we have configured an campaign, but want to change a question, this will require us to create a new campaign in the system. By implementing changes to the server, we can allow for easier updates on the campaigns.

Parsing on back-end Because of the format Ohmage has on the response data, we are required to do a lot of parsing in the source code of our applications. A solution is to move the parsing to the back-end, so it returns data in the format that is used in pmSys.

Appendix A

Accessing the source code

The source code for the pmSys project is divided into three repositories. Access to the repositories can be given upon request.

A.1 pmSys-app

<https://bitbucket.org/nktteam/pms-app>

A.2 pmSys-trainer

<https://bitbucket.org/nktteam/pms-trainer>

A.3 pmSys-push

<https://bitbucket.org/nktteam/pms-pushserver>

Appendix B

User Surveys

B.1 pmSys vs. Ohmage

Brukerundersøkelse

Fra en skala fra 1 til 5 (helhetsvurdering), hvordan vil du rangere...

***Må fylles ut**

1. Hvilken bakgrunn har du?

Studieretning / Yrke

.....

2. Driver du aktivt med sport i fritiden?

Merk av for alt som passer

☐ Ja

☐ Nei

3. Brukervennligheten til pmSys? *

Markér bare én oval.

☐ Dårlig

☐ Ok

☐ Bra

☐ Veldig bra

☐ Utmerket

4. Brukervennligheten til Ohmage? *

Markér bare én oval.

☐ Dårlig

☐ Ok

☐ Bra

☐ Veldig bra

☐ Utmerket

5. Designet (grensesnittet) til pmSys?

Markér bare én oval.

☐ Dårlig

☐ Ok

☐ Bra

☐ Veldig bra

☐ Utmerket

6. Designet (grensesnittet) til Ohmage?

Markér bare én oval.

- ☐ Dårlig
- ☐ Ok
- ☐ Bra
- ☐ Veldig bra
- ☐ Utmerket

7. Navigasjonen i pmSys?

Hvordan er det å manøvrere i applikasjonen?

Markér bare én oval.

- ☐ Dårlig
- ☐ Ok
- ☐ Bra
- ☐ Veldig bra
- ☐ Utmerket

8. Navigasjonen i Ohmage?

Hvordan er det å manøvrere i applikasjonen?

Markér bare én oval.

- ☐ Dårlig
- ☐ Ok
- ☐ Bra
- ☐ Veldig bra
- ☐ Utmerket

9. Hvordan presenteres innholdet i pmSys?

Markér bare én oval.

- ☐ Dårlig
- ☐ Ok
- ☐ Bra
- ☐ Veldig bra
- ☐ Utmerket

10. Hvordan presenteres innholdet i Ohmage?

Markér bare én oval.

- ☐ Dårlig
- ☐ Ok
- ☐ Bra
- ☐ Veldig bra
- ☐ Utmerket

11. Hvilken applikasjon foretrekker du?

Merk av for alt som passer

☐ Ohmage

☐ pmSys

12. Har du kommentarer til pmSys eller Ohmage? *

.....

.....

.....

.....

.....

Drevet av



B.2 Rating of pmSys

Brukerundersøkelse

PmSys brukerundersøkelse

*Må fylles ut

1. Hva synes du om prosessen for RPE, wellness og injury rapportering? *

Markér bare én oval.

- ☐ Dårlig
- ☐ Ok
- ☐ Bra
- ☐ Veldig bra
- ☐ Utmerket

2. Hvilken rapporteringsverktøy foretrekker du?

Markér bare én oval.

- ☐ Penn og papir
- ☐ Excel
- ☐ Web survey
- ☐ Mobil applikasjon
- ☐ Annet

3. Hvis annet, hvilke?

.....

4. Sparer pmSys deg for tid ved rapportering?

Er pmSys raskere enn andre verktøy du har brukt?

Markér bare én oval.

- ☐ Ja
- ☐ Nei

Fra en skala 1 (dårlig) til 5 (utmerket), hvordan vil du rangere (helhetsvurdering)...

5. **Brukervennligheten til pmSys? ***

Markér bare én oval.

- ☐ Dårlig
- ☐ Ok
- ☐ Bra
- ☐ Veldig bra
- ☐ Utmerket

6. **Designet (grensesnittet) til pmSys?**

Markér bare én oval.

- ☐ Dårlig
- ☐ Ok
- ☐ Bra
- ☐ Veldig bra
- ☐ Utmerket

7. **Navigasjonen i pmSys?**

Hvordan er det å manøvrere i applikasjonen?

Markér bare én oval.

- ☐ Dårlig
- ☐ Ok
- ☐ Bra
- ☐ Veldig bra
- ☐ Utmerket

8. **Hvordan presenteres innholdet i pmSys?**

Markér bare én oval.

- ☐ Dårlig
- ☐ Ok
- ☐ Bra
- ☐ Veldig bra
- ☐ Utmerket

9. **Nyttigheten av pmSys sine funksjoner?**

Påminnelser for rapportering? Visualisering?

Markér bare én oval.

- ☐ Dårlig
- ☐ Ok
- ☐ Bra
- ☐ Veldig bra
- ☐ Utmerket

10. **Utbytte ved bruk av pmSys?**

Har du fått noe igjen av å bruke pmSys?
Markér bare én oval.

- ☐ Dårlig
- ☐ Ok
- ☐ Bra
- ☐ Veldig bra
- ☐ Utmerket

11. **Har du andre kommentarer til pmSys? ***

Forbedringer / Ønskede funksjonalitet / Ris / Ros

.....

.....

.....

.....

.....

Drevet av



Appendix C

Surveys in pmSys

C.1 RPE Survey

Rating of Session RPE

When?

Within 15 minutes after each training session and match. (typically in the dressing room). If that is not possible, do it as soon as possible. If you want to register the session you had yesterday, please check for "yesterday" in the check box on the last page of the registration.

Registration contents

Please answer the following questions:

1. Was this a (match, team session or a individual session)?
2. Which type of session (football session, endurance session, strength/speed session or other)?
3. Duration? (number of minutes)
4. How was your session today (0-10)?

What kind of information do we get?

Session RPE provide information on intensity, duration and frequency of your training sessions and matches. By tracking these data over time, the coach can supervise the total training load and the variation in training load.

Clarifications

- Dictionary
 - Match: official match or friendly match.
 - Team session: Training session that include the team or part of the team
 - Individual session:
 - Football session: a session on the field which includes the ball
 - Endurance session: For example a running, cycling or swimming session ment to improve endurance
 - Strength training: Strength training or core training (typically in the gym)
 - Other: Other activity that doesn't fit into the other (e.g. playing tennis, yoga)
- Rate how intense you experienced the session
- The rating should be an average of the whole session (fig.1). Take into account periods of high intensity running and periods of standing still.
- 10 is the highest exertion that you can imagine. Imagine pushing yourself running a 3000 m test, without any break.
- 0 is equivalent to rest, and should not be used in combination with training.
- It is exhausting to sprint, having a high heart rate, fast breathing, but also to tackle, jump and duel. How exhausted you feel in your muscles and mentally is also a part of the RPE.
- A match is typically rated 6-7-8, but could also be higher or lower.
- A strength training session is typically 2-3-4-5 (because of much pauses)

Minutes:	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	Average
RPE:	8	5	6	3	6	8	5	6	9	8	7	6	4	6	4	6	7	10	6,5

Fig. 1. Example from a 90 minutes session. The assessment should reflect an average of the whole session. Imagine that you rate every 5th minute of the session and then calculate the average value.

Rating of Session RPE

Visualization

- Training load: is the RPE score multiplied by the duration of the session. High training load occurs either by high RPE score, high duration or both.
- Weekly load: is the average training load the last 7 days.
- Monotony describes the variation in training load over the last 7 days. High monotony means low variation in training load.
- Strain is the average weekly load multiplied by the monotony. High strain means that the weekly load is high combined with low variation in training load. High strain means less time for recovery and is associated with overtraining or injuries.

Session RPE - rate of perceived exertion

Rating	Explanation
0	Rest
1	Very, very easy
2	Easy
3	Moderate
4	Somewhat hard
5	Hard
6	
7	Very hard
8	
9	
10	Maximal

C.2 Wellness survey

Rating of wellness

When?

Please rate your wellness every morning, 7 days a week. The rating must take place after getting out of bed, but before training. For example before or after breakfast or in the dressing room before the session.

Registration contents

1. "Readiness to play"
2. "Fatigue"
3. "Sleep Quality"
4. "Hours of Sleep"
5. "General Muscle Soreness"
6. "Stress Levels"
7. "Mood"

What kind of information do we get?

Wellness indicates how well the players overcome or responds to the training load and how well he recover? A lower score than normal over time may indicate a higher risk of overuse injuries.

Clarifications

- Rate as best as you can according to the questions
- On the scale, 3 is normal, 1 er "worst" and 5 "best".
- "Readiness to play" has a scale from 1-10, where 1 is "not ready at all" and 10 is "maximally ready".
- Dictionary
 - *Fatigue*: means tiredness resulting from mental or physical exertion or illness.
 - *Sleep quality*: means "how was your sleep last night?"
 - Hours of sleep: "how many hours did you sleep last night?"
 - *General muscle soreness*: means general soreness in the musculature (especially in the legs)
 - *Stress levels* means a state of mental or emotional strain or tension resulting from adverse or demanding circumstances
 - *Mood* means emotionally state of mind
 - *Readyness to play*: means "how ready (physically and mentally) are you to play if there is a match today/tonight?"

Visualization

On the visualization page, you can view your latest rating of fatigue, sleep, soreness, stress and mood. You can also view your ratings for the last 30 days. A thick red line represents the average of the five wellness parameters.

	5	4	3	2	1
Fatigue	Very fresh	Fresh	Normal	More tired than normal	Always tired
Sleep quality	Very restful	Good	Difficulty falling asleep	Restless sleep	Insomnia
Hours of sleep	-	-	-	-	-
General muscle soreness	Feeling great	Feeling good	Normal	Increase in soreness/tightness	Very sore
Stress levels	Very relaxed	Relaxed	Normal	Feeling stressed	Highly stressed
Mood	Very positive mood	A generally good mood	Less interested in others and/or activities than usual	Snappiness at team-mates, family and co-workers	Highly annoyed/irritable/down

C.3 Injury survey

Injury registration

When?

Once a week on a fixed day.

Registration contents

Part 1: Please answer the following questions as best as you can

1. Have you had any difficulties participating in normal training and competition due to injury, illness or other health problems during the past week?
2. To what extent have you reduced your training volume due to injury, illness or other health problems during the past week?
3. To what extent has injury, illness or other health problems affected your performance during the past week?
4. To what extent have you experienced symptoms/health complaints during the past week?

Part 2: If you have experienced injuries/illnesses, you will continue with these questions

1. Is the health problem an injury or illness?
2. Select the area that best describes the injury / illness?
3. Please state the number of days over the past 7-day period that you have had to completely miss training or competition due to this problem?
4. Is this the first time you have reported this injury?
5. Have you reported the problem to the medical device?
6. Do you have more injuries to report?

What kind of information do we get?

The injury registration systemize information about acute injury, overuse injuries and health problems. The registration may detect health problems and symptoms before it develops into an overuse injury. It also record small injuries/illness that are often overseen in traditional injury registration

Clarifications

It is important that you register all your health problems every week, even if you have registered the same problem before, or if you are receiving treatment for it. If you have several injuries/illnesses within one week, be sure to record all of them by going through the registration several times. Record the most serious injury/illness first.

Your team physician/physiotherapist/fitness coach will receive a message when you record and injury. It is important to emphasize that this system does not replace your regular contact with the medical team. Please continue to make direct contact with the team physician or physiotherapist when you need it.

Visualization

The visualization indicates a severity score of the injury/illness to be used in research. Each of the questions (1-4) scores 0-25, and the larger the sum is, the larger the severity score is. It is important to emphasize that only your team physician or physiotherapist can diagnose and decide how seriously your injury or illness is.

Bibliography

- [1] FIFA. FIFA Ballon d'Or.
<http://www.fifa.com/ballon-dor/player-of-the-year/>. Accessed = 2015-04-10.
- [2] Joel S. Brenner. Overuser injuries, overtraining, and burnout in child and adolescent athletes. *PEDIATRICS Volume*, 119(6):1242–1245, 2007.
- [3] Medianorge. Access to ict-equipment at home.
<http://www.medienorge.uib.no/statistikk/medium/ikt/249>. Accessed: 2015-02-01. Accessed = 2015-02-01.
- [4] Haavard Wiig. Research project: Load monitoring in football.
<http://www.nih.no/en/research/prosjektarkivet1/forskningsprosjekter-ved-nih/load-monitoring-in-football/>. Accessed = 2015-02-01.
- [5] Google. Definition of monitor. <https://www.google.no/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=monitoring%20definition>. Accessed = 2015-04-20.
- [6] Peter J. Denning, Douglas E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young. Computing as a discipline. *ACM*, 32(1), 1989.
- [7] NRK. New app shall secure euro-success.
<http://www.nrk.no/troms/ny-app-skal-sikre-em-suksess-1.12048486>, 2014. Accessed = 2015-05-01.
- [8] Carl Foster, Jessica A. Florhaug, Jodi Franklin, Lori Gottschall, Lauri A. Hrovatin, Suzanne Parker, Pamela Doleshal, and Christopher Dodge. A new approach to monitoring exercise training. *Journal of Strength and Conditioning Research*, 15(1):109–115, 2001.
- [9] Kid-Edgar Sørensen. Ruoksat: A system for capturing, persisting and presenting the digital footprint of soccer knowledge and expertise. Master's thesis, University of Tromsø, 2013.
- [10] Blade D. McLean, Aaron J. Coutts, Vince Kelly, Michael R. McGuigan, and Stuart J. Cormack. Neuromuscular, endocrine, and perceptual fatigue responses during different length between-match microcycles in professional rugby league players. *International Journal of Sports Physiology and Performance*, 5:367–383, 2010.
- [11] Benjamin Clarsen, Grethe Myklebust, and Roald Bahr. Development and validation of a new method for the registration of overuse injuries in sports injury epidemiology: the oslo sports trauma research centre overuse injury questionnaire. *Br J Sports Med*, pages 1–8, 2012.
- [12] Medianorge. Access to smartphones.
<http://medienorge.uib.no/statistikk/aspekt/tilgang-og-bruk/379>. Accessed = 2015-02-01.
- [13] Deborah Estrin. Participatory sensing: Applications and architecture. *IEEE Internet Computing*, 2010.

- [14] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. *ACM*, 2006.
- [15] World Health Organization. Chronic diseases and health promotion. <http://www.who.int/chp/en/>. Accessed = 2015-03-01.
- [16] Deborah Estrin and Ida Sim. Open mhealth architecture: An engine for health care innovation. *Science*, 330(759-760), 2010.
- [17] H. Tangmunarunkit, C. K. Hsieh, J. Jenkins, C. Ketcham, J. Selsky, F. Alquaddoomi, D. George, J. Kang, Z. Khalapyan, B. Longstaff, S. Nolen, T. Pham, J. Ooms, N. Ramanathan, and D. Estrin. Ohmage: A general and extensible end-to-end participatory sensing platform. *UCLA Computer Science Technical Report*, 2014.
- [18] DHIS2 Documentation Team. Dhis 2 user manual. https://www.dhis2.org/doc/snapshot/en/user/html/dhis2_user_manual_en.html. Accessed = 2015-03-01.
- [19] AC Milan. Milan lab. http://www.acmilan.com/en/club/milan_lab. Accessed = 2015-03-05.
- [20] MLab London. Inside ac’s milanlab. http://www.meerssemanlab.com/Meersseman_Lab/Inside_AC_Milan_Lab.html. Accessed = 2015-03-05.
- [21] Lovdata. *Law about processing peronal information*. Ministry of Justice and Public Security, 2013.
- [22] Ken Schwaber. Scrum development process. *Advanced Development Methods*, 1995.
- [23] Trello. <https://trello.com/>. Accessed = 2015-04-10.
- [24] Scrum process. [http://en.wikipedia.org/wiki/Scrum_\(software_development\)](http://en.wikipedia.org/wiki/Scrum_(software_development)). Accessed = 2015-02-01.
- [25] Jon Froehlich, Mike Y. Chen, Sunny Consolvo, Beverly Harrison, and James A. Landay. Myexperience: A system for in situ tracing and capturing of user feedback on mobile phones. *ACM*, 2007.
- [26] Inc Joyent. About nodejs. <https://nodejs.org/about/>. Accessed = 2015-03-01.
- [27] Inc MongoDB. Mongodb explained. <http://www.mongodb.com/nosql-explained>. Accessed = 2015-03-01.
- [28] Ohmage: About the client-server protocol. <https://github.com/ohmage/server/wiki/About-the-Client-Server-Protocol-and-System-Entities>. Accessed = 2015-04-01.
- [29] Ohmage api: Error codes. <https://github.com/ohmage/server/wiki/Error-Codes>. Accessed = 2015-02-01.
- [30] Ohmage: About users, classes and campaigns. <https://github.com/ohmage/server/wiki/About-Users,-Classes-and-Campaigns>. Accessed = 2015-04-01.
- [31] Iso 8601 (timestamp format). <http://www.w3.org/TR/NOTE-datetime-970915.html>. Accessed = 2015-03-09.
- [32] Ohmage api: Prompts. <https://github.com/ohmage/server/wiki/Campaign-Definition#prompt>. Accessed = 2015-03-01.

- [33] Renien John Joseph. Single page application and canvas drawing. *International Journal of Web and Semantic Technology*, 6(1), 2015.
- [34] Jesse James Garrett. Ajax: A new approach to web applications. *Adaptive Path*, 2005.
- [35] IBM Software Group. Native, web or hybrid mobile-app development. Technical report, IBM Corporation, 2012.
- [36] Apache Software Foundation. Apache cordova. <https://cordova.apache.org/>. Accessed = 2015-03-28.
- [37] Drifty Co. Ionic. <http://ionicframework.com/>. Accessed = 2015-03-09.
- [38] Iqbal H. Sarker and K. Apu. Mvc architecture driven design and implementation of java framework for developing desktop application. *International Journal of Hybrid Information Technology*, 7(5):317–322, 2014.
- [39] Jatin Chhikara. A web architectural study of html5 with mvc framework. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(12):1451–1454, 2013.
- [40] Google. Angularjs. <https://angularjs.org/>. Accessed = 2015-03-03.
- [41] Angularjs: ngroute. [https://docs.angularjs.org/api/ngRoute/service/\\$route](https://docs.angularjs.org/api/ngRoute/service/$route). Accessed = 2015-03-25.
- [42] Angularui. <https://angular-ui.github.io/>. Accessed = 2015-03-25.
- [43] X2js. <https://code.google.com/p/x2js/>. Accessed = 2015-03-05.
- [44] Nvd3. www.nvd3.org. Accessed = 2015-03-01.
- [45] Moment.js. <http://momentjs.com/>. Accessed = 2015-03-05.
- [46] Cordova, local notification plugin. <https://github.com/katzer/cordova-plugin-local-notifications/>. Accessed = 2015-03-01.
- [47] Apple Inc. App store review guidelines. <https://developer.apple.com/app-store/review/guidelines/>. Accessed = 2015-04-25.
- [48] Apple Inc. Apple push notification service. <https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html>. Accessed = 2015-03-13.
- [49] Google. Google cloud messaging. <https://developer.android.com/google/gcm/server.html>. Accessed = 2015-04-25.
- [50] Ubuntu Community Help Wiki. Cronhowto. <https://help.ubuntu.com/community/CronHowto>. Accessed = 2015-04-15.
- [51] Node-crontab. <https://www.npmjs.com/package/node-crontab>. Accessed = 2015-04-15.
- [52] Express. <http://expressjs.com/>. Accessed = 2015-03-15.
- [53] Evan M. Hanh. *Express.js in Action*. Manning Publications, 2015.

- [54] Introduction to redis. <http://redis.io/topics/introduction>. Accessed = 2015-02-07.
- [55] About bootstrap. <http://getbootstrap.com/about/>. Accessed = 2015-03-03.
- [56] Npm. <https://www.npmjs.com/>. Accessed = 2015-04-20.
- [57] Aaron J. Coutts, Karim Chamari, and Ermanno Rampinini Franco M. Impellizzeri. Monitoring training in soccer: Measuring and periodising training. Accessed = 2015-04-17.
- [58] Robert W. Pettitt. The standard difference score: A new statistic for evaluating strength and conditioning programs. Technical report, Department of Human Performance, Minnesota State University, 2010.
- [59] Simple-statistics. <https://www.npmjs.com/package/simple-statistics>. Accessed = 2015-04-20.
- [60] Peter Harrington. *Machine Learning In Action*. Manning Publications, 2012.
- [61] Hurlabs jumping board. <http://www.hurlabs.com/>. Accessed = 2015-04-28.
- [62] Fitbit. <https://www.fitbit.com/uk>. Accessed = 2015-04-17.
- [63] ZXY Sports Tracking. <http://www.zxy.no/>. Accessed = 2015-04-26.