UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

Privacy-Preserving and Secure Pilot Self-Assessment

The development of Gearggus

Kim Grønning Eide

INF-3990 Master thesis in computer science … May 2022

Abstract

There is a strong culture for using safety risk management tools to monitor the different parts of the operation in the aviation industry. However, most of the monitoring that is being done is done on the technical conditions of the aircraft rather than on the pilot. While there is an expectation that the pilots self-assess their health regarding how fit the pilot is to operate an aircraft, all the tools given are checklist-based. Checklists are widely used in the aviation industry to ensure all tasks are done as they should for mechanics and pilots. However, the drawback of checklist-based systems is that they do not monitor anything over time.

As a pilot has responsibility for many passengers on every flight, the consequences of mistakes can be considerable. By not monitoring the health over time, some of the crucial information when considering whether the pilot is fit to fly or not may be forgotten. Fatigue and stress are two essential topics for ensuring the focus is on operating the aircraft rather than either zoning out or being concerned about something else during flight. As the EU work hour regulations exempt everyone within the aviation industry, pilots can work at any time during the day. As the pilots can work at any given point during the day, they have to self-regulate whether they can work. If they do not track topics such as sleep and nutrition, they can be fatigued and lose focus on the work to be done.

This thesis presents Gearggus, a self-assessment tool that can assist the pilot in assessing their health based on the information given by a questionnaire. The user answers questions based on how important the data is monitored over time. Based on the answers, there is calculated feedback on how ready the pilot is to operate an aircraft. The data is presented on a history page, so the user can see what the score is based on and how to adjust to gain a better score.

Gearggus was evaluated with a qualitative interview with experienced aviation personnel and the Department of Aviation employees at the Unversity of Tromsø. Both parties acknowledge the issues Gearggus is trying to solve, but with modifications to the system. The required changes differ between the

parties.

Acknowledgements

First and foremost, I want to thank my supervisorors Dag Johansen and Pål Halvorsen for their guidance and being available when I needed help or a boost of motivation during the project. Their passion for the field has indeed been outstanding and invaluable source of inspiration.

Furthermore, I want to thank the rest of the CSG group for their help when I've been here. I also want to thank our student advisor Jan Fuglesteg for being available almost all the time; whether it is during weekends, holidays, or late evenings, you always answer any questions.

My sincerest gratitude is to all my friends and fellow students who have made the last five years a genuinely fantastic experience. I want to give special thanks to Christer for sticking with me the entire time and Lemet for helping me name the system.

Finally, I want to thank my family for all the help and support.

Thank you.

Contents

List of Figures

List of Tables

1

Introduction

Since Orville and Wilbur Wright manufactured the first aircraft in 1903, the aircraft systems have gotten more advanced and secure. While there has been much focus on automating the aviation systems to prevent accidents, there has been less focus on the pilots flying the aircraft to avoid accidents. Accidents have globally decreased since the 1969s, where the main contribution to the decrease can be attributed to improvements in aircraft reliability, and safety [1]. This trend started to plateau in 1997 and has remained in the majority on that same plateau since then. Today it is estimated that 75% of all aviation accidents can be attributed to pilot errors.

There is a strong culture for using safety risk management systems (SRMS) to reduce accidents within the aviation industry. However, most of the SRMSs that are being used are focused on the technical condition of the aircraft rather than the pilot's well-being[2]. The SRMSs employed for pilots cover the pilot's overall health regarding safely operating an aircraft. These systems are described in further detail in chapters 2.1.1, 2.1.2 and 2.1.3.

A recent study that outlines requirements to increase the well-being of pilots [2] does suggest that the crew monitoring is minimal. It also points toward that stress, physical activity, sleep, and diet are important factors to consider when evaluating physical health. The stress on everyone in society, including pilots, has increased with the recent covid outbreak. When the pandemic started, 60-80% of the flights were canceled. As the outbreak started, the pilots didn't know whether they had a job and would take a flight they otherwise should

not have taken.

During this thesis, all the current systems found while researching focused on monitoring crew members' health in the aviation industry were checklist-based. These are not written down but rather meant to invoke a thought process to mental process if all the boxes are ticked off. As a result, they all have the same limiting factor. They do not monitor the health over an extended time to help the crew members assess their health status for the flight they are about to embark on.

1.1 Self-assessment systems for pilots

The pilots are currently using systems that heavily depend on how the pilot feels as the data is filled in, rather than monitoring the health over time. Some checklists require the pilot to remember all the points checked for before flying. In a survey that did look at the fatigue for pilots over a week [3], it is suggested that the pilots rarely rate their stress or fatigue levels to be high risk. This is while the pilots also said they were exhausted. The results here may have been skewed due to the pilots' responsibility not to fly while being fatigued.

The systems currently used among pilots are checklists that check for illness, medication, stress, alcohol, fatigue, and emotions. However, there are no questions to raise the awareness of what these different parts of the checklist do cover to help remind the pilots of what is covered and what might affect the performance of the task ahead. There are some of the topics that do cover a wide range of activities that may affect them. For example, fatigue covers parts such as, but is not limited to, sleep, physical activities, and thought-provoking activities, such as chess or other similar activities.

Everyone within the aviation industry is exempted from the European time working directive [4]. As such, pilots are shift workers who can work at any given time during the whole year, including holidays. Therefore, anyone within the aviation industry has a greater responsibility to ensure they have a healthy work-life balance and manage their sleep schedules to cope with different work hours, which may change from week to week.

As the systems that are currently in use are checklists, pilots may have a more challenging time telling anyone about their fatigue levels, as suggested in a survey conducted among french firefighter pilots [5]. An application that tracks the fatigue level over time may assist the pilots in tracking their fatigue levels and showcase how they are currently feeling.

1.2 Privacy in management systems

In "Determine Readiness to Fly: Pilot Risk Management" [5] it was found that not everyone in the focus group would answer honestly if their supervisors had access to the data provided to the system. For this reason, the system needs to be designed with privacy in mind to let the users control if anyone has access to any data. As the data provided can be regarded as health data, there also needs to be a layer of security to avoid any data leakage during transfer to anywhere but the intended end-point.

As the system is intended to be used within the EU, the General Data Protection Regulation (GDPR) law must be followed in the system's design process. This means that any personal data identifying any alive person needs to be protected. Suppose different pieces of information collected together can be used to identify a person. In that case, this will also be included in personal data [6].

GDPR also encourages data protection by design. An example of this is that instead of storing a name to a user, an artificial identifier is used instead, which does not give access to whom the data may consider. Any messages sent should also be encrypted by default to make it harder for any third party involved to find any information about the users.

Any data collected should be stored for as short a period as possible. A time limit should be included to either erase or review the data. The data should only be stored if the purpose of the data still applies and is still valid.

1.3 Problem definition

Systems that can help the crew members assess their body health levels would benefit from monitoring their health over time. What the pilots have done during a particular day will fade over time. To avoid this happening, a mobile application that is available to be filled in when the pilot needs it would store these details. The collected data are to be analyzed to determine how ready a pilot is to fly. While doing this, the application will be a tool the pilots can use to assist in deciding if they are fit to fly or not.

In "Determine Readiness to Fly: Pilot Risk Management," a study was conducted among French firefighter pilots/cadets on how such a system would be set up. A system got proposed based on the findings with different questions to ask to gain information about the health status needed to assess whether the pilot is fit to fly or not. During the survey, it was also found that not all pilots would

answer honestly on all questions if a supervisor had access to their answers. Therefore, the systems should be designed to hide the identity of the users in the system.

This thesis will explore and define the system requirements for a privacy-preserving self-assessment system for pilots and then design, implement and test such a system.

To evaluate this thesis, we first aim to define system requirements and design and implement a prototype of the system described in "Determine Readiness to Fly: Pilot Risk Management" [5] called IMReady. This version will then be further iterated upon to design and implement a similar system aimed toward the general aviation pilots, called Gearggus[1]. Gearggus will be based on a smartphone/tablet use interaction, hereafter called mobile devices. As 96% of Norway's population had access to a mobile device in 2020, it is considered an applicable platform. The reasoning behind this is that these are devices that the users can carry with them all the time to fill in the information needed to help assess their mental health, general well-being, and stress levels.

Gearggus should be evaluated by actual end-users to evaluate how it operates to assess the thesis further. Gearggus is intended to help the end-users assess their mental health, general well-being, and stress levels; the best way to evaluate Gearggus is to let the end-users interact with it.

1.4 Scope and limitations

Throughout this thesis, we make certain assumptions about how pilots would interact with a mobile application. The thesis will be about the design and implementation of the technical part of Gearggus and not the health-related part of the system.

- There is a version of the system that is implemented based on the findings in "Determine Readiness to Fly: Pilot Risk Management" [5]. IMReady is a system version that naively implements the system specifications given by a firefighter pilot. This version is implemented to showcase how the system would look when implemented and in a working condition.

- After IMReady is implemented, this will be reiterated to a version that is designed based on a research methodology. Gearggus will be the final version of this thesis.

1. Gearggus means ready on Sami, the language of the natives in Norway.

- While scalability is a crucial concern with such a system, the evaluation of the system will be done with a small-scale deployment. The reason behind this is to keep the experimentation in a controlled environment. In a significant deployment, software and configuration changes would take longer.

- As IMReady is intended for use in a military setting, the testing of the system will be limited, as it is hard to gain access to military personnel.

- During the system's design process, the usability of the system has to be considered. The end users are pilots, who may range from 18 to 65 years old for commercial pilots[7]. Therefore, the system needs to be designed with good usability for pilots with varying technical abilities.

- A limiting factor of the project is a dataset with health data from pilots, as there is no current dataset available. Therefore, a less ideal version that does not require a dataset will be implemented.

- As some parts of the system have to abide by laws and regulations when such events occur in this thesis; the system will follow the Norwegian laws. Suppose the system is to be made available in the future in other countries. In that case, the system has to be evaluated with the local laws and regulations in mind.

1.5 Methodology

Computer science is a science within a technology-oriented discipline rooted in mathematics and engineering[8]. The core concepts for both computer science and computer engineering are the same. As such, they share the same research methods. The research methods applied in computer science are defined in three major paradigms for research:

Theory is rooted in mathematics and studies objects whose properties and relationships whose clearly defined. There are formed hypotheses about the objects to be studied and whether the relationships are accurate before the results are interpreted. A hypothesis in this field can be about a specific algorithm that can be proved with logical reasoning.

Abstraction is rooted in an experimental scientific method. Abstraction stems from natural science disciplines such as chemistry, biology, and physics, which observe different objects to construct accurate models based on the rules and laws of the object. The models in abstraction are constructed based on a

hypothesis about an observable object. The objects are then designed and used to experiment with the model to form a result to be analyzed. The objects used could be how the software performs in its intended environment or if some hardware components work in an environment as expected.

Design is rooted in engineering. This discipline constructs systems or devices to solve a set of problems. The system is constructed around functional and non-functional requirements and specifications based on what problem the system is set to solve. The system is then tested to verify that it meets the requirements given.

In the computer science discipline, these processes are so intricately intertwined that it is irrational to rely on one of the three paradigms to be the most fundamental. They do, however, still represent distinct areas of computer science. The theory concerns the ability to describe how the different objects work together, the abstraction concerns how these relationships can work together, and the design to implement a system based on the relationship between the objects.

The work in this thesis is not of a theoretical nature. However, it is still based on experiments from other systems implemented. Many different health applications can be used to study the theory behind from. Mental health applications (Mhapps) can represent a new compelling way to deliver a self-guided psychological intervention and stepped care[9].

1.6 Research Context

This thesis has been performed in context with the Corpore Sano center and a collaboration with the French airforce. The Corpore Sano center is a part of the CSG (Cyber Security Group) research group at the UiT. The Corpore Sano center target area is research on computer systems to help top athletes and medicine improve how their work is being done. The research group also has focused on other areas such as computer security architectures and computer systems for the fishing industry.

1.6.1 Corpore Sano

The Corpore San project started in 1988 with the development of Stormcast, an expert-based distributed artificial intelligence application for severe storm forecasting [10].

A collaboration between the Department of Computer Science at the University of Tromsø, Cornell University, and the University of California resulted in the TACOMA (Tromsø And COrnell Moving Agents) project. The TACOMA project is concerned with implementing an operating system support agents, which are agents that migrate through a network [11]. The use of the TACOMA agent was proved with a re-implementation of Stormcast [12].

MapReduce is an implementation for processing and generating large data sets [13]. In 2012, the Corpore Sano center further explored the areas where MapReduce can be used with Cogset. Cogset is a generic and efficient engine for reliable storage, and parallel processing of distributed data sets [14].

Fireflies is a scalable and secure intrusion-tolerance membership and gossip service protocol that makes cloud overlay networks fault-tolerant through intrusion-tolerant distributed hash tables or overlay network routines[15, 16] . Fireflies were developed in 2006 and further expanded with the gossip protocol in 2015. The Balava system put further emphasis on cloud research. Balava [17] is a system for managing computations that span multiple clouds and involve data with confidentiality constraints. Vortex [18] is a multiprocessor operating system that is entirely event-driven.

In recent years the Corpore Sano center has focused its research on how high-impact life science research can help with injury prevention, sports performance development, preventive health care, large-scale population screening, and epidemiological health studies. The systems developed help both elite soccer clubs and the national team prevent injuries during training sessions using technology developed within the Corpore Sano project [19]. The Corpore Sano Center has also done research on Real-Time Detection in Football by using neural networks [20].

The systems developed include support for effective monitoring, collection, back-end storage systems, machine learning, advanced analytic systems, and user interaction and access. All the systems done with the Corpore Sano project have been developed with system robustness by design approach, focusing on fault-tolerance, security, and privacy.

The French airforce also had an internship within the research group from March-June 2021 from 2nd lt. Thibault Kheng, who came up with a baseline design of a system whose goal is to be able to assist firefighter pilots in assessing how fit for a flight they are. The system is described in "Determine Readiness to Fly: Pilot Risk Management" [5]. The system should also be able to give feedback to the pilots on how ready they are, and if they are not 100% fit, come with suggestion on how to get closer to it. This internship has been followed up by corp. Chloé Homolle, who has continued on the same project as 2nd lt.

Thibault Kheng worked on.

1.6.2 Sustainable Fishing Industry

This project started within the research group after getting external funding from Sparebank1 Nord-Norge and was decided with both involvements from them and over 20 experts within different areas.

As a result of the funding, there has been created a system called Dutkat [21]. Dutkat is a multimedia system for catching illegal catchers while preserving the identity of the anglers. The system monitors the fishing vessels with a camera. While the faces of the anglers are made unrecognizable, the system catches any illegal fishing activities. There is also an edge distributed network onboard the fishing vessels that sends the data from the ship to shore when there is an opportunity to do so [22].

The system is being developed with help from private companies with experts in different fields from those that work within the research group. To make sure the system meets the criteria from a legal standpoint, a Ph.D. student works with the legal parts within the research group. As the fishing vessels outside the arctic are often far from shore, the system relies on satellite communication to send the findings back to the mainland.

1.6.3 Computer Security Architecture

There is research on systems such as information flow in the computer security architecture part of the research group. The research here is for practical use for information flow and adapting information flow use towards new languages—information flow controls who has access to the data.

Another task within this field is creating a system for sharing datasets for research purposes while still ensuring the datasets cannot be shared further with any party with whom the dataset owner has not shared the data.

1.7 Contributions

The project's main objective is to design and implement a self-assessment system that can help pilots assess their mental health status on aspects concerning how to operate an aircraft. The thesis only focuses on the technical parts of the system and not any health-related subjects. The different aspects that should

be checked for are fatigue levels, stress, nutrition, and a general well-being of the pilots.

There will be designed a system architecture, and a prototype of a suggested system will be implemented based on the design given in [5]. There will be a short evaluation of this system to assist with a reiteration of the design of the system based on research methodology. This system will then be evaluated and tested by end-users based on the initial idea of what the system should achieve.

1.8 Outline

The rest of the thesis is organized as follows:

- **Chapter 2** covers Background and related work. This includes a case study to find systems with similar privacy-preserving requirements and existing systems that pilots use.
- **Chapter 3** covers the requirement specifications of the system. The requirements covered represent both IMReady and Gearggus.
- **Chapter 4** covers the system architecture design of IMReady and Gearggus. The architecture behind both the systems is the same, with similar requirements covered in chapter 4.
- **Chapter 5** covers the design, implementation, and a small evaluation of IMReady, which are based on the findings in "Determine Readiness to Fly: Pilot Risk Management"[5].
- **Chapter 6** covers requirement changes, design, and implementation of Gearggus, which is adjusted from IMReady.
- **Chapter 7** covers an evaluation of Gearggus and a summary of how the different non-functional requirements given to the system were solved.
- **Chapter 8** concludes the thesis, where all the different findings are summarized. As the thesis covers prototypes, there is future work to be done with Gearggus, which is also listed in chapter 8.

/2

Background and Related Work

Since the first iPhone by Apple was released in 2007, smartphone technology has increased to make the workday more efficient. The rise of smartphones, helped by many companies since then, has contributed to shaping how we use the technology today; they are being used as tools to assist with different tasks throughout the day. Today 96% of Norway's population has access to a mobile device [23] which does make it a good tool for self-monitoring and self-assessment.

Within the medical field, there started to be developed self-assessment mobile applications in 2007[24]; the early versions of the self-assessment mobile application were used for home monitoring of symptoms in patients. Since then, a lot has changed regarding the capabilities of the different self-assessment and self-monitoring mobile applications. The technology is widely used in health applications (mApps), mental health applications (Mhapps), training, and educational fields.

However, there is not much research on self-assessment mobile applications for pilots in the aviation field. There are proposed systems to help manage pilots' well-being and work-related stress,[2], but any version of the system has not yet been released. There is already a robust performance monitoring culture within the aviation industry, but most of this is focused on monitoring

the aircraft instead of the pilots [2].

This chapter outlines an overview of current self-assessment tools used by pilots, self-assessment, and self-monitoring mobile applications in various fields, how they perform, and some security and privacy measurements. We will review general system components for mobile applications and data storage and describe how they can be implemented using frameworks.

2.1 Pilots self-assessment systems

For pilots, there are a few self-assessment systems that are used. The typical outlines for these are that they are, for the most part, used as guidelines rather than actively used. The different systems are also based on checklists covering different parts, from how the pilots' health shape to flight conditions. Checklists are a commonly used tool within aviation to use both before the flight and during flight and maintenance. The pilots are therefore used to using checklists to monitor performance through checklists.

The different checklists cover some of the rules for pilots on when they are legally allowed to fly and other aspects of their health and well-being. Some also cover how their flight is and their flight planning. Topics such as the conditions of the airports and how familiar they are with the airport, and flight paths to the airport are covered.

2.1.1 "IM SAFE" checklist

The "IM SAFE" checklist[26] is a tool that checks how ready the pilots are to operate an aircraft. The "IM SAFE" checklist is more of a mental process-based checklist than a checklist where the instances are checked off. The checklist is used more as a guideline or reminder than actively used due to the lack of checkmarks. It covers areas for both legal reasons, such as alcohol, illness, and medication, and health topics such as stress, fatigue, emotions, and nutrition. The different parts of the checklist mean the following:

Illness Pilots do have a medical examination by an aviation medical examiner every three years[27]. Nevertheless, this does not cover illnesses such as cold or flu. Illnesses can cause problems for the pilot as they can get problems equalizing the pressure inside the ears. As such, their focus can be redirected towards the pain rather than focus on the operation of the aircraft[26].

Figure 2.1: The "IM SAFE" Checklist[25].

If the pilot does develop an illness during the five years that would prevent them from obtaining a medical certificate, the pilot is also not allowed to fly. The pilot is responsible for the flight, including their health is up to par during the flight.

Medication If any pilot has any illness, they might take medications to help cure it. If any medications are being taken, the pilot should contact their aviation medical examiner to check whether the medication has any effects on operating the aircraft or not. If the medicine taken, has a risk for flight safety, the pilot also has to be aware of any residual effects, which can last for some time after the medication has stopped. The FAA, therefore, recommends waiting five dosage periods after the medication has stopped before flying again [26]. For example, if the medication were taken once a day, the pilot should wait five days before flying.

Stress There are not all kinds of stress that are bad. There are stress factors that can be fun, exciting, and motivating. This kind of stress both help to raise the energy level as well as heart function and stamina and strength[28]. If there is good stress, our thinking and mental ability get enhanced. On the other hand, distress can cause anxiety or a feeling of displeasure. As distress normally has a bad impact on our performance, it can, in the worst case, cause detrimental damages for pilots.

How stress is perceived differs from person to person and how the stress is managed. When stress occurs, the body tries to overcome the source of the problem and uses all its resources to do so. If

the body can not win over the stress, it may lead to factors such as depression, hypertension, and coronary diseases. While if the body does win over the stress, the body will recover afterward [28]. There are three main categories of stress to be aware of:

Physiological stress is stress that occurs when the human body fails to respond appropriately to the physical demands that are placed on it. Factors that can lead to physiological stress are fatigue, poor physical condition, hunger, disease, working irregular hours, and a change in the sleeping schedule[2, 9]. For pilots working irregular hours, a change in the sleeping schedule does happen quite often, as the travel pattern of the world is in constant change.

Environmental stress are factors that are around us that we cannot control. For example, if the neighbor plays loud music while you are trying to sleep, it may become a stressor. Some other examples are if there is cold at the office, it may also become a stressful factor if there is no heating that works. If a pilot has to dive deep because of loss of pressure inside the cabin but without access to oxygen, it would also be a stress factor[29].

Psychological stress comes from areas that impact the mind. Common stressors for psychological stress can be areas such as conflicts at home, new demands at work, financial trouble, loss of someone close, health problems, moving to a new place, or exposure to one or more traumatic incidents[30].

From the categories above, we can observe that many different situations can lead to distress. A common effect of the different stressors is headaches, sleep disturbance, and digestive issues. Suppose the stress lasts over an extended period. In that case, it may cause high blood pressure, weakened immune system, and can cause depression, confusion, and anxiety[30].

Alcohol affects different components which are needed for a safe flight, such as the brain, eyes, ears, motor skills, and judgment[31]. The effect of alcohol is on the central nervous system. The parts that are affected by alcohol are, therefore, systems such as skilled performance, analysis of information, the control of movement patterns, and short-term memory[32]. All of these skills are used by pilots, and as such, the pilot should not be affected by alcohol during the flight.

The regulations regarding alcohol consumption within Europe should be at least eight hours from bottle to throttle. However, the blood

alcohol concentration should not exceed 0.02%, and no tasks related to operating an aircraft should be done while affected, including flight preparation[31]. The general recommendation is that at least 24 hours from the latest alcohol consumption until operating an aircraft again, as being hungover has some of the same effects as alcohol.

Fatigue is a non-specific symptom because it can be caused by many different conditions such as physiological stress and excessive muscular activity. It can also come from different medical conditions such as infections or bacterial and psychiatric disorders such as depression and anxiety. Fatigue may also come from a constant change in sleep schedule, excessive alcohol or caffeine consumption, or delayed effects of a traumatic experience[33]. Fatigue, in general, negatively impacts work performance, family life, and social relationships[34].

Fatigue is a self-reported condition, as there are no tests to find if a human is fatigued or not. As so many different things cause fatigue, measuring it is a complex topic. General aviation pilots change their sleep schedules frequently, as their work hours can change from early mornings to late evenings in Europe. At the same time, other parts of the world have departures during the night. Another issue that pilots from the northernmost part of the world experience that causes fatigue is the polar nights or midnight sun. This is because it is tiresome to be dark all the time, or it can become harder to sleep due to it not being dark during the night. Crew members are as follows affected by prolonged fatigue where it has been reported that between 75% to 91% of the participating pilots in a study were affected by prolonged fatigue [35].

Emotions / Eat: The last letter of the checklist can mean different things based on who made the checklist, however, both parts are important in some of the earlier parts of the checklist we have covered. The E can either mean emotions [26] which is a sort of psychological stress factor, or it can mean Eat[36], which is important both in physiological stress and fatigue. As such, we will cover both areas here.

Emotions in this checklist are when there are negative emotions that could lead to danger in the flight, such as depression, anger, or impatience. These are all rather hard to say a go/no-go decision for, and it can be challenging for the pilot to analyze the emotional state[37]. Suppose these parts are not respected enough, it can

have severe consequences, as happened in the Germanwings flight 9525, where a pilot ended up crashing the plane for suicidal reasons, which stemmed from depression[38].

When considering the nutrition and eating patterns, as pilots are working early mornings and late evenings, their eating patterns are constantly changed. As such, their fatigue may arise due to the change in eating behaviour[9]. Nutrition can also be used to reduce the fatigue levels, where a healthy nutrition pattern can have positive effects with reduced fatigue [39].

As we can observe from the checklist, many of the different parts interact with each other. In a study conducted by Bandeira et al.[40], they observed that 80% of the errors that occur with a flight are due to human errors. These errors stem from stress, fatigue, emotional state, sleeping patterns, and physical state. These are all factors that can be made aware of with a self-assessment system and let the end-user know the different risk factors. Everyone working within aviation is not included in the European regulation for working hours[4]. As the pilots are exempted from the European working hours, they have less predictable working hours than most other lines of work. As their working schedules change a lot, there is a constant change in their sleeping schedules and eating patterns. Therefore, monitoring the sleeping hours and quality might help ensure everyone gets enough rest.

2.1.2 PAVE Checklist

Before every flight, as part of the preparation for the flight, the pilots must evaluate the different risk factors associated with the flight they are about to fly. The PAVE checklist[37] can be used to evaluate the different topics that concern the flight. The parts covered in this checklist are Personal/Pilot, Aircraft, enVironment, and External pressures, representing different risk factors for the flight.

Personal/Pilot is a personal health check before the flight to check that the pilot is healthy. In order to check for this part of the checklist, the "IM SAFE" checklist is used, which is covered in more detail in chapter 2.1.1. What is checked for here is if anything is happening in the pilot's personal life which can danger the flight or if the pilot has taken any substances that will affect the pilot's performance. It is also checked for if the pilot is fatigued and therefore can not perform to the same standard as they usually would do.

Aircraft is concerned that the pilot has checked for the aircraft limitations

and decides if the plane can manage the flight or not. The pilot should also calculate how much fuel is needed for the flight, that the aircraft is airworthy, that it has all the proper equipment needed to perform the flight, and that the pilot is certified on the aircraft[37]. For example, if the weather conditions determine that the flight has to be carried out using instruments only, the required instruments are available to the aircraft. To evaluate if an aircraft is airworthy, it has the following maintenance schedule presented by an EASA part-m[41], which is carried out by an EASA part-145 organization[42].

The pilots should also perform a visual assessment of the aircraft to confirm there is no damage to the aircraft since the last inspection was performed. There should also be checked that the correct documents are on board the aircraft; these are Airworthiness Certificate, Aircraft Registration Certificate, Operating limitations, and Weight and balance sheet for the aircraft[37].

The **enVironment** part of the checklist is not something the pilot can mitigate, as bad weather conditions will always be bad. This part is mainly concerned with parts around the flight, such as the weather and airports. It is also concerned that if there are Instrument Meteorological Conditions (IMC), the pilot is proficient in flying by the Instrument Flight Rules (IFR). Another part is that if there is a divergence in the flight path, this could lead to uncertainty for the flight.

Suppose the pilot is not familiar with the airport. As pilots may have departure or arrival at foreign airports, this may also affect how the pilot feels about the flight. It may cause additional pressure when listening to the airport's Air Traffic Control(ATC), where to tax the plane, or not being sure about the airports terrain.

The environment is also concerned with how comfortable the pilot is. If it is cold or hot inside the cockpit, the pilot might lose attention to the tasks performed. Suppose the pilot, for some reason, needs oxygen, or there is a situation where additional oxygen could be preferable but is lacking. In that case, this can also affect the performance of the pilot.

External pressures are topics related to if the pilot is stressed, anxious, or feels forced to do something. If the flight is delayed, the pilot has to make up time during the flight or a passenger making trouble can both lead to stressful situations. If the pilot lacks the skills for

the flight but does not say it to the colleagues, that can also lead to stress. The pilot can also be anxious about losing their job due to the ongoing pandemic, which puts additional pressure on the pilot.

If the pilot is a firefighter pilot and is sent on a dangerous mission, or a civil pilot has to take additional flights due to a lack of personnel, external pressures can also be set on the pilot.

2.1.3 Federal Aviation Administration Aviation Safety Team Flight Risks Assessment Tool (FAAST FRAT)

FAAST FRAT is an automated spreadsheet where the user can cross off what applies for the flight. In the example set up provided by FAA, there are items for if the pilot has slept for more or less than eight hours, how the day is generally going, and how the flight conditions are. When the spreadsheet is filled in, it will give a result back with a number that indicates the risk factor involved with the flight[43]. The sheet was developed during a safety campaign FAA had in 2016 to make awareness of the extra safety in the aviation industry as a whole[44].

FAA has defined three different levels of risk in the spreadsheet, with the lowest being "Not a complex flight", the middle being "Exercise caution", and the highest risk factor being "Area of Concern". Even if the score is the lowest, there will always be some risk involved with a flight[45]; however, the score indicates how much additional risk there is. If the risk lands in the highest area, there should be ways to reduce the risk levels, or the flight can be delayed to another time if possible.

The sheet that FAA made is regarded as an example of a sheet that the General Aviation Operators can adjust to how it would fit them better[43]. There are, however, no guidelines on how to adjust the sheet to make it fit each instance better. In the example sheet presented in figure 2.2, the lowest risk zone is between 0 and 9, the middle between 10 and 19, and the highest from 20 and onwards. However, if the sheet were used for a flight academy, some additional questions would be needed to change the number rating on the risk factors. Different questions would be needed if the sheet were used in an airline, while a private flight would require a third set of questions. The FAAST FRAT sheet can be adjusted to fit the three different types listed or if there are any additional requirements.

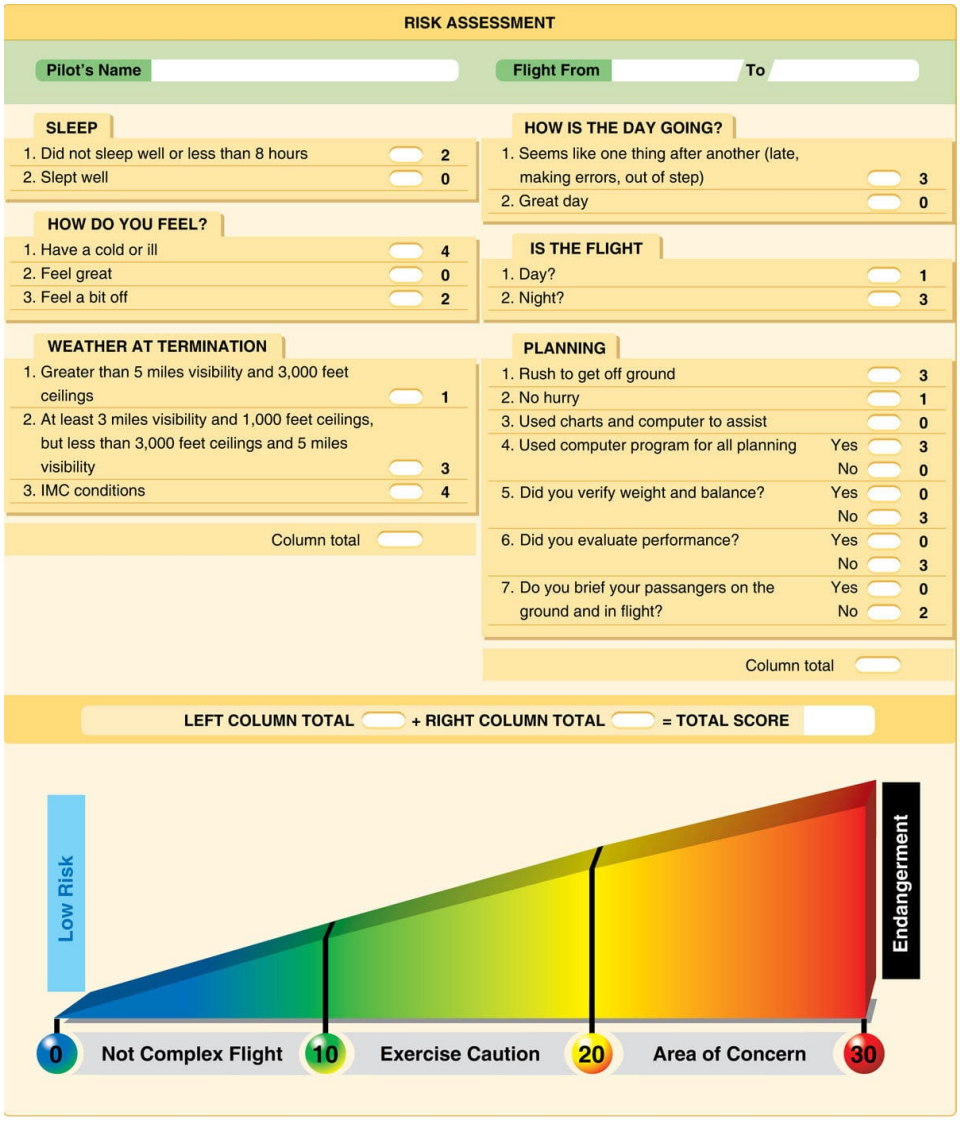


Figure 2.2: An example of a quick and easy FAAST FRAT setup[44]

2.2 Self-monitoring through use of mobile applications

There is no mobile application within the aviation industry to either monitor or help to self-assess how the health conditions are before a flight. There are, however, mobile applications with similar capabilities or requirements in other fields. Since the first iPhone was released in 2007, the health sector has used mobile applications as a tool to self-monitor and self-assess the patients after operations[24].

While mobile applications do different things, the health sector has similar requirements storing user data as the system described in this thesis has. We will also look at a performance system, which evaluates data to come up with recommendations to players and coaches to prevent injuries. This part is used to understand better automatic feedback systems based on the user input on the system.

2.2.1 Player Management System

The Player Management System (PMSys) is a system that originally were developed as a master thesis project [46, 47, 48, 49]. It has later been redeveloped in order to comply with the GDPR.

The system itself is a mobile device-based athlete monitoring and reporting system. It is based on injury prevention and readiness to train and play. The players can use a mobile application to report how their training session has been and if they have any injuries anywhere. The coaches can then access the data the players have reported in a web portal. They can there gather information about both the whole team or single-player use.

PMSys also supports injury prediction, to be able to predict when a player is about to get injured to be able to prevent it. The injury detection has been proved to be quite effective in a limited test group[50].

PMsys utilizes the mobile device as a primary node, where the questionnaire is saved in a small SQLite database. The data on the mobile device is then synchronized with a centralized server upon updates and when an Internet connection is restored. As the storage on the server is a critical component for the system, it is kept minimal and efficient and only interacts with a PostgreSQL database. The server replicates the data located on the user's mobile device. The user ID comes from a third-party OpenID Connect (OIDC) service to keep the users anonymous. The users are then only granted access to data where

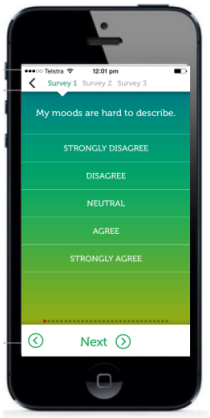


Figure 2.3: Example question from MoodPrism [51]

their authentication token matches the user ID on the server.

2.2.2 MoodPrism

MoodPrism is a mobile application that offers self-monitoring of mental health. The mobile application has an automated feature that tracks the music played and reads messages sent on social media. This data was processed locally on the mobile device and never sent to the server. The only part that is uploaded is an aggregate word count from Facebook and Twitter posts[52].

The mobile application also has an experience sampling part, where they offer a short set of questions to the users daily. The questionnaire is delivered at a random point during a pre-defined time frame of the day, for example, between 09:00 and 21:00. Suppose the notification of filling out the questionnaire comes during a busy time of the day. In that case, the questionnaire can be filled out at some point during the day before midnight or ignored altogether. The questionnaire is then stored on the mobile device before being sent to a central server once every 24 hours.

All data that can identify a user is anonymized, where the mobile device is the authentication token. The user can access the data through a Secure Shell (SSH). When the data is sent from the mobile device to the server, the data is encrypted during transfer. The server has a firewall, and security protocols are also updated regularly.

When evaluating the application in [53], they experienced that their users had more significant decreases in depression, anxiety and greater increases in mental well-being than the users who had positive and engaging experi-

ences using the MoodPrism. This suggests that using a mobile application for self-monitoring of mental health does help the users if they have a positive engagement with it.

2.2.3 MoodMission

MoodMisson is a mobile application that monitors and raises the self-awareness of mental problems the users are experiencing[54]. The mobile application is free of charge and open for everyone to use. The application was tested with stakeholders who received feedback during the design process and users who found the mobile application on Apple's app store. During testing, it was found that the mobile application increased emotional self-awareness and reduced depressive symptoms in depressed participants compared to a control group.

The way the mobile application is designed is to let the user engage with the mobile application when they are in a bad mood and gain tasks to improve it. The user is then given missions to cope with the bad mood. These are designed to ensure the user does not lose confidence and disengagement from them. They are, however, not a perfect solution, and the designers do not think it is the definite solution, but instead a suggestive activity to help out. There are also designed many different missions to give the users different options for coping with their cognitive behavior; with this, there is an attempt to provide as many coping strategies as possible without over-complicating user engagement. There is also taken care to avoid impressions of an expected result, so badges are rewarded for completing a mission rather than experiencing decreases in distress.

As the mobile application is intended as a coping solution when the user experiences a bad mood, the designers had three different primary aims with the mobile application:

- Provide self-administered prevention and self-help strategies to reduce the risk of clinically significant mood and anxiety disorders.
- To support stepped-care interventions as a platform for access to low-intensity intervention for low-level clinical symptoms or subclinical symptoms of depression and anxiety.
- As an adjunct to psychotherapy or other face-to-face treatments for mood and anxiety disorders.

As the mobile application is designed for ages 12 to 70 years, the mobile

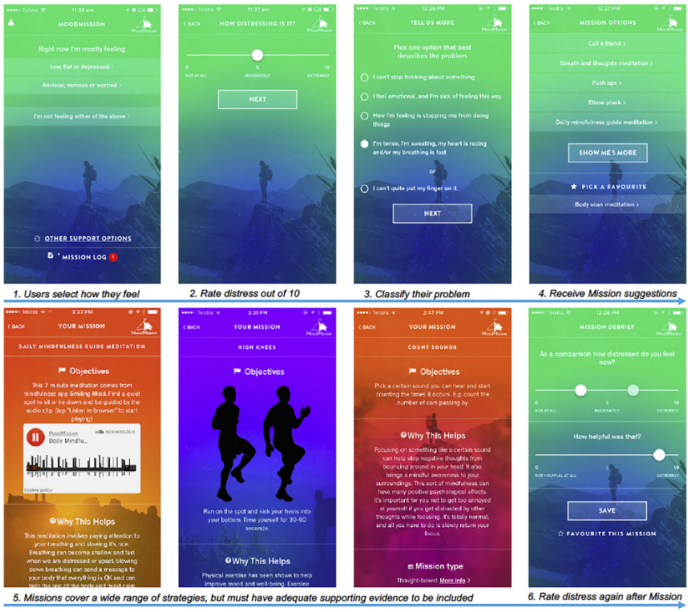


Figure 2.4: Different sample screens displaying how MoodMission looks.[54]

application has to be easy to use while still engaging. To increase engagement, the missions are gamification, which lets the users be more interactive in them. There is also listed a distinct purpose of why the mission should help cope with the bad mood the user is experiencing when interacting with the application. If the user has not interacted within two days, they receive a push notification to make the user aware of the mobile application again.

The mobile application stores data from the users to provide a history of when the user experiences terrible moods and login data. User data such as email and passwords are stored on a Google Firebase Database. When the data is stored, it is scraped for any data that can identify a user. The different missions recommended for the user are stored in a frontend service created with WordPress and stored on an Amazon Web Service server. It is harder to identify a user as there are competing cloud providers as storage providers. The front end of the mobile application was initially designed for use with iOS, and after the testing started, it was modified to be compatible with Android users.

2.2.4 CrowdHealth

CrowdHealth is a protocol that was formed based on the GDPR rules when handling health data[55]. For authentication, authorization, and access control,

they use Privacy Enhancing Technologies (PETs). For user authentication, the protocol relies on OIDC. This scales well as it is lightweight and flexible. They use OAuth 2.0 for authorization; this allows third-party access to their resources without performing authentication and maintaining user credentials. A benefit of using OIDC instead of using something own is that the complexity is handed over to the authentication provider instead of the system and client dealing with it. OIDC also supports mobile devices and web browsers so that the system can be run anywhere.

CrowdHEALTH offers management and analysis of anonymized health data from various data sources. It uses an Attribute-Based Access Control (ABAC) to use effective access control policies. Through ABAC, the system can access Read-only, Write-only, Read-Write, and admin access to the resources.

When CrowdHealth removes data that can identify a user, it is done in two steps. The first one removes data such as names, social security numbers, email addresses, ID Card numbers, addresses, and phone numbers. The second step aims to pseudo-anonymize the indirect identifiers so that the user's privacy stays secure. To suppress any indirect identifiers, the numbers are replaced with an asterisk. This is done at the data source to ensure no leaks during the data transmission to any other parts of the system. Each user has their data stored in the database with a random user ID number to preserve privacy.

2.3 Privacy-preserving systems

As the system will collect a lot of personal data, access control to whom can view the data is crucial. This section discusses the different systems to grant access control to the data. The different components that are explained are tied to each other in that they are dependent on each other to work. Lastly, the provider of the access control system chosen will be described.

2.3.1 OpenID Connect (OIDC)

OIDC is an application identity layer built on the OAuth 2.0 framework. This allows a user to log in on a third-party application and use an authentication provider to verify the user's identity. By using this identification method, there is not stored any user data on the servers, like names or email addresses. The authorization tokens are sent through a JSON web token.

An authentication provider is a third-party provider like Google, Microsoft, or Facebook that can authenticate users with their login credentials in the

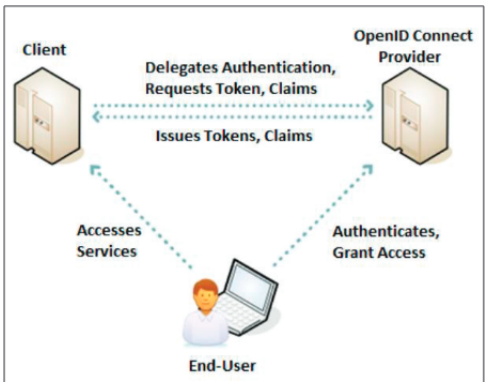


Figure 2.5: Illustrates the execution flow to OpenID Connect [55].

system.

Using OIDC is for user authentication; it gives the user one login for multiple sites. For example, if an application that utilizes OIDC to log in, the user can then, for example, be sent to Microsoft or Google, log on with their login credentials, then a user authentication code is sent back to the application, which the user did want to log in to.

Some identity providers also utilize cookies to help make the authorization process faster[56]; instead of then having to do the entire process each time, the authorization token is stored in the cookie which the provider then gives to the application.

As there is over 60,000 different application that utilizes OIDC[56], which means if there are any security threats that are identified, there are many developers who will be interested in fixing the threat. During a threat review done for the OIDC protocol, it was found that if the users do not grant attackers access to the phone, for example, through malicious applications, the protocol is secure but does say there is a need for further investigation[56].

2.3.2 Open Authorization (OAuth) 2.0

OAuth 2.0 is a framework to help with authorization for web applications and native applications. It enables applications to gain limited access to user accounts on an HTTP service. It does so by delegating user authentication to whoever hosts the user account and then authorizing a third-party application limited access to that user account[57].

When a connection starts, the application must be registered with the server

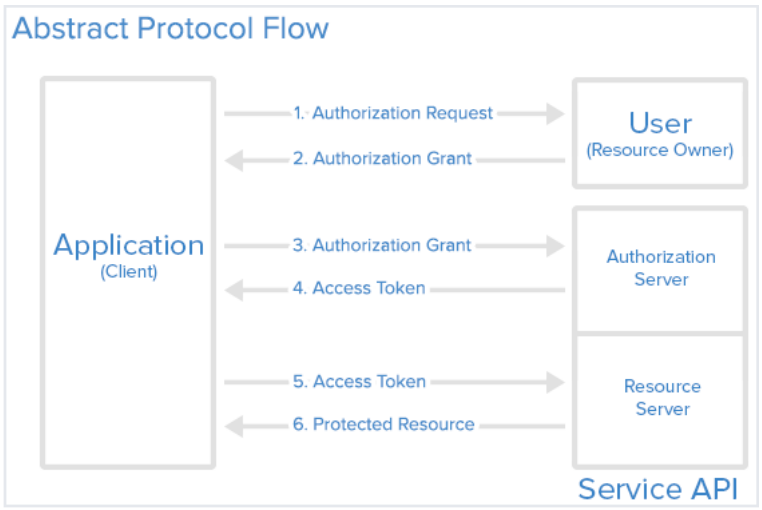


Figure 2.6: Illustrates the the different components in a OAuth2 system interacts with each other[58].

providing the user details. After being registered, the service gains the application name, application website, and a redirect URI from the application. When an application is registered with a service, the service sends client credentials back to the application. These are client identifiers and client secrets and are sent in a public string, which is used by the service to identify the application[58].

The most common use of OAuth 2 on mobile devices is through browser redirection. The application utilizes the mobile device's browser for the authentication service. The user can log on to the service before it is redirected back to the application with an authorization code. Once the application has the authorization code, it can ask the authentication service for an access token, granting the application access to the user resources hosted at the resource server.

An area where the framework can be attacked is if there is a malicious application running on the user's mobile device. The attacker can steal the user's credentials through the web browser; this threat does exist in both the embedded web browser or system native browser[59]. The credentials can be stolen through the web browser through malicious javascript code that can steal them when the submit button is clicked. In terms of the system's native browser, the application can present its UI instead of the user authentication UI. In the paper [59] there is a proposed solution to how to solve these security issues.

2.3.3 Firebase Autentication

Firebase is a backend service for implementing, among others, authentication using OIDC. The way it works is that a user logs in using an identified provider. Bypassing in the user credentials to the Firebase authentication service, the service then verifies the credential passed on and returns a response to the client. For an application to be able to use Firebase, the application has to be registered in its console. This generates a return address for the different identified providers after a user has been signed in[60].

Firebase was developed by a Google security team that implemented their password manager and Sign in systems. While Google backs it, it is an open-source project where all its source code is open for anyone to review. This enables more people to find potential security weaknesses and fix them quickly.

As Firebase typescript is implemented by using web tokens, and as discussed in chapter 2.3.2, this is one of the weaknesses of using OIDC in native applications; there are other open-source projects which enable the use of Firebase on the native layer.

2.4 Storage Systems

The storage systems used have as requirements: they have to be efficient in retrieving data from storage and not use much extra space to install the database systems. In this section, both the cloud database and local database, which will be used on the system, will be described.

2.4.1 PostgreSQL

PostgreSQL is an open-source database management system(DBMS) that has been community developed over more than 20 years. It has been developed over a long time and has high resilience, integrity, and correctness. It supports both relational (SQL) and non-relational (JSON) queries, which allows it to link with other data stores[61]. PostgreSQL has write-ahead logging, which makes it a highly fault-reliant database system. It has performance optimization features, which generally are only available in commercially developed DBMS.

As PostgreSQL is open source, this also means that the source code is open for anyone who wants to adjust it to their own needs. If any systems have any special features needed for their DBMS, the fact that PostgreSQL is so customizable will be an advantage of the system. It also has a low maintenance

cost in using it, which, if it is used in a system where new developers may take over the project in the future, is an advantage[62].

In a performance evaluation done towards MongoDB, it was observed that PostgreSQL has a lower average response time for both indexed search and non-indexed search. This remains true on both single-node and multi-node databases, which proves that the performance optimization features are working as intended[63].

2.4.2 SQLite3

SQLite3 is a library that installs a self-contained and zero-configuration DBMS system. It is self-contained as it has very few dependencies. As SQLite is zero-configuration, there are no additional installations required to run the DBMS. The source code for SQLite is publicly available and free for anyone to use. There are no additional server processes to access the stored data, so it is a popular choice for application usage. There is no additional space required for the DBMS, as it reads and writes directly to the disk[64]. Even with this, the performance of the DBMS is still high.

The DBMS development is done by a team working full-time to develop SQLite. This ensures that the performance and reliability of the system do increase as time goes by. The DBMS is also tested a lot during development to ensure it stays reliable[64].

The lite part of the name refers to the DBMS being lightweight in terms of setup, administration, and resource usage. SQLite does not require a separate server process to operate. This lets the application communicate directly with the DBMS instead of communicating through a client/server connection[64].

2.5 Server hosts

The server host in the system is used to host the database system. Therefore, it should be a secure host, both fast and available when data storage is needed. As the system scales, there might be a need for a server to perform calculations in the future, and as such, the server host needs to support the option to do so. As listed in [65] the transfer of data should be encrypted when dealing with health data, and the server needs to support this as well. This is both to ensure the user's privacy and ensure the data is not modified while transferring.

2.5.1 Azure

Azure is Microsoft's cloud computing platform. Azure supports, among others, cloud computing and cloud storage for data. This enables applications to do some sort of computation on data stored in the cloud. When computing with Azure, a role has to be assigned before Azure does the load balancing between nodes to perform the task[66]. Depending on the task given, the roles assigned are either a web role, worker role, or VM.

Azure storage is not just limited to its storage systems, consisting of blobs that can be configured to hold whatever data, with additional information about that data. It can also use storage systems such as relational databases. When storing data in Azure, the data stored is replicated three times[66], all in close proximity to the region chosen as the primary storage node, for lower latency during the transfer of data.

A fabric is used to determine the different sizes needed for the application, as well as optimize the hardware of where the new application run, to make applications less affected by others running on the same server [66]. A fabric controller is responsible for placing the tasks where they should be stored, and it seems a storage unit as an application and can therefore place it where it would be the most effective to run from.

When using Azure for data storage with an open-source type of database, there are two main options of where the database is placed. Azure offers to use a Virtual Machine (VM) where the database can be set up, with complete control over all the configurations that are done to the database[67]. If a VM is used, the database has to be installed on the server before implementing all the different security measures required for updating the database and during the transmission of data between a client and the server. The other option is a pre-installed instance of the database. With this solution, Azure handles all updates to the database and has all the security measures installed.

Figure 2.7 depicts how the architecture of a flex server instance of Azure is set up. The user pays for the space required to host a server, database, or VM instance on a flex server instance. The data is stored at the user's availability zone, which is selected when setting up the server. The data on the server is replicated to three other instances to ensure no data is lost if a server breaks down.

There are different databases to choose from depending on the size requirements needed for the particular database. The other option that can be used is to use a pre-installed database instance. When choosing a pre-installed instance, Azure makes sure the instance is up to date with the latest available

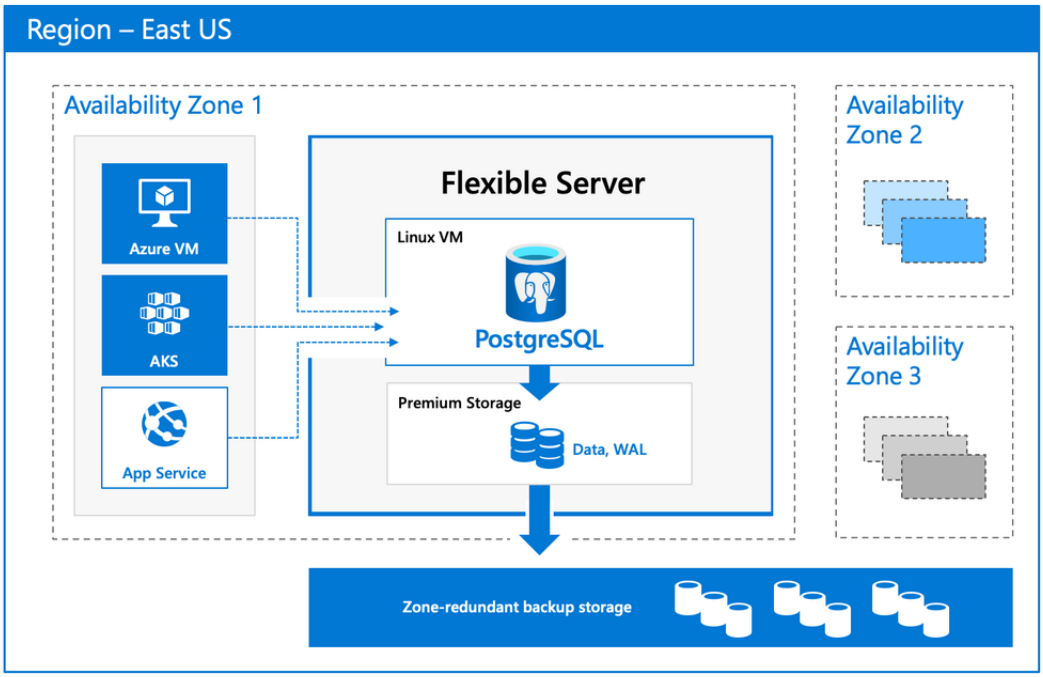


Figure 2.7: Displays the architecture of Azure PostgreSQL server[68].

version and the security measures used to protect the data. For transmitting data, Azure supports SSL.

Secure Socket Layer (SSL)

The secure socket layer is a security protocol developed by Netscape, which provides secure communication between a client and server[69]. It provides security when the client contacts the server; it exchanges a public key with which the data is encrypted. The public key is located in a certificate database, which can guarantee the client that it gets the correct key, and no middle in the man attack has to occur while exchanging keys. As SSL can guarantee security while transmitting data, it has become the standard Internet protocol for secure communication. There has been an increase in the HTTPS protocol, which uses SSL as the security measure.

SSL does not require any changes in the application layer while sending but instead sits between the application and transport layers. The transport layer is where the Transport Control Protocol/Internet Protocol (TCP/IP) resides, governing data transport over the internet.

The SSL protocol has two sub-protocols, a record, and handshake protocols. The record protocol defines the format which is used to transmit data. The handshake protocol deals with how the client and server communicate before data transmission can start. This is designed to be done in the following order:

- The server is authenticated to the client.
- A cryptography algorithm is chosen, which both the server and client support.
- a Public key is used to encrypt a shared secret, used for communication.
- An SSL encrypted connection is established.

The public key, which is used for encrypting a shared secret, is found by using a certificate, a digital document attesting to whom the public key belongs. A certificate can be found by contacting a certificate authority where each is stored. When a certificate is used, it can be guaranteed that there is no phony key sent with a man-in-the-middle attack.

By utilizing SSL when transmitting data between a server and a client, the following benefits are ensured:

- Authentication: The client can ensure that it is the correct server to which the data is sent.
- Message confidentiality: All the data is encrypted during transmission using a session key, and no third party can intercept it.
- Message integrity: It can be ensured that the correct data is stored and received.

2.6 Summary

This chapter has covered some existing tools the pilots can use to help with self-assessment of whether the pilot is ready to fly or not. The existing tools focus on both the pilot's health and how focused it is estimated that the pilot can be with the current conditions of fatigue and stress levels. Another part covered in the existing tools is the preparation the pilots do before a flight is performed if all the correct documents needed for the flight are present.

Other systems have been observed which handle user and health data to gain insight into systems with similar privacy requirements and how these are built. Of the systems that have been observed, one focuses on professional athletes and how they are training to prevent injuries. Two different mhApps, which collect information about users and give feedback on tasks that can be done to improve their mental health, have also been observed.

There has also been given background information on the privacy-preserving systems used in the system. Different frameworks and protocols are utilized to preserve the user's identity while still being unique in the system. The different storage systems are also described, with how the DBMSs work.

The cloud service provider (CSP) describes how data replication is handled to ensure fault tolerance and how the CSP determines the size and hardware to run on. The CSP offers two different options on how to store data described and how privacy is ensured during data transfer between the mobile device and the CSP.

3

Requirement specifications

This chapter will describe the system requirements based on the problem definition in chapter 1.3 and the background knowledge shared in chapter 2, Background and Related Work. The functional and non-functional requirements and the overall conceptual system model will be outlined.

3.1 Functional requirements

For the pilots to track their conditions before a flight, a few key features have to be included to get the system to work.

- A questionnaire for the pilots to fill in to collect the data.
- Sending the data from the client to a server.
- Presenting the data to the end-user in a graphical way.
- Storage of the data.
- User authentication.

The questionnaire for the users to fill in is for collecting the data needed to give the pilots a result based on the information given to them. The data will

be collected on a mobile application, where the user can fill in a questionnaire for further analysis. The questionnaire will contain questions based on the research done in "Determine Readiness to Fly: Pilot Risk Management"[5]. These cover all the parts of the "IM SAFE" checklist [26] as well as some extra for military usage. The questionnaire will be separated into different parts to avoid the process being too tedious a task to be done. The information can then be given when it applies to the user.

The data transfer should be made secure, as the data contains health information about the user. However, even if the transfer is secure, it should not cause the system to slow down for the user, as the user should not have to wait for the system to transfer anything. The proposed solution to this is based on the updated PMSys[50] scheme, where the mobile device is the primary storage node of the data. The system is based on a mobile application; the data should be stored on a separate server and the device. At the same time, the server acts as a replica. This way, the user can fill in the questionnaire while the device is offline and send the data when it is possible. All the information which can identify a user should be removed before sending any data.

In order to preserve the identity of the user while being able to separate the different users, OIDC will be used. The user will be identified on the system with a randomly generated number by the authentication services. This makes it so no user information (names, emails, addresses, and phone numbers) has to be stored on the server. While the authentication services do know that their users are using the particular application, they do not have access to the personal data which are used[59].

When the data is presented in an easy-to-understand graphic, the user engagement with the application is believed to increase. As it should not be a requirement for the user to understand health data, the data should also come with a suggestion as to whether it is safe to fly or not.

3.1.1 Frontend

Through a graphical interface, the end-users interact with the system. The input data from the user will be in the form of a questionnaire they fill in daily. The results of the data given by the end-user in the questionnaire will then be presented graphically. This gives the end-user insight into the data stored about them and can make better judgments by gaining this information.

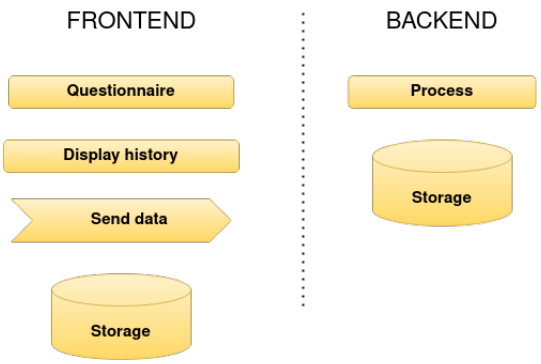


Figure 3.1: Illustrates the architecture and how the part interacts with each other.

3.1.2 Backend

As the mobile device is the primary node for data storage, the backend server will be a replica node, where the data is received if the mobile device has had an update and the internet connection is restored. The backend will be where the replica data is stored and where data processing happens. The backend will also handle any machine learning parts based on the data given by the application to give accurate advice if the end-user is ready to fly or not.

3.2 Non-functional requirements

In the following section, the non-functional requirements that are needed to develop the system per the problem definition stated in chapter 1.3 will be discussed.

3.2.1 Security & Privacy

The system is based on sensitive personal data; security and privacy while handling the data are therefore crucial. The end-users also expressed concerns about letting their supervisors know about the data that is stored about them[5]. To ensure each user has access to their own data only, login with OIDC will be deployed based on the OAuth 2 framework. By doing this, there is no information stored concerning to whom the data belongs. The sending of the data between the mobile device and the server should be via an encrypted connection to hide the identity during transfer. In order to achieve this, both a Secure Shell (SSH) encrypted connection and an HTTPS connection can be used.

3.2.2 Reliability & Availability

As the users are busy people, who are likely to use the mobile application just before a flight, the mobile application has to be available when the user needs it. In order to achieve this, the questionnaire to be filled out is stored on the mobile device, and there is a small SQLite3 database on the mobile device to store the data.

3.2.3 Scalability

The system must be able to handle an increasing number of users without severely impacting the system's performance. The issues surrounding this will be storing data and letting multiple users update their data at once. The testing of the system will be done in a limited group, so the extension of storing a lot of data and allowing many users to access their data at once will be considered for future work.

3.2.4 Fault-Tolerance

As the end-users do not want to wait for any downtime in the system, the system's fault tolerance should reflect this. Personal data should be stored at different locations to aid the system's reliability. If the end-user does not have access to an Internet connection, it should still be able to use the system. Therefore the questionnaire to be filled in should be stored locally on the mobile device and sent when the internet connection is restored. Suppose the mobile device gets lost or broken, or the user cannot access the mobile device for any other reason. In that case, all the data should be replicated on a server to ensure the data is not lost.

3.2.5 Dependency

This system will have a high degree of dependability on other systems, as the login system requires other systems to be available to work. However, the services the system relies on, use the services on their systems and are therefore monetarily incentivized to keep it up and operating. Another dependency required is for the user to have a mobile device on which the mobile application runs.

3.2.6 Maintainability

The system should be designed to be easy for other developers to maintain. In order to achieve this, it has to be simple for others to understand. Maintainability is essential; if the developer who initially developed the system leaves, others can further expand on the project with ease. It should be easy for a new developer to see what has previously been done.

3.2.7 Usability

The end-users of the system are aircraft pilots. As there is no requirement to be a computer scientist to become an aircraft pilot, the technical knowledge among the pilots may range from deep technical understanding to non-technical. The user interface (UI) of the application should therefore keep this in mind and be designed to be an easy understanding application. Thus, the application's design should be highly intuitive, leading to higher system usability.

3.2.8 Performance

Performance is concerned with how quickly the system responds to user action when using the system. As the end-users are expected not to be willing to spend much time on the mobile application when filling in questionnaires, the part of filling it out should be designed to be done quickly. As the questionnaire is filled out, it should be stored on the mobile device and sent in the background without any user interference for them to be sent to the server as replica storage.

3.2.9 Compliance

Compliance requirements are concerned with any law or regulations the system will encounter[70]. The system will store personal information such as sleep, medications, and illness. The GDPR will apply. There has to be ensured that no data stored on the system can be tied to any user of the system. The system, therefore, cannot store any data that can identify a user, such as an email, names, phone numbers, and addresses.

3.3 Summary

In this chapter, the functional and non-functional requirements have been listed. The functional requirements listed show how the end-users interact with the system to collect data and how the data is presented to them. There is also listed how privacy is preserved in order to ensure it is only the end-users who can access their data. There is listed a server and transfer of the data between server and client as requirements. These ensure that no data is lost if the mobile device is damaged.

As for the non-functional requirements, these list what laws and regulations the system will encounter, performance, how easy it is for the end-users to interact with the system, how to handle fault, maintain the system, and how well the system scales.

4

Design

This chapter will outline the architecture and design of the system based on the background information listed in chapter 2 and the requirements listed in chapter 3. The architecture will focus on the type of systems used, as two different implementations are described in chapter 5 and 6, focusing on the system design requirements which were needed for each version.

4.1 System Architecture

Chapter 3 outlined some requirements regarding how the architecture of the system should be defined. The architecture defined in this chapter is the required architecture which covers both the system described in chapter 5 and 6. The difference between the two systems is that the system in chapter 5 is intended for firefighter pilots, while the system in chapter 6 is for general aviation pilots.

In figure 4.1 the overall system architecture of the system is depicted. The system consists of a front-end application as shown in figure 4.1 A). The front-end application is a mobile application where the user answers a questionnaire. An analysis is done on the answers to determine how ready the user is to operate an aircraft. The system's frontend also has a database where the answers are stored.

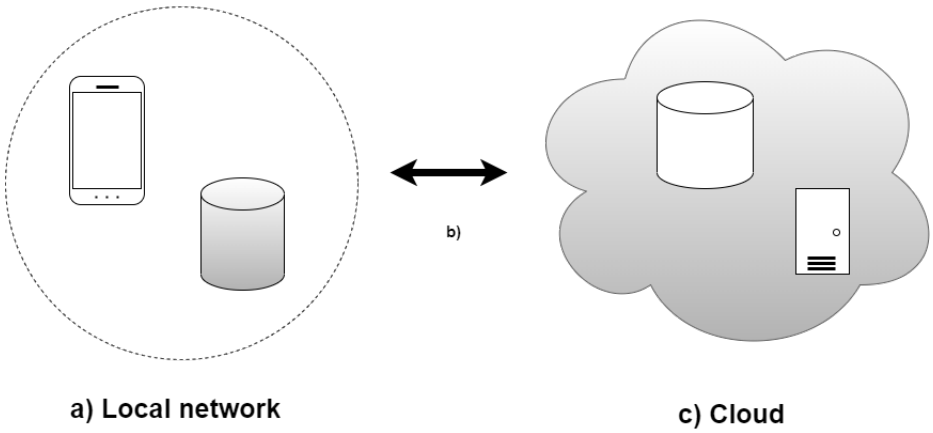


Figure 4.1: Illustrates the system architecture. a) Shows the front-end, as well as a storage node locally on the mobile device. b) shows cloud storage, as well as capabilities for computing in the backend.

Figure 4.1 B) depicts the backend of the system, which holds a cloud database to store the data to ensure no data is lost within the system. The backend of the system also has a server, which in the future can determine how ready the pilots are to operate an aircraft based on the answers given by all the users of the system.

4.1.1 Front-end

The architecture model used in the system is a client-server model, where the client provides a GUI (Graphic User Interface) for the user. The end-user will give input to the system by filling out a questionnaire, where the end-user gives information about topics such as sleep quality, sleep length, stress, and physical health. This data will be stored in a local database on the mobile device and stored in a cloud database for redundancy. The data collected will then be further analyzed to recommend how ready the pilot is to operate an aircraft.

In the GUI, a graph indicates how ready the pilot is to fly. If the end-user does not gain the highest score, a history tab shows where the user can improve to become fit for flight. The answers given to the system for the past seven days are checked on a few different parameters to determine how ready the pilot is to operate an aircraft. If the answers are the most positive possible, a green light will be returned, indicating no additional risk is associated with the pilots' ability to operate an aircraft. If a yellow light is given back, some additional caution should be expressed with the flight, as the pilot can have thoughts

somewhere else or be fatigued and lose focus. A red light given indicates a significant risk with the flight, and the flight should be considered postponed or be carried out by someone else.

The client is presented in the form of a mobile application on a mobile device. The mobile application can be accessed even without an Internet connection using this solution. If a user wants to fill out a questionnaire without an Internet connection, the data will be stored on the mobile device and later sent to backup when a connection to the Internet is restored.

A disadvantage of using a mobile application instead of a web application to access the data is that the mobile application has to cover both Android and iOS and be accessed on the mobile device. In contrast, a web application can be accessed through a computer and a web browser on any OS on a mobile device. However, the storage of data and accessibility do outweigh this, as this improves the reliability and availability of the system, being able to access the system when needed, and the storage of data even if the bandwidth is overloaded when using the system.

4.1.2 Backend

The backend of the system is a database that uses PostgreSQL for storage. This database store the data for all users of the system. To ensure privacy is preserved for the users, the system does not save any data that can identify the users. As each user on the system is represented with a uniquely generated artificial identifier provided by the login system, a query is done by using this identifier to secure the privacy by only allowing the user whom the content is intended for to be able to access the data. This identifier token is also used to separate the users from each other. It is the only form in which the users are presented on the server.

4.1.3 Access Control

As the user data is stored on the system, access control is required to access the system. The login system utilized on the systems will differ, as there is a different set of requirements given to the login system. Both systems have identical requirements in that no user information should be present on the same server as the data is stored.

Gearggus can utilize a third-party authentication provider, which stores the user details on their system. The user can then use login credentials from services such as Google, Facebook, and Microsoft, and Gearggus will receive

an ID token from the provider. The trade-off with using an authentication provider is that the provider will know Gearggus did contact them for login in about the user.

The military will, for this reason, keep a login system to themselves, where IMReady can use an existing login system the military already uses, or a new login system has to be created. Either way, the login system should still provide an ID token, which is an artificially generated identifier to preserve the user's identity. There has been a switch towards using Gearggus instead, so this has not been implemented.

4.1.4 Storage

A storage solution on the system is needed to achieve higher reliability and the need to store the data collected by the end-users through the questionnaire. The databases chosen also needs to be suitable for the device used. For fault-tolerance reasons, as listed in chapter 3.2.4, the data is replicated over different instances to avoid a single point of failure.

The system is based around the client, a mobile application on a mobile device, and a database on the mobile device are used. The database that is being deployed is a small SQLite database. This lets the mobile application interact directly with the database instead of communicating through a client-server connection. With this solution, the communication gets more effective and less workload for the user when using the mobile application. As stated in chapter 3.2.2, the users should also be able to interact with the mobile application even when no Internet connection is present. Having a small database on the mobile device will enable this.

SQLite has the benefits of being a relatively small database, making it a better use on a mobile device than some databases that may scale or perform better but use more resources. Another part of using SQLite is that it does not need much from an OS or library, making it work better in embedded devices such as mobile devices. Since there is no server connection between the database and the mobile application, there is nothing that needs to be installed or configured for the DBMS to run. This increases the usability of the mobile application.

The trade-off with having a DBMS on the mobile device is that it will use more space as time goes by, as there is more data stored. A solution to this can be deleting data after some time has passed. The deleted data from the mobile device will then have no replica and will only be stored on a server, but it solves the local storage issue on the mobile device.

The primary storage node is the mobile device. However, these might get damaged or lost. This makes a replica necessary to preserve the data and not lose any data in such occurrences. As listed in chapter 3.2.3, the system should be able to scale well without severely impacting the performance for the user. This implies that the DBMS on the server-side needs to scale well for multiple users to use it simultaneously. As multiple users are using the same DBMS system, the users need to be differentiated from each other without any personal information getting leaked. In order to achieve this part, the users are identified by their ID tokens, which are unique for each user, without giving any details on who the user might be.

Both scaling and performance are essential for the replica DBMS, as there can be many users using the system simultaneously while expecting to get results back fast. PostgreSQL has a fast response time on both a single node and multiple nodes during indexing or no index access to the data[63]. As it supports both relational and non-relational query systems, the query systems can be easily maintained in the future.

PostgreSQL is an open-source DBMS; it can be used in IMReady and Gearggus as cloud storage. In Gearggus, the DBMS can be hosted from a CSP, while in IMReady, the military can host the DBMS on their servers with complete control over the DBMS. By using this attribute, IMReady and Gearggus can be maintained as different systems but with the same properties and therefore ease the maintenance between the systems as the same components can be used. The choice of the cloud DBMS will be further explored in chapter 6.

Local storage on the device

For local storage, the most used DBMS systems in Ionic are IndexedDB, local storage, pouchDB, and SQLite in some form[71]. Local storage and IndexedDB are key-value type DBMS systems, while SQLite and pouchDB are based on SQL and relational databases.

Local storage is a key-value pair-based storage system. It is a web browser-based database where the data does not expire when the session ends. All the data stored in local storage is stored as a string, where if the data is an integer, the data will be converted to a string if used[72]. This database is primarily used for browser applications that do not require storing loads of data at a time.

PouchDB is an in-browser database that enables the storage of data locally. As the name may suggest, it was inspired by CouchDB as a database designed to run well within a browser. A property of the DBMS is that it synchronizes

between the clients. While the primary function is to be used for browsers, it still supports native applications. Under the hood, PouchDB uses IndexedDB or falls back to WebSQL where IndexedDB is not supported[73]. PouchDB does, however, not synchronize with other DBMS systems.

IndexedDB is client-side storage for storing significant amounts of structured data. It is a key-value store, and the API enables high-performance data searches. IndexedDB is a transactional DBMS, like SQL-based RDBMS. However, unlike SQL, which uses fixed column tables, indexedDB is a java-script-based object which uses keys to retrieve the object from the given table. While IndexedDB does work on native devices, its main purpose is as a browser-based database used to store client data on the browser[74].

Capacitor Storage is a key-value persistent store for lightweight data. While lightweight, it is not meant to store a lot of data, data that require complex querying, or systems with a high read/write load[75].

SQLite is a library that implements a serverless SQL database engine. It utilizes structured query language (SQL) to access its data written and read from disk. As it writes directly to disk, with no installation overhead, it is a popular choice for use in mobile application development. As all the data are stored on the disk, there are no additional installation requirements while still maintaining high performance[64].

When looking at the different popular database options to use in Ionic, most are browser-based database systems. When utilized on a mobile device, these are converted from a browser-based to a native-based database. By design, SQLite is a database meant to be used as a native database whose primary focus is to store data on a disk made lightweight and take less space on the device it is used on. A drawback with browser-based database systems is that the OS on both Android and iOS prioritizes clearing the data from such places in low-storage situations[76].

SQLite is such a popular option for storage in Ionic applications. The Ionic team has also made a version where the data are stored encrypted on the disk for security purposes as SQL can be used more reliably, secure, and take less space on the device.

SQLite is designed to be used in native applications. The storage being written to the disk directly and the fact that a browser-based database can lose the data stored in the mobile device lacks space is why SQLite is utilized in IMReady and Gearggus. Another reason is the security properties added with SQLite, as Ionic has developed an expansion to SQLite, so it encrypts the data stored in the SQLite database.

4.2 Summary

In this chapter, the overall architectural design choices are explained. The design choices listed are valid for a version aimed toward firefighter pilots and general aviation pilots. More specific design choices, as well as the system design, will be discussed in chapters 5 and 6.

The application's front end is based on a mobile application that collects data through a questionnaire. The collected data is then analyzed to recommend whether the pilot is fit to fly or not. The recommendations follow a traffic light system, where green indicates no additional risk is associated with the flight, and yellow indicates some extra caution should be expressed. In contrast, red indicates a considerable risk with the flight, and it should be considered if the flight can be postponed or carried out by someone else.

In order to access the mobile application, there is a login system, which is based on the OIDC framework. There is a difference between military and general aviation in the required login systems. General aviation can use a trusted third-party authentication service such as Google, Facebook, or Microsoft as their authentication service. At the same time, the military either has to use an existing login system or develop a new login system that the military has complete control over.

IMReady and Gearggus utilized a local storage DBMS on the mobile device to store the data as a primary storage node. The DBMS system utilized as local storage is SQLite. SQLite was chosen on the mobile device for its performance, security, and the fact that for the other options considered, the OSes on the mobile device might overwrite the data if there is a lack of storage space.

Cloud storage is used on both systems to increase the reliability of the systems. The requirements for cloud storage are different for Gearggus and IMReady; as for the general aviation operators, a CSP can host the database as long as the data is encrypted and securely stored. The military, however, will control the storage of their data to ensure no mission details ever will leak. Therefore, an open-source DBMS that can be hosted on a CSP and the military's servers has been chosen in PostgreSQL. The reasoning behind the choice for PostgreSQL will be further expanded on in chapter 6.

5

IMReady

In this chapter, the system design and implementation of IMReady are described, which is the system described in "Determine Readiness to Fly: Pilot Risk Management"[5]. The system is based on a questionnaire to be filled in on a mobile application, which will be analyzed to recommend to the user whether they are ready to fly or not. The questions used in the survey were formed with one initial survey among pilots/cadets in the French airforce before the result was evaluated, and the questions were adjusted accordingly. The paper's author formed the first version of the questions in coordination with his supervisors for the thesis. The paper's author adjusted the final questions based on the feedback given by a survey conducted among the French firefighters.

The outline of this chapter will first summarize the system designs as described in "Determine Readiness to Fly: Pilot Risk Management" [5]. The architecture requirements and implementation details will then be explored for IMReady based on the observations from other systems explored in chapter 2.

5.1 Summary of the design

The system design is based on a questionnaire, with questions grouped by when to be answered. Some are to be answered every day, while others are before and after a flight. The system also has a privacy requirement from the pilots. From the findings from a survey, we can observe that not all the pilots

will answer honestly on the questions if their supervisors have access to their answers. Therefore, the answers should be anonymized before being stored any external places from where the mobile application is located.

The system should also show the previous answers given to the system, visualized by a histogram, and show how ready the pilot is for operating an aircraft based on the answers given the previous days. If any warnings are given, such as very little sleep, these should be viewed and given a countermeasure, such as taking a power nap to become ready to fly.

5.1.1 Summary of questions

In the questionnaire in the system, different question topics are aimed to learn different things about the pilot in order to be able to assist with the self-assessment of the pilot. The three different categories of questions to be asked are:

Questions to be answered everyday This group contains questions about general health each day, whether the user has been ill or anything in a similar category. It also covers how many hours of sleep and the sleep quality to check the fatigue level. The fatigue is further evaluated by asking for physical and mental activities to determine whether there might be physical fatigue or if the pilot rests enough. A series of questions about stress aims to evaluate how stressed the pilot might be and if the stress that is being perceived is of a positive or negative nature. At the end of the daily questions, there are questions regarding the eating patterns and how healthy the food is. This is to remove any concerns of fatigue that stems from unhealthy nutrition.

These questions are asked every day to track the user's general health over time. These questions cover topics where monitoring overtime might be helpful to understand how ready the user is to operate an aircraft in terms of the general health of the pilot. A self-monitored tracking of the different topics covered in this section of questions has been done successfully with mobile devices in studies performed earlier [77]. Self-monitored tracking can also lead to a change in the patterns of the pilot in order to live a more healthy life [53].

Before Flight questions is a group of questions to check for the preparations done for the flight, as well as the health status on the given day of the flight. The health status questions such as illness, medications, and alcohol are covered in this category. Illness is regarded in terms of flue or any similar illnesses. These are illnesses that may let the pilot focus on the inconvenience rather than the operation of the aircraft. If there is a severe illness, a flight medical doctor should be consulted. Medications are looked at to ensure there are

no medicaments that can lessen the pilot's focus. There are questions about alcohol as a reminder to ensure the pilot is not influenced by any alcoholic substance when starting the preparations for the flight.

The preparation questions are aimed at the armed forces, where the system is designed to be utilized. Therefore, the preparation questions are aimed toward the mission the firefighter pilot is about to encounter. These questions aim to check how ready the pilot is for the mission they are about to perform. A pilot in a firefighter prepares every detail about every flight, down to visualizing all the different actions to be taken during the flight. It is, therefore, crucial to have a good preparation for the flight before starting it. The general health questions about alcohol, medications, and illness are topics covered in the "IM SAFE" checklist [26], and are covered in more detail in chapter 2.1.1 why it is so important to have questions about these topics.

After flight questions is a group of questions to evaluate how the flight went and if the pilot has plans for any social, training, or relaxation activities. The evaluation of the flight covers areas such as how the mission went overall and whether it was a success or not. It also covers how the flight went and how the pilot feels after the flight. Therefore, the evaluation questions aim to learn from the last flights and let the system learn how the pilot reacts to a flight to further improve the system's determine readiness for flight capabilities.

Another aspect of the flight evaluation is to start a mental process for the pilot, to self-evaluate the flight to improve for future missions. The system can learn how the pilot perceived the flight with the current data given to the system and, by doing this, learn better how to optimize the assessment given to each pilot.

5.2 Architecture

The architecture of IMReady is based around a mobile application as the front-end. The front-end application is where the information about the user is gathered, and a result from the computing is shown to the user for feedback. A mobile application was chosen, as most people have access to and carry with them a mobile device. Another property of mobile devices is storing data securely for fast access. As the data is being stored on the mobile device, it enables a higher availability of IMReady, than what would be the case with a web application due to the possibility of losing Internet connection.

The answers given in the questionnaire are stored locally on the mobile device in an SQLite database, as discussed in chapter 4. When the data is being stored,

they are also sent to a cloud database to ensure there is a replica of the data. No data is being used within IMReady that can identify any users. There is a need to separate the data given by different users on the cloud database. The users are uniquely distinguished by utilizing an artificially generated identifier tied to their user account. In the prototype version implemented, this is generated by an authentication service. The military would enable the identifier in an existing login system in a production setting. Alternatively, an added solution would be to generate an identifier with user registration in a new login system.

Access control is handled in the prototype version using OIDC, with Firebase as a provider handling the different third-party authentication providers, which will be discussed in chapter 6. The login system would need to be adjusted toward integrating IMReady with an already existing login system within the military in a production setting. Alternatively, there would be a need to develop a new login system, to fit the need for IMReady and security requirements set by the military.

Storage systems

IMReady utilizes local storage and cloud storage for the storage of data. The local storage is an SQLite database that does not require any installation and is, therefore, space-effective on the mobile device. It also has good performance, where the data is read from disk. The local storage is considered the primary storage node, where the system accesses the data presented to the user in the history tab. This data is also used to determine the readiness to fly in the current version but will be subject to change for the cloud storage; this is covered more in chapter 8.2.

The cloud storage used is a PostgreSQL database, which is discussed further in chapter 6 around why PostgreSQL was selected. The data is transmitted to the cloud database upon entry to the local database. The primary responsibility of the cloud storage is to serve as a replica of the primary storage to ensure no data is lost if the mobile device is lost or damaged. Utilizing a replica database gives the system higher fault tolerance, as the data is immediately stored. The consequence of sending the data directly to the cloud storage instead of sending a bigger batch of data increases the bandwidth used by the system. However, less data is lost if any damage to the mobile device should occur.

The military wants to host their instance of a PostgreSQL database. This is to control all access to any ongoing military operation, and IMReady collects data from the preparation and debriefing of the missions. Even though it would not be possible to identify the users, there could be a way to see some operations

if the encryption were decrypted.

Access control

In IMReady, there is an access control based on the OIDC framework. IMReady utilizes the Firebase authentication service to handle the login requests from the users and stores the application identity and application secret from the authentication providers.

If IMReady changes from a prototype implementation to a system used for production, the login system has to change. The current login system is based on the OIDC protocol, while a new one would have to be connected to an existing military system or a newly developed one. However, both options would require the login system to generate some identifier that could not easily be tied with the user.

If it requires a new login system to be developed, it needs to store any user detail on a separate server from the cloud storage for the data collected about the users. The login system would also be required to have an artificially generated identifier, to distinguish the users apart from each other uniquely.

5.3 Implementation

The front-end was implemented using Angular, with Ionic providing the native capabilities of the front-end. *Angular* is a fast and easy front-end framework that handles page switching effectively. *Ionic* enables a single codebase to be used in multiple different environments such as web applications, Android and iOS. Ionic focuses on creating robust code with high performance on the different platforms it supports. Angular is one of the frameworks that Ionic supports to convert to native code, with other examples being React and Vue.

The front-end applications have a few dependencies with Firebase for access control, chartjs for creating the different histogram charts to present the data collected, and SQLite for the database. Chartjs is one of the most popular Javascript-based libraries for creating different charts.

The components selected for data storage support encrypting the data during transmission and storage. As the data is encrypted, there is a limited chance that any military graded information is spooked from any other applications on the mobile device.

5.4 Evaluation

The system is based around a questionnaire, where there is a few different groups of questions the pilots are to answer depending on the category the questions are placed within. The different groups are questions to be answered every day before and after the flight. By dividing the questions into groups like this, it is easier to find the questions to be answered at the time. Once inside a section, the user can pick a question to answer instead of having to answer the complete survey at once. This can either be divided into all the questions concerning the topic or just a single question at a time; there are benefits to both, with the former letting the user know all the different questions to be answered at once, while the other lets the user pick a question while relevant.

For some of the category names of questions, there is room for improvements to be made for the names. For example, while alimentation is the correct term to use, another word like nutrition might be easier to understand for non-native English speakers. Another category is mental activities. While the wording does describe the topic, it might let the users feel like they are at a psychologist every day. During the evaluation of MoodPrism [53], they found that their patients were more likely to use a mhApp than to seek help from a psychologist. By utilizing this knowledge, the user engagement could be increased by avoiding any question or topic that may connect with a psychologist like "Mental Activities" could do.

A question that the pilot is being asked every day is when they are flying that particular day. The question could be rephrased to if the pilot is flying that particular day and then prompted when the flight takes place afterward. Another option could be to add a new section for questions to be asked on workdays could be added. Selecting either of these solutions would reflect that pilots have days off work.

IMReady has a questionnaire covering areas surrounding fatigue, stress, and nutrition. Answering questions about these topics can help the pilot recall the different activities done throughout the day. The answers can then be reviewed on a history page. The history page presents the answers given the past week regarding the time spent doing activities related to fatigue, stress, and nutrition and how these were perceived. These answers also determine how ready the pilot is to operate an aircraft.

In the prototype of IMReady, only the required functionalities for the mobile application to work as intended are included. The part included in the prototype is a login screen in which the user logs in through an authentication provider, as displayed in figure 5.1. The first providers being enabled were Microsoft,

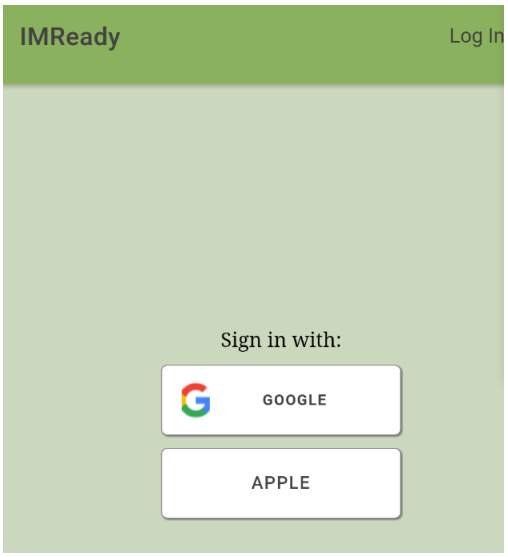


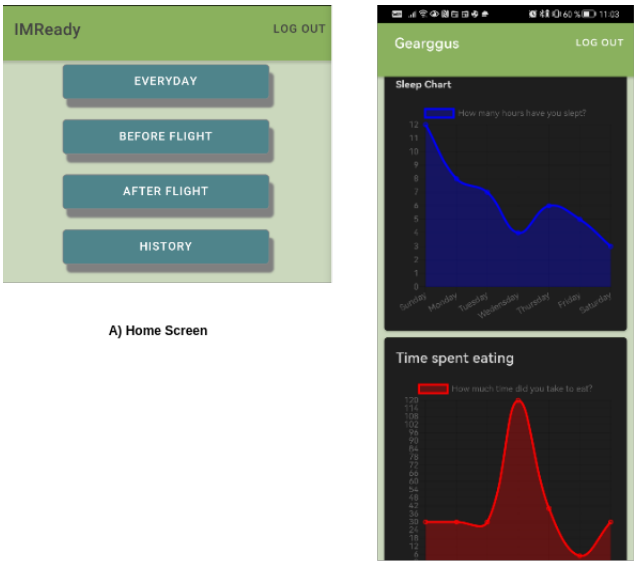
Figure 5.1: The user has to login to the system using an authentication provider.

Google, and Apple. Apple must be enabled for the mobile application to be accepted through the app store, while the others are trusted providers.

When the user has logged in to the system, a home screen is displayed, as shown in figure 5.2 A). The user can select either of the three questionnaire options of general questions, before flight, and after flight. The user can also select to show the history of data entered, displaying answers given the past week as shown in figure 5.2 B).

A series of topics is then displayed, depending on which questionnaire section was selected. An example of a questionnaire section is the everyday questions as shown in figure 5.3 A). It ranges from one to several questions underneath each topic. An example is displayed in figure 5.3 A). The different questions have different entry options, such as multiple-choice, a numeric value, or a date.

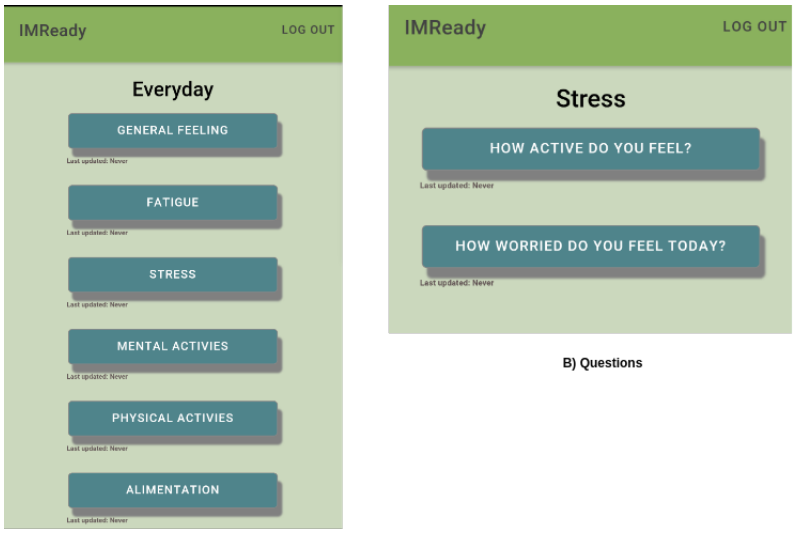
IMReady was based on the thesis written by 2nd lt. in the French airforce Thibault Kheng for a self-assessment system for pilots in a military setting; there are a few differences in the architecture for this version compared to Gearggus, which is aimed toward the general aviation. In IMReady, the data has to be stored in a database the military controls to preserve any secrets that might be hidden in the data. This requirement is lifted in general aviation, and the data can be stored in a cloud database; this will be explored further in chapter 6.



A) Home Screen

B) Histogram

Figure 5.2: On figure A) the home screen of the application is displayed, which contains the different questionnaire section as well as the history for last week, figure B) shows a graph that displays some of the history of the user for the week.



A) Questionnaire

B) Questions

Figure 5.3: Figure A) shows the different topics within the everyday section of the questionnaire, while picture B shows how the different questions are displayed within the category.

5.5 Summary

In IMReady, it was used a naive implementation of the system design outlined in "Determine Readiness to Fly: Pilot Risk Management"[5]. It was described as a version designed for firefighter pilots, and some questions were formed for the system. The main questions are based on the same topics described in the existing checklists that are being used, such as "IM SAFE" and the "PAVE" checklist. The different questions are placed in sections and topics to reduce the load of information gained at once and only show the relevant questions for what the pilot wants to give information about.

When IMReady got implemented, the system design was based around a front-end application, where the user could fill out a questionnaire and show some details from the results given. The results are stored in a database located locally on the mobile device and cloud storage for redundancy reasons. The local database is an SQLite database, which is the primary storage node of the system, while a cloud database is added for storing replicas of the data.

In the prototype, the access control to the system was based on the OIDC framework. Using OIDC, the system can leave any user details out of the system and only operate with an artificially generated identity token. If this iteration were taken further, instead of focusing on Gearggus, a new login system would be implemented or connected to another login system to let the military control the access.

While the system's overall structure with how the questionnaire and histogram interact gives the end-user help with focusing on how much rest there is and overall health. There are still improvements that can be made for the placement of different questions and the naming of the different questions.

6

Gearggus

IMReady was implemented based on the requirements defined in "Determine Readiness to Fly: Pilot Risk Management"[5]. IMReady is designed for military use and has different requirements than a general aviation system. In general aviation, any flights performed are open to the public, unlike the military. In the event of an accident, it is also society's responsibility, represented by a crash investigation unit (CIU), to determine why the accident did occur and how to prevent it in the future. In contrast, the military does its close investigations in the event of an accident.

In this chapter, an iteration from IMReady, with adjusted requirements aimed toward general aviation, will be explored. The new system will be named Gearggus, which is "I am ready" on Sami, the language of the natives in Norway. The name builds on the "IM SAFE" checklist, where the system aims to achieve the same goals but with a mobile application rather than a checklist. This chapter will outline the new requirements and the design changes made to form Gearggus.

6.1 Requirements

With the introduction of Gearggus, some requirements are to be adjusted from IMReady to suit a version aimed toward the general aviation better. In a regular operation, the pilot is still the only party with access to the data entered to

Gearggus. However, if an accident were to occur, a CIU should be able to access the data entered. The access must be managed in a way to ensure the CIU can access the data, but only with valid reasoning. A legal governor should control the access based on the reasoning given by the CIU.

IMReady relies on a login system managed by the military to handle the access control to the system. As Gearggus is not part of the military, Gearggus can use existing third-party login systems if the service provider can guarantee that the user details are kept separate from the data collected.

IMReady is designed around the reality that the military would host the database to control any data concerning any military operation, as discussed in chapter 5. In Gearggus, there should not be different databases for each airline, but rather singular for each country Gearggus is located. The database can be hosted by a third-party instance as long as the data is stored and transmitted securely with encryption. The data storage also has to follow the laws and regulations in the country of residency of the user.

6.2 Design

In this section, we will go over the different changes in the design in Gearggus compared to IMReady. Some of the changes introduced on the system side are adding a CSP to host the database, changes to the system's access control, and an access point for a crash investigation unit. In addition, to introduce the different changes, the choices regarding why these were made compared to other design methods will be discussed.

6.2.1 Cloud Service Provider

In Gearggus, as the requirements defined, the CSP has to be located in the country of residency for the user. Therefore, it is an important part when selecting a CSP for hosting the database where the location for the data center is located. In Norway, the biggest airline is Scandinavian Airlines (SAS). SAS is based in all three Scandinavian countries. Therefore, the CSP selected should ideally have a data center located in all three countries. Some other requirements for the system are good availability and fault tolerance. In Gearggus, the only external factor that can impact the availability is the CSP, as the rest of the system is located on the users' mobile device. Therefore, the CSP should be able to guarantee high availability.

A shortlist of different CSP providers was made when considering the different

Cloud Provider		Location	
Amazon Web Services (AWS)[78]	Sweden	United Kingdom	Germany
Microsoft Azure[79]	Norway	Sweden	Denmark*
Google Cloud Storage[80]	Finland	United Kingdom	Germany
IBM Cloud[81]	Sweden	Germany	United Kingdom
Oracle Cloud[82]	Sweden	United Kingdom	Germany
Alibaba Cloud[83]	United Kingdom	Germany	
Salesforce Cloud[84]	Germany	United Kingdom	France
SAP Cloud[85]	Sweden	Norway	Netherlands
Rackspace Cloud[86]	United Kingdom	Germany	Netherlands
VMWare Cloud[87]	Ireland	United Kingdom	Germany

Table 6.1: Displays a table of the 3 closest data center locations to the Scandinavian countries. The information was gathered 4th of February. * indicates that this data center is announced by the provider to come soon.

CSP options, all relatively big providers. As many different providers can be selected, it has to be limited to a select few due to time restraint, which providers are considered. Significant CSPs are only considered because they are more likely to have the funds to provide greater availability and have data centers located in more countries than smaller ones. With data centers located in more countries, Gearggus can focus on fewer CSPs.

When evaluating the different locations where the different CSPs are located, their homepages were looked at to find the data centers closest to the Scandinavian countries. These were noted down in table 6.1. In table 6.1 we can observe that Microsoft Azure is the only provider that either has or has planned a new data center to come soon in all of the Scandinavian countries.

When considering how the system can scale with the use of Azure as a CSP, Microsoft has data centers located in more than 35 countries[79] worldwide, which can serve most of the world. This ensures that if the country requires the personal health data is stored within the borders of the users' residency, Azure is more likely to provide this than the other providers as they have locations in fewer countries.

Azure also has multiple locations for data centers in Norway and Sweden, which ensures there are replicas of the data stored in a different data center but still within the same borders. This helps to increase the system's fault tolerance while still ensuring all the data is stored in the same country, as the data can be stored in both Oslo and Stavanger at the same time, which is located in different regions of Norway.

6.2.2 How the database is stored on the cloud

Azure supports two different ways of hosting databases. A database can either be hosted on a virtual machine (VM) or through a pre-installed instance of the database[67]. There are pros and cons regarding the database setup and how to access the different parts of the database with both options.

When using a **VM** to host a database on Azure, it offers the system designer to have complete control over the database, including all security measures, version of the database as well as how to connect to the database. The database will also be able to connect to a backend part of the system without transferring data through an Internet connection. As a result of not having to transmit data between two different servers, the system will be able to read the data more efficiently and lead to a faster result. All the architecture around the database has to be implemented from scratch when using a VM. The developer has to provide all the security of both storage and transmission of the data. The API for connecting to the database has to be provided as well. The developer also has to maintain any updates that come to the DBMS (database management system), and as such, there is increased maintenance required by the use of a VM.

On the other hand, while using a **pre-set database installation** done by Azure, there is an already installed SSL encryption of the transmission of the data. The connection API to the database is already set up, and Azure handles all the updates to the database. Azure offers different scaling options, where if there is a change in demand of database size, this can be adjusted accordingly within seconds [67].

The trade-off with using the pre-set database installation is that if there are additional backend requirements for the system, this has to be set up in either an additional server or VM installation, which comes with additional costs. With additional server or VM installation, there is also a delay in accessing the data if there is any computation required with the data.

Both options are covered with pros and cons. The system utilizes the pre-installed database on the Azure systems. As Azure handles all the different security requirements and makes sure the latest version of the database is installed to add security elements to the DBMS. The trade-offs involved regarding the latency are assumed to be reduced by hosting the server on the same instance as the DBMS are hosted.

6.2.3 Selection of cloud database

Azure has a few different pre-installed databases to select from. As we discussed that a pre-installed database would be the best option as Microsoft will update the database when needed and ensure all the latest security features are installed when dealing with the database, the main priority will be on the existing databases. However, if there for any legal reason has to be used a DBMS that is not available on Microsoft's servers, we will try to avoid the Azure only DBMSs. However, if there is a significant advantage of using those DBMSs, they should still be considered going forward and are therefore included in the consideration for DBMS.

MySQL is a relational database, which MySQL AS initially developed in Sweden before later being bought by Oracle and is now supported by Oracle. It is used by many big tech companies like LinkedIn, Facebook, and Wikipedia. As the name suggests, it utilizes Structured Query Language (SQL) to query the data stored and store the data. It is a very popular database; due to that, it can run on multiple platforms like Linux, Windows, and Mac iOS while being efficient on all of them. As the database is open source, many stakeholders can help correct the error and find bugs and other security breaches if an issue arises. The database supports a client-server model, where a client can store all its data on the server and communicate back and forth for storing and retrieving the data. It is also a secure database through the use of a user account management system, as well as access control, where it supports the use of authentication for contacting the database[88].

PostgreSQL was initially developed at Berkeley, University of California, and is now supported by a global development community to evolve the database, which makes it both open source and free to use for everyone. When it was first deployed, it was a relational database; based on the SQL protocol, it later evolved to support relational and non-relational (JSON) data types. The use areas of PostgreSQL include web applications, mobile applications, and analytical applications. While the original version was created for UNIX-based platforms, it has later evolved to include Windows, macOS, and Solary. PostgreSQL is used by big tech companies like Apple and Instagram[89].

MongoDB is a non-relational database where the data is stored in a binary JSON (BSON) format. This is done to support more data types than what would be able to in a standard JSON format. MongoDB is built for a distributed setting, as the creators faced scalability issues in the relational databases. As a NoSQL DBMS, there is no need for any schemas; instead, it supports any data type; this also makes the DB easier to scale than a relational database[90]. As it has to scale appropriately, automatic sharding is provided in the DBMS for distributing scaling horizontal as data volumes, and throughput requirements

increase.

There are, however, some downsides to MongoDB as well. The DB uses a single master node to control the data flow. If the master fails, it can take a couple of minutes to resolve back to regular operation. The single master node can also be a bottleneck, as it limits how fast data can be written to the DB, depending on the capabilities of the master node. On the security part, user authentication is not enabled by default but must be enabled by the user. This can lead to attacks on the DB if it is not enabled by the user[90].

Microsoft SQL server is a relational database, which was initially developed for Windows, but later extended to Linux as well. Microsoft developed it, and it builds on top of SQL, as the name suggests. It is structured because a database engine controls two smaller blocks, with a storage engine and a query engine. The storage engine controls how the data is stored and retrieved from the disk, while the query engine determines the best way to execute a query. While there is a free version, it has limitations that are either only for development or for small databases. For access to all features, it comes with a charge[91].

CosmosDB is a fully managed NoSQL DBMS. Being fully managed means that the developer does not need to spend weeks setting up the database as needed but rather install the DB and the backend developed by Microsoft to set it up correctly. As the DB is a NoSQL database, the data can be stored in different forms such as document, graph, key-value, or a column-based. CosmosDB is only available as a Platform as a Service (PaaS), which means a third party cannot host it. It does, however, come with an instant horizontal scaling. Horizontal scaling means that the database can handle the increased load by adding more servers to the cluster if needed. When data are added to the DB, the data is automatically indexed, which lets the developer focus less on setting up the DB. Microsoft offers a 99.999% availability on the DB. While the DB does come with a guaranteed speed at any scale, there is a limiting factor with that the DB comes with different consistency models depending on what the user needs. If the user needs strong consistency, the performance of the database will take a hit from it, while eventual consistency comes with a high performance[92].

MariaDB was created by the original developers of MySQL after a concern when Oracle acquired MySQL that it would no longer be open source. Therefore, there are many similarities between the two databases, which are further empathized with a monthly merge between the codebases to ensure the latest bug fixes for MySQL are available in MariaDB. As MariaDB is an answer to secure the MySQL codebase stays open source, it is guaranteed to stay open source by the developers. Furthermore, with the monthly updates with MySQL, the databases are highly compatible with each other and use the same client-

server model. MariaDB does have some features, extensions, and bug fixes that are not available in MySQL, such as advanced clustering, a compatibility feature with Oracle DB, and temporal data tables, which allows the user to query the data as it were at any point in the past. MariaDB was designed for speed, reliability, and ease of use. It also has the same security features which are available in MySQL[93].

The first requirement to keep in mind when selecting a database is that the server should be located in the country of residency of the user has. For ease of data transfer, the database should be a relational database to store the data in the same format as it is stored locally on the mobile device. This can best be achieved by using an open-source database; if the system scales in the future, the database can be installed on a different CSP server. By utilizing an open-source database, the component which sends the data to the cloud database can be the same as the one used for IMReady, as the systems can utilize the same DBMS. Utilizing the same components would ease the maintenance between the two systems, as the same update can be used on both.

With these requirements in mind, there are three different choices of database: MySQL, MariaDB, and PostgreSQL. They are all secure at storing the data, with access control systems enabled by default to ensure any third party does not access the data. At the same time, all three databases are designed with the performance of databases in mind. In a performance analysis done [94] PostgreSQL has an overall better performance than MySQL has. PostgreSQL is also built for analytical work, which is a feature of the system in terms that the data should be analyzed to find whether the pilot is fit to fly.

6.2.4 Access Control

The login system is handled by a Firebase component, which Google supports. Firebase makes it easier to log in to the system, as they provide a library of functions for login and access to user data. The only part of the user data stored on a server is the id token, which the authentication providers send. It is an artificially generated identification token to hide the actual identity of the users.

As the CSP is Microsoft Azure, Microsoft is not included as an authentication provider to avoid anyone having access to personal and user data. With competing providers having access to user data and personal data, it is less likely anyone has access to both the user and personal data. It is assumed competing providers will hide the stored data from each other for commercial purposes. When looking at different providers for secure login, the services that the users generally use should be available to make the application more acces-

sible for all users rather than only users who use the selected authentication providers.

Use of server for secure login

In order to use OIDC, there is a need for a server where the client sends a request and access the information for the different authentication providers' needs like secure ID and shared secret to be stored. While these can be located at the client, this comes with a security risk as the application secret is used to generate access tokens. For this reason, authentication providers like Facebook have as a requirement that the application details (ID and secret) are not located in a client-side code or decompilable code[95].

To secure the application secret, a server can be implemented to handle requests, hiding the application secret from the client code. This solution also raises the need to create a login system for the application. Another option here is to use existing servers to handle the login to the application, which securely stores the app secret. Experts often implement these systems within computer security, and complex login systems make them more secure than a login system created specifically for this system. Some of the options that were considered as server hosts were these:

Firebase authentication is built on the open standards, implementing OIDC and OAuth 2 protocols. While implementing these protocols, their login components are created to fit the standard based on the years of experience Google has in the area. Within the Firebase authentication team, some members were part of designing the OIDC and OAuth2 standards and helped with ID tokens and native applications anti-spoofing mechanisms to secure the use on mobile devices.

As Firebase has a storage part, they are well suited for storing application secrets in the login system. However, they also support the application to use an email/password or phone number login systems. Firebase has features such as resetting the password, account linking, and login hints if needed for the users. Another feature implemented in Firebase is a smart lock for Android, which lets the application remember the user login tokens and automatically log in to the user when the application is opened. This lets the user be able to use the application even while being offline[96].

Okta is a leading single sign-on (SSO) provider. They offer IaaS (Identity as a service) solutions for enterprises and make digital interactions between employees and customers safer. While they offer internal SSO systems, they also support the OIDC protocol, where users can use other services such as

Google and Facebook to login in. When using OIDC with Okta, the application has to be registered on their platform, and then all the details concerning the clientID are handled within the application[97].

Ory's Hydra Ory has multiple different components, which handle security. As Ory has divided the different components apart lets the developer choose which part is needed for the system. For example, Ory Kratos is an identity and user management component, while Oathkeeper is an identity and access proxy [98].

Ory hydra is the part of the ecosystem that implements OAuth2 and OIDC. As such, it provides OAuth2 access, Refreshes, and ID tokens. For the application secret and such, Ory has storage where these are kept[99]. While Ory offers to use their Ory Kratos to manage login systems, they do not require it and let the developer choose which components to use.

While all three login services handle login through an authentication provider and have a redirect option back to the application, the most significant difference is how the application identity and application secrets are stored. Firebase and Ory have as their baseline to store the details on their servers, while Okta has as a baseline to store the details within the application's codebase. Facebook has as a security requirement to their application that the storage of these details should be located on a server[95] and let all the login details be handled there.

While both Firebase and Okta provide all the services needed or none, Ory offers all the different components needed to form the system, but the developer can choose which parts are needed and which are not. As for the developers who have developed the different systems, the team behind Firebase is also behind Google's login system and is backed by Google. Okta has got different rewards for being the best IaaS provider, while Ory does not have any big companies backing them but are more focused on open source and can get help from the community.

When choosing which provider to utilize, Firebase offers storage on their servers to hide the application details from the application code. Being open-source and backing from Google and their different components are well documented, Firebase was the best option. The system also has a smart lock feature for Android, remembering the user. Therefore the application can be accessed without an Internet connection. These are some of the main reasons why Firebase is chosen over the other providers for handling the login through OIDC.

Access for crash investigation units

In Gearggus, there is an access point for CIUs to access the data provided by the pilots in the event of an accident. This access point is managed so that there is no shape or form of user identification and user data stored at the same place or so that the CIU can access the data whenever they please. For the CIU to access the data, an accident has to occur, and a legal governor, such as a court or a similar instance, has to approve of the access the data.

The architecture of the access point is managed where the CIU learns what the ID token to the pilots is. The ID token is sent from the mobile application, as shown in figure 6.1 a). The ID token is then stored securely at a server at the CIU as depicted in figure 6.1 e). The server stores the data sent as shown in figure 6.1 c).

The access point is managed in a way where the pilot reports their ID token to the CIU, who then stores the ID token securely on their servers. In the event an accident occurs, the CIU sends a request to the legal governor to gain access to the data of the pilot involved. If the legal governor accepts the request, the ID token is then given to the instance that controls access; the data is accessed without any other information about the pilot.

The access is limited to prevent the CIU from accessing the data of the pilots if they are curious about the pilot or for any other reason they would want to access. This ensures that the privacy of the pilots is protected at all times. The pilots' ID token is set to report to the CIU, where the ID token is located in the history tab. With the pilot having to send the ID token manually send the ID token to the CIU, it allows more control for the pilot over access to their data.

Other options considered were to send the ID token automatically when starting the mobile application for the first time or a button that sends the ID token. By automatically sending the ID token, there would be no forgetting to share the information, and the ID token would be correct at all times. It comes with a downside. If the user changes the device, the information will be sent when it is not needed.

With a button that sends the id token directly to the CIU, there is no possibility of errors in how the id token is sent. However, as Gearggus is designed to preserve privacy, the pilots' names are kept out of the system. The button sending the id token would therefore not include the pilot's name, and the CIU would not be able to identify the pilot. Implementing an endpoint at the CIU would be easy to add to the system. The endpoint server at the CIU is not implemented, and therefore any automatic sending of data is not made.

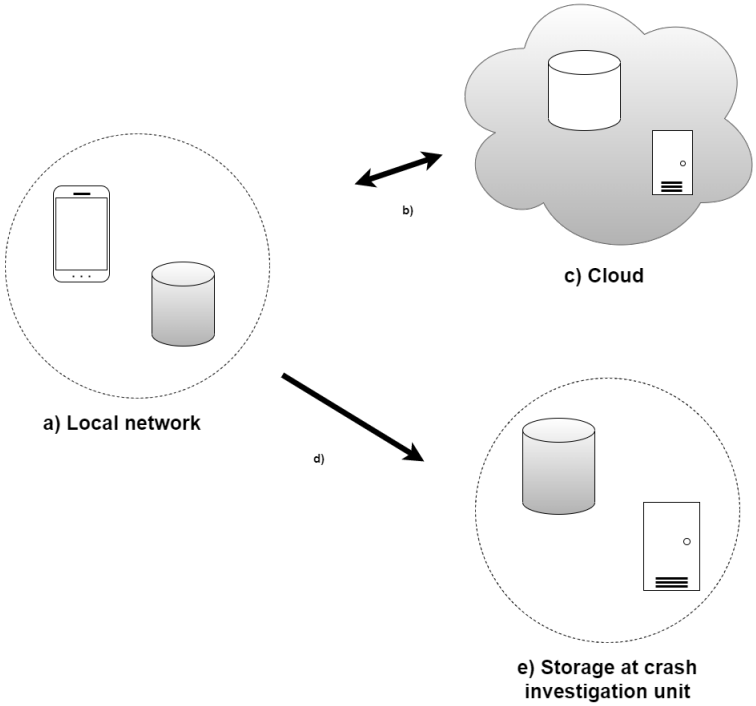


Figure 6.1: Components included in the infrastructure of storing data to find the user after an incident occurred. a) refers to the components on the local mobile device, which includes an application and storage. b) refers to the connection link between the mobile device and the cloud. c) refers to the cloud server, including a database and a VM for performing different calculations. d) is the transfer link between the mobile application and the crash investigation unit. e) is local storage for the crash investigation unit, where the different user names and their artificial identifier.

Another part of automatic sending would be to change which CIU receives the data depending on the country of residency for the pilot.

The option that is left is manually typing the ID token in a mail and sending it off to the CIU. This requires more human resources, as someone has to receive the email with more opportunities for errors. A sending of the data through a button would be the best option. However, the lack of a receiving endpoint where the country of residency can be selected, depending on which country should receive the data, is why manually sending the data is the current version.

Figure 6.2: The figure displays the different colors that indicate how ready the pilot is to fly. Green indicates ready, yellow the pilot should show extra caution, while red indicates some measures should be taken to become more ready. The colors indicate the extra dangers with a flight, as there is always something that can go wrong.

6.2.5 Determine readiness for flying

There is a check to figure out whether the pilot is ready to fly or not in the system. The check is a naive implementation that looks at the answers given in the past seven days. If only the most positive answers are given, there is given a green color with ready. If only parameters can be improved, there is a caution given. While if there are any of the worst answers given, a danger is given.

The answers being evaluated are the multiple-choice questions, where it is evaluated in which order the options are given. This is used to determine which category the answer should be placed within. Suppose the answer is located at some point between the outer edges. In that case, the pilot is asked to exercise caution while flying—the questionnaire is set up with the most positive answers given first and negative at the end.

Three different colors can indicate how ready the pilot is to operate the aircraft, as depicted in figure 6.2. Where green indicates there are no additional dangers with the flight, yellow indicates some small dangers, and red indicates measures should be taken to reduce the danger level of the flight.

In order to find the improvements to be made on how to improve the scale, the pilot can review the history tab, where all the inputs in the past seven days can be reviewed. There are also graphs of sleep duration, how long time is spent eating, and how long time is spent on mental activities in the history tab. In figure 6.3 there is an example of how the history page will look.

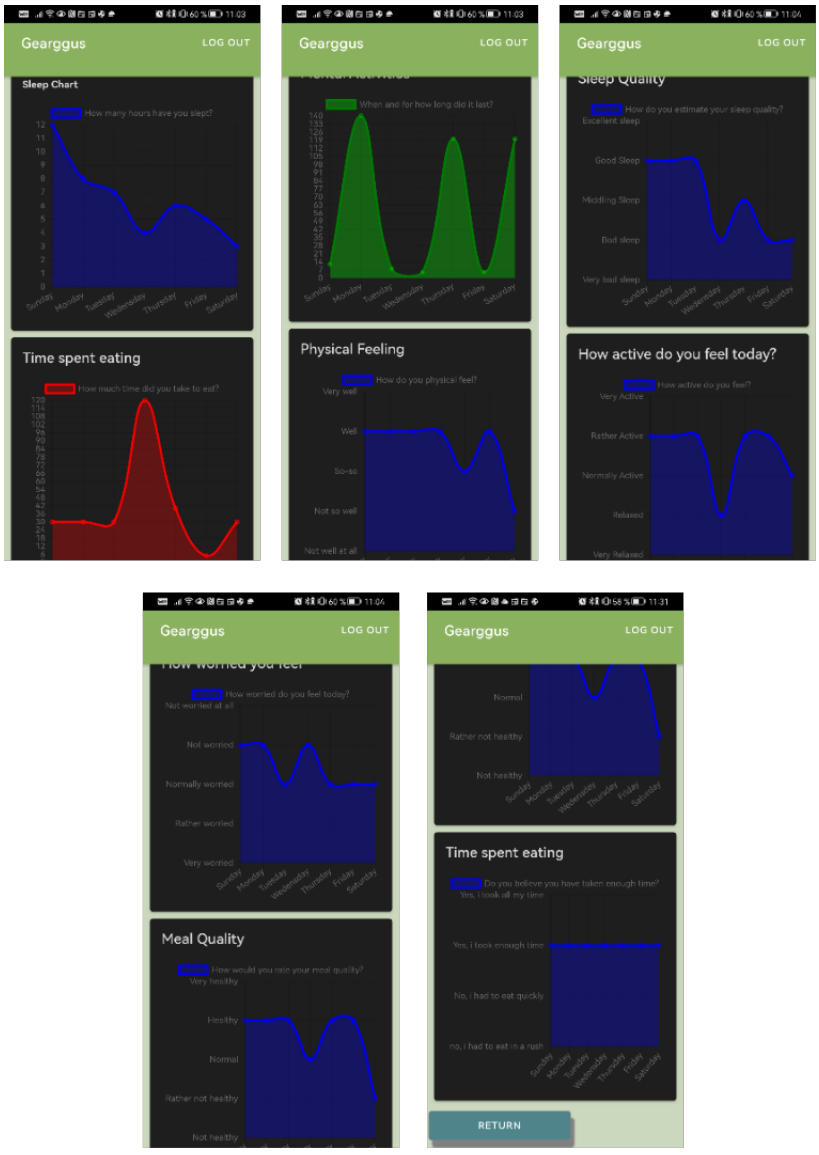


Figure 6.3: The different histograms that are present on the history page.

6.3 Implementation

The front-end of the system is written in Typescript, utilizing the frameworks of Ionic/Angular. *Ionic* is the framework that translates Angular code to native Java code. These are frameworks there was expertise in from people working around me while developing the system. As such, there it was easier to get help if stuck on any particular part of the implementation.

Other frameworks considered to use for the front-end were Flutter and React. As Flutter is a newer framework, it was expected that there would be less open-source code available to learn from than what is the case with Angular and React. In the comparison between Angular and React, it came down to that there is more knowledge about Angular than React around me, while I had no real experience with either of them when starting this thesis.

6.4 Summary

In this chapter, we have defined some changes in the requirements for the system when going from a military setting to a more general aviation-focused system. The adjusted requirements change how the login system is set up, storing the data, and adding a CIU access to the data. A CIU might have access to the data the pilot entered before an accident occurred. The CIU can then evaluate whether the pilot had anything to do with the accident.

With the changes in the requirements to the system, the updated design was discussed. The choice of a cloud provider was explained; with access to a data center in the Scandinavian countries, the data storage can happen in the country of residency of the user. Microsoft Azure was the only provider who had a plan to accomplish this, with data centers available in Norway and Sweden, noting that a data center is coming soon to Denmark. While a date for when the data center is opening is not announced, Microsoft has ambitions to upskill 200,000 to use the services within 2024[100].

The different options available on the Azure platform were all considered on the cloud database. PostgreSQL was chosen as it is relational and aimed toward analytical data processing. While there are other options on the Azure platforms that can do this, such as MySQL, Microsoft SQL, and MariaDB, this is the type of data handling that PostgreSQL is designed for and, therefore, probably performs better at. Another consideration was taken in selecting open source databases if the system is used in a country where Azure does not have a data center, while, for example, Amazon does. These systems should then be able to use the same database while they are hosted from different CSPs.

The access control system used was developed by Firebase. This enables easy use of a login system. The application ID and application secret are stored safely on the Firebase servers instead of in a decompilable code. Firebase also provides access to login with different providers and access to an id token from the providers. This enables the system to identify the users without revealing the person's ID uniquely.

When it comes down to the provider of the secure login system, different options were considered to solve the hosting of data in different ways. As the storage of the application ID and security in the application might be a risk to the application, this was avoided. An easy-to-setup solution was also preferred, where it is a fast but secure login system setup. Firebase covers both of these, with secure storage on their servers while providing the components that handle login and authentication. Firebase also has Google smart lock for Android enabled, giving an automatic login to the application.

The CIU access is enabled by the pilots giving their ID token to the CIU, who then store these securely. If the CIU wants access to the data, they have to request access to the data after an accident has occurred. This then functions as a control system to reduce the chance of accessing the data when an accident has not occurred and, therefore, not the correct reasoning behind it.

There is also added a function to analyze the data to determine whether the pilot is ready to fly or not. The function looks at the data provided the past week and then checks based on the provided data if the pilot is 100% ready or not. Feedback is given through a traffic light system, with red, yellow, and red colors provided. The green color represents no danger to the flight; yellow showcases some improvements, while red showcases that the pilot should consider not flying.

/7

Experiments

To investigate if the mobile application does help pilots with their self-assessment, we designed and implemented Gearggus as described in chapter 4, 5 and 6. This chapter will evaluate how the performance of Gearggus with user tests to test if it can be of help to the users.

Gearggus collects data through a questionnaire. The data is stored on a local database on the device and in cloud storage. A small function naively looks at the inputs given and hands out a result to determine readiness for flight.

To evaluate whether Gearggus can succeed in helping pilots to assess if they are fit for flight or not, we seek to answer the following questions:

Question 1: Is it user-friendly?

Question 2: Would it help with self-assessment decisions taken?

Question 3: If made available, would it be used?

By answering these questions, we can determine if Gearggus can be a tool that can assist with the self-assessment of how fit the pilot is to operate an aircraft. The aim behind the design of Gearggus is that people with less technical ability should still be able to use the mobile application. Therefore, it should be evaluated whether this was achieved. The evaluation of how helpful the system is would indicate how much interaction the users will have with the

system.

Question 4: Do the user feel the privacy is ensured in the system?

Another critical part of the system design is to keep the users' privacy preserved throughout the entire system. According to the GDPR rules, systems should be designed with privacy in mind[6]. As such, the privacy-preserving methods used should be evaluated as well. An important part of the system to assess here is how the CIU access control interference is perceived. As there are systems in place to limit the access of the CIU to the data, it is important to evaluate whether this is perceived as enough for the pilots.

To answer these questions about how Gearggus performs based on usability, the author tested the mobile application over three weeks to see which improvements could be made to make the mobile application more user-friendly. As the mobile application was only installed on the author's phone, running on an Android OS, there is limited testing of how the mobile application works over a more extended period.

Pilots have to be asked to determine if it would help with the self-assessment decisions taken, ideally over a more extended period. As the mobile application has only been installed on a particular mobile device, it is harder to evaluate how Gearggus has performed on this metric.

Senior aviation personel

7.1 User testing

When the author tested the system over an extended time, the lack of a reminder to open the mobile application every day to input the data were noticeable. The days when the author did not work on the project were the most prominent of this issue. By adding a push notification, which reminds the user to answer the questions, it will be easier to remember even on days when the user does not work to fill in the questionnaire.

The determine readiness function works well on the same day the information was provided to the system. However, if an earlier day in the week were bad but got picked up, later on, the function did not reflect this. For example, bad sleep quality on a Monday still got reflected in the result on the Friday. Another part that could be improved is to add the ability to change the impact from the day. The bad sleep quality on Monday could be fixed with a powernap to pick up power again.

As for the questions, some of the wording can be confusing, where it is not defined whether the times given are expected to be minutes or hours. Some of the wording of the topics and questions related can also be either confusing or create the wrong impression of what is meant. For example, "Mental Activities" can get the feeling of going to a psychologist every day. Another question is "How active do you feel?" under stress. Whether this question is related to stress or physical activities can be confusing. If it is regarded in terms of stress, then which type of active is meant with the question.

Some of the good parts of the mobile application are that it is responsive, with the information coming in quickly. When following the determine readiness function, it is relatively easy to see which improvements are to be made to be more aware during daily life. During the testing, the author slept more than otherwise and, as a result, was more focused on the tasks to be done during the day. Following where to improve from the graphs was more effortless in the last week than what was the case during the first week. The information displayed in the charts can still be improved to improve usability further.

More users would have to test the system over an extended period to gain more accurate results on how the mobile application is perceived over an extended time. Further testing is done over interviews where the user interacts with the system over a short time and is presented with the solutions on how to preserve the user's identity during testing.

An experienced captain and two experienced instructors at the Department of Aviation at UiT were interviewed to give their evaluation of the system. A qualitative interview was conducted with the captain, presenting the system to the captain. During the interview, the captain gave feedback about the system and insight into how the aviation industry operates. The instructors provided input on how the system is likely to operate in practice, with their experience in how the aviation industry operates.

7.2 Qualitative interview with a captain

When Gearggus were to be evaluated, a qualitative interview with an experienced captain for a Norwegian airline was a part of the evaluation. The interview covers multiple different topics that touch on the current situation within the aviation industry and different solutions to solve the issue. Regarding the current situation in the industry, it was touched upon both in terms of fatigue and how the working environment is, both with the big reputable[1]

1. Example of reputable airlines are SAS, Lufthansa and British Airways.

airlines and the big cynical[2] airlines.

In the interview, it was mentioned how especially fatigue is a significant concern with the pilots. The fatigue experienced is caused in terms of how the work schedule of the pilots is organized. In earlier years, the pilots had an easy-to-understand work schedule, where they could rest out between the periods when they worked. In the later years, there has become an increased pressure from their employer to work more hours. The company's current work hours is a workweek that is close to 50 hours a week. In terms of what EASA has set, the legal work hours are toward a 60-hour week, which the cynical companies are using. The airlines do have a rotation of when the pilots work; an example was six days at work and two days off. In the rotation, the pilots could have an early shift, where they start close to 5 AM, at the start of the rotation, and having a late shift, where they could be off the clock at midnight in the end. By having these work hours, the pilots will be losing a day of the two days of work, which means they effectively only have one day off in total. In other airlines, the pilot may be located in Bodø, but the first flight of the workweek starts in Oslo. The pilot then has to fly to Oslo on their day off to start their week. Suppose the week ended with a flight to Hungary, and the next flight was to depart the next day; the pilot then would lose another day in travel to get back home. These work schedules do not take the important work-life balance into account. As a result, the pilots can't see their families as often as they otherwise would.

Another part covered in their work schedule is that one workday can exist for up to 16 hours a day, including an extended time if any unforeseen event were to occur. With this, there was also said the pilots did feel a responsibility for the customers to take the flight, as the inconvenience for the customers can be a burden. Another part would be the additional cost for the airline if they did not take the flight, as the airline would pay for hotels and food for the customers in the worst-case scenario of canceling the flight.

The increased pressure from reputable airlines regarding how much work there is expected to work can be linked to more cynical airlines. As more airlines push the ticket prices lower, all the different airlines have to follow, to have customers buying tickets from them. As this has been going on for over a decade, the whole field has become a more cynical industry, as all the airlines have to follow. Consequently, airlines have started to use contract workers at a more rapid phase than workers who are employed within the airline itself. With the use of contract workers, there is believed that in any accident where a contract worker is involved, the airline would take no responsibility. The lack of responsibility is not just limited to a contracted pilot being involved in an

2. Example of cynical airlines are WizzAir, Ryanair and Easyjet.

accident, but also if an aircraft were hired in with a crew but flying under the airline's color.

There was mentioned during the interview how it is handled when a pilot report that they are too fatigued to take the flight. How this is dealt with differs depending on the airline in question. In a reputable airline, they are more open to accept that a pilot is too exhausted to take the flight and find a replacement pilot than a cynical airline. In terms of a contract pilot, there may be a fear of being fired if the pilot reports being tired a couple of times. Therefore, a contract worker is more likely to fly while being tired than an employee would be.

Suppose a pilot is too tired to fly or any other incidents were to happen for some reason. In that case, the airlines typically have a reporting system to notify the airline about any problem the pilot has or if anything happened. If the pilot were to report issues regarding work hours or warnings of fatigue, the captain did not feel the airline took the reports into account. As a consequence, there are a lot of incidents that are not reported. How the fatigue is perceived differs depending on where in Norway the pilots live as well. A pilot that lives in the northern parts of Norway may be extra tired during the polar night. The tiredness is not so much the case at work, but rather when they are off work and trying to rest. During the working hours, the pilot gets to see the sun, while during their time off, it will be dark all the time. The polar night is less of an issue in the southern part, as they have sun during the day.

To get a job in an airline as a pilot, the airline often wants the pilot to already have a type certificate on the particular aircraft they are to operate. To gain such a certificate is very expensive, putting the pilots at a financial risk. On top of this, there is often a requirement of a certain amount of airtime to get a job in an airline. The most cynical airlines are likely to take advantage of this, where they take paid from the pilot to let them gain the experience required to take such a job.

With all the different factors affecting the pilots, with a high number of work hours and various certificates required to operate an aircraft, the pilots want to be appropriately compensated. There is also the stigma in the society, which is placed by the company union, that pilots are overpaid. The pilots are often arguing with the airlines against this part. A suggestion mentioned to overcome this was having the same pay scale as the industrial workers in Norway have and, therefore, having an easier time justifying the payment. As previously stated in chapter 2.1.1, workplace issues are stressors that can lead to a negative impact on the pilots' ability to operate an aircraft. A stressor can be removed from the pilots by changing the pay scale.

These issues are more prominent in Europe than what the case is in the USA. For example, while the pilots in Europe are allowed to work 60 hours a week, the pilots in the USA are limited to 30 hours a week, according to the captain. There is also a limit on 10 hours a day in the USA, instead of the 16 hours in Europe. The reasoning behind why this is the case is that the worker unions are both fewer and more robust than in Europe. In Europe, there are 47 different countries where the power of the worker unions ranges from next to nothing to being very strong. Another point mentioned is that there are about five major airlines in the USA. At the same time, in Europe, there are believed to be over 400. The airlines in Europe are structured where some of the biggest airlines may have created smaller ones, which aim to compete against the most cynical airlines. At the same time, there are also many different airlines in each country. As there are so many different airlines and worker unions to regulate, there are harder to limit the environment within each airline. With fewer regulations on how to operate airlines, the reputable airlines are dragged more toward the cynical airlines in order to compete with them.

A result of the most honest companies drifting more toward the cynical approach to stay competitive within the aviation industry is more stress and fatigue on the pilots. The captain said that the operating environment is vastly different in Northern Europe compared to Southern Europe. For example, while in Southern Europe, most of the airports are located in flat places, without snow or ice, Northern Europe can be windy and slippery inside a mountain pass during the winter. There may be difficulties landing in such conditions when the pilot has been at work for 15 hours. The captain also said that in the Norwegian airlines, it is open for the pilot to land at a different airport like Oslo if the conditions do not feel safe enough at the airport they were supposed to land. In the major cynical companies, the leaders may tell the pilots to land anyways.

7.2.1 Suggestion to solve the issues

A few ideas were suggested during the interview to solve the issue and let the pilots rest more. The captain highlighted that while it is still safe to fly, the aviation industry does move in the wrong direction. To learn from how the aviation industry in the USA operates, stronger worker's unions were suggested. The worker's unions can ensure a better work-life balance than what the case is currently.

What was suggested was to evolve how Gearggus operates and change the direction of the CIU access. As Gearggus already collects information about topics related to fatigue and stress, an access created toward the local aviation safety agency was suggested. The access would require the local aviation safety

agency to know the pilot's airline. If there is registrated x% of the pilots within the same airline are majorly fatigued, a warning will be displayed inside the safety agency. However, with such an access, the pilot's names would still have to remain anonymous to preserve the users' privacy. Still, some sanction opportunities would then be applied, based on the fatigue levels of the pilots, to the airline to correct the working environment. With this, however, there has to be taken into account that some pilots may want to answer poorly on the questionnaire, to shed a bad light on the airline in the case they were to leave.

As previously mentioned, within Europe, there are local aviation safety agencies in most of the countries. If the proposed system design were to be done, the airlines registered in the most regulated countries would lose an advantage over airlines from other countries. For example, Ireland's working environment rules differ from Norway, Sweden, and Denmark. Part of the reason why there are registered over 400 airlines in Europe is to avoid this, as there are registered airlines in other countries to be regulated under a different set of regulations. The registered airline in the lenient country will still fly under the same name as the airline registered in a strict country, but with different rules. For example, an airline can be registered and recognized in Norway. All the customers think of that airline as a Norwegian airline. That airline can then create a sub-company registered in Ireland and fly under Irish regulations, which are more lenient than the Norwegian regulations. All the customers would then assume all the employees working in that airline work under the conditions set by the Norwegian laws, while the reality is that they work under Irish regulations. By doing this, the airlines will operate under a different set of rules while keeping their reputation.

The structure of the proposed system design was then changed to accommodate these realities. Instead of reporting to a local aviation safety agency, the system will have to report to EASA. EASA would then look into how the work environment is present inside the airline. If too many pilots report fatigue or stress that exceeds the healthy levels, EASA would be able to look into the case and sanction the airline accordingly.

However, to adjust Gearggus toward creating a more healthy and safe environment for the pilots, psychological and legal experts must be consulted to further shape how such a system works. The current proposed changes to Gearggus are presented based on how Gearggus currently works and the desired path in how Gearggus can help change the working environment to a safer environment for everyone.

7.3 Evaluation with pilot instructors

Gearggus got presented to professor Vegard Nergård and Bengt Svendsen. They are work at the Department of Aviation at UiT and are educating future pilots. During the evaluation, the inputs they had on how the system would work on a paper compared to how the practice of the system was evaluated. How such a system would work as well was discussed.

The evaluation of the system was in regards to how the idea of the system, the issues it tries to solve, and how pilots are likely to view the system. The professors acknowledged the importance of the data collected by Gearggus. They also acknowledged the analyzed data to give feedback to the pilots. However, pilots are very skeptical of any data collected about them in practicality. If anything happens during a flight, the reasoning behind why the incident occurred was appointed to the pilots within 24 hours.

A few examples were given regarding how the pilots are getting the blame and how pilots handle any data collection. An accident with a de Havilland Canada DHC-6 Twin Otter from Wideroe was taken as an example. The pilot got the blame for the accident because he had taken medication that did not interfere with his ability to fly. The reality of why the accident did occur was appointed toward turbulence. An example of pilots who do not want any data collected is a study conducted to experiment with smartwatches to see how stressed pilots are for Luftambulansen. Before the experiment started, a big argument was given for how much the experiment was even legal. During the experiment, it was observed that very few of the pilots did wear the watches, as they did not want any result to come from the study that forced them to collect data about stress.

Pilots get the blame because the airlines can not take any responsibility for the accidents as they would have to pay instead of the insurance companies. As the responsibility is put on the pilots, whether its post-mortem or not, it is in the pilots' interest to not help collect any data that can be used against them if any incident or accident occurs. To protect the pilots, the pilots are trained at UiT to ensure they keep the certificate of their co-pilot and the aircraft's logbook in the event of an accident to prove the technical ability of the personnel and aircraft. This reduces the chance of responsibility put on the pilots for the accident.

The instructors said airlines are legally bound to create work schedules that align with research done regarding sleep schedules. However, when some airlines conducted research regarding fatigue among the pilots, the data collected were likely skewed due to the pilots did not want their work schedules to change. These results keep iterating under the narrative that it is in the pilot's interest

to keep any data regarding fatigue and stress from being available.

7.4 Reliability, fault tolerance, scalability, security, and usability

In chapter 3 some of the requirements listed for the system were how to deal with reliability, fault tolerance, scalability, security, and usability. Some of these are intertwined, as the same solution can cover multiple requirements. This section will cover how the different non-functional requirements were dealt with.

Security and privacy are vital in terms of protecting the user and its health data throughout the system. The system is designed with security and privacy in mind, making sure steps are taken to ensure privacy by design. The login system is an OIDC implementation done by Firebase. The users can log in with Google or Apple (Apple is only available on iOS). Using an authentication provider, user information such as name, email, and phone number is not stored within the system but on the authentication provider's servers. By having an authentication provider storing the user data, there is less of a chance for a third party to identify a user. To further ensure this, as Microsoft is the CSP, they are not available as an authentication provider to ensure no third party has access to both the data provided and the user data.

The databases used can encrypt the data stored to secure the data used within the system. The data transfer is a weak point of securing the data, where malicious attackers can catch the transfer; the data is further protected with SSL during transfer.

Reliability and Availability have been achieved with storage locally on mobile device, as well as the smart lock feature provided by Firebase, which enables the application to be used even without an Internet connection available. The smart lock feature is only tested on Android and can be subject to change on an iOS device, as Google developed it. The system is available whenever the user wants to use it with these two parts, whether in the cockpit just before a flight or at home. The only requirement is that the mobile device is available.

Scalability is handled through Azure's ability to scale the cloud database when needed. Azure provides different options on how much space is available, which can be changed within seconds [67]. With this, the only limit for how much it can be scaled is the price for data storage and the space on the servers of Azure, which they are monetarily incentivized to increase as needed. The other

parts of the system are not affected by scalability, as it is located on the users' mobile device.

Fault tolerance is ensured with a cloud database. If a user loses or damages their mobile device, the data is still available on the server. The CSP is also monetarily incentivized to ensure they do not lose any data. As such, the data is replicated on their servers as well. When the data is added to the local database, it is sent to the cloud database if an internet connection is available on the mobile device. The data is replicated whenever possible, and as such, there is less of a chance of losing any data.

The *dependencies* of the system are Firebase as the login provider and Google and Apple as authentication providers for the login systems. As for storing the data, there is an internal database provided by Ionic, which converts the code from a browser-based interface to a native code. The cloud database is hosted by Azure and provided by PostgreSQL. When creating the different graphs on the history page, chartsjs are used. These are the different dependabilities of the system.

Maintainability is achieved by dividing the system into different components. Any communication between the components is done through services to lessen code reuse. The code is also documented through variable names, function names, and comments to assist others in understanding the code. Any dependables used in the system are only used if the component is well documented in the providers' documentation.

To make the questionnaire easier to edit in the future, all the data is stored in a JSON object. Each question is assigned a unique ID to separate from the other questions in the questionnaire. The ID number is significantly arbitrary, making it a more straightforward process to add new questions to the questionnaire, making room for many new ID numbers.

To provide *usability* for the users of the application, each question can be answered separately, rather than having to answer a whole section at once. Further, a box indicates how ready the pilot is to fly once a question is answered. Green, yellow, and red indicate the different readiness levels. To inspect where the various data occurs from, there is a history tab that provides all the various inputs of data provided over the last seven days.

Performance is ensured by letting the application only load the parts that are needed for each function to work as intended. By running everything which is not dependent on earlier parts synchronous instead of letting each part complete before running next where applicable, the process of loading each page is faster. On the history page, the data which the graphs are made from is

generated after the page is loaded and then updated as the different parts are completed. As the system sends the data directly to the cloud when answers are given in, rather than once a day, the bandwidth usage does increase. However, suppose the device is stored once a day. In that case, it has to be when the device is active, as some OSes might deny any background operations to save battery power. If the application is not being used through the weekend and the mobile device gets lost, all the data might be lost.

7.5 Discussion

During the interview conducted with the aircraft captain, a few different changes were discussed to align Gearggus toward the aviation industry better. The proposed changes discussed with the captain were regarded in terms of how to better monitor the data collected and how to report any outliners with how the aviation industry is organized. The system design that were proposed during the interview are listed in chapter 7.2.1.

However, during the evaluation with the two professors, a few concerns were made concerning the proposed changes. They highlighted a few different concerns with the different types of pilots, such as a pilot who is an airline employee, a contracted pilot, and a pilot who has to pay to have enough hours to operate the type of aircraft in question. Depending on the different circumstances the pilots fly under, different stressors impact the pilot. An incident pointed toward is an accident that occurred in China on the 21st of March 2022. The pilot was contracted to an airline in China; he got fired from the airline and, as a result, lost all his pension. The pilot decided to commit suicide with an aircraft full of passengers. This indicates how fateful the consequences can become due to the working environments.

The system requirements, design, and implementation of Gearggus have been done based on the requirements defined in "Determine Readiness to Fly: Pilot Risk Management" [5], and the experiments done in regards to systems within the same area such as PMSys [50, 20]. A prototype of a proposed system was created to understand and evaluate how such a system was to be developed for general aviation. The prototype of Gearggus showcased in which direction the system was intended to take and to evaluate whether the direction was correct or not. Based on the evaluation from an experienced captain and two experienced pilot instructors, other issues within the aviation industry are not accounted for.

An issue that is pointed toward from both the captain and the professors, is that there are issues for the pilots. Gearggus showcases that a technical solution are

possible. However, if Gearggus collects quantitative data, such as how many hours has the pilot worked for and how many days in a row the answers are less likely to be skewed from the inputs generated by the pilot to gain an answer. Another solution can be to have a game that tests the reaction time, or any other factors that are affected by fatigue.

7.6 Summary

The author tested the system for three weeks to see how it worked over an extended testing period. During the testing, it was found that a reminder to enter data if it was forgotten, especially on days where it was not worked on, would be a helpful addition. While the determine readiness function did improve the sleep pattern during the testing period, it was found the time frame it reads data from has to be shortened. The reason is that the perceived fatigue had less to do with the data a week prior than the past few nights. Another part that was noted was the lack of a overwrite on the color chosen. For example, if the color was red, and a powernap solved the lack of sleep, then another color like orange would be helpful to display.

A qualitative interview was conducted with an experienced captain in a Norwegian airline about their working conditions. The captain did talk a lot about how fatigue and stress are factors that come into play, where the pilots are being forced to work more to keep their job. There are also different airlines in Europe, where their operations differ a lot. The captain mentioned two main categories of airlines: honest/reputable airlines and the more cynical ones. The most cynical airlines are explained as those who shape how the aviation industry was formed. The reputable airlines either have to follow, or they will lose money.

As a result of the control the cynical airlines have, combined with many different work regulations within Europe, the pilots can legally work 60 hours a week or toward 16 hours a day. As the pilot can work that much, the work-life balance can take a hit, adding another stressor to the pilot.

During the interview, a suggestion was to adapt how to whom Gearggus can report data to. If Gearggus finds too many pilots within the same airline being severely fatigued or stressed, a notification can be given to EASA. EASA can more effectively control the airlines to ensure they respect the pilot's well-being.

Two employees of the Department of Aviation at the University of Tromsø were interviewed to evaluate their perception of how Gearggus would perform if it

were evolved to a production-scale system. They did acknowledge the issues Gearggus are trying to solve. However, they pointed out that, in their opinion, pilots are not likely to be willing to give any data that Gearggus collects. The consequences of collecting such data are probably too high, as there are too many examples of pilots getting all the blame for any accidents. Within 24 hours, the responsibility for the accidents is usually pointed toward the pilot. As such, pilots are more unwilling to give any data.

It has been proven that such a system is technically possible to achieve. However, there are other factors that are not present in systems in similar domains. Based on the evaluation data, a different system based on quantitative data or collecting data through games may have a higher chance of collecting the correct data than user inputs.

The different requirements listed in chapter 3 are handled by using storage systems with both a local and a cloud database, where the cloud database is replicated to at least three different instances. This provides both reliability, availability, and fault tolerance. For security and privacy's sake, an authentication system is another provider than the CSP that handles the login requests for the users. The data is also encrypted when stored to make it less likely that anyone can gain information about the data provided. The system uses different components to handle login, storage, creating charts, etc. It is dependable on other providers. There are used services to handle events that occur in different components to increase maintainability. For example, everything related to reading and writing to the databases is dealt with in a service. The different components then call the database service to store the data.

8 Conclusion

This chapter summarizes the thesis and restates our findings, describing how Gearggus operates and how this affirms our thesis. To recapitulate our problem definition for this thesis, it is that:

This thesis will explore and define the system requirements for a privacy-preserving self-assessment system for pilots and then design, implement and test such a system.

To evaluate this thesis, we explored and defined system requirements, designed, implemented, and tested the Gearggus system. Gearggus consists of a mobile application and a backend, which has a cloud database to store the data. The data is collected through a user's questionnaire in the mobile application. These answers are then stored on the mobile device. The cloud database stores the answers from the mobile application as a replication of the data to ensure no data is lost. As Azure hosts the cloud database, they will also replicate the data stored at three different locations to ensure they will not lose any of their data either.

8.1 Conclusions

In chapter 4 the architecture of both the systems is presented. This architecture will work for the military and general aviation settings with some case-specific

modifications. These differences were then explored in chapter 5 and 6. The version evaluated in chapter 7 is Gearggus, which is designed for general aviation.

In chapter 5, the design of IMReady, which is based on "Determine Readiness to Fly: Pilot Risk Management"[5] is explained. As the paper is based on findings on how such a system would work within the french air force, this is aimed at the military. While it was created as a server code, it has not been tested thoroughly to handle a database on a server. This server code does not concern itself with essential security practices, which would have to be implemented if moved in this direction. A login system for this version was never implemented, nor was how to handle storage of user information secure from any officer's eyes.

In chapter 6 a general aviation (GA) version of Gearggus is explored. Some new requirements came with the GA version; these are explained before the changes to Gearggus to fit the GA than the military better. These include using an authentication provider to handle the system's authentication, a cloud database for storage, and replicas of the data collected. A CIU addition was added to Gearggus; in the event of an accident, the CIU could explore if the pilot's mental state had anything to do with the accident.

During the evaluation of Gearggus, interviews were conducted with experienced aviation personnel and employees of the Department of Aviation at the University of Tromsø. During the interviews, Gearggus was evaluated regarding the problems it tries to solve within the aviation industry. Based on the evaluation done on Gearggus, a few new system designs are proposed. The requirements are based on the evaluation received through the evaluation of Gearggus.

Based on the work presented in this thesis, the following conclusions can be made:

1. Gearggus can help pilots with self-assessment, coming with recommendations for whether the pilot is fit for flight or not.

2. Gearggus allows pilots to easily access data from earlier in the week to remember how the health state was to judge the pilot's state better.

3. Required changes to Gearggus are outlined based on the evaluation

8.2 Future Work

Gearggus is a prototype of a system, and as such, there are many different options for future work to further iterate over the system to improve it.

Improve the determine readiness to fly function

While the determine readiness to fly function of the system can recommend to some degree, based on the data given the previous week, if the pilot is fit to fly or not, it does not learn anything while doing so. The currently implemented version is very naive in the approach, just looking directly at the selected options and then displaying a color based on these. A better solution would be to analyze all the different options selected and give feedback based on this. The questions could also be weighted differently; for example, if the pilot is stressed or lacks sleep is valued differently from nutrition. This part would have to be incorporated with an expert within the field.

Using machine learning to analyze the data to recommend whether the pilot should fly could yield an accurate answer. By using data from all the pilots, the model would be more accurate than by only the pilot in question [50]. Machine learning can be utilized on all the different pilots who have data stored in the database without compromising privacy [101]. Therefore, the recommendation could be calculated on a server rather than a mobile device to gain more accurate answers. The trade-off to this solution would be the availability of recommendations on whether the pilot is fit to fly or not would take a hit, as the Internet connection could be lost.

As the analysis can happen in the same data center, the performance of getting the data would be quick and not put additional pressure on the bandwidth outside of the data center. With this solution, the users would get a more accurate answer by the first day of using the system and if the system were not used for a couple of days.

Another way to further improve the determine readiness function is to determine which parameters are decreasing with importance as the days pass and let the most recent be of more importance than six days earlier. For example, the sleep duration six days ago may mean less than the sleep gained the previous day as the body could catch up on some of the sleep.

Push notifications

Create an option where push notifications can be turned on to remind the user to fill out the questionnaire for the day. The push notification should be possible to turn off if the user wishes to and change the time of the day for when it is notifying. It should also only push if no answers were given on a particular day to not become bothersome for the user.

Improve feedback from determine readiness function

There is no feedback on why the "traffic light" colors are displayed as they are in the current state. In order to improve on this, there should be made a readily available list where it is displayed which areas to improve on to get the green light if not present. If the light is red, there should be an explanation of why it is red and a function to overwrite the red. When the pilot is about to answer a question that would turn red, the pilot could also get a warning for this; this could, however, lead to skewed answers as the pilot could avoid putting answers in that would be red.

Dynamic selection of database

The cloud database is always located in Norway with the system's current setup. A way to select a different region of where the database is located would be needed in a production setting to store the data where the user's residency is located. If the release countries are more than the Scandinavian countries, each country has to explore the different locations of the data centers to make sure the data is stored in the correct place. This dynamic selection can be achieved by adding different environment variables and loading the correct ones depending on the data given.

Backend for CIU units to receive data

The current version of a CIU backend, where the data is being stored, is a theoretical system. In order to improve the way, the data is being sent from the mobile application to the CIUs, a backend system that receives the data must be designed and implemented. This backend should receive data encrypted from the mobile application, revealing the ID token and the pilot's name. The pilots' name is available through the OIDC framework, with the access token received from the authentication providers. After that, the data sent should be added to a database, which encrypts the data on a secure server to ensure no attack is made on the data.

Automating sending and storing the data instead of manually adding the data reduces the possibility of error. No one has to type the ID token or read an email and pass it to the system. Automating the sending and receiving of data will also reduce the time spent handling the ID token and, therefore, more likely to be done. When an ID is sent, it should look up if the ID already exists in the database to ensure there is no double entry of IDs for each pilot.

Change the questionnaire

The questions asked in Gearggus are not adjusted compared to those in IMReady due to time restraint and access to a professor within psychology who can help with the questions for aircraft pilots. These questions, especially those before and after flights, are aimed toward firefighter pilots rather than general aviation pilots.

Adapting Gearggus toward EASA

In chapter 7.2 a qualitative interview was performed, where some changes to Gearggus were suggested. The proposed modifications are to add the current airline the pilot works for within the Gearggus mobile application. It will then be determined how ready the pilot is to take a flight on a server. The results of this function will then be summarized for all the pilots working within the same airline. If there are too many pilots, based on set criteria, which can be determined as severely fatigued or stressed, EASA will receive a notification. The identity of the pilots should remain a secret, but instead, indicate these many pilots are fatigued/stressed.

Bibliography

[1] Damien Kelly and Marina Efthymiou. "An analysis of human factors in fifty controlled flight into terrain aviation accidents from 2007 to 2017." In: *Elsevier* 69 (2019). doi: https://doi.org/10.1016/j.jsr.2019.03.009.

[2] Joan Cahill et al. "The Requirements for New Tools for Use by Pilots and the Aviation Industry to Manage Risks Pertaining to Work-Related Stress (WRS) and Wellbeing, and the Ensuing Impact on Performance and Safety." In: *MDPI* 8 (2020). doi: https://doi.org/10.3390/technologies8030040.

[3] Simon Ashley Bennett. "Pilot workload and fatigue on short-haul routes: an evaluation supported by instantaneous self-assessment and ethnography." In: *Journal of Risk Research* 21.5 (2018). doi: http://dx.doi.org/10.1080/13669877.2016.1235603.

[4] European Parliament. *Directive 2003/88/EC of the European Parliament and of the Council of 4 November 2003 concerning certain aspects of the organisation of working time*. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32003L0088. 2003.

[5] Thibault Kheng. *Determine Readiness to Fly: Pilot Risk Management*. 2021.

[6] European Commission. *EU data protection rules*. https://ec.europa.eu/info/law/law-topic/data-protection/eu-data-protection-rules_en.

[7] R. Simons et al. *Age Limitations Commercial Air Transport Pilots*. https://www.easa.europa.eu/sites/default/files/dfu/EASA_REP_RESEA_2017_1.pdf.

[8] P.J. Denning et al. "Computing as a discipline." In: *Computer* 22.2 (1989). doi: https://doi.org/10.1109/2.19833, pp. 63–70.

[9] Joan Cahill, Paul Cullen, and Keith Gaynor. "Interventions to support the management of work-related stress (WRS) and wellbeing/mental health issues for commercial pilots." In: *Cogn Tech Work* 22 (2020). doi: https://doi.org/10.1007/s10111-019-00586-z.

[10] Gunnar Hartvigsen and Dag Johansen. "Stormcast — A Distributed Artificial Intelligence Application for Severe Storm Forecasting." In:

IFAC Proceedings Volumes (1988). doi: https://doi.org/10.1016/B978-0-08-036938-9.50021-2.

[11] Dag Johansen, Robbert Van Renesse, and Fred B Schneider. "Operating system support for mobile agents." In: *Proceedings 5th Workshop on Hot Topics in Operating Systems (HotOS-V)* (1995). doi: https://doi.org/10.1109/HOTOS.1995.513452.

[12] Dag Johansen. "Mobile agent applicability." In: *Personal Technologies* (1998). Doi: https://doi.org/10.1007/BF01324935.

[13] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." In: (2008). doi: https://doi.org/10.1145/1327452.1327492.

[14] Steffen Viken Valvåg, Dag Johansen, and Åge Kvalnes. "Cogset: a high performance MapReduce engine." In: (2012). https://doi.org/10.1002/cpe.2827.

[15] Håvard Johansen, André Allavena, and Robbert van Renesse. "Fireflies: scalable support for intrusion-tolerant network overlays." In: *ACM SIGOPS Operating Systems Review* (2006). doi: https://doi.org/10.1145/1218063.1217937.

[16] Håvard D. Johansen et al. "Fireflies: A Secure and Scalable Membership and Gossip Service." In: *ACM Transactions on Computer Systems* (2015).

[17] Audun Nordal et al. "Balava: Federating Private and Public Clouds." In: *IEEE World Congress on Service* (2011). https://doi.org/10.1109/SERVICES.2011.21.

[18] Robbert Van Renesse et al. "Vortex. An event-driven multiprocessor operating system supporting performance isolation." In: (2003). https://munin.uit.no/handle/10037/367.

[19] Corpore Sano. *Projects*. https://site.uit.no/corporesano/projects/.

[20] Olav A. Norgård Rongved et al. "Real-Time Detection of Events in Soccer Videos using 3D Convolutional Neural Networks." In: *2020 IEEE International Symposium on Multimedia (ISM)* (2020). https://doi.org/10.1109/ISM.2020.00030.

[21] Tor-Arne S. Nordmo et al. "Dutkat: A Multimedia System for Catching Illegal Catchers in a Privacy-Preserving Manner." In: *ICDAR '21: Proceedings of the 2021 Workshop on Intelligent Cross-Data Analysis and Retrieval* (2021). https://doi.org/10.1145/3463944.3469102.

[22] Joakim Aalstad Alslie. *Aika: A Distributed Edge System For Machine Learning Inference*. https://munin.uit.no/handle/10037/24668. 2021.

[23] Statistisk sentralbyrå. *Fakta om Internet og Mobil*. https://www.ssb.no/teknologi-og-innovasjon/faktaside/internett-og-mobil.

[24] A.Weaver et al. "Application of mobile phone technology for managing chemotherapy-associated side-effects." In: *Annals of Oncology* (2007). https://doi.org/10.1093/annonc/mdm354.

[25] Flight literacy. *Pilot self-assessment*. https://www.flightliteracy.com/pilot-self-assessment-2/.

[26] Sarina Houston. *The I'M SAFE Checklist*. https://www.thebalancecareers.com/the-i-m-safe-checklist-282948.

[27] European Aviation Safety Agency. *Easy Access Rules for Medical Requirements*. https://www.easa.europa.eu/sites/default/files/dfu/Easy_Access_Rules_for_Medical_Requirements.pdf.

[28] Dushyant Kumar Sharma. "Physiology of Stress and its Management." In: *HSOA Journal of Medicine: Study & Research* (2018). https://doi.org/10.24966/MSR-5657/100001.

[29] Birgitta Gatersleben and Isabelle Griffin. *Handbook of Environmental Psychology and Quality of Life Research*. https://doi.org/10.1007/978-3-319-31416-7_25. Springer, Cham, 2017, pp. 469–485.

[30] Sara Lindberg and Timothy J. Legg. *Psychological Stress*. https://www.healthline.com/health/psychological-stress.

[31] EASA. *Blood Alcohol Concentration Limits for General Aviation Pilots*. 2018.

[32] E.J.D Ogden and H.Moskowitz. "Effects of Alcohol and Other Drugs on Driver Performance." In: *Traffic Injury Prevention* (2004). https://doi.org/10.1080/15389580490465201.

[33] Susan Torres-Harding. *What Is Fatigue? History and Epidemiology*. 2005.

[34] Thomas C Rosenthal et al. *Fatigue: an overview*. https://www.aafp.org/afp/2008/1115/p1173.html. 2008.

[35] Alwin van Drongelen et al. "Risk factors for fatigue among airline pilots." In: *Int Arch Occup Environ Health* 90 (2017). doi: https://doi.org/10.1007/s00420-016-1170-2.

[36] Pilot Medical Solutions. *Flight Fitness | The "I'm Safe" Checklist*. https://www.leftseat.com/flight-fitness-the-im-safe-checklist/.

[37] Sarina Houston. *Preflight Prep and the PAVE Checklist*. https://www.businessaircraftcenter.com/articles/pre-flight-prep-pave-aviation-check-list-art0317.html.

[38] Joshua Hammer. *The Real Story of Germanwings Flight 9525*. https://www.gq.com/story/germanwings-flight-9525-final-moments. 2016.

[39] David L. Costill and Mark Hargreaves. "Carbohydrate Nutrition and Fatigue." In: *Sports Medicine* (1992). doi: https://doi.org/10.2165/00007256-199213020-00003.

[40] Michelle Carvalho Galvão Silva Pinto Bandeira, Anderson Ribeiro Correia, and Marcelo Ramos Martins. "General model analysis of aeronautical accidents involving human and organizational factors." In: *Elsevier* 69 (2018). doi: https://doi.org/10.1016/j.jairtraman.2018.01.007.

[41] EASA. *Part-M - Continuing airworthiness requirements*. https://www.easa.europa.eu/acceptable-means-compliance-and-guidance-material-group/part-m-continuing-airworthiness.

[42] EASA. *Part-145 - Maintenance organisation approvals*. https://www.easa.europa.eu/acceptable-means-compliance-and-guidance-material-group/part-145-maintenance-organisation-approvals.

[43] Chenyu Hunang, Allen Xie, and Flavio A.C. Mendonca. "Factorial Validity of the Flight Risk Assessment Tool in General Aviation Operations." In: *JATE* (2020). doi: https://doi.org/10.7771/2159-6670.1205.

[44] FAA Aviation Safety. *Flight Risk Assessment Tools*. https://www.faa.gov/news/safety_briefing/2016/media/SE_Topic_16-12.pdf.

[45] FAA. *Fly Safe: Prevent Loss of Control Accidents*. https://www.faa.gov/newsroom/fly-safe-prevent-loss-control-accidents-1?newsId=83392.

[46] Thuc Tuan Hoang. "pmSys." doi: http://urn.nb.no/URN:NBN:no-49204. MA thesis. Oslo, Norway: University of Oslo, 2015.

[47] Cong Nguyen Nguyen. "Implementation of a digital Player Monitoring System: pmSys." doi: https://www.duo.uio.no/bitstream/handle/10852/45121/7/main.pdf. MA thesis. Oslo, Norway: University of Oslo, 2015.

[48] Kennet Vuong. "PmSys: a monitoring system for sports athlete load, wellness & injury monitoring." doi: https://www.duo.uio.no/handle/10852/45135. MA thesis. Oslo, Norway: University of Oslo, 2015.

[49] Håvard Johansen, Cathal Gurrin, and Dag Johansen. "Towards Consent-Based Lifelogging in Sport Analytic." In: *MultiMedia Modelling* 8936 (2015). doi: https://doi.org/10.1007/978-3-319-14442-9_40.

[50] Håvard D. Johansen et al. *Scalable Infrastructure for Efficient Real-Time Sports Analytics*. doi: https://doi.org/10.1145/3395035.3425300. New York, USA, 2020.

[51] Nikki Rickard et al. "MoodPrism: Mental health support on your phone. Final Report." In: *Beyond Blue* (2016). doi: https://www.beyondblue.org.au/docs/default-source/about-beyond-blue/research-project-files/bw0422-rickard-final-report.pdf?sfvrsn=2d3b09ea_2.

[52] Nikki Rickard et al. "Development of a Mobile Phone App to Support Self-Monitoring of Emotional Well-Being: A Mental Health Digital Innovation." In: *JMIR MENTAL HEALTH* 3.4 (2016). doi: https://doi.org/10.2196/mental.6202.

[53] David Bakker and Nikki Rickard. "Engagement in mobile phone app for self-monitoring of emotional well being predicts changes in mental health: MoodPrism." In: *Elsevier* 227 (2018). doi: https://doi.org/10.1016/j.jad.2017.11.016.

[54] David Bakker et al. "Development and Pilot Evaluation of Smartphone-Delivered Cognitive Behavior Therapy Strategies for Mood- and Anxiety-Related Problems: MoodMission." In: *Cognitive and Behavioral Practice* (2018). doi: https://doi.org/10.1016/j.cbpra.2018.07.002.

[55] Stefanos Malliaros et al. “The Integrated Holistic Security and Privacy Framework Deployed in CrowdHEALTH Project.” In: *Acta Informatica Medica* 27 (2019). doi: https://dx.doi.org/10.5455%2Faim.2019.27.333-340.

[56] Jorge Navas and Marta Beltrán. “Understanding and mitigating OpenID Connect threats.” In: *Computer & Secruity* 84 (2019). doi: https://doi.org/10.1016/j.cose.2019.03.003.

[57] Gavin Henry. “Justin Richer on OAuth.” In: *IEEE Software* 37 (2020). doi: https://doi.org/10.1109/MS.2019.2949648.

[58] Mitchell Anicas. *An Introduction to OAuth 2*. https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2.

[59] Victor Sucasas et al. *An OAuth2-based Protocol with Strong User Privacy Preservation for Smart City Mobile e-Health Apps*. doi: https://doi.org/10.1109/ICC.2016.7511598. Kuala Lumpur, Malaysia, 2016.

[60] Firebase Authentication. *Firebase Authentication*. https://firebase.google.com/docs/auth.

[61] Amazon Web Services. *What is PostgreSQL?* https://aws.amazon.com/rds/postgresql/what-is-postgresql/.

[62] Richard Peterson. *What is PostgreSQL? Introduction, Advantages and Disadvantages*. https://www.guru99.com/introduction-postgresql.html.

[63] Antonios Makris et al. “Performance Evaluation of MongoDB and PostgreSQL for spatio-temporal data.” In: *EDBT/ICDT 2019 Joint Conference* (2019). doi: http://ceur-ws.org/Vol-2322/BMDA_3.pdf.

[64] SQLite. *About SQLite*. https://www.sqlite.org/about.html.

[65] Hardik Gajera, Shruti Naik, and Manik Lal Das. *On the Security of “Verifiable Privacy-Preserving Monitoring for Cloud-Assisted mHealth Systems”*. doi: https://doi.org/10.1007/978-3-319-49806-5_17. Sydney, Australia, 2016.

[66] David Chappell. “Introducing Windows Azure.” In: *Microsoft Inc* (2010). https://www.idt-inc.com/wp-content/uploads/sites/4755/2017/05/IntroducingWindowsAzureFinal_5_5_2011_9_55_46_AM.pdf.

[67] Microsoft. *Introduction to Microsft Azure Database for PostgreSQL*. https://azure.microsoft.com/nb-no/resources/introduction-to-azure-database-for-postgresql/. 2021.

[68] Microsoft. *What is Flexible Server in Azure Database for PostgreSQL?* https://techcommunity.microsoft.com/t5/azure-database-for-postgresql/what-is-flexible-server-in-azure-database-for-postgresql/ba-p/1741346.

[69] M S.Bhiogade. *Secure Socket Layer*. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1091.7261&rep=rep1&type=pdf. 2002.

[70] Ian Summerville. *Software Engineering 10th edition*. Pearson Education Limited, 2015.

[71] Simon Grimm. *Choosing the Right Data Storage Solution: Ionic Storage, Capacitor Storage, SQLite, or Ionic Secure Storage?* https://ionicframework.com/blog/choosing-a-data-storage-solution-ionic-storage-capacitor-storage-sqlite-or-ionic-secure-storage/.

[72] Mozilla developers. *Window.localStorage*. https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage.

[73] PouchDB. *About PouchDB*. https://pouchdb.com/learn.html.

[74] Mozilla developers. *IndexedDB API*. https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API.

[75] Capacitor Storage. *@capacitor/storage*. https://capacitorjs.com/docs/apis/storage.

[76] Mozilla web dev. *Browser storage limits and eviction criteria*. https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Browser_storage_limits_and_eviction_criteria.

[77] Dallas Swendeman et al. "Longitudinal Validity and Reliability of Brief Smartphone Self-Monitoring of Diet, Stress, and Physical Activity in a Diverse Sample of Mothers." In: *JMIR MENTAL HEALTH* 6.9 (2018). doi: https://doi.org/10.2196/mhealth.9378.

[78] Amazon Web Services. *Global Infrastructure*. https://aws.amazon.com/about-aws/global-infrastructure/.

[79] Microsoft. *Data Residency in Azure*. https://azure.microsoft.com/en-us/global-infrastructure/data-residency.

[80] Google Cloud. *Bucket Locations*. https://cloud.google.com/storage/docs/locations.

[81] IBM Cloud. *IBM Cloud Global Data centers*. https://www.ibm.com/cloud/data-centers.

[82] Oracle Cloud. *Oracle Global Infrastructure Regions*. https://www.oracle.com/cloud/cloud-regions/data-regions.

[83] Alibaba Cloud. *Alibaba Cloud Global Infrastructure*. https://www.alibabacloud.com/global-locations.

[84] Salesforce. *Where is my Salesforce instance located?* https://help.salesforce.com/s/articleView?id=000314281&type=1.

[85] SAP. *Data Center Locations*. https://www.sap.com/about/trust-center/data-center.html?mode=region&currentLevel=continent&continentId=Europe.

[86] Rackspace Cloud. *A global data center footprint*. https://www.rackspace.com/about/data-centers.

[87] VMWare. *VMWare Global Infrastructure*. https://www.vmware.com/global-infrastructure.html.

[88] Richard Boyett. *What is MySQL: MySQL Explained For Beginners*. https://www.hostinger.com/tutorials/what-is-mysql.

[89] PostgreSQL tutorial. *What is PostgreSQL?* https://www.postgresqltutorial.com/what-is-postgresql/.

[90] Bridget Botelho and Jack Vaughan. *MongoDB*. https://www.techtarget.com/searchdatamanagement/definition/MongoDB.

[91] SQLServer Tutorial. *What is SQL Server*. https://www.sqlservertutorial.net/getting-started/what-is-sql-server/.

[92] Pushpa Sekhara Reddy Matli. *Overview of Azure Cosmos DB*. https://www.red-gate.com/simple-talk/cloud/azure/overview-of-azure-cosmos-db/.

[93] MariaDB Tutorial. *What is MariaDB? How Does MariaDB Work?* https://www.tecmint.com/what-is-mariadb-how-does-mariadb-work/.

[94] I. S. Vershinin and A. R. Mustafina. "Performance Analysis of PostgreSQL, MySQL, Microsoft SQL Server Systems Based on TPC-H Tests." In: *2021 International Russian Automation Conference (RusAutoCon)* (2021). doi: https://doi.org/10.1109/RusAutoCon52004.2021.9537400.

[95] Facebook. *Login Security*. https://developers.facebook.com/docs/facebook-login/security/.

[96] Laurence Moroney. *Introducing Firebase Authentication*. https://firebase.blog/posts/2016/06/introducing-firebase-authentication.

[97] Matt Raible. *Build an Ionic App with User Authentication*. https://developer.okta.com/blog/2017/08/22/build-an-ionic-app-with-user-authentication.

[98] Ory. *Ory Cloud*. https://github.com/ory/hydra.

[99] Ory. *OAuth2 & OpenID Connect (Ory Hydra) Introducton*. https://www.ory.sh/docs/hydra.

[100] Microsoft Reporter. *Microsoft announces plans to establish a new datacenter region in Denmark to accelerate the country's green, digital transformation*. https://news.microsoft.com/europe/features/microsoft-announces-plans-to-establish-a-new-datacenter-region-in-denmark-to-accelerate-the-countrys-green-digital-transformation/.

[101] Terrance Liu et al. *Multimodal Privacy-preserving Mood Prediction from Mobile Data: A Preliminary Study*. doi: https://arxiv.org/abs/2012.02359. 2020.