



JANUARY 21, 2024

GRADUATION REPORT

AUTHENTICATION ENHANCEMENT

NURHAN ABULCAI

COMPANY: PARANTION BV GROUP

University: Saxion University of Applied Sciences



Summary

During the graduation at Parantion, I have working and collaborating in the front-end department of the company, under the supervision of Nick Henzen from Parantion and Manasse Siekmans from Saxion and collaborating with employees from the back-end department and the UI/UX designer, for additional advice, support and guidance.

The project consists of, as the title suggests, an enhancement of the authentication process for the Parantion products, in special, Scorion, which is the main product being highly developed and maintained. The company assigned me with the task of offering the users of Scorion the possibility to password less login through various methods of authentication.

During this period, I researched, analyzed and developed the established goal in great details covering in my opinion and experience all of the aspects of password less authentication, mobile applications development frameworks, backend integrations and communications, security and risk management.

The value of this project consists of offering the company a detailed and structured analysis of different approaches in the mobile application development field, password less authentication methods and systems, risk management and security measures, as well as prototypes to support my research and the main project product which are 3 methods of authentication, 2 relying on the support of a mobile application.

Table of Contents

Summary	1
Table of figures	5
1 Introduction.....	6
1.1 Stakeholders.....	6
2 Research	7
2.1 Backend communication protocols and frameworks.....	7
2.1.1 What types of backend communications exist?	7
2.1.2 What are the advantages and disadvantages of the backend communications solutions? 9	
2.1.3 What is the expected user traffic of the products?	10
2.1.4 What backend solutions can we use to send notifications to the user’s mobile phone application?	13
2.2 Hybrid mobile development frameworks	14
2.2.1 What is hybrid mobile development?	14
2.2.2 What are the current hybrid mobile development frameworks?	15
2.2.3 What are the advantages and disadvantages between a hybrid framework and a native framework?	16
2.2.4 What is the performance of each hybrid framework and how effective they are?	17
2.2.5 What is the performance of a PWA (Progressive Web Application) as a hybrid mobile app solution?	17
2.2.6 Prototypes experience.....	18
2.2.7 Evaluation and Comparison	18
2.3 Methods of authentication	20
2.3.1 How can WebAuthn be used as a password less authentication method?.....	20
2.4 What are the most common methods of password less authentication?.....	21
2.4.1 FIDO2 keys	21
2.4.2 Software tokens	22
2.4.3 Phone call / SMS	22
2.4.4 Email OTP.....	22
2.5 How can a mobile application enhance the password less authentication process?.....	22
2.5.1 Biometrics check.....	22
2.5.2 QR code scanning	22
2.5.3 OTP generation	22
2.6 Conclusion and decisions	23

3	Process.....	24
4	Quality assurance	26
4.1	Unit testing.....	26
4.2	Regression testing	26
4.3	Widget testing.....	26
4.4	Linting.....	26
4.5	Android and iOS testing.	26
5	Requirements	27
5.1	Functional requirements.....	27
5.2	Non-Functional requirements.....	27
6	Product	29
6.1	Design.....	29
6.1.1	Front-end	29
6.1.2	Back-end	30
6.1.3	Mobile application	31
6.1.4	Database	32
6.2	Implementation	33
6.2.1	Method of authentication	33
6.2.2	Back-end	43
6.2.3	Mobile application	44
7	Testing.....	45
8	Conclusion and recommendation	46
9	Reflection.....	47
10	Bibliograph	48
	Appendices	50
	Appendix 1: Ionic development experience.....	50
	Appendix 2: Flutter community.....	50
	Appendix 3: Android and iOS market shares across the continents.	50
	Appendix 4: The results of the benchmark tests.....	50
	Appendix 5: Flutter vs Native vs React-Native performance results.	52
	Appendix 6: Hybrid development experience and PWA	54
	Flutter	54
	React native.....	54
	Ionic.....	55
	PWA.....	55
	Appendix 7: Backlog and changes	56

Appendix 8: Linting criteria for Flutter	57
Appendix 9: Regression tests.....	60
Appendix 10: hybrid vs native performance	62
Performance	62
UI/UX	62
Audience.....	62
Security.....	64
Costs and time.	64
Appendix 11: PWA advantages and disadvantages	65
Appendix 12: Backlog and issues.....	66
Appendix 13: Mobile testing	69
Appendix 14: Testing methods of authentication and email validation.....	71
Laptop fingerprint using WebAuthn.....	71
Mobile QR code login	72
Mobile Biometrics login.....	74

Table of figures

Figure 1-Key encryption flow.....	20
Figure 2-System components diagram	29
Figure 3-Front-end design	30
Figure 4-Mobile app initial design	31
Figure 5-Numbers choice validation.....	32
Figure 6-Number input validation	32
Figure 7-Database diagram.....	33
Figure 8-WebAuthn registration flow	34
Figure 9-WebAuthn authentication flow	35
Figure 10-WebAuthn full flow	37
Figure 11-Mobile application registration process.....	40
Figure 12-Mobile application authentication process.....	42
Figure 13-Final mobile application design.....	44
Figure 14-WebSocket sessions testing	45
Figure 15-Gauss-Legendre android results.....	52
Figure 16-Gauss-Legendre iOS results.....	52
Figure 17-Borwein algorithm iOSresults.....	53
Figure 18-Borwein algorithm android results	53
Figure 19-StatCounter android vs iOS global share.....	63
Figure 20-StatCounter android vs iOS N.A. market share	63
Figure 21-Issue example	66
Figure 22-Parent issue	67
Figure 23-Sprint 1	68
Figure 24-Sprint 2	68
Figure 25-Sprint 3	68
Figure 26-Sprint 4	68
Figure 27-Mobile testing proof.....	69
Figure 28-Emulator UI	70
Figure 29-Phone status and language proof.....	70
Figure 30-WebAuthn Windows Hello prompt for authentication and registration	71
Figure 31-QR code display	72
Figure 32-Email code validation input.....	73
Figure 33-Email received proof	73
Figure 34-Authentication / Registration successful.....	74

1 Introduction

The company of Parantion produces and maintains software for sectors such as education, healthcare, government, and private companies. It was established in 1999 and is located inside of the city of Deventer. The company has a strong focus on developing talent and performing online research. With their use of modern-day technologies and the general atmosphere of company garners a lot of positive attention among students. This have facilitated a healthy relationship with the school of Saxion where many students have performed internships and graduation assignments at the company, me included.

Parantion emphasizes a big importance of data security which is why the company is officially ISO: 27001 and NEN: 7510 certified. This means that all their data is private and secure. It is only handled here in the Netherlands and will never be provided to any outside parties. This is important aspect of the company which facilitates a big trust with its customers.

The company consists of a little less than thirty people which are split up into three different departments: Customer-care, Operation and Development. Each department plays its own significant role in the company and has one person who assumes a managerial role to structure the workflow. Development department, which is split up into two teams, back-end and front-end.

Before starting my graduation period, I have been working at this company since February 2023, which allowed me to better understand the company goals, products and values, which helped me in achieving a productive and valuable graduation period.

Parantion is aiming to improve the login authentication component by introducing multiple authentication methods besides the username and password and in some cases 2FA. This enhancement will add extra layers of security of accessing sensitive data of the user's accounts, by implementing multiple layers of authentication for a user to successfully log in to the product and improve the user experience by offering a fast and easy ways to log into the product, without relying solely on username and password.

The graduation assignment is then extracted from this necessity, which tasked me into researching, analysing and developing various method of authentication, to allow the users to password less login into the product.

The main requirements of the company from the start was to implement a method of authentication that uses the incorporated fingerprint (if available) or pin code of the user's laptop, this will create a faster and easier process of identification of the user who is trying to login, and the 2nd requirement is to research and analyse the benefits and methods of authentication that can be created with the usage of a mobile application as support for identification of the user and granting access into the product. The mobile application research should also include the possibility of having a Progressive Web Application (PWA) as an application on the user's phone to allow the verification and authentication of the users through it.

1.1 Stakeholders

Below I crated a table displaying the stakeholders during this project, people who contributed to the project outcome, and overall interests into the project.

Table 1-Stakeholders

Name	Company	Position	Contact information
Nick Henzen	Parantion	Front-end Software engineer	nick.henzen@scorion.com
Manasse Siekmans	Saxion	University graduation supervisor	m.siekmans@saxion.nl
Randy Groot Roessink	Parantion	Back-end Software engineer	randy.grootroessink@scorion.com
Marcel van Eijk	Parantion	Medium back-end software engineer	marcel.vaneijk@scorion.com
Jericho Thijssen	Parantion	UI/UX designer	jericho.thijssen@scorion.com

2 Research

For this graduation project I was tasked on finding a solution to how to enhance the authentication process of the users using the products, this solution including the support of a mobile application and analysis of the best frameworks for this problem.

Drawing from the main problem that Parantion assigned me to find a solution through this assignment I want to answer the following question:

What is the ideal system architecture for a secure, efficient, and adaptable data-sharing mechanism to allow an enhanced user authentication experience through a variety of methods of authentication with the support of a hybrid mobile application?

During the graduation project I had three main fields of research that needed to be completed, analysed to draw a concrete system implementation and development roadmap.

2.1 Backend communication protocols and frameworks

The authentication enhancement project consists of 3 main part which are the front-end, mobile application and the backend communication between the first two, which acts as a bridge to allow the frontend to know when and if the user was successful in providing a check in his identity.

The main goal of this research is to discover what possible means of communication can be established, such as RESTful APIs, Web Sockets, Microservices and more, to achieve the goal of the project.

The contribution of this research for Parantion is to improve the decision-making process on future services projects that involve specialized backend protocols for a specific use case in a particular product. By investing in research in this domain, Parantion can gain a competitive edge by optimizing its network communication, enhancing data security, and improving overall system performance. Understanding and implementing robust backend protocols can lead to faster response times, reduced latency, and increased scalability, all of which contribute to a more seamless user experience. Additionally, staying abreast of the latest advancements in backend protocols ensures that Parantion remains adaptable to evolving technological landscapes.

The main question for this research paper is:

What backend solutions would be the most efficient, secure, and maintainable for the data sharing and communication between the front-end and mobile application?

To reach a concrete solution and conclusion a few sub questions have been established to support and provide evidence for the main question and create a clear picture of every possible option that can be used for this project.

- **What types of backend communications exist?**
- **What are the advantages and disadvantages of the backend communications solutions?**
- **What is the expected user traffic of the products?**
- **What backend solutions can we use to send notifications to the user's mobile phone application?**

2.1.1 What types of backend communications exist?

A backend communication type is a method in which data and communication between the user and the server is being achieved, a few of the most popular backend communication designs are:

2.1.1.1 Request-Response

The Request-Response pattern is a way of organizing the communication between a user and a server in which the user sends a request to the server and the server responds with the requested data or performs the requested action. (Kartik, 2023)

In this case the user initiates the communication by sending requests, that contains information about what the user is requesting, data transfer or action to be performed, the server receives and processes the request and

returning either the requested data or message on the status of the action performed (successful or unsuccessful). (Kartik, 2023)

This pattern is used in many different types of applications and is often implemented using the HTTP protocol. (Kartik, 2023)

2.1.1.2 Push pattern

In the push design pattern for backend communications, data or notifications are pushed from a sender to a receiver without the receiver having to request the data. This allows the backend to proactively send information to clients or other systems, rather than waiting for them to request the data. (Kartik, 2023)

The push design pattern is a communication pattern that can be used in backend systems to send data or notifications from the backend to clients or other systems. It is a way for the backend to push information to receivers, rather than requiring the receivers to request the information.

The most common practices for the push pattern are in push notifications or in messaging applications, and technologies like WebSocket and Server-Send-Events (SSE) are commonly used to implement such method of communication. (Enyinna, 2023)

2.1.1.3 Short Polling

Short polling is a technique used in backend design where the client repeatedly sends requests to the server at a fixed interval to check for updates or new data. The server responds with the current state of the data or with any updates that have occurred since the last request. The interval at which the client sends requests is known as the “polling interval” and is typically short, such as every few seconds. (Kartik, 2023)

The main difference between request-response and short polling is that the user sends periodic requests regardless of whether there is new data available or not, these constant flows of requests give the appearance of real-time updates.

2.1.1.4 Long Polling

Long polling is the same technique as short polling but instead of the client repeatedly sending request to the server it send one request and waits for the server response on that request, the server holds the requests and when updates or data arrives it sends the data, or in case the set timeout has been reached it will give a response. (Kartik, 2023)

2.1.1.5 Server Sent Events

Server sent events (SSE) allows a server to push updates to a client in real time. In SSE the request is as simple and same as other process, but the response of the server is large or a stream.

It is based on the concept of a long-lived HTTP connection between the client and server. This connection remains open for as long as the client wants to receive updates and can be closed by either the client or the server. (Kartik, 2023)

2.1.1.6 Publish Subscribe (Pub/Sub)

The publisher-subscriber model is a design pattern where a sender, known as the publisher, sends a message or event to a messaging system, and one or more receivers, known as subscribers, receive and process that message or event. The publisher and subscribers are decoupled from each other and communicate indirectly through the messaging system. (Kartik, 2023)

The publisher doesn't need to know how many subscribers are listening or who they are. Subscribers can come and go dynamically without affecting the publisher.

This pattern is commonly used in event-driven systems, where the publisher sends an event, and the subscribers are notified of the event as it happens. It allows for a flexible and scalable architecture, where new subscribers can easily be added or removed without affecting the publisher or other subscribers. (Enyinna, 2023)

2.1.2 What are the advantages and disadvantages of the backend communications solutions?

The advantages and disadvantages of each backend communication protocol have been concluded from various articles (Kartik, 2023) (Enyinna, 2023) (Pabian, 2021) during the research phase and drawn into a table to visualize the pros and cons of each of the protocol.

Table 2-Advantages and Disadvantages backend communications protocols

Method	Advantages	Disadvantages
Request-Response	<ul style="list-style-type: none"> • Simple data retriever, server-side rendering, API calls. • Can handle large number of requests. • It ensures reliability as a response is always received. • Error handling 	<ul style="list-style-type: none"> • The user must wait for the response before it proceeds. • Repetition and inefficiency can occur. • Delays • No real-time communication.
Push pattern	<ul style="list-style-type: none"> • Real time updates • Offloading the responsibility of requesting data from users to the server. • Reduce latency of communication between the server and users • Control over what data is sent. • Dual communication between the server and client. • Live and unique connections. 	<ul style="list-style-type: none"> • The level of complexity for maintaining and implementing a push system is greater than a request-response system. • Reliability of push notifications being reached is not great as the user can be offline. • Performance overload of the system as it is constantly pushing data on the network, it will consume bandwidth and CPU time. • Security needs to be enhanced as sensitive data can be constantly pushed on the network, requiring additional security measures to be implemented.
Short polling	<ul style="list-style-type: none"> • Easy to implement. • Low latency • Working in restricted environments (behind firewalls, low internet connection). 	<ul style="list-style-type: none"> • High server load, as the server must process a high number of requests and responses. • High network traffic • Short polling will consume the battery of the device as it is constantly making requests. • Inefficiency by constantly polling data from the server that it might not have or takes a long time for the server to process.

Long polling	<ul style="list-style-type: none"> • Small number of requests and responses sent by the user, decreasing server load and network traffic. • Updates come faster as the user only needs to wait for the response. • Working in restricted environments (behind firewalls, low internet connection). 	<ul style="list-style-type: none"> • Complexity in the server-side operations, to handle open requests. • Management for timeout requests • Scalability is limited as the server needs to keep a connection open for the user. • Power consumption as it drains battery to keep the connection open on the device.
Server Sent Events	<ul style="list-style-type: none"> • Easy implementation. • Low overhead as it only sends updates as text or JSON data. • Efficient in sending data as no new connections need to be made for each update. • One way communication as the user cannot send data to the server 	<ul style="list-style-type: none"> • One way of communication as the user cannot send data to the server can be a disadvantage. • Limited data types, SSE supports only text and JSON data. • Not suitable for duplex communication.
Publish Subscribe (Pub/Sub)	<ul style="list-style-type: none"> • The publisher and subscribers do not have direct knowledge of each other in order to communicate, offering the possibility to update parts of the system without compromising it. • It can handle a large number of users as there is no need to keep track of them individually. • Event-driven architecture, the system is suited for notifications made by the publisher. • As the publisher and subscribers don't need to communicate it allows the system to support different types of components than can have a fast or slow processing time. 	<ul style="list-style-type: none"> • The complexity to maintain and implement is high. • A message can take a long time to be received by the subscribers in case there is a large number of them. • Additional load will be put on the system as messages need to be sent and received by the system. • The integrity of the messages in the system can be tampered with or intercepted while in transit. • Duplication or loss of messages during distribution.

2.1.3 What is the expected user traffic of the products?

The purpose of this question and the investigation in this area is to visualize the workload of a server that might suffer if the authentication enhancement is online, this will help in the decision making of the protocol of communication and how to make the most efficient back-end solution.

After conducting an interview with the company medium backend developer, Marcel van Eijk, I had the opportunity to ask him questions regarding the company backend and user traffic load.

2.1.3.1 How many users are currently using the products?

According to Mr. Eijk, the exact number of users currently registered in the system could not be provided accurately, but an approximation is that it is in the thousands.

2.1.3.2 What is the peak number of access requests?

Usually, every hour there are thousands of access requests, he provided with a log file of every login access request from the past 2 months (August, September) by date and hour.

The data tables with calculations on the number of requests each hour and each day in the data period provided can be viewed on the next page.

After analyzing the data it can be clearly view that the highest number of access are during the academic year and is lower during the summer/holiday period, as compared to the busiest Hour on August being noon 14:00, 4324 requests, to the busiest Hour on September in the morning at 11:00, 9106 requests, it is an increase of 110.58%, and the busiest Day on August being Tuesday, 10197 Requests, with the busiest Day on September being also Tuesday, with 19705 it has an increase of 93.16%.

One of the most interesting aspects that can be viewed in the data table is that the majority of access requests are happening during the day after 08 which corelates to the teaching hours in the average universities are operating, decreasing to the towards the end of the day (17:00) and having another peak of access during the evening after dinner time (19:00 -> 21:00), this can be due to students using the products to complete their tasks/homework.

Table 3-Access requests analysis

Below are a few tables with the total number of access requests per hour and day for the months of August and September.

Hour/Month	00	01	02	03	04	05	06	07	08	09	10	11
08	1360	294	194	149	152	179	337	978	2490	3745	4103	4255
09	1814	401	283	282	279	335	781	2279	5644	7752	8645	9106
Growth%	33.38235	36.39456	45.87629	89.26174	83.55263	87.15084	131.7507	133.0266	126.6667	106.996	110.6995	114.0071

Hour/Month	12	13	14	15	16	17	18	19	20	21	22	23
08	3496	4083	4324	4015	3410	2025	1637	2633	2180	1974	1429	663
09	7153	8234	7153	7429	6539	4132	3742	8964	9094	7088	4125	1807
Growth%	104.6053	101.6654	65.42553	85.03113	91.75953	104.0494	128.5889	240.4482	317.156	259.0679	188.6634	172.549

Month/Day	08	09	Growth%
Monday	8166	18239	123.3529
Tuesday	10197	19705	93.24311
Wednesday	9270	18454	99.07228
Thursday	9913	19697	98.69868
Friday	5563	18482	232.2308
Saturday	3208	9792	205.2369
Sunday	3788	9408	148.3633

2.1.4 What backend solutions can we use to send notifications to the user's mobile phone application?

One of the features requested to be explored was the implementation of push notification for the mobile application, which can help in adding additional features.

The notification system can send an authentication request to the user's mobile phone without add authenticate in a pop-up window (as how Microsoft authenticator works) without the need to open the application, which would have been a great feature making the user using the application in one step.

2.1.4.1 *How do notifications work under the hood?*

The mobile application upon installation and receives permission from the user to receive notifications the app registers with the respective platform's push notification service. This might be Firebase Cloud Messaging (FCM) (Firebase, n.d.) for Android or Apple Push Notification Service (APNS) (Apple, n.d.) for iOS, as both are directly connected to the device via Google play services and Apple push notification service.

Then through these services push notifications are sent to the users even if the app is closed, as being registered to the underlying phone services you can send data to it even if the app is sleeping.

A few possible solutions in sending the mobile application notifications are:

2.1.4.2 *Own and create a notification server.*

This solution is the hardest and most difficult method to implement such features as it is necessary for the server to have a connection through the main Android and Apple notifications services (FCM and APNs), which is done through certificates contracts. (Apple, n.d.)

2.1.4.3 *Subscribe to one of the services.*

The most easy and reliable solution to ensure that the notifications are send correctly and efficiently is though the paid services of FCM and APN, or any other 3rd party that are having the base on these 2 services such as:

- OneSignal (OneSignal, n.d.)
- Pusher (Pusher, n.d.)
- PushWoosh (PushWoosh, n.d.)

This method involves having a paid subscription to keep the notifications service up and running and depending on the service it can either charge a monthly subscription or per request subscription.

2.2 Hybrid mobile development frameworks

The choice of having a mobile application for the authentication enhancement is to utilize the current mobile phone features and sensors for fingerprint or/and face recognition checks that will help in having a secure authentication into the company products. The usage of each user's personal phone to access the products either by using the biometrics sensors or just inputting a pin or one time password generated from the app, access without permission or by other individuals other than the user with the correct credentials will not be permitted.

A hybrid mobile application is the solution to accomplishing this task and it will allow the company complete control over the development and features of the app.

By comprehensively exploring the diverse landscape of these frameworks, Parantion gains insights into the distinct advantages and disadvantages associated with each, facilitating informed decision-making in mobile app development. Evaluating the trade-offs in terms of development speed, cross-platform compatibility, and user interface fidelity provides a nuanced perspective crucial for optimal resource allocation. Furthermore, a focused investigation into the performance characteristics of these frameworks ensures that Parantion can deliver high-quality mobile applications with optimal speed, responsiveness, and user experience.

The main question for this research paper is:

Which hybrid mobile development framework would be most suitable for this project involving features that will support using the biometrics sensors of the phone?

The goal is to identify the most effective mobile development framework that can efficiently support the development of a secure, user-friendly authentication system support for the front-end application. Factors considered will include framework performance, ease of use, support for security features, cross-platform capabilities, and community support.

From this goal we can extract the sub questions:

- **What is a hybrid mobile framework and how does it function?**
- **What are the current hybrid mobile development frameworks?**
- **What are the advantages and disadvantages between a hybrid framework and a native framework?**
- **What is the performance of each hybrid framework and how effective they are?**
- **What is the performance of a PWA (Progressive Web Application) as a hybrid mobile app solution?**

These sub questions will help in drawing a conclusion to the main question of this report then based on the researched information draw a conclusion on the best hybrid framework for the project.

2.2.1 What is hybrid mobile development?

Hybrid app development in simple terms is building a single mobile application compatible with all mobile platforms Android, IOS and Web using a single code base, compared to native app development where each platform is written in specific platforms.

Before the appearance of hybrid frameworks, developers were having the choice of either mastering development for IOS devices by learning Swift or Objective-C, and Kotlin or Java for Android devices, which meant that companies had to spend twice as much time and resources for a single mobile application in order for it to be delivered to the IOS and Android market. With the increased popularity of hybrid development, the industry changed by having the possibility of releasing applications faster with fewer resources consumed, many companies such as Google, Twitter, Instagram etc. have switched and are in the process of fully adopting the hybrid development of their products, popular example of hybrid apps are Gmail, Uber and eBay.

By having a single code base the maintenance, update and the costs are reduced, keeping track of the development is much easier and clearer as it can only be one development team working in the same rhythm without having to rely on progress synchronization of both the IOS and Android teams in case of native development.

2.2.2 What are the current hybrid mobile development frameworks?

In this research I will choose to present the 3 most popular hybrid development frameworks, as all of them have an increasing popularity and have a strong community base which will provide sufficient data to analyze and compare later.

2.2.2.1 Ionic

Ionic is a popular cross-platform mobile app development which allows the developers to create using a single code base written in web technologies such as Vue, React and Angular, high-quality apps for mobile and desktop, allowing developers to deploy the apps on the native platforms IOS and Android as well as Progressive Web App for the web development, everything in a single code base with HTML, CSS and JavaScript as the backbone of the framework. (ionic, n.d.)

Ionic focuses on the front-end UX and UI interaction of an app (UI controls, interactions, gestures, animations).

Ionic's Capacitor is the main cross-platform native runtime that makes the Web application built in Ionic + React/Vue/Angular run natively on IOS and Android using modern web tooling. Capacitor creates Web Native apps and provides a consistent, web-focused set of APIs that enable an app to stay as close to web standards as possible, while accessing rich native device features on platforms that support them. (Lynch, n.d.)

The community and support for Ionic's Capacitor is rather small with a total of approximately 6.800 threads on Stack Overflow, (Stackoverflow, n.d.) and about the same GitHub repositories (Github, n.d.), which is a reflection that the community is not large and that much supported.

Ionic's Capacitor can be used most often in MVC mobile applications, applications with a heavy backend integration and light weight and fast applications and provides a set of APIs that allow the develop to access native device features like the camera, accelerometer, and file system, and it is a fast and efficient development framework, by using less memory which will make the app load faster and run smoothly.

According to "[Appendix 1: Ionic development experience](#)" the development challenges with Ionic, including inefficient hot reloads, null safety issues in JavaScript, and compatibility issues with existing libraries, underscore the need for careful consideration and potential workarounds in utilizing this framework for mobile app development. (capacitor, n.d.)

2.2.2.2 Flutter

Flutter is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase, which combines a high-performance graphics engine with the dart programming language, and compiles to the native machine code which results in fast processing and efficient complex applications.

By using Dart, Flutter benefits from a type safe and static type checking development process which ensures that a variable's value always matches the variable's static type, but also offering the flexibility of using the dynamic type combined with runtime checks which can be useful during experimentation or for code that needs to be especially dynamic, and by having the build-in null safety it will ensure that the Flutter application will be protected from null exceptions at runtime, which error proofs the application. (Flutter, n.d.)

The developer experience when working with Flutter is very fast and smooth, as it provides fast incremental compilation with stateful hot reload, which increases productivity and efficiency by not reloading the whole application with every change but only focus on the specific parts.

As highlighted in "[Appendix 2: Flutter community](#)", the flutter community is large and keeps growing offering a lot of support for new developers with various examples and real-life applications.

Flutter is a growing framework with features and possibilities, the next step from hybrid development is combining the possibility to have hybrid applications for IOS and Android as well as embedding the same application into multiple technologies, the biggest example is Toyota developing the main entertainment and infotainment systems using Flutter, the reasons being the high performance + AOT consistency, smartphone-tier touch mechanics and developer ergonomics that Flutter provides. (Flutter, n.d.)

2.2.2.3 *React native.*

React Native is an open-source UI software framework created by Meta Platforms, Inc. It is used to develop applications for Android, Android TV, iOS, macOS, tv OS, Web, Windows and UWP by enabling developers to use the React framework along with native platform capabilities. (React Native, n.d.)

In the same case as Ionic, react native is based on JavaScript with the addition of its own components and APIs for building mobile user interfaces, using a combination of both native code (written in Java, Objective-C or Swift) with JavaScript to render on the device.

Because React uses a different approach when communicating between the JavaScript code and native modules It uses a bridge that provides a better compilation between the JavaScript libraries and the native code of the mobile operating system, which leads to better performance and flexibility.

The developer experience when working using React Native is fluid and similar to working on a web application, having efficient fast reload with 90% of code being reusable across Android and IOS, with a lot of compatible plugins with the device hardware components, and because of the high majority of developers familiar with JavaScript it takes less time to learn and adapt to the framework.

The community of React Native is large and in 2nd place to Flutter as being one of the most popular choices of hybrid development with approximately 168.000 threads on Stack Overflow and approximately 428.000 repositories on GitHub, and being maintained by the company Meta, it offers great support from the Meta development community.

Some of the popular applications developed using React native are Facebook, Facebook messenger, outlook, teams, discord etc. which proves that React native is great for large scale user applications that require a high level of complexity. (React Native, n.d.)

2.2.3 What are the advantages and disadvantages between a hybrid framework and a native framework?

When talking about what are the advantages and disadvantages between a hybrid framework and a native framework, we must first set a few important categories that will highlight the performance of each framework.

Every mobile operating system (Android and IOS) is written in High-level programming languages such as Swift, C, C++, Objective-C, Java and Kotlin. A high-level programming language is a programming language that can communicate the closest with the hardware components of the device, this increasing the efficiency of each computation that a program must do. (Grigalashvili, n.d.)

Native development is written using these high-level programming languages which offer the developer access to every possible feature a mobile device has to offer, by having a direct means of communication with little to no compilation of the written code.

Hybrid development is written in different kinds of programming languages which are not considered high-level programming languages and require bridges and engines to compile the written code to native code. This method highly influences the efficiency of the application as more computing power is necessary for accessing certain features.

The categories that every company and developer is looking when presenting the choice of picking a mobile development framework are performance, UI/UX, audience, security, costs, and time, which can be further read on [“Appendix 10: hybrid vs native performance”](#).

To summarize and reach a conclusion for this sub-question, choosing between a hybrid framework and a native framework highly depends on the company needs, time and finance are a key aspect of any business which highly influences the decision making. Hybrid frameworks are constantly improving are getting closer to native performance, allowing a single code base allows the developers to better keep track of the progress, versions and updates of the application which is a disadvantage for native development where every aspect of business and development must be doubled.

Native keeps up by allowing easier access to native elements of the devices and a higher security aspect of storing and working with sensitive data.

2.2.4 What is the performance of each hybrid framework and how effective they are?

The relevance of this sub-question is to highlight through extensive research the best hybrid framework in terms of performance as computing power and efficiency, this will support in the decision-making process of choosing the proper hybrid framework for the needs of the assignment.

During the research I encountered the research article “An empirical investigation of performance overhead in cross-platform mobile development frameworks” by Biørn-Hansen, A., Rieger, C., Grønli, TM., highlighting different benchmarks test of 5 different hybrid mobile frameworks (Ionic 3.9.2, React Native 0.53.2, Native Script 3.4.1, Flutter 0.5.1 MAML/MD2 2.0.0) and a Native framework.

According to the “[Appendix 4: The results of the benchmark tests](#)” it is concluded that the experiment unveiled distinct performance characteristics of various frameworks under different workloads. Flutter and Native Script emerged as top performers in terms of time-to-completion, showcasing efficiency in executing tasks. Ionic, although competitive in certain aspects, demonstrated potential delays in specific operations, raising concerns about its performance, especially in fetching geolocation data. In terms of CPU load, Flutter outperformed Ionic and React Native, emphasizing its effectiveness in handling computational tasks. Memory consumption results indicated that Flutter maintained consistently low usage, while React Native's lower results were attributed to alternative module implementations. In contrast, Ionic exhibited the highest memory consumption. These findings underscore the importance of considering specific performance metrics and workloads when selecting a mobile development framework, with Flutter and Native Script presenting strong contenders for optimal performance across various scenarios.

In the article published by Ihor Demedyuk and Naazr Tsybulskyi with the title “Flutter vs Native vs React-Native Examining performance” it is showcased how Flutter, React-Native and Native frameworks are performing under high CPU intensive tests using 2 computing algorithms.

Based on the results from the “[Appendix 5: Flutter vs Native vs React-Native performance results](#)” Flutter performance is very close to native performance and React-Native not coming even close to the values resulted from the Flutter tests.

2.2.5 What is the performance of a PWA (Progressive Web Application) as a hybrid mobile app solution?

A progressive web app (PWA) is an app that's built using web platform technologies, but that provides a user experience like that of a platform-specific app. Like a website, a PWA can run on multiple platforms and devices from a single codebase. Like a platform-specific app, it can be installed on the device, can operate while offline and in the background, and can integrate with the device and with other installed apps.

The relevance of this question is to investigate the possibility of Parantion using a PWA as a mobile solution for this assignment and future products, how secure, efficient and what applicable use cases are the most beneficial.

Concluding from the “[Appendix 11: PWA advantages and disadvantages](#)” the PWA has its advantages and disadvantages based on the requirements of the use case, it is a new technology that is rapidly developing into more flexible and secure option for mobile development without the need for installation through an app store, offering a great solution for simple in terms of functionality of applications.

2.2.6 Prototypes experience

To further expand my research into the most suitable hybrid mobile framework I am going to develop 4 small prototypes that will highlight the same function which will be a simple login page using the company API for authentication, as it would be for the final product and a main page where I can test the devices features:

- Fingerprint
- Face recognition and fingerprint sensors
- Camera.

Compare the development experience, how much time did it took to achieve the settled goal, how much support did I receive from the documentation and community, how intuitive is working in the specific framework and how well structured, error proof and efficient is to implement new features to each of them and write my evaluation for each of them.

Each framework provided a good insight into the development experience, challenges and difficulties, what are the benefits and what each framework lacks, and to draw a conclusion from "[Appendix 6: Hybrid development experience and PWA](#)" Flutter scored the best as it was the easiest framework to work with and offered a great variety of options to achieving the functionality set for this experiment, followed by React-Native and Ionic. The PWA proved to not be a great choice of mobile solution as it did not meet the goals specified for this experiment.

2.2.7 Evaluation and Comparison

Based on the presented evidence, it can be drawn out the ranking from best to worst hybrid framework to use as the date of this research paper.

The rankings on the table below are from 1 to 3, 1 being the lowest score and 3 being the highest score.

Table 4-Evaluation hybrid frameworks

Framework	Native performance	Development process	Accessibility	Community	Error proofing
Flutter	3	2	3	3	3
React Native	2	3	3	3	2
Ionic Capacitor	2	1	2	1	1

Flutter during this research proved to be the most interesting framework to work with as it is written in a high-level programming language. It offers the most error-proving development experience. The widget-based structure where the more you make each individual custom widget a separate file the better and more easily readable the code is. The only downside in Flutter is its biggest advantage which is the how meticulous the developer must be when contributing to the project, which for large projects will benefit as multiple developers can work on multiple parts of the project without frequent intersections.

React Native is an interesting approach to native development as if a developer is familiar with front-end development, mainly JavaScript, with CSS elements it will easily create and implement features and native designs in a short amount of time. The biggest downside of React Native is that JavaScript, not being a high-end programming language, is not able to communicate efficiently with the native components of a phone which will greatly influence the performance of the applications. It is a great choice for small and medium applications, that do not require advanced features to be implemented.

Ionic capacitor framework proved to be the lowest scoring hybrid framework due to the development experience and lack of in-depth documentation, especially for the installation and development process. The only good option was the usage of the plugin capacitor with the Quasar framework, but it limits the easily implemented native design of the components as it requires the developer to design each component

according to the native standards. As in the same case of React native being a framework and not communicating efficiently with the native components it has a great impact of the performance of the application. As Quasar having support for capacitor and offering Vue project the possibility to be transformed into mobile apps it offers for the company a big advantage as the main development framework for components and design for the company products are made using quasar with Vue.

Progressive Web Applications are an interesting technology, that can be used for simple applications that do not require the usage of advanced native features.

2.3 Methods of authentication

For this assignment I was tasked to implement a few methods of password less authentication into the Parantion products, the purpose of this research is to highlight the most popular, secure, efficient and common methods of authentication with and without the support of a mobile application.

For this assignment, one of the requirement methods of authentication to be research and implemented was the password less method of authentication using the laptop Windows Hello (PIN or fingerprint sensor of the laptop), for this reason a part of this research will be cover this requirement as it also supports the overall goal of the research.

The purpose of this research is to analyze and offer Parantion a variety of opportunities that can be developed for the goal of this assignment, this research can enhance security, improve user convenience, reduce password-related issues, ensure regulatory compliance, adapt to mobile usage, prevent fraud, provide a competitive advantage, and future-proof a company's authentication strategy, which are all a great benefit for Parantion.

The main research question for this topic is:

What password less methods of authentication can be used with or without a mobile application as a support?

From this main research question, I extracted the following sub-questions that can help support and give a concrete answer to the main research question.

- **How can WebAuthn be used as a password less authentication method?**
- **What are the most common methods of password less authentication?**
- **How can a mobile application enhance the password less authentication process?**

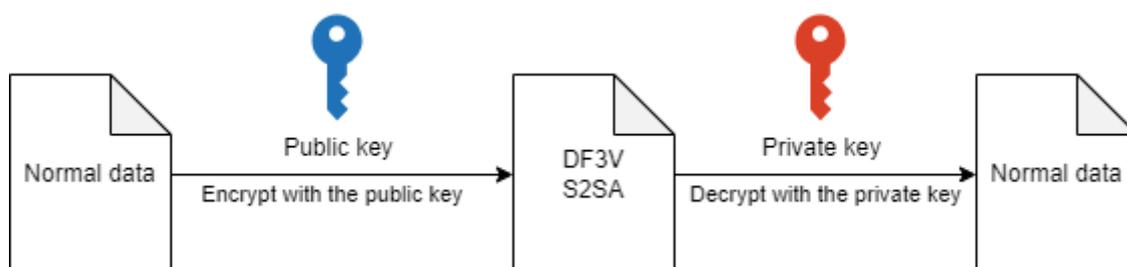
These sub-questions will guide into achieving the best methods of authentication to be developed for this assignment.

2.3.1 How can WebAuthn be used as a password less authentication method?

The Web Authentication API (WebAuthn) is a specification developed by the World Wide Web Consortium (W3C) and Fast Identity Online (FIDO) Alliance, with participation of Google, Mozilla, Microsoft, Yubico, and more. The API allows the servers to register and authenticate users using a public key cryptography instead of password, using strong authentication technologies that are built into devices such as Windows Hello, Apple's Touch ID. (WebAuthn, n.d.)

Public key cryptography also known as asymmetric cryptography was invented in the 1970s and was a solution for the problem of shared secrets being a pillar of modern internet security, using the concept of keypair, a private key that is stored securely with the user and a public key that can be shared with the server, these pairs are generated with cryptographic algorithms based on mathematical problems termed one-way functions. (WebAuthn, n.d.) (Wikipedia, n.d.)

Figure 1-Key encryption flow



The figure above is a demonstration of how a public and private key pair is used to encrypt and decrypt data. For example, in a communication between 2 Users, User 1 has the public key of User 2, he can encrypt the data

using that key that only the private key of User 1 can decrypt it, without the private key the data cannot be decrypted, User 2 using his private key decrypts the data and can visualize it.

In the case of WebAuthn, the public key of the user is stored on the server and used for the validation of a signed challenge by the private key upon the authentication process. (World Wide Web Consortium, n.d.)

The public key is not a secret in case of WebAuthn, because without the corresponding private key, it becomes useless and this implying a high security measure as the private key is never shared and securely stored on the user's device, leading to hackers and other malicious individuals not interested in databases storing public keys of users. (WebAuthn, n.d.)

During this research, I conducted a small prototype of how the WebAuthn can be implemented and used and was impressed by the large number of available solutions in different platforms and languages such as Ruby, Python, .NET, Java and much more (WebAuthn, n.d.) (World Wide Web Consortium, n.d.), and libraries that can support the proper validation of the signed data of the private key using the public key.

The prototype proven to be successful and very effective, achieving the goal of having a password less authentication process using the WebAuthn API, and providing the evidence that using WebAuthn for a method of password less authentication is a great option and be included in the development roadmap.

2.4 What are the most common methods of password less authentication?

When thinking about a password less authentication, one of the most common practices are through either email verification or the usage of a mobile application, a password less authentication works by using something that the user is in possession of or something that the user "is" to verify their identity and allow the system to properly give them access to the piece of software or application. (Magnusson, 2023)

The reason for investigating this research sub-question is to aid the decision-making process when establishing the methods of authentication that will be developed for this project. Parantion will benefit from this research through having an analysis on what are the most popular and common methods of authentication that might be implemented in future projects.

A traditional authentication method that requires only username and password are highly vulnerable as attackers can steal or even guess the access credentials to sensitive information and IT systems using a variety of hacking techniques. (Cyberark Glossary, n.d.).

Password less authentication strengthens security by eliminating risky password management practices and reducing attack vectors and improves the user experience by reducing the steps to authenticate the password and secrets fatigue, removing the need of remembering passwords, secrets and security questions answers. (Cyberark Glossary, n.d.)

A few popular methods of authentication are:

2.4.1 FIDO2 keys

There are 2 different types of FIDO keys authentication, one physical through a special device such as an USB, and a digital solution which relies on platform authenticators using the built-in security features of the devices such as laptops or smartphones. (Magnusson, 2023)

The first type of FIDO key authentication is through a USB device that can be plugged into the computer to allow the user to successfully provide his authorization to authenticate.

The second type of FIDO key authentication is using build-in authenticators such as Windows Hello, Mac OS TouchID on platforms such as Firefox, Microsoft Edge, Google Chrome and Apple Safari, this option in some cases supports the usage of a physical FIDO key, discussed earlier, to allow the user to authenticate, this offering more variety for the user to authenticate. (Cyberark Glossary, n.d.)

2.4.2 Software tokens

Software tokens are digital tokens sent to the user's smartphone, computer or tablet and consist of a one-time password which must be entered to provide the authentication. This method relies on a shared secret key and support OATH event based (HOTP) and time-based (TOTP) algorithms. (Magnusson, 2023)

Software tokens are usually used on two-factor authentication systems, and because the secret key for the generation of the software token is stored on the device, the risk of unwanted interception by a hacker or virus corruption can become a security risk. (Wikipedia, n.d.)

This method is usually used in combination with the username and password for an increase of security of the account.

2.4.3 Phone call / SMS

A method of authentication can be used through the usage of a phone call or SMS that can give the user a one-time password (OTP) or access approval links to the registered mobile number on the account.

The security risk associated with this method are lost phone case scenario or change of numbers.

2.4.4 Email OTP

Email one-time password (OTP) allows the user to receive a one-time password via the registered email address to complete in most cases a secondary authentication process. (Cyberark Glossary, n.d.)

This method should be used for extra authentication checks and not rely on solely email verification as it can create security risks and vulnerabilities associated with either email lost, changes or hacking.

2.5 How can a mobile application enhance the password less authentication process?

One of the most popular methods of authentication is through a mobile application authorization which offers the user a variety of options to use to successfully prove his identity and allow for a successful authentication.

A mobile application can support a few methods that can prove the identity of the user.

2.5.1 Biometrics check

This method of verification is through the usage of the built-in smartphone or tablet fingerprint, face or voice recognition, or retina scanning.

This method of verification can be found in popular mobile applications that rely heavily on keeping the security of the user as high as possible, such as mobile banking applications, authenticator applications (Microsoft Authenticator, Google Authenticator etc.), deducting that this method of providing a identify check to allow the authorization of the user is secure and efficient.

2.5.2 QR code scanning

By allowing the user to simply scan a unique QR code on the screen through a mobile application, it enhances the user experience and makes the authorization process much faster compared to a traditional username and password authentication process.

QR code scanning for login purposes is getting increasingly popular among software applications and websites, platforms like Steam or Discord allow the users to simply scan a QR code to login without relying on having to input the username and password, and just through one registered device on the user account to grant him access anywhere anytime using this method.

2.5.3 OTP generation

One-time password (OTP) generation can be implemented and used in a mobile application, a popular example of this functionality is Google Authenticator, which allows the user to provide an extra authentication check or in some system allowing the user to input the generated password as his password for authentication.

These 3 methods of authentication can enhance the user authentication experience and allow more secure and easy access to software products as it provides a high security solution.

2.6 Conclusion and decisions

To conclude the research “Which hybrid mobile development framework would be most suitable for this project involving features that will support using the biometrics sensors of the phone?” chapter, based on the evidence presented and discussed the best framework to choose for this project will be Flutter as it offers the best performance and native functionality, with great security and high scoring performance. Flutter can be used for complex and applications as performing the closes to the native development frameworks, having a great and large community with a lot of support developed and supported by one of the most professional company of developers, Google, it has the best potential for future hybrid development.

A conclusion to the “Backend communication protocols and frameworks” research, each communication protocol has its own advantages and disadvantages and there is no universal fit for every project as it highly depends on the use case and what is the final goal of an application. For this graduation project according to the research using a Push Pattern protocol for communication between the server and the client will offer the best solution as it offers several advantages over the traditional HTTP-based communication such as live-communication and updates from the server, this protocol is highly used in technologies such as Web Sockets and RabbitMQ (Enyinna, 2023) (Kartik, 2023) drawing from here and the familiarity of both the company development team and me as a developer the Web Socket would be a great choice for the development of a live communication backend between the front-end and back-end.

The conclusion of the “What password less methods of authentication can be used with or without a mobile application as a support?” research is that there are a variety of options to choose on what method of authentication would be best for a better validation of identity of the user, to comply and achieve the requirements of the assignment WebAuthn API would be used as a method of authentication as it proven based on the research that it is a possibility and can be achieved in the time frame of the assignment, as well as QR code login and Biometrics check authentication using a mobile application would be the most efficient, offering the user an enhanced user experience which is the one of the goals of the assignment.

These research materials conducted for the project answer the main question, offering a concrete and stable system, frameworks and the best methods to implement for the best user experiences. Allowing the users to choose between a mobile support as authentication and the own device authentication platform using the WebAuthn API, will allow a variety of options that will fit or be the most familiar for the users.

3 Process

Table 5-Timetable with competences

Week Date	1.1 04.09 08.09	1.2 11.09 15.09	1.3 18.09 22.09	1.4 25.09 29.09	1.5 02.10 06.10	1.6 09.10 13.10	1.7 16.10 20.10	Holiday 23.10 27.10	1.8 30.10 03.11	1.9 06.11 10.11	1.10 13.11 17.11	2.1 20.11 24.11	2.2 27.11 01.12	2.3 04.12 08.12	2.4 11.12 15.12	2.5 18.12 22.12	Holiday 25.12 29.12	Holiday 01.01 05.01	2.6 08.01 12.01	2.7 15.01 19.01	2.8 22.01 26.01	2.9 29.01 02.02	2.10 05.02 09.02	
Research & Analyze	Blue																							
Design						Purple	Purple	Purple	Purple	Purple	Purple	Purple												
Realize											Green	Green	Green	Green	Green	Green	Green	Green	Green	Green				
Deliver																			Red	Red	Red	Red	Red	Red

The roadmap above showcases the established course of development split into the 3 main HBO-ICT competences chosen, Analyze, Design and Realization, as well as a delivery phase where documentation, necessary graduation documents and forms to be filled and overall quality assurance proof for the company.

During this period, me and my company supervisor conducted stand-up meetings from Tuesday to Friday from 9:00 to 9:15, and at the end of the week, each Friday from 10:00 to 11:00 a progress meeting to check what has been done, what needs to be completed, and the overview of the goals and roadmap, if everything is on course or there are any problems. The realization part of this project consisted of sprints of 2 weeks, starting on a Monday and ending on a Friday, the sprints kept track of the development progress and through issues, stand-up and progress meetings, with a beginning sprint 0 of 1 week which establishes the grounds for the next sprints such as the code base project (front-end, back-end and mobile), and making the transition from design to product.

Table 6-Sprints time table

Sprint	0	1	2	3	4	5
Dates	13.11 -17.12	20.11 -1.12	4.12 - 15.12	18.12 - 29.12	1.01 – 12.01	15.01-26.01
Preparations						
Laptop fingerprint / Pin authentication						
QR code login						
Biometrics login						
Testing and quality assurance						
Documentation						

Above is a table highlighting the main development points based on the three main requirements of the project.

Drawing from the [“Appendix 7: Backlog and changes”](#) the biggest changes of the project was the initial idea of creating the necessary API endpoints for the WebAuthn API (Windows Hello / Laptop Fingerprint) method of authentication in a Symfony project but due to version compatibility it was no longer possible to achieve this goal and choose to implement these API endpoints in the Node JS Express back-end.

During this assignment, advice on back-end challenges and other backend related topics have been given by the back-end employee Randy Groot Roessink, and UI/UX design with the feedback from the company UI/UX designer, Jericho Thijssen, which both guided me into the right though direction without giving a straight answer, which benefited in the end results of the project, these advices can be better visualized in the [“Appendix 7: Backlog and changes”](#).

For this assignment I managed my code in a company provided repository supported by JetBrains Space environment which is similar to GitHub or GitLab, as it is also structured in issues and sprints, each update and commit to the repository was named after the issue that I was working on having a reference or each issue to the corresponding commit. This way it improved the management of the backlog items that I completed or changed, enhancing my productivity and code structure. The repository for the system parts was split into three sub folders, each holding the code for the front-end, back-end and mobile application. This structure allowing me to better coordinate and manage the product during the development. The coding of the project is following the default coding standards of JavaScript for the front-end and back-end with the additional Vue and Quasar coding and structuring standards to fit the company already written code, the same for the mobile application according to the framework and language standards. More detailed backlog and issues information can be viewed in the [“Appendix 12: Backlog and issues”](#).

4 Quality assurance

For this project I established five fields of testing proving that the code quality, functionality of the applications and system are working according to the goals set.

As of now testing is still in progress and is expected to be finished by the end of the final submission of the report.

4.1 Unit testing

Unit testing will provide the grade quality of the written code of the system, testing the main functionality under different circumstances and scenarios, and making sure that there are no potential security issues or bugs, by isolating the system into distinct parts.

4.2 Regression testing

Regression testing is a testing technique that involves retesting a software application to ensure that new changes or modifications after bug fixes or updates have not affected the existing functionality of the applications and system overall, the main goal of the regression testing is to identify unintended changes and side effects as a result of the project being updated.

Regression testing will be done through manual tasks listed below, each task will have the goal of ensuring that the main requirements and functionality of the system are still fully operational, through regression testing the quality of the system will be provided as if every test is successful without any remarks, it will indicate that the quality of the system is high and tested.

These tests can be further investigated on "[Appendix 9: Regression tests](#)" which highlight the name of the test, description and steps needed to be taken to successfully complete the test.

4.3 Widget testing

Widget testing will be done on the mobile application to test the individual components of the application such as buttons and functions.

4.4 Linting

To provide proof that the mobile application code quality is good and up to the standards I am using the command-line tool provide by the Flutter SDK that performs static code analysis on the Flutter project "flutter analyse", with this command it will be easy to identify and report code issues through errors, warnings, suggestions and style inconsistencies to confirm that the written code is in conformity with the Flutter Dart code standards.

Based on the "[Appendix 8: Linting criteria for Flutter](#)" it can be proven that using this method of code quality testing will ensure that the code base of the Flutter project covers and meets all the coding standards of the framework, allowing for a more readable, performance increasement and consistency.

4.5 Android and iOS testing.

To test and ensure that the hybrid mobile application works according to the project goals, a series of test on various emulators and physical devices will be conducted. iOS will be tested on a MacBook provided by the company with iPhone emulators, the Android testing will be done on a few Android Emulators provided by Android Studio which simulates a real device, these tests will cover various operating systems versions and sizes to verify the UI/UX responsiveness and speed, as well as on a Samsung S21+ physical device that will handle the testing for the biometrics checks feature.

5 Requirements

The lists below are the functional and non-functional requirements established for the project.

- RQ – Requirement
- F – Functional
- NF- Non-functional
- B – Business
- U – User
- S - System

5.1 Functional requirements

Table 7-Functional requirements

Number	Description	MoSCoW
RQ-F1	The login component design must be approved by the UI/UX designer.	Must
RQ-F2	The user must be able to login with the username and password.	Must
RQ-F3	The login component must log in the user.	Must
RQ-F4	The login component must offer multiple ways to log in.	Must
RQ-F5	The mobile application must be accessible for both IOS and Android	Must
RQ-F6	The mobile application should provide notification and sensational feedback on events through the phone's sensors.	Should
RQ-F6	The mobile application design must be approved by the UI/UX designer.	Must
RQ-F7	The login component must offer an option to select the preferred authentication method	Must
RQ-F8	The login component must offer the option to use the laptop fingerprint/touch id or Windows hello pin code as authentication option.	Must
RQ-F9	The login component must offer an option to register multiple devices under the same account.	Must
RQ-F10	The mobile application could provide 2FA codes	Could

5.2 Non-Functional requirements

Table 8-Non-Functional requirements

Number	Description	MoSCoW
RQ-NF-B1	The mobile application must comply to the guidelines and requirements of the app stores.	Must
RQ-NF-U1	The user must be able to complete the authentication checks in maximum three steps.	Must
RQ-NF-U2	The user must be able to select a preferred method to authenticate	Must
RQ-NF-U3	The user must be able to change his preferred method of authentication.	Must
RQ-NF-S1	The UI/UX must be responsive for various devices.	Must
RQ-NF-S2	The mobile application must support multiple languages and must include support English and Dutch.	Must
RQ-NF-S3	The data transmission must be encrypted and secured over HTTPS	Must
RQ-NF-S4	The data stored must be encrypted.	Must
RQ-NF-S5	The login component must require authentication methods based on the user role.	Must
RQ-NF-S6	The login component must support the option to log in using the laptop touch id or fingerprint sensor.	Must
RQ-NF-S7	The mobile application must provide code input for login.	Must
RQ-NF-S8	The backend must handle encryption and decryption of data.	Must
RQ-NF-S9	The database must store encrypted data	Must
RQ-NF-S10	The mobile application must support code authentication	Must
RQ-NF-S11	The mobile application must support biometrics authentication	Must

RQ-NF-S12	The login component must support using the laptop Windows Hello or Touch id/Pin code.	Must
RQ-NF-S13	The login component must have a fallback option in case the new backend methods are not functioning.	Must
RQ-NF-S14	The mobile application must support QR code login	Must
RQ-NF-S15	The mobile application must inform the user if the device can support or not any of the methods of authentication (QR code login or Biometrics check).	Must
RQ-NF-S16	The back end must send email code verification	Must
RQ-NF-S17	The WebSocket must handle connection interruptions and reset the registration session of the user.	Must
RQ-NF-S18	The login component could offer the registration of the account to multiple mobile applications.	Could
RQ-NF-S19	The back end could handle multiple linked mobile application to the same account	Could
RQ-NF-S20	The mobile application could support 2FA code generation	Could

In the table above it can be viewed the functional and non-functional requirements of established for this project, the requirements marked in **GREEN** are completed requirements, the **YELLOW** are still in progress as of the writing of this report, and the **RED** requirements have not been yet completed and can be moved to recommendations for future development.

6 Product

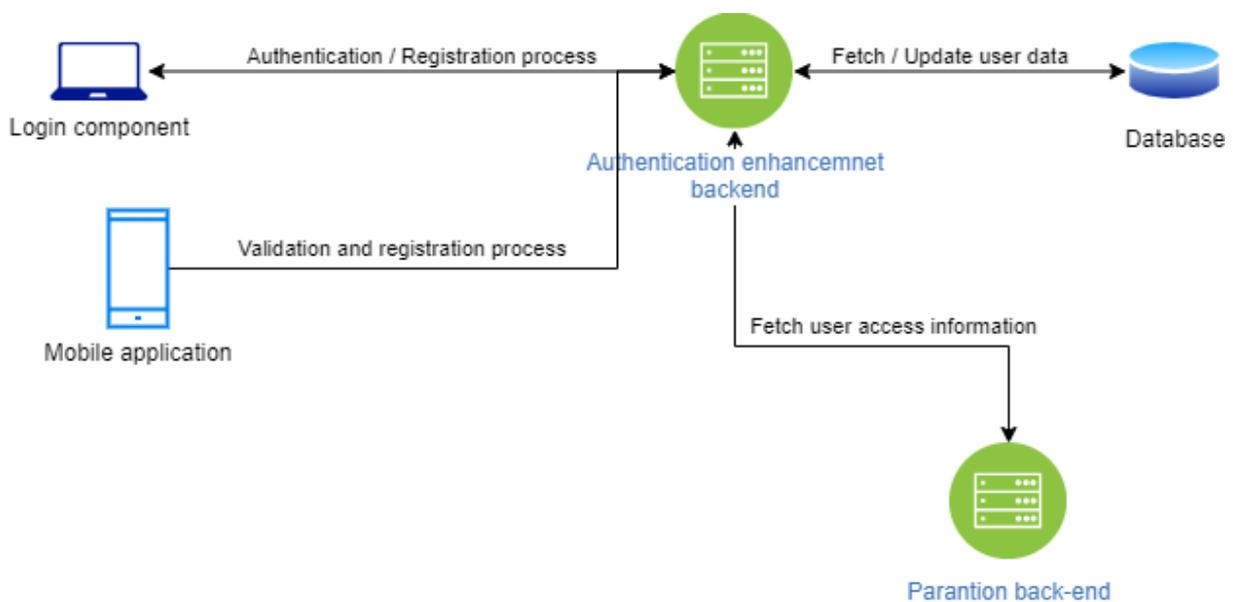
In this section I am going to detail the process of how the product has been build, decision, design evolution, and changes along the development.

6.1 Design

For this project 1/3 of the product is categorized as brownfield as the login component is already a working and fully functional part of the company product and my assignment is to enhance it and add improvements and new features along with 2/3 of the project categorized as greenfield which are the newly backend implementation for the to be developed features and a hybrid mobile application which the company currently does not have one.

This allowing me to offer the company through this project the base of a mobile application in Flutter that can be further used in other projects or plans of the company

Figure 2-System components diagram



In the diagram above it can be viewed an overview of the system components and architecture.

The Login component (Front-end) will make the authentication and registration requests, as the back end receives the data it stores it into the database to later be used upon the authentication process to fetch the user access information from the company back-end API. The mobile application has the role of validating the access requests, and the backend will grand de access to the user based on the received data from the mobile application and stored data in the database.

The UI design of the mobile application and login component changes would fit the newly rebranding of the company to the specific colours and fonts.

6.1.1 Front-end

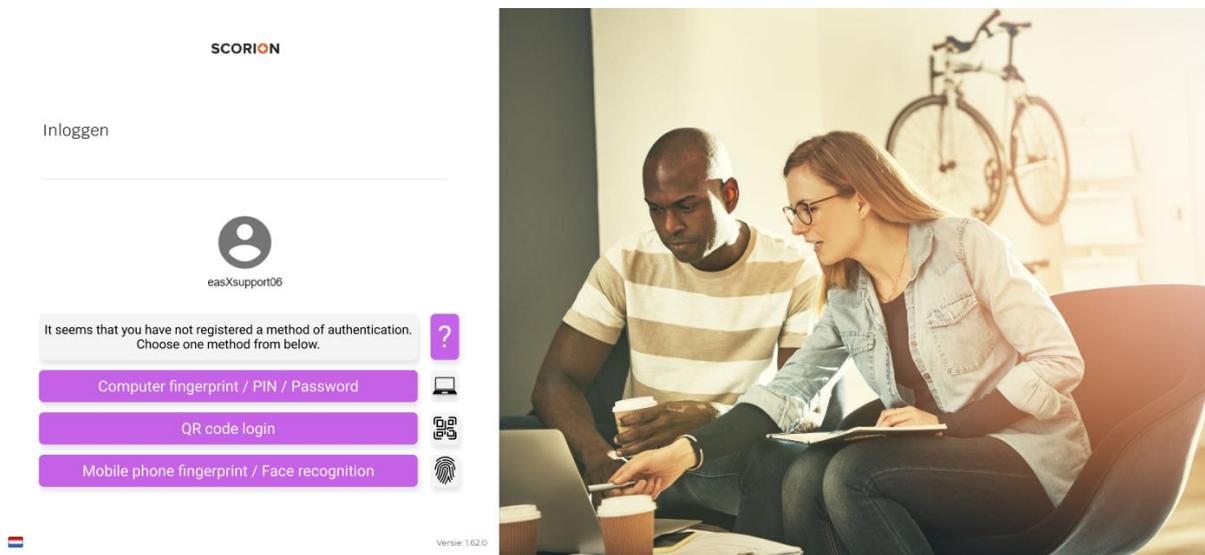
To clarify about the brownfield part of the project, the login component is a component written in Vue 3.0.0, and servers as the interface where the users can login into the company products with their username and password.

For this project I was given the latest version of the login component to work on and have my separate repository where I can modify it and test it to prevent any override of the current live environment of this component until the project is proven to be successful and fully tested.

The login component first modifications have been when the development of the first method of authentication has been done which is the Windows Hello / Fingerprint usage to allow the user to password less authenticate.

These modifications have been decided through some design improvements as to offer the user to initially select which method of authentication would like to register.

Figure 3-Front-end design



In the picture above you can view the design changes of the login component after the user logged in, but it does not have a registered password less method of authentication.

The three methods of authentication have been chosen based on the requirements of the company and the results from the conducted research.

6.1.2 Back-end

In the beginning of development of this project the back-end application would serve as additional endpoints for verification, these endpoints were initially decided to be built in a simple Symfony project, Symfony being the company's main backend framework and by integrating the necessary changes into a compatible back-end project as the main backend framework it would ease the transition of integrating the features and changes for production.

Unfortunately in the beginning of the development, in sprint 1, this decision has been revoked due to version compatibility, the company Symfony version is not compatible and does not support the latest verifications and validations of the Windows hello / Fingerprint method as it based on FIDO validations and configurations, FIDO being the core library that supports the validation of authentication and registration through Windows hello, which I will dive into details later in this paper.

This change made the movement to the second option of back-end framework which is NodeJS Express, this framework is very familiar and commonly used for small tasks around the company called services and due to the complexity of this project we decided that it would be the best option to keep it as a service backend.

The back-end features the main API endpoints for registration, authentication and validation of the users credentials through for the Web Authentication API (FIDO validation), and a because of the analysis and conclusions drawn from the "Backend communication protocols and frameworks" research, a WebSocket service is used to handle the communication and validations of the users sessions when authenticating with the mobile application, as it provides live communication, secure channels and unique session identifiers.

The back end features an email verification function, which allows for a simple Gmail account used for the purpose of increasing the security and have a more valid proof of the identity of the user during the registration process.

The email verification was thought to be implemented after a security risk was discovered during the registration process which was if two persons are doing the same registration process under the same

username and database the back end will have to know which active registration session needs to save the registered device data.

A possible solution for this problem was to allow only one session per username and password to be active and if another person tries to create a similar session it will automatically be closed, this idea was not implemented as it can fall in the unlikely scenario of what if the 2nd user that tries to do the registration process is the actual owner.

6.1.3 Mobile application

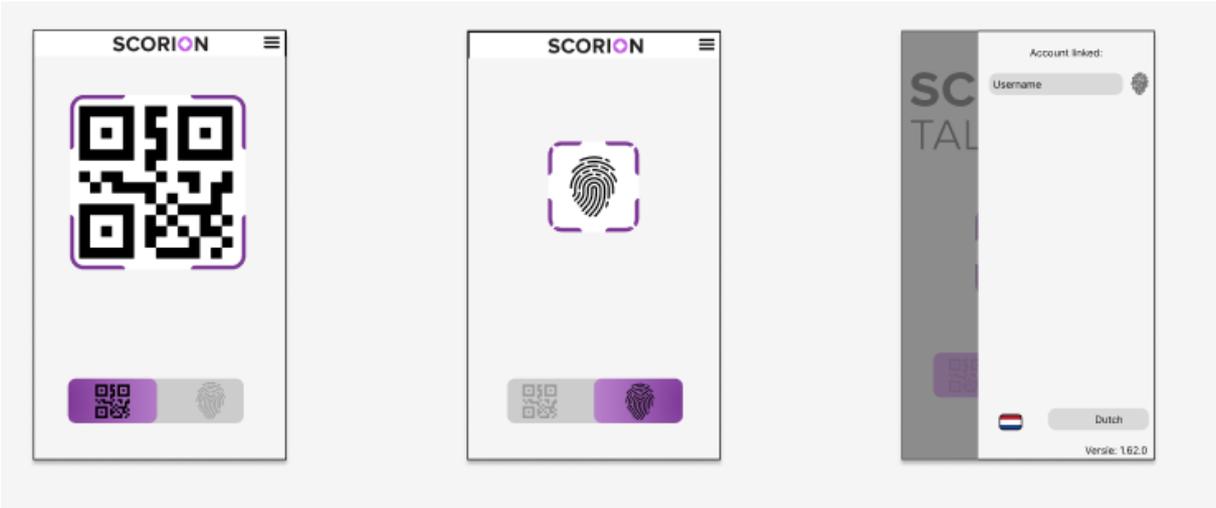
The mobile application has been built in a hybrid development framework, Flutter, as based on the research conducted in the beginning of the project proved to be the most efficient, fast, error-proof and as close as possible to a native application, serving the access to numerous native features that can be used in the future.

After the research phase has ended and the development phase was approaching, a decision was made between 2 paths of development frameworks for the mobile application, one of them being the usage of Progressive Web Application (PWA) which offers the users the ability to add to their home page of the phone the application to use for the authentication and registration processes without any installation processes or the need of an app store to host the application which sounded great as it turned down cost of production, hosting and it would have been more time efficient, as a PWA is in essence a Web Application that can access some of the mobile phone native features. And the other path was the usage of a secure hybrid framework.

The reasons the PWA was not chosen was due to security risks and limitations of native features, as it is impossible to store sensitive data on the user device, because being a JavaScript based application it does not have access to the phone internal storage where data can be safely secured, compared to the hybrid frameworks where they offer this possibility each of them with a grade of safety, Flutter, scoring the best in this field.

The design of the mobile application has gone through some changes reflecting the feedback of the company UI/UX designer, as I was trying to make the application UI look and feel according to the company theme and new UI colours and shapes as currently is undergoing a branding transition, and would greatly benefit if the mobile application is keeping up with these updates.

Figure 4-Mobile app initial design



The initial design would feature a slider to choose between the two different method of authentication, QR scan login and biometrics check, along with a drawer to view the current registered account, and a possibility to change the language.

I chose this design of the drawer to have room for a feature of this system to handle multiple accounts with the same device and be showcased as a list of accounts.

Initially the camera for QR code scanning would occupy the space of the square as seen in the picture above, but later changed to a larger view occupying two-thirds of the screen leaving room only for the slider and top bar.

During the development of the 3rd method of authentication, the biometrics check, I discovered a security issue that compared with the QR scan login method, after the validation of the biometrics it is impossible to know which session to approve the login as there might be the very low change of 2 people trying to login under the same username at the same time.

And initially implement a chose between 3 numbers to validate which session is the actual owner of the acount.

Figure 5-Numbers choice validation.

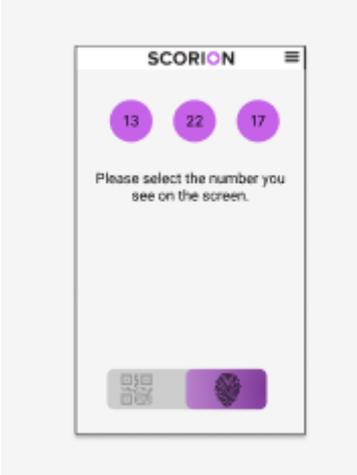
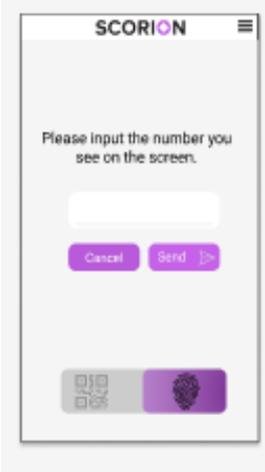


Figure 6-Number input validation



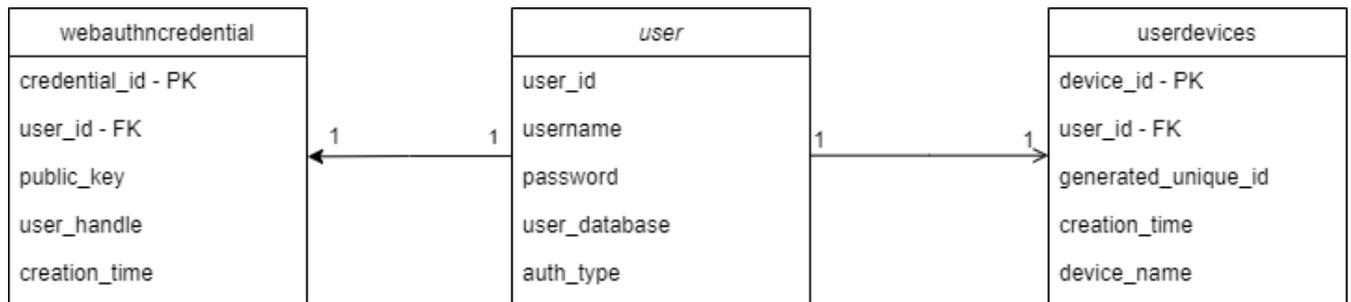
But because there might be the very unlikely scenario of having more than three active sessions under the same username I implemented to check if there are more than one session active to force the user to input the displayed number.

The logic and reasons of implementation would be discussed in the next section of this report.

6.1.4 Database

The database choice of this project was the most efficient and straight forward decision as it is the same database framework, MariaDB, as the main company database framework, being a very popular choice among the developers being open sources, featuring great security features and being high available and scalable, being an SQL based relational database management system, is a standard language to use for managing and manipulating relational databases, making it compatible to a wide range of applications and tools.

Figure 7-Database diagram



As it can be viewed in the diagram above, for this project I store the user credentials (username, password and database) to use when authenticating using the company API to fetch an access token and the necessary information for the user to login into the products. Each user can have a registered device which holds a unique device id a server generated unique id and the device name. Each user WebAuthn credentials are stored in the “webauthncredential” table which contains the public key of the user credential_id and handle to be used in the authentication process for validation.

6.2 Implementation

In this section I am going to dive into the details of implementation of each of the section of the project.

During the development of this project, I provided the company with good results during the design and research phases of the project, that I had the opportunity to allow the further development of the WebAuthn method of authentication using the laptop fingerprint sensor through Windows Hello, the company upgraded my working laptop to a newer model featuring a fingerprint sensor, this upgrade allowed me to increase my development speed and testing of the newly implemented features.

6.2.1 Method of authentication

For this project I implemented three methods of authentication without the necessity of a password. To enhance the user experience and a allow a more secure and fast login process.

6.2.1.1 Windows Hello / Fingerprint login

During the setup of this project one of the established features was to implement the WebAuthn process into a backend developed in the same framework as the main company backend, Symfony, but due to version compatibility of the library for which registered and validates the authentication process, would no longer be possible to fully implement this method into a Symfony framework backend.

The reason for this drop of implementation was due to several reasons such as older versions of this library no longer being supported which will create compatibility risks in case of updates to the backend framework in the future as well as provide instability and bugs which can be detrimental to the project, as a new version of this framework might cause dependencies issues leading to a complex cascade of changes, avoiding this can simplify project management and reduce the risk of introducing new issues. One of the goals of this project being to offer the company an easy to implement the desired functionality.

Due to this reason and the lack of proper documentation and support for the older versions, the learning curve turned to be difficult which is another reason to support this decision.

WebAuthn defines an API enabling the creation and use of strong, attested, scoped, public key-based credentials by web applications, for the purpose of strongly authenticating users.

WebAuthn, or Web Authentication, is a web standard published by the World Wide Web Consortium (W3C) in alliance with the FIDO Alliance. It represents a significant leap forward in online security, offering a more secure and convenient alternative to traditional password-based authentication methods. Here is a brief introduction to how WebAuthn works and its key component.

6.2.1.1.1 The registration process.

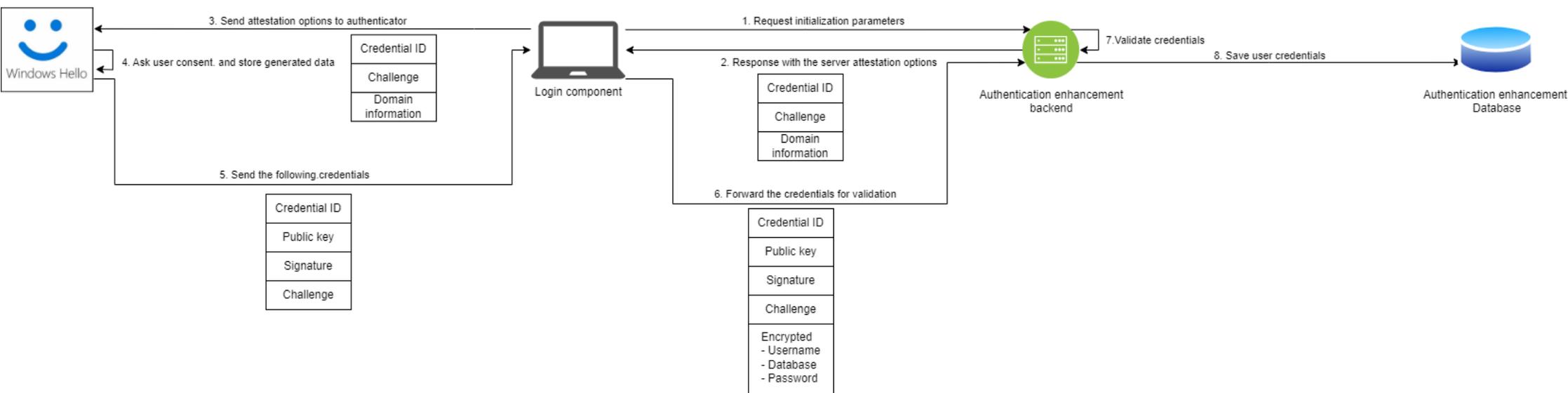
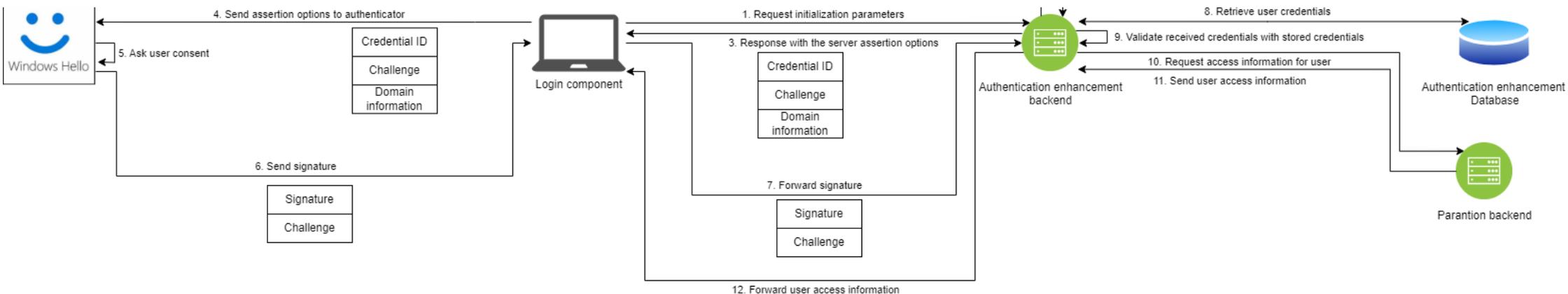


Figure 8-WebAuthn registration flow

After the user has successfully submitted his username, a registration check would be made and if the user is not registered with a method of authentication, the password would be asked from the user and if the password is valid, it would be forwarded to a menu to select a method of authentication. The diagram above describes the process of registration using the WebAuthn method.

1. The user requests the initialization parameters from the backend.
2. The backend response with the attestation options for the FIDO verification later + a generated unique Challenge and credential ID,
3. The login component requests the authenticator, in this case the Windows Hello authenticator to generate a unique public and private key pair.
4. The authenticator asks the user to make and identity check (PIN, Biometrics)
5. The authenticator responds to the login component with the generated public key, signature and generated challenge and credential id from the server.
6. The login component forwards the received credentials and requests the backend to register these credentials with the user credentials (username, database and password).
7. The backend validates the received credentials from the authenticator.
8. The backend saves the received credentials to the database to later be user in the authentication process.

Figure 9-WebAuthn authentication flow



6.2.1.1.2 The authentication process.

After the user has successfully submitted his username, a registration check would be made and if the user is registered with a method of authentication and be forwarded to the method of authentication registered. The diagram above describes the process of authentication using the WebAuthn method.

1. The login component requests server initialization parameters.
2. The server retrieves the stored credential id of the user.
3. The server responds with the credential id of the username, a generated challenge and domain information.
4. The login component requests the authenticator to validate the received domain information and credential id.
5. The authenticator validates the received data and asks for user consent.
6. Send the generated signature and challenge.
7. Forward the signature and challenge to the backend.
8. Retrieve the user credentials.
9. Validates the received credentials with the stored ones and compares them.
10. Request the access token and information with username and password from the company backend.
11. Receives the user data.
12. Forwards the user access data to the login component.

6.2.1.1.3 Key elements

6.2.1.1.3.1 Fido2

The validation of this process is done using the FIDO2 library (fido2-lib), which is a set of technology standards, including WebAuthn, developed by the FIDO Alliance. It is designed to provide simpler and stronger authentication experiences.

6.2.1.1.3.2 Challenge

The challenge in this system is a random string generated by the server during both registration and authentication phases. The purpose of this challenge is to prevent replay attacks and ensure the authenticity and freshness of the authentication session. This ensures that intercepted data cannot be reused as the challenge will be different in the next session.

6.2.1.1.3.3 Credential ID

The Credential ID is a unique identifier associated with a specific set of cryptographic keys (a public and private key pair) used for authentication. During the registration process the credential ID is associated with the public-private key pair generated by the user authenticator and during the authentication process it is used to identify which key pair should be used to sign the challenge from the server to be validated with the stored credentials.

6.2.1.1.3.4 Initialization parameters

The initialization parameters are certain domain information that represents the server using the parameters:

- Relying party id: Is a string that represents the domain name of the relying party (the server chosen name), It is used to ensure that the authentication request is indeed intended for the specific website or service the user is trying to access. This helps in preventing phishing and man-in-the-middle attacks.
- Origin: includes the scheme ("https") and the host to help the browser verify that the authentication request is coming from a legitimate source.
- Challenge: random generated string or byte sequence from the relying party (server) to ensure that each authentication and registration session is unique.

6.2.1.1.3.5 Public key / Private key

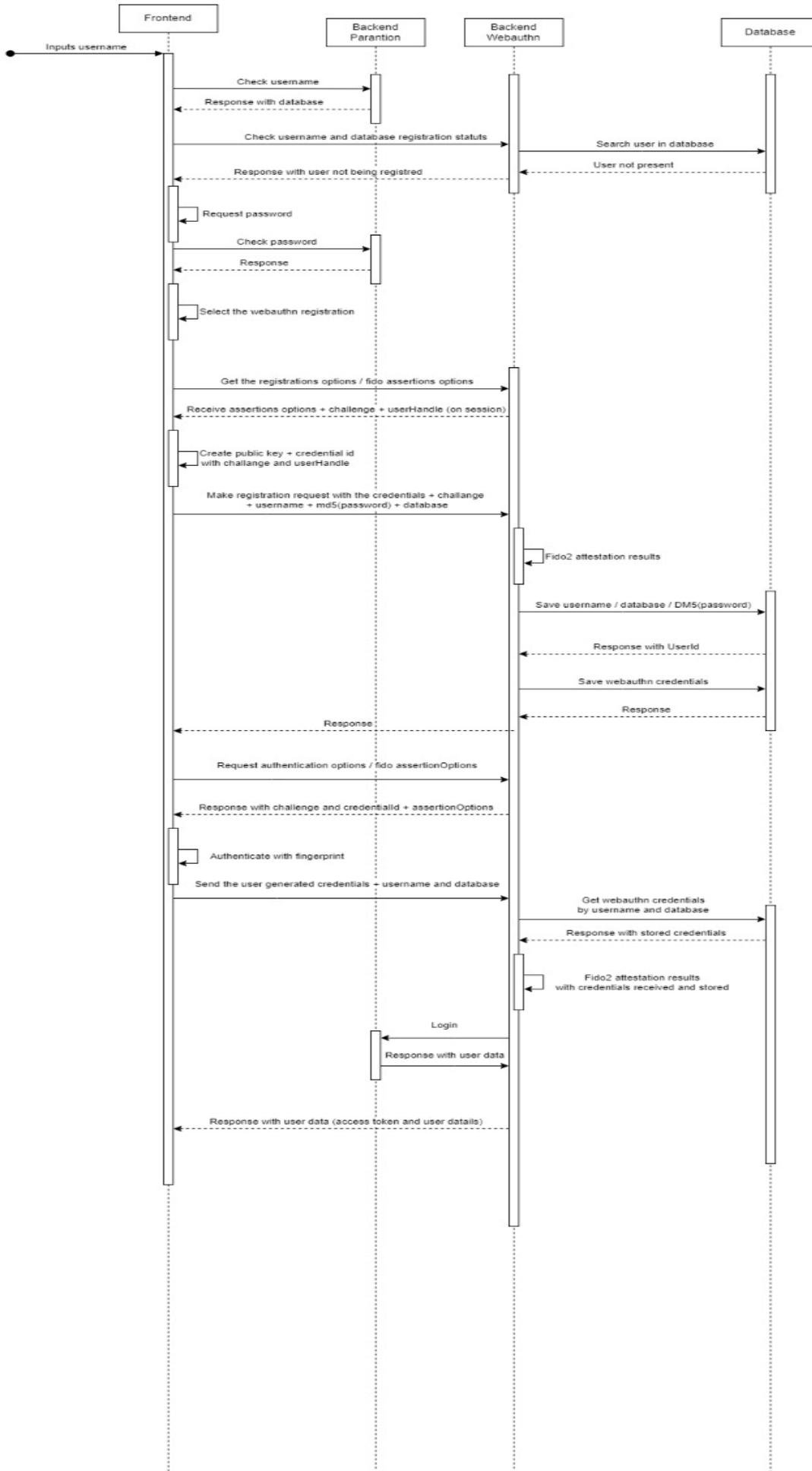
The public key is generated upon registration by the authenticator which generates a public-private key pair, the private key is securely stored on the user's device and not shared, and the public key is sent to the server to be stored.

The server uses the public key to validate the signed credentials with the challenge received. The public key can only verify the signatures made by the corresponding private key which is highly secured against external attacks.

6.2.1.1.3.6 Signature

The signature is the data signed by the private key along with the challenge to send for validation.

Figure 10-WebAuthn full flow



In the diagram on the left a detailed view of the process of registration and then authentication of the user using the Windows Hello / Fingerprint authentication.

As it can be viewed the front-end checks initially with the Parantion backend and the Authentication Enhancement backend (Backend Webauthn in the diagram) if the user credentials are correct and if they are registered with the method by checking with the database.

The user does the registration process discussed above and later validates it for a successful authentication.

6.2.1.2 QR code login

The QR code login works with the support of the mobile application, Authenticator, the mobile application allows the user to register his device details to the specific account upon registration and store a unique identifier generated by the server during the registration process.

The stored unique identifier along with the device unique identifier and name would be then used for the validation during the authentication process. The QR code login registration process works by checking initially if the username and database of that username have a registered device, if not the registration process will begin through an open WebSocket.

The front-end will receive from the backend a unique identifier of the connection, a server generated unique identifier, and the username and database of the account to be displayed in a QR code to be scanned. The Authenticator will be used to scan the QR code and if there is no already stored account on the device it will store the server generated unique identifier scanned from the QR code and the username and database. The unique identifier of the device and the device name is fetched using the library "[client information](#)" which provides a unique and persistent unique identifier of the device.

These identifiers are then sent through an API call to the backend and be validated and allow the session where the QR code has been scanned to move to the next phase which is the validation through an email confirmation.

This step is an extra security measure as there might be a small change of 2 or multiple users trying to register a device to the same account, by implementing a code received through the user's email address it will ensure that the session that successfully scanned the QR code is the actual owner of the account.

This service has been realized using the "[nodemailer](#)" package that allows a service to send an email using an already existing email address in this case a Gmail address, that will send custom emails. I choose this method as the mailing service already constructed by the company is at the moment inaccessible unless you are a high ranking employee, and due to the nature of this project being a proof that multiple method of authentication can be implemented and allow for a faster pace and success rate I was encouraged to choose this method rather than waiting for approval and spend time on a relatively small part of the system which can be later replaced with the existing emailing service.

The user will view a code format XX-XXXX with the first XX- being revealed to correctly input the rest of the code received by email.

After the code is validated, the user is registered and later can authenticate using only a QR code scan.

The authentication process is less complicated as upon the input of the username, the front-end will check with the backend if the user has a registered method of authentication, it will then create an open WebSocket connection and receive to be displayed on a QR code, a unique identifier for the connection.

The Authenticator will then scan and check if there is a stored account on the device and send to the backend for validation the stored data and session id from the QR code.

The backend will validate the data and make an API call with the stored user credentials in the database to the main company backend which will provide the access information of the user to be forwarded to.

6.2.1.3 Biometrics login

The biometrics login works through the same WebSocket system as the QR code login detailed in previous section of the report, the only key difference is that it relies on a randomly generated code that needs to be input into the Authenticator to validate the session upon the registration of the device, followed by the email validation with the same XX-XXXXX code sequence.

The authentication process proven to be the most difficult part of the process as compared with the QR code login where a session unique identifier can be passed through the scanning process, a security problem raised from the biometrics login as in case of multiple session under the same username and database combination, there is no way to correctly identify only through already registered unique identifiers.

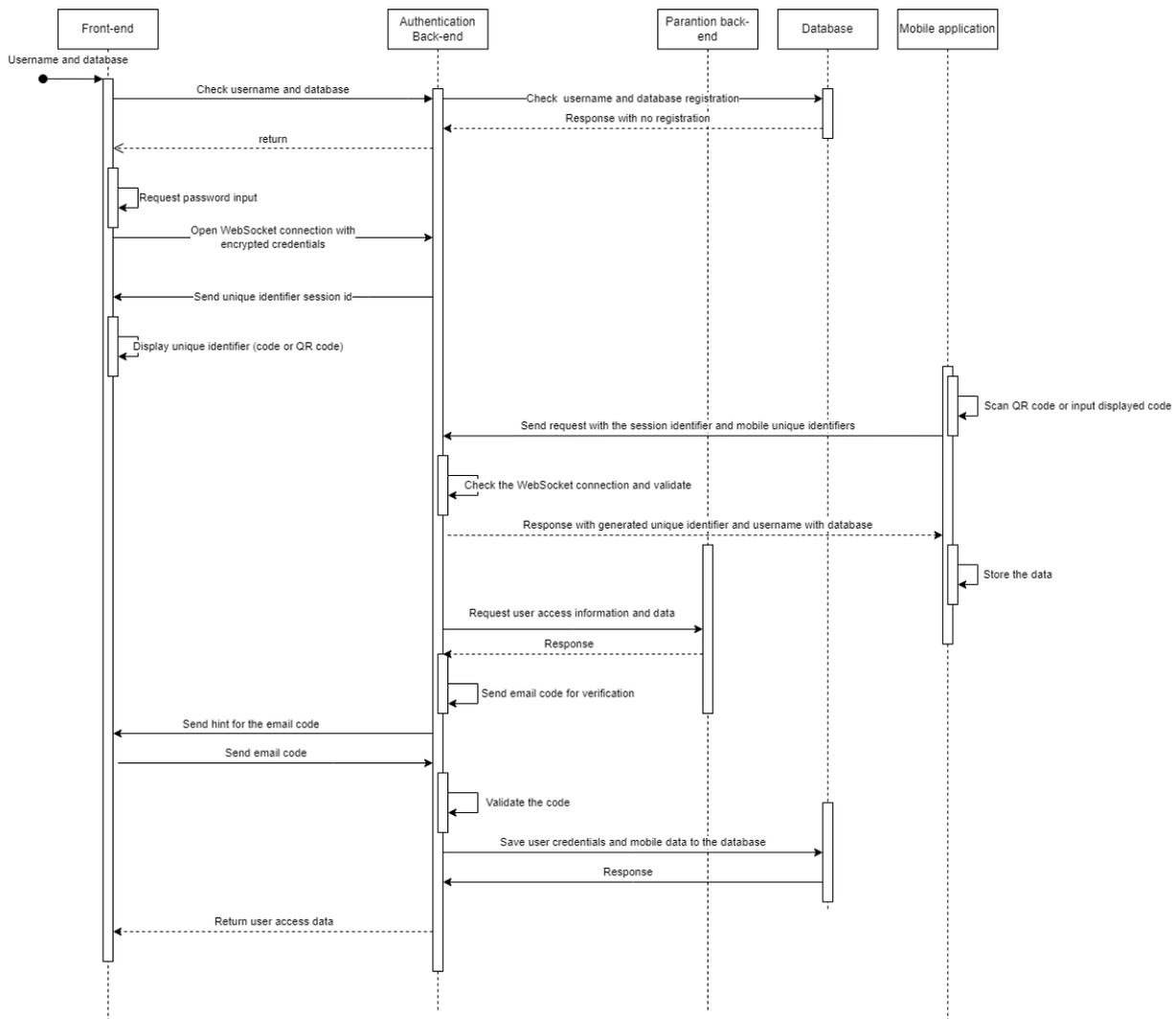
To counter this problem the initial solution was to hold out any other session creations under the same username and database combination, then it will fall into the scenario of what if the 2nd or 3rd session is the actual owner of the account.

To counter this I came up with the most reliable solution to this dilemma which allows the system to correctly identify the session by assigning each session a random 2-digit number that is displayed on the front-end to be input into the mobile application after a successful biometrics check.

This problem is a very unlikely scenario that can happen in this system and the user will not have to use this input feature most of the time.

The biometrics check is done using the mobile phone native fingerprint or face recognition sensors, through a library called "[local_auth](#)" which allows a simple integration and usage of the biometrics pop-up check of the phone.

Figure 11-Mobile application registration process



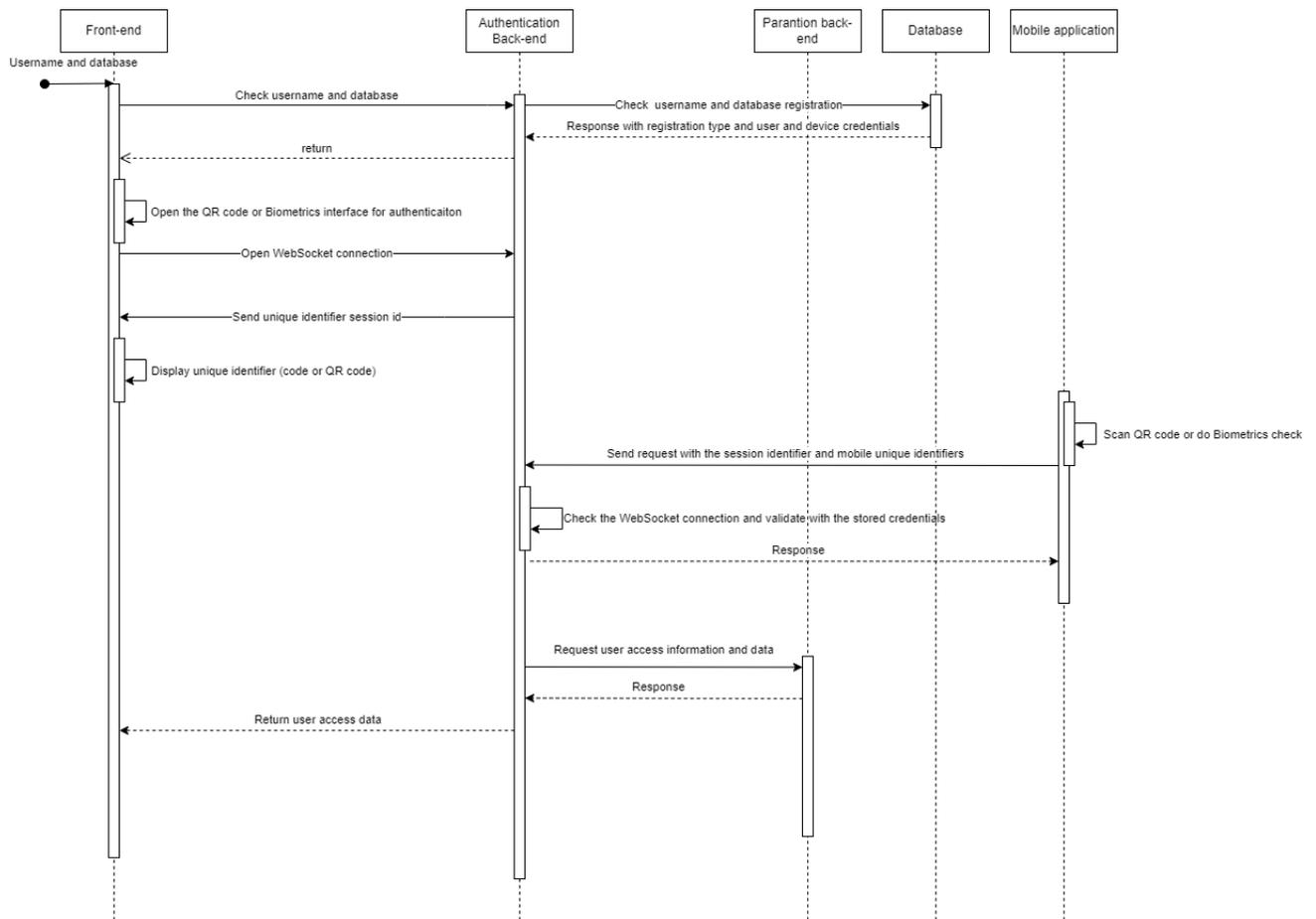
6.2.1.4 Registration process

Above is the sequence diagram of how each component of the project communicates in order to achieve the successful registration of the user using the mobile application QR code scanning or Biometrics check functionality.

1. The user will input his username and database (if registered in multiple databases)
2. The front-end starts by checking if the user and database combination are registered into the system.
3. The back end will search through the database for this combination.
4. Response with no registration found.
5. Request input of the password from the user and choose one of the methods that require the mobile application. (QR code login or Biometrics).
6. The front-end opens a WebSocket communication session with the back end with the encrypted username, database and password.
7. The back end will respond with a unique identifier of the session.
 - a. For the QR code login process it will only send the unique generated server identifier and WebSocket session id.
 - b. For the Biometrics login process it will send a generated numerical code in the format XXXX-XXXX to be displayed on the screen to be input on the mobile application. This code acts as the unique identifier of the session.
8. The mobile application will then be used to scan or input the unique identifier from the login component.

- a. For the QR code login process it will scan the displayed QR code and extract the unique session identifier.
 - b. For the Biometrics login process a code input will be requested and then a biometrics verification.
9. The mobile application will send a request to the back end with the unique identifier fetched and mobile unique identifiers (id and name).
10. The back end will check if the received data matches with a WebSocket connection.
11. The back end will respond to the mobile application with the username database and unique identifier generated to be saved on the device for later use.
12. The back end will request from the Parantion back-end the access information and user data.
13. An email will be sent to the user with a numerical code in the format XX-XXXXX
14. The back end will send to the login component the hint for the code and request an input.
15. The user will input the code and the login component will send that code to the back end.
16. The back end will validate the code and save the credentials of both the user and the device to be registered into the database.
17. The back end will forward the access information of the user to the front-end.

Figure 12-Mobile application authentication process



6.2.1.5 Authentication process

In the sequence diagram above it can be viewed the authentication flow for the QR code and biometrics authentication.

1. The user inputs the username and database (if registered in multiple databases)
2. The back end will check for the registration of the username and database combination.
3. The response with the type of registration is forwarded to the login component.
4. The login component will display the type of authentication required based on the registered response and display the QR code or input code to the user.
 - a. In the case of biometrics authentication if multiple sessions under the same username and database are active, a unique 2-digit number will be displayed on the front-end to be used for extra check in the mobile application.
5. The mobile application will scan the QR code and send the stored data during the registration process and the unique device id and name to the back end along with the session identifier extracted from the QR code.
 - a. In the case of biometrics authentication, it will make a request to the back end with the number of active connections under the username and database combination and if multiple connections are active, it will display a code input to the user. If there is only one connection active it will automatically ask for biometrics authentication and send the stored data to the back end.
6. The back end will validate the received data with the stored data.
7. The back end will make an access request with the user credentials to the Parantion back-end and forward the data to the front-end.

During these processes of registration and authentication multiple error checks are implemented during each step of the process, if one of them fails the session will either be terminated or inform the user of the error.

Fallback methods have been implemented as if specific server-side errors are triggered, for example if the authentication enhancement back-end or database is not active it will automatically revert the user to only username and password authentication to not keep the user from using the product.

6.2.2 Back-end

The back-end integration is a Node JS Express service that handles the FIDO verification and validation through API endpoints and a WebSocket service that allows for communication during the QR code and Biometrics authentication methods.

Initially the backend was to be constructed with the same framework and version as the main company backend, Symfony 5.4, but due to compatibility issues of the FIDO library as it has not received updates to meet the necessary requirements for a proper validation it has been decided to not go further and move to a simple Node JS Express service.

The back-end implementation for the WebAuthn API works in a 2-step process for authentication and registration.

For the registration the user makes a get request with the username and database to get the server registration-options, during this call the server checks if the received request data is already registered, if not it can proceed in creating a credential ID and challenge with the server preset preferences and details.

After the front-end makes the registration, another request is sent to the server with created credentials to be stored into the database after the validation using the FIDO verification of the signed challenge.

The authentication process works the same way as the registration process as initially it requests the registered credential id from the server and after the verification of Windows Hello or Mac OS Touch ID, a signed challenge by the private key of the user is send to the serve to be validated with the stored public key.

After the validation is completed, the back end will make an access request to the company backend with the user credentials (username, password and database) to be forwarded to the front-end.

The WebSocket service of the back end handles the communication and sessions of the users during the authentication process.

During the authentication process if the user is having a registered method of authentication, it will be forwarded to the chosen method.

In the case of the two methods QR code login and Biometrics check using the mobile application, the front-end will open a WebSocket connection with the backend and send the username and database of the user that is trying to do the login.

The back end handles what type of authentication between the 2 should it choose based on the stored data in the database, and what type of validation should it expect.

In the case of QR code authentication, the process is more straightforward as in the QR code data, the unique session of the user is integrated in the form of an unique UUIDv4 ID, I choose the UUIDv4 as it offers a large number of possible unique values (2^{122}), using this identifier the mobile application send a request to the back-end through an API endpoint with the phones unique identifier and stored generated unique identifier that was made during the registration process.

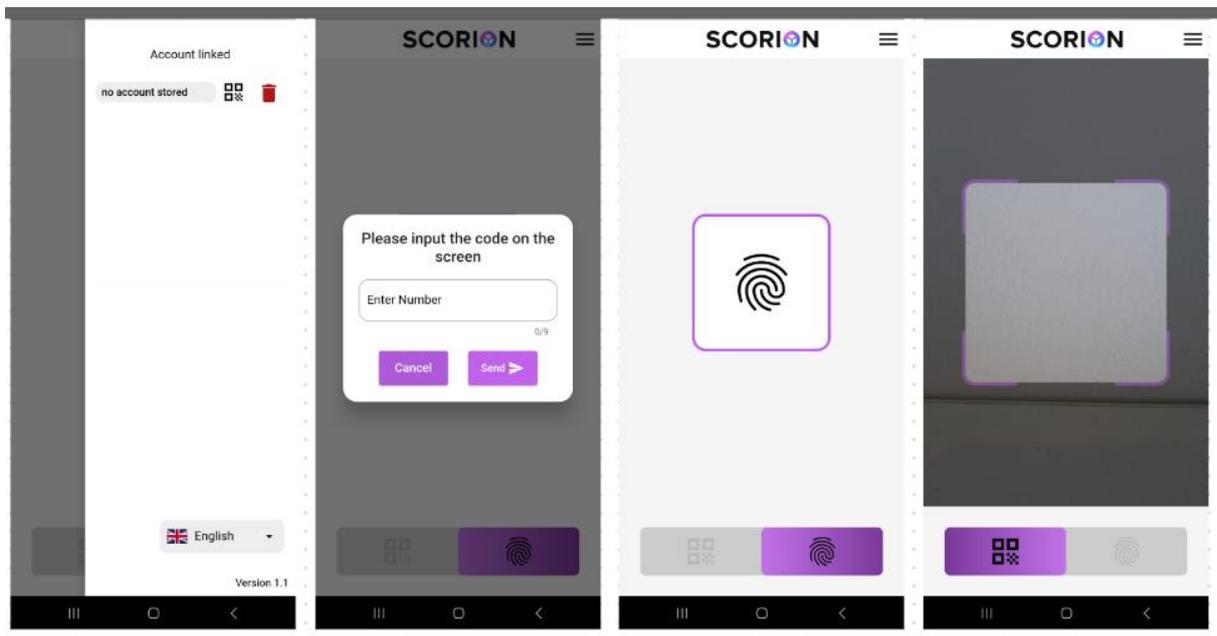
The back-end was constructed in a way that if anytime during the registration or the authentication processes, an error is thrown due to database failure or any other system failure, an error will be send to the frontend to revert to only username and password authentication to not allow the user to be locked out of the system.

6.2.3 Mobile application

The mobile application is a hybrid application developed in Flutter and handles the QR code scanning login and biometrics login, working by sending a unique identifier of the device to be checked and validated to the registered credentials received during the registration process.

One of the challenges of this part of the project was the fetching of the unique phone identifier which in present is no longer possible due to privacy reasons, but through research I discovered a library that can give a unique identifier for the application that the phone will assign to it, this id is persistent and does not present a security issue.

Figure 13-Final mobile application design



In the picture above are the latest UI/UX changes, a screenshot of the Biometrics check interface is not possible due to phone privacy reasons restricting the usage of taking a screenshot. During the development of this assignment, working with the feedback and advice of the UI/UX designer of the company, Jericho Thijseen, which gave the approval of the design choices that I made to achieve this final UI/UX interface. The company provided me with the necessary UI colours, shapes and dimensions to properly integrate them into the mobile design.

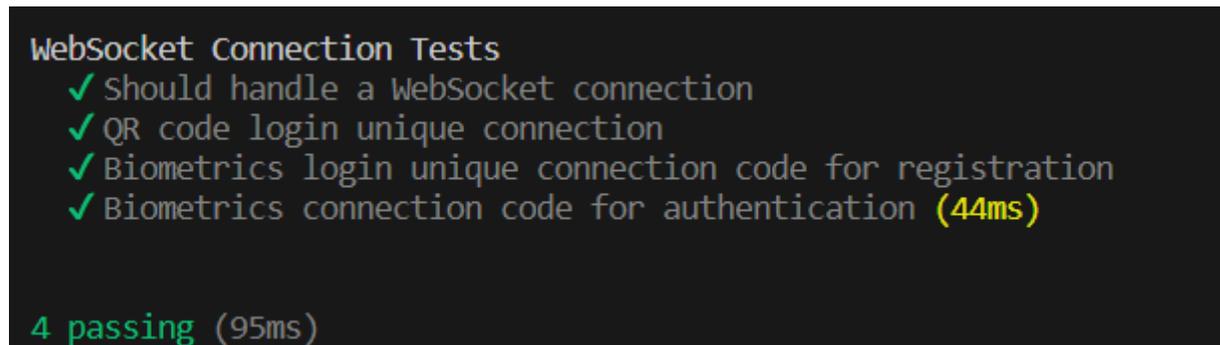
The mobile application development went as planned during the design phase each feature working accordingly and smoothly, code standards have been followed accordingly allowing for a structured and easy to read and understand code base for future development.

7 Testing

As of the submission date, the testing phase is still undergoing and currently in progress, and further testing proof will be offered during the defence of the project.

The current testing elements that have been made are a few UNIT tests on the back-end Web Socket service, these tests are to check and verify the unique session characteristics of each of the 2 method of authentication created that are handled through WebSocket connections, QR code and Biometrics authentication. The tests have been going successfully and proving proof that the back-end can handle multiple session and have unique ids assigned to each of them.

Figure 14-WebSocket sessions testing



```
WebSocket Connection Tests
✓ Should handle a WebSocket connection
✓ QR code login unique connection
✓ Biometrics login unique connection code for registration
✓ Biometrics connection code for authentication (44ms)

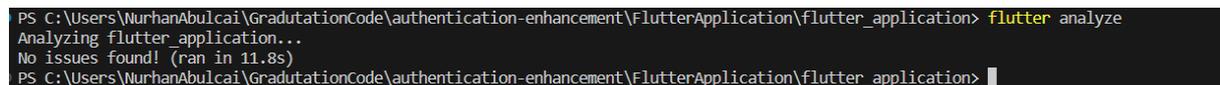
4 passing (95ms)
```

In the figure above, it can be viewed the WebSocket unique sessions check tests, which succeed, which matches with the requirement “RQ-NF-S17” that requires that the back-end must support multiple unique sessions.

Drawing a conclusion from “[Appendix 13: Mobile testing](#)”, the requirements for mobile testing have been met as it handles QR code and Biometrics authentication, due to the nature of the tests, video evidence would be best suited for this proof of testing, which it can be demonstrated during the defence presentation of the assignment. The mobile testing was successful through different emulators and providing proof that the requirements for mobile have been successfully tested.

And according to the “[Appendix 14: Testing methods of authentication and email validation](#)”, the requirements for the front-end testing proofed through a several screenshots, of each of the methods of authentication developed, proofing the testing of the system successfully and the quality of the application as established through the requirements.

To provided quality assurance testing, as to the submission of this assignment, the linting validation of the mobile application through the linting criteria presented at “[Appendix 8: Linting criteria for Flutter](#)”, can be viewed below.



```
PS C:\Users\NurhanAbulcai\GraduationCode\authentication-enhancement\FlutterApplication\flutter_application> flutter analyze
Analyzing flutter application...
No issues found! (ran in 11.8s)
PS C:\Users\NurhanAbulcai\GraduationCode\authentication-enhancement\FlutterApplication\flutter_application> |
```

The testing phase as of writing this report is undergoing successfully and without any issues, the manual tests that can be viewed in the “[Appendix 9: Regression tests](#)” have been completed successfully, another phase of these manual tests repeating in the following weeks with additional cases.

8 Conclusion and recommendation

To conclude this report, the project is a success as it meet all the initial requirements and provided a great viable product to be implemented and used in the production of the products of the company.

During the development of the mobile application, the structure of the mobile application is made in a way for continuous development and additional features to be added.

The resulted research, prototyping and development of the methods of authentication with the support of a hybrid mobile application, this assignment answered the main research questions put in the beginning of the project, through proof and great results, the company will benefit from this assignment and have the opportunity to allow the users to have a better, faster and more secure authentication experience, offering a solution to the problem that Parantion was facing before the start of this assignment.

The product will allow the users and employees of Parantion to access Scorion, the main product developed by the company, without the necessary of password inputs and allow for fast identification and password less authentication of the owner of the accounts.

The product during the graduation period did not suffer massive changes which most of them have been explained during the previous chapters, these changes have not impacted the main goal of the assignment and through risk management and advice, I managed to keep the main goal intact and accomplished.

Some requirements have not been implemented due to time constraints of the project, which allows me to recommend further improvements and recommendations to how this project can move forward and additional features to be implemented.

A recommendation is the addition of multiple registered accounts under the same device, and have the mobile application handle multiple types of registered accounts that can be either QR scan login or biometrics login.

The process during this graduation period has been going smoothly and according to plan, the only issues that encountered were the early deadlines regarding the final report as initially I had planned them 1-2 weeks later, resulting in a delay of proof of testing which is still undergoing as to this submission.

The company employees have been great during this project as I could always call out for advice and help if necessary, during various parts of the project, I had great support from the company management as I was offered the upon showing promising results and progress an upgrade of my laptop to help in the development of the Windows Hello / Fingerprint method of authentication. The laptop received features this technology of fingerprint scanning to unlock the laptop, which was a significant help for the development. As well as IOS testing support, having the opportunity to test the application in an IOS environment through a company MacBook.

I believe the organization fit the assignment and allowed me to grow and explore new territories in terms of software development, risk management, organizational skills and overall complexity of the assignment which was a challenge.

9 Reflection

In my opinion the graduation period went as planned from the start, the research, development and now testing phase have been undergoing accordingly and planned, the results and expectations have been met and I gave all my might and knowledge for the success of this project.

I learned a great deal of software skills in all the fields front-end, back-end, mobile development which made me a more organized and better decision-making person. I encountered problems and dilemmas which helped me grow the success of this project.

During this project there were a few significant changes that led to some challenges along the way, but because of the helpful guidance, advice and support from the company employees that I interacted with and my supervisor, I managed to overcome these challenges without any issues.

One of the important changes was the change of planning of development for the back-end part of the assignment, where initially it was supposed to be developed in Symfony, the company's main back-end framework, but due to computability issues that were explained in the previous chapter, it was no longer a possibility, but as I was expecting an issue similar to this might happen in the future development of the project, a back-up plan was already in the planning, the back-up being the implementation of the functionality from Symfony to a Node JS Express framework, as it was already in development. What I learned from this change and problem, was to always think about the risks that can happen in the development period of a project, always make a good risk management analysis and prepare a few back-up options.

Another important challenge was during the development of the QR code and biometrics check authentication method, as it presented with a dilemma that how can we allow the correct session to successfully register the device details to the account. After discussing this possibility with my supervisor we had the idea of allowing only one session per username and database combination to be active, but this solution created another dilemma which was how can I know if the 1st person is the owner of the account, and what if the 2nd or 3rd person that tries to register the device is locked out and not allowed to do the process as it is already undergoing to another person.

This issue was solved by introducing the email code verification functionality, which allows the back end to send an email to the user's registered email address and if it receives the correct code, it will allow the registration process to be finalized. I learned from this situation to think more ahead and visualize a lot more use cases as there is always an angle that was not properly explored.

Collaborating through advice and feedback with the UI/UX designer of the company helped me grow my professional UI/UX skills as I discovered a new field of knowledge and details that I was previously not aware of the depth of how much there is to be learned in this field. Learning tips and how to improve the design helped me learn how to better structure and create unique and correct designs.

Organizing meetings, presentations and interviews with the employees at Parantion increased my management, communication and professional skills by receiving different points of view from different departments and professions, back-end, front-end, design, allowed me to visualize the project in a different manner and from different points of view during this period.

What I would have done differently during this graduation period is do a broader analysis of the potential risk factors that can happen, a better understanding of the risks in terms of software testing and that a lot of scenarios can happen that initially have not been thought.

10 Bibliography

- Apple. (n.d.). *developer.apple*. (Apple) Retrieved October 3, 2023, from <https://developer.apple.com/documentation/usernotifications>
- Apple. (n.d.). *developer.apple*. Retrieved October 6, 2023, from Apple: https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server/establishing_a_certificate-based_connection_to_apns
- Biørn-Hansen, A., Rieger, C., Grønli, T., Tim, A. M., & Gheorghita, G. (2020, June 9). An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Empirical Software Engineering.*, 25(4), 2997-3040. doi:<https://doi.org/10.1007/s10664-020-09827-6>
- Brainhub. (n.d.). *Brainhub*. Retrieved October 27, 2023, from Brainhub: <https://brainhub.eu/library/progressive-web-apps-advantages-disadvantages>
- capacitor, i. (n.d.). *capacitor*. Retrieved September 27, 2023, from capacitorjs: <https://capacitorjs.com/docs>
- Cyberark Glossary. (n.d.). *cyberark*. Retrieved October 26, 2023, from cyberark: <https://www.cyberark.com/what-is/passwordless-authentication/>
- Dart. (n.d.). *Dart*. Retrieved January 3, 2024, from Dart.dev: <https://dart.dev/tools/linter-rules>
- Enyinna, C. (2023, September 12). *freecodecamp*. Retrieved September 14, 2023, from <https://www.freecodecamp.org/news/communication-design-patterns-for-backend-development/>
- Firebase. (n.d.). *Firebase.google*. (Google) Retrieved October 3, 2023, from <https://firebase.google.com/docs/cloud-messaging>
- Flutter. (n.d.). *Flutter showcase*. Retrieved September 28, 2023, from Flutter: <https://flutter.dev/showcase/toyota>
- Flutter. (n.d.). *Flutter.dev*. Retrieved September 27, 2023, from Flutter: <https://flutter.dev/>
- Github. (n.d.). *Github*. Retrieved October 20, 2023, from Github: <https://github.com/>
- Grigalashvili, E. (n.d.). *Anyforsoft*. Retrieved September 29, 2023, from Anyforsoft: <https://anyforsoft.com/blog/hybrid-app-frameworks-comparison/>
- inVerita. (n.d.). *medium*. Retrieved October 24, 2023, from medium: <https://medium.com/swlh/flutter-vs-native-vs-react-native-examining-performance-31338f081980>
- ionic. (n.d.). *ionic*. Retrieved September 28, 2023, from ionic: <https://ionicframework.com/>
- Kartik, N. (2023, January 16). *Medium*. Retrieved September 27, 2023, from <https://medium.com/@naikofficial56/design-patterns-for-backend-communication-446823dee2c0>
- Lynch, M. (n.d.). *Ionic capacitor*. Retrieved September 26, 2023, from Ionic: <https://ionic.io/blog/capacitor-everything-youve-ever-wanted-to-know>

Magnusson, A. (2023, June 22). *strongdm*. Retrieved October 24, 2023, from strongdm:
<https://www.strongdm.com/blog/passwordless-authentication#:~:text=Passwordless%20authentication%20examples%20can%20be,%2C%20badge%2C%20or%20software%20token>.

OneSignal. (n.d.). *OneSignal*. Retrieved October 6, 2023, from onesignal:
https://onesignal.com/unlock-growth-ga-lp?utm_source=google&utm_medium=cpc&utm_campaign=BOF_Brand_Search_EMEA-A&_bt=652032114925&_bk=onesignal%20push%20notification&_bm=e&_bn=g&_bg=129004913243*&gad_source=1&gclid=CjwKCAiA75itBhA6EiwAkho9e35gi_cU8iYI4D_zl

Pabian, G. (2021, March 28). *levelup.gitconnected*. Retrieved September 20, 2023, from
<https://levelup.gitconnected.com/backend-to-backend-communication-d9fe85234ead>

Pusher. (n.d.). *Pusher*. Retrieved October 6, 2023, from pusher: <https://pusher.com/>

PushWoosh. (n.d.). *PushWoosh*. Retrieved October 6, 2023, from pushwoosh:
<https://www.pushwoosh.com/>

React Native. (n.d.). *React Native*. Retrieved September 29, 2023, from React:
<https://reactnative.dev/>

Stackoverflow. (n.d.). Retrieved October 20, 2023, from <https://stackoverflow.com/>

Stat counter. (n.d.). *statcounter*. Retrieved October 19, 2023, from gs.statcounter:
<https://gs.statcounter.com/os-market-share/mobile/worldwide/>

WebAuthn. (n.d.). *WebAuthn.guide*. Retrieved September 19, 2023, from WebAuthn:
<https://webauthn.guide/#about-webauthn>

Wikipedia. (n.d.). *Software token*. Retrieved September 26, 2023, from Wikipedia:
https://en.wikipedia.org/wiki/Software_token

Wikipedia. (n.d.). *Wikipedia*. Retrieved October 20, 2023, from Wikipedia:
https://en.wikipedia.org/wiki/Public-key_cryptography

World Wide Web Consortium. (n.d.). *w3c*. Retrieved October 20, 2023, from w3c:
<https://w3c.github.io/webauthn/#sctn-sample-scenarios>

Appendices

The purpose of this section is to dive into more detailed information about the research material of the report highlighting more data.

Appendix 1: Ionic development experience.

The development experience using Ionic is that it does not offer an efficient hot reload of the application changes, every time the developer performs hot reload the app will be reloaded as a whole not only the specific component which the developer is working on, this resulting in a slower development process and high waiting times for compiling in large applications, and JavaScript not being considered a null safe language, allowing variables to have the value of null or undefined, it will result in unexpected errors and crashes which will render the application unusable.

Because Ionic is based and runs using JavaScript someone might expect that a lot of libraries can be used in the development of the mobile app, but due to compatibility issues, Ionic's Capacitor might not work with most of the already made JavaScript libraries due to either dependency issues or compiling issues, which leaves the developer in the position to either create using pieces the functionality of the JavaScript library that he want to implement or using different more complex methods.

Appendix 2: Flutter community.

The community of Flutter is growing exponentially and is now having a total of approximately 168.000 threads on Stack Overflow (Stackoverflow, n.d.) and approximately 580.000 repositories on GitHub (Github, n.d.), with a very supporting and devoted community, according to the JetBrains 2021 State of Developer Ecosystem survey, Flutter is the most popular cross-platform framework in the world.

Flutter provides with a total number of 15.590 packages that are fully compatible and can be integrated in any application, with most of them having constant support and updates from the community, offering a lot of diversity and the possibility to develop complex applications.

Some of the most popular apps developed using Flutter are Google Pay, BMW customer app, eBay, Google Classroom, PUBG mobile etc. which proves that Flutter is a popular, secure and efficient way to develop and deploy applications for large scale usage.

Appendix 3: Android and iOS market shares across the continents.

The European market consists of 65.62% Android users and 33.83% of iOS users, the remaining 0.53% is unknown. (Stat counter, n.d.)

The Oceanian market consists of 55.66% Android users and 43.29% iOS users. (Stat counter, n.d.)

The Asian market is even more Android dominant with 79.93% users having an Android and 19.39% using IOS, the remaining 0.66% are using unknown operating systems. (Stat counter, n.d.)

The same Android dominant trend is reflected in the South American market with 85.29% being Android users and 14.33% iOS and in the African market 83.36% of users having an Android powered phone and 14.03% iOS. (Stat counter, n.d.)

To choose between a native application and hybrid application depends heavily to which market is it going to be delivered. The Netherlands consists of 59.79% of users having an Android phone and 39.53% IOS, which having a very close and even distribution of operating systems it would be wise to develop an application for both Android and iOS. (Stat counter, n.d.)

Appendix 4: The results of the benchmark tests

The experiment consisted in a various benchmarks tests featuring the Accelerometer, Contacts, File system and Geolocation, monitoring different parameters such as time-to-completion(TTC) metric, CPU load, idle-state memory usage (PreRAM) and RAM usage during the benchmarking, which gave clear indications which framework is performing the best under different workloads, and comparing them to a native test for comparison.

The TTC metric showed that there are major differences between each framework, Flutter and Native Script did not show any fluctuating results however Flutter compared to the other frameworks had a higher mean TTC, making Flutter and Native Script with the best results in this test, followed by React Native, excepting the Ionic framework where the benchmark indicated that it may cross the 10.000ms mark for fetching for example, geolocation data, that other implementations. (Biørn-Hansen, Rieger, Grønli, Tim, & Gheorghita, 2020)

The CPU load test highlighted that Ionic and React Native had equivalent results and both being in the last places and standing out as less effective compared to Flutter, having values close to the native test.

The memory consumption benchmark highlighted the results for the PreRAM and RAM, showcasing the impact on memory usage caused by it, from these 2 values a 3rd value ComputedRAM was extracted by subtracting the PreRAM from the RAM during the test. "If an app consumes 85MB PreRAM in idle state, and 100MB RAM during a task run, the calculated usage for that task is 15MB – which is what the metric ComputedRAM reflects." From this benchmark it was shown that Flutter has the highest PreRAM usage but compared to the ComputedRAM results it has the lowest usage, and can be observed that has a consistent low memory usage, React Native results are low when in the RAM and PreRAM category according to the article the reason being "This is caused by alternative module implementations for the different features that deviate from recommended practices for a hand-written implementation (e.g., the geolocation module which performs slower and less accurate Facebook Inc 2019, cf. Section 4.7).", while Ionic performed used the most memory out of all of them. (Biørn-Hansen, Rieger, Grønli, Tim, & Gheorghita, 2020)

Appendix 5: Flutter vs Native vs React-Native performance results.

Below I extracted the results from the algorithms Gauss-Legendre which was used to view the CPU load, and Borwein algorithm which was used to view the memory load. (inVerita, n.d.)

Figure 15-Gauss-Legendre android results

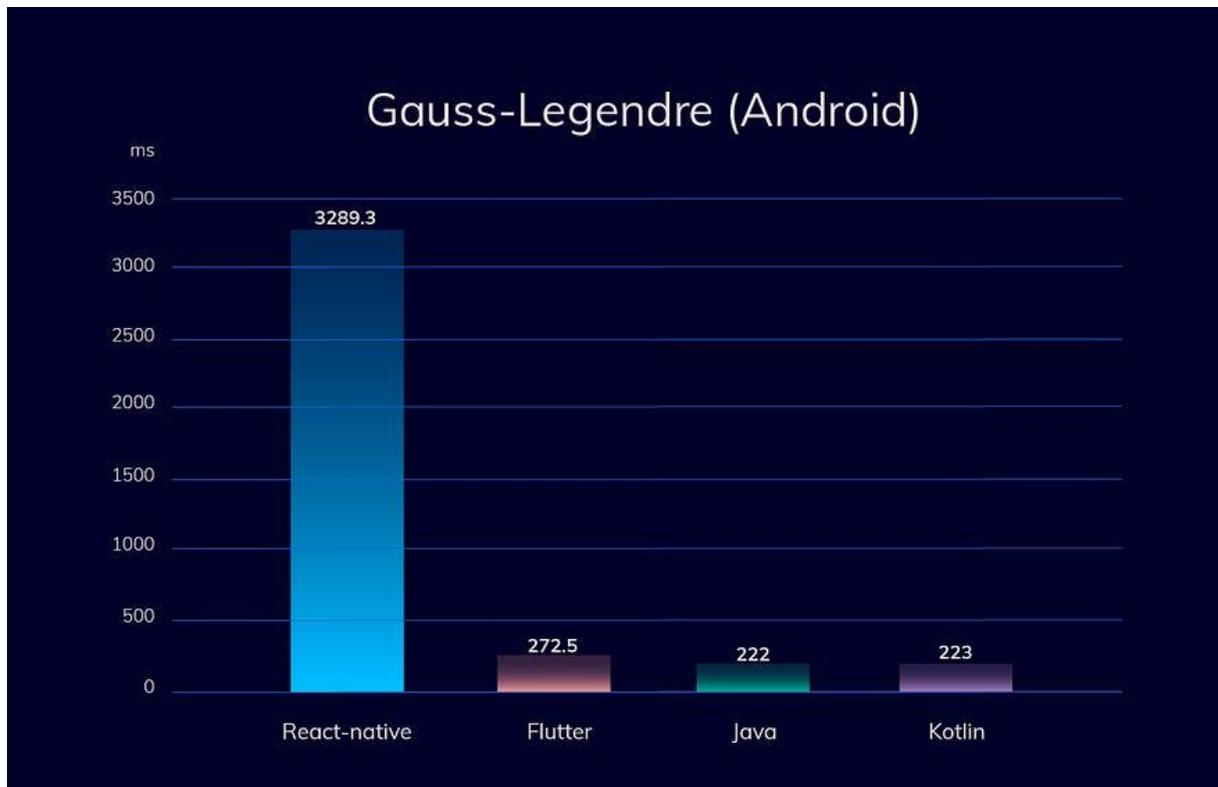


Figure 16-Gauss-Legendre iOS results

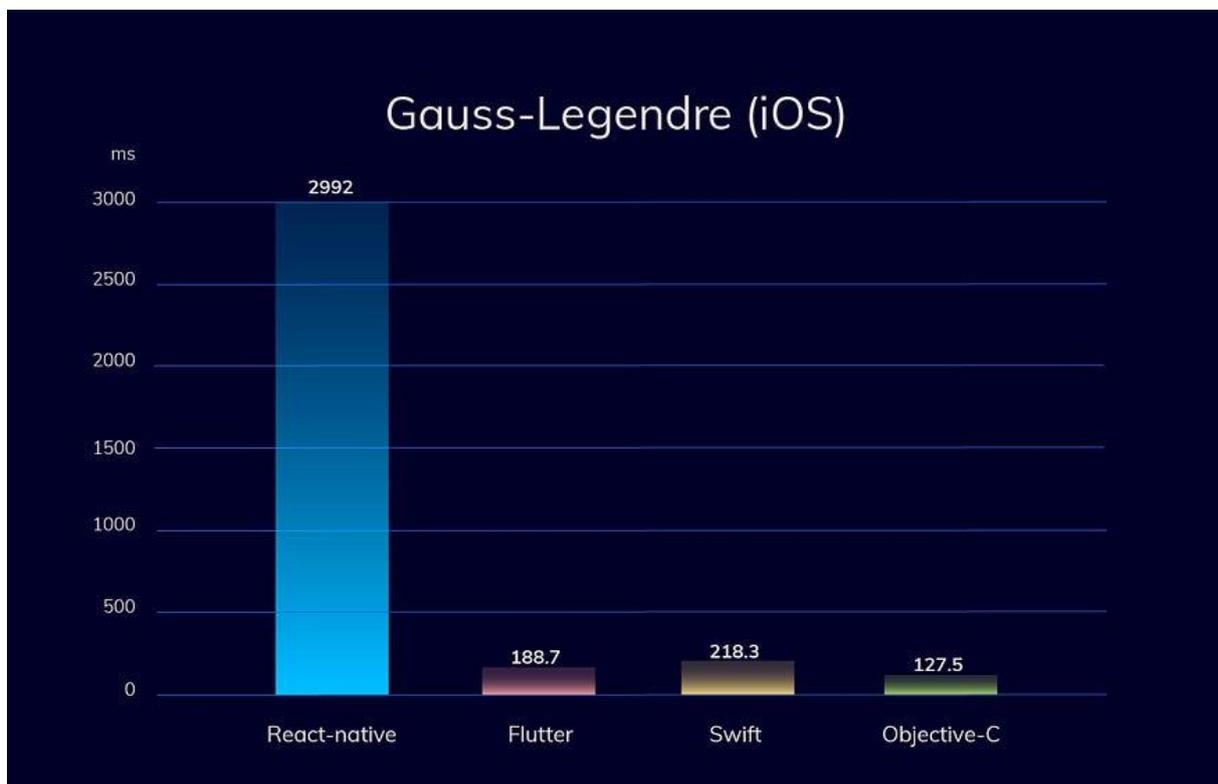


Figure 17-Borwein algorithm iOSresults

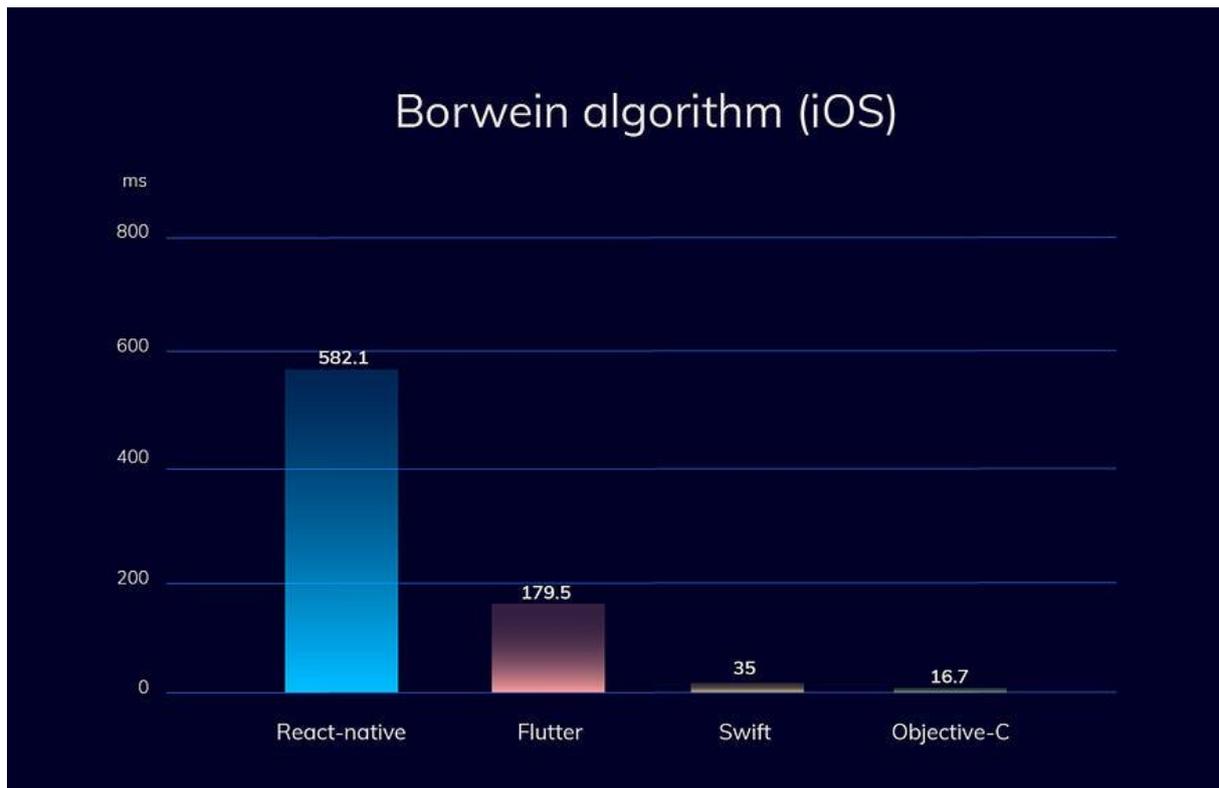
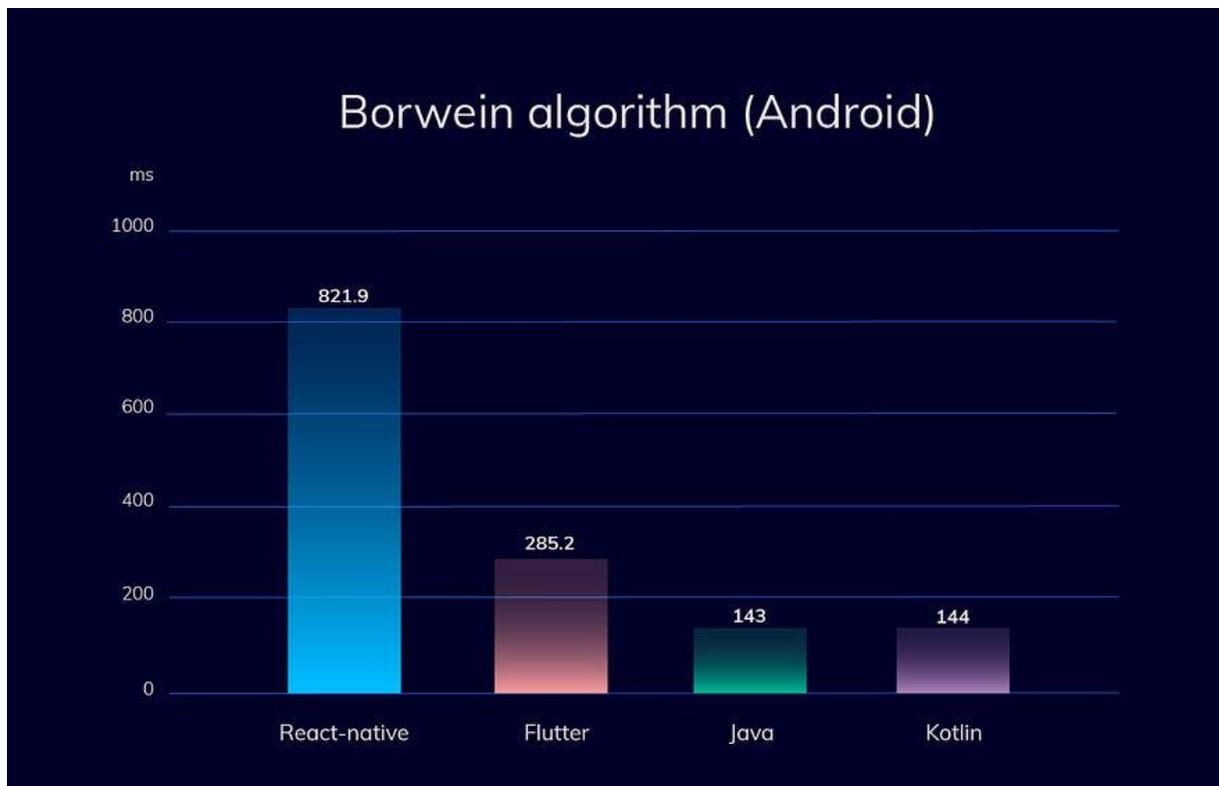


Figure 18-Borwein algorithm android results



The Borwein algorithm and Gauss-Legendre algorithm are methods of numerically approximating mathematical constants, by calculating the digits of mathematical constants like π (pi). These algorithms are great for testing the computing power of various devices due to their iterative and computationally intensive nature.

From these charts we can see that Flutter compared with React Native is the superior in both memory and CPU load, flutter being approximately 20% slower than the native frameworks, where React Native is 15 – 20 times slower than native. One exception where Flutter was 15% faster than the native language Swift was on the Gauss-Legendre algorithm for the IOS.

The article tests were done on physical devices an iPhone 6s with IOS 13.2.3 and Xiaomi Redmi Note 5 with Android 9.0. (inVerita, n.d.)

Appendix 6: Hybrid development experience and PWA

Flutter

The installation process for flutter is easy and takes approximately 5 minutes using VS code. To install flutter, you must download the Flutter SDK and add it to the environment's paths, and it is required to have Android studio or Xcode for IOS in order to create emulators, which can later be used for serving the application on. For this prototype I followed the documentation instructions which pointed me to install VS code with the necessary Flutter extensions, where after a few commands I managed to create a simple application.

The flutter extension on VS code offers the beginner the option to have a simple application with descriptive comments or a skeleton application with a list view and the most basic options such as settings, change of theme and routing, with detailed comments on what each widget does and how it is running inside the structure.

The implementation for biometrics authorization is straightforward with a library "[local_auth](#)" it allows the developer to check and call for a biometric authorization, it can be viewed what type of biometric sensors are available, in case of failure to authenticate using the sensors the application can requested to enter the phone password.

Other sensors such as vibration and camera accessibility have been easily implemented with little to no errors as the documentation for the packages are clearer with detailed examples on what are the capabilities of package.

The most challenging part of the prototype was writing the code as it is structured in widgets it can get very ramified into smaller and smaller parts that the benefit for it is the focus on development without the need to reload the application. The architecture is very structured and as the previous sections mentioned Flutter is a very null proof application as it is necessary for every operation with objects to have special classes that manage the manipulation of data.

In the beginning is a slow-paced environment but as familiarity increases, with the support of the community examples and documentation the widgets can get made fast.

React native.

The installation process and to run React Native was quicker compared to Flutter as it was necessary only a line of code to fully create a project and in combination with the app Expo Go the developer can view the application and change in real time.

Compared with Flutter React Native offers a faster set up time with a physical device, as it is necessary to only be on the same network and access using the app Expo Go the running app. (React Native, n.d.)

The implementation of the established features took less time to write the code debug and test compared to Flutter, and it supports all the features without any issues. The application using the Expo environment allows the developer to access the available libraries from the development team behind expo to use the devices sensors (vibration, camera, and biometrics). As being a framework written in JavaScript it had the most familiarity in terms of creating screens, accessing functions and implementing components.

The most challenging part of this prototype was the authentication process with API calls as the library '[axios](#)' is will through an error as the compiler of the app confuses it with some IOS component, to solve this issue the function fetch has been used.

Ionic

The installation process for Ionic capacitor was the hardest out of all the frameworks tested so far, it had compiling issues, version matching issues and lack of documentation for debugging these issues.

The development experience was the most difficult out of the tried hybrid development frameworks as it caused a lot of errors and compatibility issues with java and Gradle versions in order to properly run, the documentation is not straightforward in terms of how to use and update the application when adding libraries as example. I had to follow extensive tutorials and guides on the most basic aspects of Ionic Capacitor.

The only instance that worked without that many issues was using the quasar framework in combination with the capacitor plugin, which allowed me to access the device native features (biometrics, camera and other sensors) with already made capacitor libraries.

In order to have access to the biometrics sensors of the phone, only a single library was compatible with the latest version of capacitor the "[aparajita/capacitor-biometric-auth](#)", as the rest of the available libraries are outdated and the default capacitor functionality does not support yet biometric authentication functions.

The community for issues and bugs regarding compiling is small and it does not provide clear and multiple solutions to any errors encountered.

PWA

The installation process of a PWA is quite simple as it can only be done from a single command and choose the PWA template which will set the necessary dependencies for a web application to become a PWA.

The main objectives established for the prototyping have been achieved by using the WebAuthn API to access the fingerprint sensor for authentication on the device. The technology works by creating a public and private key pair, during the registration process the fingerprint check will be triggered as to proof identify of the user trying to create the pair on the device.

During the prototyping of this technology, the usage of the fingerprint sensor and checking the identity of the user anytime without relying on the WebAuthn API is currently not possible.

A PWA is great for simple tasks that does not require complex features with high processing power usage.

Appendix 7: Backlog and changes

In this appendix I am highlighting what has been discussed during the progression meetings and an overview of the changes and decision taken during this graduation period.

Table 9-Backlog and changes

Date	Progress	Changes
10.11.2023	First progress meeting. Deciding what frameworks and technologies are going to be used.	Initially it was the decision between a PWA and Flutter as a mobile solution but eventually choose Flutter as being a more secure and flexible solution.
17.11.2023	Code base for each part of the project. Documentation.	Changes in the back-end system to move the API endpoints for the WebAuthn method of authentication (Windows Hello / Fingerprint) from Symfony to Node JS Express due to compatibility issues.
24.11.2023	Initial design of the changes in the front-end Basic API endpoints for the WebAuthn API. Database implementation and design.	
1.12.2023	Fully implementation of the Windows Hello / Fingerprint method of authentication using the WebAuthn API. Documentation.	
8.12.2023	Base design of the mobile application Implementation of the WebSocket communication. Drawing the login between how the mobile application will communicate with the back end. Necessary API endpoints	
15.12.2023	Fully implantation of the QR code login method of authentication. Design improvements Documentation.	
22.12.2023	Fully implementation of the biometrics method. Design improvements. Documentation.	

29.12.2023	Holiday	
5.01.2024	Starting the testing phase. Documentation and final report. Improving the design and bug fixing the mobile application.	
12.01.2024	Unit testing Linting and code quality tests iOS Testing Documentation Final report update.	
19.01.2024	Final report update. Android testing.	

Appendix 8: Linting criteria for Flutter

The analyses criteria that this linting testing does are: (Dart, n.d.)

Table 10-Linting criteria details

Name	Description
avoid_print	Discourages using the “print” function (the equivalent of console.log ()), as it can lead to performance issues and cluttered output in production apps.
camel_case_types	Ensures that type names are in Upper CamelCase, as it improves code readability and consistency as well as conforming to the Dart’s naming conventions. Classes and typedefs should capitalize the first letter of each word (including the first word) and use no separators.
constant_identifier_names	Enforces that constant names are written in uppercase with underscores, as it makes constant variables more identifiable and distinct to improve code readability and consistency.
empty_constructor_bodies	Ensures that there is no usage of empty bodies for constructors. In Dart, a constructor with an empty body can be terminated with just a semicolon. This is required for constant constructors for consistency and brevity
prefer_const_constructors	Encourages the usage of constant constructors which improves performance by enabling compile-time constant expressions.
always_declare_return_types	Ensures that all functions declare their return types explicitly, to increase code clarity and understanding the function’s intent.
always_require_non_null_named_parameters	Requires named parameters without default values to be marked as @required, enhancing API clarity and prevent runtime null errors.
annotate_overrides	Enforces the use of @override annotation for overriding methods and properties and improves code readability and accidental method overloading.

avoid_init_to_null	Discourages explicitly initializing variables to null, as it reduces unnecessary code since variables in Dart are null by default.
avoid_null_checks_in_equality_operators	Provides a warning against the usage of null checks in equality “==” methods preventing incorrect null comparison behavior and improves method consistency.
avoid_relative_lib_imports	Discourages importing libraries with relative paths, enhancing code maintainability and clarity by using package based URIs.
avoid_return_types_on_setters	Checks that setters do not have a return type, as per Dart language conventions setters should not return values.
avoid_shadowing_type_parameters	Creates a warning against shadowing type parameters within the same scope to prevent confusion and potential errors due to type parameter shadowing.
avoid_single_cascade_in_expression_statements	Discourages using cascades for single method invocations to allow the code to be more simplified and readable without unnecessary cascade syntax.
avoid_types_as_parameter_names	Prevents using type names as parameters names improving code readability and structure.
await_only_futures	Verifies that “await” is only used in “Future” objects to prevent runtime errors and ensures correct use of asynchronous programming constructs.
camel_case_extensions	Extensions should be capitalized on the first letter of each word (including the first word) and no separators, to improve consistency and readability according to Dart naming conventions.
curly_braces_in_flow_control_structures	Checks the usage of curly braces in structures to improve maintainability and readability in complex structures.
empty_catches	Avoids empty catch blocks, to not silently ignore exceptions and always handle them.
library_names	Enforces naming libraries using the lowercase_with_underscores, as some file systems are not case-sensitive, it is required filenames to be all lowercase and have underscores to improve readability.
library_prefixes	Ensures that when specifying a library prefix it is in lowercase_with_underscores, improving a uniform and readable library prefix across the code.
no_duplicate_case_values	Checks for duplicate values in switch case statements to prevent logical errors and make each case unique. (
null_closures	Prevents passing null as an argument for closures as it can cause unexpected behavior and runtime errors.
prefer_adjacent_string_concatenation	Checks for concatenating strings that are adjacent literals, to simplify string handling and improve performance.
prefer_collection_literals	Encourages the use of collection literals to initialize collections to improve code conciseness and readability.

prefer_conditional_assignment	Suggests the usage of "??=" for assigning a value to a variable if it is null, to reduce code redundancy and improve clarity.
prefer_contains	Suggests the usage of contains for List and String in testing
prefer_final_fields	Suggests declaring fields as final if they are not modified after initialization, to encourage immutability leading to a safer and predictable code.
prefer_for_elements_to_map_fromIterable	Recommends using for-elements when converting an iterable to a map, to enhance code readability and efficiency in collection manipulation.
prefer_generic_function_type_aliases	Encourages using generic function type aliases over typedefs with parameterized return types to help in a clearer and more concise function signatures.
prefer_if_null_operators	This rule suggests the usage of if null operators instead of null checks in conditional expressions
prefer_inlined_adds	Checks that elements declared in a list are inline rather than using add and addAll methods, to improve readability and conciseness
prefer_is_empty	Suggests using 'isEmpty' to check for empty collections as it is more expressive than checking the lengths.
prefer_is_not_empty	Suggests using 'isNotEmpty' over '!isEmpty' to improve code readability.
prefer_iterable_whereType	Encourages using "whereType" for filtering elements of a specific type, to simplify and clarify type filtering collections
prefer_single_quotes	Encourages using single quotes for strings when they do not contain single quotes, improving maintainability in string declaration
prefer_spread_collections	Encourages the use of spread operators ("...") in collections, to increase code clarity.
recursive_getters	Give warnings against getters that are recursively call themselves, to prevent potential stack overflow errors and enhances code safety
sort_child_properties_last	Suggest the placement of child and children property of the widgets at the end in terms of declaration order to enhance readability by following a conventional and predictable order in the widget tree and properties.
type_init_formals	Discourages the repletion of the type of parameter in functions and constructor as it can lead to redundancy in the code
unawaited_futures	Provides warnings about "Future" functions that are not awaited or returned, to help in preventing unintentional asynchrony bugs and ensures that the functions are handled properly
unnecessary_brace_in_string_interps	This rule advises against the usage of braces in string interpolation when not necessary, as it helps simplify the string interpolation and creating a clearer and readable code
unnecessary_const	This rule gives warnings when unnecessary "const" keyword is used to reduce code clutter and emphasizes where "const" is important and needed

unnecessary_getters_setters	Checks and discourages using getters and setters without additional functionality and logic, it provides the usage of simple alternatives to improve code simplicity and readability
unnecessary_new	Suggests against using the “new” keyword as it is optional in Dart, to contribute to a more cleaner and modern Dart syntax
unnecessary_null_in_if_null_operators	Checks and warns against using null as the second operand in “??” operators, to avoid redundancy by highlighting the intention of null-aware operators
unnecessary_this	Gives warnings when unnecessary usage of “this” is made, to improve readability and removing redundant language constructs
unrelated_type_equality_checks	Checks and gives warnings to equality checks between unrelated types, to help detect potential logical errors due to comparing incompatible types
use_full_hex_values_for_flutter_colors	Recommends the usage of full hex values for color codes in Flutter, to enhance consistency and clarity in defining color values.

Appendix 9: Regression tests

Below there are a table of the regression test cases that covers the main functionality of the project and ensures that after each change or update of the system the previous and already implemented features have not been affected.

Table 11-Regression tests

Name	Description	Steps	Bugs
Windows Hello / Fingerprint authentication	Login into an account that has a windows hello / fingerprint authentication method set.	-Login into: X account -Windows hello interface should be prompted. -Provide fingerprint authentication -User should be logged in.	
QR code scan login	Login into an account that has QR code scan as a method of authentication.	-Login into: X account -A QR code should be displayed -Open the Authenticator application -Slide to open the camera for QR code authentication. -Scan -User should be logged in	
Biometrics login	Login into an account that has biometrics as a method of authentication	-Login into: X account -A message of waiting for confirmation should appear -Open the Authenticator application -Slide to the right for the biometrics authentication	

		<ul style="list-style-type: none"> -Provide the biometrics check -User should be logged in 	
Register method of authentication (Windows hello / Fingerprint)	This test should check if a fresh account can register the method of authentication Windows hello / fingerprint	<ul style="list-style-type: none"> -Login into: X account -A menu of three methods of authentication should be displayed -Choose first option -Provide the Windows Hello fingerprint to register -Provide the windows hello fingerprint to authenticate -User should be logged in 	
Register method of authentication (QR code scanning)	This test should check if a fresh account can register the method of authentication QR code login	<ul style="list-style-type: none"> -Login into: X account - A menu of three methods of authentication should be displayed -Choose second option -Open the Authenticator application -Slide to the QR code scan to open the camera -Scan the QR code on the screen -The front-end should ask for a code starting in the format XX-XXXXX with the first two numbers being revealed -Provide the code from the email -User should be logged in 	
Register method of authentication (biometrics login)	This test should check if a fresh account can register the method of authentication biometrics login	<ul style="list-style-type: none"> -Login into: X account - A menu of three methods of authentication should be displayed -Choose third option -Open the Authenticator application -Slide to the right for biometrics authentication. -Input the code displayed on the screen -Provide a biometrics check -The front-end should ask for a code starting in 	

		<p>the format XX-XXXXX with the first two numbers being revealed -Provide the code from the email -User should be logged in</p>	
--	--	---	--

Appendix 10: hybrid vs native performance

In this appendix I am highlighting the advantages and disadvantages of the hybrid and native frameworks in terms of performance and business benefits. (Grigalashvili, n.d.)

Performance

In terms of overall performance of the application, the relationship between the language that the framework is written makes a stark difference in how an application will have an efficient way of using the phones hardware and processing power as well as battery consumption.

Native will overpower hybrid development in this category as being the closest to utilizing the devices hardware components will require less computing power and overall have a low battery consumption.

Hybrid on the other hand is constantly evolving and engines and frameworks that support the compilation of the hybrid code is getting increasingly efficient with each version or even new frameworks that appear. Which makes the performance between Native and Hybrid close and soon might be a close to match.

The performance of a native framework currently is considered faster than a hybrid framework.

UI/UX

The UI/UX experience is one of the key factors in deciding which framework to use as the main way users will decide to use or not the app.

Native frameworks have the advantage of utilizing the already made native interfaces of the operating system which make each application look and feel friendly and easy to use for the users of their specific platform.

Hybrid frameworks will not be able to 100% match the UI/UX of a native created application as they will never provide users a fully native experience, and only by compromising performance hybrid frameworks might match and give the native look for the application.

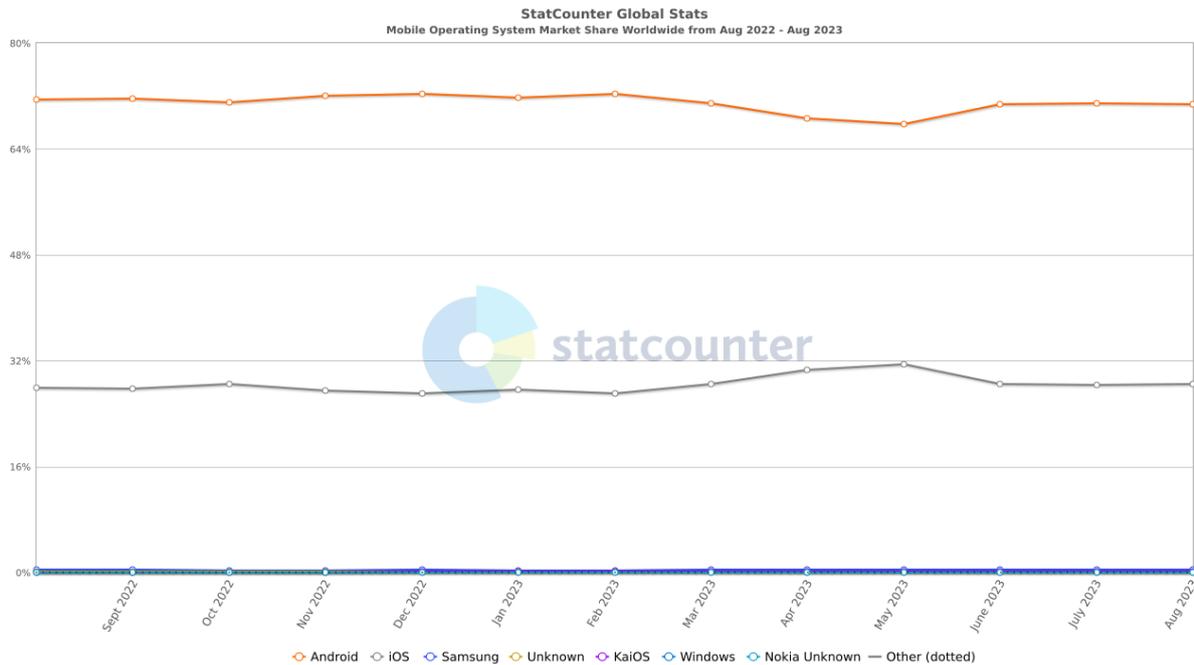
The overall conclusion is whatever the application needs to have a native look or keep the same design across all platforms, and to what extent should the performance be sacrificed in case of hybrid development.

Audience

When choosing between creating applications it is important to whom it is going to be delivered to and to what market.

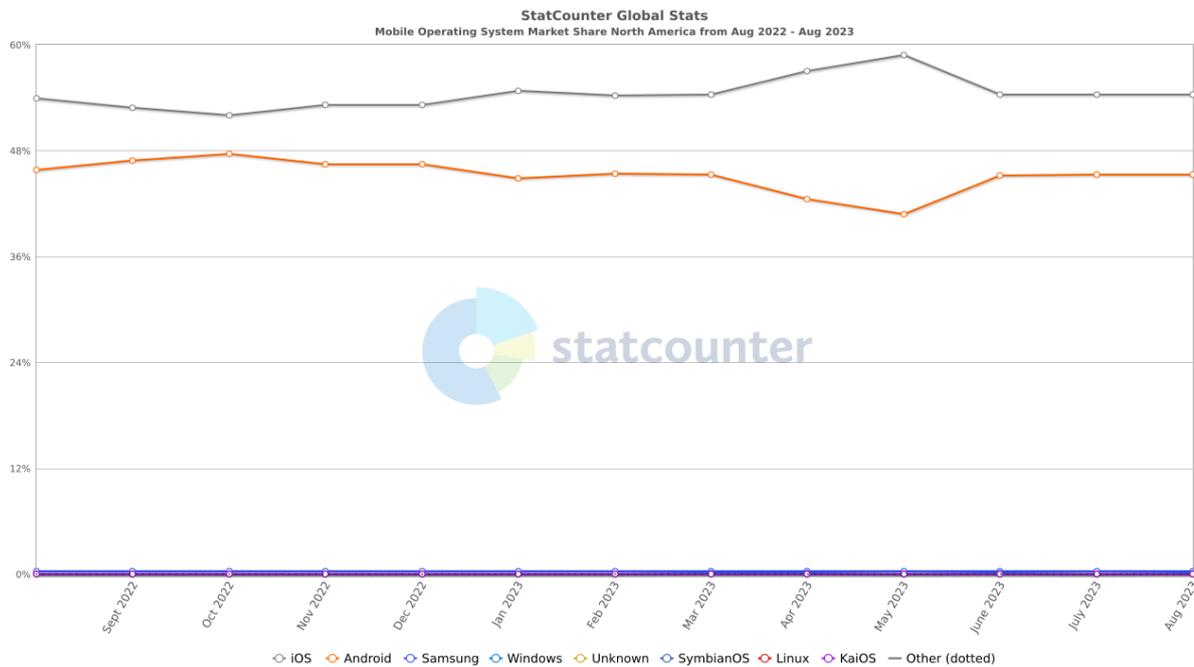
For example, according to this chart by Stat Counter, worldwide there is a majority of users using a phone with Android as the operating system.

Figure 19-StatCounter android vs iOS global share



But in the chart below if the company is based or wants to deliver the application to the North America market, it will be faced with different values as the IOS operating system is in the majority of the user’s mobile phone. With values of 54.32% using IOS and 45.24% Android (Stat counter, n.d.)

Figure 20-StatCounter android vs iOS N.A. market share



Based on the “[Appendix 3: Android and iOS market shares across the continents](#)” the regional distribution of mobile operating system highlights the Android dominance in regions like Asia, South America and Africa with a more evenly split market in the Netherlands and Europe.

Security

When it comes to how secure an application is, we must look at what makes an application secure in the first place.

The Native frameworks due to the nature of the accessibility to all the mobile phone features it is advantageous in implementing security measures and due to the policy of the App Stores in having each Native application undergo more stringent reviews on the submission process, it will ensure that as soon as a native application is on the store meets certain security standards and guidelines.

Running directly with and on the device's, operating system provides a degree of code isolation which makes penetrating the application harder, this comes with also the disadvantage on the application security relying on the operating system security which if a security vulnerability is found, the native application must wait for the operating system to release an update. But having different native applications consist in different approaches when making the application secure, which leaves the app being vulnerable depending on the platform, as IOS and Android operating systems work differently when it comes to security measures, IOS being considered most secure and harder to penetrate than Android.

The hybrid frameworks having the advantage of sharing a single codebase across multiple platforms can make it easier to maintain and improve the security measures consistently, but if vulnerabilities exist in the shared code, it can affect multiple platforms.

A hybrid application can use third-party plugins and libraries which can induce security risks if not carefully implemented and documented, and it is important to keep those components secure and up to date. Due to this factor hybrid applications undergo additional security by having third-party plugins implemented.

By not relying mostly on the operating systems security, hybrid applications can be developed with more flexible and improved security measures which are isolated from the underlying system on which they are operating.

Costs and time.

And the most key factor when choosing between a native framework and a hybrid framework is the financial consumption and the time it takes for development and updates.

Native development can only be done by specialized teams that are familiar with programming languages like Swift, Kotlin, Java, C++, C or Objective-C. This factor alone requires the need of having multiple employees working on different teams which is an excessive cost for a single application to be developed.

Native development can have the issue of desynchronization of the development, as one platform can have a faster or slower development than the other and for releases and keeping track of the progress and features of the application will be very challenging, as for example the Android team already be finished with the application and the IOS team requiring additional time, which ultimately makes the Android development either go forward increasing the gap in the synchronization between the app versions and features or hold the development and lose time and have financial consequences. With also the risk of one platform being more vulnerable or less vulnerable than the other security issues will arise.

This process alone makes choosing a native development a hard decision for each company, as it requires professional employment in both the platform fields and the management field in order to have a proper development.

Hybrid development in this category has the advantage, by having a shared codebase for multiple platforms it is easily maintained, secured, and provides a faster development with twice the difference in costs, as it is only necessary for a single team to handle the development of the application which removes the potential of desynchronization between the platforms and creates an efficient and fast pacing environment for the development. Updates and version control is done efficiently, and the release of updates is faster which makes updating the application in case of vulnerabilities or glitches greater than waiting for native development on both platforms.

Overall hybrid development is the better choice for financial costs and time efficiency, as it requires a single unified team with a single code base for all the platforms that the app is releasing on, and updates and fixes are faster and easier to implement.

Appendix 11: PWA advantages and disadvantages

A PWA is written in JavaScript, CSS and HTML and looks and behaves just like a regular web application, and it can be installed on any native device, which gets rid of the need to creating separate applications in either native or hybrid frameworks. (Brainhub, n.d.)

The PWA as in the case of a hybrid application it shares a single code base and are a great alternative to hybrid frameworks offering a cheaper alternative with a faster development time with about 50%-75% less time than a traditional native and hybrid development time. And it is not necessary for the users to install the application as a traditional application, it would simply be added as an icon to the phone.

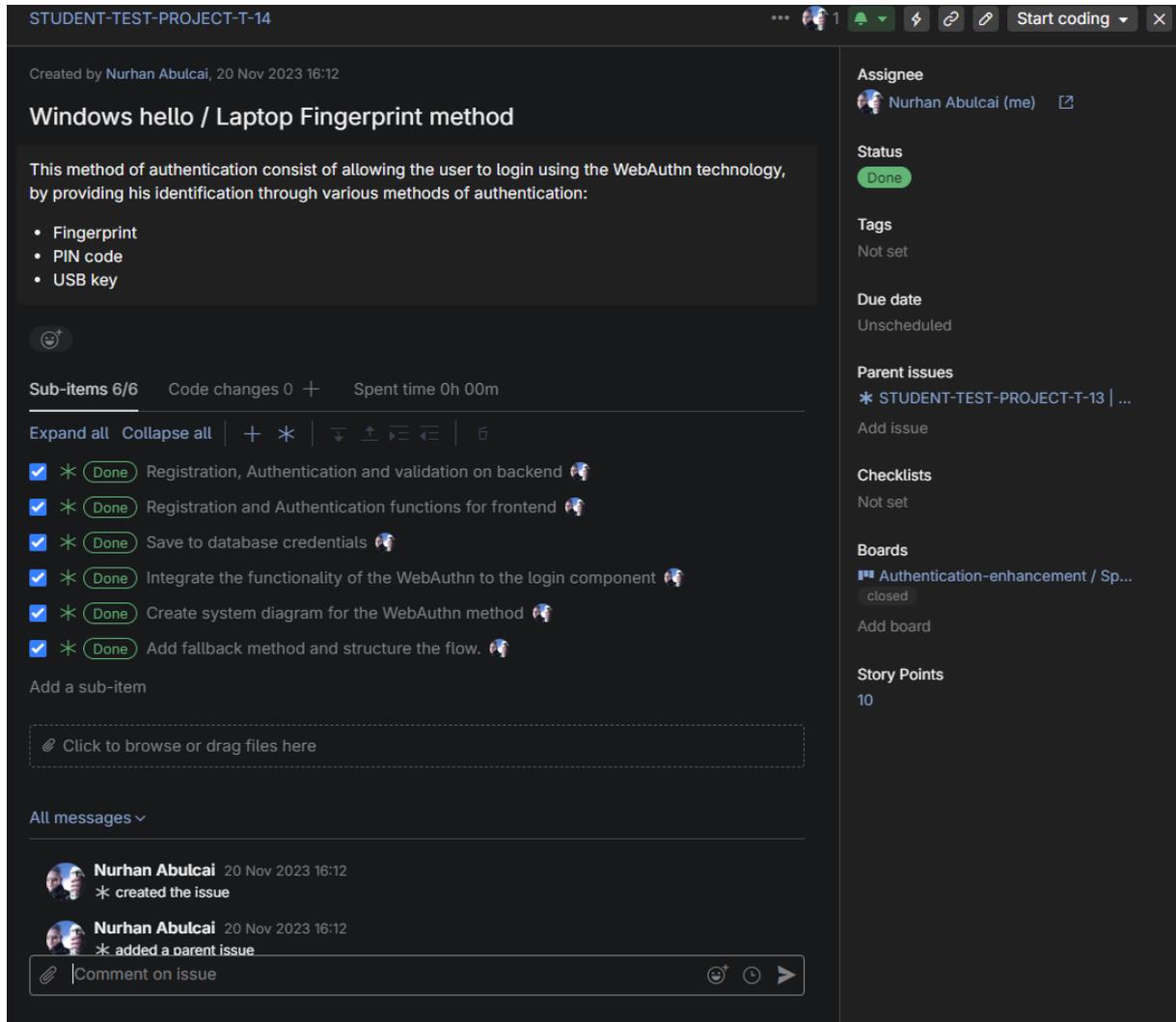
The biggest disadvantages of a PWA are the limitations in terms of native features and accessibility as being a recent technology not many features are as easily accessible compared to the native or hybrid frameworks.

Compared with hybrid applications, a PWA application is a fast and cost-effective but with the limitations of performance, features and low security protection, being a great solution for simple and not advance mobile feature dependent applications.

Appendix 12: Backlog and issues

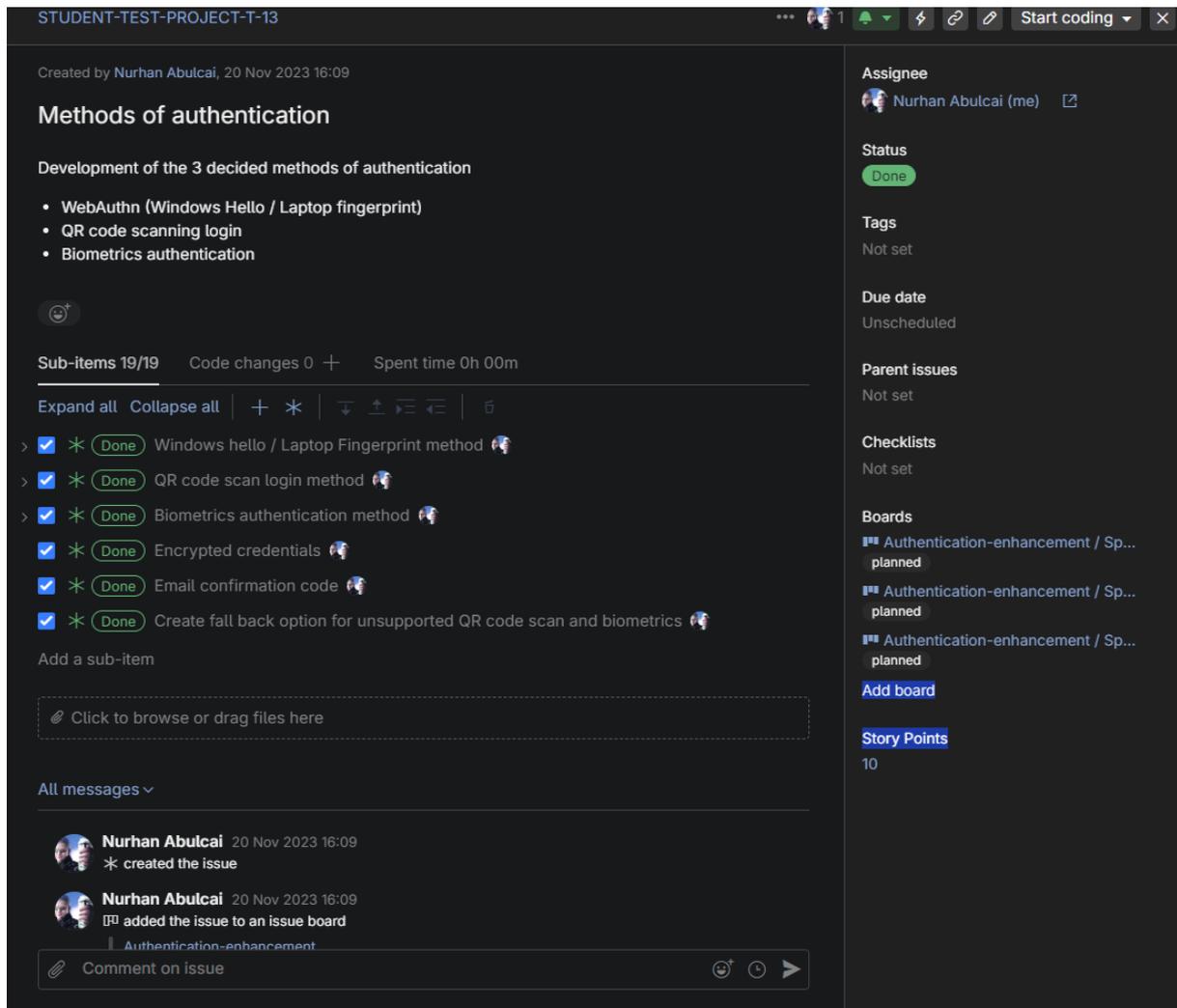
In this section I am going to highlight the backlog and principal issues that had been completed for this project.

Figure 21-Issue example



The picture above it can be viewed the main issue and sub-issues of the first authentication method developed using the laptop fingerprint sensor thorough the WebAuthn API, the issue features sub issues that by completing them results in a fully functional feature for the project. This issue being a sub-issue to the most important issue of the project which is the Methods of authentication implementation issue, being the parent issue for the 3 methods that have been developed and the additional tasks that needed to be completed for the goal of implanting the 3 methods to be fully completed.

Figure 22-Parent issue



This issue was keeping track of the progress and what needed to be achieved during the project, such as the completion of the Windows Hello (WebAuthn) authentication, QR code scanning login and Biometrics authentication with encrypted credentials and email confirmation during the registration, and much more sub-issues that collectively result in the successful build of these 3 methods and the main goal of the assignment.

Below there are screenshots of proof of sprint completion and the achievement of the goals, with sprint five being still undergoing a few issues in terms of testing and documentation are still in progress.

Figure 23-Sprint 1

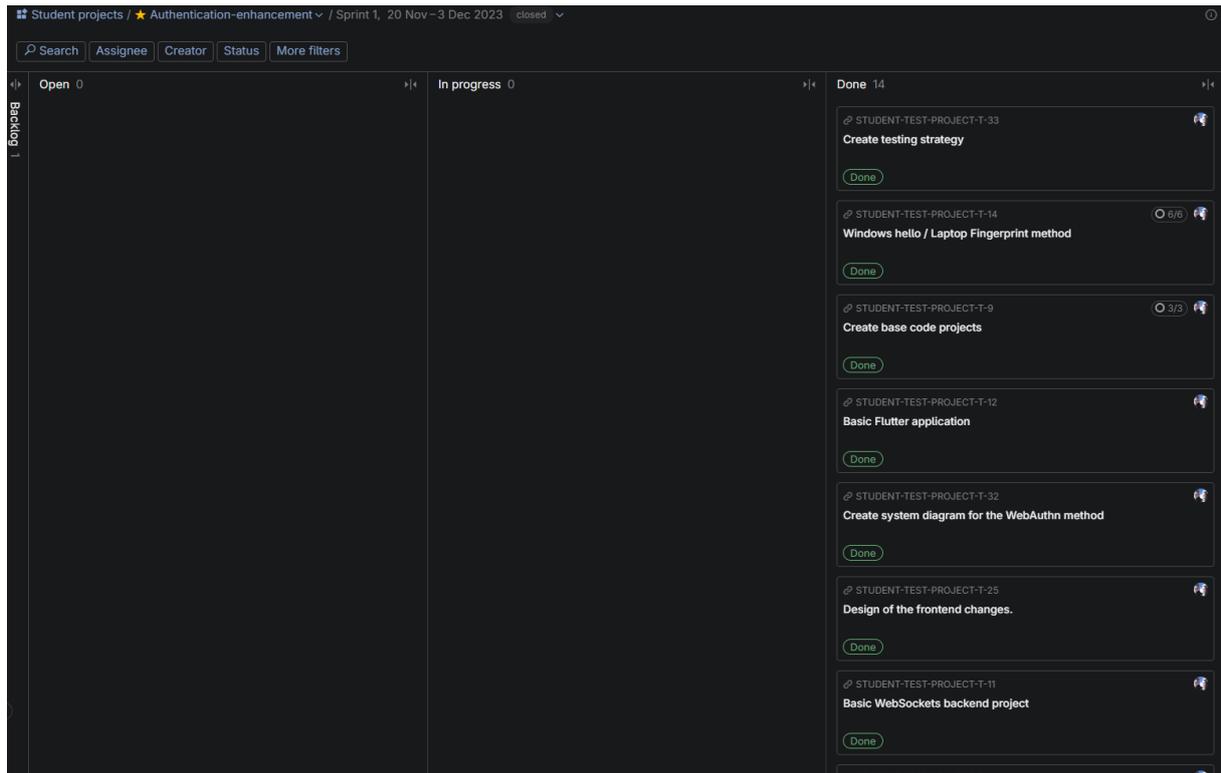


Figure 24-Sprint 2

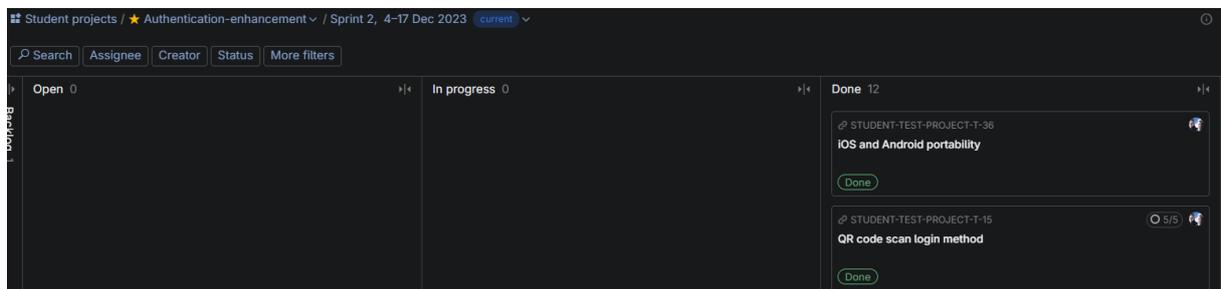


Figure 25-Sprint 3

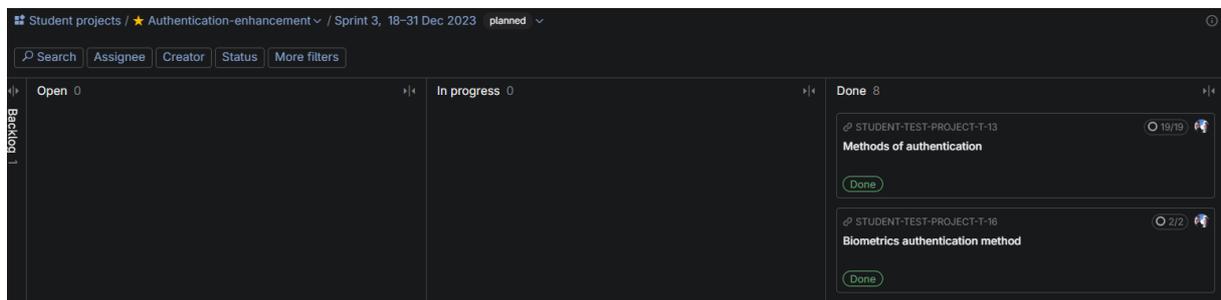
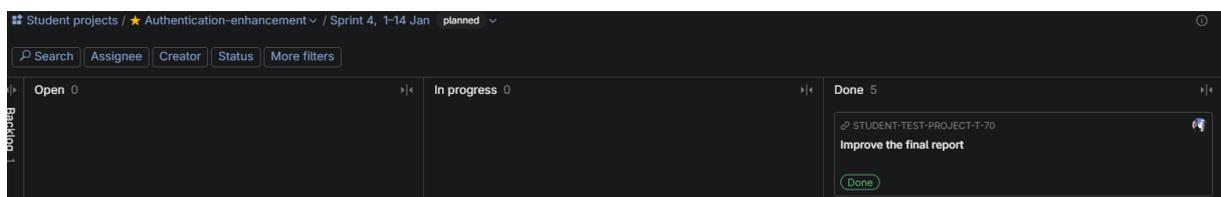


Figure 26-Sprint 4



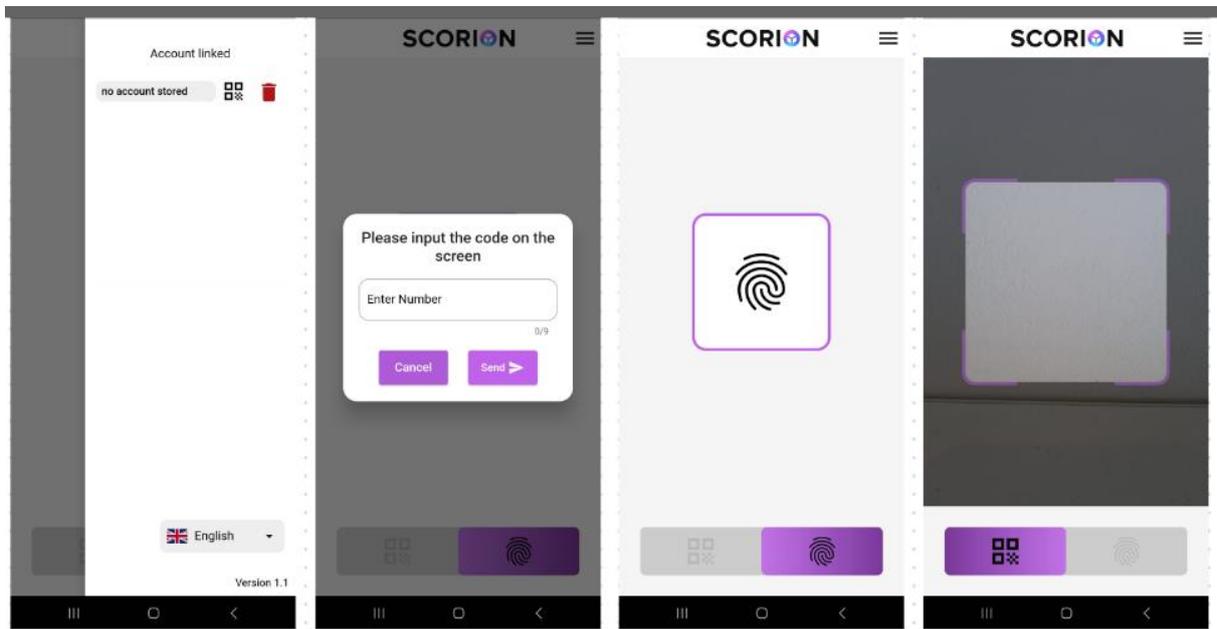
Appendix 13: Mobile testing

The requirements established for this project regarding the mobile application have been tested through several manual testing which cover:

RQ-F5: The mobile application must be accessible for both iOS and Android.

This requirements has been tested using the company MacBook through Xcode that handles the iPhone emulators, these emulators have been used successfully to run the application, the testing for iOS is still undergoing as a proper physical device is required to be tested on as Xcode does not support camera simulation. The android testing can be viewed in the screenshot below as the application is successfully run and operational on a physical Samsung S21+ device. Which also cover the requirement RQ-F7 that the design must be approved by and the UI/UX designer.

Figure 27-Mobile testing proof



The requirements RQ-NF-S1, RQ-NF-S2, RQ-NF-S14, RQ-NF-S15 have been tested through manual testing on various emulators, Nexus 5X, Pixel 3a etc. to cover a variety of screen sized to check UI/UX responsiveness. The language implementation for Dutch and English as well as the requirement for the phone status, can be viewed in the figure below.

The testing on the emulators and physical device have been successful without any issues.

Figure 28-Emulator UI



Figure 29-Phone status and language proof

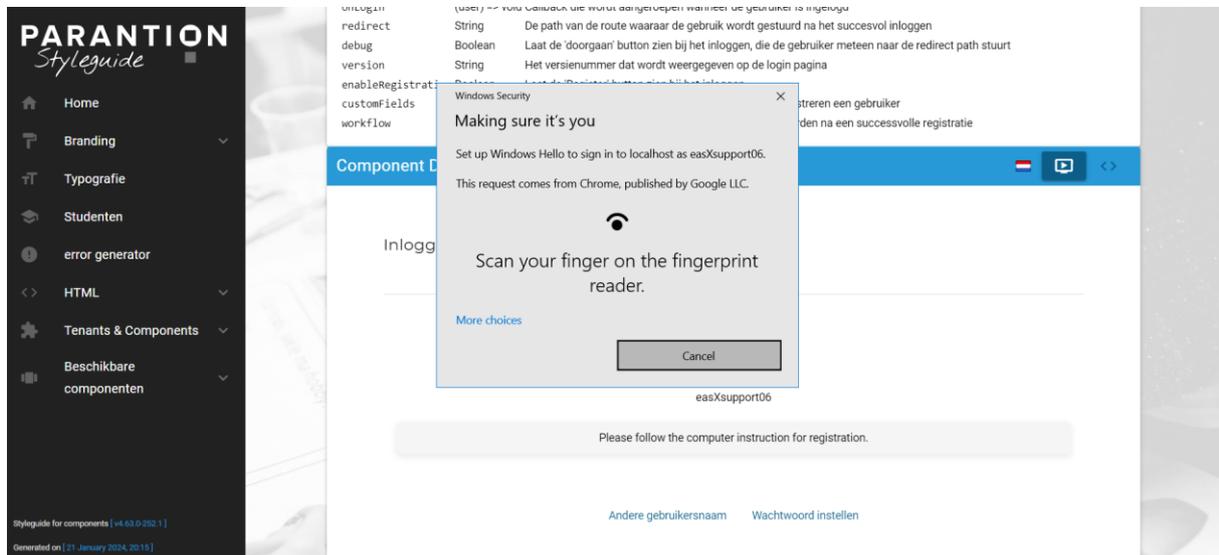


Appendix 14: Testing methods of authentication and email validation

Below there are several screenshots highlighting the manual testing of each of the methods of authentication as well as the email validation.

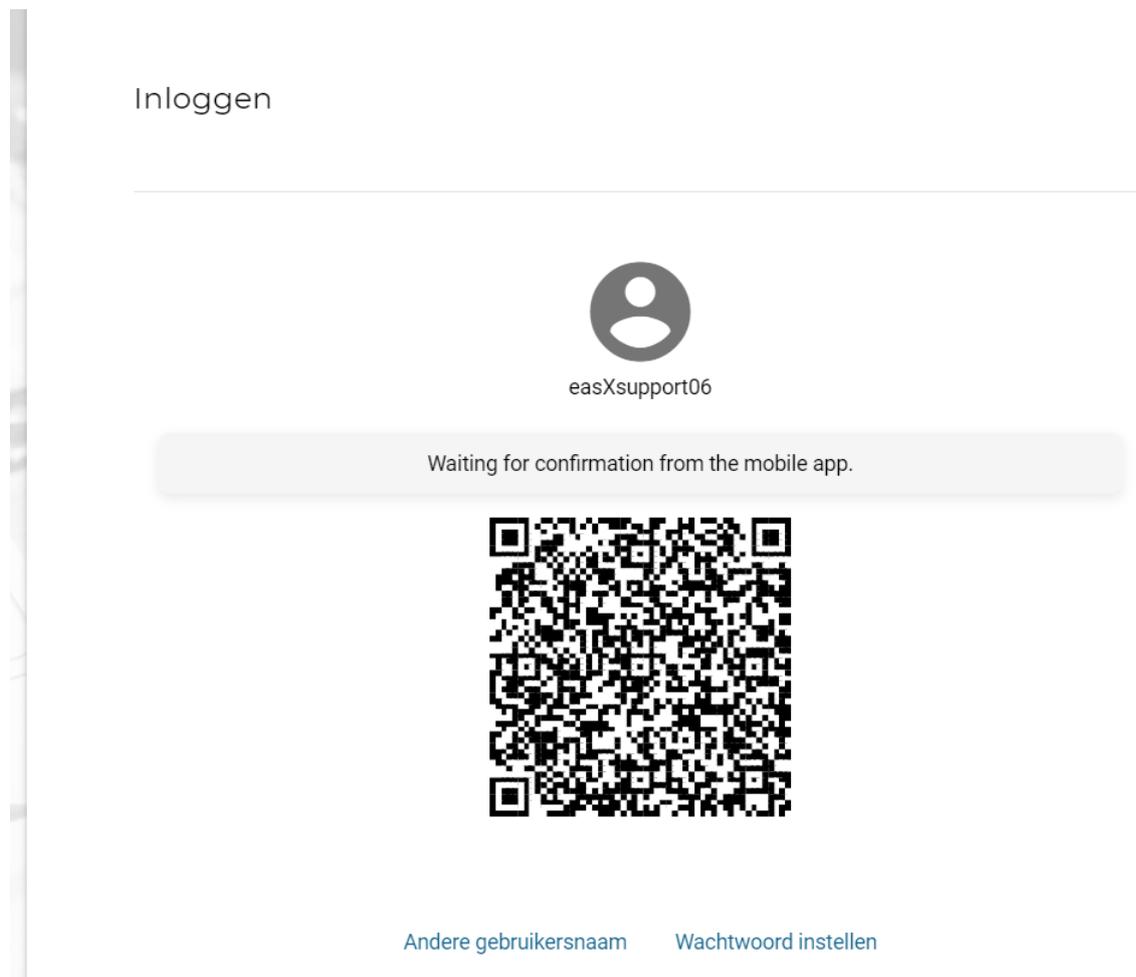
Laptop fingerprint using WebAuthn.

Figure 30-WebAuthn Windows Hello prompt for authentication and registration



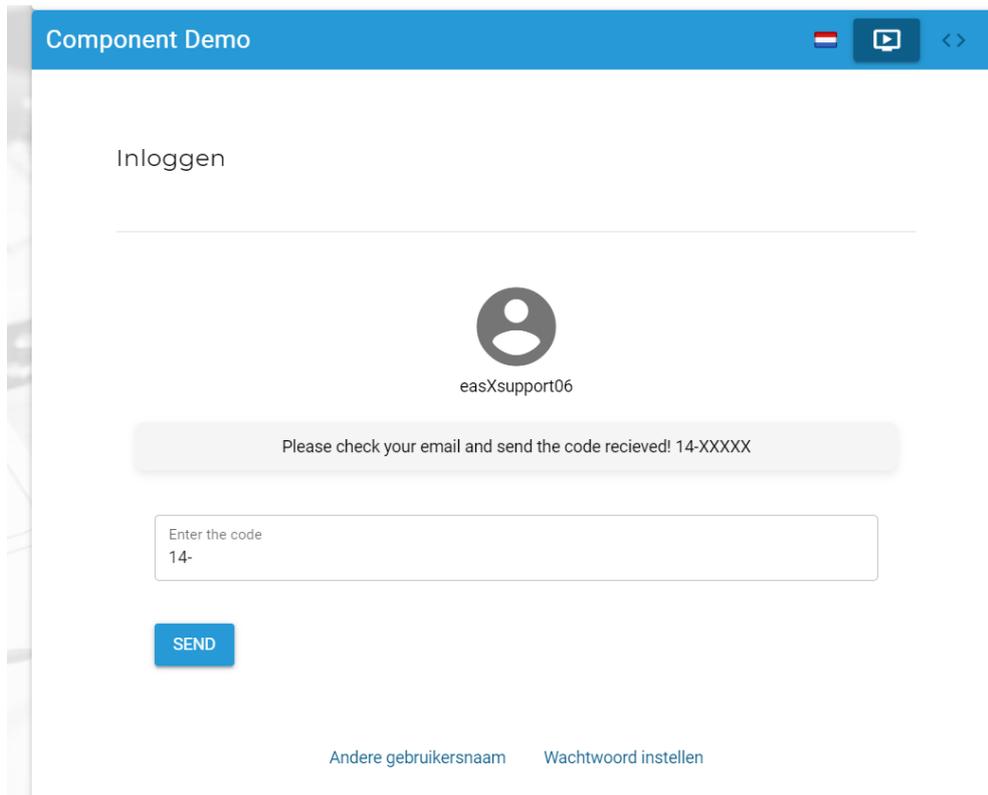
Mobile QR code login

Figure 31-QR code display



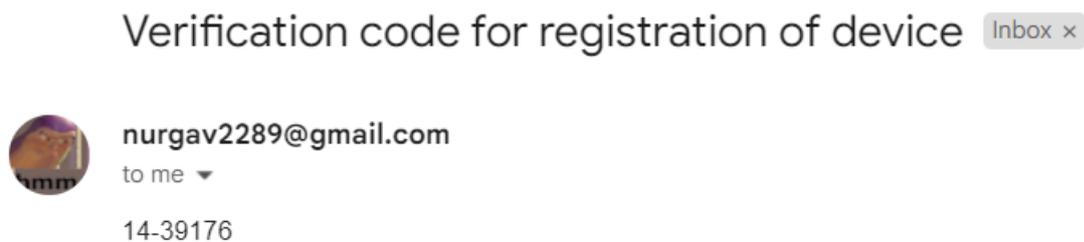
In the figure above the QR code for registration and login can be viewed.

Figure 32-Email code validation input



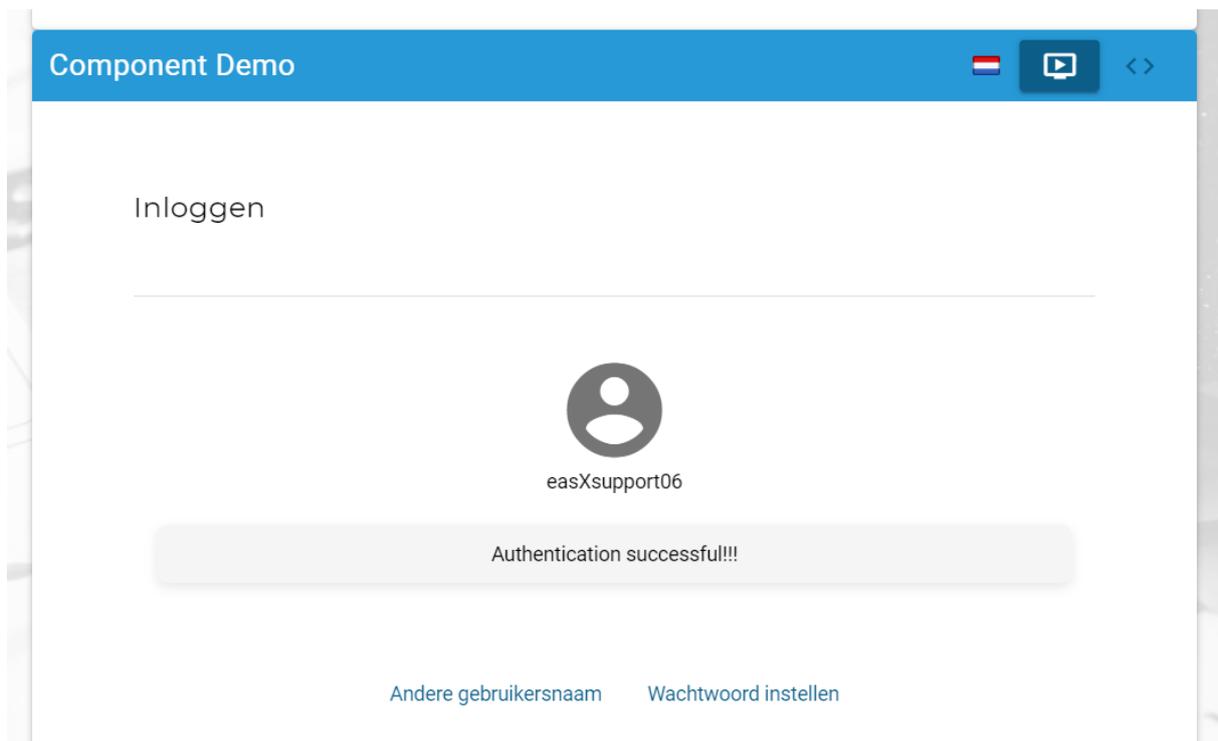
After scanning the QR code with the mobile application, it the figure above displays the request of email code validation according to the requirement RQ-NF-S16.

Figure 33-Email received proof



In the figure above is proof that the code is received through email.

Figure 34-Authentication / Registration successful



And the last step displaying that the authentication is successful and the user is logged in.

Mobile Biometrics login

Due to privacy reasons of the smartphone operating system, screenshots and video evidence cannot be provided to better highlight the proof of testing of this method, during the final defence of this assignment a demonstration can be provided as proof of testing of the biometrics check.