

TI Design Project
Faculty of Electrical Engineering, Mathematics and Computer Science

Design Report

ZGT: Monitoring of in-hospital patients with diabetes
through continuous glucose monitoring devices

April 17, 2024

Group 4:

Petras Baublys
Ciprian Bica
Stefan van Gurp
Alexandra Murgea
Maria Petrova
Daniel Stanchev

Supervisor:

Nikolaos Alachiotis

Table of Contents

Table of Contents	1
1. Introduction	3
1.1 Problem & Context	3
2. System specification	4
2.1 Stakeholders	4
2.2 Requirements specification	4
2.2.1 Functional requirements	5
2.2.2 Hardware requirements	5
2.2.3 Quality requirements	5
2.2.4 Requirements update	5
2.3 Requirements prioritisation	6
2.3.1 The MoSCoW classification	6
3. Project proposal	7
3.1 Implementation trajectory	8
3.2 Testing strategy	9
3.2.1 Unit Tests	9
3.2.2 Integration Tests	9
3.2.3 User Tests	10
3.2.4 Stress Tests	10
3.3 Risk management	10
3.3.1 Hardware Risks	10
3.3.2 Software Risks	11
3.3.3 Risks Summary	11
4. Planning & management	12
4.1 Methodology	12
4.1.1 The Agile & Scrum approach	12
4.1.2 Communication, Sprints & Tasks division	12
4.2 Time planning	13
5. Design choices	15
5.1 System design	15
5.2 Hardware architecture	16
5.3 Front-end design	17
5.4 Server architecture	18
5.5 Database design	18
6. Implementation	19
6.1 Hardware prototype(s)	20

6.1.2 Battery	20
6.1.3 RFID reader	21
6.1.4 Case	22
6.2 GUI transformations	23
6.3 Device configuration	26
6.4 Information transfer	27
6.4.1 GUI to Server:	27
6.4.2 GUI to SATO BANI:	27
6.4.3 SATO BANI to Server:	28
6.5 Database evolution	28
7. Evaluation	29
7.1 Functionality evaluation	29
7.2 User experience	29
7.3 Security evaluation	29
7.4 Scalability	29
7.5 Battery reliability	30
8. Conclusions	30
9. Follow up	31
10. Contributions	32
Resources	32

1. Introduction

This report aims to present an overview of the lifecycle phases involved in the design of a proof of concept prototype intended for the medical sector, more precisely related to continuous glucose monitoring devices in patients with diabetes. The entire process was executed in close collaboration with our client, the ZGT hospital and our supervisor from the University of Twente as part of one of the final graduation modules. Therefore, in order to showcase the knowledge and skills accumulated so far, the following sections will treat all the relevant steps involved in the development process: from requirements elicitation, design choices, prototype development and testing to other aspects related to team management, communication, results and discussions.

1.1 Problem & Context

Diabetes is a long-term medical disorder which affects more and more people every year, being ranked as one of the most common causes of mortality in many developed and newly industrialised countries (Kakraniya et al., 2024). The patients manifest elevated glucose levels which are often caused by either a deficiency in the insulin production (type I diabetes) or a certain degree of insulin resistance (type II diabetes) (Tao et al., 2015). Over the years, multiple efforts have been made in order to prevent the life-threatening complications associated with this disease. Among those innovations we can name non-invasive glucose monitoring devices (NGM) and continuous glucose monitoring systems (CGMS), two essential tools designed to offer a convenient way of checking one's glucose levels during the day with the purpose of ensuring the values stay within the normal range (Vashist, 2013).

Our client, ZGT (Ziekenhuisgroep Twente) is a Clinical Hospital based in two locations: Almelo and Hengelo. They are specialised in complex diabetes, oncological care and obesity and are continuously improving themselves through participation in medical research alongside other hospitals (Over ZGT, n.d.). During our first meeting, they brought to our attention that a large number of their patients with diabetes have adopted these devices, in particular CGMSs like the Freestyle Libre which uses a small sensor that gets applied to the skin to measure the glucose level in a real-time manner via the insertion of a small needle in the adipose tissue. These readings become accessible to the patients via an app and based on the results they adjust their insulin intake.

They also mentioned that although these devices are of great help to the patients in their day-to-day life, they pose multiple limitations for the healthcare providers given their lack of an universal platform or API that would give the doctors access to the raw-data glucose readings. Currently, our client's only data is collected from outpatients which come to the hospital for different concerns other than diabetes. Given the previously mentioned drawbacks of CGM, during those visits, the medical staff has to resort to the traditional method of manually inserting a needle in the finger 3-7 times a day to ensure that the treatment will not negatively affect the patients.

As one can imagine, this approach is rather inefficient. In this sense, ZGT aims to provide the patients that come to their hospitals with a CGM with the condition of finding a way to transfer those real-time readings from the devices to their main server in a safe and secure manner to protect their patients' privacy.

2. System specification

In the following section we will dive deeper into the problem by addressing aspects related to identifying the main stakeholders as well as exploring their expectations and needs for the system.

2.1 Stakeholders

As the name suggests, the stakeholders are the people whose interest is at stake with regards to our project and whose feedback and requests are of great importance. Through discussions among the team members we came up with the following list of stakeholders which we considered relevant:

- Diabetic patients who are admitted in the hospital and are using the glucose patches
- ZGT Hospitals
 - Hospitals' personnel who take care of the patients and helps them set up our system with their glucose patches
 - Hospitals' developers who intend to use the data from the glucose patches for their own purposes
 - Hospitals' doctors who will have an easy and real-time way to track their patients' glucose levels
- Our supervisor, Nikolaos Alachiotis
- University of Twente
- Other hospitals involved in the research
- Us, the development team

2.2 Requirements specification

Requirements engineering is essential in the development cycle as it is the first step in forming a strong foundation for all further collaboration between the client and the team. During these phases multiple activities take place in which the stakeholders express their requirements which get further analysed and discussed in order to treat issues related to prioritisation, feasibility and validation (Hudaib et al., 2018).

The following list of requirements resulted from an inspection of the project description offered by ZGT in addition to a further meeting in which the requirements elicitation took place. Here we make a distinction between functional, quality and hardware requirements.

2.2.1 Functional requirements

- The solution should be able to read the glucose levels continuously through a sensor.
- The system should be able to immediately process and store the glucose readings.
- The solution should be able to transfer the readings to the server.
- The system should allow access to the most recent readings.
- The system should allow data visualisation to the healthcare providers.
- The GUI should be able to communicate with our server.
- The GUI's interface should offer the ability to set specific WIFI settings.
- The GUI's interface should offer the ability to enter a patient's identification.
- Our server should receive patient's data from the Research Servers.
- Our server should be able to send the patient's glucose readings to the Research Server.
- Our database should store information about the patients and data about their glucose readings.

2.2.2 Hardware requirements

- The SATO BANI should be able to run on a battery.
- The SATO BANI should be able to charge the battery through a USB C port.design
- The SATO BANI should be able to read data from the sensor through NFC/Bluetooth.
- The SATO BANI should be able to be configured through a GUI.
- The SATO BANI should be able to communicate with our server through WIFI.

2.2.3 Quality requirements

- The solution should be compatible with the CGM devices provided by ZGT.
- The solution should operate entirely within the hospital environment.
- The solution should NOT rely on cloud applications.
- The solution should be scalable without compromising performance.
- The system should update the data with a delay of maximum 5 minutes.
- The data should be stored in a readable format in the database.
- The backend should be dockerized.
- The system should contain a security protocol to ensure data confidentiality and integrity during the entire process.
- The data should be encrypted during transmission and storage.
- The system should adhere to privacy regulations.
- The SATO BANI should be able to extract and transfer data for 24 hours without being charged.
- The SATO BANI device should be as small as possible.

2.2.4 Requirements update

During the sprint meeting of Week 8, the client communicated that there had been a change in requirements, and that we would no longer be required to connect to the hospital database and send the data there, as the process turned out to be quite involved and would fall outside the scope of the project. A more detailed motivation for this change will be outlined in the

implementation section (see Section 6) of this report. The requirements which ended up being dropped are the following:

- The solution should be able to read the glucose levels continuously through a sensor.
- Our server should receive patient's data from the Research Servers.
- Our server should be able to send the patient's glucose readings to the Research Server.
- The SATO BANI should be able to read data from the sensor through NFC/Bluetooth.
- The solution should be compatible with the CGM devices provided by ZGT.
- The solution should operate entirely within the hospital environment.
- The backend should be dockerized.

2.3 Requirements prioritisation

Oftentimes, the discussions with the client uncover a large number of requirements which differ in terms of duration, expenses, stakeholders' value, risks, etc. and therefore need to be carefully analysed and prioritised for a smooth execution of the product (Asghar et al., 2017). By identifying the most important functionalities we benefit from getting incremental and timely feedback which helps solve scheduling problems, correcting errors or clearing misunderstandings early on, improving our chances of delivering satisfactory results (Hudaib et al., 2018).

2.3.1 The MoSCoW classification

One of the simplest and most popular prioritisation techniques is called MoSCoW. This name of the method consists of abbreviations for the four priority categories proposed (Must, Should, Could and Won't) which get depicted below along with a short description according to Kukhnavets and Kukhnavets (2022) :

Must

This category contains the requirements with the highest priority which are mandatory for the successful completion of the project's scope.

- The solution should be able to read the glucose levels continuously through a sensor.
- The system should be able to immediately process and store the glucose readings.
- The solution should be able to transfer the readings to the server.
- The system should allow access to the most recent readings.
- The SATO BANI should be able to run on a battery.
- The SATO BANI should be able to charge the battery through a USB C port.
- The SATO BANI should be able to read data from the sensor through NFC/Bluetooth.
- The SATO BANI should be able to be configured through a GUI.
- The SATO BANI should be able to communicate with our server through WIFI.
- The GUI should be able to communicate with our server.
- The GUI's interface should offer the ability to set specific WIFI settings.
- The GUI's interface should offer the ability to enter a patient's identification.
- Our database should store information about the patients and data about their glucose readings.

- The solution should be compatible with the CGM devices provided by ZGT.
- The solution should operate entirely within the hospital environment.
- The solution should NOT rely on cloud applications.
- The system should update the data with a delay of maximum 5 minutes.
- The data should be stored in a readable format in the database.
- The system should contain a security protocol to ensure data confidentiality and integrity during the entire process.
- The data should be encrypted during transmission and storage.
- The system should adhere to privacy regulations.
- Our server should receive patient's data from the Research Servers.
- Our server should be able to send the patient's glucose readings to the Research Server.

Should

These requirements are still really important to the client however they are not crucial to the system's execution.

- The SATO BANI should be able to extract and transfer data for 24 hours without being charged.
- The SATO BANI device should be as small as possible.

Could

They reflect wishes or bonus features whose implementation takes a low priority or are even excluded based on time/resources constraints.

- The system should allow data visualisation to the healthcare providers.
- The solution should be scalable without compromising performance.
- The backend should be dockerized.

Won't

At last, we have functionalities that will not be part of the final product. They serve as a bound for the client's level of expectation.

- The system will not be directly integrated into the hospital system.
- The system will not be communicating with the hospital's main database.
- The final product will not include a dashboard for the glucose readings.

3. Project proposal

After acquiring a better understanding of the problem domain and analysing the needs and desires of our client which resulted from the requirements elicitation phase (see Section 2), we started looking into different alternatives that would make these concepts take shape. In this section we discuss our proposed ideas for the system as well as the considerations behind them. In addition, we will treat some of the potential risks and mitigation strategies that were identified during the decision process along with a testing strategy.

3.1 Implementation trajectory

We are aiming to provide ZGT with a prototype for a continuous glucose reading system which will serve the in-hospital patients and provide access to meaningful data to the medical personnel. Our initial proposal depicts a system composed of 4 individual components which are discussed in more detail at the end of this subsection.

To offer a general overview of our vision, one of those components will be an embedded system device called “*Sato Bani*” that is going to be attached or carried alongside the patient to read the glucose readings out of a CGM device. During our development and testing we are going to be using the Freestyle Libre 2 CGM that was provided by our clients. The other three components are going to be software solutions: the Sato Bani configuration GUI, a server that orchestrates the whole system and a database, to hold patient data alongside the glucose readings.

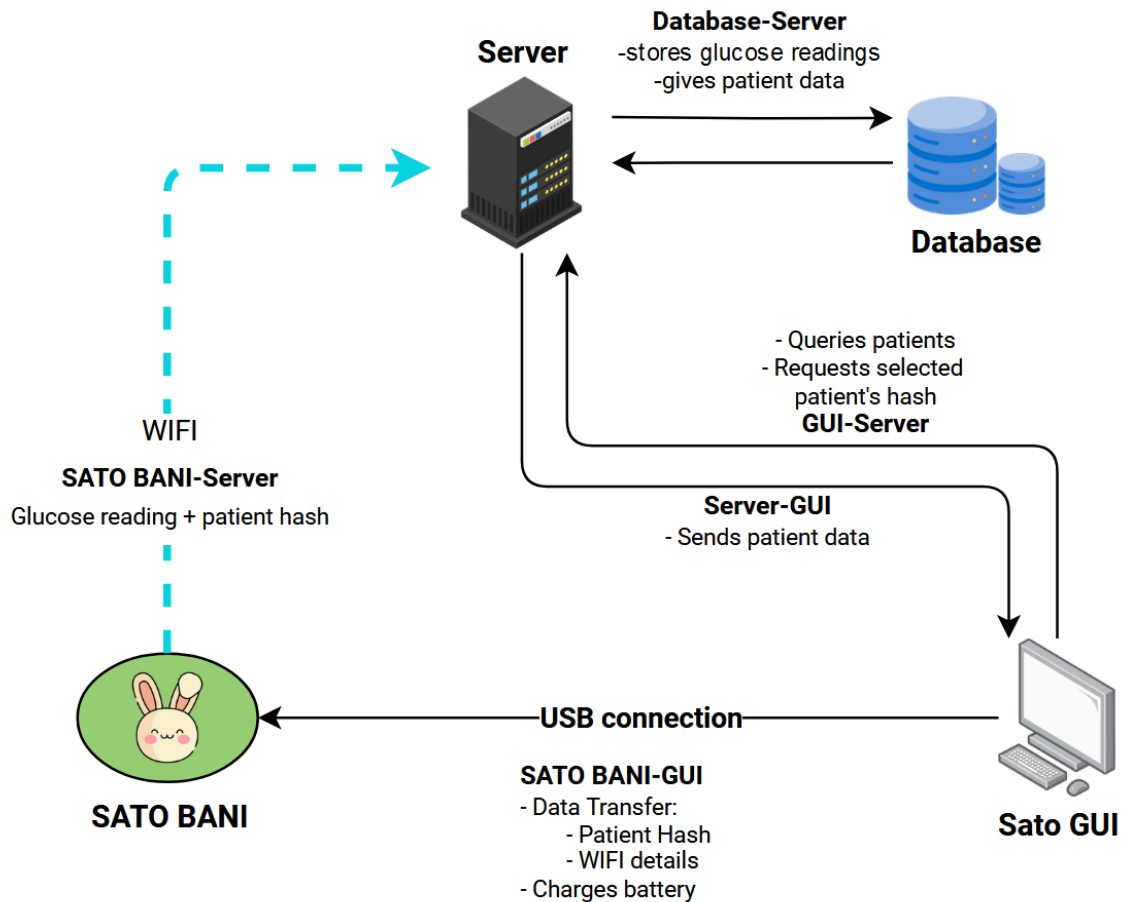


Figure 3.1: System workflow

Individual component descriptions:

Sato Bani – The hardware component responsible for reading the data from the patients GCM and sending it to the server over wifi. In the current design it consists of an ESP32-C3, a 250mAh single cell lipo battery and a mini RFID RC-522 reader. All the components were

chosen with the purpose of reducing the size of the device as much as possible. The ESP32-C3 enables wifi and bluetooth connectivity and has a footprint of only 17x21mm, which we consider the best choice for us. A further minimization which we explored was an RFID board that would allow us to connect an external antenna to further reduce the device size, but we deemed it outside the scope of this project.

GUI – A website built on Electron to run as a desktop application with Node.js. The choice of using Electron was a simple one as we needed access to usb ports to configure the Sato Bani device and websites do not provide that level of functionality. Electron allows us to transfer our web development skills to a desktop environment and to access usb ports for the Sato Bani's configuration.

Server – Dockerised Python Flask backend. Dockerization was one of the clients requirements and python was chosen for its simplicity and quick development. Even if python sacrifices some server speed and optimisation it's a good tradeoff for the gain in speed of development.

Database – MySQL, same type as the hospital's server to ensure consistency and compatibility. The MySQL server is encapsulated inside a docker container.

3.2 Testing strategy

Testing is an essential step in the development process, meant to evaluate the product for possible errors or points of improvement. Therefore, this process requires careful considerations and planning in order to gather meaningful insights on the functionality and performance.

For this purpose, our strategy implied using the Python "unittest" library for conducting unit tests on the server, as well as on the database. Regarding the rest of the components, especially the GUI, we carried out manual testing during the development in order to inspect the components' integration and their behaviour under stress.

3.2.1 Unit Tests

Unit tests evaluate a specific function within our server infrastructure. Our aim is to build several tests for the server that ensure that all endpoints of our server are functional, and that the server behaves correctly. All test files will create and run an instance of the server when it is executed. The procedure will include checking how the server handles both expected and unexpected inputs and requests like invalid credential, wrong types or empty data. In addition, we will be testing the core functionalities like: login, patient search, glucose readings update and some extra features related to the display of the search results. Unit tests will also be employed to test the database, with a focus on the capacity and performance.

3.2.2 Integration Tests

Integration tests intend to verify the functionality of several functions working together. We would like to perform manual input tests, which entail running aspects of the application ourselves, to ensure the correctness of the system. We would like to perform integration tests on the frontend

and backend communication, for example logging in a user. Also, we would like to ensure that the server and database communication works as expected, for example logging in a dummy (fake) user or inputting some dummy data. We would like to note that these tests will be using the functions that have been already tested through unit tests. We perform integration tests to find any bugs or edge cases, that is cases outside the norm, for example if some data was still encrypted in the database given that should not be the case.

3.2.3 User Tests

With permission from the client, we would like to ask a staff member to perform tasks such as: logging in, configuring the wifi, inputting patient details and configuring the Sato Bani. This would allow us to spot any flaws in our design of the frontend and allow us to identify problems from the perspective of a stakeholder.

3.2.4 Stress Tests

As we are unaware of how long our battery will last, we intend to run the device for as long as possible and time how long it takes for the battery of the Sato Bani to run out. In addition, as one of our requirements was that the system we create should be scalable, we will create some stress tests for the server, to see if it can handle multiple devices being connected at the same time. This will be achieved by simulating an adequately large number (100) of devices and making them all connect and send readings to the server in a short interval of time, to simulate an extreme use case of the system.

3.3 Risk management

During the brainstorming and design of our solution some concerns were raised regarding potential malfunctions of the system. Therefore, we decided to acknowledge these problems and prepare mitigation strategies to ameliorate these risks.

3.3.1 Hardware Risks

Regarding the hardware components the most notable problem would be the battery life. If the device is active constantly, it is likely that the battery will run out in a short amount of time. Therefore we plan to make the battery rechargeable via a USB connection after which the Sato Bani device should be reconfigured via the GUI.

The device could also pose a physical danger to people and objects nearby. Our choice of a Lithium Polymer battery does make our device rather volatile and can catch fire if the battery is misused or mismanaged. This problem will be addressed with three different measures. Firstly the battery will be placed inside a plastic case to protect it from punctures. Secondly we will be monitoring the voltage of the device's battery and turn off the device when it gets too low. Thirdly, users of the device will be instructed to inform the hospital's staff if the device is beginning to bulge, meaning that the battery has to be changed.

3.3.2 Software Risks

As our software is a web based application, it is prone to threats such as SQL injection, JavaScript Injection and man-in-the-middle (MITM) attacks.

For some context, injection attacks describe the situations where malicious users input code into the application in an attempt to break it or retrieve information that should be kept private. This can be done with SQL, which is known as an SQL injection, or with JavaScript, a JavaScript injection, both of which we plan to mitigate by sanitising the user input. This method implies that special characters that may be an indicator of an injection attack such as ' or " will be replaced with another string of characters. For example, ' could be replaced with ''. Consequently, if there is a malicious user attempting to inject code, their code will not take effect and break when trying to run.

On another note, to mitigate MITM attacks, we plan to use encryption algorithms on the data, particularly when we need to send the readings from the device. We will encrypt this data before it gets sent and only decrypt it before inserting it into the database. We intended to have the data encrypted in the database too, but at the request of the client we will keep the database in plain text format for the purpose of readability.

3.3.3 Risks Summary

Here we have summarised the risks involved and given each a level of severity if this occurs and a probability of this risk happening and how we intend to mitigate these risks as described above.

Risk	Probability of happening	Level of severity	Mitigation
SQL Injection	Medium	High	Sanitise all user input going to the database
MITM attack	Low	High	Encrypting data being sent between components e.g. frontend and backend
Battery life	High	Low	Rechargeable battery
JavaScript injection	Medium	Medium	Sanitise all user input
Fire hazard	Low	Medium	Care for battery life
Electrocution	Low	Low	Encasing Sato Bani so wires are not exposed

4. Planning & management

With a fixed goal in mind we are moving onto creating a comprehensive plan to achieve desirable results. The following section will outline aspects related to project management techniques, time planning and communication which will be pivotal to the project's execution.

4.1 Methodology

New projects emerge at a rapid pace, each with unique scopes and restrictions making it impossible to design a project management methodology that would account for all individual and diverse needs and challenges. Multiple models have been adopted during the years based on the particular strengths that these methods pose.

4.1.1 The Agile & Scrum approach

With the progression of technology, the traditional project management methodologies started encountering difficulties in accommodating the rapid change in requirements, leading to the emergence of a new solution, known as Agile, which excels in flexibility, adaptability and speed (Merzouk et al., 2018). By making a parallel between Agile and Traditional we observe an emphasis on: people and their interactions over procedures and equipment, functional software over thorough documentation, client cooperation over contract negotiations, and adaptability over a fixed schedule (Daraojimba et al., 2024).

There are multiple methodologies that adhere to the "agile" style like Scrum, Extreme Programming and Kanban which strive for the realisation of continuous increments in short iteration called sprints (Daraojimba et al., 2024). The advantage of such divisions consists in the efficiency of receiving timely feedback which means that potential misunderstandings or errors are solved early on, hence being less expensive from the point of view of reallocating resources like time, money, workforce and others.

Our team decided to adopt the Agile approach during the entire process of this project, in particular the Scrum methodology. Apart from the already mentioned benefits we can name a few more reasons for this choice like: the presence of stand-up meetings and sprint reviews which we deemed essential for ensuring a close collaboration both within the team and with the client and supervisor, increased flexibility so that the final prototype could encompass the client's emerging and changing needs and the idea of continuously producing tangible results meant to balance our tight deadline and maximise the chance of delivering a working prototype.

4.1.2 Communication, Sprints & Tasks division

In our case, the execution of some of the Scrum and Agile methodologies differed to some extent from the general tendency. To be more precise, in most cases project that follow the Agile method organise sprints of 2-4 weeks and daily stand-ups, however we choose to have a stand-up approximately every other day (Monday, Wednesday and Friday), as well as two weekly sprint meetings, one with the client (on Fridays), and one with the supervisor (on Wednesdays). These choices came as a result of the team members' availability as well as the

need to ensure that we are on the right track from the beginning in order to make the most out of the available time. During the stand-ups we discuss our findings, accomplishments and struggles. As a result, we either offer more support to the people who are behind or assign new tasks to the ones that are done. The changes and progress get monitored via a Trello board.

In order to divide the workload, we generally discuss what tasks should be completed that specific week and each team member picks the ones they think are best suited or most preferred. Another strategy worth-mentioning, that worked really well for us, was splitting the team into two groups, one with a focus on hardware (components, wiring, data transfer methods and protocols) and one on software (server, desktop app, database). This approach turned out to be really effective and allowed everyone to work on the area they find most interesting keeping everyone motivated. In addition, there are two weekly tasks where almost everyone is involved, which is updating the main project report and documenting their progress in a separate document.

In order for this system to work, communication was essential, as we needed to keep everyone updated and maintain a strong relationship between the team, the client and the supervisor. For discussions or stand-ups within the team we used: Whatsapp, Discord, Email and Trello whereas for meetings with the supervisor or sprint meetings with the client we opted for Microsoft Teams and Email. Lastly, all meetings are scheduled effectively based on the availability expressed by each member in a shared Google Calendars.

4.2 Time planning

In regards to this project we were allocated around ten weeks for the completion of a working prototype. Because of the short timeframe it was essential for our team to establish a comprehensive plan that would outline our goals and expectations for each of the sprints as well as provide a way of checking whether or not we were on track. A table view can be found below, reflecting our intentions regarding the execution of the project:

Sprint	Start date	End date	Goal	Expected results
1	05/02/24	11/02/24	Establishing contact & Planning	<ul style="list-style-type: none"> - team building activity - first meeting with the client - getting to know the context and problem - defining project structure - time planning & team management
2	12/02/24	18/02/24	Requirements elicitation, Research & Brainstorming	<ul style="list-style-type: none"> - requirements elicitation & prioritisation - finding a supervisor - initial project proposal - writing the testing strategy

				<ul style="list-style-type: none"> - research hardware, docker & frontend options - initial design for the device, frontend
3	19/02/24	25/02/24	Design choices, Database creation & Familiarisation with the components	<ul style="list-style-type: none"> - creation of necessary diagrams - deciding on design & architecture of the system - database design - internet configuration of the device - exploring ESP32-C3 usb connection - research into flask - soldering components - stub implementation for the backend - research into bluetooth options
4	26/02/24	03/03/24	Start of development	<ul style="list-style-type: none"> - server creation - transfer protocol creation - database creation - unit testing - familiarisation with Electron & Node.js - start on the desktop app
5	04/03/24	10/03/24	Development	<ul style="list-style-type: none"> - completion of unfinished parts from sprint 4 - desktop app page 1 - desktop app page 2 - 06/03/24 deadline group report plan
6	11/03/24	17/03/24	Security protocol	<ul style="list-style-type: none"> - implement encryption methods - unit testing - backend dockerized - desktop app page 3 - start writing on the reflection report - 14/03/24 deadline individual reflection

7	18/03/24	24/03/24	Integration of components	<ul style="list-style-type: none"> - integration of components - integration testing - 21/03/24 deadline final group reflection report
8	25/03/24	31/03/24	More in-depth testing	<ul style="list-style-type: none"> - user testing - stress testing
9	01/04/24	07/04/24	Review project	<ul style="list-style-type: none"> - complete source code - finalise report
10	08/04/24	14/04/24	Preparations for the presentation	<ul style="list-style-type: none"> - prepare slides - 08/04/24 deadline individual reflection - 19/04/24 final presentation

5. Design choices

The same idea can be represented in various ways, hence numerous factors need to be taken into consideration prior to the development stage given these decisions will directly impact the development time, quality and visuals of our final product. In this section, we will discuss the design choices we made with regards to the components of our system as well as some of the reasons behind them.

5.1 System design

In order to illustrate the intended succession of steps involved in the process of transmitting the glucose readings from the CGMs to the server we decided to use a sequence diagram. As it can be seen in Figure 5.1, we opted to make a distinction between the four components of the Sato Bani system so that the interactions between them, as well as the ones with the users are clearly depicted.

To offer a short description, the staff member needs to log in via the GUI, requesting access which gets granted after there is a match with the database credentials. As a next step, they need to connect the Sato Bani device to the GUI in order to configure the network details and search for the respective patient so that the glucose readings will be associated with the right person. At this step the device will transmit the data to the server and successively to the database as long as the device has enough battery.

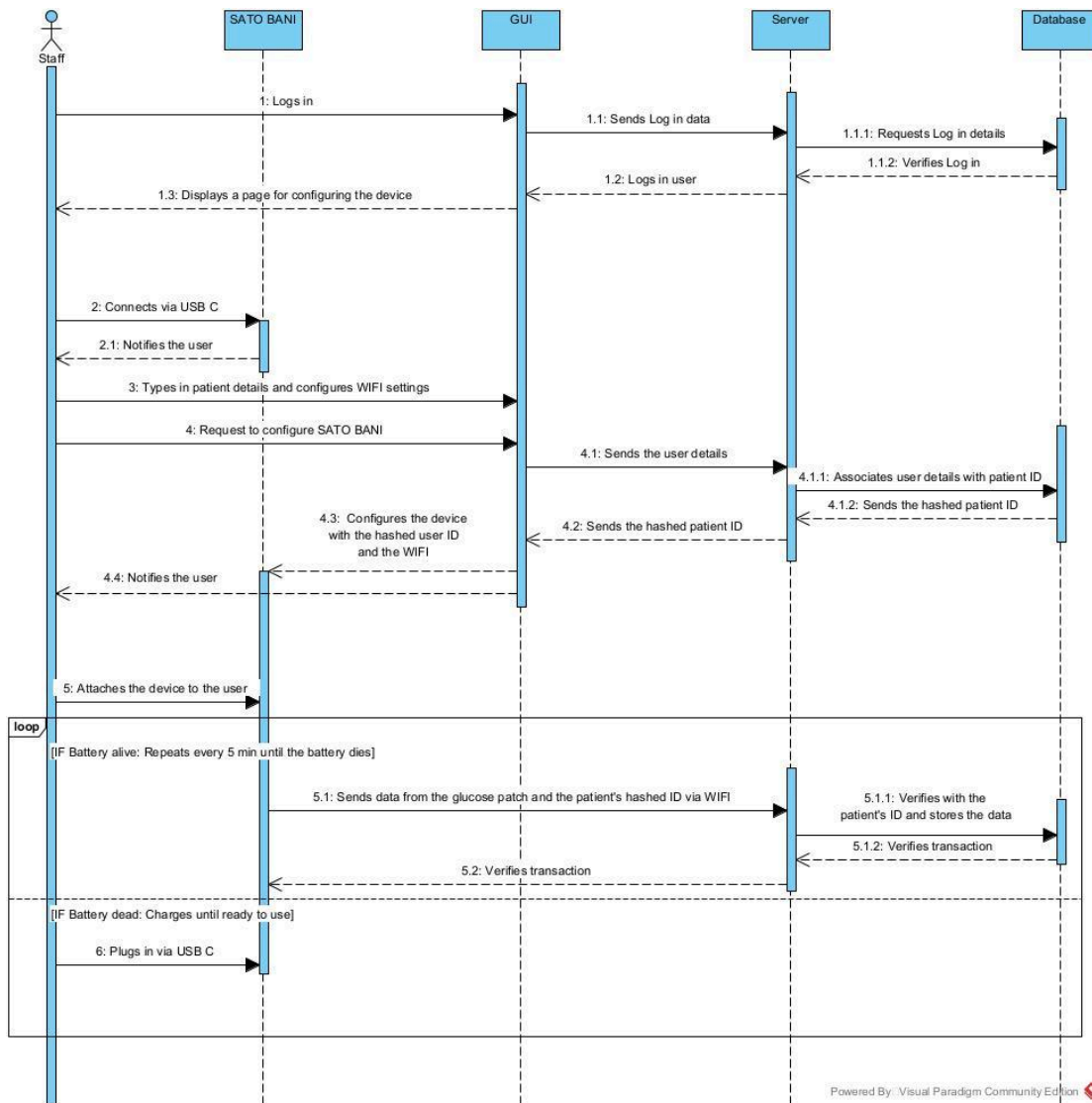


Figure 5.1: Sequence diagram of data transmission

5.2 Hardware architecture

Designing the hardware was not an easy process as it required a rather thorough understanding of the system we are expected to deliver as well as comprehensive research. In the end, our team has settled on the following components:

- An ESP32-C3
- A PN5180-NFC board
- A LIPO 250 mAh battery

To offer some context, the ESP32-C3 is going to handle all the communication between the glucose sensor and the SATO server. Therefore, this board supporting both WIFI and BLE, turned out to have the perfect combination of features for this project. Another consideration that

we took into account was the size of the board, as we aimed for it to be as small as possible. This requirement was guided by our approach of designing a system which is small enough to be portable and easily attachable to a patient. As per the appearance and structural needs of the Sato Bani we decided to use a 3D printed enclosure, as it is one of the cheapest, lightest and most accessible methods of creating it. Originally we wanted to use the RFID reader to communicate with the glucose sensor and receive blood sugar readings. Unfortunately due to time constraints and subpar support for the PN5180-NFC we could not implement that functionality. To showcase how the prototype works despite this setback we implemented a program that simulates a data stream of glucose readings.

5.3 Front-end design

Regarding the frontend we have looked into different alternatives before settling on using ElectronJS, a framework that allows us to convert standard website code (HTML, CSS and JavaScript) to desktop applications. This was done to give us more control over usb ports on the computer and to make it easy to use for the hospital staff. Furthermore, ElectronJS was chosen over other GUI options because our whole team is well educated on website development which means our skills easily translate to ElectronJS.

In addition, we have all agreed that the most important themes for the desktop application are simplicity and usability. Doctors and nurses are going to be using the system to register patients to the continuous glucose monitoring system. Reducing the time spent on our system, means making them more efficient and allowing them to treat more patients. For our initial GUI design (see Figure 5.2) we used Figma to sketch the three pages which we deemed relevant for the purpose of: ensuring only authorised personnel has access, choosing the right patient for the data collecting and setting the network details.



Figure 5.2: Frontend initial design sketches

5.4 Server architecture

The server structure is rather simple as we planned to have three classes: Server, Security and Database. Naturally, in the Security class we aim to have all functions related to security which can be imported into the other two classes. Hopefully, this will make it easy to encrypt/decrypt data and sanitise any user input being received from the GUI. The Database class will be used to communicate with the MySQL database provided by the client. Lastly, the Server class will house the functions necessary to communicate with the GUI. For this scope, we decided to use Flask as it makes implementing the communication more convenient as we are already familiar with the logic Flask uses from previous modules. These classes as well as some of the functionalities and connections can be seen in the following class diagram:

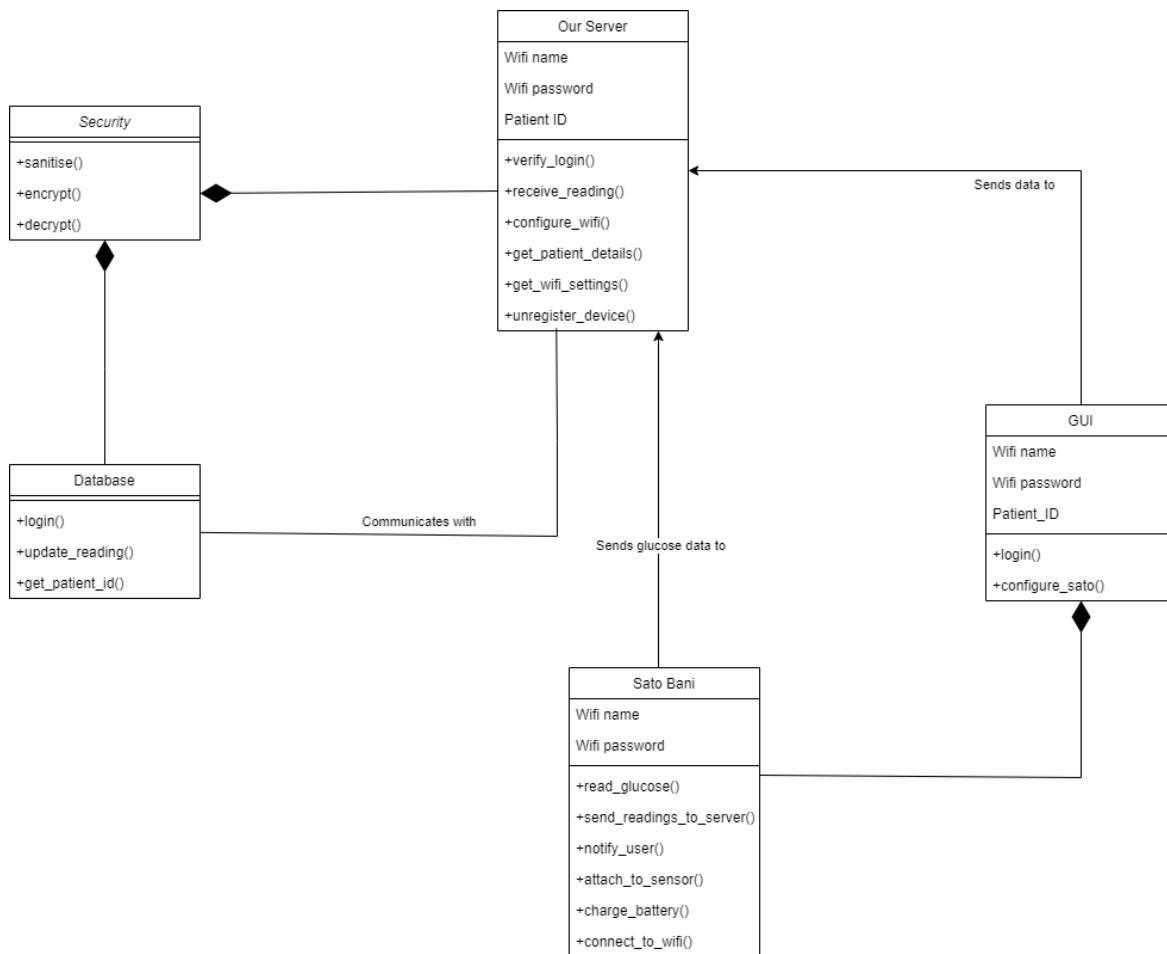


Figure 5.3: Class diagram of the system

5.5 Database design

From the description gathered from a dedicated discussion with the client we opted for a simple design whose main purpose is to offer the medical staff all the requested data in a readable manner. As illustrated in Figure 5.4, we will have two main tables for the patient's personal data and the glucose reading which will be transmitted by the Sato Bani device. This separation was

necessary for the purpose of avoiding data duplication. To offer a more detailed understanding, the two tables will be linked using the ZGT_id of each patient which also serves as a primary key for the Patient table. As for the readings they will be uniquely identified by a composite primary key using the ZGT_id and timestamp as no patient will be monitored twice from the same device at the same time. The data types were chosen to match the hospital's other tables which contain similar fields. In addition to these structures, there is currently one more table for the staff credentials for the purpose of testing which will be discussed with the client.

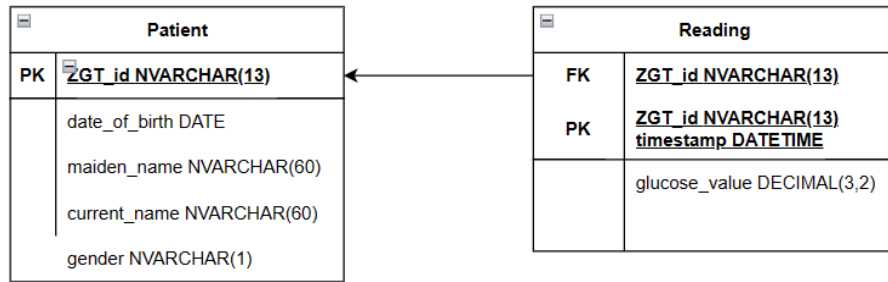


Figure 5.4: Database diagram

6. Implementation

In the following section, we will describe the evolution of the system's components, highlighting the main functionalities and the changes encountered. It is important to note that, during the implementation process, not only our initial designs regarding the server and front-end went through some modifications, but also the requirements. The main differences in requirements imply: the project will no longer be integrated with the hospital systems meaning that the dockerization step will also be dropped and the sensor to read the CGM device will not be connected.

Regarding the first change, this came from the client themselves. It took approximately seven weeks for us to obtain a functioning account in the hospital system, their explanation was that their bureaucracy is slow and their systems so complex it would take them half a year to make their systems compatible with our project. For further explanation, our project requires the use of docker, python and a database. The client has a system running docker services, another running their databases and they are in the process of creating a system to run python. All three of these systems currently can not interact with one another at the same time and will not be able to for the remainder of our project timeline.

Next, we had significant issues regarding the hardware component of our project. There were delays in the delivery of the PN5180-NFC board which only arrived in the later stages of the project. Despite this, we fully attempted to integrate it with our existing infrastructure, however, due to lack of information and the novelty of this board, there seem to be little to no answers in making this board work. The most helpful tool we could find was a PN5180 library¹, however,

¹ PN5180 library: <https://github.com/ATrappmann/PN5180-Library>

this was incomplete and was unable to communicate with the CGM device provided by the client. Therefore, in consultation with our supervisor, we simulated the functionality that the PN5180 should have done.

Furthermore, we decided not to implement session IDs for the server or GUI. Given that the client's systems are very complex, we felt that even if we had implemented this, it would have been extremely inconvenient for them to remove and rebuild if needed.

6.1 Hardware prototype(s)

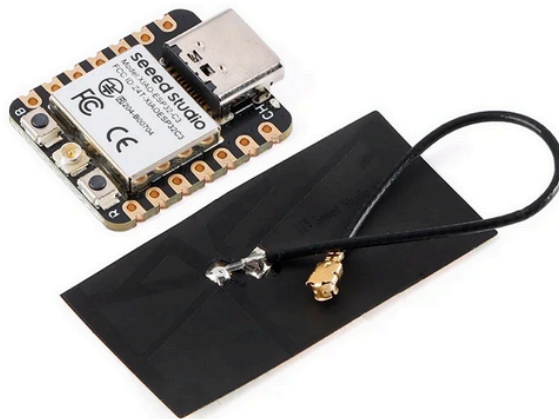


Figure 6.1: ESP32-C3

During the development stages of the SATO BANI the only constant was the ESP32-C3 board (see Figure 6.1) upon which we built on to make a more complete prototype. In the initial version, as we were attempting to implement the BLE connection between the FreeStyle Libre, there was no need for other components than the ESP32-C3 itself as it came with an attachable antenna that supports both BLE and wifi. As we pivoted development to the communication between the GUI and the SATO bani we only used the bare ESP32-C3 and its usb-c connection.

In order to start experimenting with the NFC protocols of the FreeStyle libre we attached a rc522 board to the ESP32-C3. Shortly after, it was discovered that our existing board did not support the needed NFC protocol to communicate with the Freestyle Libre (see section 6.1.3). When the new PN5180 boards arrived we connected them to our three ESP32-C3s and started extensive development. This was later stopped due to the limitations mentioned previously and the NFC readings decided to be simulated. One of the main requirements of the device was portability so we attached a battery to the ESP32-C3 (see section 6.1.2).

6.1.2 Battery

The battery turned out to be one of the easiest parts to accommodate as the ESP32-C3 has very good battery support and many inbuilt safety and utility features. For the battery itself we

have chosen the smallest model we could so that the footprint of the device is as small as possible. Keeping that in mind we ordered a 250mah 3.7v little one cell lipo battery. Because the original plan was to run the device 99% of the time in deep sleep mode, the small battery was adequate for our needs.

Battery management features:

- Low voltage protection (in-built into the ESP32-C3 and the battery circuit itself)
- Indication when charged (in-built into the ESP32-C3, the led on it stops shining)
- Overvoltage protection (in-built into the ESP32-C3, stop charging battery at about 4.1v)
- Voltage monitoring (wired and developed ourselves, so that we can send voltage readouts to the server live)

6.1.3 RFID reader

Initially, our prototype used a different board namely the RC522 mini. The board was chosen for its compact design, as we were trying to optimise the size of the Sato Bani. After wiring the components (see Figure 6.2), we realised the sensor and reader used different communication protocols (reader: ISO/IEC 14443, sensor: ISO/IEC15693). Due to this oversight, a new reader was ordered, the PN5180-NFC as it was said to support all NFC protocols (see Figure 6.3). Utilising the new component proved to be difficult within the remaining time. The library we found for the PN5180 had issues, many of which we did not have the expertise or time to fix. As such, we decided to try using an SPI library built into the Arduino IDE. However, despite the datasheets provided by the manufacturer on using the PN5180 without a library², we did not have the time or knowledge to successfully implement the NFC communication.

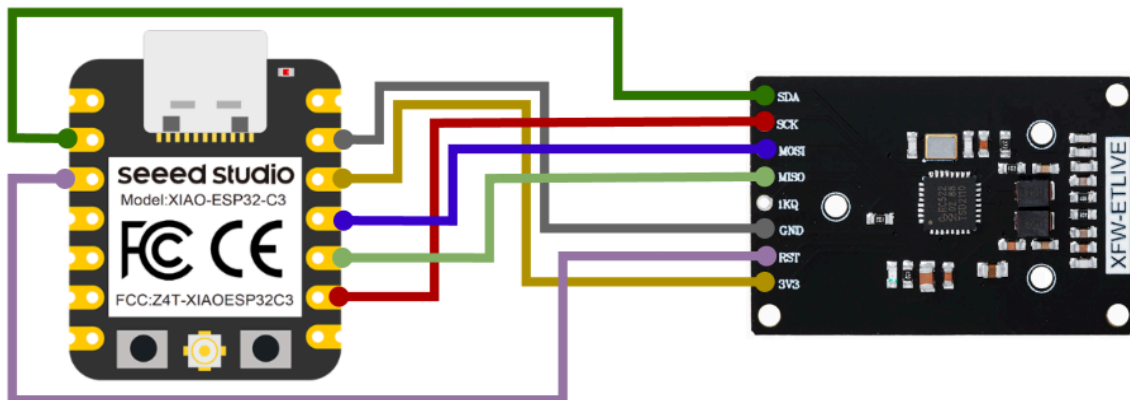


Figure 6.2: Connection between XIAO ESP32C3 and RC522 mini using SPI

² PN5180 datasheets:

<https://www.nxp.com.cn/docs/en/application-note/AN12650.pdf>

https://www.nxp.com/docs/en/data-sheet/PN5180A0XX_C3_C4.pdf

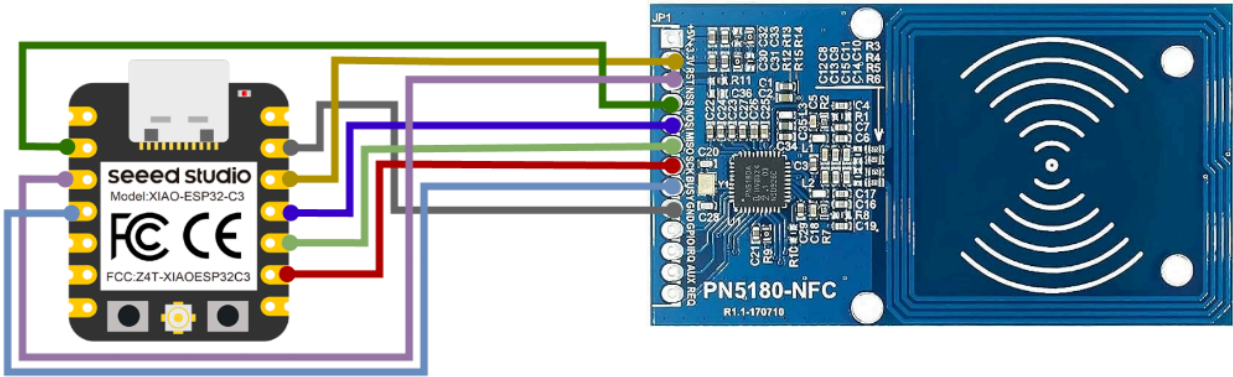


Figure 6.3: Connection between XIAO ESP32C3 and PN5180-NFC using SPI

6.1.4 Case

The case was designed in Fusion360, with the base of a dovetail box, to which we added handles on the side for the straps, an opening for the sensor, and a USB-C port to charge and connect to the ESP32. The first version we printed had some shortcomings: the opening for the sensor was too small, standard USB-C cables could not reach the ESP, and there was no mechanism to keep the lid in place. To address these shortcomings we expanded the sensor slot, made a groove on the inside of the USB-C port to allow the ESP to be pushed further into the wall, and added three small half spheres on the bottom of the lid that fit in three indents in the box to secure it (see Figure 6.4).

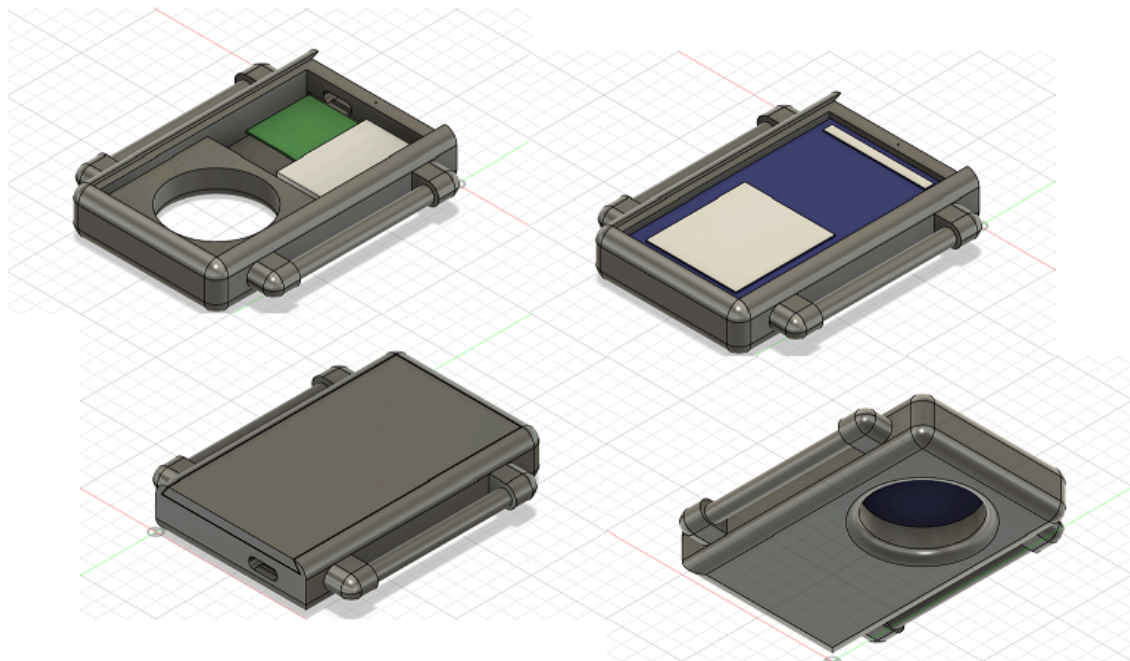


Figure 6.4: Final box design

6.2 GUI transformations

The creation of the GUI went through many stages, featuring ideas from both the team members and our clients (the hospital personnel). For the implementation of the technical parts we used: HTML, CSS, Bootstrap, Javascript and Electron JS. As for the creation of the illustration components, the team worked with Inkscape and FLATICON. Screenshots from the final version of the GUI have been included in this subsection alongside informative descriptions.

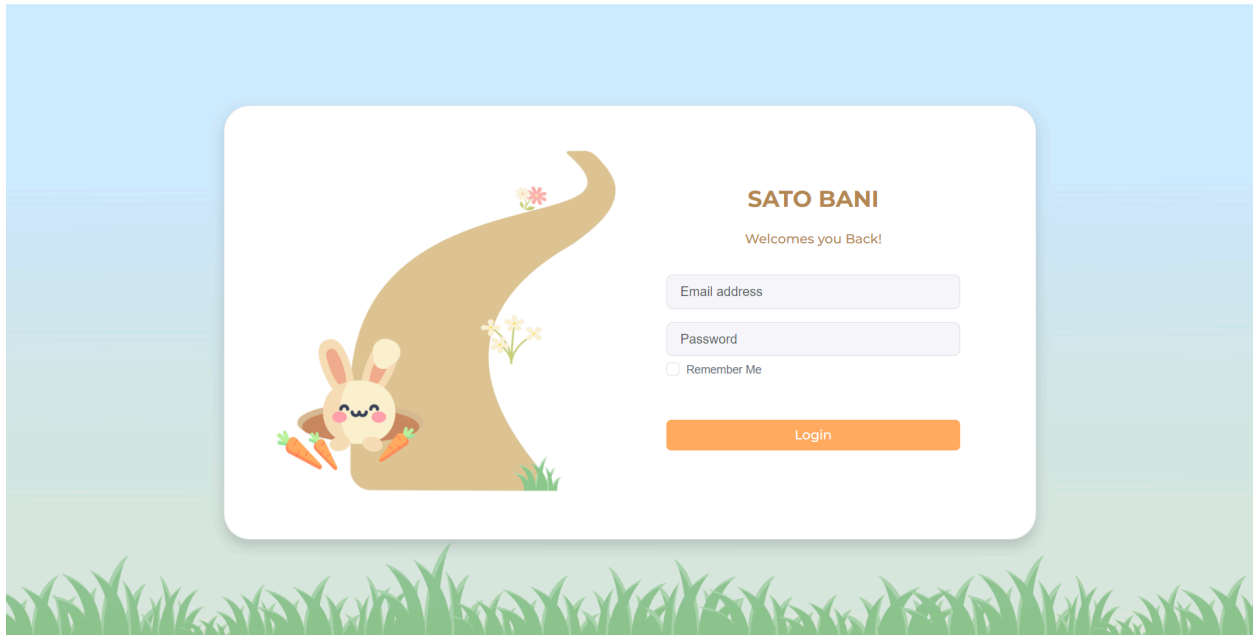


Figure 6.5: Login page

The first screen, illustrated in Figure 6.5 is designated to the “Login Page”. This page contains two input boxes in which the user can type their login credentials. To finalise the login process, the user has to press the login button, which can be seen in vibrant orange. In the scenario that the information is invalid an informative message appears. Furthermore, the user can choose whether they want to stay logged in even after the application is closed by selecting the “Remember me” box.

After the login was successful, the users are redirected to the “Home page”, depicted in Figure 6.6. On this page we give a short description of how to use the application. The instructions consist of four main steps, starting from “How to connect?” to the SATO BANI and guiding them through the whole procedure including how to select a patient and how to upload the details to the SATO BANI.

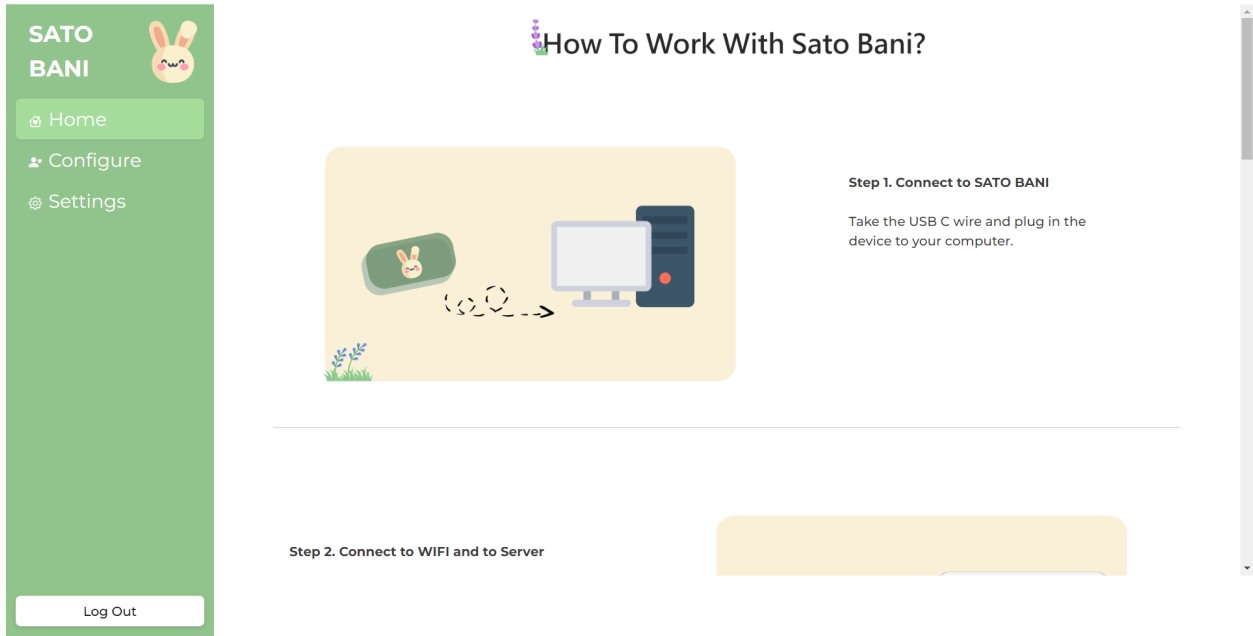


Figure 6.6: Home page

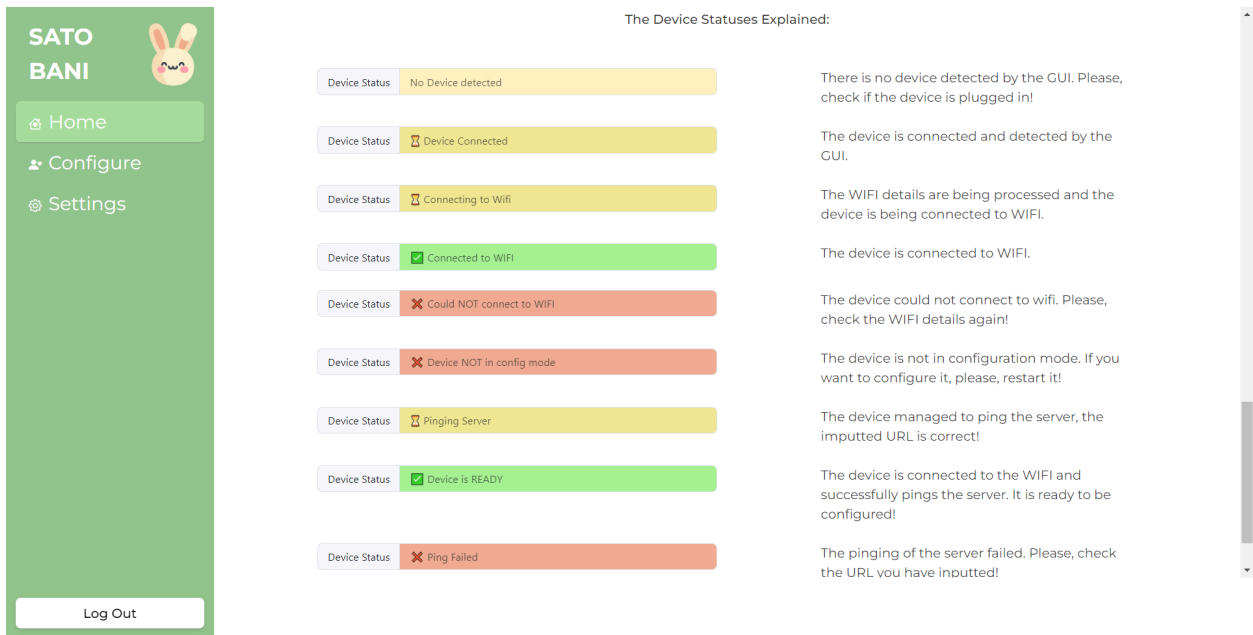


Figure 6.7: Device statuses

At this step, the user can navigate to the rest of the GUI by using the tabs on the left, namely the “Configure” and “Settings” buttons which connect to two other pages: the Configuration page (see Figure 6.8), and the Wifi Settings page (see Figure 6.10) respectively. In addition, by clicking the “Log Out” button, the user can log out of their account.

Furthermore, at the top right corner of the screen in Figure 6.8, there is a small window showing the current status of SATO BANI. Throughout the whole process, from connecting to the device, to uploading patient's data, the user can see the statuses of the device, which are also explained in the Home page (see Figure 6.7).

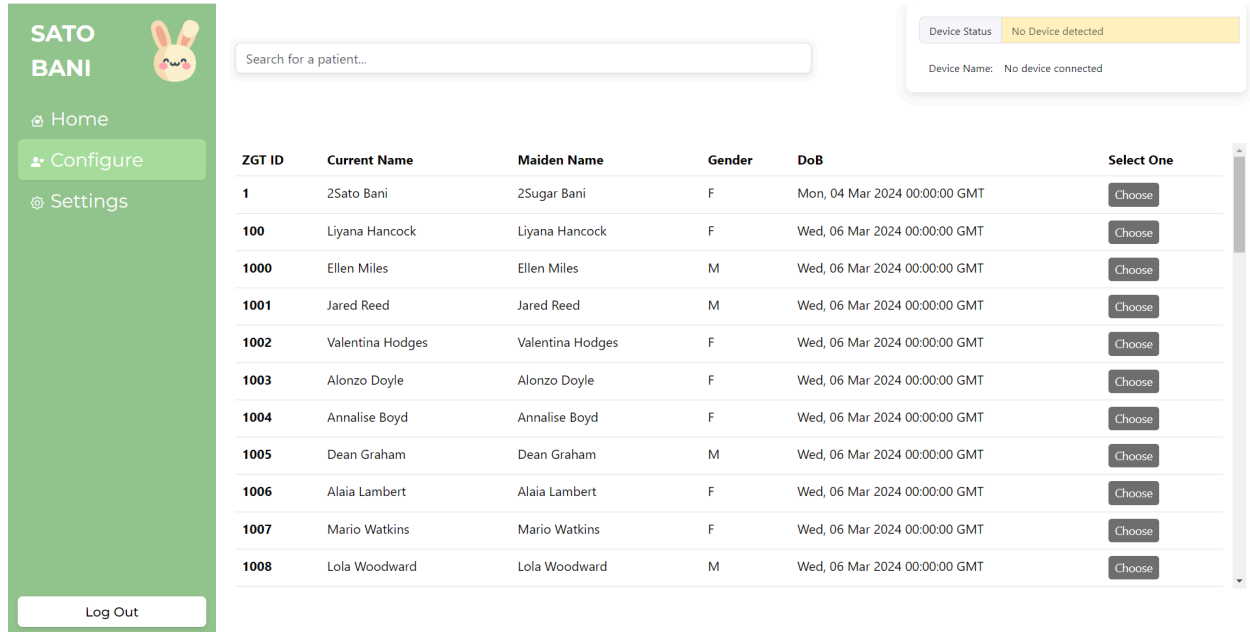


Figure 6.8: Configure page

The “Configure page” (see Figure 6.8) supports selecting a patient of choice and uploading their details to the SATO BANI. To select a patient, the user is first required to search for their name by using the search bar, located at the top of the page. According to the user's input, the table below shows all the possible results for the given search input. To select a patient, the user has to click on the “Choose” button for the associated patient. Once the device is ready to receive the patient's details, all the “Choose” buttons turn green, signifying that they can be pressed. To mitigate possible user mistakes, the “Choose” buttons lead to a pop-up, as shown in Figure 6.9, asking the user if the currently selected patient is their final choice.

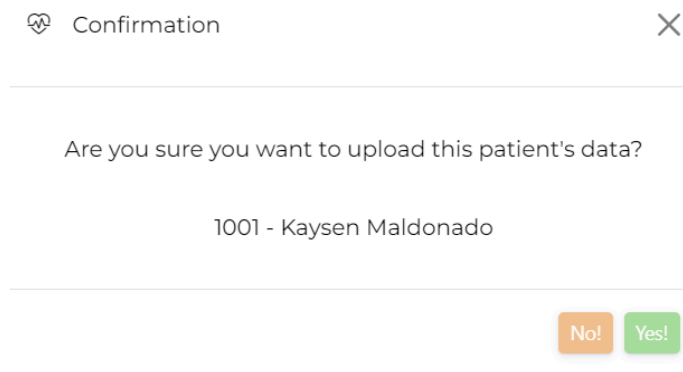
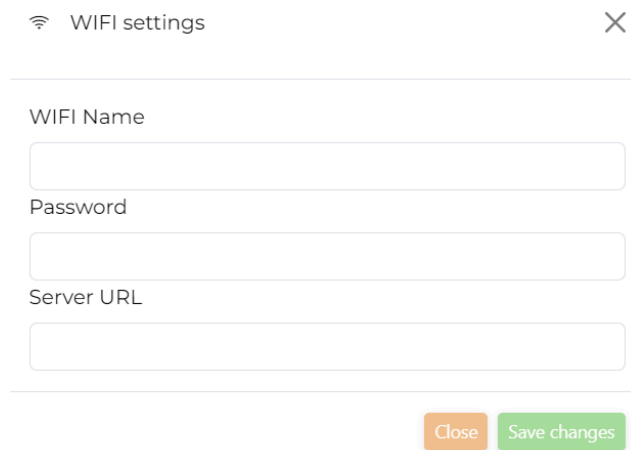


Figure 6.9: Confirmation pop-up

In the previous step, we mentioned the device being “ready”. This step is assigned to the “Settings” tab which triggers a pop-up asking the user for the: WIFI name, WIFI password and a Server URL, as showcased in Figure 6.10. These are the WIFI settings which the user has to provide to the SATO BANI so that it can send data to the Server.



The image shows a mobile application dialog box titled "WIFI settings". The dialog has a white background and a thin border. At the top left, there is a Wi-Fi icon followed by the text "WIFI settings". At the top right, there is a close button represented by an "X" icon. Below the title bar, there are three text input fields stacked vertically. The first field is labeled "WIFI Name", the second is labeled "Password", and the third is labeled "Server URL". At the bottom right of the dialog, there are two buttons: an orange button labeled "Close" and a green button labeled "Save changes".

Figure 6.10: Wifi settings

6.3 Device configuration

As mentioned previously, in order to configure the device, the user must first log in to ensure that only authorised personnel can access the system (see Figure 6.5). Once the user provides his details, a HTTP request is created in JavaScript using the fetch API to post the details to the server. The server will respond with either a true or false response if the details are correct. If a true response has been received, the user is presented with the “Home page” (see Figure 6.6).

Next, they can navigate to the configuration page as seen in Figure 6.8. When the page loads they will be shown a table like structure with the first fifty patients from the database. The user can then either search for a patient using the search box or scroll through the list.

If they choose to scroll through the list, when the end of the fifty selected patients is reached, the system will call a JavaScript function that retrieves the next fifty patients and so on. This is called lazy loading, a method applied to improve the response-time of the application as it only loads resources on demand. If there are less than fifty patients to retrieve, the server will respond with as many patients there are left.

Similarly, if the patient chooses to use the search bar, as the input is typed, the system calls a JavaScript function that requests patients with that combination of characters in their name from the database via the server and updates the table in the page accordingly. To clarify this idea, initially the selection is not filtered so the first fifty patients from the database are displayed. Then if the user searches for ‘Harvey’, as every character is typed the system requests patients

with 'H' in their name, then 'Ha' and so on until the full name has been searched. This also makes use of the lazy loading feature.

After the search is finalised, in order to select which patient will be associated with the device, the user can use the "Choose" button which only becomes clickable if the wifi settings have been configured and the device is connected. This wifi configuration is done via the pop-up opened by the "Settings" tab where the user can upload their wifi settings and the URL of the server (see Figure 6.10). After confirming, the details will be sent to the Sato Bani device where it will try to connect to the wifi. The screen will update the status in the configure page to show if the connection was successful. All the possible statuses of the Sato Bani device are listed in Figure 6.7.

6.4 Information transfer

Information transfer and management were sensitive topics during development as we had to ensure the design of our system was promoting security by design. That meant reducing the footprint on which patient data appears and using hashes where needed.

There are 3 different types of communication in the system:

- GUI to Server over internet (wifi or wired).
- GUI to SATO BANI over usb wire.
- SATO BANI to Server over hospitals wifi.

6.4.1 GUI to Server:

In this phase, the SATO GUI communicates with the server through the internet and has to transfer highly sensitive patient data. In short all the data that appears on the GUI is sent from the server to the GUI. This channel is secured by TLS (Transport Layer Security).

6.4.2 GUI to SATO BANI:

The second communication, between the GUI and the SATO BANI is done over a usb wire to provide the device with all the information it needs to send glucose data to the server. The data includes:

- Wifi data:
 - Wifi name
 - Wifi password
- Server address
- Patient hash

The patient hash is associated with a certain patient in the database and gets changed every time a new device is configured for said patient. This update does not influence the history of readings in the database as the glucose levels are linked to the patients' ids through their respective hash at that time. This ensures that no patient data gets accidentally leaked or stolen through the device. The wifi is also secured with TLS, although the current implementation of the SATO BANI code does not use TLS to secure the communication given we are only running

development environment code. However, TLS is a must in a production environment and must be arranged.

6.4.3 SATO BANI to Server:

In the last communication, the SATO BANI connects to the server over the hospital's wifi to send the glucose readings. This implies a single POST method with the glucose reading and the patient hash obtained from the GUI during the setup stage. This channel is also planned to be secured by TLS.

6.5 Database evolution

In order to develop our system and ensure its functionality is working correctly, we created our own MySQL database using docker. We also communicated with the client to ensure that the database schema and data types are kept the same for both parties, as agreed during the design phase. Due to the change in requirements, as mentioned at the beginning of this section, we continued using our docker database for the rest of the project.

During the development of the project, we realised that a login function would be necessary for the project for security purposes and therefore we added a new table. This table, named 'staff', includes a SID standing for 'staff id', a username and password. We initialised this with an arbitrary number for the SID and 'admin' and 'password' for the other two values. When this project gets transferred to the client, this table should be removed and replaced by their own system. It was necessary for us to create this table as it would have been a large liability for us to have access to the client's login system and database. These changes are reflected in the new structure shown in Figure 6.11.

Furthermore, we also filled our database with dummy data in order to test that our functionalities are working as intended. To optimise the searching time for a patient we added indexes on the two columns which would be searched, namely the 'current_name' and 'maiden_name'. In addition, a new column was added to the patient table that stores the current hash for the patient id (for more details on its purpose and use see section 6.4. Information transfer).

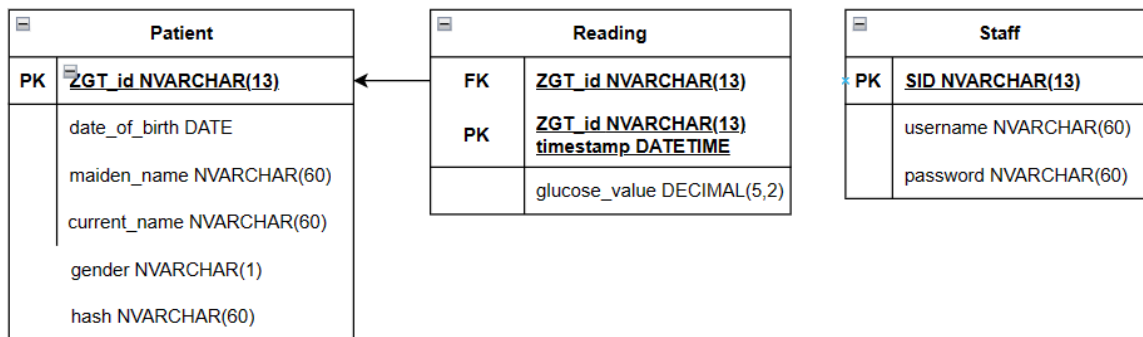


Figure 6.11: Database structure

7. Evaluation

In order to assess the correctness and performance of our system we followed the testing strategy agreed in our proposal (see section 3.2). The following section highlights the results of the testing procedures concerning four main themes: functionality, usability, security and scalability.

7.1 Functionality evaluation

Assessing whether the Sato Bani fulfils its intended functionality, i.e. transmitting live glucose readings from the CGM to the database was challenging. Currently, we are simulating the connection between the ESP32-C3 and the FreeStyle Libre. This introduces a level of abstraction on our process, but although the actual NFC sensor is not used, we still think this can offer valuable insight into the system's complete functionality and behaviour under controlled conditions. The device can handle simulated glucose readings in real time and can support quite a large number of devices, all sending data at the same time.

7.2 User experience

For an evaluation of the user experience, we are confident that the system is fairly intuitive and easy to use, mainly due to how the home page contains a list of detailed instructions, and also due to the fact that the hospital staff themselves expressed their positive feedback on this topic. A lot of the decisions that went into creating the GUI were taken with the help of the client, and were made keeping streamlining the process a priority.

7.3 Security evaluation

From a security standpoint, we performed a risk assessment (see 3.3.3 Risks Summary) and tried to mitigate all risks mentioned as much as possible. We took all necessary precautions, even though the intended place of the system is the relatively closed environment of the hospital.

Some of the security measures include:

- User authentication
- Using prepared statements when querying the database
- Using a unique hash of the patient ID

7.4 Scalability

The scalability of the project was one of the main requirements for the project, so we tried our best to make the system as extensible as possible. Our tests show that the server can support 100 devices sending data at the same time with no major problems. The database can also easily support a large number of patients, and is not overwhelmed when handling these large numbers, mainly due to the lazy loading functionality that keeps it from overloading the GUI with entries, thus avoiding slowdown.

The least scalable component of the system is the device itself, as the hospital would need to buy one for every patient, although the components are not hard to come by and can probably be ordered in bulk.

7.5 Battery reliability

While originally planned to be manually implemented most of the battery management features are provided internally by the esp32-c3. Features include:

- Battery charging
- Overvoltage protection
- Undervoltage protection
- Charging indicator light

The current chosen battery for the device is a very small 250mah 1 cell Li-po battery and while it depletes relatively fast when in use, the SATO BANI is planned to be running in deep sleep mode 99% of its operation, which we estimate will make it last longer than twenty four hours according to the current behaviour.

8. Conclusions

During the lifetime of this project we managed to create a prototype designated to the transfer of glucose levels for in-hospital diabetic patients. The final version included a robust database designed identically to the hospital's table structure, a complete server that is able to handle all communication end-points and security related aspects, a detailed and comprehensive GUI which fully supports the configuration of the device with wifi and patient data and a compact hardware device which is able to simulate the reading of glucose data. The database, server and GUI are fully functional and integrated among themselves and with the Sato Bani hardware device. In regards to the device itself, the only incomplete part is the integration with the NFC reader which would require future research.

To conclude, we will offer a short reflection on our journey, addressing the success and the failures encountered along the way. The aspect we are most proud of is that the final deliverables were well received by the ZGT Hospital staff and that they were very happy with the progress which we have achieved during the module.

We delivered:

- The Development environment of the server and database which orchestrates all the functionalities in the system.
- The Development environment of the Desktop application (GUI) which is able to communicate with the server and configure the SATO BANI device. This component has all the features implemented which we had originally planned.
- The SATO BANI device prototype. While in the current state we were unable to extract data from the FreeStyle Libre, it is still fully integrated into the development environment and sends simulated glucose values to the server.

Despite our efforts, there was a cascade of problems that gave us very little time for NFC communication development between the SATO BANI and the FreeStyle Libre 2. Initially, we assumed that because the FreeStyle Libre can communicate with phones using BLE, we can solely use BLE to establish communication between the ESP32-C3 and the FreeStyle Libre. However, after a few weeks of research and development this turned out to be false. We discovered that either approach would need to implement some level of NFC communication, and given the time limitation we decided to develop only the NFC scenario.

Furthermore, because of an incompatibility issue regarding the communication protocols (see section 6.1.3), the hardware development was slow during the time it took for the correct board to arrive from abroad as the PN5180 is a very specific model which could not be purchased in the Netherlands. Despite acquiring the correct board, the Arduino libraries designed to interface with the PN5180 did not work as we needed and implementing the functions ourselves was not an option time-wise.

Overall, one after the other we accumulated huge delays in the development of the interface between the SATO BANI and the FreeStyle Libre, which in the end did not give us enough time to complete the integration. However, we are still very proud of the state of the product which we have managed to create during this module.

9. Follow up

Given the potential of this prototype, there are many ways in which it can develop in the future, and in this section we will outline some of the potential development directions. Firstly, future development on this project should aim to complete the functionalities we were unable to complete, as mentioned in Section 6. This includes either replacing or implementing the PN5180 board as well as dockerizing and integrating the project with the client's systems.

The device itself would benefit from a lot of changes that could be done to it, but require extensive developmental time:

- As mentioned above, implement the NFC communication
- After the NFC implementation try to pivot the device to work using BLE. That would allow the device to not be strapped to the arm of the patient, instead just needing to be in close proximity to the patient to read the glucose values.
- Adding a screen for the patient themselves to be able to see the blood sugar readings and a status letting them see that the device is still in contact with the server. This will instate a lot of trust in the system giving the patient the reassurance that their blood sugar readings are in order and that the nurses can see them live. This is in part needed because once our device is connected to the FreeStyle Libre the patient can no longer interact with the reader to read the glucose values on their own.

The GUI could be expanded to have a dashboard for the hospital staff to see the real time updates of the glucose readings. At the start of this project the client mentioned they would like

to create this themselves, however, the possibility is always there to build it on our existing GUI and integrate this dashboard with our own work. Furthermore, we also recommend including into the dashboard the statuses of all live devices alongside their battery usage.

Aside from extensions on the current system, we recommend future developers to undertake a usability test. We mentioned in our project proposal that we wished to perform this type of testing if the timeframe of our project allowed, which it unfortunately did not. Our team made an effort to provide all the instructions in the GUI in a hopefully intuitive manner so that users outside of the Computer Science field would be able to follow it as well.

10. Contributions

At the beginning of the project, there was a clear distinction between the parts of the project we would need to address: the software and hardware. Therefore, we decided to divide the team into two subteams: the software side consisting of Stefan, Alexandra and Ciprian and the hardware half with Petras, Maria and Daniel. Naturally, team members should focus on the tasks assigned to their respective division, but we helped each other when either side needed more support. For example, as there were delays with the arrival of the hardware components, members from the hardware half helped developing the software. On a similar note, as the software was developed quite rapidly, members from the software side also focused on working with the hardware device when it arrived. The responsibilities each team member had are listed below:

- Stefan: Lead Backend Developer, Communications Manager, Main Presenter, Organiser, Frontend Developer, Poster Designer
- Alexandra: Lead Report Writer, Database Designer & Manager, Planner, Backend Assistant, Poster Assistant
- Maria: Lead Frontend Developer, Poster Designer, GUI Designer, Hardware Assistant
- Petras: Lead Hardware Developer, Frontend Developer, Main Presenter, Poster Assistant
- Daniel: Hardware Developer, Box Designer
- Ciprian: Diagram Designer, Backend tester

Resources

Kakraniya, P., Ambad, R., Jha, R. K., Jadhav, D., Dhawade, M. R., & Wankhade, Y. (2024). An Epidemiological Study on Diabetes and Pre-Diabetes in an Urban Area with Reference to Lifestyle Modification. *E3S Web of Conferences*, 491, 03014.

<https://doi.org/10.1051/e3sconf/202449103014>

- Tao, Z., Shi, A., & Zhao, J. (2015). Epidemiological Perspectives of diabetes. *Cell Biochemistry and Biophysics*, 73(1), 181–185. <https://doi.org/10.1007/s12013-015-0598-4>
- Vashist, S. K. (2013). Continuous Glucose Monitoring Systems: a review. *Diagnostics*, 3(4), 385–412. <https://doi.org/10.3390/diagnostics3040385>
- Over ZGT. (n.d.). <https://www.zgt.nl/over-zgt/>
- Asghar, A. R., Tabassum, A., Bhatti, S. N., & Jadi, A. M. (2017). Impact and challenges of requirements elicitation & prioritization in quality to agile Process: Scrum as a case scenario. *2017 International Conference on Communication Technologies (ComTech)*. <https://doi.org/10.1109/comtech.2017.8065749>
- Merzouk, S., Elhadi, S., Cherkaoui, A., Marzak, A., & Sael, N. (2018). Agile software Development: Comparative study. Social Science Research Network. <https://doi.org/10.2139/ssrn.3186323>
- Hudaib, A., Masadeh, R., Qasem, M. H., & Alzaqebah, A. (2018). Requirements prioritization techniques comparison. *Modern Applied Science*, 12(2), 62. <https://doi.org/10.5539/mas.v12n2p62>
- Kukhnavets, P., & Kukhnavets, P. (2022, August 25). What are prioritization techniques? MOSCOW method. Gantt Chart GanttPRO Blog. <https://blog.ganttpro.com/en/prioritization-techniques-and-methods-for-projects-with-advantages-of-moscow-model/>
- Daraojimba, E. C., Nwasike, C. N., Adegbite, A. O., Ezeigweneme, C. A., & Gidiagba, J. O. (2024). COMPREHENSIVE REVIEW OF AGILE METHODOLOGIES IN PROJECT MANAGEMENT. *Computer Science & IT Research Journal*, 5(1), 190–218. <https://doi.org/10.51594/csitrj.v5i1.717>