Demenchair

EEMCS Pervasive Systems

April 24, 2024

Mayank Thakur (s2838613)
Vithursika Vinasiththamby (s2818531)
Meenakshi Girish Nair (s2837471)
Twan Berg (s2806746)
Anthony Anazo (s2781808)
Tom Nieuwland (s2812304)
April 2024

Contents

Chapter 1

Introduction

As the global population ages and lives longer, healthcare for the elderly is an increasingly important issue in the world. A large portion of the elderly suffers from chronic conditions that come with age, with dementia being one of the most common conditions among these. People with dementia experience a gradual decline of thinking skills and cognitive abilities as well as memory loss and impaired judgement, causing significant problems in everyday life and impairing their ability to function independently. The increasingly difficult situation for people with dementia and their caregivers in the current tight labor market as well as the global rise of dementia introduces the need for innovative solutions to support individuals living with dementia and their caregivers. The challenges of improving the quality of life of dementia patients while also allowing them to remain living at home using unobtrusive and non-invasive technologies.

With the rise of AI and pervasive computing, the opportunity to address this challenge arises. In this project, the creation of an innovative specialized and modular system, placed on a central chair in the room of a person with dementia, with the intention to revolutionize in-home dementia care is executed. This system, appropriately named 'DemenChair' from 'Dementia' and 'Chair', leverages radio wave-based sensing technology combined with pervasive computing and AI aims to provide a new way of monitoring key aspects of the independent functioning of individuals with dementia like vital signs, activities and behaviour. Harnessing the power of the ESP32-enabled radio wave sensors and pervasive computing allows us to build a system offering assistance to caregivers with continuous and real-time information, enhancing the well-being and quality of life of these individuals with dementia. While also respecting their privacy and potential aversion to technology by collecting this information in a non-invasive way.

In this proposal, the plan to develop the DemenChair system details the general overview of the project, the goals and risks of the proposed system as well as a planning and responsibility overview. Through collaborative efforts, an effective and compassionate approach to monitoring dementia patients and improving their care can be taken.

In Chapter 2, an analysis is made of the domain this project is situated in as well as the technologies used in this system. In Chapter 3, the system proposal is outlined including a preliminary description, requirements, planning, risk analysis and testing strategy. In Chapter 4, a detailed requirements specification is discussed. In Chapter 5, an overview of the design of the resulting product is explained. Then in Chapter 6, we delve deeper into the details of the design for each specific part of the system. In Chapter 7, our test strategies, test plan and test results are discussed. As we deal with an AI system used by dementia patients we need to consider Ethical issues, which we do in Chapter 8. In Chapter 9 we propose future improvements and next steps in this project. Then in Chapter 10 we evaluate on the process and the result of the development process of the last 10 weeks. Finally in Chapter 11 we draw our conclusions about the project.

Chapter 2

Project Analysis

In this chapter, the process of our research into the domain and the project is discussed. We look into the problem statement we have developed for this project. We discuss our research into existing solutions and the new technologies we will use for this project. Finally we present our stakeholder analysis of all stakeholders involved in this project.

2.1 Introduction to the problem

The various challenges posed by dementia, including memory loss and reduced cognitive functioning, have a great impact on the ability to function independently in everyday live and on the quality of life of individuals with dementia and their caregivers. Traditional monitoring systems and dementia care often involves invasive technology to be able to allow to continue living at home or requires them to move to a nursing home. To address these challenges we propose the development of a new technology to help with in-home dementia care without compromising privacy and comfort by using radio wave-based technology and advanced processing techniques including machine learning to capture subtle changes in vital signs, behavior and movement, enabling personalized care and quick assistance when needed. The biggest challenge we face developing this project is the integrating of state-of-the-art technology in a way that is both unnoticeable or at least unobtrusive to the user while still allowing for detailed and real-time extraction of as much information as possible and doing this while not infringing on privacy or reducing the quality of life of the individuals with dementia and their caregivers.

2.2 Research into existing solutions

Many ideas, technological as well as non-technological, have been brought forward and implemented to improve the quality of life and monitoring of dementia patients. These solutions all have pros and cons which will be listed.

The main solution for supervising patients with dementia is through supervision by family or caregivers. If the family and caregivers are educated in the treatment of dementia patient this will improve the quality of life for the patient which is a great advantage. However there is a downside to this which is the time it costs to supervise these patients. These patients need to monitored daily which is considered a full time job. For family members this would result in losing a lot of time and when caregivers are used this could also be a costly disadvantage. Furthermore in the Netherlands there is a staff shortage in the care and welfare job market which is big and estimated to increase to 138.000 people in 2031 and 155.000 people in 2032 [Alg23]. Considering the staff shortage in the job market for care and welfare this could result into a shortage in caregivers for dementia patients which means that this current solution might not be sustainable.

Other technological solutions have been developed to mitigate this problem by supervising the patient through a device instead of the caregivers supervising directly to reduce the number of people and amount of time needed to supervise a patient. One such example is somewhat similar to the demenchair project, the smart home, where a sensor-enhanced private space can provide unobtrusive health monitoring [Wan+21]. This can be accomplished through multiple different kinds of sensors such as PIR (passive infrared) motion sensors, contact sensors and many other sensors which will be placed in locations where their functionality is working while remaining unobtrusive. There are two kinds in the sense that some smart homes use wireless sensors and others use wired sensors the former being superior since installation while remaining unnoticed is an easier task.

Another solution could be a smart wearable device capable of monitoring the health of the user which is noticeable and thus not favourable since the user needs to be capable of understanding the difficult technology but since patients with dementia have reduced capabilities such a solution is not the best fit for them.

A questionable solution is the GPS tracking devices which are widely used to help caregivers locate patients who tend to start wandering when they are confused or are missing, giving the caregivers some peace of mind as they can now always find their patients. This solution has a multitude of disadvantages, as it only works if the patient always keeps their tracker on them. Additionally, this is a serious infringement on the privacy of the patient which might be uncomfortable for both the patient and the caregivers.

These solutions are all used in many places and provide a way to monitor the activities or events they are intended to monitor. Excluding the smart home these solutions can not monitor the patient unobtrusively while keeping check of metrics such as vitals, eating and drinking cannot be noticed or measured using these solutions.

With that said, a lot of opportunities arise for us to propose a new system including a modular approach to monitor a subset of these signs, to be chosen by the user which signs wished for.

2.3 Research into the ESP32

To implement the project the ESP32-S3 will act as a sensor. Having a holistic understanding of the ESP32-S3 as a micro-controller chip will be an asset to completing the project successfully. The chip is an enhanced version of its predecessors as it has better performance, features and security [Esp24b]. Some of the most important features are:

1. Wireless Connectivity
2. Memory and possibility for external storage
3. I/O Capabilities

ESPs are commonly known for their application in IoT (Internet of Things). One of the advantages of ESPs are that they are easily programmable and therefore widely applicable. One of the applications of the ESP32-S3 is in smart home devices. Smart lights, smart plugs and other smart appliances require Wi-Fi and/or Bluetooth connection. The ESP can support Wi-Fi and Bluetooth 5 and is therefore the best candidate.

The ESP32-S3 can be programmed through the ESP-IDF (Espressif IoT Development Framework) or the Arduino IDE. The advantage of ESP-IDF is the available tools and libraries that can be used to implement any project.

The DemenChair specifically requires radio waves as those contain crucial information about the patient. The ESP32 can be connected over Wi-Fi and therefore can be used to read the radio waves of the environment [Esp24a]. Furthermore, to guarantee a movable design it is of utmost importance that the ESPs can be connected wirelessly to the central server. Thus, the ESP and the server can be connected over Bluetooth. This allows the sensor itself and the server to be at a reasonable distance, but still able to communicate.

To access the right information it is crucial to extract the CSI data from the Wi-Fi information [Auf22]. CSI stands for Channel State Information and it contains knowledge about characteristics of the signal between the ESP and the central server. The CSI data contains information about the phase and amplitude of the signal.

2.4 Stakeholder Analysis

Stakeholders are defined to be individuals, groups or any entity with interest in a particular project. In our project the following stakeholders can be identified:

1. Topvorm Twente as our client. As our client, they have an interest in the success of the project, not only in the commercial sense but also in the sense of innovation. Innovation leads to success in many domains, and the health sector is one of them. They will provide us with general guidance from the industrial perspective.
2. Jeroen Klein Brinke as our supervisor and client will play a pivotal role in the successful completion of our project. His responsibilities will include but are not limited to guidance, setting goals and providing resources for development. As he plays a dual role, his strong involvement and constant feedback in the project will be important.
3. Dementia patients are the end-users of our project. They will be the ones that will be monitored through our pervasive system. We must make sure while designing and implementing the system it does not interfere with the lives of the elderly. The system should enhance their quality of life and comfort.
4. Caregivers, both medical professionals and family members, will be crucial in the later stages of development and the deployment. Caregivers will be interacting with the system and, depending

on the model deployed, it will allow them to rely on the system to send alerts about the condition of the patient.

5. Researchers will be interested in our project since the DemenChair contributes new technology to the field of patient care. They will be interested in the data collected and other anomalies detected during the testing and deployment phase.

Chapter 3

System Proposal

In this chapter, we present our project proposal. The first two weeks of the development process where spent trying to understand the problem, the solution we want to develop, analysing discussions with the client and supervisor, elicitating requirements and developing a plan for the development of the project. In this section we will talk about the system proposal that we created together with our client. We will describe the entire proposed system and take a look into some of the potential risks we foresaw with the project. We will also have a brief look into our original planning for the project and our testing strategy. A more detailed view of the final system and testing will be given in a following section.

3.1 Requirement elicitation

The design of the system is mostly based on the first meeting we had with our client. Our client had a somewhat clear idea of the system he wanted. This meant we could quickly could put pen to paper and start writing our first system proposal. Already in our second meeting with the client we managed to finalize our system proposal after a few minor tweaks to out original idea.

3.2 Description of the system

The system we propose will consist of many parts, integrating together to form a streamlined and efficient monitoring system. It will also allow for a lot of customization by the user due to the modular approach we will use to develop the system.

As said before the system will work based on radio wave signals. We will capture these signals with ESP32 modules, which can continuously detect and record radio wave data using radio frequency sensing technology. The radio wave data captured by these sensors can reflect changes in human movement, breathing, vital signs and other activities providing valuable health indications of the patient. Using these sensors we can monitor the radio waves coming from the patient and the surroundings from different angles. The data is captured and preprocessed before being streamed in real-time to the central hub.

This central hub will be realised by a Raspberry Pi 5, this powerful mini computer will receive the data and process it further. This may involve filtering, normalization, synchronization between the different sensors and feature extraction before sending it through the Neural Network to extract a real-time result from the data. The data transmission and receiving protocol between the ESP32 sensors and the Raspberry Pi will go via Wi-Fi or Bluetooth. This data streaming from the sensors to the central hub will be a continuous process allowing for real-time and frequent feedback and updates about the activities of the user.

The next part of the system is the Neural Networks that will be used to extract actual information from the captured changes in radio wave signals. Neural Networks will be trained on sample radio wave data along with corresponding knowledge about activities the user was doing at the time of capturing the data. Once trained, then neural network will be uploaded to the RPi which will take the data and run it through the neural network to infer predictions on the user's current state or vital signs. By training the model on different activities or vital signs and due to the modular approach with the ability to upload neural networks to the RPi we can detect several different signs, activities or potential hazards.

The system will also offer the user with the ability to extract the raw radio wave data. This data can then be used in order to train the Neural Networks. This training has to be done remotely, due to the limited computing power of the RPi.

Finally, the system will include a web interface to configure, manage and monitor the system. The web interface allows the caregiver to configure the type of neural network used on the RPi to manage what kind of activities or signs are measured. Additionally, this web interface will display the real-time result of the monitoring system so the caregiver can monitor the patient from anywhere.

Integrating all these systems in the modular approach as described will lead to a continuous and non-invasive way of monitoring individuals with dementia and providing valuable insights in the health and activities of these patients to the caregivers.

3.3 Risk analyses

In the development and execution of any innovative project, identifying and managing potential risks is crucial to ensure its successful completion. The following risk analysis section is dedicated to systematically identifying potential technical, organizational, and external risks associated with our project. This analysis not only outlines the potential challenges we may encounter but also proposes mitigation strategies to address these risks proactively. By acknowledging these potential hurdles upfront, we aim

to demonstrate our comprehensive understanding of the project landscape and our preparedness to navigate the complexities inherent in bringing our vision to fruition. Through this thoughtful approach to risk management, we are committed to minimizing the impact of unforeseen challenges and ensuring the project's objectives are achieved well.

Risk	Cause	Level	Mitigation
Integration complexity	Trying to combine separately developed parts of the project without a good communication protocol into one.	High	To develop each component in a modular fashion such that integration testing can be done in the early stages of the project.
Hardware failure	Misuse or unfortunate circumstances	Low	Read and comply by guides.
Software Bugs	Lack of understanding of libraries and code	High	Doing research prior to coding, or consulting experts during the development process can reduce the likelihood of catastrophic bugs
Skill Gaps	Lack of necessary technical skills within the team, particularly in areas related to Hardware engineering	High	Doing research beforehand, planning the process of development, and consulting our advisor will incredibly reduce the chances of our skill gap preventing us from developing a complete product.
Scope creep	A not well investigated project proposal can lead to scope creep	Medium	A well detailed list of requirements that will lead to a complete project with approval of the client.
Unsatisfied Client	Lack of communication about what we, as a team, can do.	Medium	Communicate consistently about what we are and aren't capable of doing. Explain organizational and technical issues we face in order to modify the scope of the project to better suit our abilities.

Table 3.1: Risk Analysis

3.4 Project implementation planning

In general, we will be following the Agile methodology for this project as we believe that splitting the implementation phase into different sprints would allow us to get feedback after each sprint to make sure that the system we are developing satisfies the expectation of the client and supervisor.

We will split the time we have into 4 sprints and in each sprint we will be working on the ESP32s, the backend and the interface in parallel. Each sprint will be 2 weeks long. The last sprint will be used for any leftover work that we were not able to finish in the anticipated time and for further extensions that are not mandatory for the MVP.

In order to ensure that we are on track with the work, we will have a team meeting at least every 2 days. We will also be having weekly meeting with the supervisor every Friday to update him about the progress and discuss any questions we have. Every second meeting with the supervisor we will then also use as a sprint presentation.

We did not plan any specific time for testing since we will be testing every feature as soon as possible during the implementation. This will make sure we do not work for a prolonged time with bugs or issues that will force us the undo work we previously might have done.

3.4.1 Sprint 1

In our first print we plan to task a team of 3 in making a design for both front-end and back-end of our system. This will involve a visual design for the front-end and a working framework. For the back-end we will make a design for the database and make a list of all required API calls. We will also draw up the connection between the front-end and the back-end through these API calls. A second team of 3 members will start looking into the ESP's. They will do research on how they function and how we can connect them to the Pi and possibly start extracting data from them.

3.4.2 Sprint 2

In the second sprint the team focusing on the front-end and back-end will start with the actual implementation. The plan is to have a working web-app, including connection to the database via API calls. The option to upload Neural networks and a login should be working by the end. The second team will focus on getting the data from the ESP's into the database. When this is possible they will try and make the ESP's function wireless.

3.4.3 Sprint 3

In this sprint the teams will work together to integrate the entire system in order to finalize the MVP. This means that the uploaded models can use the ESP data to classify it and present the classification to the web-app. We will also create the data extraction feature.

3.4.4 Sprint 4

In our final sprint we will finish up any left-over tasks that we failed to finish in the earlier sprint. When this is done there we will focus our attention to some extra features if we have time left. This sprint will also be used to work on the report and final presentation of the project.

3.5 Testing strategy

Developing an extensive and comprehensive testing strategy is crucial for our project to ensure our product functions as expected, satisfies the expectations of the users and reliably gives intel on the situation of the user. Many different testing strategies exist and testing can be done on several levels. We have selected some specific strategies and levels to build a procedure we can implement to test our product both at the end of the project and in the earlier stages to test specific features and the integration between parts of the system.

The first testing strategy we will implement is unit testing. With unit testing we mean testing specific parts and individual components of the system, we do this without considering the integration with other parts of the system. That means the goal of this test is to ensure that small independent parts of the system work as expected.

After we have applied all the unit tests we can take some parts of the system and put them together, we can then test if these parts work together. We will do this using an end-to-end testing framework called [Cyp]. This framework will allow us to test whether actions performed on the web-app will travel correctly to the back-end and return a correct response to the web-app again.

The next strategy we want to apply is to test whether our product can monitor the vital signs and daily activities of the user correctly. For this we will need to use manual testing in combination with a trained Neural Network. We can setup the sensors around a test subject and monitor both the subject and the result of the Neural network to see if they match.

The final strategy that we will apply is user testing. This means that we will take actual users and have them use the system and assess the ease-of-use and satisfaction this system brings them. Within the scope of this project the test subjects will only include us personally and our client. Testing with dementia patients requires a lot of considerations to be taken into account meaning it falls outsied the scope of a 10 week project.

Chapter 4

Requirements

In this chapter, the process of elicitating and specifying the requirements of the system is discussed. Additionally, an analysis of the elicitated requirements is presented. The requirement specification was guided by an Agile approach. Iterative loops of development and requirement specification have occured throughout the project and the final result is discussed here. Two levels of requirements are discussed. The User Requirements, specifying the functionalities and actions the user should be able to perform, and the System Requirements, specifying the technologies and functionalities of the system itself.

4.1 Introduction

During our first few meetings with the supervisor we have elicited a list of requirements for the DemenChair project. These requirements varied from functionalities that should be supported, technologies to be used and other design or functional necessities of the project. Along with all these requirements we had discussion about potential additional or future features. We ranked all these requirements based on importance following the MoSCoW principle. (Analysis & Iiba, 2009) Using this prioritization our MVP will include all Must have requirements and as many of the Should have requirements as possible. Additionally we implemented some Could have requirements. The Won't have this time requirements are to be left for future improvements. Based on these ranked requirements we determined our planning and our priorities to get the most important parts done first, to then look at extra features.

4.2 User Requirements

1. [M] As a user I want to log in and out of the system with a 6-digit pin
   A user has to be able to log in and out of the system. The login should be a 6 digit pin as password.
2. [M] As a user I want to see an overview of all uploaded neural network models on a dashboard
   A user has to be able to visit a dashboard containing an overview of all uploaded neural network models. On this dashboard all controls for each model, like activation, deactivation, deletion and creation will be available too
3. [M] As a user I want to upload new TensorFlow neural network models
   A user should be able to upload multiple neural network models representing several different possible vitals or activities that can be extracted from the collected radio wave data. These models should be in a TensorFlow format
4. [M] As a user I want to see the live results of each uploaded neural network model
   Each uploaded neural network is used in the backend to get a live classification using the csi data collected by the ESP sensors. This classification should be visible on the dashboard. A horizontal colored bar will be visible indicating the current and previous classifications by color.
5. [M] As a user I want the live results of each neural network model to update periodically
   The live classifications of each model should be updated periodically to get the latest result. Only the models declared as active will update their results periodically.
6. [M] As a user I want to connect to the web interface from a laptop
   The entire application, including the web interface, is run on a Raspberry Pi 5. A user should be able to visit the web interface running on the RPi from a laptop.
7. [M] As a patient I want an unobtrusive and unnoticeable system
   The patients in question are dementia patients. These patients could have an aversion for technology so the system should not trigger any discomfort in these patients by being unnecessarily noticeable or obtrusive.
8. [M] As a user I want a modular and movable system
   The system should be made so that it can be placed on a chair, in order to support this for all types of chairs and to support other location in the future the system needs to be easily movable and modular so that individual parts can be moved or replaced.
9. [M] As a user I want to move ESP sensors individually
   The ESP sensors are very susceptible to very minor changes in environment, each application of the ESP sensor data requires a slightly different setup. In order to support all these applications the user should be able to place each ESP sensor in it's ideal location.
10. [S] As a user I want to be able to upload the same model multiple times with different names
    This feature would go hand in hand with controlling the individual ESP sensors per uploaded model. By uploading the same model multiple times but assigning different ESP sensors to each model we could measure differences between these sensors.

11. [S] As a user I want to be able to activate and deactivate each neural network model individually.
A user could have different requirements for the system at different times. By training and uploading models specialized in a certain area all these requirements can be met at the same time. However, not each model is always needed so a user has to be able to deactivate and activate each individual model.

12. [S] As a user I want to delete neural network models
Once a model is not needed anymore a user has to be able to delete it from the system.

13. [S] As a user I want to make a capture of the raw CSI data for an undetermined time period in JSON and NumPy format.
Since each neural network is specialized in a certain behavior or vital sign a user should be able to get the raw CSI data from the ESP for a certain amount of time so that the user can use this data to train new models with new applications.

14. [C] As a user I want a snapshot of the CSI data of a variable amount of past seconds in JSON and NumPy format
The system relies on models uploaded by the user to make the predicitions about the classification of the current state of the patient. In order to build and train these models it is required to have the data associated with a known state. For example, if something specific happened in the last 10 seconds we can make a quick capture of those last 10 seconds to get the data associated with that event. Additionally we can start a recording manually and gather the data from that point on until we stop the recording which will also give the data associated with what happened in that timeframe. This data should be available both in JSON and NumPy format, the NumPy format is used for training the model and the JSON format is used for readability for the user.

15. [C] As a user I want to use models with multiple output values and see all results simultaneously
A neural network model could have branches build into it that result in multiple outputs for the same model. When the user uploads such a model the user should be able to monitor all these outputs simultaneously.

16. [W] As a user I want to add and remove ESP sensors seamlessly
Supporting multiple ESP sensors improves the accuracy and speed of the predicitions. Additionally, in order to develop a truly modular system it should be easy to connect and disconnect ESP's to and from the system.

17. [W] As a user I want to see all the connected ESP sensors
Being able to view all the connected ESP sensors improves the usability and could help indicate problems with broken or disconnected sensors.

18. [W] As a user I want to activate and deactivate connected ESP sensors individually
Being able to activate and deactivate sensors could help improve the quality of the system as some models might work better with a certain subset of the sensors enabled.

19. [W] As a user I want to copy an uploaded neural network model
The user should be able to add the same model multiple times without having to upload the model themselves every time.

20. [W] As a user I want to select, for each individual uploaded neural network model, which ESP sensors are used
As with a previous requirement, some models might work better with more or less activated sensors or we could want to measure the difference between multiple sensors for the same model in order to find the optimal placement of the sensors. By being able to select for each model which EPS sensors are used we can achieve this functionality.

21. [W] As a user I want to upload PyTorch neural network models
To open up the system to more users we want to support both PyTorch neural network models as well as TensorFlow neural network models.

22. [W] As a user I want an E-Ink display to see the live classification

An addition to the chair would be to add some sort of display on the side of the chair displaying the live classification. That way both the patient and the user can monitor the state without having to visit the web interface.

4.3 System Requirements

These are the technical requirements of the system. Including the technologies and hardware used as well as the functional details of the system.

1. [M] The system must run on a Raspberry Pi 5
   In order to make a completely modular system the entire program must be run on a Raspberry Pi 5. This way we can easily place this system anywhere and can we reproduce the system on multiple setups.
2. [M] The system must store all uploaded neural network models
   The uploaded models are the core of the system so they must be stored and be available on demand for the user.
3. [M] The system must include a ESP sensor to collect radio wave data
   The most important part of the system is the collection of radio wave data as this data is used for making predicitions about the classification of the current state of the patient. We use ESP sensors to collect data of these radio waves.
4. [M] The system must use the phase and amplitude of the radio wave data from the ESP as input for the neural network models
   The radio wave data collected by the ESP sensors includes among other things a phase and an amplitude of the radio wave, this is the data used as an input of the neural network models.
5. [M] The system must send the data from the ESP sensor to the Raspberry Pi over a USB connection.
   One requirement of the communication between the ESP sensors and the Raspberry Pi is that this communication can happen over USB by connecting the Pi and the ESP together.
6. [M] The system must stream the ESP sensor data through the neural network models to get a classification
   Once collected the radio wave data of the ESP is send to the Pi which then stream this data through all activated neural network models uploaded to the system to get a classification of the current state of the patient.
7. [M] The system must store the classification for each neural network model in the database
   Once a classification is made by a model using the incoming radio wave date of the ESP this result needs to be stored in the database so it is available for the user on demand.
8. [M] The web interface of the system must be build with SvelteKit
   The preferred framework for building the web interface was the SvelteKit framework.
9. [M] The API backend of the system must be build with Python FastAPI
   The preferred framework for building the backend API was the Python FastAPI framework.
10. [M] The web interface must run in a Docker Container on the Rapsberry Pi 5
    In order to ensure compatibility and stability of the system on all Raspberry Pi's we run each segment of the system in a seperate Docker Container, which creates an environment in which the program can stably run. One of these segments is the Web Interface.
11. [M] The API backend must run in a Docker Container on the Rapsberry Pi 5
    In order to ensure compatibility and stability of the system on all Raspberry Pi's we run each segment of the system in a seperate Docker Container, which creates an environment in which the program can stably run. One of these segments is the API backend.
12. [S] The backend communicating with the ESP sensors should run in a Docker Container on the Raspberry Pi 5

In order to ensure compatibility and stability of the system on all Raspberry Pi's we run each segment of the system in a seperate Docker Container, which creates an environment in which the program can stably run. One of these segments is the ESP sensor communication

13. [S] The system should send the data from the ESP sensor to the Raspberry Pi over a Bluetooth connection
    Another requirement of the communication between the ESP sensors and the Raspberry Pi is that this communication can happen over Bluetooth by connecting the Pi and the ESP together.

14. [S] The system should save the raw ESP sensor data in the database
    The raw ESP sensor data can be used for training new models so we need to store this data to make it available on demand to the user.

15. [W] The system won't have support for multiple ESP sensors over USB nor Bluetooth yet
    Having multiple ESP sensors could improve the accuracy and speed of the system. However, this is not achieved yet with the current system and will be left as a future improvement.

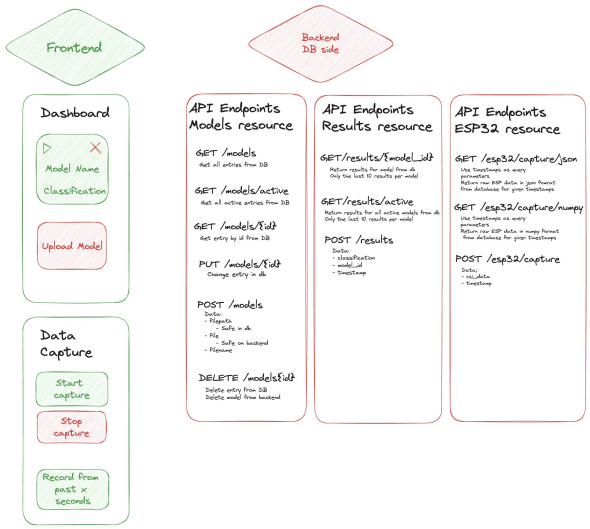
Chapter 5

Design overview

In this chapter, we will look into the design we made for the system as a whole. The system structure is discussed on a high level and we will give an overview of the individual components and how they link together in order to establish a system. In the next chapter we will look into each individual component in more detail.

5.1 Front-end and API calls

Figure 5.1: Overview of Frontend and API integration



In figure 5.1 is the design we made for the front-end with the corresponding API calls to the back-end. On the left hand side we have the two main tabs of our web application; The Dashboard and the Data capture page.

5.1.1 Dashboard

The dashboard will consists of model cards and an upload model card.

The deletion of models will be handles by a button on the corner of the model cards. When clicked this button will ask for confirmation in order to prevent accidental deletions and then make the back-end call to delete this model.

In order to activate or deactivate a card there will be a pause/play button in the other corner of the model card. When pressed this will make either the activate or deactivate call depending on the current state. This will update the cards' state in the database.

The last thing displayed on the model card will be the result of the models classification. This will be fetched every 5 seconds from the database using the results call. The last 10 classifications will be displayed on the card using a color coded, stacking, horizontal bar-chart.

The other card on the dashboard will be the upload model card. This will allow the user to create a new model card based on a name and file that can be uploaded when the card is pressed. After providing the name and file the upload call is made and a new model card will be created.

5.1.2 Data capture

The data capture page will consists of 3 buttons and an input field.

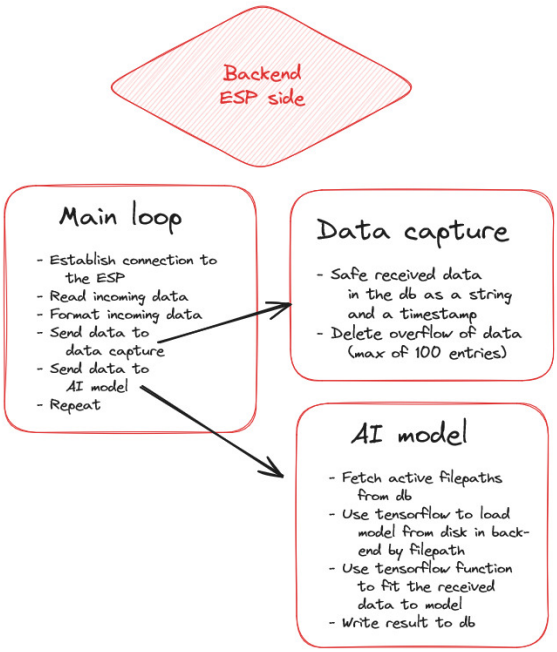
The first button, labeled 'start capture' will be used to start a recording. After pressing the start button the recording can be stopped using the 'stop capture' button. After the stop capture is pressed the capture call is made to the database to request data for the time interval.

Another way of requesting data is to enter a number of seconds into the input field of the last button. After you input your desired number of seconds you can press the request button. This will then also create a capture call. For this call the current time will be used as the stop time and the start time will be determined by the current time, minus the input time.

After using either of the methods to request data. The requested data will be presented to the user in the form of a downloadable file in both Numpy and Json format.

5.2 ESP

Figure 5.2: Overview of ESP service layer



In image 5.2 is the design we made for the ESP side of the system. It consists of a main loop that will gather the raw channel state information and will then use it to perform two other tasks; streaming the data through the AI models and saving it in the database. We will also the design of the code that is flashed onto the ESP in order to have it comply with the rest of the system.

5.2.1 Main loop

The main part of our system will be a loop that will run infinitely when the system is start. In this loop we repeatedly collect the raw data from the ESP. We do this by first establishing a connection the the ESP. When we have a secure connection with the ESP we can read the incoming data that the ESP will be continuously streaming over the channel. We will then convert this raw data into a usable format for our system. This data is then provided to a method that will handle the uploading of this data to the database and a method that will handle the fitting of this data to the uploaded AI models. After providing both methods with usable data the loop will repeat.

5.2.2 Database uploading

The method responsible for the uploading of the data into the database will do this using an API call. It will provide the formatted data and the timestamp at which the data was recorded. After uploading the data it will perform a check to see how many entries are in the database table. Any entries over 100 will be deleted in order to keep storage size down. The oldest entries will be deleted first.

5.2.3 AI model classification

Before we try to fit the data to the trained AI models we first check which models are active. We do this by fetching the file-paths for all active models from the database. We then use these file-paths to load

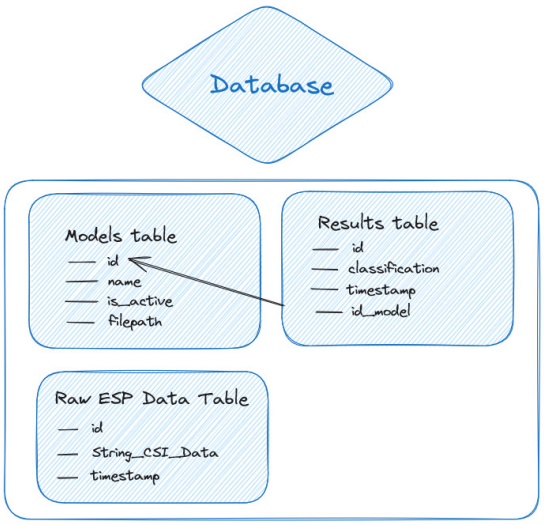
the TensorFlow models. We then fit the data to each of the TensorFlow models. Then TensorFlow model will then provide a classification. This classification is sent to the database and stored there.

5.2.4 Flashed ESP

The ESP is used to capture the WiFi CSI data from the router. In our system, the pi acts as the router and we ping the pi to send the CSI data. The pi creates a hotspot and the CSI data of that wifi is read by the ESP. The code that connects the ESP to the hotspot created by the pi, and reads the CSI data is a GitHub code that is flashed to the ESP using the Espressif IDF. Before the code is flashed onto the ESP, we have to configure certain parameters in the code in order to successfully capture the CSI data. Some of these parameters include data like the ssid and password of the wifi to connect to. Once this is done, the ESP starts reading CSI data. The ESP is connected to the pi using a USB cable and hence, we have a python code that runs on the pi which reads data from the serial port the ESP is connected to. The data read from the serial port is then sent to data capture or AI model as mentioned before.

5.3 Database

Figure 5.3: Database overview



In this section we will take a look at our database design, see image 5.3. The database consists of 3 different tables; the models table, the results table and the ESP data table.

5.3.1 Models table

In this table we store the file-paths to all uploaded models along with their name and active status. We do not store the models themselves since they are files, this makes it easier to store them separately in a folder on the back-end and store the file-path in the database.

5.3.2 Results table

In this table we will store the classifications of the AI models. Each time a model is used for a classification we store this classification along with the timestamp of classification and the model id.

5.3.3 ESP data table

In this table we will store up to 100 entries of CSI data. This is the data that can be extracted from the data capture page to be used for training models. The entries will consist of the data in string format

together with the timestamp at which they were recorded. After each insertion in the database it will be checked whether the limit of 100 is exceeded. When this is the case the entries with the oldest timestamp will be deleted.

Chapter 6

Detailed design breakdown

In this chapter, we will go over all the individual parts of the system and explain them in detail. We will talk about and justify the design choices that were made for the components and how they are implemented.

6.1 Front-end

In this section we will take a detailed look into the design of the front-end. We will go over the design choices that we made both functionally and visually on our way to creating this system. We will have a look into each of the individual components of the front-end and the front-end as a whole.

6.1.1 The whole web-application

Our system will for now only be used by our supervisor as a tool for scientific research. There were no real styling requests for the web-application so we decided to go with a minimalist design. This would make sure the main focus of the application was in the functionality and not the beauty. All the buttons that are displayed on the web-application will create a pop-up alert in the bottom right corner of the screen. This pop-up will inform the user of the performed action in a clear but not distracting way. This was done to create the sense of responsiveness from the web-application, since not all button presses result in immediate, clear and visual change. Our supervisor requested us to build the web-application using Sveltekit since he was familiar with this. This would allow him to continue building upon our application further into the future. Along with Sveltekit we use Tailwind-CSS for our styling. The web-application and back-end will all be running locally on a Raspberry Pi as per request from the supervisor.

6.1.2 The login

Since the web-application will be hosted locally on a Raspberry Pi and will only be accessible via the IP address of the Pi there was no need for very strict security measures. The system will also not have any roles for different users, this means the whole system could be protected by the use of one password that has to be entered when the web-application is visited. Together with the supervisor it was decided to use a 6 digit pin as the password. This would be easy to remember and share with other researcher if necessary and still provide some basic level of security. This password is hashed using SHA-256 and stored as an environmental variable on the Raspberry Pi. This makes it hard for attackers to find the hashed password and makes it close to impossible to turn the hashed password back into the actual password. The design of the login page, figure A.1, can be found in Appendix A

6.1.3 The data capture page

The design for the data capture page was very minimalist, the design of this page, figure A.4, can be found in Appendix A. There were two ways that the supervisor wanted to collect data.

The first method was to start a recording by pressing a button and later on stop that recording. This would then return him the data for the time interval between the two button presses. We achieved this by creating the big green 'Start recording' button. When pressed it turn into a big red 'Stop recording' button. It will also record the current timestamp and safe it as a local variable. When the 'Stop recording' button is pressed the timestamp is recorded again and saved as a second variable. Another alert will be provided to confirm the recording has ended. These start and stop variables are then used as query parameters in the capture API call. The 'Stop recording' button will turn back into the 'Start recording' button for future use.

The second method was to have an input field where he could specify a time amount for which to record data. In order to turn this into a responsive action we decided to have this feature provide data from the past. This would allow the user to receive data instantly for requested time duration. We do this by recording the timestamp when the button is pressed and saving this as the end time in a local variable. We than deduct the input time from this in order to get the start time and save this as well. Then we can use the same API call as before and provide the start and stop variables as query parameters.

The API call returns the data in the from the database that fits in the provided time-frame. With some embedded code in the web-application we turn this data into both a Numpy array and a Json object. We then convert these into files and display a hyperlink to download these files on the interface. When a second request is made using either option, the hyperlinks will be overwritten by the new request.

6.1.4 The Dashboard

The design for the dashboard follows a similar minimalist approach as the other pages, an overview of the design, figure A.3, can be found in Appendix A. The main vocal point on the dashboard are the rectangular cards generated for each uploaded model. This design separates the uploaded models from each-other and clusters all information of a single model neatly into one place. It provides the user with a quick overview of all the models that are uploaded while still allowing for a clear view of each distinct model. Next to these model cards there is one similarly shaped card, the upload model card. This card is shaped in the same way as the other in order not to form a distraction. It also allows the page to scale in a nice way.

When there are no models uploaded at all. A placeholder image will be displayed to inform the user that no models have been found yet. It will also display a large upload button in the middle of the page. Pressing this button will open the upload model pop-up dialog. This dialog is explained further below.

6.1.4.1 Model card

Each model card contains the same layout with a few necessary components. These components include a button to delete the model, a button to activate and deactivate the model, a display of the model name, a display of the latest classification and a color coded bar for the classification over time.

Each model will feature the model name towards the top left corner of the card. It will be displayed in order for the user to clearly see which model card is connected to which AI model. These names are chosen by the user themselves when they upload a model.

The delete button is placed in the top left corner of the card and is displayed as a big red circle with a cross in the middle. This makes the delete button standout from the rest of the card which makes sure it can be easily found and not easily mistaken for something else. When the delete button is pressed there will be a pop-up dialog that will ask the user to confirm the deletion. This is done to prevent a model card from being deleted on accident by an unwanted mouse click. This pop-up dialog features a warning text, a cancel button and big red confirm delete button.

The activation/deactivation button will be displayed on the top right corner of the card. When the model is not active the button will appear as a green play button. This will give a clear indication to the user that this button can be used to activate the model. When the model is already active the button will appear as a orange pause button. This will be a clear indication to the user that this button is meant to pause this model. When a model is on pause, a text will signal the user that the model has to be activated in order to display data. When the model is active it will display the latest classification and the classification over time.

The latest classification will be presented to the user right under the name of the model, in the middle of the card. This makes sure it stands out as the most important thing on the card. This classification will be given as a number due to the nature of the AI models used in the system. This most likely not understandable for anyone not familiar with the uploaded model but is an direct consequence of the models that are used and can therefore not easily be altered.

On the bottom of each card there will be a color coded bar that displays the latest 10 classifications. Each classification will be linked to a color, in this way the last 10 classification form a colored bar in the bottom of the card. When a new classification comes in, the bar will shift from right to left, adding the newest classification and pushing the oldest one out.

Some model cards can have multiple pairs of the latest classification and color coded bar. This is the case for models that are trained to produce multiple results. When such a model is uploaded there will be one latest classification and one color bar per result displayed underneath each other. The card will then also grow in order to keep all the elements confined within the card boundry.

6.1.4.2 Upload card

The upload model card will contain a big circle surrounding a plus sign in the middle of the card. This is done to signal that the entire card can be pressed as a button. The outline of the card is also dashed

instead of solid compared to the other cards. This is done to further emphasize that this is a placeholder card that can be replaced by an actual model when a model is uploaded. Underneath the circle there will be a text saying 'Upload model' to remove any left doubt to the use of this button.

When the button is pressed a pop-up dialog will be displayed with two input fields. In the first field the user is asked to type a name for the model to be uploaded. This name is used as the model name on the card. The second input is used to browse the file explorer for a model file. These models are uploaded as files with the '.keras' extension. When browsing the file explorer, only files with this extension and folders are shows to make searching easier and prevent the user from uploading files that do not follow this extension. Underneath the input fields is an upload button that will finalize the upload. When this button is pressed a check is performed to ensure both input fields contain valid information. If this is not the case the user will be alerted and can try again.

6.2 Back-end

In this section we will look into the design and functionality of the back-end of the system. All design and implementational details concerning this part of the application will be discussed. The back-end of our application consists of 3 main parts. The API, the Database and the ESP communication service.

6.2.1 The API

The first part of the back-end part of the application is the API. An API, or Application Programming Interface, is a set of protocols and rules to allow certain parts of a software application to communicate together. In our project the primary communication the API provides is between the web interface and the database. As well as communication between the ESP service and the database. All data displayed on the web page, as well as all updates and inserts of the data happen through the API. That means that both the web interface and the ESP service do not know about the implementations details of the API back-end and the database. This allows for a widely used abstraction from the complex details of the underlying system which in turn promotes the reusability, replacability and consistency in the development of our project. The API we have build relies on a framework called FastAPI build in python. This provides a framework for building API's following the industry standard and general guidelines to ensure we build a modern and maintainable system.

6.2.1.1 The API Routes

An API works primarily through so-called routes. These routes specify an address to which we can make requests, to access a specific resource in the back-end or in the database. In our application we have three main resources to which we can make requests.

- `/models`
  The first resource in our application is the Models resource. This resource contains all the uploaded models of the system and the details of those models.
  - `GET /`
    Making a get request to the `/models` route returns all models saved in the system.
  - `GET /active`
    Making a get request to the `/models/active` route returns all models saved in the system that are currently activated.
  - `GET /{model_id}`
    Making a get request to the `/models/{model_id}` route return the specific model associated with the variable passed in to `{model_id}`
  - `POST /`
    Making a post request to the `/models` route creates a new model in the database. The details of the model should be passed in the body of the request. At least a model name and the actual model file should be passed in. The model will be saved on the disk and a reference to this location is stored in the database.

    - PUT /{model_id}
      Making a put request to the /models/{model_id} route updates a model in the system. The details that should be updated should be passed in the body of the request. The model that will be updated is specified through the {model_id} property.

    - DELETE /{model_id}
      Making a delete request to the /models/{model_id} route deletes the model specified through the {model_id} property from the database.

- /results
  The second resource in our application is the Results resource. This contains the classifications made by the system using the uploaded models and the radio wave data coming from the ESP sensors.

    - GET /active
      Making a get request to the /results/active route returns all results of models that are currently activated in the system. Only results from the last 10 minutes are included here because there is only a limited space to display these results and results of longer ago are not relevant anymore.

    - GET /{model_id}
      Making a get request to the /results/{model_id} route returns all results of the last 10 minutes for a specific model in the system, specified through the {model_id} property.

    - POST /
      Making a post request to the /results route creates a new result in the database. The details of the result should be passed in the body of the request. At least a classification, model that produced the result and a timestamp should be included.

    - DELETE /{model_id}
      Making a delete request to the /results/{model_id} route deletes the model specified through the {model_id} property from the database.

- /esp32
  The last resource in our application is the ESP32 resource. This contains the raw radio wave data captured by the ESP32 sensors.

    - GET /capture/json?starttime={starttime}&endtime={endtime}
      Making a get request to the /esp32/capture/json route returns the raw data collected by the esp32 sensors in JSON format. The time interval of which the data should be gathered can be passed in as query parameters, indicated by the {starttime} and {endtime} paramaters.

    - GET /capture/numpy?starttime={starttime}&endtime={endtime}
      Making a get request to the /esp32/capture/json route returns the raw data collected by the esp32 sensors in NumPy format. The time interval of which the data should be gathered can be passed in as query parameters, indicated by the {starttime} and {endtime} paramaters.

    - GET /{model_id}
      Making a get request to the /results/{model_id} route returns all results of the last 10 minutes for a specific model in the system, specified through the {model_id} property.

    - POST /
      Making a post request to the /esp32 route creates a new entry of raw esp data in the database. The details of the esp data should be passed in the body of the request. At least a timestamp, and the raw data string should be included

6.2.1.2 The API Controllers

Another part of the API are the controllers, the controllers form a service layer providing the actual communication between the database and the API. Each API route calls a function in a controller associated with the intended action for that route. The controller then handles the action and all related tasks including finding a resource in the database, creating a new resource, updating a resource or deleting

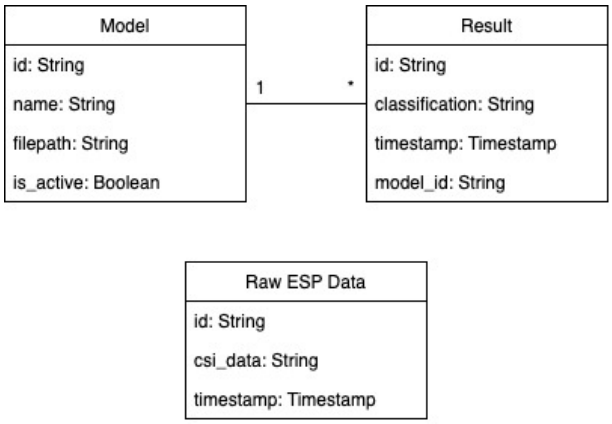
a resource. The controller layer is the only layer in the system with direct access to the database. This ensures consistency and modularization of the system as this part can easily be swapped out by another system without having to change the details of for example the API routes.

6.2.2 The Database

A crucial part of any software system is the persistence of data. Generally, a database is used for this purpose to store and retrieve data. A database is a structured collection of data that is organized in a way that makes it easy to manage, access, and update. It acts as a central repository for storing and retrieving information, ranging from simple lists to complex data sets. Many forms of database exist, often running on a separate server in the cloud. This is not the route we opted for though.

As our system entirely runs on a pi we host the database on the pi as well, the database is a SQLite database which is stored on the pi as a normal file, acting as a database. As we are not dealing with a very large amount of data passing through the system at any point we made the decision to keep the database simple and local for a quick setup and easy access from the system to the database. Since we run our database as a file locally on the Pi, we need to ensure that this file does not get too big. Although the SD card, being the storage of the Pi, can easily go up to 32GB and thus provides plenty of space, we still took precautions to ensure not too much data is saved. Additionally, older data is not relevant to the system anymore. The system is primarily build to display real-time data, so any result of a model older than a couple of minutes is already not relevant. We can also only show the latest 10 results of each model on the dashboard. Similarly for the capture feature, it is not relevant to have data of a few hours ago as it is really hard to pinpoint what the exact behaviour was during that capture. Therefore both for the `Results` and `Capture` resource we have installed a limit on how many records are stored for those resources. This can easily be changed by changing just 1 variable to a higher or lower value.

Figure 6.1: Class Diagram of the database



The database consists of three tables as shown in image 6.1. The first table, the `Model` table, holds the information for all uploaded models. As explained before we store only the filepath to each model in the database while storing the actual file of the model on the disk. The second tabel, the `Result` table, holds the results for all models. The `classification` is the actual result generated by the model using the live input data coming from the ESP sensors. With the `model_id` referencing the `Model` table by the `id` property. As seen in the class diagram a model can have many results but each result only belongs to 1 model. Finally we have the `Raw ESP Data` table, as it says this stores the raw ESP data coming from the ESP sensors. This data can be used for training new models after downloading this data using the capture feature on the web application.

6.2.3 The ESP Communication Service

The final layer of our back-end is the service layer handling the communication between the ESP32 sensors and the database. This layer handles three processes regarding the data coming from the ESP32's: Receiving the data, using the models to make a prediction of the classification and storing the data.

6.2.3.1 Data receiving

Two ways of receiving data from the ESP32 sensors are included in our system between which we can easily swap by changing 1 variable indicating which way to use. One way is over USB. By connecting the ESP32 sensors to the Pi using a USB cable and sending the captured data through this connection, we can read the data captured by the sensors from the serial port to which the USB cable is connected. The other way is over Bluetooth. In this use case the sensors send the data over Bluetooth to Pi once a connection between the ESP and the Pi is established.

Both ways of retrieving the data from the ESP sensors return a similar type of data including among other things the CSI Data, which is the actual radio wave data captured by the ESP. This CSI data consists of a list of 32 tuples of the phase and amplitude of the radio wave. We take the magnitude of this tuple by considering it as a vector and calculating the relative magnitude. This results in a list of 32 numbers.

These lists of 32 numbers get combined into groups based on the packet rate that the system is set to. This can be changed in order to control how many packets are received before streaming the data through the neural network models. An important consideration here is that the packet rate must match the input size of the model. For details how this packet rate can be changed, see the manual in Appendix C.

6.2.3.2 Storing the raw data

The next step in this layer is to store the raw radio wave data captured by the ESP to the database. For this we take the converted data from the ESP sensors (the tuples converted to a single number representing the magnitude), and store this array as an entry in the database. To store it in the database we make a request to the API layer of the back-end, using a post request to the route `/esp32`. This layer then handles the communication with the database.

6.2.3.3 Classification of the data

The final step in this service is to use the uploaded neural network models to classify the data in real-time. In the system the user has the ability to upload models as well as to activate or deactivate each individual model. In this step of the process we only use the models marked as activated. So the first step is to get the active models through the API, using a get request to the route `/models/active`. Once we get these models from the database we go through each one of them and load the respective model from the filesystem using build-in TensorFlow functions. Then we feed the received data into the model, which returns a prediction for the classification of the current state of the patient. This classification is the result we want to store and display to the user. So the final step in this process is to store this result to the database. For this we make another request to the API, using a post request to the route `/results` to create a new result in the database.

Which concludes the entire process of everything happening on the back-end ranging from API calls to the handling of the incoming data from the ESP32 sensors.

6.3 ESP

In this section, we will go over the design of the ESP. We will give an insight into the different designs we considered and the choices we made.

6.3.1 Capturing CSI data

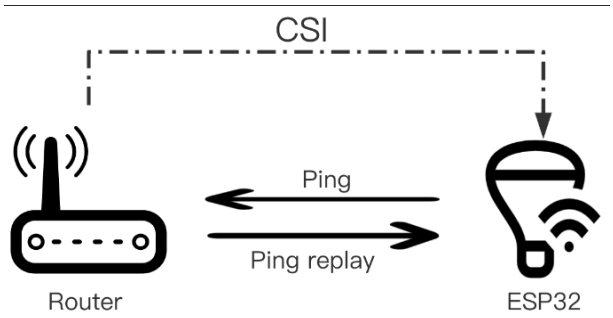
In order to capture CSI data, we considered implementing multiple designs. Initially, we tried implementing a model which would configure one ESP to act as the sender and the other one as the receiver [Zha23a]. However, after multiple attempts to implement this model, we realised that we misunderstood the design and that we do not want two ESPs to send and receive data between each other. Instead, we want the pi to be the access point and the esp to read the CSI as shown in Figure 6.2. Hence, we decided to look for other models and we decided to prioritise getting one ESP to read the CSI data from an access point.

Accordingly, we tried to implement a model that actively listen to the access point and try to read the CSI data [Her21]. However, we faced some implementation issues when trying to flash this code onto the ESP as there were some libraries and files required to run the code which was not available.

Subsequently, we tried to implement another model which would scan all the frequencies and collect all the CSI data that it receives. The idea then was to filter on the MAC address to get the CSI data from the access point we want and not from all the surrounding access points. However, we were not receiving any CSI data when running this code and hence we came to the conclusion that we need to specify the access point we want to connect to.

Finally, we decided to use a model which would ping the access point and the ping command triggers the router to send packets from which the CSI data can be extracted as shown in Figure 6.2 [Zha23b]. In this model, we can also set the ssid and password of the hotspot created on the access point and hence it reads only the required CSI data. The code connects the ESP to the hotspot created on the access point, which in our case is the pi, and it initialises the collection of CSI data. Then it regularly sends pings to the router to read the CSI data at uniform intervals.

Figure 6.2: CSI data collection



6.3.2 Communication through serial port

The communication between the pi and the ESP can be done in two ways, either by establishing a bluetooth connection or by connecting the ESP to the serial port of the pi using a USB cable. For the MVP, we decided to focus on getting the ESP to send the data to the pi using the USB cable. In the python code that runs on the pi, we set the serial port to the port that the ESP is connected to and the python code reads the data coming in to the serial port and puts it to the database or sends it to the AI model. The data read from the port is parsed into an array of integers, in which each pair of integers in the CSI array is squared and summed in order to get the amplitude. Hence, the initial CSI array we receive is an array of 64 integers and we return an array of 32 integers.

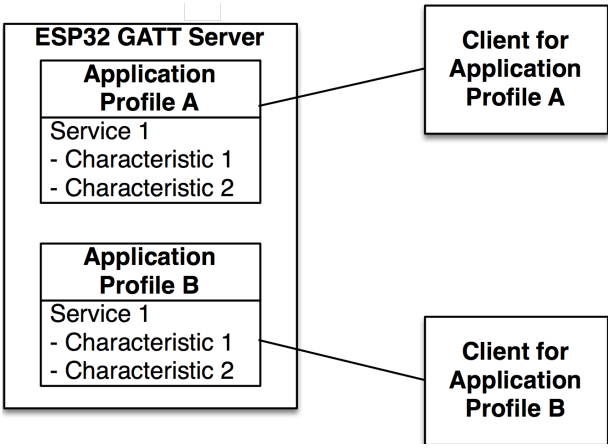
6.3.3 Communication through bluetooth

In order for the ESP to communicate with the pi, we need to implement the GAP and GATT profile. The GAP profile handles the device discovery and the connection to Bluetooth Low Energy (BLE) stack. GATT, on the other hand, handles how the data is transmitted over bluetooth once there is a stable connection.

Using the GAP profile, we can determine how the BLE device will establish connection with other BLE devices. The device can be either configured to have a broadcasting role or a connection role. We chose to have a connection role for the ESP and the pi as we do not want to publicly advertise the data packets. Our goal is for the ESP to connect to the pi and send the data packets. The ESP acts as the GATT server and the pi acts as the GATT client. In order to store and transfer the data over the bluetooth connection, we use services of the server. The service is a set of functionalities the server offers and within a service, there are characteristics, which are the data that will be sent to the client over the bluetooth connection.[Ste24] This is depicted in Figure 6.3 below. The client can then read from or write to the characteristics. When the client sends a read request, it triggers a read event in the ESP, where the CSI array is split into separate blocks and sent to the client. The GATT server uses the Attribute protocol

(ATT) to communicate with the clients. In our system, once the connection has been established, the pi reads the CSI data from the characteristic of the ESP.

Figure 6.3: GATT server and client



The characteristic attribute can only store a certain size of data and hence it does not fit the whole 64 integers array of CSI data that we receive at once. Hence, on the ESP, when we get a CSI data, we keep appending it in an array and this data is then split into smaller blocks and set as a characteristic. Once the block of data is set as the characteristics, this block of data is then removed from the array. On the pi, we read the characteristic of the server and we build the CSI data array again by appending this blocks until we get to a size of 64 integers. The amplitude is then computed from this data and sent to the AI model and put into the database.

Chapter 7

Testing

In this chapter, the test plan and the test results are provided. The different functionalities which are tested are indicated, the test approach is explained, test criteria are provided and the risks and contingencies are indicated.

Developing an extensive and comprehensive testing strategy is crucial for the project to ensure the product satisfies the expectations of the stakeholders and functions accordingly. Many different testing strategies exist and testing can be done on several levels. Some specific strategies and levels to build a procedure have been selected such that the product both at the end of the project and in earlier stages can be tested on specific features of the system and the integration between parts of the system.

7.1 Unit testing

With unit testing specific parts and individual components of the system are tested for their sole functionality to see whether they as components function correctly. This is done while disregarding the integration of the component with other parts of the system. The goal of these unit tests is to ensure that small independent parts of the system work as expected. This has been done for components of the back-end of the web application. The main goal was to test all the API routes that are in used by the ESP side and front-end.
These test were done using MagicMock from the Unittest.mock library. This creates a mock database for the testing based on the actual database. Then you can than test requests to the database, by checking whether the return value from the mock database matches the expected output format. This was done for the following API routes:

- models
    - GET /
    - GET /active
    - GET /{model_id}
    - POST /
    - PUT /{model_id}
    - DELETE /{model_id}
- results
    - GET /active
    - GET /{model_id}
    - POST /
    - DELETE /{model_id}
- esp32
    - GET /capture/json?starttime={starttime}&endtime={endtime}
    - GET /capture/numpy?starttime={starttime}&endtime={endtime}
    - GET /{model_id}
    - POST /

7.2 End-to-end testing

In this section we will discuss out e2e testing. E2e testing is done to test the entire application workflow from beginning to end, simulating real user scenarios to ensure that all components function in harmony as expected. Below a list is made of end-to-end tests and what they test. All end-to-end tests were made using [Cyp].

7.2.1 Login testing

In this section we will describe our tests with regards to the login.

1. A test for attempting to log in with the correct password. This test first heads to the login page of the website. Then the password is inserted and submitted. After this, a check is done to see if we are now at the dashboard page.
2. A test for attempting to log in with an incorrect password. This test first heads to the login page of the website. Then an incorrect password of correct length is inserted and submitted. After this, a check is done to see if the user is notified that the password was wrong.

3. A test for attempting to log in with a password that is too short. This test heads to the login page after which it tries to insert and submit a three-digit password. After this, a check is done to see if the user is notified the password was too short.
4. A test for attempting to log in with a password that is too long. This test heads to the login page after which it tries to insert and submit a nine-digit password. After this, a check is done to see if the user is notified the password was too long.

7.2.2 Model testing

In this section we will describe our tests surrounding the uploading and deletion of models.

1. A test for attempting to upload a new model using a name and the correct file format. This is done by clicking the upload model button and inserting a name into to the name field. Then a .keras file is selected after which the submit model button is pressed. After this, a check is done to see if the new model is now present on the dashboard.
2. A test for attempting to delete a model after it has been created. This is done by creating a model as described in the test above. Then the delete button of this model is pressed and the confirm delete after that. After this, a check is done to see if the new model is now present on the dashboard.
3. A test for attempting to upload a model without entering a name. This is done by clicking the upload model button and only selecting a .keras file. Then the submit model button is pressed. After this, a check is done to see if the user is notified that a name is required.
4. A test for attempting to upload a model without a model file. This is done by clicking the upload model button and inserting a name into to the name field. Then the submit model button is pressed. After this, a check is done to see if the user is notified that a valid file is required.
5. A test for attempting to upload a model with a name that already exists. This is done by creating 2 models as described in the first test. Both models will have the same name but a different .keras file. After this, a check is done to see if both models exist on the dashboard page.
6. A test for attempting to upload a model with a .keras file that already has been uploaded. This is done by creating 2 models as described in the first test. Both models will have a different name but will select the same .keras file. After this, a check is done to see if both models exist on the dashboard page.
7. A test for attempting to delete all the models from the page. This is done by deleting each model one by one as described in the second test. After this, a check is done to see if the dashboard page no longer contains any models.

7.2.3 Activation testing

In this section we will describe our tests for the activation and pausing of model cards.

1. A test for attempting to activate all models on the dashboard page. This is done by first creating 3 models. Then for each model we press the activation button and check whether the user is notified that the model is activated.
2. A test for attempting to deactivate a model. This is done by selecting the second model on the dashboard and clicking the pause button. After this, a check is done to see whether the user is notified that the model is now deactivated.
3. A test for checking whether the previously activated models are still active after the second model got deactivated. This is done by checking that the first model now contains a classification.

7.2.4 Capture testing

In this section we will describe our tests with regards to the data capture page.

1. A test for attempting to capture live data. This is done by clicking the start recording button. We then wait for 10 seconds before clicking the stop recording button. After this, a check is done to see whether the page now contains two files ready for download and that the stop button has turned back into a start button.
2. A test for attempting to request data from the past with a non numerical input. This is done by typing 'five' into the input field. After this, a check is done to see whether the input field is still empty.
3. A test for attempting to request data from the past using a numerical input . This is done by typing '100' into the input field. After this, a check is done to see whether the page now contains two files ready for download.

7.2.5 User Testing

User testing is performed to test whether the interface is well design and intuitive. This is done by having a person not completely familiar with the system try out the project without much interference from out side. This will force the test person to figure things out on their own, which allows us to see whether the application is intuitive.

We performed 2 user tests, one with our supervisor and one with someone who is completely unfamiliar with our project. The user tests can be found in Appendix B.1 and Appendix B.2 respectively.

Chapter 8

Ethical considerations

In this chapter, we will talk about the ethical questions surrounding our project. We will discuss what the questions are and why they are relevant for this project. We will then discuss the ethical topics themselves and explain how they influenced our project and final product.

8.1 Ethical issues

There are a few important ethical problems that surround our project. On the one hand we need to have a look at the privacy of the patients and their ability to give consent. On the other hand we need to look at the ability for patients and caregivers to interpret the output of the system.

Since our system is used to classify behaviour and vital-signs of patients, there is a clear ethical question surrounding privacy. Regarding the consent, we may not forget that we are dealing with people that suffer from dementia. This brings with it difficult ethical considerations.

When looking at the ability to interpret the output of our system, we have the issue of AI models and their black-box design.

8.1.1 Privacy

When looking at the privacy concerns regarding our project we started by having a look at the General Data Protection Regulations [Uni16]. This is a regulation by the European Union in order to protect peoples privacy. It limits companies in their ability to collect data about people, only allowing for the collection of relevant data when consent if given.

When comparing the GDPR to our project we can see that our project does not fall under GDPR. This is due to the data, radio waves, we collect in order to make the classifications. Radio waves are not considered to be identifiable and therefore do not fall under GDPR.

Not falling under GDPR does not mean that we can not and do not care about the privacy of the patients that use our system. We do not have to follow the rules by law but we still try to respect the privacy of the patients that will use our system. In order to do this we have limited the amount of data we collect and store without undermining the functionality of our system.

8.1.2 Consent

Consent is a vital ethical consideration in our project, because of the involvement of dementia patients. Dementia patients can however lack the cognitive capacity to comprehend what our project entails and how it could effect them. Consequently, receiving consent from dementia patients is not feasible in all circumstances. In these cases, informed consent should be given by family members or legal guardian who can make this decision on behalf of the patient. These representatives should be fully informed about the project including not only the benefits but also the potential risks. This is pivotal in upholding the ethical standards of the project and the autonomy and dignity of the patients.

8.1.3 Interpretability of the system

The last ethical concern we will discuss is the ability to interpret the output of our system. This is made difficult by the so called "black-box" design of the AI models that are used by our system. Caregivers and medical professionals are relying on the classifications that the system provides to them, therefore it is imperative that they can understand how these classifications are made. In order to accomplish this, extensive collaboration with caregivers and medical professionals is required during the training of the AI models. This ensures that they are well aware of how these models are made and the way they determine their classification. With this knowledge they can interpret the output of the system better, knowing the limitations that the models might have.

Chapter 9

Future planning

In this chapter, we will talk about the future of this project. As with most Design Projects, we think there are many ways to iterate on our project. Most of these iterations have to do with making the user experience better, and are relevant for the Web App as well as the ESP

9.1 Backend and ESP

With regards to the ESP, the most annoying aspect is having to find the port when connecting it to the Raspberry Pi. This is something that we struggled with as well, during development. Ideally, we would like for the ESPs to be "plug and play". That is to say, we should be able to flash any number of ESPs with the correct software, and then plug them into the Raspberry Pi. Then, have the back-end detect what port they are on, and allow the user to select that ESP (for data collection or live classification) using the Web-app. For the time being, the port is hard coded into the back-end, and for each new ESP, we have to instantiate a python script in the back-end (with a hard coded port number) which collects and sends the CSI data to the database. This heavily limits the scalability of the system. This is a problem that is partially solved by the Bluetooth aspect of our system. This is done using the GATT protocol, which allows the Raspberry Pi to subscribe to different Services and Characteristics on the ESP, which allows for uninterrupted data transfer. However, this also requires the MAC Address of the ESP to work. This could be solved using a Bluetooth connection interface on the web-app. However, for the time being, the MAC address is hard-coded. That being said, Bluetooth circumvents the issue of not having enough USB ports, it doesn't require multiple instances of a python script, and allows for long range data transfer, which are all desirable aspects of our system as they make it less pervasive.

9.2 Web-app

As with the web-app, we again would like to improve the user experience. For now, this means being able to control the ESPs from the web-app. Using the CLI version of ESP-IDF, we believe it could be possible to flash multiple ESPs from the web-app, having just connected the ESP to the Raspberry Pi via cable. This would make it a lot easier for the user to repeatedly flash ESPs as need be, instead of flashing it from somewhere else and then connecting it to the Raspberry Pi. This would also help the system be more self-contained. Furthermore, we would also like to provide the user with the option of using a single Raspberry Pi to conduct multiple experiments. This means that we would give them the option to select which ESP feeds into an AI model. Lastly, to increase the scope of our product, we would like to support more kinds of AI models, namely, PyTorch models. Support for even more models could be added, but Pytorch and Keras models already cover a vast majority of Python AI researchers, and would definitely be sufficient for the Pervasive Systems department at UTwente.

Chapter 10

Evaluation

In this chapter, an overall evaluation on the project is provided by means of elaborating on its planning, identifying the different responsibilities within the project team, evaluating the project team, looking at challenge we faced during the development and discussing the final result.

10.1 Result

The resulting system we have developed over 2 weeks of planning and about 8 weeks of development is a working system which integrates multiple parts. Starting from an ESP32 sensor capturing radio wave data, it is communicated over USB or Bluetooth to the Raspberry Pi 5, the central unit of the system. Then, the data is stored in the database, and analyzed using an AI model to classify the current state of the patient, which is then also stored. This together with a web interface that has been build to view real-time results of these AI model predictions, upload new AI models, activate and deactivate individual models and extract raw radio wave data captured by the ESP32 usable for training new models, creates a complete system allowing the user to continuously monitor patients in real-time. Looking at the requirements elicited in the first phase of the project, we can confidently say that all Must have, Should have and most of the Could have requirements have been successfully implemented into the system. While we definitely have delivered a Minimal Viable Product based on the requirements and wishes discussed in the initial stages and revisited in later stages, there still is much room for new developments and more requirements to be added in potential next iterations and generations of this product.

10.2 Planning

In our project proposal made in the first stage of the project we had a relatively strict planning regarding the requirements we would tackle in each sprint while also allowing room for changes and updates not only in the planning itself but also in the exact requirements and priorities. By evaluating the progress with the supervisor on a weekly basis and having loads of discussions regarding the progress and priorities as well as new idea's or changes in needs and wishes we managed to have steady progress and mostly stay on the intended timeline developed in advance of starting the development of the product. We planned to use a system called Trello to monitor our progress and keep track of what features where a to do, in progress or done. However, during the development we sometimes forgot to keep track of exactly what progress was being made within Trello. On the other hand, by infrequently updating this we still managed to make this useful by at least having an overview of what was done and what still needed to be done, helping us stay in control of the progress. Furthermore, we met frequently throughout the development process to keep each other accountable and to keep up with each others' progress. We also exchanged long update messages after long days of work, to make sure we were all on the same page throughout.

10.3 Responsibilities

Looking at the responsibilities of the members of our group and the group as a whole we all believe each one of us has done their fair share of work and made valuable contributions to many parts of the system. The primary division of responsibilities in our team was between the development of the web interface and the back-end of the system and the part working with the ESP32 sensors. This way each team consisting both of 3 members could focus on their own individual part, while also regularly coming together to discuss issues, progress and mostly to integrate the two parts together.

For a more detailed overview of the responsibilities we will delve into the tasks of each team:

10.3.1 The Frontend-Backend Team

As primary responsibility of this team we had the development of a web interface showing all the required features and information. This information is gathered from the back-end so it made sense for this team to also work on this back-end.

In between iterations of the product regular meetings took place between both teams to integrate the two parts. Much of the work done by the other team resulted in data that needed to be handled on the back-end. Working on this integration together had several advantages. First, this way everyone on the team was aware of what the other team was working on while also understanding the integration. Additionally, this way we could also help each other out with making important decisions or problems encountered on the way. Getting new insights from another point of view is always useful, especially

when encountering unforeseen problems. Finally, this allowed us to achieve steady iterative progress by encouraging both teams to finish the requirements of that sprint in time. The following is the member-by-member breakdown of the responsibilities:

1. Twan: Front-end design and implementation, back-end design and implementation, Dockerization
2. Anthony: Front-end design, back-end design and implementation, Dockerization, Testing
3. Tom: Front-end design and implementation, back-end design and implementation, Testing

10.3.2 The ESP32 Sensor Team

This team had as primary responsibility to get the ESP32 to capture data and get this data onto the Raspberry Pi 5. This team started out by developing the capturing of the data by the sensor itself. Once that was working we moved to the communication protocol of this data to the Pi, initially over USB. The next step once the entire process worked over USB was to try to get it to work over Bluetooth. This meant a new iteration of a similar process. First, getting the ESP to read data and send this over Bluetooth. To then receive this data on the Pi over Bluetooth.The following is the member-by-member breakdown of the responsibilities:

1. Vithursika: ESP Set up, over cable communication, Bluetooth communication.
2. Meenakshi: ESP Set up, over cable communication, Bluetooth communication.
3. Mayank: ESP Set up, over cable communication, Bluetooth communication.

10.3.3 Group Responsibilities

Alongside the responsibilities of each development team during the development phase of the project we had many other responsibilities that we took up as a team. Including among other things, communication and meetings with the supervisor, requirements elicitation, writing of the system proposal, writing of the final report, poster design and presentations.

We took these responsibilities up as a group, each member focusing on the parts they had more interest in or were more familiar with. For example, while writing the final report, the members of each development team naturally focus more on their own respective sections as they have a deeper understanding of that part of the system.

10.4 Challenges

10.4.1 Frontend-Backend

In the development of the Front-end and back-end portions of the system we faced three major challenges. The first challenge we encountered was the wide range of new technologies we had to use. Although some of us had some experience with a front-end framework, none of use had experience with SvelteKit. Additionally, none of us had a lot of experience in building a back-end and database. Learning these technologies and getting it to work together was a challenge. Secondly, the Dockerization of the program was quite difficult. For all of us, we had seen docker before but had never built our own dockerized multi-layered application. Aside from learning how to create this new docker we also faced issues with linking the separate parts of our application over docker using the right URL's and IP addresses and issues with accessing the ports to which the ESP sensors where connected. Finally, the last issue we faced was with the file uploads of .keras models. Issues with storing, reading, maximum filesizes and sending files in requests occured and had to be solved.

10.4.2 ESP32

One of the primary issues with this portion of the system was getting the libraries to work. More specifically, the ESP-IDF library. This was the main component which was used to flash the ESPs, and get any software running on the sensors. We also had a pretty new style of ESP, the ESP32S3, which

was difficult to configure, given that most of the existing code is for older versions of the ESP32. After that hurdle was crossed, we had trouble understanding a lot of the libraries that are used to gather and send data, because they all made use of internal commands of the ESP-IDF library, which were mostly undocumented. This was compounded by the fact that a lot of the code we found was very long and extensive, which meant that we had to spend a lot of time figuring out what we were actually using, before being able to modify it to fit our function. After we figured this out, things started to make a lot more sense, which was proven by the fact that the Bluetooth portion of the system took much less time to develop than the over-cable portion.

10.5 Team Evaluation

Overall as a team we are extremely satisfied and content with the end result as well as the process to get to that result. Through many iterations, trial and error and other challenges we managed to develop a working product satisfying the requirements and needs as elicited in the first stage of the product. Even though some challenges occurred during the development of the project we never had any problems within the team and managed to overcome all these challenges as a team. The cooperation and communication within the team was always positive and we have had a lot of fun while developing this project.

Chapter 11

Conclusion

Dementia, a disease suffered by many elderly people, demands new solutions to increase patient well-being. The DemenChair offers an innovative solution to continuously monitor dementia patients in real-time. Allowing caregivers and doctors to gather valuable information about the patients well being and behaviour without invading the privacy and comfort of patients.

The aim of this project was to deliver a specialized and modular system, employing radio wave-based sensing technology to be used for in-home dementia care. The development and implementation of the DemenChair project resulted in a significant step towards improving the care for dementia patients. This project, situated at the intersection of pervasive computing and healthcare, has created a modular and unobtrusive system to be used to monitor vital signs and daily activities of patients.

The primary use of this system will be to enable and simplify further research into the topics of healthcare for dementia patients, radio wave-based sensing techniques and unobtrusive monitoring systems. The supervisor and client has indicated this system will be incredibly useful in his ongoing and upcoming research in this field and will be used by him and his research department, as well as open up more possibilities for new design or research projects to further enhance this product.

Although a successful preliminary version of the intended system has been created during the development of this Design Project, continued research and development in this area and on this project is crucial to advance this project to a stage where it can actually be implemented in homes of dementia patients. The potential this project holds is immense if it can be harnessed by means of extended research and development on top of this initial delivered system.

As we look towards the future of healthcare and innovative solutions, the DemenChair shows that with cutting-edge technology we can develop new ways of significantly improving the quality of in-home dementia care.

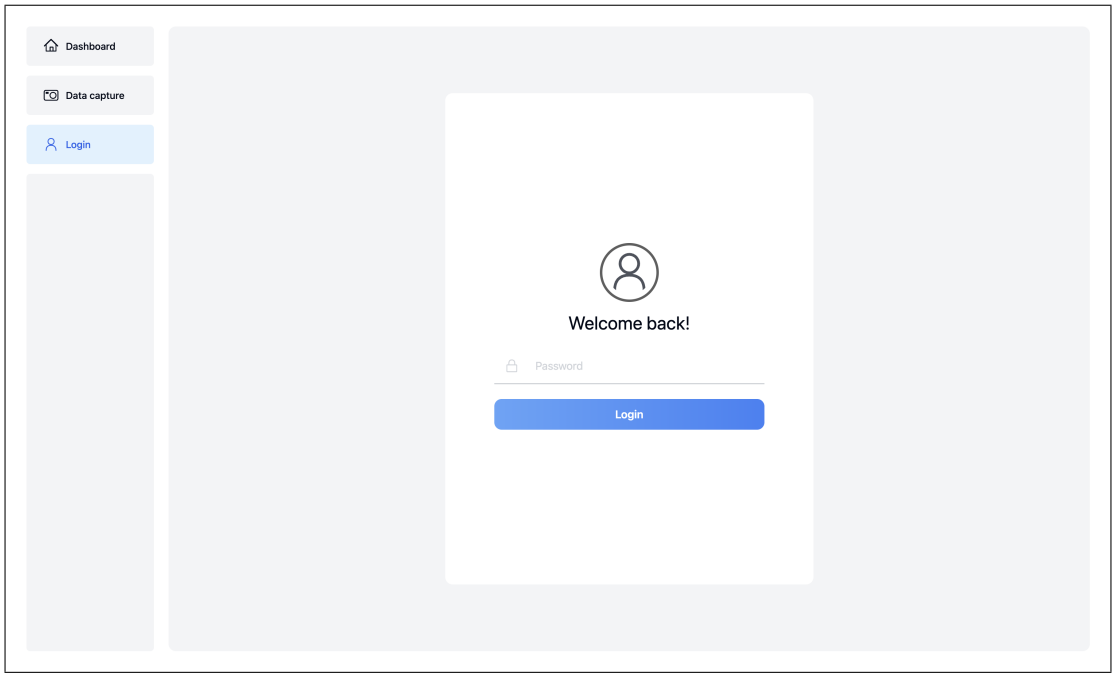
Appendix A

Mockups Web Application

In this section we have an overview of all the mockups of the design of our web application.

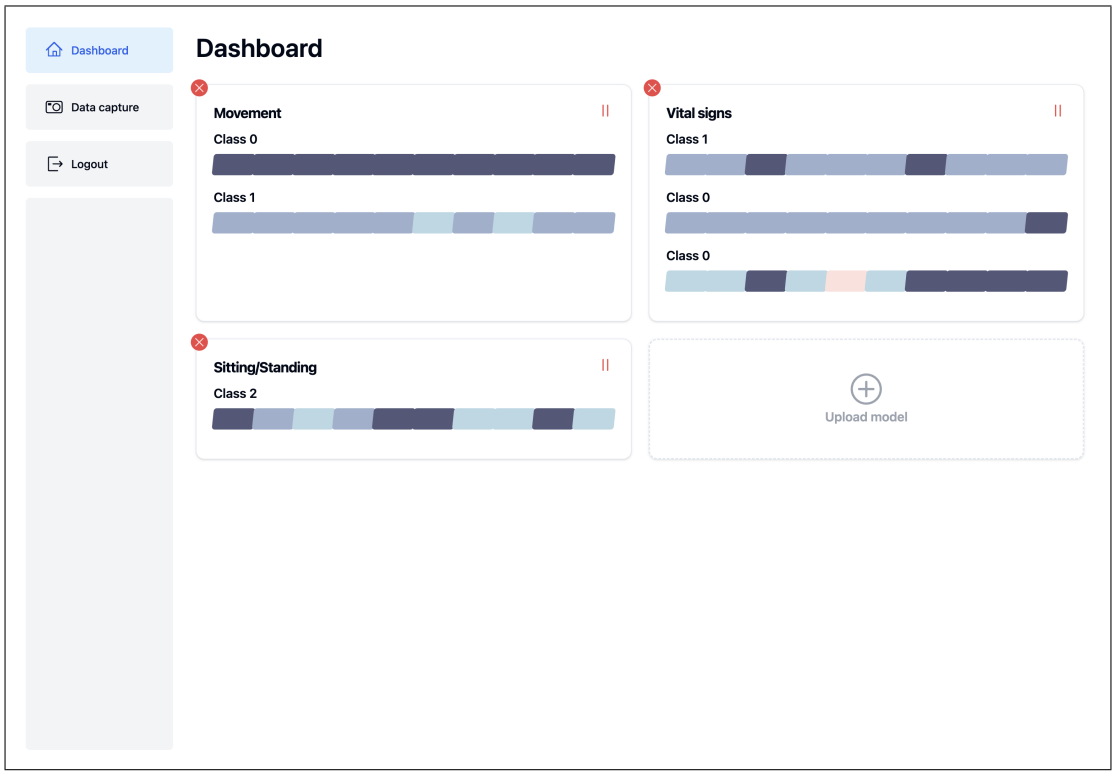
A.1 Login Page

Figure A.1: Login Page



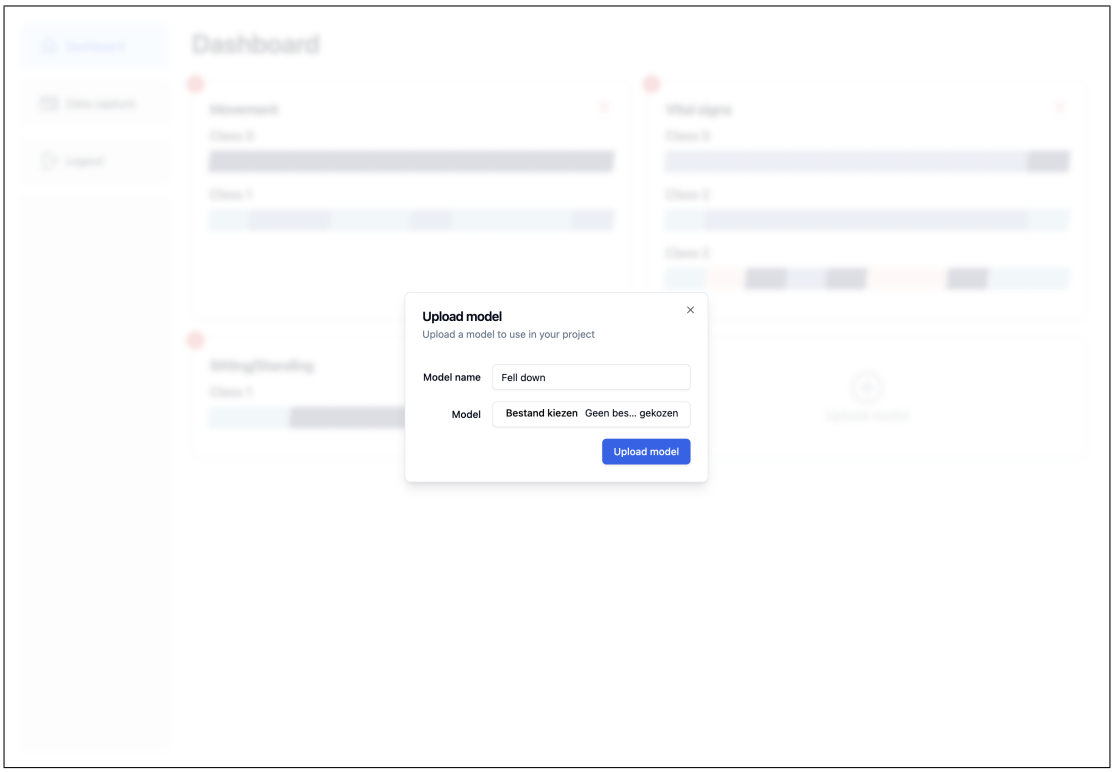
A.2 Dashboard Page

Figure A.2: Dashboard Page



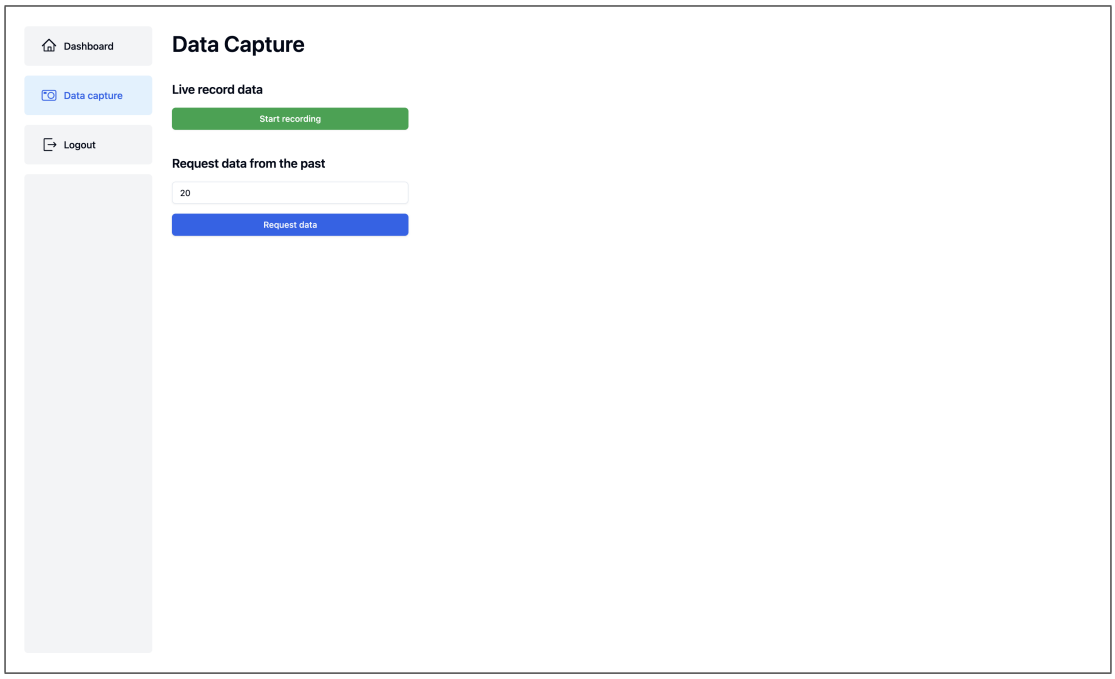
A.3 Upload new Models

Figure A.3: Upload New Model Dialog



A.4 Capture Page

Figure A.4: Capture Page



Appendix B

User tests

B.1 User test 1

This user test was conducted on: 23/04/2024
This user test was performed by: Jeroen Klein Brinke
This user test was supervised by: Anthony Anazo and Tom Nieuwland

B.1.1 Setting up

Here we let the test subject try to setup the system using the Manual from Appendix C. We will only do chapter 3 from the setup to keep the user test short on time. Below we will list some of the observations we made during this part of the test:

1. Startup went well. Jeroen managed to log into the system correctly.

After this part of the test was concluded we asked a few questions about the experience. Those questions together with the corresponding answers are listed below.

1. Q: What are your thoughts on the presumptions we outlined in the Manual?
   A: Maybe some more knowledge can be inserted in the presumptions.
2. Q: Where their parts in the manual that were not fully clear?
   A: It was clear.
3. Q: Was there anything missing in the manual that could have made the process better?
   A: No, it was easy to follow.

B.1.2 The system

Here we let the test subject browse the entire web-application. We do this with as little interference as possible. This is done to check whether the layout is intuitive. Below we listed some observations we made during this part of the test:

1. We asked Jeroen to upload a model. This went well
2. Jeroen found the interaction with the system very intuitive.
3. Jeroen commented on the confirm delete. He liked this feature a lot.
4. Jeroen commented the interface was nice and clear. Also for students to use.
5. Jeroen commented on the notifications. He likes those notifications for clarity.
6. Jeroen captured some data using the website. This went easy.
7. Jeroen was not sure if the links for downloading information updated correctly.

8. Jeroen commented there might be some problem with the data capture. It looks like the data from the past is the same data as live capture but with a different timestamp. It seems to be a problem with the ESP sending the same data each time

After this part of the test was concluded we asked a few questions about the experience. Those questions together with the corresponding answers are listed below.

1. Q: Did you find the system intuitive to navigate?
   A: Yes, very much
2. Q: What do you think about the overall styling and layout of the system?
   A: It looks great!
3. Q: Does the system give a finished and polished look?
   A: Yes, it looks great!
4. Q: Did you experience any difficulty with logging into the system?
   A: No, it was easy to log in.
5. Q: Was there anything unclear on the dashboard page of the system?
   A: There was nothing unclear.
6. Q: Were there any missing functionalities on the dashboard page?
   A: There are no missing features. A feature to see raw CSI information would have been nice. But that was not listed as a requirement.
7. Q: Was there anything unclear on the data capture page of the system?
   A: It was slightly unclear to see if the links for download updated.
8. Q: Were there any missing functionalities on the data capture page?
   A: No, very nice!
9. Q: How would you rate the entire experience of interacting with the system?
   A: Very good.

B.1.3 Conclusions

In this section we will list the things that we learned from performing this user test.

We learned that there might be an issue in the data capture, where even though the timestamp is different, the data is the exact same which is highly unlikely given that the data is csi data. We also learned that the user liked the entire interface, all of it was very intuitive and easy to grasp according to him. We did however learn that whether the links were updated when capturing data wasn't obvious.

B.2 User test 2

This user test was conducted on: 19/04/2024
This user test was performed by: Twan's Dad
This user test was supervised by: Twan Berg

B.2.1 Setting up

The user is a non-technical person with very basic knowledge in the field of computers. Not more than simple everyday computer use. Therefore we set up the system beforehand as this user has not enough knowledge in the field to follow the instrutions and set it up himself.

B.2.2 The system

Here we let the test subject browse the entire web-application. We do this with as little interference as possible. This is done to check whether the layout is intuitive. Below we listed some observations we made during this part of the test:

1. Logging in seemed intuitive. After telling the user the right password he managed to login in 1 try.
2. As this user was a complete outsider in terms of knowledge about the project I had to guide him quite a lot. Guiding in the sense that I had to explain what each button was for and what the purpose of it was. But the user indicated that after explaining the bare functionality and purpose of the system that all buttons did make sense. For example, the delete and upload buttons were very obvious. The play/pause button wasn't exactly clear, although the text of no results yet helped. However with more knowledge of the system the user indicated that a play/pause button was the most logical choice for activating and deactivating the specific model.

After this part of the test was concluded we asked a few questions about the experience. Those questions together with the corresponding answers are listed below.

1. Q: Did you find the system intuitive to navigate?
   A: Yes, there are only a few pages with a limited amount of buttons and things happening which made it very clear what every page and feature indicated. However, as I am not familiar with the product it did not mean much to see for example `Class 1` on the cards. It is hard to say as I have very limited technical knowledge what the system does exactly aside from the explanation I got during the test.
2. Q: What do you think about the overall styling and layout of the system?
   A: It looks great. Very simple and stylish with not any unnecessary things.
3. Q: Does the system give a finished and polished look?
   A: Mostly yes, the capture page seems a bit empty. On the other hand, it is very clear what the purpose of the page is and maybe this allows for extensions in the future.
4. Q: Did you experience any difficulty with logging into the system?
   A: No, managed to do it first try.
5. Q: Was there anything unclear on the dashboard page of the system?
   A: Only the play/pause button wasn't entirely clear but that had more to do with my limited knowledge about the product. And the meaning of each model card was not clear to me. Again this will tell you more if you are more experienced with the system and know what the uploaded models are for.
6. Q: Were there any missing functionalities on the dashboard page?
   A: Not in a position to give a good answer to that as I don't know the exact requirements.
7. Q: Was there anything unclear on the data capture page of the system?
   A: No, only two buttons that say what they do. The purpose of capturing data had to be explained to make sense of it. But after the explanation that it can be used for training new models this was clear.

8. Q: Were there any missing functionalities on the data capture page?
   A: Again not really in a position to answer this. Only thing I can think of based on the things that are there is to make a capture of a specific amount of seconds in the future. Similar to the feature where you can request so many seconds of the past.
9. Q: How would you rate the entire experience of interacting with the system?
   A: Great, it is nice to see an actual working application for once as most of the time I have no idea what is actually going on in the study. The website was smooth and fast and I experienced no issues.

B.2.3 Conclusions

From the second user test we can conclude two major points. First of all, the system seems intuitive. There are not too many buttons, pages or things going on in general which makes for a nice experience and a clear purpose of the system. Secondly, we got some confirmation that you do really have to be involved in the project and have to know what is going on and what the purpose of the system is to understand all the features and the output of these features. This is to be expected as it is quite a specific use case for a system like this and is not very obvious to non-technical or uninvolved uesers.

Appendix C

Manual

C.1 Prerequisites

When attempting to setup our system we assume a few things to be already working.

1. A working Raspberry Pi 5 with 32 or more GB of storage
2. A stable version of Rasbian OS is running on the RPi
3. An internet connection on the RPi
4. Access to the RPi over SSH
5. Knowledge on how to flash an ESP
6. Genereal knowledge regerding interacting with Raspberry Pi's, ESP32's and GitHub

When one of these presumptions is not correct. Refer to the official [Pi] or [Zha].

C.2 Setup

In this section we will look into setting up the project on a new Raspberry Pi 5. This should only be done once. After is has been done follow the start-up section to start the system repeatedly.

C.2.1 Software setup

In this section we will look into how to setup the software side of the project, ready for use on a Raspberry Pi. It will include instructions on how to clone the project into the Raspberry Pi and what files need to be edited in order to make the system run correctly.

C.2.1.1 Cloning the project from GitHub

There are two ways to clone the project from GitHub. Either via the terminal or via an internet browser on the RPi. We will explain here how to do it via the terminal.

1. Move to the directory in which you want to clone the project using

   ```
   cd /path/to/project
   ```

   This will create a `/DemenChair` folder with the project inside this directory on top of the directory navigated to.

2. Clone the project using

```
git clone https://github.com/TwanBerg/DemenChair.git
```

3. Follow the instructions in the terminal to authenticate yourself to GitHub

C.2.1.2 Setting variables

Here we will highlight some paths that need to be changed to the correct directories for the system to work.

1. Move to the root directory of the project using

   ```
   cd /path/to/project/DemenChair
   ```

2. Edit the docker-compose file using

   ```
   nano docker-compose.yml
   ```

3. In line 9 if the file replace

   ```
   <path_to_project>
   ```

   With the path to the root directory of the project so it becomes

   ```
   - /home/<pi-name>/path/to/project/DemenChair/uploaded_models:/code/app/uploaded_models
   ```

4. Exit the file by pressing: `Control + x`
5. Safe the file without changing the filename by pressing: `y` followed by `Enter`
6. Edit the model route using

   ```
   nano ./Backend-API/app/api/routes/model.py
   ```

7. Change the `uploaded_models_path` variable to

   ```
   "/home/<pi-name>/path/to/project/DemenChair/uploaded_models"
   ```

8. Exit the file by pressing: `Control + x`
9. Safe the file without changing the filename by pressing: `y` followed by `Enter`

C.2.2 Hardware setup

In this section, we will look at how to set up the ESP and the Raspberry Pi. We will also provide instructions on how to flash the code onto the ESP.

C.2.2.1 Setting up the ESP and the PI

In order to set up the system, we need to connect the Raspberry Pi 5 to a power source. Once this is done, the ESP can be connected to pi using a USB-C cable. Connect the ESP to one of the USB ports on the pi. In order to improve the collection of data, the antenna can be attached to the ESP.

The ESP can also be connected to the Raspberry Pi using Bluetooth. Instructions on how to connect the ESP and Raspberry Pi using Bluetooth is given below.

C.2.2.2 Flashing the ESP (cable)

In this section we will look at how to flash the code that collects the CSI data onto The ESP. It will include instructions on how to clone the GitHub project and what parameters need to be configured for successful collection of CSI data.

C.2.2.2.1 Cloning the GitHub Here, we will explain how to clone the GitHub repository using the terminal.

1. Open command line and move to the directory we want to clone the project in. This can be done using the command:

   ```
   cd /path/to/project
   ```

2. Once we are in the right directory, clone the GitHub using the command:

   ```
   git clone https://github.com/espressif/esp-csi.git
   ```

C.2.2.2.2 Configuring the parameters In this section we will explain what parameters need to be set before flashing the code onto the ESP.

1. Open command line and move to the directory we want to work in. This can be done using the command:

   ```
   cd ..path/esp-csi/examples/get-started/csi_recv_router
   ```

2. Now, we need to set the ESP as target. In order to do that, run the command:

   ```
   idf.py set-target esp32
   ```

   where the path is the folder where the GitHub was cloned to.

3. Once we are in the right directory, open menu configuration using the command

   ```
   idf.py menuconfig
   ```

4. Go to Component Configuration → WiFi and select the WiFi CSI(Channel State Information) option

5. Go to Example connection configuration and set the ssid and password to the ssid (demenchair) and password (demenchair) of the hotspot

Once the parameters are set, flash the code onto the ESP.

C.2.2.3 Flashing the ESP (Bluetooth)

C.2.2.3.1 Configuring the parameters In this section we will explain what parameters need to be set before flashing the code onto the ESP.

1. Open command line and move to the directory we want to work in. This can be done using the command:

   ```
   cd /DemenChair/Backend-ESP32/Bluetooth/
   ```

2. Now, we need to set the ESP as target. In order to do that, run the command:

   ```
   idf.py set-target esp32
   ```

3. Once we are in the right directory, open menu configuration using the command

   ```
   idf.py menuconfig
   ```

4. Go to Component Configuration → WiFi and select the WiFi CSI(Channel State Information) option
5. Go to Component Configuration → Bluetooth → Select Bluetooth
6. Once Bluetooth is enabled, go to Bluedroid Options → Enable BLE 5, BLE 4.2 features and Enable BLE high duty advertising interval feature
7. Go to Example connection configuration and set the ssid and password to the ssid (demenchair) and password (demenchair) of the hotspot
8. Go to Partition Table → Partition Table → Select Single Factory app (large)

Once the parameters are set, flash the code onto the ESP.

C.2.3 Finalizing setup

In order to finalize the setup of the project all that is left to down is to build the docker container. This should be done from the root directory of the project `/DemenChair` using the following command to build the docker image

```
docker compose build
```

C.2.3.1 Setting up the Rasperry Pi WiFi hotspot

The ESP32 sensor works by pinging a WiFi hotspot for CSI packets. This hotspot will be made available by the Rasperry Pi itself. We can setup this hotspot to always activate once the Raspberry Pi boots up.

1. The first step is to open the correct file so we can add a start up instruction. Run the following command to edit this file.

   ```
   sudo nano /etc/rc.local
   ```

2. The last line of this file is

   ```
   exit 0
   ```

3. We want to add the instruction to create the WiFi hotspot somewhere before this. So anywhere before the line `exit 0` we add the instruction

   ```
   sudo nmcli device wifi hotspot ssid <wifi_name> password <wifi_password> ifname wlan0
   ```

4. Save the file by pressing `Ctrl + X` and then `Y` and then `Enter`
5. Keep in mind that the name and password of this hotspot should be the same as the hotspot the ESP32 sensors connect to, otherwise the ESP32 won't establish a connection and won't send any data. Finally we need to reboot the Pi in order for these changes to take effect. We do this with the following command.

```
sudo reboot
```

C.3 System start-up

In this section we will discuss the steps that need to be taken to start up the system and how to access it. This step can only be performed after the software and hardware setups have been completed successfully. Once the setup of the system is complete we can move to the start-up phase of the system.

When the Raspberry Pi has been turned off and on again and has not been connected to either WiFi or Ethernet, the internal date-keeping of the Pi will not be correct. This should be fixed before the system is started. Either by connecting the Pi to WiFi manually, or temporarily connecting the Pi to an Ethernet cable. Both these actions will reset the internal timing of the Pi.

C.3.1 Starting the system

The final steps to run and use the system are as follows.

1. Start by connecting the WiFi hotspot of the Rasberry Pi
2. We start by accessing the Raspberry Pi over SSH. Ensure that the your laptop is connected to the same WiFi network as the Rasbperry Pi, that is the hotspot of the Raspberry Pi that we created in the previous step.

```
SSH <raspberry_pi_name>@raspberrypi.local
```

   The Pi we used in development had the name `grp20` and the password `asdfghjk`
3. Once we have access to the Raspberry Pi we need to navigate to the root folder of the project. This folder is named `DemenChair`. Use the following command to get there:

```
cd /path/to/project/DemenChair
```

4. Ensure that the directory you are now in is called `DemenChair` and contains a file called `docker-compose.yml`. This is the same folder to which you cloned the DemenChair Git Repository.
5. Then run the following command to start the project.

```
sudo chmod +x ./start.sh && ./start.sh
```

6. This will start the project and once the project is running we can connect to it. Finally, open up a browser and navigate to

```
http://10.42.0.1:3000
```

   to see and use the Web App and all it's functionalities. Ensure that you are still connected to the hotspot of the Raspberry Pi to access this page.

C.3.2 Stopping the system

One final important remark to make is how to stop the system or close the system and restart it if any issues occurs. When we want to stop and clean up the system we need to follow these steps.

1. Access the Raspberry Pi again over SSH. Ensure that you are connected to the hotspot of the Raspberry Pi

   ```
   SSH <raspberry_pi_name>@raspberrypi.local
   ```

2. Navigate to the root directory of the DemenChair project.

   ```
   cd /path/to/project/DemenChair
   ```

3. Stop and clean the Docker containers and stop the ESP program using

   ```
   sudo chmod +x ./stop.sh && ./stop.sh
   ```

4. To restart the project refer to the section `Starting the system`

C.4 Fine-tuning the System

Below we have listed some ways in which the system can be fine-tuned. Be carefully when tuning the system. If certain things are changed but not in all required places the system can experience problems! After any change is made, the docker image has to be rebuild from the root directory using

```
docker compose build
```

And the system needs to be restarted as explained in the previous chapter.

C.4.1 Possible changes

- The limit for the amount of data captures in the database can be changed. This is done in the file:

  ```
  DemenChair/Backend-API/app/api/controllers/capture.py
  ```

  In this file there are 2 variables called `capture_limit` and `capture_slack` located in lines 7 and 8 respectively. The `capture_limit` is responsible for the amount of captures that can be stored in the database. The `capture_slack` is used as an offset for when to start deleting entries from the database. When there are more entries than the `capture_limit` and `capture slack` combined, the oldest entries above the `capture_limit` are deleted.

- The limit for the amount of model classifications in the database can be changed. This is done in the file:

  ```
  DemenChair/Backend-API/app/api/controllers/result.py
  ```

  In this file there is a variable called `classification_limit` located on line 9. The `classification_limit` is responsible for the amount of classifications that are stored per model in the database.

- The colors used in the classification bar can be extended when necessary. This has to be done when a model is uploaded that has a larger than 6 class output. This can be done in the file:

  ```
  DemenChair/Frontend-SvelteKit/src/lib/components/dashboard/ModelCard.svelte
  ```

  In this file there is a list called `customColors` that is located on line 9. This array holds the colors for the classification bar. Colors can be added to this array and the existing colors can be changed to keep the styling consistent and readable.

- The 6-digit password for the login can be changed. This is in the file:

  ```
  DemenChair/Frontend-SvelteKit/.env
  ```

In this file there is a variable called `SECRET_PASSWORD_KEY` located on line 1. This holds the SHA-256 hash of the 6-digit pin. In order to change the pin, this variable can be set to a new SHA-256 hash of a 6-digit number. In order to get this hash you can use the [SHA] to generate the hash. The default password is set to '111111'

- The wifi name and password of the Raspberry Pi can be changed. Refer to the section `Finalizing setup - Setting up the Raspberry Pi WiFi hotspot` to see how to do this. When changing the WiFi network name and password we need to reflect these changes in the code flashed on the ESP32 sensors aswell. As these sensors use this WiFi network to capture CSI packets. Refer to the section `Hardware setup` to see how to set the WiFi network the ESP32 uses and how to flash the new code onto the ESP sensor.
- Using USB or Bluetooth. When switching between the usage of the ESP32's over USB or Bluetooth we need to change a few things. First of all, the code flashed onto the ESP32 needs to be updated. Refer to the section on `Hardware setup` for instruction. Secondly we need to change the variable in the Backend-ESP code to reflect the use of USB or Bluetooth. The file to update is

  ```
  /DemenChair/Backend-ESP32/main.py
  ```

  In this file a variable called `usb` on line 13 needs to be set to `True` if using ESP over USB and to `False` if using ESP over Bluetooth
- The ping rate of an ESP can be changed when the ESP will be used via cable connection. This can be done in the file

  ```
  /path/esp-csi/examples/get-started/csi_recv_router/main/app_main.c
  ```

  In this file there is a variable called `CONFIG_SEND_FREQUENCY`, located on line `32`. This can be changed to the desired ping rate. However we recommend staying within the range of 20-100 packets per second. Rates outside this range might not function as expected and were not tested for.
- Changing the number of CSI packets that need to be collected by the system before the active Neural Network models are run can be changed in the file

  ```
  /DemenChair/Backend-ESP32/main.py
  ```

  In this file there is a variable called `packets` located in line 16. This can be set to the number of packets to be collected. Keep in mind that this changes the input shape to the Neural Network models to `(None, 1, <packets>, 32)` and that the uploaded models should be updated accordingly. When changing the number of CSI packets, the type of connection and code on the ESP should be taken into consideration. For an ESP connected via Bluetooth we recommend a number of packets around 5. For an ESP connected via cable we recommend the ping rate of the ESP itself.

C.5 Troubleshooting

Below we have listed some common errors and issues we encountered during the development and how you can solve them.

One issue that can occur is that no results for active models are displayed. There are 2 reasons this can occur.

1. The first reason could be the ESP not receiving any data. The ESP can be a bit finicky at times. To solve this the easiest way is to stop the system. See the section `Stopping the system`. Then disconnect and reconnect the ESP sensor to the Raspberry Pi. We can test whether the ESP is reading data using the command

   ```
   cat /dev/ttyACM0
   ```

This will print out the serial output of the ESP. As long as this tells us that it is disconnected from the WiFi network we need to wait for a connection. Once a connection has been made it should start to print out CSI data. At that point we can restart the system. See section `Starting the system`

2. The second reason could be due to the internal communication of the different parts, front-end, back-end, and esp part, of the system. Some of this communication goes over requests made to the IP-address of the back-end hosted on the Raspberry Pi. The common default IP-address of the Raspberry Pi when hosting it's own WiFi hotspot is `10.42.0.1`. However, this does not always have to be the case and since this IP-address is static, we need to manually change it if this is not the IP-address of the Raspberry Pi. To check what the IP-address should be we can access the Pi over `SSH` (ensure the laptop is connected to the hotspot of the pi). Run the command `ifconfig` to see all configurations of the Pi and find a `IPv4` address most likely similar to `10.42.0.1`. Once we have this IP-address we need to change the code in 2 places to use this IP-address.

   (a) The first place is in the front-end code. The file we need to update is the `.env` file on the front-end. `/DemenChair/Frontend-SvelteKit/.env` In this file is a variable called `PUBLIC_BACKEND_IP_ADDRESS` on line 3. Replace the `10.42.0.1` in this file by the new IP-address

   (b) The second place is on the side of the ESP capture program. In the file `/DemenChair/Backend-ESP32/api.py` is a variable called `baseUrl` on line 5. Replace the `10.42.0.1` by the new IP-address

Bibliography

[Uni16] European Union. *General Data Protection Regulation GDPR*. 2016. URL: https://gdpr-info.eu/.

[Her21] Steven M Hernandez. *ESP32 CSI TOOL. Active CSI collection (Station)*. Accessed: 2024-04-10. GitHub. 2021. URL: https://github.com/StevenMHernandez/ESP32-CSI-Tool (visited on 04/16/2024).

[Wan+21] Ju Wang et al. "Unobtrusive health monitoring in private spaces: the smart home". In: *Sensors (Basel)* 21.3 (Jan. 2021), p. 864. DOI: 10.3390/s21030864. URL: https://www.mdpi.com/1424-8220/21/3/864.

[Auf22] Jean-Luc Aufranc. *ESP WiFi CSI Detects Humans with WiFi Signals Only, No Sensor Needed*. Accessed: 2024-04-02. CNX Software. Aug. 2022. URL: https://www.cnx-software.com/2022/08/08/esp-wifi-csi-detects-humans-with-wifi-signals-only-no-sensor-needed/ (visited on 04/16/2024).

[Alg23] Ministerie van Algemene Zaken. *Kamerbrief over nieuw arbeidsmarktprognose zorg en welzijn*. Apr. 2023. URL: https://www.rijksoverheid.nl/documenten/kamerstukken/2023/03/22/kamerbrief-over-nieuw-arbeidsmarktprognose-zorg-en-welzijn.

[Zha23a] Zhan Zhancheng. *esp-csi. CSI-receive and CSI-send*. Accessed: 2024-03-29. GitHub. 2023. URL: https://github.com/espressif/esp-csi/tree/master/examples/get-started/csi_recv_router/main (visited on 04/16/2024).

[Zha23b] Zhan Zhancheng. *esp-csi. Recv router CSI example*. Accessed: 2024-03-29. GitHub. 2023. URL: https://github.com/espressif/esp-csi/tree/master/examples/get-started/csi_recv_router/main (visited on 04/16/2024).

[Esp24a] Espressif Systems. *ESP WiFi API Reference. ESP-IDF Programming Guide*. Accessed: 2024-04-02. Espressif Systems. 2024. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/api-reference/network/esp_wifi.html (visited on 04/16/2024).

[Esp24b] Espressif Systems. *ESP32-S3. Product Information*. Accessed: 2024-04-03. Espressif Systems. 2024. URL: https://www.espressif.com/en/products/socs/esp32-s3 (visited on 04/16/2024).

[Ste24] Scott Stewart. *Understanding GAP and GATT profiles. What Is BLE, And How Do Its Related GAP And GATT Profiles Work?* Accessed: 2024-04-10. Cardinal Peak. 2024. URL: https://www.cardinalpeak.com/blog/what-is-ble-and-how-do-its-related-gap-and-gatt-profiles-work (visited on 04/16/2024).

[Cyp] Cypress. *Cypress testing framework*. URL: https://www.cypress.io/.

[Pi] Raspberry Pi. *Getting started with your Raspberry Pi*. URL: https://www.raspberrypi.com/documentation/computers/getting-started.html.

[SHA] SHA256. *SHA 256 hash generator*. URL: https://emn178.github.io/online-tools/sha256.html.

[Zha] Zhan Zhancheng. *ESP - CSI*. URL: https://github.com/espressif/esp-csi/tree/master/examples/get-started.