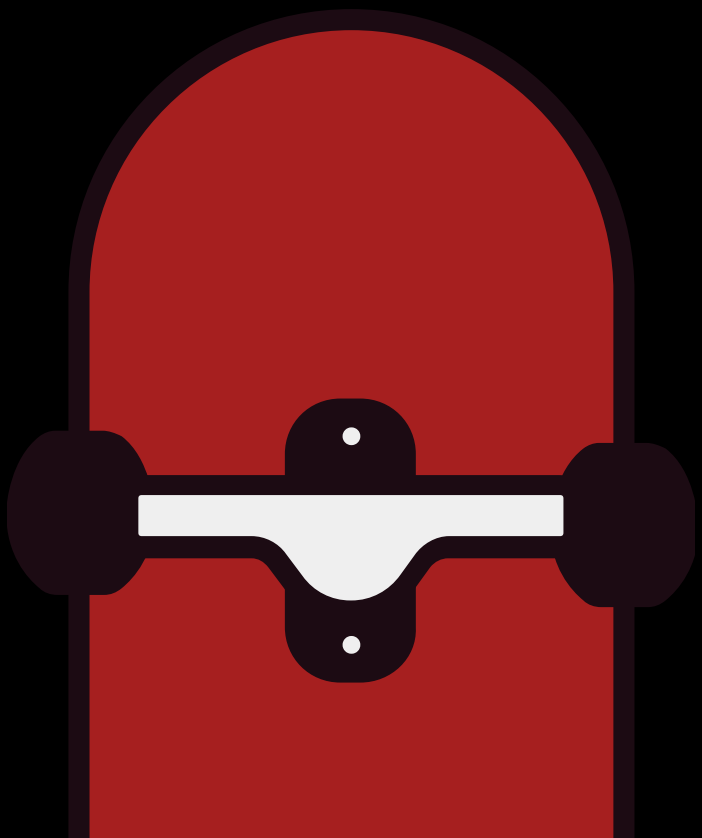
04
22

PROCESS REPORT

SKATERIXX



Introduction

The aim of this process report is to outline our progress and development regarding the Skatrixx Industry Project in the fourth semester of the Smart Mobile Specialization.

In the following chapters we will break down the different steps of design thinking we have followed and all the research methods we have applied in order to develop a skateboard system including a progressive web application.

Our team was tasked with creating a smart mobile application to help people learn how to skate by providing them with video tutorials for different types of tricks and using a combination of sensors to measure data from the skateboard and thus validate whether a trick has been performed successfully. We were given an initial timeframe of ten weeks that was later expanded to include the entirety of the semester allowing the team to make a more polished and ready to use product.

The team followed the Design thinking process and the Agile methodology, splitting the project into seven two-week sprints which allowed us to focus on different features throughout each sprint and finish them to completion before starting work on the other ones.

The design thinking process helped us continuously develop the application working on multiple iterations for each feature in combination with feedback that we received from our client and teachers during weekly feedback sessions we held with them. The different phases of the process have been listed and briefly described below.

Empathizing phase was used to gain an empathetic understanding of the problem at its core by conducting user research, observations and interviews.

Defining phase was used to accumulate and process the previously gathered information and draw conclusions out of it that would benefit our ideas and production process.

Ideation phase helped us generate ideas that would further develop the concept of the application. Focusing on the first two phases helped us think more “outside the box” and come up with ways to make the application feel more customizable and provide more incentives to our users for more frequent use – like achievements and rewards.

Throughout the design phase the team went through multiple iterations of the user interface and changed them accordingly based on feedback received in meetings with the client as well as user testing.

Implementation phase was the focus of the project since our team already had some available materials regarding the research and design. This phase was used to actually develop the front end and back end as well as connect the hardware parts together.

Design process

Empathize, define, solve:

The design thinking technique we followed was the Double Diamond strategy model. First, we started exploring/researching the problem (with the feedback of teachers and volunteers). Then we brainstormed possible solutions and selected the best ones to be implemented, followed by more testing and validation.

At the beginning of the project, we were given the research data made by the previous groups that worked on this project and analyzed all their documentation. The data in those reports included about 10 interviews, surveys, target audience definition, market research, and many more. We also had already defined features that were validated to be worked on and had to stick to all these guidelines.

To make the app our style and to improve the existing features, of course, we had to add up to those research documents, work on improving the existing features, and get deeper insights. Also, to fully understand our target users, we once again visited Area 51 to both do an observation and interview some more skaters.

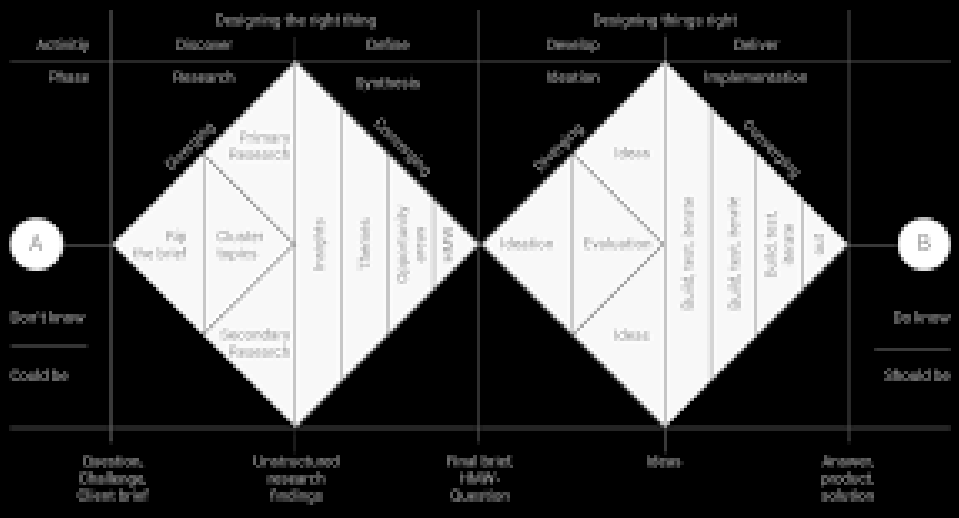
The workflow was about gradually building the prototype and at the same time getting user feedback on whether they would have a tough time using the product.

Overall, we have about 5 interviews (concept and get to know our potential users) and 5 testing scripts (usability). For the concepting interviews, we had a base level prototype that we showed to some volunteers in the area and asked whether they see meaning in the features and would they use it. Almost all the answers we got were positive.

The problems we faced were

- What can we do in order to improve the existing design?
- How can we give the skaters more freedom in using the app?
- How can we make the skaters feel more comfortable using the app?
- How can we make Skatrixx exciting and fun to use?
- How can we prevent users from getting confused when using our product?

How we worked is everyone contributing to this stage, as we formed the questions and problems definitions in long meetings discussing everything simultaneously.

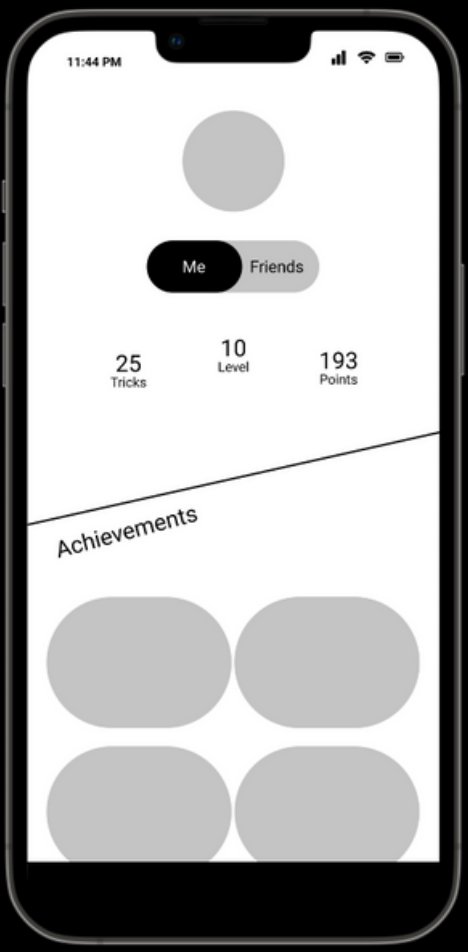


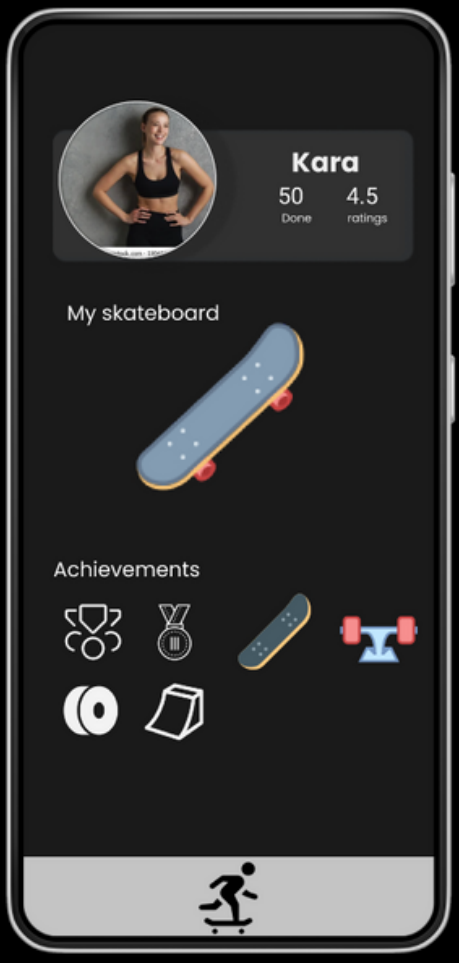
Desing

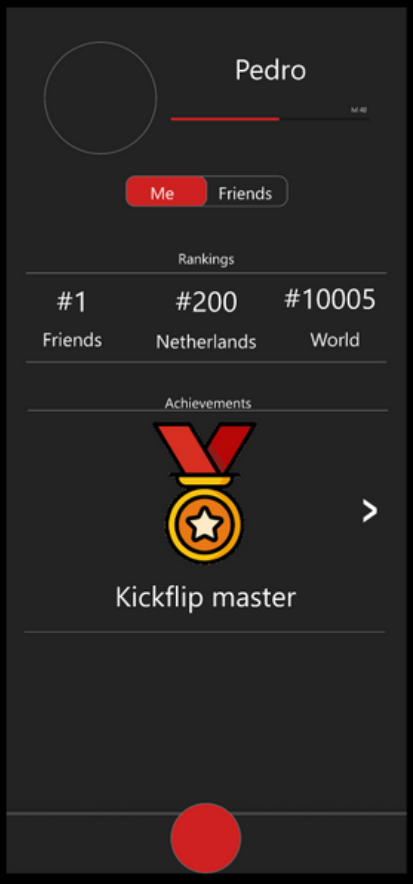
- **What do we use as software?**
  - ***Figma*** *is a tool where you can create prototypes*
- **What colours and typefaces do we use?**
  - *We used the colours that were provided to us, which were* ***black, white and red****.*
  - *After we got all the information that we needed from the previous groups as well as made our research we started doing the design part. One of the requirements of the client is not to change the colours and typeface and to stick to what we had already.*

- **What did we do?**
  - ***Wireframes*** *- Everyone in the group has the task to come up with some ideas about how the design looks like as we did low-fidelity prototypes and more specifically wireframes*
  - ***High-fidelity prototype****- at the end we combined all the ideas and made them into one prototype. We wanted the app to look modern because we felt that the previous app looked old-fashioned, so we decided to include a gradient. However, our team did some testing and gathered feedback that it does not look playful and appealing, so we decided to put some background images. After we did it, our team believed that it makes the design way more interesting and lively as well as added some depth to the app.*
- **What is our goal?**
  - *To e a user-friendly, playful, and meaningful app for skaters, where they can learn but also compete with their friends.*

Wireframes

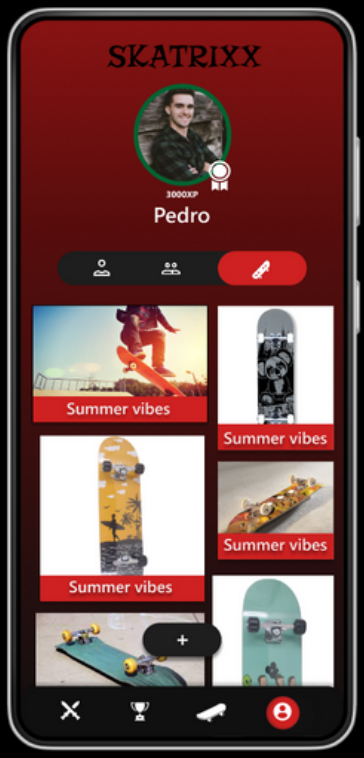

11:44 PM
Me
Friends
25
Tricks
10
Level
193
Points
Achievements


Kara
50
Done
4.5
ratings
My skateboard
Achievements

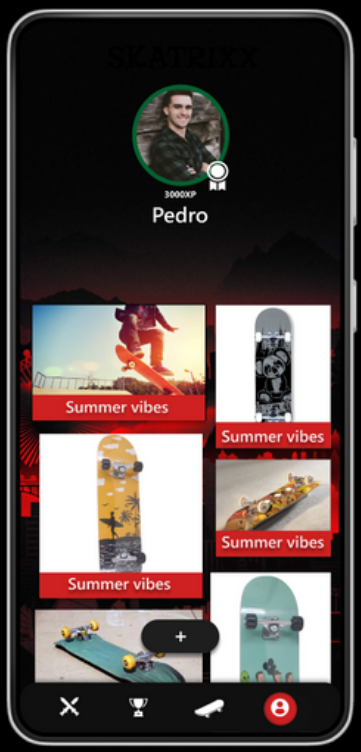

Pedro
Me
Friends
Rankings
#1
Friends
#200
Netherlands
#10005
World
Achievements
Kickflip master

Prototypes

previous prototype



prototype - v1



prototype - v2

Implementation

In the scope of the first phase of working over the Skatrixx project, we considered doing our server that will provide our client-side Application with RESTful APIs. So, our software pack consists of two servers, one for the client-side and one for the server-side. In the section below we will explain our Software design decisions, the database approach, dependencies we used and the APIs we created for communication with the hardware module (for which we will explain more in the Hardware section below)

Software

General overview of the servers and technologies we used for developing the application:

*For the **Frontend** we implemented:*
- React JS – JavaScript Library
- React Native – dependencies
- Pure CSS – for the design part

*For the **Backend** we implemented:*
- NodeJS – server
- Express JS – a framework for APIs providing

For the **Database** we used:
- Cloud-based MongoDB Cluster that is directly connected to the Express server.
- Connection link: https://data.mongodb-api.com/app/data-ncpjx/endpoint/data/beta

For the **Hardware** module we used:
- Pure C – on Arduino IDE
- External libraries for reading data from the sensors. (More information in the Hardware section)

Additionally, we implemented **Google Authentication:**
- Google Firebase - provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app.

Hosting services:
For hosting the **Backend-Server** we used:
- Heroku - a container-based cloud Platform as a Service (PaaS). Developers use Heroku to deploy, manage, and scale modern apps. / Using Heroku-20 stack in Europe Region

For hosting the **Frontend-Server** we used:
- Hera server - ...

Description of the backend structure:

For the structure of the Backend, we used a three-layered architecture so that we could separate the concern easier and exclude most of the complicated logic into another service/business layer. Each layer has a specific set of responsibilities that are clearly defined and easy to grasp. Each layer accesses the layer below it, never above it. Serving a request touches each layer starting from the top, travelling down, and then resurfacing back to the topmost layer.

Image of the layers' structure!

- **Models:** This layer is the schema definition of the object model we used.
- **Services:** This layer performs the complicated logic of the system. They validate inputs against predefined rules and call other services in the Service layer. If they need to talk to outside systems, they use the Integration layer to do that.
- **Routes:** This layer is a container that creates router handlers. In other languages, this layer may be called Controller. The routes layer also validates input requests from both the PWA and the hardware module.

Dependencies integrated:

- For the **Backend**, we integrated: CORS, dotEnv, mongoose, nodemon, socket.io, sockjs, typescript, ws
- For the **Frontend** we integrated: axios, firebase, react-notifications, react-qr-code, react-qr-reader, socket.io-client, sockjs-client

Environment:

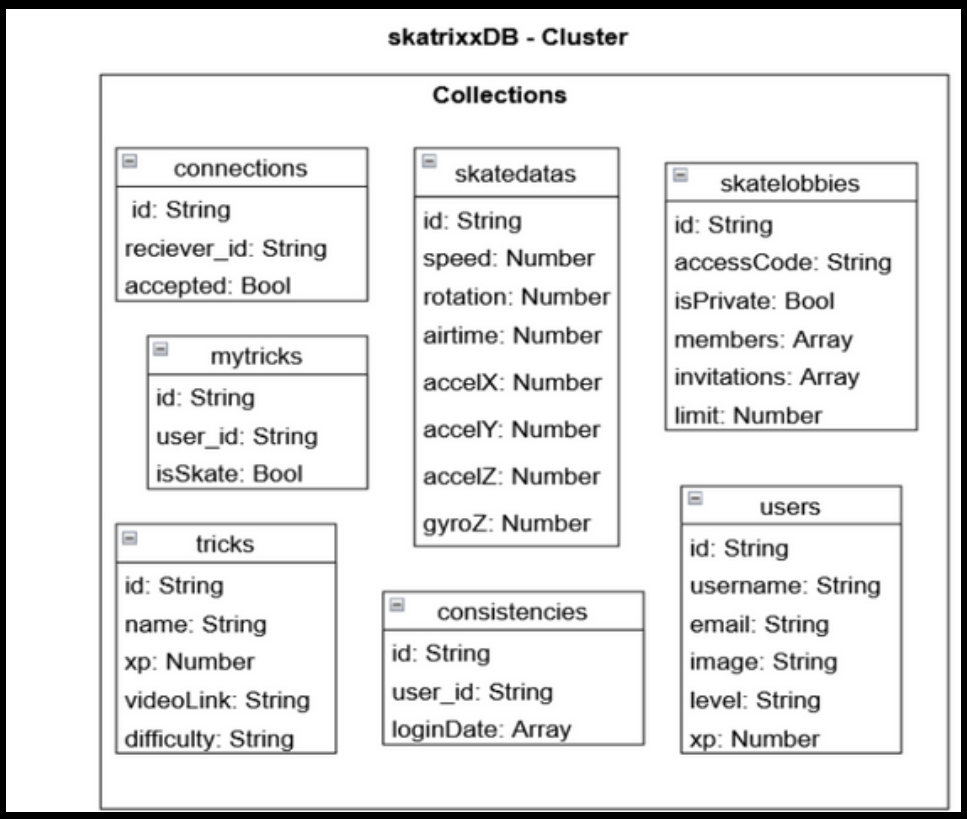
For the scope of our Industrial project as development environment we used PWA, so that the application we are building to be compatible in a mobile environment as it was required at the beginning of the project. The power of PWA environment is that you can run the application in the browser or as a desktop.

Below, we will provide screenshots of our service-worker and the manifest of the application.

```
const CACHE_NAME = "version-1";
const urlsToCache = ['index.html', 'offline.html'];

const self=this;
//instal SW
self.addEventListener('install', (event)=>{

    event.waitUntil(
        caches.open(CACHE_NAME)
        .then((cache)=>{
            console.log("Opened cache");

            return cache.addAll(urlsToCache);

        })
    )
});

//listen for requests
self.addEventListener('fetch', (event)=>{
    event.respondWith(
        caches.match(event.request)
        .then(()=>{
            return fetch(event.request)
            .catch(() => caches.match('offline.html'))
        })
    )

});

// Activate the SW
self.addEventListener('activate', (event)=>{
    const cacheWhiteList = [];
    cacheWhiteList.push(CACHE_NAME);

    event.waitUntil(
        caches.keys().then((cacheNames) => Promise.all(
            cacheNames.map((cacheName)=> {
                if(!cacheWhiteList.includes(cacheName)){
                    return caches.delete(cacheName);
                }
            })
        ))
    )
});
```

```
{
    "short_name": "SkatrixxApp",
    "name": "Skatixx application",
    "icons": [
        {
            "src": "/images/logo.png",
            "type": "image/png",
            "sizes": "1024x1024"
        }
    ],
    "start_url": ".",
    "display": "standalone",
    "theme_color": "#000000",
    "background_color": "#ffffff"
}
```

Database Cluster Diagram



Hardware

Introduction

For this project, we started by analyzing what type of hardware is needed and what hardware other groups before us have used.

- **What have the previous group used as hardware?**
  - The previous group decided to use the ESP32 module as the main device managing the hardware system. For communication between the hardware and the mobile application, they have used a Bluetooth connection. The module for reading skate data they have used is MPU9250.
- **What have we used as hardware?**
  - We decided to use a slightly different hardware structure based on our research and brainstorming on how to make a good and useful skate system.
  - We have used ESP32 as the main device that is responsible for managing our hardware part of the system. ESP32 has wireless connectivity through Wi-Fi and Bluetooth which ease the hardware structure and avoids adding additional Wi-Fi modules for example.
  - For the connection between the hardware and the mobile application, we have used Wi-Fi as we discuss this is a good option for saving new data from the sensors on the server(backend) and then rendering it in the frontend by accessing the backend API.
  - We have chosen MPU9250 as the main module for reading skate data using an accelerometer and gyroscope for getting specific data used for skate attempt analysis. We have an Ultrasonic sensor for height measurement as well.

ESP32

- ESP32 is a low-cost and low-power system on a chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth that is smaller than Arduino Uno and can be programmed in C/C++ using Arduino IDE.

Hardware system consist of:

MPU9250

- MPU9250 is a module consisting of an accelerometer, gyroscope and magnetometer. It is a small and nice choice if you do not have enough space.

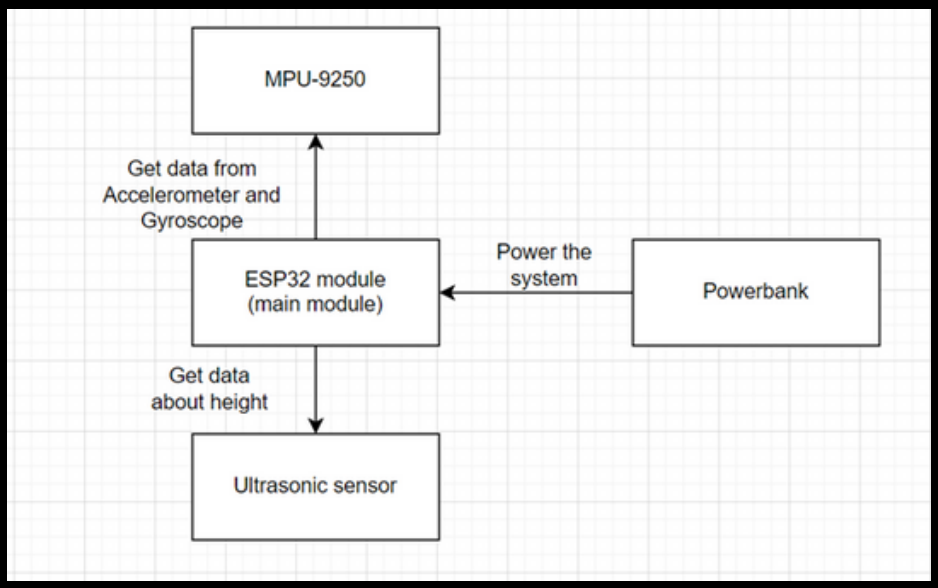
Ultrasonic sensor

- The ultrasonic sensor is a distance sensor. This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. Each HC-SR04 module includes an ultrasonic transmitter, a receiver and a control circuit.

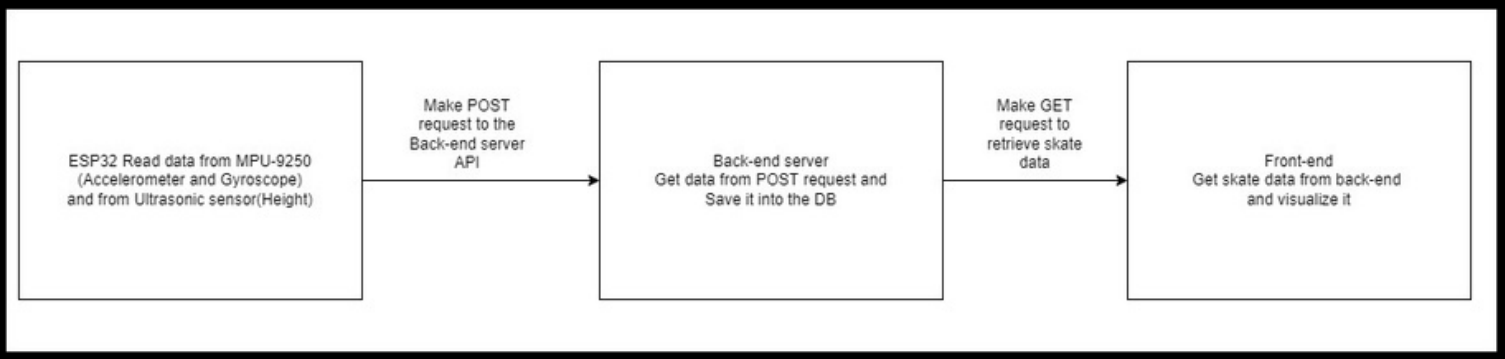
Power bank

- We have used a power bank to power the ESP32. The battery provides 5V output and we can charge the ESP32 through the USB connector.

Hardware schema



Hardware flow



Conclusion

To sum up, our group went through a couple of design thinking iterations which helped with improving the previous groups' concept that we used for reference. The implementation process went with a lot of researching and going through new and interesting coding methodologies. Which was realised in our application, puting a lot of effort and inteligence.

Our final opinion about the project is very positive. We are very happy with the final result and what we were able to deliver for this first phase of the project.

For the future, we are thinking about improving the user experience with the game and implementing other exciting and inovative features which would benefit our Skatrixx project and it's concept.