Nikodem Drabik ,Thomas Cogzell, Joseph Troughton

   Farfields John McDermott

[Project 2003]

Contents

[Project Requirements. 4](#_Toc186751130)

[Basic Requirements: 4](#_Toc186751131)

[Our Understanding: 4](#_Toc186751132)

[Mission Statement 5](#_Toc186751133)

[Value We provide 5](#_Toc186751134)

[Application Purpose POV 5](#_Toc186751135)

[Additional Features that would benefit the client: 5](#_Toc186751136)

[User Analysis/ Possible Users: 6](#_Toc186751137)

[Contractor/Device Installer: 6](#_Toc186751138)

[Supervisor / IT administrator 6](#_Toc186751139)

[Hobbyist 7](#_Toc186751140)

[Benefits of IoT lifecycle Management 7](#_Toc186751141)

[Cost Saving 7](#_Toc186751142)

[Improved Efficiency 8](#_Toc186751143)

[Improved Security 8](#_Toc186751144)

[Promotes Regular Maintenance 8](#_Toc186751145)

[Reducing E-Waste 8](#_Toc186751146)

[Project Roles 9](#_Toc186751147)

[Stakeholders 10](#_Toc186751148)

[Client 10](#_Toc186751149)

[Users 10](#_Toc186751150)

[Team 10](#_Toc186751151)

[Project TimeLine 11](#_Toc186751152)

[Research 13](#_Toc186751153)

[Statement 13](#_Toc186751154)

[Critical Requirements for functionality 13](#_Toc186751155)

[Existing Solutions 13](#_Toc186751156)

[AWS IoT 13](#_Toc186751157)

[Microsoft Azure IoT 14](#_Toc186751158)

[Our Difference 15](#_Toc186751159)

[QR Code readers 15](#_Toc186751160)

[How Do QR Code Readers Work? 15](#_Toc186751161)

[Open-source solutions 16](#_Toc186751162)

[ZXing ("Zebra Crossing") barcode scanning library for Java, Android 16](#_Toc186751163)

[ML Kit by Google 17](#_Toc186751164)

[Conclusion 17](#_Toc186751165)

[Final Choice 17](#_Toc186751166)

[React Expo Camera 17](#_Toc186751167)

[Summary 18](#_Toc186751168)

[LoRaWAN and LoRA RF 18](#_Toc186751169)

[Front End Design 20](#_Toc186751170)

[Iteration One: 20](#_Toc186751171)

[Iteration Two: 24](#_Toc186751172)

[Prototype 1 25](#_Toc186751173)

[Technical Analysis 31](#_Toc186751174)

[Initial Security analysis of applications and threats: 31](#_Toc186751175)

[Important Laws: 32](#_Toc186751176)

[How it applies: 33](#_Toc186751177)

[Risk Management 34](#_Toc186751178)

[Quality Assurance 35](#_Toc186751179)

[What is it? 35](#_Toc186751180)

[Why it’s needed? 35](#_Toc186751181)

[How will it be achieved? 35](#_Toc186751182)

[KPI’s / Testing 37](#_Toc186751183)

[What is it? 37](#_Toc186751184)

[Our KPI’s 37](#_Toc186751185)

[Testing: 38](#_Toc186751186)

[Approval Process 39](#_Toc186751187)

[Step 1 – Team Review 39](#_Toc186751188)

[Step 2 - Client Review 40](#_Toc186751189)

[Step 3 – Final Approval 40](#_Toc186751190)

[Firebase: 40](#_Toc186751191)

[Database Explained: 40](#_Toc186751192)

[Database Low Fidelity Plan. 40](#_Toc186751193)

[Features to be used: 40](#_Toc186751194)

[To look out for: 41](#_Toc186751195)

[Summary 41](#_Toc186751196)

[Overview 41](#_Toc186751197)

[Backend Architecture 42](#_Toc186751198)

[Budget Analysis - Firebase 43](#_Toc186751199)

[Proof of Concept 47](#_Toc186751200)

[Concept/Testing 47](#_Toc186751201)

[Rules for test database used of Testing 47](#_Toc186751202)

[Test Table Designs: 48](#_Toc186751203)

[Database CRUD: 49](#_Toc186751204)

[Project 2003 GitHub Repository 53](#_Toc186751205)

[Additional Resources 53](#_Toc186751206)

[Additional Research 53](#_Toc186751207)

[Topic - AI (Late Stage idea) 53](#_Toc186751208)

[Project 2003 Approval Signature 54](#_Toc186751209)

[Team Members 54](#_Toc186751210)

[Nikodem 54](#_Toc186751211)

[Joseph 54](#_Toc186751212)

[Thomas 54](#_Toc186751213)

[Client 54](#_Toc186751214)

[Follow Structure this: 55](#_Toc186751215)

# Project Requirements.

## Basic Requirements:

A database app to manage the lifecycle of connected IoT assets from cradle to grave. As a device moves from manufacture into installation, operation, and eventual removal and recycle there is a need to record information about the device, e.g. serial number, owner, location, installation, connection, maintenance etc. The project objective is an app that records device associated information into a database to record its life history as a 'source of truth' for use by device manufacturers and business users. Emphasis on this project would be a UI experience that is more oriented towards the workflow of the lifecycle.

1)      Use of Firebase to manage data.

2)      API calls preferred to abstract out services so recommended to use microservice architecture.

3)      The project should be extendable to multiple clients running off the same database.

4)      There should be option for data to be stored on client side or client-side API keys, but functionality on cloud.

5)      Stretch goals - think scalability from 5 devices to 1000 or more.

## Our Understanding:

A structured way to keep track of devices in an IoT-based application allows for the user to monitor known statistics and update data for each device in whatever format they like. It should be flexible by allowing dynamic grouping and user-saved documents and files while also being structured enough to serve a large customer base.  – **technical aspect of Database**

A mobile interface that allows businesses and contractors to monitor and maintain their IOT devices in the field and update each device in real time. Each device should have a GPS coordinate, last serviced, and notes about the device and how it should be maintained in the future. - **implement a Real time Database**

The User Interface should be easy to understand for users of different tech literacy levels as this could be used by a manner of people tracking servers to cow trackers. **- the application of User Interface**

## Mission Statement

To provide a software solution to manage and track the Life Cycle a wide range of IoT devices in one simple application.

Providing a seamless, scalable software solution for managing and tracking IoT device lifecycles, ensuring security, sustainability, and ease of use.

## Value We provide

With the shutdown of the Amazons IOT solution we come to fill in the market gap to provide a solution for the specific use case scenarios that our possible clients. We provide a simple solution to tackle the issue of not being able to track IoT devices while they are deployed in the field and the remain aspects of its Life Cycle.

Customers can track all their IoT device from Manufacturing to Decommission in one simple application without the need of entering a single ecosystem. With the use of Firebase, you will be able to access this data from anywhere with internet connection in real-time. Our solution simplifies the tracking and maintenance efforts required by our customers while enabling maximising operational longevity and efficiency.

## Application Purpose POV

Create software that allows a business, and individuals track their IOT devices lifecycle from manufacture to installation, operation, maintenance, and eventually, recycling if possible.

Whether managing a handful of devices to scaling to thousands, our solution should be able to adapt to meet the diverse needs of clients, offering flexibility in our client-side integration as well as on our database.

### Additional Features that would benefit the client:

Maintenance scheduling

QR code scanning/reading to easily access device information.

LoRaWAN and LoRA RF communication with devices \*more information found under LoRaWAN and LoRA RF

## User Analysis/ Possible Users:

### Contractor/Device Installer:

#### Who are they

This person would be working for the company who owns the device.

The main uses of the app for this user would be:  
 Modify Maintenance Logs

Adding Devices to company database.

Removing Devices from database

#### Technical Ability

The user may not be the most technically inclined to navigate a complicated app. They can range for 18 years olds up to 65 years as it’s likely to be a manual labour position. The app must be easily teachable to all users.

### Supervisor / IT administrator

#### Who are they

This person would be managing all the devices from a centralised location like and office.

They will be in charge of ensuring all devices are up to date and if not ensure a contractor is sent to fix the issue at hand. They must have access to all devices that are in their network and no other network.

#### The main use of the app for this user?

This user will mostly be using the manage device section of the app as they are not likely to be creating new user in the device. They may be centralised in one location or be moving from one location to another so they must have remote access like a contractor to their service.

They may need to create updated or delete devices in case of issues in the real world and possible remoteness of the devices not allowing for real time updates to the database or fix human errors created by others.

#### Technical Ability

This user can be a wide range of of age however often are older due to the seniority of the position meaning that the app mist be tailored to them and their specific uses.

#### Problems this app may solve for them

An active view of most of there deployed devices. It will allow them to manage them better and be aware of their status at moment’s notice. It should fix issues of large amount of paper-based documentation that need to be looked through to find the status of a device.

#### Important features they would benefit from?

Notifications – these would provide reminders for maintenance schedules.

View of all current devices and their current stages in their lifecycle and last maintenance date.

### Hobbyist

#### Who are they

This person enjoys experimenting and utilising and messing with many electronic devices and have many devices that they need to maintain in their home for example routers, servers, NAS devices.

#### The main user of the software

They would like to keep record of all their devices they have deployed. The reason for this may be that they want to keep everything up to date with the lates driver and software updates. For this they may benefit for man app that they input data and informs them every 6 months to update their systems software.

#### Technical Ability

This type of user will be very skilled in this area as this is their passion they are likely to be much more technically inclined than the average person with tech and likely want to modify the app to be more applicable to their specific user case scenario.

#### The main issue this software would solve for them.

This software would solve the issue of maintenance scheduling of devices and life cycle of devices as all devices have a expected life cycle. This would help this users be on top of their device servicing like a MOT reminder for road users. It will help them ensure all devices are in their best shape.

# Benefits of IoT lifecycle Management

Effective management of the IoT device lifecycle is essential for ensuring security, reducing costs, and maximizing the longevity of devices. A well-structured lifecycle management approach helps organizations monitor, secure, and maintain their IoT devices throughout their entire lifespan.

## Cost Saving

Proper Lifecycle management helps organisations avoid unplanned repair and replacement costs. Through continuous monitoring, potential issues can be identified early, allowing businesses to address problems before they escalate, and plan for the timely replacement of outdated devices.

## Improved Efficiency

IoT device lifecycle management enables better resource utilization and streamlines operations. With a clear overview of device statuses and locations, businesses can optimize device usage and improve overall productivity, reducing unnecessary downtime and maximizing efficiency.

## Improved Security

With constant upkeep of devices and routine maintained it improves security. This is because it helps reduce the risk of security breaches and attacks by keeping track of all devices and ensuring they are not vulnerable.

## Promotes Regular Maintenance

Regular maintenance is crucial for ensuring that IoT devices continue to function optimally throughout their lifespan. By promoting a proactive approach to maintenance, organizations can address minor issues before they become major problems, reducing downtime and improving operational efficiency. This leads on to the next part

## Reducing E-Waste

An effective maintenance plan can significantly reduce the premature redundancy of IoT devices. By ensuring that devices are properly maintained and regularly updated, their lifespan is extended, thus preventing the need for early replacement. This approach not only supports organizations in achieving their carbon emission reduction goals but also contributes to environmental sustainability by minimizing e-waste. Additionally, regular maintenance may enable the recycling of usable components, further reducing waste and promoting the responsible reuse of materials.

# Project Roles

Tom-

**Front end/ UI/Flutter**

Task: To design and manufacture a UI for the client

Niko-

**Project Oversight, Front-end/Back-end assistance**

Task: To work with all team members to help with the deliverables. Assist in the deployment of UI and its communication with Firebase API

Joe-

**Firebase/Backend, Security of Project**

Task: To design and manufacture Firebase infostructure. Ensure that it is secure to be used by the clients.

# Stakeholders

## Client

In this project Farfields represented by John McDermott is our client that is going to be with us thought the year helping to be the main stakeholder providing critical feedback and guidance during the development. He will ensure that what the outcome is aligns with the key deliverables he expected

## Users

These are the people that we are creating the application for. They are the people who are likely going to be using this software. Understanding this Stakeholders needs and wants in critical for the projects success; due to the fact they are the centrepiece for this project.

## Team

Finaly the development team that Compromises of Niko, Tom ,Joseph who are developing this software. The team works closely with the client to ensure that the project aligns with the outlined objectives and expectations.

Each team member brings valuable qualities in the help of developing a high quality product.

# Project TimeLine

Project Start- October 25:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **At Risk?** | **TASK NAME** | **FEATURE TYPE** | **RESPONSIBLE** | **STORY POINTS** | **START DATE** | **END  DATE** | **DURATION  in days** | **STATUS** | **PRIORITY** | **COMMENTS** |  |
|  |  | Sprint Week 1 |  |  |  | 11/08 | 11/29 | 28 |  |  |  |  |
|  |  | Planning application |  | Team |  |  |  |  | Complete | High | To plan what type of software we will implement |  |
|  |  | User Analysis |  | Team |  |  |  |  | Complete | High | Analys the users of the app |  |
|  |  | Sprint Planning |  | Team |  |  |  |  | Complete | High | Plan future sprints |  |
|  |  | Sprint Week 2 |  |  |  |  |  |  |  |  |  |  |
|  |  | Plan Database | Main Database | Joeseph - Niko |  | 9/12/2024 | 16/12/2024 |  | Needs Review | Medium | Create database |  |
|  |  | Plan UI | Main UI | Tom - Niko |  | 09/12 | 16/12/2024 |  | Complete | Medium |  |  |
|  |  | Create Write Up |  | Niko - Team |  | 09/12 | 16/12/2024 |  | Overdue | Medium |  |  |
|  |  | Begin Presentation |  | Niko - Team |  | 09/12 | 16/12/2024 |  | Overdue | Medium |  |  |
|  |  | Sprint Week 3 |  |  |  |  |  |  |  |  |  |  |
|  |  | Implement Firebase create C++/C CRUD command files | Main Database | Team |  | 08/01 | 28/01/25 |  | Not Started | High |  |  |
|  |  | Implement Basic UI in flutter | Main UI | Team |  | 08/01 | 28/01/25 |  | Not Started | High |  |  |
|  |  |  |  |  |  | 08/01 | 28/01/25 |  |  |  |  |  |
|  |  | Sprint Week 4 |  |  |  |  |  |  |  |  |  |  |
|  |  | Create API for firebase and flutter connection | Main Database | Niko - Joseph |  | 29/01 | 16/02 |  | Not Started | High |  |  |
|  |  | Create Tracker Page to see items in detail | Main UI | Tom - Niko |  | 29/01 | 16/02 |  | Not Started | High |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | Sprint Week 5 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

Project End- May 6:

# Research

## Statement

The project aims to develop a database application to manage the lifecycle of IoT devices.

**As stated, it requires:** Lifecycle Tracking

User oriented UI

Firebase Database

Firebase API based microservice architecture.

## Critical Requirements for functionality

## Existing Solutions

### AWS IoT

Amazon states “the only cloud vendor that combines data management and rich analytics to create easy-to-use services designed for high-volume IoT data”

AWS IoT Is the largest cloud vendor for data management and analytics that provides real time analytics of the connected devices.

They state that is built upon a secure and proven cloud infrastructure and IoT network that scales to billions of devices

#### Pros

Scalability

Integrated with AWS services

Strong Security practices

#### Cons

Expensive for small-scale users with budget constraints

Complex to maximise to its full potential

Very reliant on the AWS ecosystem



[image by Amazon]

In the image above it shows the dashboard for the LoRaWAN system to what seems to be a small office space indicating occupied desks and rooms, open doors and two thermometer readings.

### Microsoft Azure IoT

A collection of Microsoft managed cloud services that allows the user to monitor and control their IoT devices at a scale. It specializes in providing comprehensive tools for device analytics. It is meant to seamlessly integrate with Microsoft tools included Power BI and Dynamics 365 which are two analytics tools that help visualize data for users and customers.

#### Pros

Scalable

Integrated with Azure AI services

Built in IoT Security features

#### Cons

Locked into Microsoft Eco-system

Price prohibits small scale or experimental projects

### Our Difference

Our proposed solution offers several unique advantages that set it apart from the existing solutions, such as AWS IoT and Microsoft Azure IoT. These differences are aimed at making our application more accessible, flexible, and user-centric:

1. While AWS and Azure focus heavily on analytics and scalability, our solution emphasizes lifecycle tracking. From manufacture to maintenance and eventual recycling, our app provides tools specifically designed to support every stage of an IoT device's life. Maintenance scheduling, QR code integration, and real-time database updates are just some of the features tailored to the operational needs of end users.
2. Unlike AWS IoT and Azure IoT, which are deeply tied to their respective ecosystem at the roots, our solution is platform-agnostic. This ensures that businesses and individuals can integrate our software into their workflows without the need of being locked into a specific platform. This provides greater flexibility and cost-effective alternatives to software solutions. However, it is based on the Google Cloud platform it is not tied to its services and should be detachable to other database solution.
3. Our application prioritizes a user-oriented interface that caters to varying levels of technical expertise. By focusing on intuitive design and streamlined workflows, we ensure that all stakeholders can use the app with minimal training. This emphasis on usability makes our solution stand out compared to the more complex dashboards offered by competitors.

# QR Code readers

## How Do QR Code Readers Work?

1. **Scanning**: The QR code reader captures an image of the QR code using a camera or a sensor.
2. **Processing**:
   * The software identifies the unique patterns in the QR code: black and white squares arranged in a grid.
   * Alignment markers, timing patterns, and the quiet zone around the code help locate and orient the code for decoding.
3. **Decoding**:
   * The code contains data encoded in a binary format.
   * Error correction algorithms, such as Reed-Solomon coding, ensure accurate decoding even if parts of the code are damaged or obscured. Allows for up to 30% of the code to be damaged.
   * The binary data is converted back into readable text, URLs, or other information.
4. **Action**: The reader may display the decoded information or perform and action like opening an app or a file

## Open-source solutions

### ZXing ("Zebra Crossing") barcode scanning library for Java, Android

ZXing is an open-source library developed for barcode image processing. Originally written in Java, it supports multiple barcode formats and is extensively used in Android and Java-based projects.

#### Features:

* **Support for Multiple Barcode Formats**:
  + **1D Formats**: EAN-8, EAN-13, UPC-A, UPC-E, Code 39, Code 93, Code 128, ITF, Codabar.
  + **2D Formats**: QR Code, Data Matrix, Aztec, PDF417.
* **Cross-Platform Support**: It is primarily for Java and Android but has ports in other languages (e.g., C++, JavaScript).
* **Integration**: Easily integrates into Android projects via the zxing-android-embedded module.
* **On-Device Processing**:
  + All can happen locally on the device, ensuring privacy and faster response times.

**Applications:**

* **HTML5 QR Code**:
  + The library powers projects like [html5-qrcode by mebjas](https://github.com/mebjas/html5-qrcode), which provides a client-side solution for QR code scanning in web applications.

https://github.com/mebjas/html5-qrcode

**GitHub Repository:**

https://github.com/journeyapps/zxing-android-embedded

* + This library provides an Android wrapper for ZXing, making it easier to integrate and customize with android apps.

### ML Kit by Google

ML Kit is a software solution by Google that offers machine learning-based tools for image and text recognition. Among its capabilities is barcode scanning, which includes QR code reading.

**Features:**

* **Wide Barcode Format Support**:
  + **Linear (1D) Formats**: Codabar, Code 39, Code 93, Code 128, EAN-8, EAN-13, ITF, UPC-A, UPC-E.
  + **2D Formats**: QR Code, Aztec, Data Matrix, PDF417.
* **On-Device Processing**:
  + All processing happens locally on the device, ensuring privacy and faster response times.
* **Cross-Platform**:
  + Available for both Android and iOS, with integration in native and Flutter applications.
* **Ease of Use**:
  + Provides pre-trained models, removing the need to build custom models for barcode and QR code recognition.

## Conclusion

**ZXing** and **ML Kit** both provide exceptional solutions for QR code and barcode scanning, each tailored to distinct use cases:

* **ZXing's Library** is ideal for developing a **custom firmware solution** that requires support for an **extensive barcode library** and **offline processing**. Its open-source nature makes it highly customizable and particularly suited for applications where **independence from external services** is critical.
* **ML Kit by Google** is better suited for a **quick, simple, and reliable solution**, offering robust **cross-platform support** and **seamless integration** with other Google services. It is an excellent choice for developers who prioritize ease of use and want to quickly implement barcode scanning functionality in their applications.

## Final Choice

### React Expo Camera

* **Seamless Integration**: The Expo Camera API works seamlessly across both Android and iOS platforms, eliminating the need for separate implementations for each operating system.
* **Consistent Performance**: Provides a uniform interface and experience, ensuring that the QR code scanner behaves the same regardless of the device.
* **Camera Module QR scanning**: The camera module has a built-in QR code scanning capability.
* **Privacy and Offline Use:** QR code processing done locally on the device, provides more protection to the users privacy and allows offline functionality.

### Summary

Overall, the integration of React Expo Camera with React Native is seamless and user-friendly, making it an excellent choice for developers. It simplifies the development process, enhances the overall experience, and proves to be a more convenient option compared to ML Kit and ZXing for React Native projects. After exploring and experimenting with various potential solutions during the development process, we have chosen React Expo Camera as the preferred software solution.

# LoRaWAN and LoRA RF

LoRaWAN

This consists of many end devices connected to gateways that receive messages using ALOHA-based protocols so that end devices do not need to be pared to specific gateways yet be within a gateway's range to send signals. These messages are then received by the network server; if it receives multiple copies of the same message, it only keeps one.

End devices can be a sensor, an authenticator, or even both. They are likely to be battery operated and connect via LoRa radio frequency modulation.

Gateways can be categorised into indoor (picocell) and outdoor (mactocell) gateways.

Indoor gateways are cost-effective and best for a location with many obstructions to the wireless signals. They have internal antennas or external pigtail antennas. Depending on the building, its possible for them to receive signals from over several kilometres away.

Outdoor gateways provide larger coverage in both urban and rural areas. Usually attached to cell towers or tall buildings.

Lastly, you need 3 servers: a network server, an application server, and a join server.

Pros -

Low power: battery-powered

Long range: in In Kilometres

Cost-effective: uses unlicensed radio frequency and low set-up cost

Scalable—can support thousands of devices on 1 gateway.

Secure \*

Cons-

Difficult to maintain custom protocols

LoRa RF is not designed for scalability

*\*Security*

LoRaWAN security

Built-in security mechanisms to protect data confidentiality, integrity, and authenticity.

Uses the AES-128 encryption for end-to-end as well as 2-layer encryption using network layer encryption and application layer encryption.

This is because its less complex, more efficient, and more cost-effective to run, which is a large point for most implementations. Yet 128 is still secure enough for current standards.

# Front End Design

## Iteration One:

The design needs to be modular and simple. The application should be easy to use for a wide range of people who are technical. The app is going to feature 5 distinct colours for a theme.

A screenshot of a computer screen

Description automatically generated

The app's main background will be #EDF6F9 with the font colour being #303633.

For our frontend design we will be using Flutter allowing us to port to many types of devices including phone, desktop and web giving clients many ways to access the software our primary focus will be mobile support allowing contractors and admins to add new devices on site and in real time.

A screenshot of a device

Description automatically generated

Iteration two features only three colours: white, royal blue and charcoal black. I Chose royal blue as many people associate blue with technology and with the app being a tech tracker I thought it was a perfect choice of colour. The charcoal black is a less harsh black and clashes less with the colour theme.

A blue and grey rectangular shapes

Description automatically generated with medium confidence

Screens screenshot of a phone

Description automatically generated

A blue and white background with a logo

Description automatically generatedA screenshot of a phone

Description automatically generated

## Iteration Two:

After meeting with our client, we concluded for what the UI colour scheme should be. The client has a startup company “resegva” that he has designed the website for and follows the following colour scheme convention.



Using the colour scheme the client provided I was able to amend my initial prototype. Below is the new and improved version of the UI.

A screenshot of a phone

Description automatically generated

# Prototype 1

A screen shot of a phone

Description automatically generated

This is the splash screen for the application. This page shows the application logo and colour theme. It has two large clear buttons one for logging in and another for signing up for the application. When a user presses either of these buttons it will transition them to either the login or sign up page.

A screenshot of a login form

Description automatically generated

This is the login page. It follows a similar structure to most other apps that have a login in page like Instagram and Facebook so it should be easy for a new user to understand what their next steps are to login to the app with a clear email and password form for a user to enter their details and the submit button being clearly labelled for the user to know what it would do.

A screenshot of a login form

Description automatically generated

This is the sign up page like the login in page it follows a similar structure with each text input field being in a column. Once a user inputs valid data into all fields and presses submit it will then take them to the main home page. A user can also go back to the splash page with the back button on the bottom left.

A screenshot of a cell phone

Description automatically generated

This is the Main home page of the application. The top of the page features Page description text meaning the user knows what they are currently looking at.

Theres a search bar at the top of page with bold text saying “Search” so the user instantly knows the function of the bar it also has an search icon which is the universal icon for a search bar like on google and YouTube. The search bar also shows the amount of devices that they currently have connected to their application.

Then there are two buttons to filter and change the main page if a user has too many devices it may be easier for them to find the device their looking for in a grid pattern. There is also a filter button that will bring up a filter menu.

The main page also features the cards which show the device. Each cards has an image associated to the device a name and then the auto filled information about its last maintain , last updated GPS location and any warnings like if a device needs a new battery.

At the bottom there is a navigation bar which features a QR code button that will take the user to the QR code scanner. A plus button which will take the user to create a new device page and a settings cog which will take a user to the settings page.

A screenshot of a phone

Description automatically generated

When the user presses the filter button this element will pop up and the background will darken showing that the element is the only active item on the page that the user can interact with.

The filters are in a list where a user can scroll done an select which filters they would like too add to more easily find their devices or document just a few of their devices without having to see them all. When a user is done they can Apply to see what they selected or press back to not apply their filters.

A screenshot of a computer

Description automatically generated

This is the main page in the Grid formation this can be used when the user wants to see more devices at once so if they want to find a device but cant remember its name or they have too many devices and don’t want to scroll all the way through a vertical list they can more easily search through each item.

A screenshot of a map

Description automatically generated

This is the page shown when a user clicks on a device. This page shows the user the map of the GPS location and when the device was installed , last maintained and any additional notes on the device. The user can also see the log history this would we useful to the user if they want to see previous maintenance dates notes etc. There is also a QR button that shows the devices QR code that could be printed off of shared to allow others to find the device easier.

A screenshot of a phone

Description automatically generated

This is the enter details page. When a user clicks the plus icon this page will appear and allow the user to enter new device details. Each input form is laid out in a list like the rest of the input forms to bring uniformity to the application and so that users can easily use the app.

At the bottom there is a back button and an accept button which fit the style of the rest of the application. The user will know what these buttons do automatically as these are universal accept and back button in most apps.

A screenshot of a computer

Description automatically generated

# Technical Analysis

## Initial Security analysis of applications and threats:

|  |  |  |
| --- | --- | --- |
| Risk | Affect | Countermeasure |
| User spoofing | Gain unauthorised access | 2Fa/MFA when logging in |
| Poor data base encryption | Malicious access to large quantity of data from clients stored in the database | Encrypt all data in the database to minimise the possibility |
| API attacks | Possible to infiltrate the database using illegitimate requests using the API | Ensure all requests are authorised and possibly create a rate limit on request to reduce DOS attacks on the DB |
| Malware/Ransomware | data base is taken from our control. | Frequent backups to a secondary database to minimise the threats |
| Man in the middle attack | The attacker intercepts a valid communication path between user and software and can gain access to sensitive information or inject malicious software or code into the database to gain access | Create a secure path using a multiteam of validation certificates such as TLS and SSL |
| Firebase formatting | May lead to unauthorised CRUD functionalities being performed | Format for minimum access to database. |
| File Uploads | Infected files being uploaded to the system | Ensure only authorised formats like JPG, PNG or dxcs are used and not.exe or.py to minimise the possibility of a threat |
| API key expose | Can lead to unauthorised request being sent to the firebase | Use a system that only lets authorised usage of the API port by authorised individuals. |
| Data base breaches | Data base breached by a method mentioned | To increase security, encrypt and hash all data that seems sensitive. |

### Important Laws:

There are many laws that we need to follow to ensure the project abides by the UK and EU laws, as this is where its likely to be dispatched too.

GDPA 2018 (UK specific)

GDPR (EU wide)

Computer misuse act—to be followed by all when interacting with the software

Freedom of Information act: allows user to be aware of what data is stored on our servers about them

#### Research:

Digital economy act 2019

Network and Infrastructure Security ACT 2018

Cyber Security ACT 2019

Network and Infrastructure Security Act 2018:

Perform regular risk assessments of the application to ensure that valid measures are taking place to minimise the security risk to the network.

The ACT requires us as a company to report any cyber security breaches that may affect our systems, such as cyber-attacks, breaches into our databases or networks, any ransomware or DDOS/DOS attacks, and compromise of critical infrastructure, insider attacks, or data leaks.

A couple of possible threats will likely be around our DB usage and API ports.

### How it applies:

Overall, as a team, we will need to apply and follow all rules and regulations that apply to our database, ensuring we store data securely, perform secure connections to the database, and limit access to the lowest possible viable level as to minimise possible security vulnerabilities. A large part of the security of our infrastructure is how it will be accessed. Depending, we will need to perform some user questioning to see the ways it will be accessing the network and be able to optimise the connectivity as stated before to minimise the possibility of attacks.

# Risk Management

Risk management is a critical component to ensure that the risk is identified analysed and mitigated proactively to ensure minimum damage occurs. This section potential risks their likely hood of happening mitigation strategies and contingency plan if it happens.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Likely hood (out of 10) | Impact | Mitigation strategies | Contingency plan |
| Software bugs | 8 | Low | Test code during production | Identify bugs and handle them |
| API Failure | 2 | High | Monitor API’s behaviour |  |
| Data Breach | 2 | High | Secure all code and all sensitive information with robust encryption | Notify affected users patch vulnerability and create a Incident Report |
| System Downtime - Maintenance | 5 | Medium | Inform users of the issue ahead of time |  |
| System Down | 3 | High |  |  |
| Unauthorized access | 4 | High | Implemented a RBAC model | Block suspected unauthorised accounts, investigate breach strengthen access policies notify users if affected |
| Performance Issues | 7 | Medium | Continuously monitor and test the software to address bottlenecks | Roll back of recent changes and implement hotfixes to ensure software operates as expected. |
| Unauthorized API calls | 2 | High | Restricted API calls to authorised accounts | Block unauthorized calls, notify all affected parties and update security. |
| Team Member Loss | 1 | High | Ensure all members are always aware of all current projects taking place and able to take over at any moment. | Delegate the team members workload among the remaining team members extend all deadlines re-do project requirements.  Inform all stakeholders. |

# Quality Assurance

## What is it?

Quality Assurance (QA) refers to the steps and measures taken to ensure that the final product meets high standards and aligns with the project requirements outlined in the brief. It is a process aimed at identifying and addressing potential issues during development, ensuring client satisfaction and project success.

## Why it’s needed?

It is an essential process to ensure that the final product meets the clients, industry standards and out expectations of what was expected to be the outcome.

The lack of QA can lead to the derailment of the project and the outcome not being what was expected. QA benefits In the continuous improvement of the project and a high-quality outcome.

## How will it be achieved?

This will be achieved through the following:

Communication with the client of the project in this case John McDermott from Fairfield’s

Team meeting ensuring that what everyone is doing aligns with the project plans and the main idea for the project outlined in the Project Requirements.

Have a plan signed off before implementation fully starts on the project on 7th of January. This will assist in ensuring that everyone has the same plan.

If issues arise during the project Process the following issues will be done:

**Step 1** Team meeting with Nikodem Tom Joseph. Discuss and locate the root problem and try to neutralize it.

**Step 2** If the issue persists allocate more resources to that area of the project

**Step 3** If the issue is larger and requires input from client, contact client and schedule a meeting. Otherwise skip this part.

**Step 4** Document the issue’s and what we learned from it

**Step 5** If the issue was taken longer than expected amend the sprint timeline

Lastly have continuous review of the QA plan at each step to ensure that its relevant to that section.

# KPI’s / Testing

## What is it?

Key Performance Indicators designed to analyse if the product performs as it was expected by the client.

For most KPI’s Sections we will perform test to ensure that they are within target.

## Our KPI’s

|  |  |  |
| --- | --- | --- |
| KPI | Description | Target |
| API response time | The API should have an average response around | <300ms |
| Data Completion | The number of devices in the database with all of the required data entered to ensure accurate tracking | >=95% - human error |
| Data Accuracy | The % of accurate data in the database against real information. Highly dependable on upkeep. | 95% - human error |
| Unauthorized logins blocked | Success rate of stopped access to the system with unauthorised credentials | 100% |
| Encrypted data | Data that is required to be encrypted is encrypted | 100% |
| RBAC( Role-Based access control) Accuracy | Percentage of access correctly granted/ denied based on their role | 100% |
| Unauthorized API calls detected | The % of unauthorised API calls caught and blocked | 100% |
| Data Breaches | Amount of data breaches in a tax year | 0 breaches |
| Data privacy compliance | Percentage of features that adhere to GDPR regulations | 100% |
| Task Success Rate | Percentage of users able to complete a task first time with no errors after tutorial | 95% - human error |
| Navigation Simplicity | Percentage of users that are able to navigate the app after tutorial without further assistance. | 90% |
| User satisfaction rate | A score out of 10 taken post task rating the satisfaction felt after using the app. | >8/10 |
| Error free Sessions | Percentage of session carried out without encountering critical errors | >99% |
| Data synchronizations success rate | The percentage of data updates between client and database | >99%  Only 99 as out of scope issues like internet issues may lead to incomplete transition. |
| Percentage of failed API calls | A percentage of attempted API calls that never received a call back. | <1% |
| App loading time | The time in seconds for loading into the app’s login screen | 3 seconds |
| Uptime | The uptime of the service | 99% |

## Testing:

|  |  |  |  |
| --- | --- | --- | --- |
| KPI | Description | Test | Expected Outcome |
| API response time | The API should return within 300ms | Ping the API and calculate response time | Callback sub 300ms |
| Unauthorized logins blocked | Test attempted false logins | Log in with false credentials | Reject Login attempt |
| Unauthorized API calls detected | Attempt to call the Api through unauthorised means | Ping Api while unauthorised. | No Callback or Callback 403 unauthorised |
| API callback failure | Test if the API returns a callback | Ping API | Callback present |
| App Loading Time | The time taken to boot up the app on a relatively new device | Boot the application | Log in screen present within 3 minutes |
| UI Testing | Test all sections of the App | Press all buttons and interact with everything to ensure no bug are found in the software | All sections should behave as expected |
| Filter Testing | Ensure data is correctly queried | Test all quarriable functions | No erroneous data displayed |
| Tracking accuracy | Test if the app displays the correct tracking information | Provide a location of a device and see if its correctly displayed on the app | Correct co-ordinates displayed |
| RBAC accuracy test | When logging in ensure correct role-based permissions applied | Log in with different permission-based accounts | Logged in with correct permissions |
| Data Synchronisation | Data is synchronised with user input | Modify data in the database | The database updated withing 300ms with the updated data |
| Data Accuracy Testing | Modified data should be accurately represented in the database | Modify data and check database to see if it represents the change | Data sent is the data visible |
| Data Duplication testing | Errors sometimes occur when the API receives two consecutive calls with the same data leading to duplicate data inputted into database. | Send requests to the database and monitor the database to ensure duplicate entries are not created form the same request | No duplicate entries |
| Regression testing | After new features are added, we need to test old features in the system | Partially Re-perform past test regarding functionalities | No new bugs should be present. |
| Stress Testing | Insert a large volume of records in a period. This should test efficiency and functionality | Perform a database population script to add large volumes of data. | All data successfully handled and added. |
| Error Handling Test | Purposely create issues or perform inputs that create erroneous outputs to test if the software handles such issues effectively. | Stress test input fields and any other variables | All errors should not result in crashes and appropriate error messages |

# Approval Process

The approval process for the project is structured into three clear stages to ensure that all deliverables meet the required standards and satisfy client expectations.

## Step 1 – Team Review

The first step in the process is an internal team review.

Nikodem, Thomas, Joseph will perform an internal team review.

**Assess** the deliverable for quality, accuracy, and compliance with the project objectives.

**Identify** and resolve any issues or gaps in the deliverable.

**Ensure** all relevant documentation is complete and up to date.

## Step 2 - Client Review

The client evaluates the deliverable for alignment with their requirements.

Provides comments, suggestions, or requested modifications if needed.

Approves the deliverable once satisfied.

## Step 3 – Final Approval

The deliverable is finalized, and the project proceeds to the implementation or next phase.

This marks the formal approval, signifying that all parties are in agreement.

# Firebase:

## Database Explained:

We will be using firebase to manage the data for the IoT as per the project brief. Firebase is a great choice due to its straightforward and easy implementation that is developer friendly. It provides a Real-Time database. It is a great solution for cross platform development and low entry costs. However, it is an issue as Firebase is a Google Service.

## Database Low Fidelity Plan.

A diagram of a software flow

Description automatically generated

## Features to be used:

* Firebase Authentication
  + Supports user sign-in and 2FA/MFA
* Cloud Fire Store
  + For complex queries and hierarchical data/scalability
* Realtime (Priority)
  + Synchronisation updating device statuses
* Cloud Functions
  + For micro service architectures like events and function execution
* Cloud Storage
  + Pictures of the installation, relevant documentation, etc
* Cloud Messaging
  + Send notification to users, like device status warnings
* Firebase Performance Monitoring/Crashlytics
  + Identify bottlenecks and problems

## To look out for:

Authentication of Firebase

User/Owner login info

How to handle keys and authenticate securely in app

## Summary

This backend is design with 3 keys traits in mind.

Firstly, it is designed to be scalable as per the requirements its needs to be able to handle between 5-1000 clients this means it needs to be cost effective and cheap to run at 5 clients but also scale effectively up to 1000 making an application with pay as you go/by use features would be optimal.

Secondly, It also has be customizable this is best achievable with NoSQL structures that firebase provides such as Json or their collections system which although more complicated does provide higher querying capacities than Json as well as costing less to run.

Thirdly, support & maintenance, due to the implementation being over a short period of time then being handed off our team only will work on it for a small period of time and will not be able guarantee long term support therefore it is important that the backend relies on a service that can update with legacy support. This is also important for establishing connections between service and keep the product simple and understandable so it can be picked up by a different team.

## Overview

It use case is for a IoT system and so it needs to be able to store media and key data such as meta data, this would include things like its model, manufacturer and specification or any other chosen features the client desires. Installation details like its location the date it was installed, last recorded battery life, this could also include media such as images of the device in its setup in environments as well as its QR code. It will also need to be able to store documents to give more in depth details about the item and its setup this would include it certifications, software version, etc.

How can we achieve this?

We will be using Firebase who provide a multitude of service that would be appropriate for this product based on both technical requirements as well as conforming to other preferable qualities such as scalability requested by the client. We will need setup Firebase and configure it and layout a design plan for how our program will structure itself withy Firebase once this is complete the backend can start to be programmed. We will take a modular approach focussing on reusable functions that can take in a wide variety of arguments necessary for the programs flexibility while avoiding messy code. The primary features Firebase we will be using is the Cloud Firestore database, Firebase Cloud Storage for media, Firebase Authentication and messaging/notifications.

# Frontend Architecture

We propose to use React Native as the framework for the front end interface this is a JavaScript Framework and due to this it makes the app extremely portable using Expo Go we can port the application to Web, iPhone, Android and Desktop without having to change much code this makes prototyping easy and efficient and changes to one will happen to all. React Native also using CSS for its design making it easy to design new changes and create Objects. However, the downside is that JavaScript isn’t the most efficient language and is slower compared to if we made the app run natively on iPhone, Android etc. Using C++ or Rust for the frontend however this would lead to slower development time and would have to port the application to each device individually. React Native also allows for instant builds in Realtime and so in the prototyping phase it can be quick and simple to make changes without having to wait for the entire project to rebuild ; React Native also allows for modules to be used and like libraries I can import code that is already being maintained and developed in an open-source environment and so portions of the codebase will be maintained for newer phones and devices for free until they are deprecated and at the time we could of developed an in house solution for it.

# Budget Analysis – React Native

Due to React Native being an open source project developed by Meta formally known as Facebook this means that React Native has no upfront cost to development or maintenance and don’t need to pay license fees if published on a App store on the Web. However libraries used within the App could incur a cost like the google maps API if used for the GPS locating within the app costs $7 per 1,000 API calls per month and with no planned revenue for the app could lead to a massive increase in cost per month unless we use the Embed API service which cost $0 per API call.

Overall , the cost for the application and maintenance should be $0 per moth for its lifetime however , if newer phones come out and React Native needs to be updated to support those phones it could cost money in maintenance to get the app running again on the newer phones whilst still supporting the old architecture.

# Backend Architecture

We propose Firestore over Firebases traditional Realtime Json database this is for a few reasons but the primary reason is its increased querying capability, Json is notoriously poor for making queries in but Firestore does not have this weakness as even though its Json like in structure it is also similar to SQL how each collection is like a mini table meaning advance queries can still be run with effective time and performance. Its scheme is like Json and is very flexible to accommodate different user priorities when adding devices as well as supporting subcategories allowing for users to sort and manage the database themselves without any need for more technically minded people to step in. Even though Firebases Realtime database is more suited to Realtime requests Firestore still supports and can perform to the project specification. This small disadvantage of being worse at realtime support is negated by how effective Firestore is for querying and over large data set operations and it also make more sense from a cost perspective as the ability to perform more advance queries increase efficiency and reduce the number of requests.

The structure looks like this:

Collection: devices

Document: deviceID123

- name: "Temperature Sensor"

- model: "TS-5000"

- manufacturer: "Acme Corp"

- location: { latitude: 40.7128, longitude: -74.0060 }

- installationDate: "2025-01-01"

- imageURL: "https://example.com/installation.jpg"

- technicalDocs: ["https://example.com/specs.pdf"]

Authentication – Needs further discussion with client as to the specifics of what they want depends if they want email/password, google, etc sign in even token-based sign on method as well as if they want MFA/2FA all supported and easily incorporated through firebase, though some play store features need a token to be setup.

# Budget Analysis - Firebase

Firebase can be used in a restrictive free version, this is useful for small project/ use cases however if we want to use their ‘functions’ capabilities which allow for much more serverless control over how request are managed we would need the blaze plan which is their pay as you go solution.

Free vs Paid

A white background with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a phone

Description automatically generated





A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated(OPTIONAL)

Key points:

1. **Authentication**

* Free tier: 10K monthly active users (MAUs) are free.
* Beyond 10K MAUs: $0.01 per MAU.
* Predicted: £0/per month under our use case

1. **Firestore**

* Free tier:
  + 50,000 reads, 50,000 writes, and 1GB storage per month.
* Beyond free tier:
  + Reads: $0.06 per 100K.
  + Writes: $0.18 per 100K.
  + Storage: $0.18 per GB per month.
* Assumiming each user use is:
  + 10 reads/day
  + 5 writes/day
  + 0.1MB of data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Users | Reads/Month | Write/Month | Data Stored | Cost |
| 5 | 1,500 | 750 | 0.5 MB | $0 |
| 50 | 15,000 | 7,500 | 5 MB | $0 |
| 500 | 150,000 | 75,000 | 50 MB | $1.80 |
| 1000 | 300,00 | 150,000 | 100 MB | $3.60 |

1. **Firebase Storage**

* Free tier:
  + 1GB storage, 50K downloads per month.
* Beyond free tier:
  + Storage: $0.026 per GB per month.
  + Downloads: $0.12 per GB.
* Assuming
  + 1 image (500KB) uploaded/month
  + 3 files downloaded/month

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Users | Data Upload | Data Downloaded | Storage Cost | Download Cost | Total Cost |
| 5 | 2.5MB | 7.5MB | $0 | $0 | $0 |
| 50 | 25MB | 75MB | $0 | $0.01 | $0.01 |
| 500 | 250MB | 750MB | $0.01 | $0.09 | $0.10 |
| 1000 | 500MB | 1.5GB | $0.01 | $0.18 | $0.19 |

**Total Cost**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Users | Authentication | Firestore | Storage | Total |
| 5 | $0 | $0 | $0 | $0 |
| 50 | $0 | $0 | $0.01 | $0.01 |
| 500 | $0 | $1.80 | $0.10 | $1.90 |
| 1000 | $0 | $3.60 | $0.19 | $3.79 |

If we were to conisder function + firebases app hosting feature we woud be look at an additonal 38p a month, although these are all **estimation** and subject to change based on frequency of use these values were gathered assuming low use of their services.

# Proof of Concept

Here we will be providing evidence of our proof of concept for the project.

## Concept/Testing

### Rules for test database used of Testing

Below are the rules we have implemented for our testing purposes in the database:

{

"rules": {

"users": {

"$uid": {

".read": "$uid === auth.uid", // Users can read their own data

".write": "$uid === auth.uid || auth.token.admin === true" // Users can write their own data, admins can write any data

}

},

"adminData": {

".read": "auth != null && auth.token.admin === true", // Only admins can read admin data

".write": "auth != null && auth.token.admin === true" // Only admins can write admin data

},

"publicData": {

".read": "auth != null", // Any authenticated user can read public data

".write": "auth != null && auth.token.admin === true" // Only admins can write public data

}

}

### Test Table Designs:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table\_Name | Data\_Name | Data\_Type | Key? | Security\_Type | Explain |
| USERS | ID | integer Autoincrements | Primary | N/A | Unique User ID |
|  | Name | String | No | N?A | Name of acocunt |
|  | Password | String | No | Encrypted/Hashed | Password for the account |
|  | Account Level | String | No | N/a | The permission level of account |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table\_Name | Data\_Name | Data\_Type | Key? | Security\_Type | Explain |
| DEVICES | ID | integer AutoIncrament | Primary | N/A | Unique Device ID |
|  | Name | String | No | N?A | Name of acocunt |
|  | Type | String | No | N/A | Type of device |
|  | Last Maintance date | String | No | N/a | Last maintanance |
|  | Next Maintantce date | String | No | N/A | Next Maintance |
|  | Owner Id | Int | Foreign | N/A | The Id of who owns this device in the database |
|  |  |  |  |  |  |

### Database CRUD:

We will be using Java for CRUD functions for the time being as it has the best implementation with Firestore and using IntelliJ for testing purposes using the command line to test the code I develop:

Create:  
Current test code for a basic Create Function:

*public class FirebaseInit {*

*public static void main(String[] args) {*

*try {*

*FileInputStream Acount\_Key = new FileInputStream("ADMIN-ID-KEY.json"); //get sdk key*

*FirebaseOptions BUILD = FirebaseOptions.builder()*

*.setCredentials(GoogleCredentials.fromStream(Acount\_Key))*

*.setDatabaseUrl("https://Farfield Test - Niko.firebaseio.com") //connect to database*

*.build();*

*FirebaseApp.initializeApp(BUILD);*

*System.out.println("Firebase Initialized Successfully!");*

*} catch (IOException e) {*

*System.out.println("Error initializing Firebase: " + e.getMessage());*

*}*

*}*

*}*

*public static void createUser(String name, String password, String accountLevel) {*

*DatabaseReference ref = FirebaseDatabase.getInstance().getReference("users");*

*String userID = ref.push().getKey()//gets next availeble id value*

*System.out.println("Creating user:  ")*

*System.out.println(userID,name,password,accountLevel)*

*HashMap<String, Object> user = new HashMap<>();*

*user.put("name", name);*

*user.put("password", password);*

*user.put("accountLevel", accountLevel);*

*ref.child(userID).setValueAsync(user)*

*.addOnSuccessListener(aVoid -> System.out.println("User created successfully!"))*

*.addOnFailureListener(e -> System.out.println("Failed to create user: " + e.getMessage()));*

*}*

*public static void Test\_Create(){*

*Scanner scanner = new Scanner(System.in);*

*System.out.println("Testing Create:\n\n")*

*System.out.println("Name:   ")*

*String name = scanner.nextLine();*

*System.out.println("\nPassword:   ")*

*String password = scanner.nextLine();*

*String account = "Basic";*

*System.out.println("\nCreating user:    ")*

*createUser(name,password,account)*

*public static void main(){*

*Test\_Create()*

*}*

# Project 2003 GitHub Repository

# Additional Resources

https://docs.google.com/spreadsheets/d/1M9ZoHYVwFe1-7tP7ZRFlqP-Z2iBLnT40TkonrdL\_Ek8/edit?gid=1565921477#gid=1565921477

# Additional Research

## Topic - AI (Late Stage idea)

**AI** #late-stage

Possibility of implementing AI for assistance

**AI (adverisal learning)**

The act of manipulating an AI model to provide erroneous data or otherwise data that it should not be providing to the individual interacting with it. Machine Learning is compromised by the

# Project 2003 Approval Signature

## Team Members

### Nikodem

Signature:  
Date: 01/01/2025



### Joseph

Signature:  
Date: 01/01/2025

### Thomas

Signature:  
Date: 01/01/2025

## Client

Signature:

Date:

# Follow Structure this:

Project Plan Structure

1. Project Title: Clearly state the name of the project.

2. Project Overview: Provide a brief description of the project, including its purpose, goals, and objectives.

3. Project Scope: Define the boundaries of the project, specifying what is included and excluded.

4. Project Objectives: Clearly state the measurable and achievable outcomes the project aims to accomplish.

5. Stakeholders: Identify and list all stakeholders involved in the project, including their roles and responsibilities.

6. Project Team: Outline the members of the project team, their roles, and reporting relationships.

7. Timeline: Create a detailed timeline with milestones and deadlines for key deliverables. Use a Gantt chart to illustrate this clearly.

8. Research: present findings from existing work that address the problem statement, scope and objectives of the project. Main part of this section is to address existing competition and solutions, and how your project is unique in its approach. You may pull material from your design document for this.

9. Proposed Solution: an overview of your project’s solution and methodologies. This should be in line with your overview, scope and, objectives, timeline and further elaborated on under the Work Breakdown Structure next.

10. Work Breakdown Structure (WBS): Break down the proposed solution into smaller, manageable tasks and subtasks. Create a hierarchical structure showing the relationship between different tasks. You may pull material from your design document for this.

11. Resource Plan: Identify the resources (human, financial, equipment) required for each task and allocate them accordingly.

12. Risk Management: Identify potential risks and develop strategies for risk mitigation. Include contingency plans for addressing unforeseen issues.

13. Communication Plan: Define how communication will be handled throughout the project, including regular meetings, reporting mechanisms, and channels of communication.

14. Quality Management: Specify the quality standards and processes that will be used to ensure the project's deliverables meet the required criteria.

15. Monitoring and Evaluation: Outline how the project's progress will be monitored and evaluated, including key performance indicators (KPIs).

16. Budget: Provide a detailed budget outlining the costs associated with the project, including resources, materials, and any other relevant expenses. Cost savings using open-source solutions that are free are a big plus point to be mentioned here.

17. Approval Process: Clearly define the process for obtaining approvals at different stages of the project. For this module, you will require the client’s approval/signoff for this.

18. Change Management: Describe how changes to the project scope, schedule, or resources will be identified, evaluated, and implemented. This is in case the client wants to request changes in semester 2, what’s your process for doing so?

19. Closure and Evaluation: Outline the steps for closing out the project, including a postimplementation review and lessons learned. This should be in line with the final submission checklist provided in the Handbook.

20. Appendices: Include any additional documentation or reference materials that support the project plan.