App for Microgrid Demonstration

Design Document

Team sddec21-21

Client: Anne Kimber

Advisors: Mathew Wymore , Steve Nystrom , Nicholas David

Team: Gabriel Rueger , Michael Doyle , Micheal Thai , Patrick Shirazi , William Bronson

Email: sddec21-21@iastate.edu

Website: http://sddec21-21.sd.ece.iastate.edu

Revised: March 9, 2021 - v1

Executive Summary

Development Standards & Practices Used

Waterfall Model for development cycle.
Coding Standards to maintain quality code base.
Digital design standards for optimization and consistency of our mobile application.

Summary of Requirements

- The server shall have the ability to add additional crates from the microgrid
- The database shall be configured to add additional data sources to the database for each crate
- The database shall be able to query and search different subsets of the data
- The database shall have a configurable data collection interval
- The database shall support automatic archiving of data
- The mobile application shall display data within a minute of collection
- The database size shal scale linearly with the number of data sources and with time
- The server shall reduce old data to 10 minute averages and store it in a separate database
- Frameworks, libraries, etc. for this project shall be well-supported and maintained
- Throughout this project, open and well-supported communication standards shall be used
- All decisions made throughout this project shall be well documented

Applicable Courses from Iowa State University Curriculum

SE/CPRE/EE 185
COM S 227, 228, 309
S E 329, 339

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

The EE students will be introduced to Java programming language. The EE students also learned some basics of the backend, frontend, and database aspects of this project. TBD as progress on the project continues.

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1 Introduction

1.1 Acknowledgement

Lead Project Advisor - Anne Kimber

Advisors - Nicholas David, Steve Nystrom, Mathew Wymore

1.2 Problem and Project Statement

This application will serve as an efficient way for public users and crate maintainers to analyze efficiency of power collection and distribution of the solar crate from data presented through the application. Crate maintainers will be able to gain a better understanding of how a crate collects energy over time. Crate users will know what the crate could be used to power given how much power is stored in the crate's battery storage.

The solar crate will be deployed in areas of disaster, or areas that have lost their access to electricity. The crate will be utilized as a temporary power source for these affected areas, allowing users to power essential items like mobile phones and electric cars.

Problem statement:

Retrieve and present energy data collected by solar crate to public users and crate maintainers.

Solution approach:

Build an approachable mobile application that presents energy data from the solar powered crate. Application will utilize many different graphs and tables to present this data in a comprehensive and organized way. The mobile application will be built using basic backend and frontend components seen among other mobile applications of this type (i.e. database, user-interface, API).

1.3 Operational Environment

Application will need to be functional on iOS and Android mobile platforms. This application will be exposed to and used by the public to gain information about the sun crate. It is a necessary requirement that there is built in security to protect the data being exposed to users, and cannot be modified by average users. Application components will also need to be performant to allow multiple users to analyze data presented on the application. Application will also need the ability to scale with the number of crates deployed at that time.

1.4 Requirements

The requirements from our faculty advisors for the 2021 Spring semester to develop an app to show the operational data from a solar/Tesla powerwall/storage/wind/diesel/car charging off-grid microgrid operating in Ames, IA, are as follows:

For the backend:

- It should easily extend to additional data sources and display methods.
- It should have a configurable data collection interval.
- It should have an automatic support archiving of data.
- It should take 10-minute averages for older data.

- It should be a secure but open system (it does not need authentication but should be extendable).
- The database size should scale linearly with the number of data sources and with the time.

For the frontend/display:

- It should be a mobile app (iOS & Android).
- It should support browsing raw/averaged data.
- It should support graphing data over time, configuring time range, and show the data sources.

For the whole system:

- It should fit into the client's architecture.
- It should have a max 1-minute delay between updated value and display.
- It must use open and well-supported communication standards.
- The software should be maintainable.
- The frameworks and libraries must be well-supported and maintained.

1.5 Intended Users and Uses

This mobile application is intended for the general public and will display the microgrid's overall performance. Additionally, researchers should be able to access the voltage, current, and frequency data readings.

1.6 Assumptions and Limitations

The following assumptions are as follows:

- The mobile app will only take data from one solar crate during development.
- Students will be given a way to access the microgrid data via the remote desktop.
- The mobile application should be able to display a time range of data from a crate in a graphical format.
- The mobile application only needs to use the English language.
- The team will be provided with a virtual machine to host the application.

The following limitations are as follows:

- The budget to produce the application will be negligible.
- The mobile application will be developed by the end of the 2021 Spring semester.

1.7 Expected End Product and Deliverables

The end product of this project is an application that collects and displays operational data from an off-grid microgrid in Ames. To accomplish this, a number of deliverables must be achieved. These deliverables include:
– A data aggregator to collect all the data from the data sources. This will function to access and process the operational data of the microgrid into the form that the rest of the application needs.

– A database to store the collected data. This will store all the collected data from the data

aggregator and will make the data readily available to the rest of the application.

– A mobile app to display and interact with this data. This will be an iOS and Android mobile application. It will allow users to see not just current data readings from the microgrid, but also query and view ranges of previous data for the microgrid.

– A server to manage the data between the aggregator, database, and mobile app. This will allow the different pieces of the application to communicate between each other and maintain the integrity of the system.

– Technical Documentation of the product and mentioned deliverables. The documentation will not only describe functionality and usability of the product, but also the decisions and processes the team took to create the product and deliverables.

The delivery date for these deliverables are shared among all deliverables as each deliverable will grow along with the others. The functionality of the product as a whole will define the delivery dates. The first date is May 1st, 2021, wherein the product must have functionality that encompasses collecting and displaying solar and battery data from the microgrid. The second delivery date is December 13th, 2021, at which time the product must encompass all requirements in totality.

2 Project Plan

2.1 Task Decomposition

Data Aggregator
– Setup connection between aggregator and microgrid
– Setup connection between aggregator and server
– Create functionality to read data from microgrid
– Create functionality to aggregate data together
– Create functionality to send aggregated data to server / database

Database
– Design database schema
– Setup mysql database
– Setup connection between database and server

Mobile Application
– Find graph library to help display data
– Implement graph library to display data
– Allow users to add search parameters for previous data
– Setup websocket to server to get real time data
– Deploy to Apple App Store
– Deploy to Google Play Store

Server
– Get data from data aggregator to
    – store in the database

– send real time to open mobile app instances
– Create api endpoints for the mobile application to get necessary data from
– Create websocket to send real time data to mobile application
– Create functionality to archive existing data
– Allow configuration of data collection

2.2 Risks And Risk Management/Mitigation

In any software project, risks are an issue that it is beneficial to attempt to predict and come up with solutions to mitigate. This project is no exception, and in this section we will explore the risks that we foresee might be an issue during development and beyond.

One such risk is of the graphing library that we choose. If it simply does not have the functionality that we require in displaying our data, then we will have no choice but to search for libraries or even develop our own solution for the task. But we must also consider, as for any third party software library, if there exists any vulnerabilities in the code and if the library will be supported throughout the lifetime of the project. For these instances, it might be acceptable to still continue using the library, but again most likely a new library will need to be found.

The other big risk for the application is the performance of the data flow throughout the entire application. The time from when the data is collected in the microgrid to the time the data is displayed on the mobile application must be reasonably quick (within 1 minute, and hopefully considerably faster), so if our application is not efficient in moving large amounts of frequent data, this goal will not be achieved. While this issue may not be completely avoidable due to reliance on the network as well as feasibility of the data, we plan to mitigate this issue as much as possible by using well defined and efficient methods to transport the data, as well as reduce and aggregate the data as much as possible in order to increase efficiency.

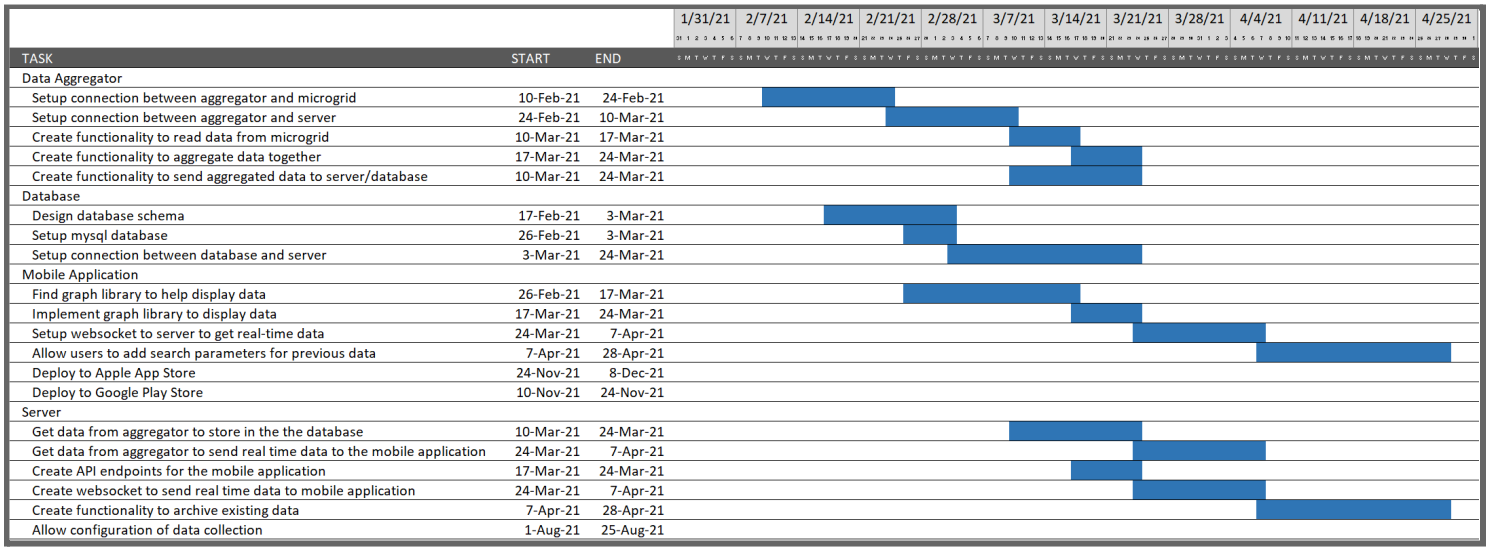
2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Given the nature of our project as a mobile application to interact with data from numerous data sources, our milestones will be quantifiable in several different ways. Some of these milestones can be quantified in terms of time, but other milestones will be quantifiable based on feedback and approval from our advisors and clients. We are very fortunate to have 3 advisors and 1 client for this project, and the relevant milestones will require approval from all 4 of these individuals. Our milestones for this project are the following:

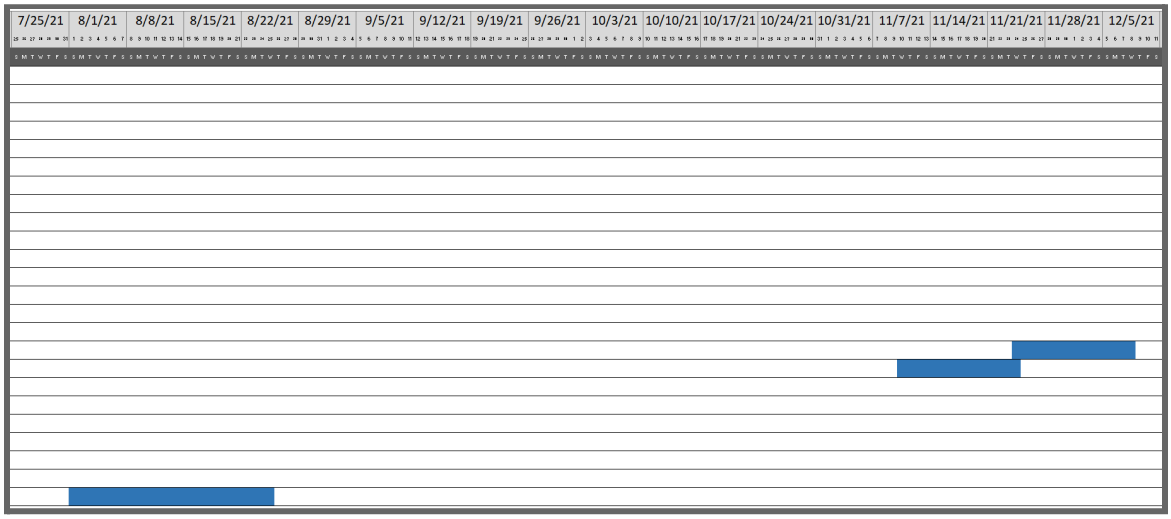
1. The mobile application displays data stored in the remote desktop which was obtained from the microgrid.
2. Data from a crate is stored at 1 second intervals for 24 hours, and 10 minute averages of that data are automatically calculated and stored separately.
3. New data from the microgrid is visible on the mobile application within 1 minute.
4. The mobile application is able to display solar data from a single crate and contains a prototype for displaying battery data.
    - This milestone is provided by the client of this project and will be considered completed on approval from the client and the 3 advisors.
5. The means of storing data from the microgrid can be configured to set the rate at which data is collected, the length of time over which average data is calculated, and the length of time that data is stored.

6. The mobile application is able to display up to one week of data at a time from a given source on the microgrid in the form of a graph.
7. The mobile application is able to display data from multiple different sources on the microgrid.
8. The mobile application has a consistent theme, intuitive user interface, and can display data in such a way that meets the requirements provided by our client.
    - This milestone will be considered completed on approval from the client and the 3 advisors, and it signifies that the project has reached a completed state.

2.4 Project Timeline/Schedule



TASK	START	END
Data Aggregator		
Setup connection between aggregator and microgrid	10-Feb-21	24-Feb-21
Setup connection between aggregator and server	24-Feb-21	10-Mar-21
Create functionality to read data from microgrid	10-Mar-21	17-Mar-21
Create functionality to aggregate data together	17-Mar-21	24-Mar-21
Create functionality to send aggregated data to server/database	10-Mar-21	24-Mar-21
Database		
Design database schema	17-Feb-21	3-Mar-21
Setup mysql database	26-Feb-21	3-Mar-21
Setup connection between database and server	3-Mar-21	24-Mar-21
Mobile Application		
Find graph library to help display data	26-Feb-21	17-Mar-21
Implement graph library to display data	17-Mar-21	24-Mar-21
Setup websocket to server to get real-time data	24-Mar-21	7-Apr-21
Allow users to add search parameters for previous data	7-Apr-21	28-Apr-21
Deploy to Apple App Store	24-Nov-21	8-Dec-21
Deploy to Google Play Store	10-Nov-21	24-Nov-21
Server		
Get data from aggregator to store in the the database	10-Mar-21	24-Mar-21
Get data from aggregator to send real time data to the mobile application	24-Mar-21	7-Apr-21
Create API endpoints for the mobile application	17-Mar-21	24-Mar-21
Create websocket to send real time data to mobile application	24-Mar-21	7-Apr-21
Create functionality to archive existing data	7-Apr-21	28-Apr-21
Allow configuration of data collection	1-Aug-21	25-Aug-21



Given our current requirements, this is how we plan to complete the project in two semesters. The main goal of the schedule presented here is to meet our requirement to be able to present a limited amount of data with our frontend application by the end of the spring semester.

2.5 Project Tracking Procedures

The progress of this project will be tracked using tools available in Git. Issue tracking in Git will be used to associate given commits with smaller tasks such as those outlined in 2.1. This issue tracking should be highly flexible in order to include other tasks and bug fixes that may become necessary as development progresses. The labelling service provided by Git should also be used to help associate merges and commits with issues. We will also use the milestone service provided by git to keep track of our progress towards the higher-level milestones outlined in 2.5.

2.6 Personnel Effort Requirements

Task	Estimated Time Requirement
Setup connection between aggregator and microgrid	20 hours
Setup connection between aggregator and server	10 hours
Create functionality to read data from microgrid	5 hours
Create functionality to aggregate data together	5 hours
Create functionality to send aggregated data to server / database	10 hours
Design database schema	5 hours
Setup mysql database	5 hours
Setup connection between database and server	15 hours
Find graph library to help display data	12 hours
Implement graph library to display data	6 hours

Allow users to add search parameters for previous data	10 hours
Setup websocket to server to get real time data	15 hours
Deploy to Apple App Store	15 hours
Deploy to Google Play Store	15 hours
Get data from data aggregator to – store in the database – send real time to open mobile app instances	20 hours
Create api endpoints for the mobile application to get necessary data from	5 hours
Create websocket to send real time data to mobile application	10 hours
Create functionality to archive existing data	15 hours
Allow configuration of data collection	20 hours

For the mobile application to be successfully implemented, our team will have to dedicate time to work on the frameworks for the app. We assigned roles based on experience with various program applications and experience with backend, database, and frontend frameworks. Ideally each team member will work with each task because the team needs to have a general knowledge of how each framework works, so we will work with each other on different tasks depending on what's needed at the moment.

2.7 Other Resource Requirements

We will use React Native open-source mobile application to build the frontend, MySQL to build a database, Spring boot to build the backend, and Virtual Machine running Ubuntu 20.04 LTS to host the database and server.

2.8 Financial Requirements

From our current expectations we do not predict any financial support will be necessary to complete the project.

3 Design

3.1 Previous Work And Literature

As far as the team is aware, there is no such existing application that provides the same functionality as our product. While some of the data sources have their own application for its own data, for example the Tesla Powerwall, these applications again are only for their own data and do not display related components data.

3.2 Design Thinking

One requirement that will largely shape the design of the application is the scalability requirements of the system to include new data sources and crates in the future. This leads to the idea of creating a modular application such that it is easy to add new pieces to it in the future. The application presented to the user should be intuitive to new users and provide information in a manner that is easy to understand. The user should have accessibility to obtain the application and see certain data that the public should be able to see.

While keeping the aspects that shape our design in mind, we made numerous design choices that will address these needs. We chose a frontend framework that can provide a user-experience typical of applications native to the devices we intend to support. We also have requested a server and chosen backend and database frameworks that can scale well and are easy to modify in the future.

Detail any design thinking driven design “define” aspects that shape your design. Enumerate some of the other design choices that came up in your design thinking “ideate” phase.

3.3 Proposed Design

When proposing a design for a given problem or to accomplish a certain task, there are many considerations that need to be made. The first thing to note is if any work up to this point has been done relevant to the task at hand. This could be work that could be transferred to contribute to this design, or it could be work that may have an effect on how this design needs to be implemented. With this in mind, consideration should then be made of the overall functional and non-functional requirements of the project that are relevant to the task.

Any design that is proposed needs to be capable of meeting these requirements no matter what, and this will help to eliminate any proposed designs that would not work. For this project, we have no standards to consider from third-party organizations. Once these steps are completed, the last thing to consider is if the proposed design has sufficient detail in order for it to be implemented. It may be helpful to consider that the team working to implement a design may not have been involved in the process of creating the design.

3.4 Technology Considerations

To highlight the strengths, weaknesses, trade-offs and choose the most effective backend framework out of Spring Boot, Laravel, Django, and Flask for our mobile app development, we focused on six key elements: License, programming language, age & documentation, performance, professional projects, and team experience.

Spring Boot Framework:

- License: Apache License (v2.0)
- Programming Language: Java
- Age & Documentation: Released in April of 2014, and official website contains various example projects and helpful guides
- Performance: Autoconfiguration may add unnecessary dependencies making binaries larger than necessary. In addition, it can handle multiple requests.
- Professional Projects: Inuit & Zalando
- Team Experience: Software and computer engineers have taken COMS 309

Laravel Framework:

- License: MIT License
- Programming Language: PHP
- Age & Documentation: Released in June of 2011, and official website contains documentation
- Performance: Generally slower for larger projects
- Professional Projects: 9gag & Kmong
- Team Experience: Unfamiliar

Django Framework:

- License: BSD 3-Clause
- Programming Language: Python
- Age & Documentation: Released in July of 2005, and official website contains documentation, tutories, topic guides, and installation help
- Performance: Performs well for large projects, feature-heavy, which may feel bloated for large projects, and can only handle one request at a time.
- Professional Projects: Pinterest, Instagram, and Robinhood
- Team Experience: Unfamiliar

Flask Framework:

- License: BSD 3-Clause
- Programming Language: Python
- Age & Documentation: Released in April of 2010
- Performance: It is suitable for smaller projects and doesn't handle concurrent requests as well as others.
- Professional Projects: Netflix, Reddit, and Lyft
- Team Experience: Unfamiliar

Additionally, we chose to pivot towards license availability and pricing, supported platforms, programming languages, maturity, and other notable aspects for the frontend frameworks.

QT Framework:

- License: Free open source license available, free educational license available, paid commercial license available, and separate distribution licensing available.
- Supported Platforms: Windows, macOS, Linux, Android, iOS
- Programming Languages: C/C++ for application framework and JavaScript, HTML, and QML for UI
- Maturity: It was first written in 1991 and had a long history of public bug reports and forums available. Also, there is extensive documentation on all available QT classes.
- Other Notable Aspects: Multithreading support

React Native:

- License: Free MIT License
- Supported Platforms: Android & iOS
- Programming Languages: JavaScript
- Maturity: It was founded in 2013 but released in 2015. It is supported by Facebook and also receives contributions from individuals and companies.
- Other Notable Aspects: It aims for a truly native feel on apps and does not support multithreading.

Flutter:

- License: Free New BSD License
- Supported Platforms: Android & iOS
- Programming Languages: C/C++, Dart, and Skia for UI
- Maturity: It was founded in 2017 and supported by Google. Because it's recent and new, there is not much support found online.
- Other Notable Aspects: It does not support true multithreading.

Finally, we chose to center the database to compare data storage, schema, team experience, and other notable aspects.

SQL (MySQL & PostgreSQL):

- Data Storage: Data is stored in tables.
- Schema: The schema defines the database structure, meaning all rows must have the same structure.
- Team Experience: Familiar
- Other Notable Aspects: It can use JSON type to handle adding new data source components, but it will require more space because JSON will be stored as a string.

MongoDB:

- Data Storage: Data is stored in a JSON-like structure.
- Schema: There is no schema, and it is much easier and efficient to change.

- Team Experience: Each team member is unfamiliar with it, and there are poor online reviews about it.
- Other Notable Aspects: MongoDB query language

3.5 Design Analysis

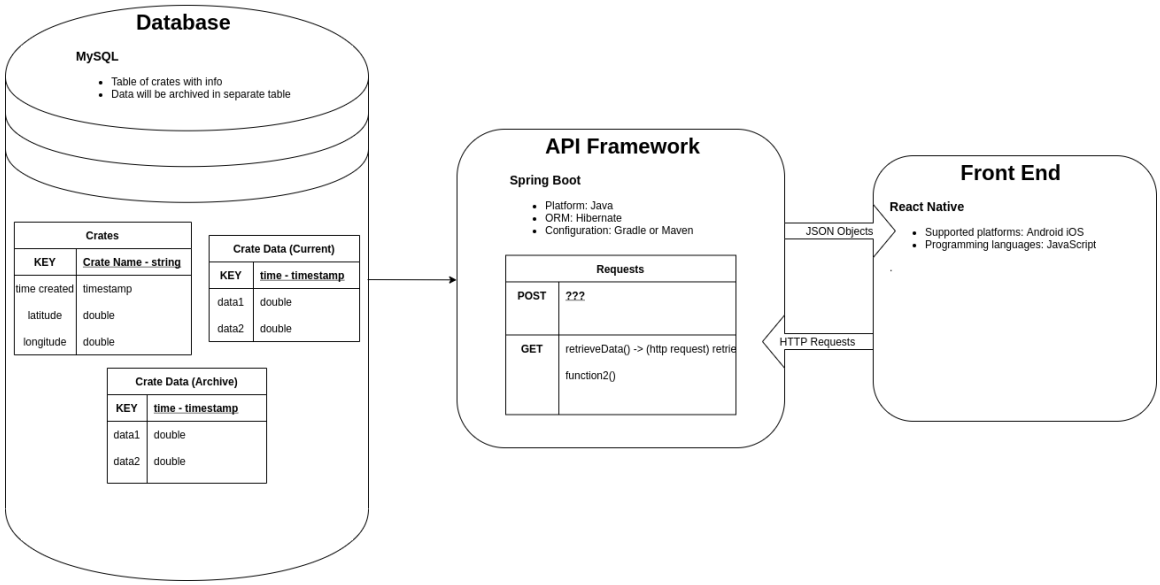
In the scenario where we are facing a problem with our design we will follow the process listed here. We will first look back at what we have tried up to this point and try to understand where it failed. Was it a shortcoming in the framework? Did we just not have the appropriate knowledge yet to use the tools available to us? Could any of our methods we've tried thus far actually accomplish our task? After asking questions like this, we then need to explore our options in addressing this problem while keeping several things in mind. For us, the higher level requirements for this project are simple, and they mainly consist of timing requirements for accessing and managing data, and what some high level features should be a part of this project upon completion. We do not have to worry about any formal standards established by any outside organization for this project.

With these things in mind, we can then start looking for solutions to the problem at hand. Some other things to consider during this process are what repercussions a given solution may have on other aspects of the project or if a proposed solution might constitute a change in requirements. Both of these scenarios should be avoided wherever possible. Finally, one last thing to consider in this process is if the description of the design needs more detail in order to guide this decision and prevent problems in the future.

3.6 Development Process

We will be taking a waterfall approach to our development phases. We had decided this approach because previous experience among team members working in this type of system. We also believed that due to the smaller scope of the project, waterfall will make it easy to stay concise and on task when it comes to scheduling a timeline.

3.7 Design Plan



Above is a diagram of the major modules of our project. These modules are required to store, access, and display the information mentioned in the requirements for this project.

4 Testing

Like any software project, testing is extremely important to determine not just whether a piece of code functions correctly or if groups of code function together correctly, but also if the project meets the requirements set out by the client. In this section, we will discuss our approach and methods used for testing.

4.1 Unit Testing

For unit testing, we plan to have testing coverage for every piece of software that we write. What this looks like for our backend server written in Java using the Spring framework is having at least one unit test for every function that we write to achieve total testing coverage of the codebase. To do this testing we use the JUnit and Mockito testing frameworks to create, define, and assist in writing and verifying the unit tests. Similarly to test our mobile application written in React Native, we will use a testing framework called Appium.

4.2 Interface Testing

The interface testing that we do mostly consists of testing the api endpoints for our backend application. To do this we can once again use JUnit and Mockito test frameworks, specifically the modules associated with REST endpoint testing.

4.3 Acceptance Testing

To demonstrate that our design requirements are being met, the team first conducts unit and interface testing described in the previous sections. If these tests are not passed, additional work is done to correct the errors until the tests pass. Once these tests are completed, the team will alpha test the product to see if it meets the requirements given by our customers. Once the team is satisfied that the requirements have been met, or have made changes to the product until the requirements have been met, the client is then given access to use and personally test the product to see if it meets their expectations. The client then gives feedback to the team, in which case the feedback is implemented and the testing process is repeated, or the client accepts the product and the phase is complete.

4.4 Results

For v1 of the document, we have not yet created or conducted any tests.

5 Implementation

Describe any (preliminary) implementation plan for the next semester for your proposed design in 3.3.

6 Closing Material

6.1 Conclusion

Summarize the work you have done so far. Briefly re-iterate your goals. Then, re-iterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

6.2 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE).

6.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar data that does not directly pertain to the problem but helps support it, include it here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc,. PCB testing issues etc., Software bugs etc.