**Indiana University Southeast**

**2021-22 CSCI Capstone Project**

**Workshop Management Web Application**

**Sponsor: Dr. Suranga Hettiarachchi**

**Programmer’s Manual**

**James Schlesener**

RS-9

1. Vision statement  
2. Introduction  
3. Component Overview (one for each)  
    a. Subsection for component (one)  
4. Tool overview (one for each)  
    a. Subsection for tool/language (one)  
5. Project Repository  
    a. Software  
    b. Test Cases  
    c. Documentation  
    d. Test platform description  
    e. Test scripts  
6. Installation for new install  
7. Installation for new platform (preserves data from previous)  
8. Further development statement (if I had another year to do this I would …)

Programmer’s Manual

Unified Utility Failure Alert Application

**Vision Statement**

The goal of this project is to create an Android smartphone application that will allow consumers and utility managers to inform each other of any utility failures based on their location.

**Introduction**

Sri Lanka is a small developing country in South Asia with 21 million people with comparatively established public utility infrastructure, such as electricity, water, and telephone. Though different agencies manage the public utilities, the Public Utilities Commission regulates all the agencies. One of the main disadvantages of the available web portals or applications is the lack of a mechanism for consumers to alert or inform authorities and other consumers of any service failures. We propose an Android smartphone application to alleviate this disadvantage. Our application will allow consumers and utility managers to inform each other of any utility failures based on their location. We believe this alert system empowers both consumers and utility managers to provide efficient and effective services to the consumers.

**Component Overview**

1. **Registration**

The registration portion of our application is how we create a user in the MariaDB database and allow them to sign in to our application and gain authorization to use the backend API. This requires a username, email, and password, which then sends this information along with the hashed password to the API and stores them within the database.

This then allows the user to sign in to the application, which takes a username and password, hashes the password, and then verifies the login with the API. The API then returns the user a JSON Web Token that authenticates them to use the API for the further functions of the application. When a user logs out of the application this JWT is removed, preventing any additional calls to the API.

1. **Reports**

The reports system is the backbone of the application that stores the data for users to view. This system integrates with the map system to take in data on the report and then formats it correctly and submits it to the API. The API then takes the object and stores it in the database.

1. **Map**

The map is the primary way of interacting with this application. It displays the created reports to the viewer. This also is integrated with the reports system to provide longitude and latitude to the report when you click on the map. It also displays the vendor and date of the other created rep

**Tool Overview**

1. MariaDB is one of the most popular choices when looking for an open-source backend database. It can seamlessly replace MySQL and also provides powerful features. It also has features that MySQL currently does not have.
2. Docker is a common containerization tool for setting up the backend databases for our project.
3. Java is a programming language built on the Java Virtual Machine (JVM). It is common in many android applications.
4. Android Studio is a common tool for making Android applications that is built on top of IntelliJ from JetBrains.
5. Material Design is a CSS library that styles the visual components of the webpage (buttons, fonts, etc).

**Repository**

1. **Software**

Our software is structured in two parts: an android studio app, and a spring-boot API. The application is constructed of java and xml files that follow the standard android studio structure. It utilizes retrofit to call the API and generate RESTful requests.

The API is a spring-boot Java implementation that utilizes several Google services to implement location storing and address lookups.

1. **Documentation**

Documentation Resides within the RFXX folder. There is an additional readme in the main folder of the repository, and another readme within the API subfolder. Additionally, we have a swagger site built into our API that documents the structure of the API calls and the requirements that they have.

**Installation**

The installation process is fairly simple for the end-user. It would be as simple as opening and installing an SDK at the moment, and eventually simply installing from the Google Play Store.

Server-side, this would require purchasing or utilizing some server resources and installing the API to it, and then exposing that API to the internet for our application to call from any user device.

**Installation for a New Platform**

Installing this for a new platform would be much simpler than the original implementation. As we have an API system, the only development would need to be for the UI/UX and converting it to the new platform’s operating system. This would require learning the programming language(s) and tools for building an application for that specific OS. As the data is stored on an API, there would be no data loss or replication required.

**Further development statement**

If we had another year to continue developing this Utility Alert Application we would start with adding more settings to the application. Currently, we only have a delete account button. The plan would be to allow a normal user to apply to be a vendor. Thus allowing this account to confirm or deny reports to assure the accuracy of the application. We would also give a few more options in the settings like changing their password, changing their username, changing their email, and track the location of their current device. These simple settings would improve the user experience.

Another thing we would look into doing is adding the feature in the application to see all the reports within a mile radius. The backend/endpoint has already been set up. We just would need to add this into the map display. This would allow users to view reports within their local area.

Another thing we would like to do before wrapping up this application is changing the design some. There are lots of additional ways to tweak and improve the overall design and layout of the application. We would be able to further improve the user interface and create a more pleasing design.

Finally, we would get this app on the Google play store to increase visibility and allow for ease of installation on the user’s end. This is just a matter of completing some of these other additions and then submitting our application to Google.