Roomi

CENG 319

December 14th, 2018

Denald Demirxhiu

Jacob Ladan

Marko Javorac

## **Abstract**

This project involved developing an Android application that will eventually evolve into our Capstone Project. The main goal was to combine our previous and current course knowledge, and learn new things to demonstrate our capabilities of going from a simple idea to full-fledged functioning application that is ‘consumer grade in its quality’. We use industry standard development techniques including VCS through Github and agile workflow with the use of Gantt charts. The application design was based on the Android UI guidelines but includes some of the subtle changes. We used a realtime database by Google’s Firebase to host and interact with our application using cyber-security best practices as well as the Authentication database which provides powerful user management. We designed a modern and intuitive interface that's both aesthetically pleasing and functional. The app was tested with various Android versions to ensure reliability and compatibility. Furthermore, we ensure that the app supports two languages useful for the users, English and French. In addition to everything mentioned above, we use a variety of test cases to reduce any probability of errors while using the application. We then reflect on the skills learned throughout this project and how they have elevated our abilities.

## 

## **Introduction**

Roomi is a mobile android application that helps users manage security preferences and smart home devices. It features an intuitive graphical user interface (GUI) and real-time cloud-based data management. The application is designed to work in parallel with the Hardware Production Technology sensors/effectors. These devices include: the PN532 NFC reader/writer, the Digole LCD-Touch Display, and the ISL29125 RGB Colour Sensor. Roomi also uses the Android Things API to interact with IoT Devices resulting in a cohesive application for security and smart home management.

## 

## 

## 

## 

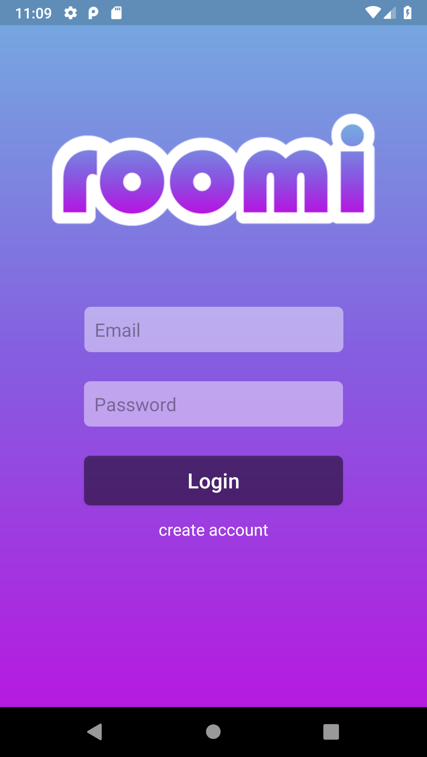
## **Software design and specification**

As we have three sensors to work with for the project in Hardware Production Technology, we only developed the functions of room access control, and light colour/brightness. The application gives users the ability to update the database with different temperature and light intensity values, select which room configurations to display and change, and even select which users are capable of accessing a room.

The mobile application communicates directly with the Firebase realtime database wherein it stores the user’s information, privileges, permissions, and room settings. The communication between the database and the application includes both reading and modifying the data, while the communication between the database and the mobile application consists of only reading the data. The database holds user account information for the purpose of Authentication. It retrieves a user ID used to associate users with their data. The user is capable of changing their username, password, and even delete their account if needed. Each of these steps requires re-authentication from the user. The communication between the application and the database is handled by several calls that request the information securely.

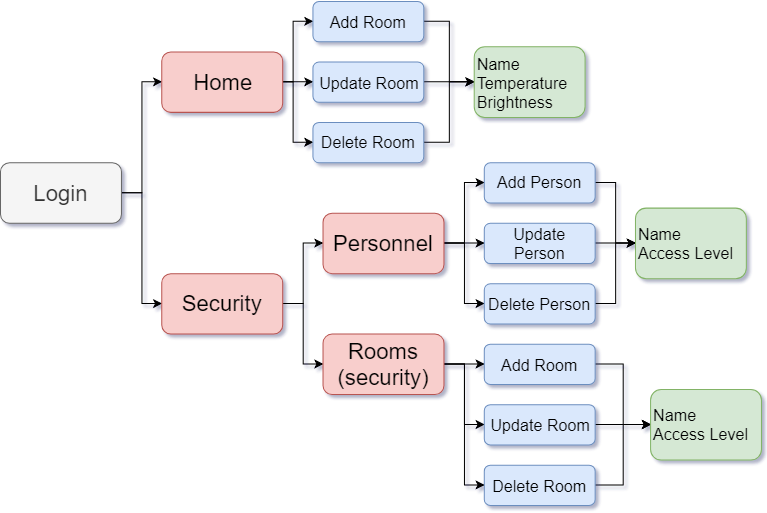
Design wise, when the user first launches the app it will be presented with a splash screen that includes the name of the application. Immediately after, it will present the user with the login screen where the user might log into their own account or create a new account (which might need to be approved by the administrator). After the user has logged in they will be presented with a greeting and two features that the user wishes to control. These two sections include *home* where the user might add new rooms and control their settings. And the *security* section where users can setup and configure rooms for enhanced security. The users are also supplied with an about page that provides a simple description of the developers of the application.

### **Application User Interface Sample**

****

**Figure 1: Application UI Sample**

### **Application Flow**



**Figure 2: Application Flow Diagram**

**Data Structures**

Our project presents three data structures that have been used to hold the data being retrieved for the application from our database. These three data structures are user, room, and personnel.

### User

* Email (String) - Used as the application’s main method of authentication
* First Name (String) - Additional information that can be used to verify a user in the event that a password or email has been compromised or forgotten. Additionally, the first name is displayed when successfully logged in as greeting
* Last Name (String) - Additional information that can be used to verify a user in the event that a password or email has been compromised or forgotten

Note: Passwords are used as a credential for login purposes, however, they are not stored in the data structure for security reasons.

### Room

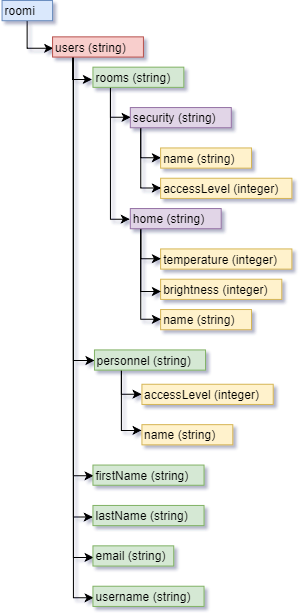
* Home Setup (Boolean) - Used by the application to indicate whether the room has been set up for temperature and light controls
* Security Setup (Boolean) - Used by the application to indicate whether the room has been setup for security purposes (RFID)
* Name (String) - Used by the application for identification in displaying room information to the user
* Temperature (Integer) - Used by the application to indicate the room’s temperature and to adjust the value. Stored as degrees Celsius, the acceptable temperatures must be in a range of 15 - 25 °C
* Brightness (Integer) - Used by the application to indicate the room’s light’s brightness level and to adjust the value. Stored as a percentage with an acceptable range between 0 (no light) and 100 (full brightness)
* Access Level (Integer) - A value between 1 and 5, used by the room to indicate the strictness of access for personnel

### Personnel

* Access Level (Integer) - Used alongside room’s access level to determine a personnel’s access permittance. If the room’s access level is higher than the personnel’s access level, then they are not permitted access
* Name (String) - Used by the application for identification in displaying personnel information to the user
* Avatar Colour (String) - Used by the application for user interface purposes allowing a customized look for each added personnel

## **Database**

The database used for the application is a real-time database hosted on Google’s firebase platform. The data is stored in JSON format. At the top level, only one branch is presented, each child holding a unique string value for registered users. Continuing, each user node holds the pertinent information used for authentication as well as two branched nodes; rooms and personnel. The rooms branch holds all the information associated with the user’s rooms that have been set up through the application. The personnel branch, similar to the rooms branch, holds all the information associated with the personnel set up by the user through our application. Figure 3 shows the database structure.



**Figure 3: Database Structure**

## **Hardware Integration**

### Digole LCD touch screen

The purpose of the LCD screen will be to display pertinent information related to the room it is associated with. The information will include the room’s name and access level. Additionally, the LCD screen will display general information such as the date. The LCD will be implemented using a Raspberry Pi 3B+. The Pi will connect to the internet using wifi and communicate directly with the database retrieving and updating any information necessary for display.

### 

### PN532 NFC reader/writer

The purpose of the NFC reader/writer will be to control access to the room it is associated with. Along with the LCD screen, the NFC will be implemented using a Raspberry Pi 3B+. The Pi will connect to the database using wifi to retrieve the access level associated with the room. Knowing this information the NFC will be able to grant access if the personnel’s access level is equal to or greater than the room’s access level or deny them if it is lower.

### 

### ISL29125 Colour Sensor

The purpose of the colour sensor will be to change room lighting based on the smart lights the users might have in their rooms. The colour sensor will be mounted in the Raspberry Pi 3B+. The database will store the colour values for the users as well as a list of colour values that have been scanned previously.

## 

## **Test Cases**

Test cases described below deal mainly with authentication and user input for sensitive fields such as the ones found within the login form and registration form.

**Test Case #1**

**Email validation**

**Purpose**: Checking that the supplied email is valid.

**Precondition**: The user must have entered an email address both in the Login page or the Registration page in order to log in or sign up.

**Test** **Procedures**: The email entered in the field will be verified whether it is valid by checking the appropriate Exception the database request throws.

**Expected** **Results**: Passing the validation will allow the user to access the application.

**Test Case #2**

**User cannot login with invalid credentials**

**Purpose:** Prevent the user from accessing the app with invalid credentials.

**Precondition:** The appropriate email and password fields must have been filled by the user

**Test Procedures:** Get the provided information from the respective fields and check the information available in the database. If the credentials are correct allow the user to login, otherwise prevent the user to login and display an error message.

**Expected Results:** Having supplied invalid credentials, deny login and display an error message

**Test Case #3**

**Ensure that the password length is at least 6 characters**

**Purpose:** Prevent the user to login or register if the supplied password is less than 6 characters

**Precondition:** The user has entered a password in the password field

**Test Procedures:** When the user presses the login or signup button get the value provided in the password field, and check the number of characters in the supplied value. If the number of characters in the password field is less than 6, display an error message and prevent login or signup, otherwise continue with the appropriate operations.

**Expected Results:** An error message is displayed if the number of characters is less than 6 and login or signup is prevented, otherwise login/signup functionality is allowed.

**Test Case #4**

**Check if the required fields are empty**

**Purpose:** Prevent the user from logging in, registering, changing the email, changing the password, or deleting the account if the required fields are empty.

**Precondition:** The user presses the appropriate button that allows the procedure to take place

**Test Procedures:** When the user presses the appropriate button to perform the specified action (login, register, delete account, etc.) get the text provided to each field. If the text obtained from the fields has a length of 0 prevent the procedure and display an error message.

**Expected Results:** When the fields are empty an error message is displayed and the appropriate action is denied.

**Test Case #5**

**Checking if the current logged in user verifies their identity before changing email, password, or deleting the account**

**Purpose:** Prevent other users of the phone/tablet from performing unauthorized operations

**Precondition:** The user must have entered the information in the specified fields and the user must press the appropriate button.

**Test Procedures:** When the button is pressed, obtain the information supplied in the appropriate fields and check if the supplied information matches the user’s credentials in the database. If it matches then allow the user to perform the desired action, otherwise prevent the action from being performed and display an error message.

**Expected Results:** Display an error message and prevent the action from being performed if the credentials don’t match the respective ones in the database. Otherwise, allow the action to be performed.

**Test Case #6**

**Checking if the user is still logged in after closing the application**

**Purpose:** Allow the user to return to their work without making them login every time they open the application.

**Precondition:** The application must have been closed

**Test Procedures:** After the application has been closed and removed from the multitasking tray, open the application and check the database if the user is currently logged in. If they are logged in then the main screen of the application is shown, otherwise, the login page is shown.

**Expected Results:** The home screen is shown after the application has been re-opened and the user is logged in, otherwise the login screen is shown.

**Test Case #7**

**Checking if the provided access level is between 0-5**

**Purpose:** To ensure access levels are consistent with the security scheme.

**Precondition:** The user is adding a room with security functionality

**Test Procedures:** When the user presses the add room button, the applications tests if the entered value is between 0 and 5.

**Expected Results:** If the test passes, the user will be allowed to add the room into the system. If it fails, a message will appear notifying the user of the rule for access level.

**Test Case #8**

**Check if the user cannot access other pages except for login or register if they are logged out**

**Purpose:** Prevent the user from accessing app functionalities if the user is logged out.

**Precondition:** The user has been logged out.

**Test Procedures:** Each time an activity other than login or sign up is accessed check if the user is currently logged in. If they are logged in then allow the activity to be displayed, otherwise the user will be redirected to login activity.

**Expected Results:** The user will be shown the appropriate activity if they are logged in, otherwise they will be redirected and shown only the login or signup activities.

**Test Case #9**

**Checking if the application is restarted after closure and reopen**

**Purpose:** Close and remove the application from memory.

**Precondition:** The user presses the exit button or quits the application by removing it from the multitasking tray.

**Test Procedures:** After the application has been closed, check if the application was restarted. If it was restarted then the application was fully closed and removed from memory. Otherwise, the application was not closed properly and the test failed.

**Expected Results:** The application restarts after pressing the exit button or removing it from the multitasking tray.

**Test Case #10**

**Check if the new person being added already exists.**

**Purpose:** Prevent the user from creating personnel that already exists and mishandling access level grants.

**Precondition:** The user is adding a person in the security system.

**Test Procedures:** The application will test if the name already exists in the database prior to adding it.

**Expected Results:** If the name is not in the system, it will be added without issue. If the name does exist, a toast will appear notifying the user that the person is already in the system.

## **Android Components**

**Viewgroups**

* ConstraintLayout
* LinearLayout
* ScrollView
* DrawerLayout

**Views**

* Button
* TextEdit
* TextView
* ProgressBar
* ImageView
* NavigationView

**Database**

* FirebaseAuth
* FirebaseUser
* FirebaseDatabase
* FirebaseReference

## **Contributions**

Marko

* Database Architecture
* Security Side layouts and activities
* Home Security software implementation
* Personnel Security software implementation
* Reports/documentation/presentation

Jacob

* UI/UX Design
* Database Architecture
* *Home* Software Implementation
* Documentation
* Planning

Denald

* UI/UX Design
* User Authentication Structure
* Account Validation
* Account Registration / Deletion / Email - Password Change Implementations
* Language Selection Implementation
* App Navigation Implementation

## 

## **Skills Learned/Conclusion**

After 14 weeks of work, there are plenty of valuable lessons that the team has learned. First and foremost was collaboration and team workflow. Learning industry standard VCS through Github and bi-weekly code reviews were key to combining each individual’s code. Enhanced time management and task delegation was important in making sure milestones were accomplished within their deadlines. General Android development workflow and the Android Studio IDE. Learning how to work with a realtime database such as Firebase.

Through designing, developing, and implementing Roomi we have elevated our software engineer skills substantially and in the process helped us further our journey to becoming career professionals.