Princess Sumaya University for Technology جامعة الأميرة سميّة للتكنولوجيا

NOORAK

Software Technical Documentation

Prepared By:

Hala Farraj, 20180664

Raghad Abdelhadi, 20180422

Supervised By:

Dr. Firas Alghanim

Project Submitted in partial fulfillment for the degree of Bachelor of Science in Software Engineering

Fall 2021-2022

Acknowledgment

First and foremost, we would like to thank our great supervisor Dr. Firas Alghanim who guided us in doing this project and throughout the whole journey. He provided us with invaluable advice and helped us in difficult times. His motivation and help contributed tremendously to the successful completion of this project.

We are indebted to all the professors for their help during working on this project. They always gave their support for the in-depth discussions about various research problems.

We are most grateful to our parents. They have always loved us and supported our every choice. As we know, they are the happiest and the proudest seeing their daughters get this degree, so we dedicate this project to them.

Table of Contents

List of figures

List of tables

Chapter 1
Introduction

1.1 Introduction

Noorak is a solution composed of a mobile application and Arduino technology that is developed to provide smart lightning control to classical traditional light bulbs. This aims to provide a luxury feature with less money and reduce energy consumption. Lights can be switched ON/OFF at sunrise and sunset automatically. Moreover, they can switched ON/OFF at specific times that a user set. A notification will be sent to a user, through a mobile application, after a specific time that the user can specify; since some lights at some rooms are switched on until that specified time has finished. Switching lights ON/OFF manually, measuring the consumed energy and reporting the energy usage weekly, all of these are some features that will be provided by our product

Institutions with a huge number of rooms face a problem with the time and effort needed to switch off all the lights in there, which also leads to spending a lot of money on electricity bills. In short, we aim to make life easier and save money.

In this technical report, guidelines are provided for the following:

Section	Description
Chapter 1	Introduction
Chapter 2	Software Project Management Plan
Chapter 3	Software Requirements Specifications
Chapter 4	Software Design Description
Chapter 5	Software Implementation

Chapter 6	System Test Document
Chapter 7	Software Documentation
Chapter 8	Conclusion and Future Work

1.2 Problem Statement

The purpose of this project is to develop a smart system to control lightning in houses and other institutions. Throughout an application, users can easily switch their light on and off even from far away distances. This simple form of act can save lots of effort and a significant amount of energy that is easily and regularly wasted when lights are forgotten in unoccupied rooms.

1.3 Evaluation of Existing Systems

Google Home, Alexa, Philips and Orange are the most popular brands when we are talking about smart lighting systems. You can control your entire home via your smartphone. These techniques are very expensive to use, at huge firms for example. Most of them don’t even support Arabic language for Arabic speakers

1.4 Objectives

The overall objective of this project is to reduce electrical consumption in this town and get a smart, easier luxurious life. It may save institutions’ and citizens’ money, time and effort. In addition, it is considered as a first step to involve Arabic language in home automation and IoT willing to see it supported in popular technologies.

1.5 Stakeholder Analysis

Stakeholder	**Interest / stake**
Citizens in Amman	Monitor lights at their place and control them and save their money.
Security Department in institutions	Save their time and effort that they usually need to turn off the institutions’ lights.
Institutions Owners	Reduce energy consumption which reduces electricity bills cost.

Chapter 2
Software Project Management Plan (SPMP)

2.1 Project Overview

2.1.1 Project Purpose and Objectives

Smart Lighting Control's purpose is to become one if not the best IoT projects in Jordan that is created by Jordanian hands.

Smart lighting in the residential environment is a part of the ‘smart home’ concept where the main goal is to provide and promote user comfort, convenience, security, and to satisfy residents’ needs. The project tends to provide a smart lighting system that supports Arabic and English languages, which is not supported by most of the popular systems. In general, smart lighting systems consist of energy-efficient light sources such as LEDs, a wireless communication network including software, and optional sensors aiming to provide optimal lighting solutions integrated with a control system. With the advancement of technology, smart lighting can now offer opportunities in addition to energy saving to users, for the provision of a comfortable atmosphere and the maintenance of user well-being.

From a broader perspective, based on the living conditions in Jordan, this project will provide an affordable solution that each building can own without even changing the electrical wiring. The completion date of SLC is set to be June, 2022. Costs of creating this solution must not exceed the limit. Moreover, the team shall take on the quality standards needed to create the application and most importantly achieve customer satisfaction.

2.1.2 Project Charter

Project Title: Smart Lighting Control Start date: 14/11/2021 Finish Date: 1/6/2022
Key Schedule Milestones: ● Deliver a prototype as a proof of concept by Nov, 30, 2021

- Complete first version of the software requirements specification (SRS) by Jan, 07, 2022.
- Complete a functional version of the software by April, 30, 2022.
- Complete a production version of the software with documentation by June, 1, 2022

Budget Information: 60 JDs have been allocated for the project, the majority of the budget is spent on the sensors and hosting the application. Additional funds may be provided if needed.

Project Manager: Hala Farraj, 20180664, hal20180664@std.psut.edu.jo.

Project objectives: This project will be made for our graduation project; thus, the objective is to have a running version of the software and a complete documentation by the specified deadline.

Main project success criteria: All specified requirements are implemented; the software is well tested and delivered on time, and the documentation is complete.

Approach:

- Allocate time for tasks as they occur
- Meet up with the supervisor every week to review the progress
- Acquire all the necessary sensors and needed
- Acquire all the necessary hosting services as they are needed

Roles and responsibilities

Name	Role	Position	Contact information
Dr. Firas Alghanim	Sponsor & supervisor	Professor	f.ghanim@psut.edu.jo
Hala Farraj	Project manager and team member	Student	hal20180664@std.psut.edu.jo
Raghad Abdelhadi	Team member	Student	rag20180422@std.psut.edu.jo

Figure 2.1: Project Charter

2.1.3 Scope Statement

Project Title: Smart Lighting Control

Justification: Citizens suffer from the high cost of living that prevents them from achieving their well-being using new technologies. Moreover, most of the new technologies that are concerned about IoT are overpriced and do not support Arabic language.

We aim to offer an affordable user well-being technology that most of the citizens can own; which aims to reduce energy consumption and monitor user's electrical bills.

- Scope Description:

  - In-scope:
    The system will allow users to control each room lighting at their place, no matter where the user is, monitor energy consumption for each room and calculate the daily use of lights.

    For more details, please refer to the Software Requirements Specifications.

  - Out-scope:
    The system will not include controlling other devices such as conditions and TVs. It will also not include a monitoring system.

- Deliverables:
  Project:
  1. Work Breakdown Structure (WBS).
  2. Risk Management and Assessment.
  3. Cost Management.
  4. Software Requirements Specification.
  5. Software Design Specification.
  6. Software Implementation Documentation.

  Product:

1. A mobile application.
2. A firebase database that connects Arduino with the mobile application
3. A back-end server
4. Arduino microprocessor and sensors that are compatible with the mobile application

- Acceptance Criteria:

  The project shall be accepted if all of the previous deliverables are delivered on time and the final product satisfies all requirements.

- Milestones:

  - 10/Jan/2022:
    1. Work Breakdown Structure (WBS)
    2. Risk Management and Assessment
    3. Software Requirements Specification
    4. Class diagrams and the ER diagram (part of the Software Design Specification)

- 24/May/2022:
  1. Cost Management.
  2. Remainder of Software Design Specification.
  3. Software Implementation Document.
  4. Software Testing Document.
  5. A mobile application
  6. A firebase database that connects Arduino with the mobile application
  7. A back-end server
  8. Arduino microprocessor and sensors that are compatible with the mobile application

- **Constraints:**

  1. Time: We are constrained by the allowed time for the project, and thus, must work as hard as possible to deliver the project within this time.
  2. Communication: We meet up two times a week either offline or online via Zoom.
  3. Scope: We are limited by the specified scope, and we can't go beyond it.
  4. Cost: Budget is limited, but we will try to make the most with it.

- **Assumptions:**

  1. Knowledge: The team has the required knowledge in mobile development, Back-end development and Arduino.
  2. Cost: Budget is enough to cover the costs of all resources that may be needed during the project.
  3. Time: Given time is enough to complete the project.
  4. Communication: Meeting up offline is needed to work on the hardware part, on the other hand, online meetings can be enough when we are working on the software part.

2.2 Risk Management and Assessment

2.2.1 Risk criteria

- **Risk Impact**

We defined the risk impact as the effect of the risk on time, cost or scope. Each risk has an impact on one of these constraints or more. The following table shows the criteria for determining the impact of a risk:

Table 2.1: Risk impact criteria

Impact	Effect on time	Effect on cost	Effect on scope
Low(1)	5 days extra of work or less	20 JDs increase in cost or less.	4 changed requirements or less

Medium (2)	6 to 14 days of extra work.	21 to 40 JDs increase in cost.	5 to 9 changed requirements.
High (3)	More than 14 days of extra work.	More than 40 JDs increase in cost.	More than 9 changed requirements.

- **Risk Probability**

We defined the risk probability as the chance of a risk to occur.
The following table shows the criteria for determining the probability of a risk:

Probability	Description
Low (1)	Between 1% and 30% chance of occurring
Medium (2)	Between 31% and 60% chance of occurring.
High (3)	More than 60% chance of occurring (but less than 100%).

- **Risk Severity**

We defined the risk severity as the probability x impact of a risk, it helped us in determining how to deal with each risk. The following table shows the description for each severity rating:

Severity	Description
1-3	Risks have a low priority and may not be taken seriously.
4-6	Risks should be addressed whenever possible
7-9	Risks should be addressed immediately.

2.2.2 Risks

The following table shows the risks that were defined for this project:

Table 2.4.: Risk with their description

Risk no.	Statement	Notes/Description	Date happened
R1	Not completing the project on time.		
R2	Allocated budget is not enough		
R3	Team members need more time to learn the technologies they are working with.		
R4	Facing difficulties while finding a way to control Arduino with a mobile application	The limited resources about this topic may cause difficulty to achieve it.	
R5	Facing difficulties with learning the hardware part	This is because there's no previous experience with using Arduino and hardware parts as we are software engineering students.	
R6	Getting injured with electric shock		

Risk Source

The only source to determine risks was brainstorming. We had a few meetings where each team member specified things that he sees as risks, and then we settled for the risks mentioned in the previous table.

2.2.3 Risk Assessment

We assessed each risk based on the criteria we defined previously in Risk Criteria section, the results were as follows:

Risk	Probability	Impact	Severity
R1	Low(1)	Medium(2)	2
R2	Low(2)	High(2)	5
R3	Medium(2)	Medium(2)	5
R4	Medium(2)	High(3)	7
R5	Medium(2)	High(3)	5
R6	High(3)	High(3)	7

2.2.4 Risk Response

We specified how to respond to each risk if it occurred based on the previous assessment, the following table shows the response for each risk:

Table 2.6: Risk response

Risk	Response	Description
R1	Avoid	Increase the daily working hours
R2	Accept	Dedicate more money to the project

R3	Mitigate	Spend extra time to learn the needed skills
R4	Avoid	Search for more resources to solve this problem
R5	Mitigate	Assign extra time to learn more about sensors and microprocessors
R6	Mitigate	Being aware when working with electrical parts

2.3 Cost Management

The following table shows everything that paid for during this project:

Table 2.7: Spendings during the project

Item	Description	Cost	Date
Arduino Uno	A microprocessor that controls all the sensors	9 JDs	30/10/2021
Arduino Udemy course	A course about using and programming Arduino microprocessor and its sensors	15 JDs	2/11/2021
Sensors	Sensors that are needed to achieve the project’s goal	8 JDs	3/11/2021

NodeMCU	A microprocessor that can be connected to WiFi	9 JDs	3/11/2021

2.4 Software Development Life Cycle (SDLC)

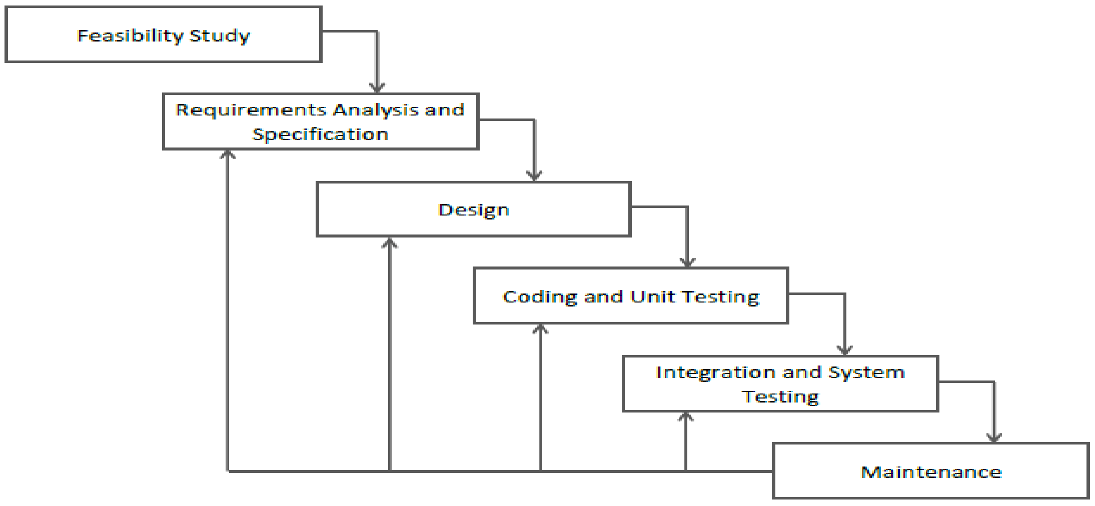

Feasibility Study
Requirements Analysis and Specification
Design
Coding and Unit Testing
Integration and System Testing
Maintenance

Figure 2: Iterative Waterfall Model

The SDLC for the project is Iterative. We had to start with a prototype as a proof of concept then by time we kept adding new features and editing the prototype. It may seem exhaustive but without a proof of concept everything would be more complex

2.5 Tools

This section shows the tools that were used during the project, these tools helped make our work easier.

- Zoom: This is the main communication tool, because it is simple and effective, it has voice chat and screen sharing and messaging, which is exactly what we needed.

- Microsoft Office Word: We used word to write drafts for all sections of the project, because it is easier to modify documents written using word.

- Google Drive: We used Google drive to store a back-up of all documents, just in case anything goes wrong with our devices.

- Arduino IDE: This is the main working environment that is used to program Arduino Uno and its sensors.

- Github: It is a version control system for software developers, we used it throughout the development to track changes and store our code remotely in case anything goes wrong with our devices.

- Android Studio: It is an IDE dedicated for developing apps that use the Android OS, we used this IDE to develop the flutter mobile application.

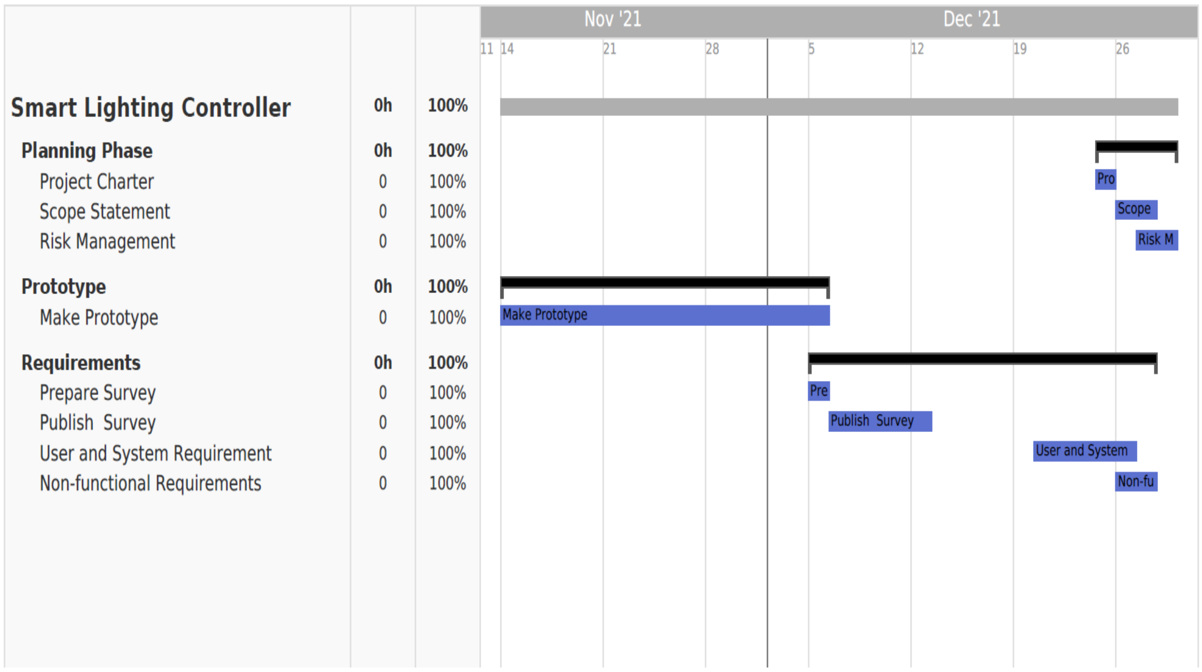
2.6 Tasks and Project Timeline

2.6.1 Task Allocation

#	Task	Owner	Date planned	Date actual complete	Status
1	Smart Lighter Controller	Raghad, Hala	14-11-2021	1-6-2021	☺☹

#	Task	Owner	Date planned	Date actual complete	Status
2	Planning phase	Raghad, Hala	23-11-2021	28-11-2021	☺
3	Project charter	Raghad, Hala	23-11-2021	23-11-2021	☺
4	Scope statement	Raghad, Hala	26-11-2021	27-11-2021	☺
5	Risk management	Raghad, Hala	27-11-2021	28-11-2021	☺
6	Requirements	Raghad, Hala	5-12-2021	10-1-2022	☺
7	Prepare Survey	Raghad, Hala	5-12-2021	6-12-2021	☺
8	Publish survey	Raghad, Hala	6-12-2021	12-12-2021	☺
9	Brain Storming	Raghad, Hala	5-11-2021	14-11-2021	☺
10	User and System requirements	Raghad, Hala	21-12-2021	26-12-2021	☺
11	Requirements Interdependencies	Raghad, Hala	14-11-2021	14-11-2021	☺
12	Non-functional Requirements	Raghad, Hala	26-12-2021	27-12-2021	☺
13	Requirements Traceablility	Raghad, Hala	14-11-2021	14-11-2021	
14	Requirements Priotirization	Raghad, Hala	14-11-2021	14-11-2021	

2.6.2 Project Timeline



Smart Lighting Controller	**0h**	**100%**
Planning Phase	**0h**	**100%**
Project Charter	0	100%
Scope Statement	0	100%
Risk Management	0	100%
Prototype	**0h**	**100%**
Make Prototype	0	100%
Requirements	**0h**	**100%**
Prepare Survey	0	100%
Publish Survey	0	100%
User and System Requirement	0	100%
Non-functional Requirements	0	100%

Chapter 3
Software Requirements Specification (SRS)

3.1 Introduction

The software requirements specification will fully describe the behavior of the Smart Lighting Control. The main objective of this document is to illustrate the requirements of the system. This document gives a detailed description of both functional and non-functional requirements. It will also explain the system's constraints, interface and interactions with other external applications.

3.2 Requirements Elicitation

This section shows the various methods that we used to collect and write requirements for the system.

3.2.1 Survey

We published a survey that targeted people interested in attending events, we asked about their opinion regarding multiple features that we had in mind. Results for this survey can be found in Appendix A: Survey Results.

3.2.2 Brainstorming

On multiple occasions, we met together to discuss what features we are able to include, and which features we are not. Then we started writing the requirements together while keeping in mind that we covered all aspects that we wanted to cover. After writing the first version of the requirements, we reviewed them multiple times to discover any missing and incomplete features.

3.3 User Requirement

3.3.1 Functional user requirements

[UR-1] Users should be able to set up the controlled rooms by providing the number of rooms and their names.

[UR-2] Users should be able to check their place lights status.

[UR-3] Users should be able to view statistical information about energy consumption for each room and the expected cost.

[UR-4] Users should be able to control lights and their intensity.

[UR-5] Users should be able to control light colors.

[UR-6] Users can specify a time to get a notification based on it when a room is empty.

[UR-7] Users should be able to specify times that lights automatically turn on based on it.

[UR-8] the user's location is compared to the home location to turn on a specific light that he specified

3.3.2 Non-functional user requirements

[UR-9] Mobile application must be user-friendly and easy to use by non-technical users

[UR-10] Users should be able to receive mostly correct expected statistics.

[UR-11] The system must be highly secure.

3.4 System Requirements

3.4.1 User's application system requirements

[FSR-1] Users should be able to set up the controlled rooms by providing the number of rooms and their names:

FSR-1.1 Number of rooms
FSR-1.2 Rooms names

[FSR-2] Users should be able to check their place lights status:

FSR-2.1 View lights status.

[FSR-3] Users should be able to view statistical information about energy consumption for each room and the expected cost

FSR-3.1 Selecting a single room or multiple rooms to check their energy consumption.

FSR-3.2 Showing Reports and statistics about each room.

[FSR-4] Users should be able to control lights and their intensity. This can be done be selecting the required room to be controlled and change its status and intensity based on the user's desire (such as reading mode, sleeping mode, cooking mode, etc.)

[FSR-5] Users should be able to control light colours. If a room's lights support RGB LED, a user can change their color directly by selecting the desired room.

[FSR-6] Users can specify a time to get a notification based on it when a room is empty.

[FSR-6.1] specify a time that a notification must be sent by, when it's empty

[FSR-6.2] Specify if lights should be switched off directly after this notification or based on the user's selection

[FSR-7] Users should be able to specify times that lights automatically turn on based on it (in evening lights turn on automatically and on sunset all lights turn off automatically)

[FSR-8] The user's location is compared to the home location to turn on a specific light that he specified.

FSR-8.1 Enter the home location

FSR-8.2 Give access to the app to get the phone's location

FSR-8.3 Specify the desired room to be turned on automatically based on the two locations

3.4.2 Non-functional system requirements

[NFSR-1] The system must be user-friendly and easy to use by non-technical users by having as less views as possible on each screen

[NFSR-2] Users should be able to receive mostly correct expected statistics. Statistics should be done based on the tax and currency of the user's country and show the difference between energy consumption for each month.

[NFSR-3] The system must be high secure. Due to the critical information that a user provides such as his home location.

3.5 **Platform Requirement**

For IOS devices, flutter supports 9.0 version and later. For Android devices, flutter supports Android 4.1 or higher.

However, the application require access to the internet to be up-to-date with the firebase status.

3.6 Hardware requirements

3.7 Requirement Management

3.7.1 Requirements Interdependency

Requirements interdependency matrix shows how system requirements are related to each other based on different criteria.

Table 3.1: Requirements traceability matrix Part 1

SR/UR	UR-1	UR-2	UR-3	UR-4	UR-5	UR-6	UR-7	UR-8
FSR1	x							
FSR2		x						
FSR3			x					
FSR4				x				
FSR5					x			
FSR6						x		
FSR7							x	
FSR8								x
FSR9								x

SR/UR	UR-9	UR-10	UR-11
NFSR10	x		
NFSR11		x	
NFSR12			x

Table 3.3: Requirements traceability matrix part 2

3.7.2 Requirements Interdependency

Requirements interdependency matrix shows how system requirements are related to each other based on different criteria.

Table 3.3: Requirements Interdependency

Req.	Similar to (similarity %)	Requires	Conflict with	Increase value	Decrease value	Increase cost	Decrease cost
FSR1							
FSR2				FSR1			
FSR3				FSR1 /4	FSR2		
FSR4	Similar50% FSR5				FSR5		FSR5
FSR5				FSR1			
FSR6				FSR2			
FSR7					FSR4/ 5		
FSR8				FSR9			

FSR9							

3.7.3 Requirements Prioritization

Requirements prioritization was done using MoSCoW technique, which gives each requirement a priority as follows:

M (Must Have): critical requirements that must be delivered in the system.

S (Should Have): Important requirements, but less critical than Must Have.

C (Could Have): desirable requirements that are not that important but could improve the quality of the system. Implemented if there is extra time and resources.

W (Wish to Have): possible requirements but not important and are unlikely to be implemented in the first release of the system.

We chose this technique because the number of requirements is few and does not require difficult technology and calculations.

Requirement	Priority
UR1	M
UR2	M
UR3	C
UR4	S
UR5	W
UR6	S
UR7	W
UR8	W

3.8 Use Case Diagrams

3.8.1 Application use case diagram

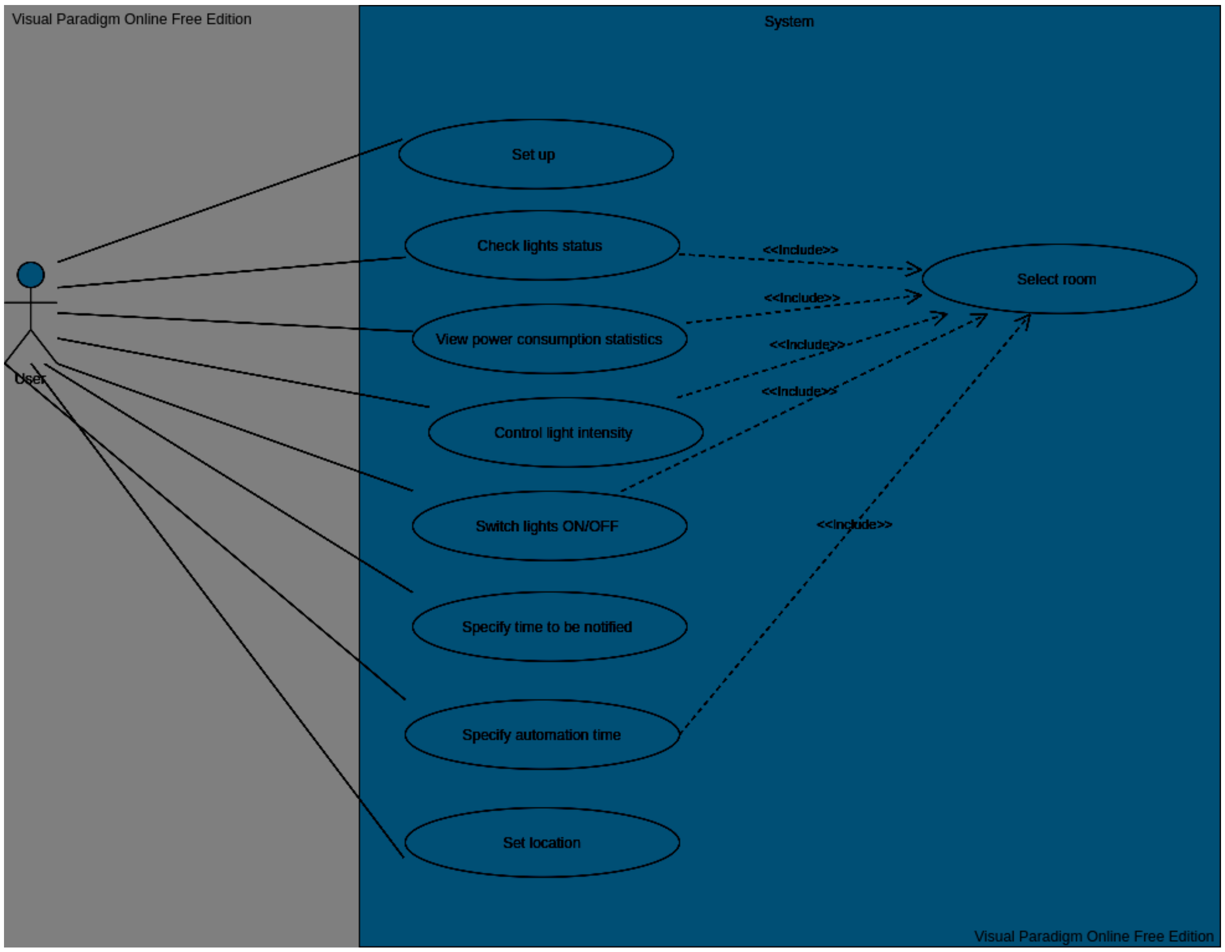


Figure 3.1 shows the use case diagram for the application, which contains all the services that are provided by our application to the user. The user can use these use cases to control lights at his place.

3.8.2 Application use case diagram

Title: Switch lights ON/OFF
Date: 12/1/2022

Review date: 15/3/2022
Actor: User
Description: This use case shows how users can control lights for a specific room
Extension points: None
Pre-conditions: ● The user has to have his account set up with the NodeMCU ● The user has to select a specific room to control its lights status
Primary path: 1. The user taps on the room that he wants to check its lights status 2. The system responses with the room light status
Alternative path(s): If the light is already off, the light bulb appears as a gray bulb, hence: 3. The user taps on the light bulb 4. The light appears as a yellow bulb and the light is switched on. If the light is already on, the light bulb appears as a yellow bulb, hence: 5. The user taps on the light bulb 6. The light appears as a gray bulb and the light is switched off.
Exception path(s): If the user doesn't select a room, the system sends a notification to him to select one.
Post-conditions:

The light status is switched

Figure 3.8.2 Use case description for ‘Switch lights ON/OFF’ use case

Title: View power consumption statistics
Date: 12/1/2022
Review date: 15/3/2022
Actor: User
Description: This use case shows how users can view the power consumption statistics
Extension points: None
Pre-conditions: ● The user has to have his account set up with the NodeMCU ● The user has to select a specific room to view its statistics
Primary path: 1. The user taps on the room that he wants to check its statistics 2. The system shows a statistical graph and information about the electrical consumption for the selected room. 3. The system shows the expected cost of electrical consumption in JDs
Alternative path(s): None

Exception path(s): If the user doesn't select a room, the system sends a notification to the user to select one.
Post-conditions: ● Statistics and expected cost is shown to the user.

Figure 3.8.3 Use case description for 'View power consumption statistics' use case

Title: Specify automation time
Date: 12/1/2022
Review date: 15/3/2022
Actor: User
Description: This use case shows how users can specify a time to let lights switch on and off automatically based on it.
Extension points: None
Pre-conditions: ● The user has to have his account set up with the NodeMCU ● The user has to select a specific room to set a time for automation
Primary path: 1. The user taps 'Set Time' button 2. The system shows Set Time screen

3. The user enters switching on time 4. The user enters switching off time 5. The user specifies the days to apply this process in 6. The user taps on 'Save Changes' button 7. The system shows a pop up telling the user that the operation is done successfully with the remaining time for the first automatic operation
Alternative path(s): None
Exception path(s): ● If the user doesn't select a room, the system sends a notification to the user to select one. 6a- The user taps on 'Cancel' instead of 'Save Changes'
Post-conditions: The lights of the selected room are switched ON/OFF automatically based on the time that the user has set up.

Figure 3.8.4 Use case description for 'Specify automation time' use case

Figures 3.8.2, 3.8.3 and 3.8.4 show the description for the three use cases that appeared in the application use case diagram. 'Switch lights ON/OFF' use case description shows the exact steps that a user must execute to switch on/off lights.

'Specify automation time' use case description shows the exact steps that the user needs to carry out to set a time to get lights switched ON/OFF automatically based on.

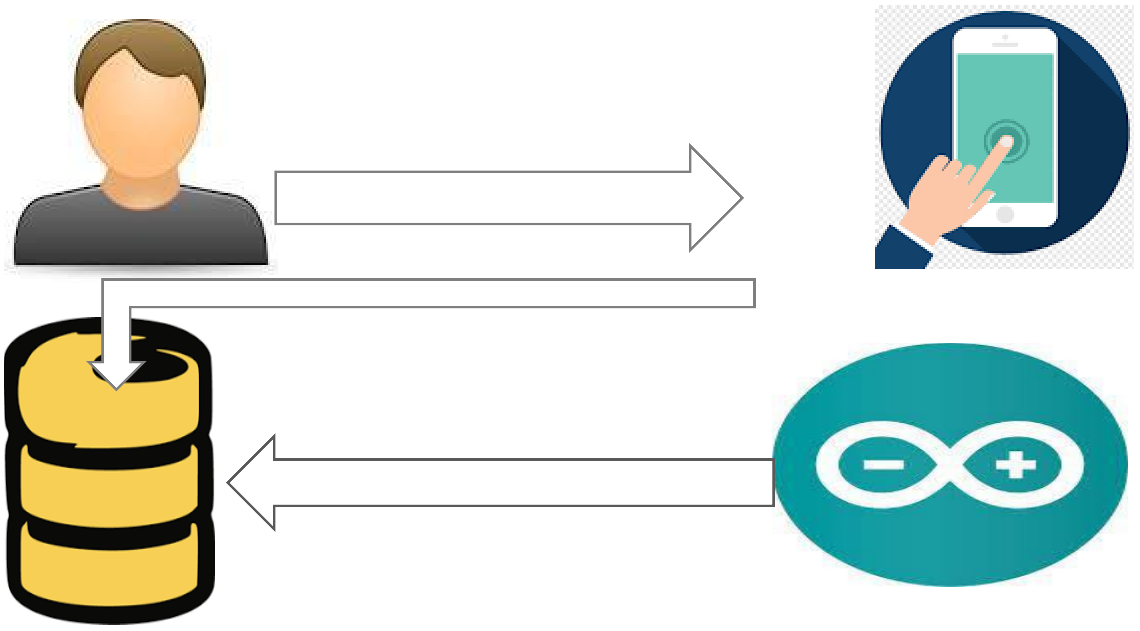
Chapter 4
Software Design Description (SSD)

4.1 Introduction

This section will define a high level design and technology decisions of the Noorak Application. Application Architecture is a way to give the overall view of a system. This allows for the reader and user of the document to orient them to the design and see a summary before proceeding into the details of the design.

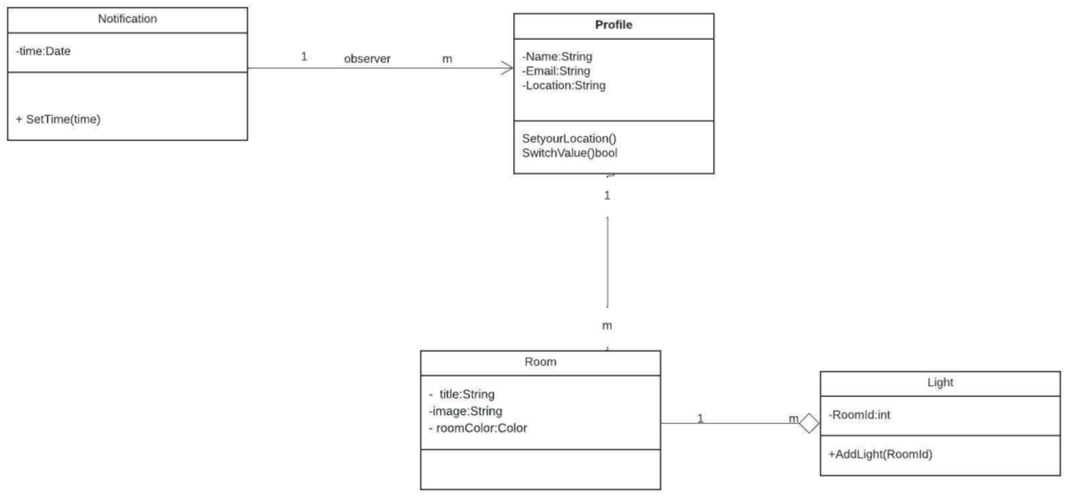
4.2 System Architectural Diagram

Overall structure for the Noorak System:

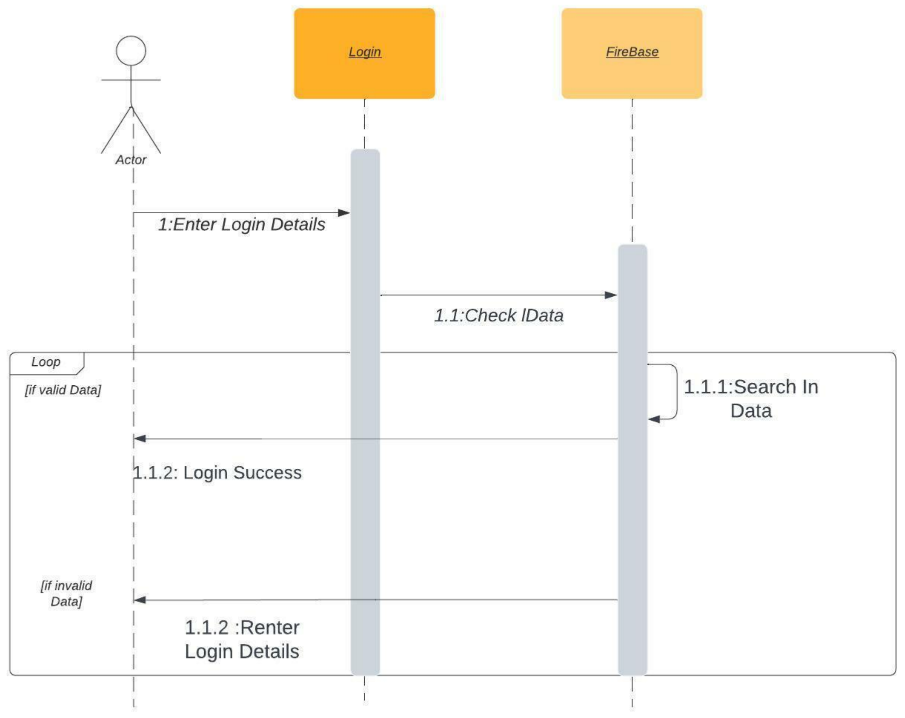


4.3 Class Diagram

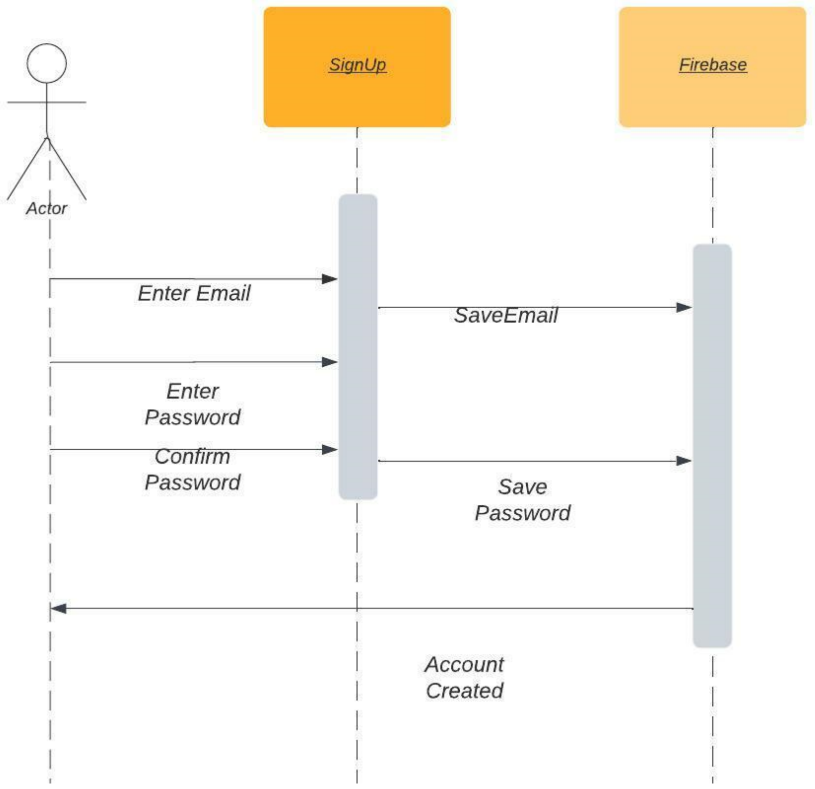
A class diagram is used to describe which classes the software product will be composed of. The class diagram describes the relation between the classes which are presented, and the way they interact together.


Notification
-time:Date
+ SetTime(time)
1
observer
m
Profile
-Name:String
-Email:String
-Location:String
SetyourLocation()
SwitchValue()bool
1
m
Room
- title:String
-image:String
- roomColor:Color
1
m
Light
-RoomId:int
+AddLight(RoomId)

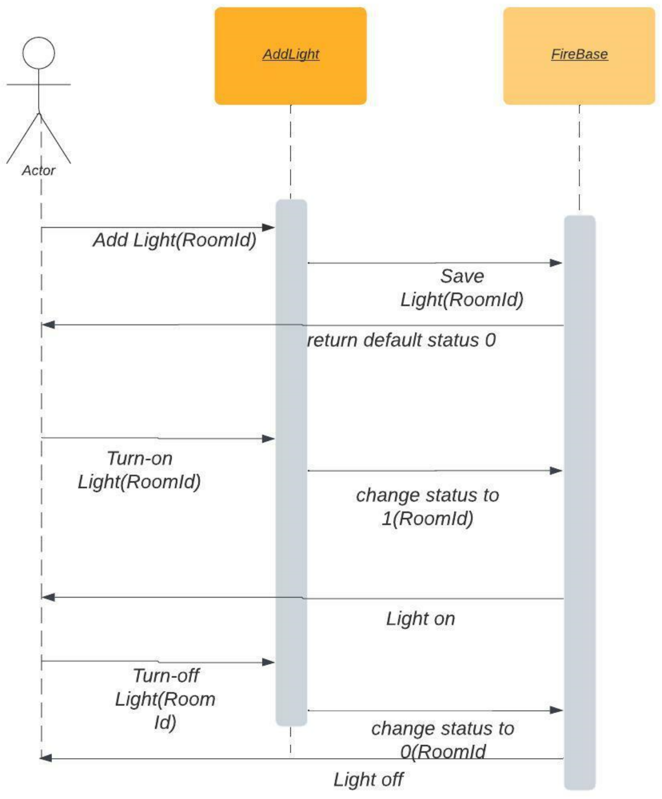
4.4 Sequence Diagram



Login sequence Diagram

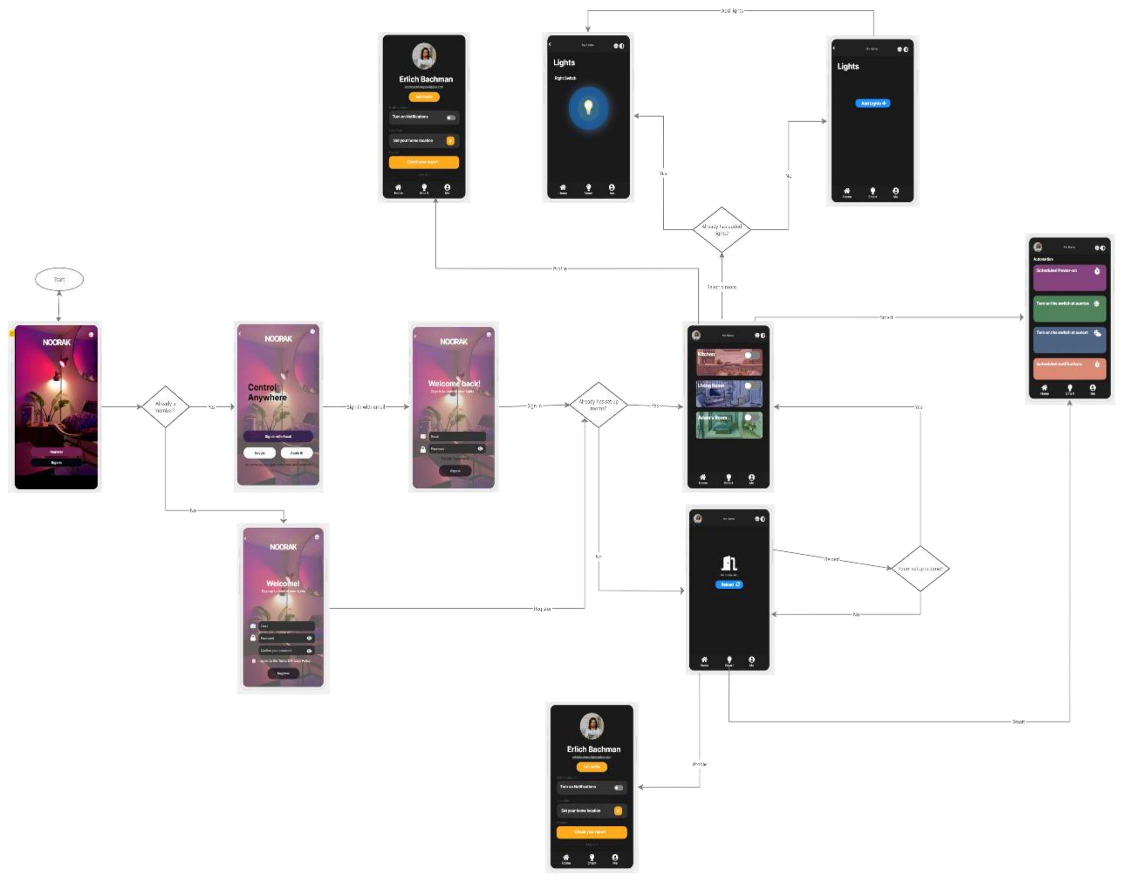


Signup Sequence Diagram



Manual control sequence diagram

4.5 System Layouts



Chapter 5
Software Implementation

5.1 Introduction

In this chapter, we are going to discuss the software implementation of the system; Starting from explaining the programming languages and technologies that are used, going through the whole process of choosing these languages, technologies and explaining all the external libraries that are used. Then, discussing some features and the logic behind them.

5.2 Technologies and programming languages

5.2.1 Front-End

Flutter is used for front-end development. We preferred Flutter over Java and Kotlin for many reasons. Flutter is the most trending cross-platform technology so choosing to program using it could improve our skills that are needed by the labor market. Moreover, Flutter is very compatible with firebase, since they are both developed by Google. We use firebase RTDB to store and retrieve the real time data. One more thing to mention, it provides such a nice UI toolkit that serves our goals.

5.2.2 Back-End

Dart is used as a back-end language for a flutter app. The reason behind choosing it, is that dart is very compatible with flutter and firebase so it can be used effectively. Moreover, this is a great opportunity to learn a new language by applying its concept.

For the database, RTDB has been chosen since our data needs to be transferred in a real-time database. This has been done to serve the user experience, so he can control his place without any delay.

5.2.3 Hardware

Our project needs a microcontroller to get the desired features done. Arduino controllers and sensors are developed using Arduino IDE which uses C++ as a development language; Since C++ is a middle-level language which can control hardware in a very efficient way.

5.3 External Libraries

5.3.1 Mobile App Libraries

These are a combination between the external used libraries for the front-end and the back-end.
They have been used to achieve and support different functionalities. Some of them were used to connect the logic of the app to the RTDB.

```
dependencies:
  flutter:
    sdk: flutter
  lite_rolling_switch: ^0.1.1
  flutter_switch: ^0.3.2
  flutter_localizations:
    sdk: flutter
  firebase_core: ^1.18.0
  flutter_local_notifications: ^9.6.1
  firebase_database: ^9.0.16
  timezone: ^0.8.0
  flutter_countdown_timer: ^4.1.0

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  firebase_auth: ^3.3.20
  uuid: ^3.0.6
```

5.3.2 Hardware Libraries

Most of the libraries were used to access the RTDB in Firebase. Moreover, some libraries are used to connect to the internet and reach the store data in the NodeMCU's memory.

```
#include <Arduino.h>
```

```
#include <FirebaseESP8266.h>

#include <ESP8266WiFi.h>

#include <ESPAsyncTCP.h>

#include <ESPAsyncWebServer.h>

#include <ArduinoJson.h>

#include <FS.h>

#include <NTPClient.h>

#include <WiFiUdp.h>

#include "ESP8266TrueRandom.h"

#include "addons/TokenHelper.h"

#include "addons/RTDBHelper.h"
```

5.4 Implementation Details

5.4.1 Mobile App Code

1- Register a new account

First, the user must fill up all the text fields in the Register screen. Once, "Register" buttons is tapped, the following code will be executed:

```
//----------------Register Button----------------

 Container(
   width: MediaQuery.of(context).size.width,
   height: 50,
   margin: const EdgeInsets.fromLTRB(0, 10, 0, 20),
   decoration: BoxDecoration(borderRadius: BorderRadius.circular(90)),
   child: ElevatedButton(
     onPressed: () async {
```

```
                    // if the validation is correct, the user can is registered
                    if(formKey.currentState!.validate()){
                      try{
                        errorMessage='';
                       await FirebaseAuth.instance
                      .createUserWithEmailAndPassword(
                        email: _emailTextController.text,
                        password: _passwordTextController.text)
                    .then((value) async{
                      print("Created New Account");
                      final DatabaseReference db
=  FirebaseDatabase.instance.ref(apiServices.get_UID());
                      await db.update({"alias":value.user!.email!.split('@').first});
                       // ignore: use_build_context_synchronously
                       Navigator.pop(context);

                       });

                      }

                      on FirebaseAuthException catch(error){
                          errorMessage = error.message!;
                      }

                    }
                  },
```

FirebaseAuth is the library used to authenticate a new user in Firebase using Dart. This code demonstrates the exceptions handling such as an existing account is being registered again by a user, or the fields are empty. If everything went as expected, the new account would be created and authenticated with the firebase users.

An account is a must to have to get all the rooms and lights defined under this account ID in the RTDB. Moreover, the textfields checks the inputs based on a given Regex as shown below :

```
    TextFormField(
        controller: _emailTextController,
        validator:(value){
         if(value!.isEmpty){
            return 'Email is required
          }
       else if(!RegExp(r'^[a-z0-9]+@[a-z]+\.[a-z]{2,4}$').hasMatch(value))
                 {
           return 'Enter correct email';
          }
         else{
              return null;
            }
          },
)


               TextFormField(
             controller: _passwordTextController,
             validator:(value){
                  if(value!.isEmpty ){
                    errorMessage=' ';
                    return 'Password is required!';
                  }
                 else if (RegExp(r'^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-
Z]).{8,}$').hasMatch(value)){
                    return 'invalid password';
                  }
                  else if(value.length < 8){
                     return 'Password must be atleast 8 characters';
```

```
                      }
                    else{
                      return null;
                    }
                  },)
```

2. Control all the lights in one room by one switch

```
 FlutterSwitch(
              width: 50,
              height: 25,
              valueFontSize: 35.0,
              toggleSize: 50.0,
              value: _switchValue,
              borderRadius: 30.0,
              onToggle: (value) {
              final DatabaseReference db =
FirebaseDatabase.instance.ref(apiServices.get_UID()).child("rooms").child(widget.
roomID).child("lights");

                if(value == true){
                  get_lights(db,1);

                  // rooms_listener(db);
                }

                else{
                  get_lights(db,0);
                }
                setState(() {
                  _switchValue = value;
                });

                widget.onToggle();
              },
            ),
```

In the ‘Home’ page, the user gets all the defined rooms under his UID in the RTDB and can control a whole room’s lights status using a switch button. This code shows the logic behind the on toggle() function that executes when the button is toggled. A databaseReference is created to reach the “lights” child in the RTDB based on the user’s ID and the selected roomID. get_lights(DatabaseReference db, int led_status) is used to bring all the lights in the database and set their status based on the switch value, which applies the required feature. get_lights(DatabaseReference db, int led_status) logic is shown below:

```
void get_lights(DatabaseReference db, int status) async{
    final DatabaseEvent event = await db.once();
  final Map data = event.snapshot.value as Map;
  final List lights  = data.keys.toList();
  for(String light_id in lights){
    // print(data[light_id]['led_status']);
    await db.child(light_id).update({'led_status': status});
  }

 }
```

3. Add a new light

After having a new account and a defined room by the NodeMCU button, the user needs to define new lights based on the number of the physical switches in the desired room. To get this done, a popup will show to the user when he presses on the add icon which has this backend:

```
Future<String?> openDialog() {
  final TextEditingController controller = TextEditingController();
  return showDialog<String>(

  context: context,
  builder: (context)=> AlertDialog(
   title: Text("New Light"),
   content: TextField(
    autofocus: true,
    decoration: InputDecoration(hintText: 'Enter your light alias '),
    controller: controller,
   ),
   actions: [
    TextButton(
     onPressed: () {
              // ignore: unnecessary_null_comparison
              if(controller.text == null || controller.text.isEmpty) return;
              apiServices.addLight(controller.text, widget.roomID);
              Navigator.of(context).pop();
     },
     child: Text('Submit'))
   ],


  )
  );
 }
```

Once ‘Submit’ is tapped, a new light will be created in the RTDB under the selected room ID. If the alias was null, no room is created.

```
Future addLight(String alias, String roomID) async
 {
  final DatabaseReference  db =
FirebaseDatabase.instance.ref(get_UID()).child("rooms").child(roomID).child("ligh
ts");

  final DatabaseEvent event = await db.once();
  late Map data;
  final List exitsingRooms;
  const Uuid uuid = Uuid();
  late String newUUID;
  try
  {
    data = event.snapshot.value as Map;
    exitsingRooms = data.keys.toList();
    while(true)
    {
      newUUID = uuid.v1().toString();
      if(!exitsingRooms.contains(newUUID))
      {
        await db.update( {newUUID: {"alias" : alias}});
        await db.child(newUUID).child("led_status").set(0);
        break;
      }
    }
    }
  catch(_)
  {
    newUUID = uuid.v1().toString();
    await db.update( {newUUID: {"alias" : alias}});
    await db.child(newUUID).child("led_status").set(0);
  }


 }
```

4. Show automation details that has been set for a specific room

```
 StreamBuilder(
        stream: apiServices.roomDetails(widget.roomID),
        builder:  (BuildContext context, AsyncSnapshot snapshot){

        if(!snapshot.hasData)
        {
          return Center(child: CircularProgressIndicator(color: Colors.white,),);
        }

        Map detailsData=<String,String>{ };
        try
        {
          final Map data = snapshot.data.snapshot.value;
          final List keys = data.keys.toList();

          if(keys.contains("power-on")){
             detailsData["Power-on"] = data[keys[keys.indexOf('power-on')]].toString();
          }
          if(keys.contains("power-off")){
             detailsData["Power-off"] = data[keys[keys.indexOf('power-
off')]].toString();
          }
          if(keys.contains("scheduled-notifications")){
             detailsData["Scheduled-notifications"] =
data[keys[keys.indexOf('scheduled-notifications')]].toString();
          }
           if(keys.contains("sunrise")){
             detailsData["Sunrise"] = "is set";
          }
           if(keys.contains("sunset")){
             detailsData["Sunset"] ="is set";
          }

            if(detailsData.isEmpty){
          return Center(child: Text("No automation details yet..",
                    textAlign:TextAlign.center,
                    style: TextStyle(
                      color: Colors.white,
                      fontSize: 15,
                    ),
                    ),
                    );
          }
```

A streamBuilder is used to create a new widget for each ‘Smart Feature’ to show its details. This helps the user to check what smart feature he has set for each room. If no smart features are set, the screen shows ‘ No automation details’.

5. Scheduled Power-on

```
child:  StreamBuilder(
        stream: apiServices.rooms(),
        builder: (BuildContext context, AsyncSnapshot snapshot)
        {
          if(!snapshot.hasData)
          {
            return const Center(child: CircularProgressIndicator(),);
          }
          // final Map data = snapshot.data.snapshot.value;

           try{
              final Map data = snapshot.data.snapshot.value;
              final List roomKeys = data.keys.toList();
              return Column(
                children: [
                  Flexible(
                    child: ListView(
                      // crossAxisCount: 2,
                      children: List.generate(roomKeys.length, (index) =>  Rooms(image:
"3596801", title: data[roomKeys[index]]['alias'], roomColor: Color.fromARGB(186,
232, 232, 232),
                      onSelect:(){
                        selectedRooms.contains(roomKeys[index])?selectedRooms.remove(
roomKeys[index]):selectedRooms.add(roomKeys[index]);

                      }
                      )),
                    ),
                  ),
                  ElevatedButton(
                    onPressed: (){
                      for(String id in selectedRooms){
                        apiServices.setFeature("power-off",  _selectedTime.toString(), id);
                      }

                      openDialog();

                    }
                  , child: Text("Set")
                  )
                ],
              );
          }
          catch(_) {
              return ListView(
        children: const [
          SizedBox(
            height: 120,
          ),
          Padding(padding: EdgeInsets.symmetric(vertical: 150,horizontal: 150),
          child: Icon(
                Icons.meeting_room_sharp,
                color: Colors.white,
              size: 100,
            ),

          ),
          SizedBox(height: 10,),
          Padding(padding: EdgeInsets.symmetric(vertical: 250,horizontal:160 ),
          child :Text("No rooms yet...",
          textAlign:TextAlign.center,
          style: TextStyle(
            color: Colors.white,
            fontSize: 10,
          ),
```

This demonstrates how a scheduled power-on feature is done for any room. Simply put, the user can choose the desired time to switch on a room by selecting the room. Moreover, this can be set for many rooms at the same time.
The streamBuilder uses a created function – apiServices.rooms() that brings all the rooms based on the UID – for the signed in account and apiServices.setFeature() to set the selected feature under the selected room(s) ID(s). They are implemented as follows:

```
// returns a stream of rooms
 Stream<DatabaseEvent> rooms()
  {

    final FirebaseDatabase db = FirebaseDatabase.instance;
    return db.ref(get_UID()).child("rooms").onValue.asBroadcastStream();
 }

  Future setFeature(String feature, String value, String roomID) async{
   final DatabaseReference db
=  FirebaseDatabase.instance.ref(get_UID()).child("rooms").child(roomID);
   await db.update({feature:value});
  }
```

6. Switch-on at sunset

```
StreamBuilder(

stream: apiServices.rooms(),

builder: (BuildContext context, AsyncSnapshot snapshot)
{

  if(!snapshot.hasData)
  {
    return const Center(child: CircularProgressIndicator(),);
  }


   try{
      final Map data = snapshot.data.snapshot.value;
      final List roomKeys = data.keys.toList();
       return Column(
        children: [

     const Padding(
       padding: EdgeInsets.only(left: 10,top: 10),
       child: Text(

        "Turn on the switch at sunset",

        style: TextStyle(

            fontSize: 20,
            color: Color.fromARGB(255, 255, 255, 255),
            fontWeight: FontWeight.bold),
    ),
     ),
            const Text(
             "Sunset is at :  7:46 PM ",
             style: TextStyle(
                 fontSize: 20,
                 color: Colors.grey,
                 fontWeight: FontWeight.bold),
           ),

                 Flexible(
                   child: GridView.count(
                     crossAxisCount: 2,
                     children: List.generate(roomKeys.length, (index) =>  Rooms(image:
"3596801", title: data[roomKeys[index]]['alias'], roomColor:Color.fromARGB(166,
168, 255, 196),
                     onSelect:(){
                       selectedRooms.contains(roomKeys[index])?selectedRooms.remove(
roomKeys[index]):selectedRooms.add(roomKeys[index]);

                     }
                     )),
                   ),
                 ),
                 ElevatedButton(
                   onPressed: (){
                     for(String id in selectedRooms){
                       apiServices.setSunsFeature("sunset", true , id);
                     }
                     openDialog();
                   }
                 , child: Text("Set")
                 ),

               ],
            );
        }
```

Once ‘Set’ button is tapped, a new feature is set under the roomID in RTDB using setSunsFeature() which is implemented as the following code shows:

```
 Future setSunsFeature(String feature, bool value, String roomID) async{
   final DatabaseReference db
=  FirebaseDatabase.instance.ref(get_UID()).child("rooms").child(roomID);
   await db.update({feature:value});
  }
```

5.4.2 Hardware Code

1. Store and retrieve data using SPIFFS

SPIFFS is a library that is used to store and upload files to the microcontroller’s memory.

```
void loadConfig()
{
  File configfile = SPIFFS.open("/config.json", "r");
  deserializeJson(Config, configfile);
  configfile.close();
}

void saveConfig()
{
  File configfile = SPIFFS.open("/config.json", "w");
  serializeJson(Config, configfile);
  configfile.close();
}
```

2. Sunrise feature

This feature is done by reading the sunrise feature that has been set by the user using the mobile application. If it has been set ‘true’, light status is changed based on the photoresistor readings.

```
//*********** sunrise feature ***********************

String get_sunriseValue(String path)
{
  path+="sunrise";
  if (Firebase.getString(fbdo, path)) {

       Serial.println("get_sunriseValue is called");

       return fbdo.stringData();
      }
      else{
        return "N/A";
      }
}

void sunrise(String path){
    path+="lights/light1/led_status";
   photo_value = analogRead(photoresistor);
    Serial.println(photo_value);
    if (photo_value > 800)    // sunrise
    {
      Firebase.RTDB.setInt(&fbdo,path,0);
      delay(1000);
    }
}
```

3. Get time

Microcontrollers do not have a built-in clock that computers have. But, since NodeMCU can get connected to the internet, time can be caught using the NTP protocol.

```
// get the current time using ntp protocol and set the AM/ PM
String get_time()
{
  timeClient.update();
  String timee="\"";
  int currentHour = timeClient.getHours();

  if(currentHour > 12)
  {
   currentHour-=12;

  }
  if(currentHour == 0)
  {
    currentHour = 12;
  }
//  Serial.println(currentHour);
  timee+=String (currentHour);
  timee+=":";

  int currentMinute = timeClient.getMinutes();
  if(currentMinute<10){
    timee+="0";
  }
//  Serial.print("Minutes: ");
//  Serial.println(currentMinute);
  timee+=currentMinute;

  return timee;
}
```

```
String set_AM_PM(){
  timeClient.update();
  String am_pm="";
  int currentHour = timeClient.getHours();
  Serial.print("This is the current hour from AM_PM funtion:");
  Serial.println(currentHour);
  if(currentHour>=12)
  {
    am_pm=" PM";
  }
  else if(currentHour == 0)
  {
    am_pm=" AM";
  }
  else
    am_pm=" AM";

    Serial.println(am_pm);

   return am_pm+"\"";
}
```

4. Create a new room with a random ID

```
//Create new room when button is pushed

String randomID_generator(){

//  ESP8266TrueRandom is a library that is used to generate random UIDs
  ESP8266TrueRandom.uuid(uuidNumber);
  String uuidStr = ESP8266TrueRandom.uuidToString(uuidNumber);
  Serial.println("The UUID number is " + uuidStr);
  return uuidStr;
}
```

```
void create_new_room(String path){
  Serial.println(" Button state: HIGH");
    Serial.println("HIGH");
    FirebaseJson json;
     String roomID = randomID_generator();
       json.set(roomID+"/alias",roomID);
   if (Firebase.updateNode(fbdo,path,json)){
       Serial.print("A new room is added to:");
      Serial.println(fbdo.dataPath());
      randomID_generator();
      Config["roomID"] = roomID;
      saveConfig();
     }
}
```

Chapter 6
Software Test Document

6.1 System overview

Testing document will explain the items to be tested, and features to be tested and not tested. It will identify and explain the test approaches that will be used. And list the test deliverable that will be provided upon the completion of testing.

6.2 Test Items

This section of the test plan lists all the items of the Noorak that will be tested.

6.2.1 Features to be tested

- "Log In" Activity
- “Registration” activity
- “Forget Password” Activity
- “Edit room name” Activity
- "Delete room " Activity
- "Logout button" Activity
- "Set button" Activity
- “Change language” Activity

6.2.2 Features not to be tested

Not all features will be tested since only the main functionalities will matter in this test plan. Therefore the user interface test against the best practices of the Human Computer Interaction will not be tested.

6.3 Black box testing

In this section black box testing techniques will be used.

1. Functional: (a) Black box testing:

- Static: which includes high level and low level requirement reviews.

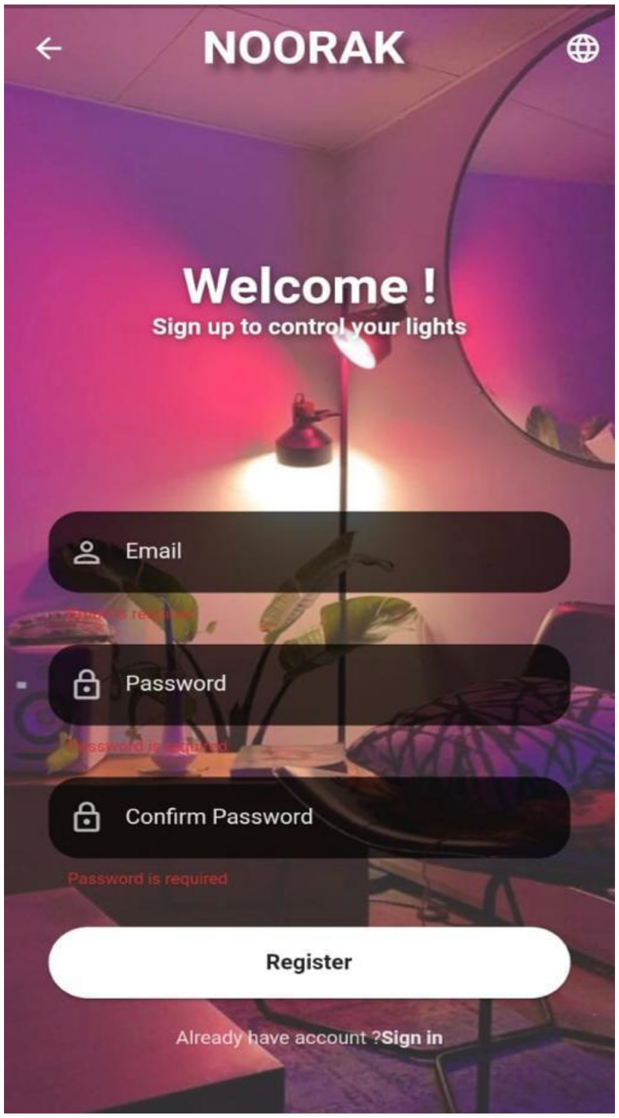
- Dynamic: which includes test to pass, equivalence partition and data testing (boundary value analysis, null testing and bad data testing)

2. Non-functional:

- Usability testing.
- Performance testing.

Functional Testing:

6.3.1 The Registration interface:

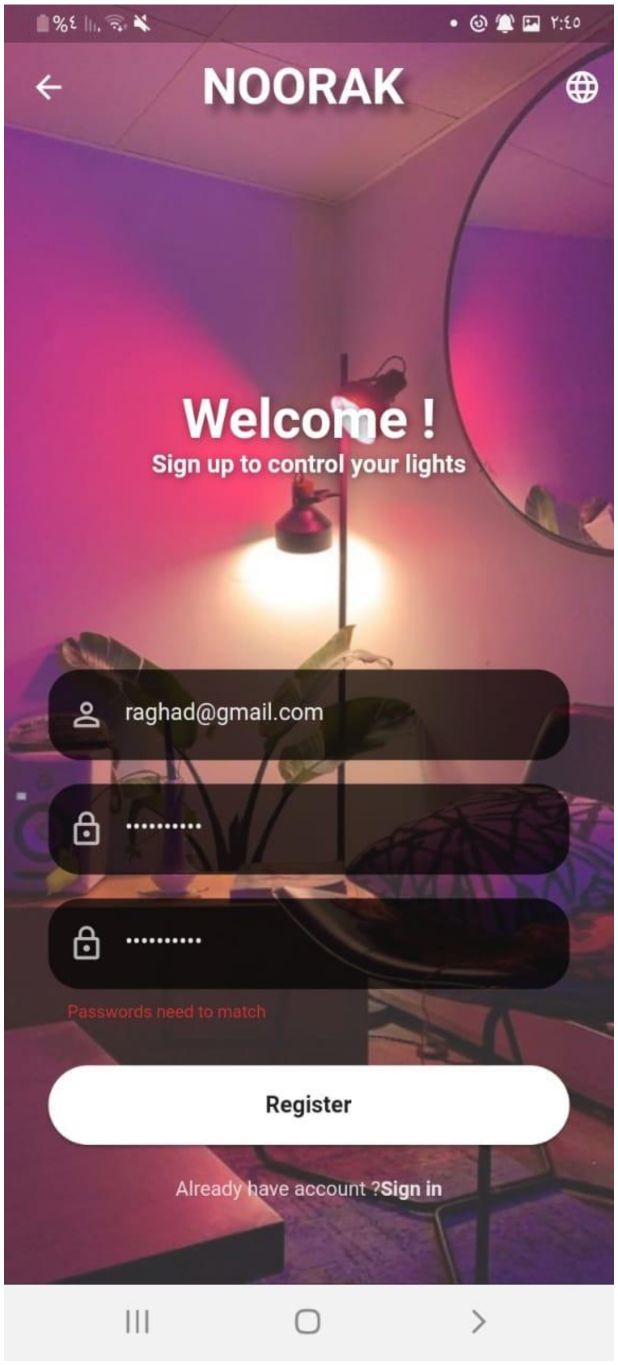


[illegible] is required

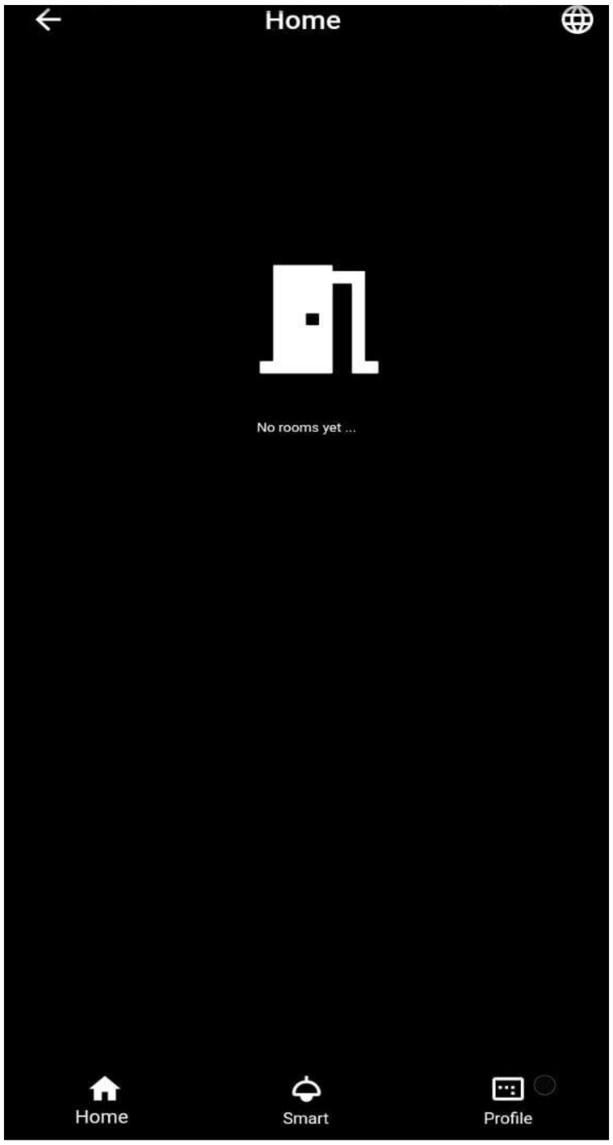
Password is required

Password is required

Here we can see if you pressed register and you don’t enter any data the warning

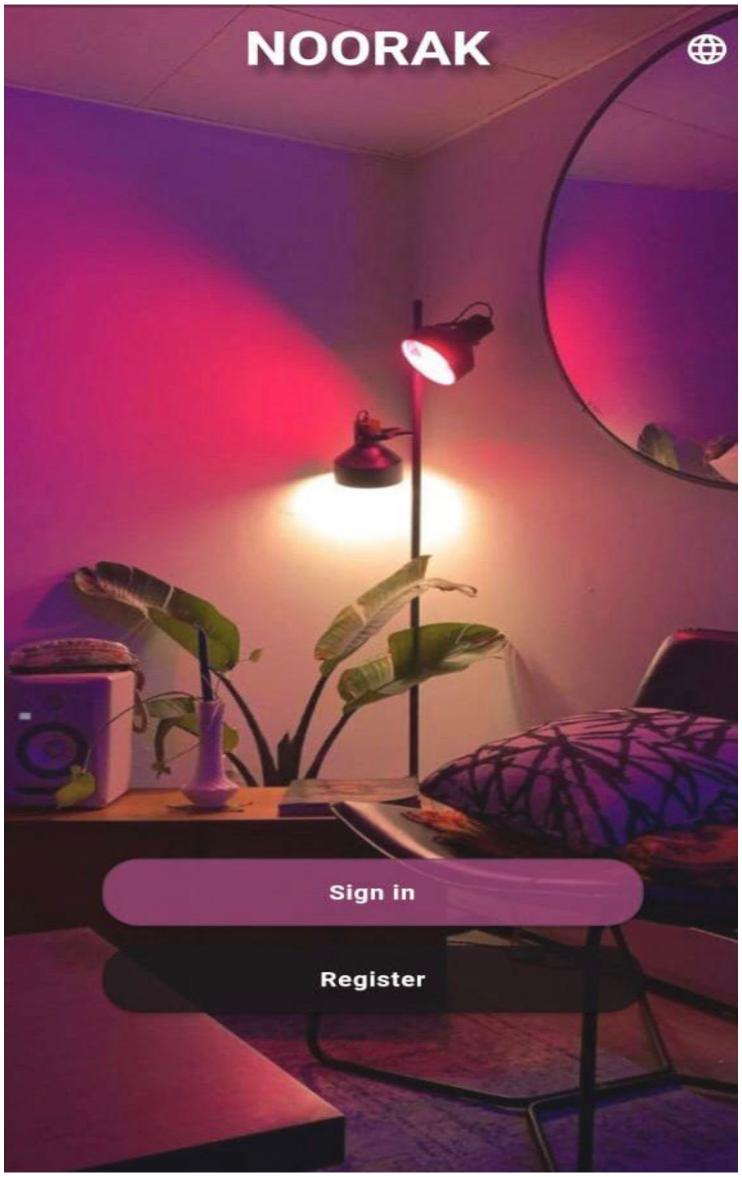


We can see the warning if you enter not match password

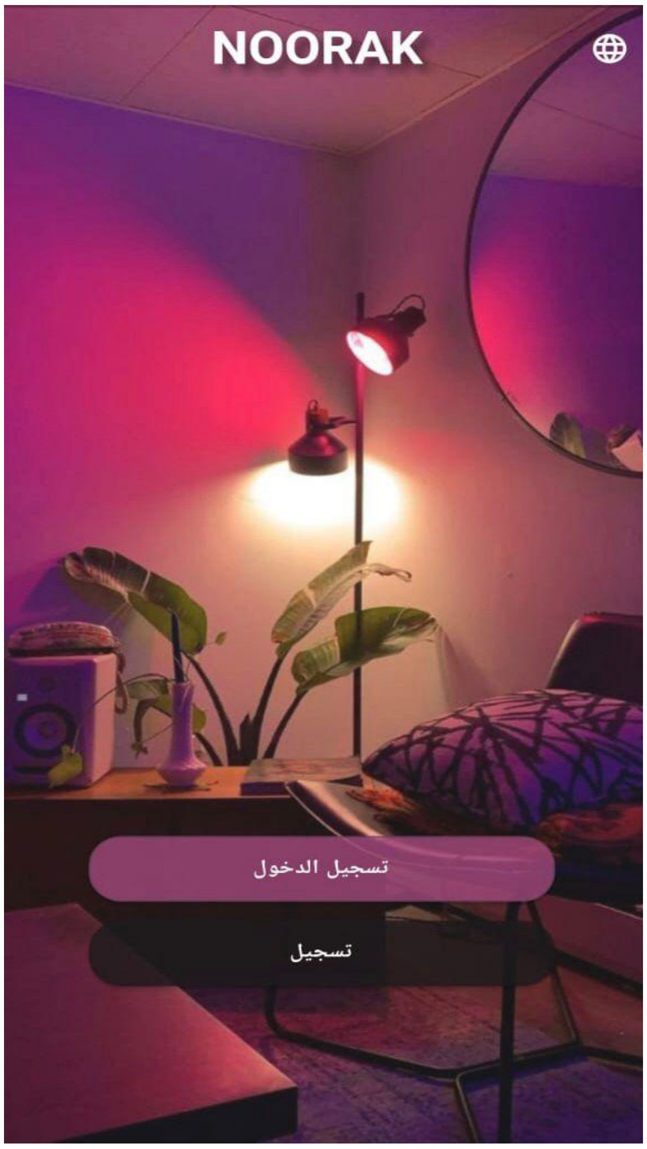


After register and login this is the first page

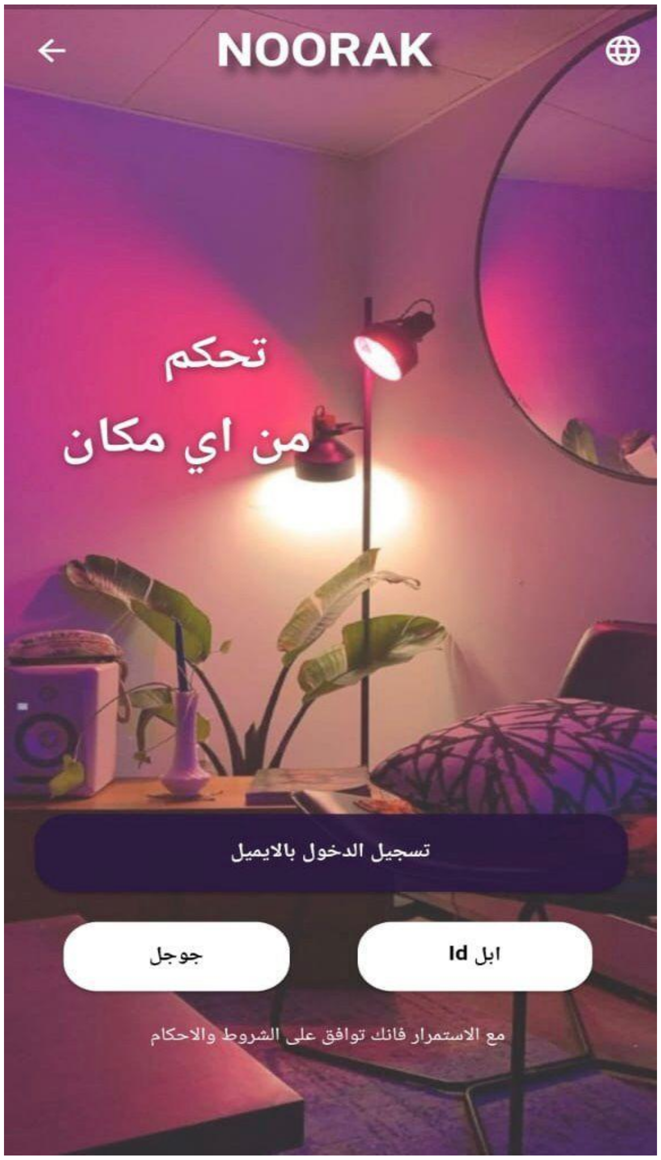
6.3.2 Translation:



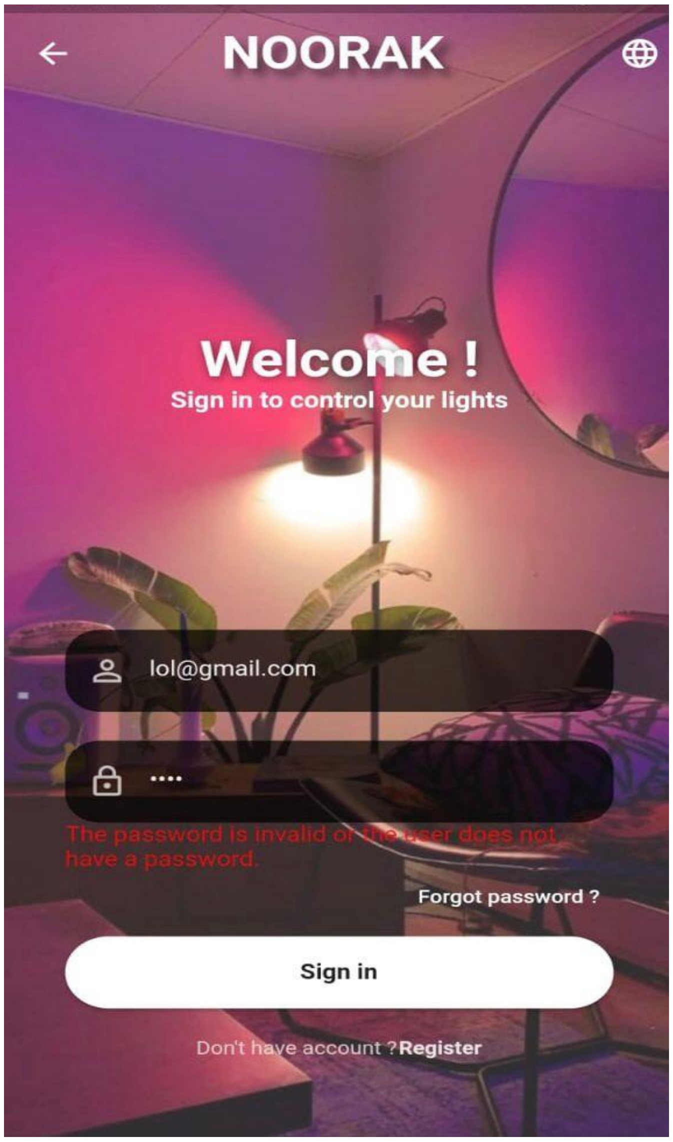
Page in english



Page in arabic

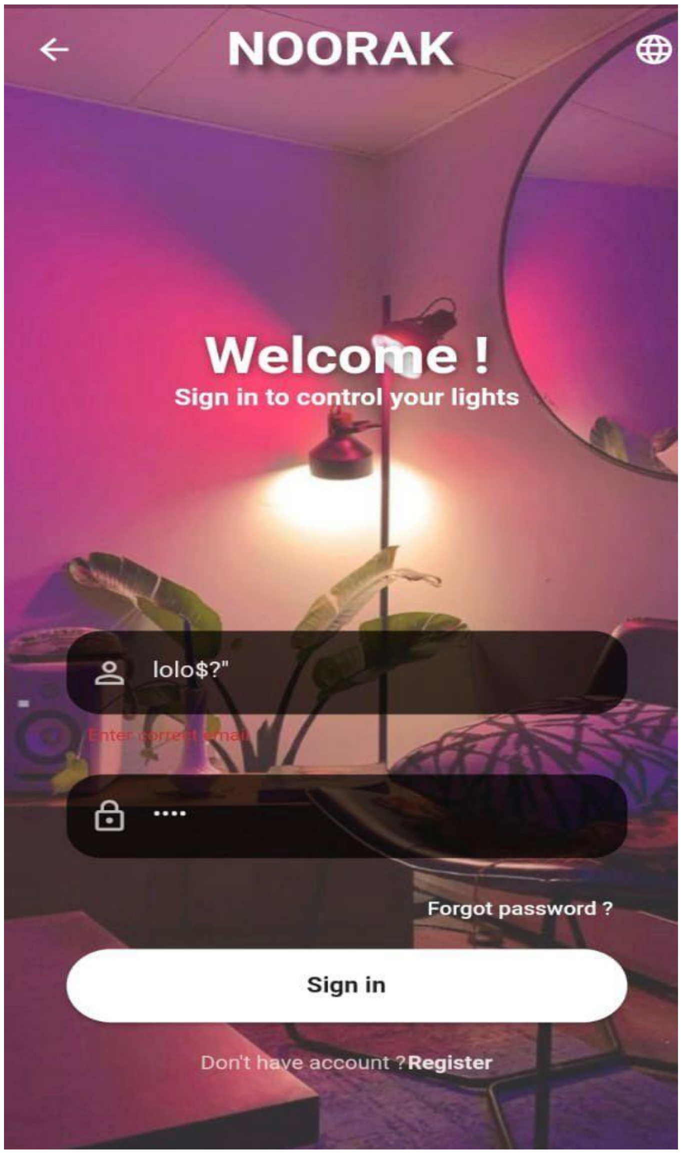


Page in Arabic Login testing :



The password is invalid or the user does not have a password.

Page in English Login testing :
If you enter invalid password it will give you this warning



Welcome !

Sign in to control your lights

lolo$?"

Enter correct email

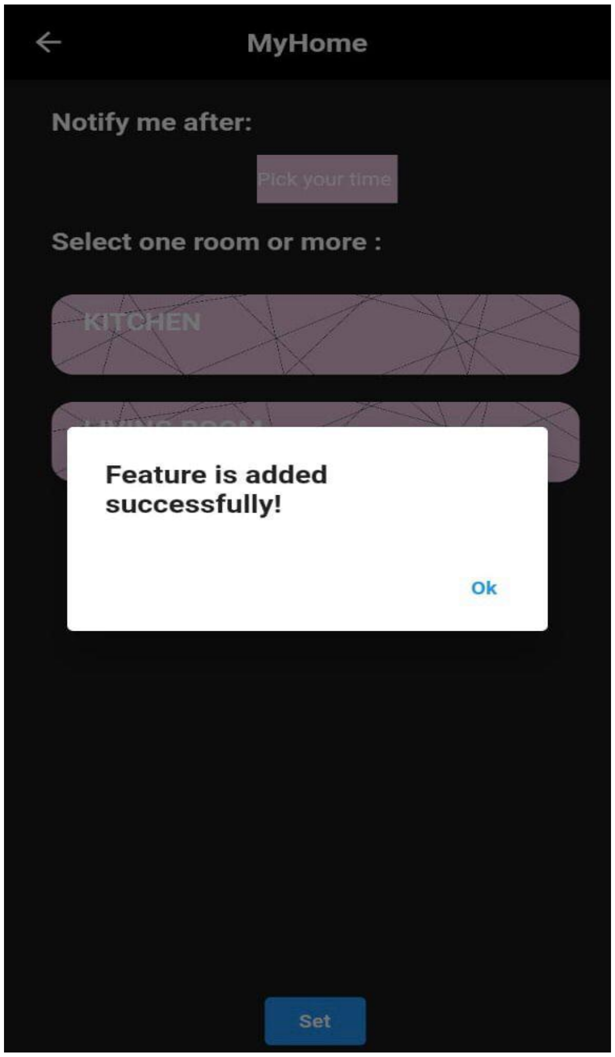
••••

Don't have account ?Register

Page in English Login testing :

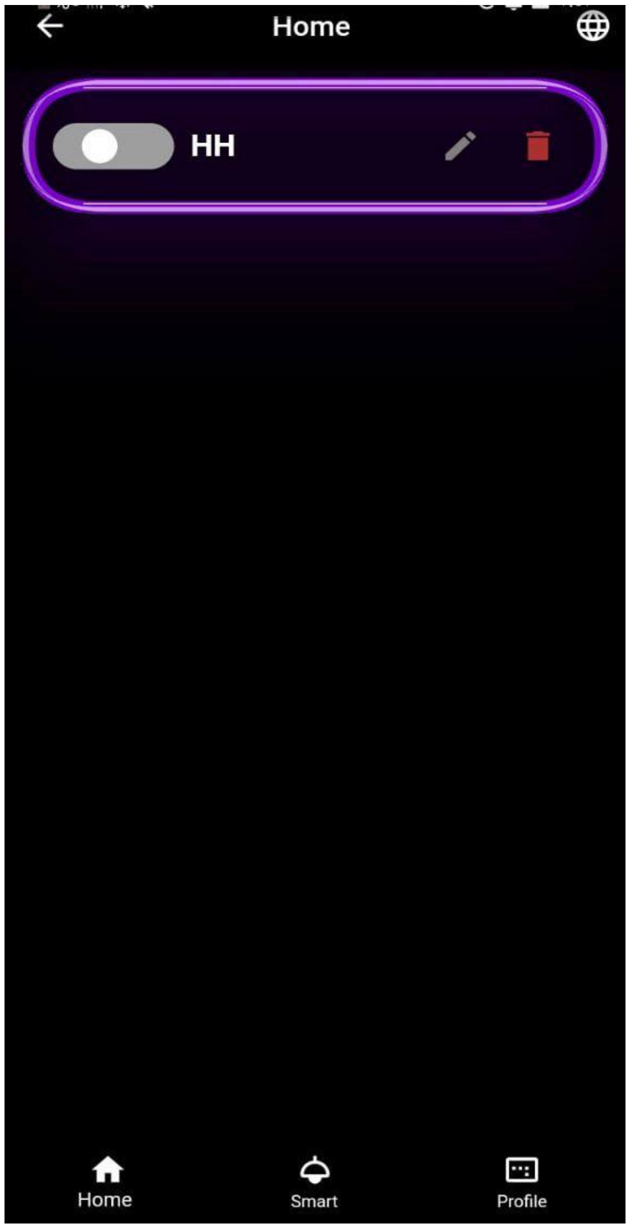
If you give it invalid email will give this warning

6.3.3 Set Button:

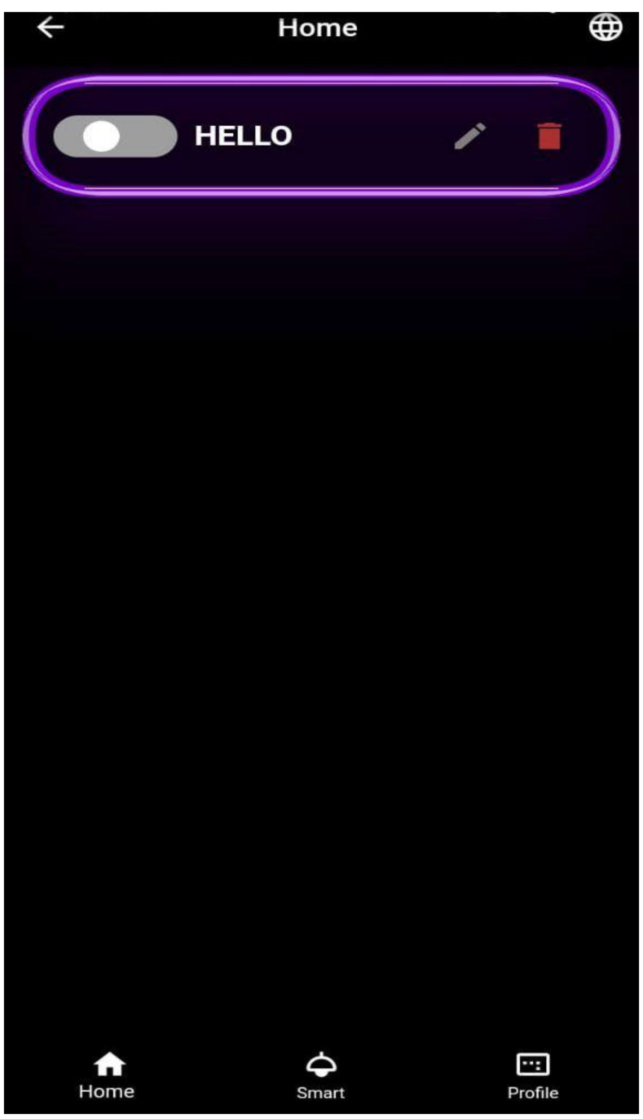

Feature is added successfully!

Set button:

6.3.4 Edit room name:

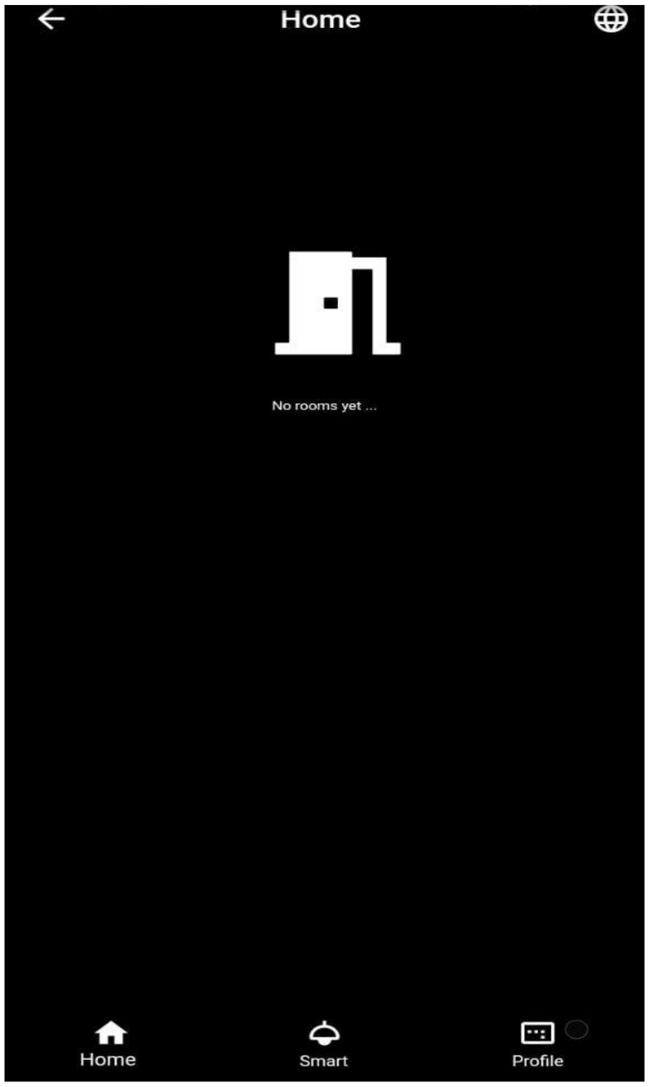


Before edit



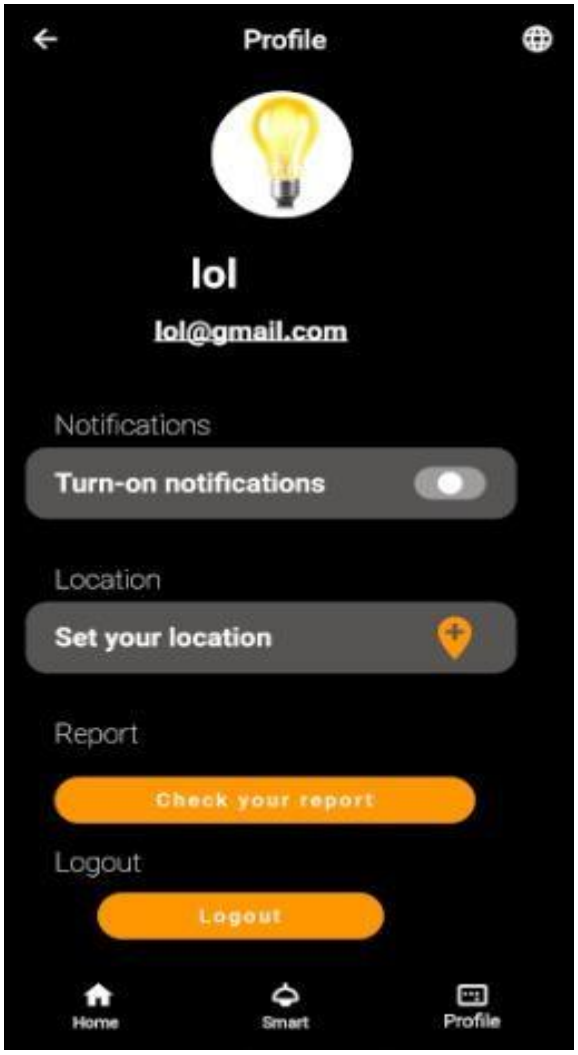
After edit

6.3.5 Delete room

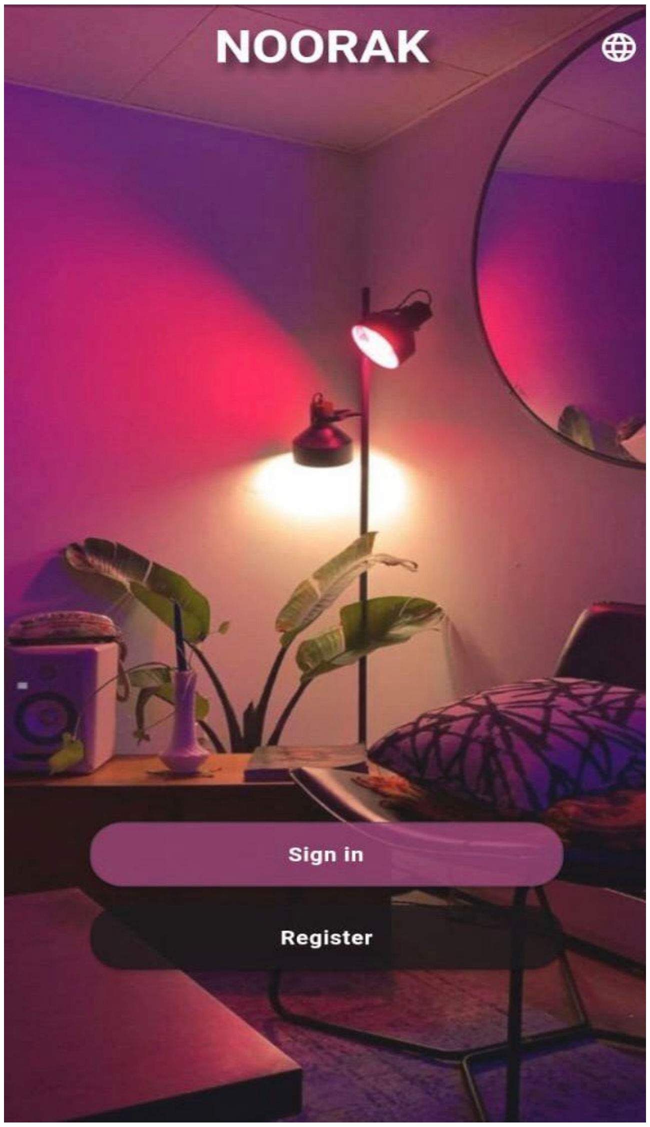


After Delete Room

6.3.6 logout testing:

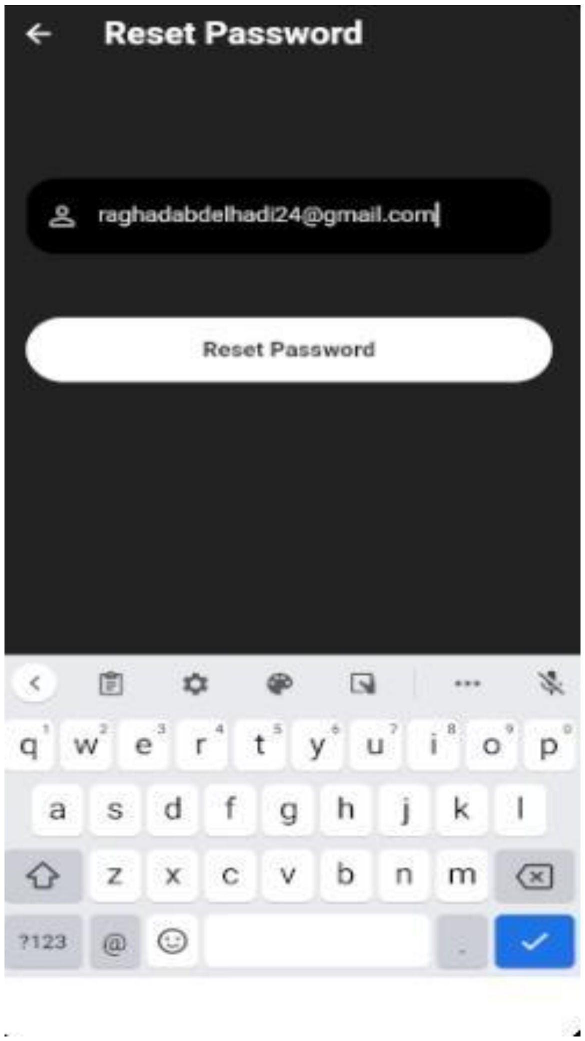


before logout

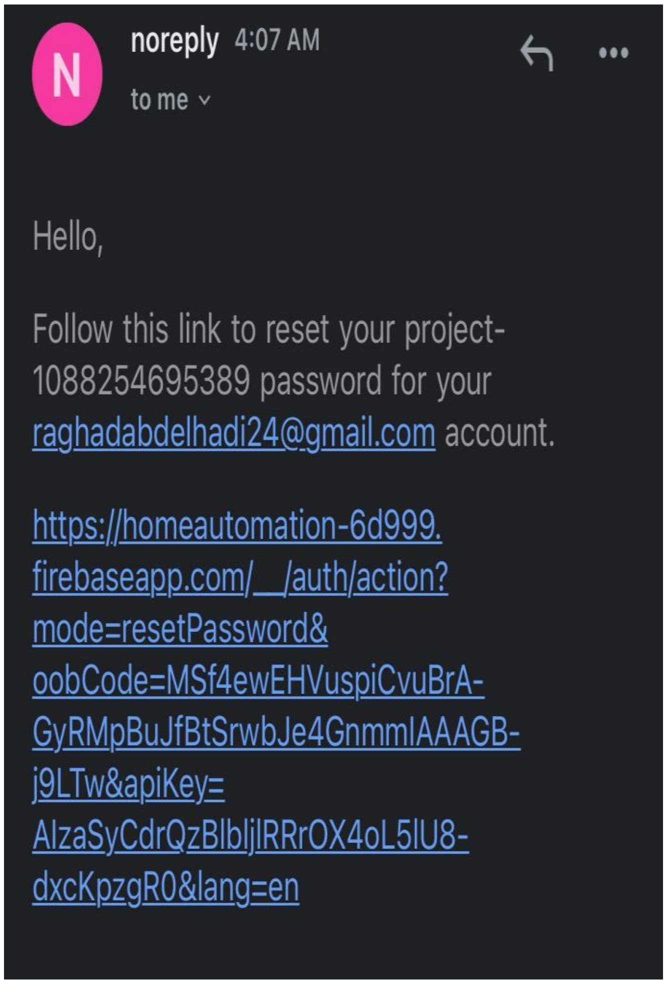


After logout

6.3.7 Forget password



write the email to reset password


noreply 4:07 AM

to me

Hello,

Follow this link to reset your project-1088254695389 password for your raghadabdelhadi24@gmail.com account.

https://homeautomation-6d999.firebaseapp.com/__/auth/action?mode=resetPassword&oobCode=MSf4ewEHVuspiCvuBrA-GyRMpBuJfBtSrwbJe4GnmmIAAAGB-j9LTw&apiKey=AIzaSyCdrQzBlbljlRRrOX4oL5lU8-dxcKpzgR0&lang=en

The link send to the user email to reset password

Non-functional testing:

Usability testing

For usability, it was decided to use the heuristic approach, which works by specifying a set of rules related to usability, and then rating each action/screen based on how they match the specified rules, that being said, here are the rules that were selected:

1. Strive for consistency. (by Shneiderman): which simply means that repeating actions should use the same steps, and repeating commands, prompts, menus and error messages should use the same terminology.

2. Offer informative feedback. (by Shneiderman): which means that for each action there should be some form of feedback, the intensity of the feedback should match the importance and recurrence of the action.

3. Error prevention. (by Nielsen): which means that the system should prevent the user from making errors.

4. Simple error handling. (by Shneiderman): which means, when a user makes an error, the system provides the user with a simple way of fixing it.

5.Permit easy reversal of actions. (by Shneiderman): this allows the user to experiment with unfamiliar options in the system, because they know that they can undo options if the result didn't satisfy them.

6. Support internal locus of control. (by Shneiderman): which means that the user should always feel like they're in control, and that the system responds to their actions, meaning that the user should be the initiator of the action.

Chapter 7
Future Work

As a future work, we are aiming to complete implementing some requirements that we could not implement due to the limited time we have to complete the whole project. Here are some new feature and improvements to the current features

7.1 Improvements to the current features

7.1.1 Signup and login using Google account and Apple ID

The current application gives the user the ability to register a new account by entering his email only. What we aim to achieve is, the user can register a new account using his google email and Apple ID and verifying that the entered email is an existing email. This could help in reducing the number of emails that are registered by mistake.

7.1.2 Scheduling Notifications

Scheduled notifications are currently based on the light status and the switch value of this option in the profile screen. What we need to achieve is, to get an infrared sensor that reads if any human being does not exist in a specific room; so the scheduled notification is based on the existence of people, light status and the switch button in profile screen.

7.2 New Features

7.2.1 Track the user's location

As provided in the mobile app, the user could have set his place location in the profile screen. The user's location must be tracked, by taking his permission, to measure the distance between his current location and the set location in the profile screen. If they are far away from the set location for 1 KM or less, for instance, some rooms that are chosen by him will be switched on.

7.2.2 Report energy consumption

Using a current sensor, this new feature will be provided. Since one of the main reasons to implement this application was reducing the energy consumption, we thought that this feature must be done as soon as possible. This feature can show the user a report consisting of the energy consumption of each room in the place and a graph of them. It can also show the expected electrical bill in JDs based on the electricity tariff.

7.2.3 System Security

Since this application contains some information about a user such as his location, security is very important for the apps that control some physical parts.

7.2.4 iOS Version

We aim to gain more user engagement to our application by providing a version for the iOS users. They must get the same fabulous, unique experience that Android users just do!