



ORTA DOĞU TEKNİK ÜNİVERSİTESİ
KUZEY KIBRIS KAMPUSU

Smart Home

Project Report

CNG 492 – Senior Project

Team Members: Temirlan Bayeshov 2041226
Anıl Peker 2016129

Table of Content

Table of Figures	5
1. Problem definition	6
1.1. What is Smart Home?	6
1.2. Why We Choose this project?.....	6
1.3. How we will succeed?	6
1.4. Project Objectives	6
1.5. Project Scopes	7
2. Identification of constraints	7
3. Literature Research	8
4. Requirements	9
4.1. Functional Requirements	9
4.2. Non-functional Requirements	9
4.3. System Specifications	10
4.4. Use Case Diagram.....	11
4.5. Context Diagram and Architecture Diagrams.....	12
4.6. Process Model.....	15
4.7. Sequence Diagram	16
4.8. Graphical User Interface	17
4.9. Test Requirements	18
4.10. Test cases	20
4.11. Test Plan.....	20
4.11.1. Introduction.....	20
4.11.2. Testing Strategy	21
4.11.3. Test Items (Functions)	21
4.11.3.1. Unit Testing	21
4.11.3.2. System and Integration Testing.....	21
4.11.3.3. Smoke Testing.....	22
4.11.3.4. Performance and Stress Testing	22
4.11.3.5. Alpha/Beta Testing	22
4.11.3.6. Security Testing	23
4.11.3.7. Validation Testing.....	23
4.11.4. Summary of Features to Be Tested	24

4.11.5. Summary of Features Not to Be Tested	24
4.11.6. Item Pass/Fail Criteria.....	25
4.11.7 Test Deliverables.....	25
4.11.8 Test Schedule	26
5. Design details.....	27
5.1. The architecture models- for all levels.....	27
5.2. Entity Relation Diagram Model.....	27
5.3. Data Flow Diagram.....	27
5.4. Sequence Diagram	28
5.5. List of Classes, Definitions and Hierarchy	28
5.5.1. Arduino Side (Nodemcu)	28
5.5.2. Android Side	29
5.6. List of Interfaces Definitions	29
6. Implementation Details	30
6.1. Project Management	30
6.1.1. Project Estimation	30
6.1.2. Project Tasks and Durations Assumptions.....	32
6.1.3. Milestones	33
6.1.4. Activity Diagram.....	34
6.1.5. Gantt Chart.....	35
6.1.6. Task allocation chart	35
6.2. Testing section	36
6.2.1 Unit Tests	36
6.2.2. System and Integration Testing.....	39
6.2.3. Alpha/Beta tests	41
6.3.4. Test Requirements.....	41
6.3.4.1. Summary of the Tests Passed.....	41
6.3.4.1.1. Unit tests for R calculating functions.....	41
6.3.4.1.2. Unit tests for Android Credential Control with Non-User Person	42
6.3.4.1.3. System and Integration Tests for connection of R and Database.....	42
6.3.4.2. Summary of the Tests Failed	42
7. User Manual.....	43
7.1. System Overview	43

7.2. Organization of the Manual	43
7.3. SYSTEM SUMMARY	44
7.3.1. System Configuration	44
7.3.2. User Access Levels	44
7.3.3. Contingencies.....	44
7.4. GETTING STARTED	45
7.4.1. Installation and Logging	45
7.4.2. System Menu	46
7.4.2.1 Sample Fan Dashboard	47
7.4.2.2. Sample Door Dashboard	48
7.4.2.3. Sample Light Dashboard.....	49
7.4.3.4. Sample Valve Dashboard.....	50
7.4.3.5. Sample Pump Dashboard.....	51
7.4.4. Changing User ID and Password	52
7.5. REPORTING	52
7.5.1. Bar plots	52
7.5.2. Sensors statuses.....	52
8. Installation Manual and Deployment Details.....	53
8.1. Introduction.....	53
8.2. Revision history	53
8.3. Intended audience and reading suggestions.....	53
8.4. Server Configurations	53
8.4.1. Server 1 (Database).....	53
8.4.2. Server 2 (Mobile Application)	55
8.4.2.1 Packages.....	55
8.5. Software Installation	56
8.5.1. Server 1 (Database).....	56
8.5.2. R studio	56
8.6. Testing the Installation.....	57
9. Possible extension details	58
10. References.....	59

Table of Figures

Figure 1:Use Case Diagram.....	11
Figure 2:Context Diagram	12
Figure 3: Architecture Diagram Level 1	13
Figure 4: Architecture Diagram Level 2	13
Figure 5: Architecture Diagram Level 3	14
Figure 6:Process Diagram.....	15
Figure 7: Sequence Diagram.....	16
Figure 8: GUI of Smart Home Android APP.....	17
Figure 9: Smoke Testing Flowchart [8]	22
Figure 10: Token Generator Algorithm[10].....	23
Figure 11: V-Model [11].....	24
Figure 12: Test Schedule Gantt Chart.....	26
Figure 13: ERD Model.....	27
Figure 14:Data Flow Diagram Lever 0	27
Figure 15: Data Flow Diagram Level 1	28
Figure 16: Activity Diagram	34
Figure 17: Gantt Chart	35
Figure 18: Task Allocation Chart.....	35
Figure 19: Login Screen.....	45
Figure 20: Main Dashboard	46
Figure 21:Fan Dashboard.....	47
Figure 22: Door Dashboard	48
Figure 23: Light Dashboard	49
Figure 24: Valve Dashboard	50
Figure 25: Pump Dashboard	51
Figure 26: Wamp Server Configuration Page[12]	54
Figure 27: Volley Framework Benefits[13].....	55

1. Problem definition

Our graduation project is about Smart Home. We will divide our general work into several parts. Now, we will explain why we choose this project and importance of our product, what is the smart home system in general and how we will improve it.

1.1. What is Smart Home?

Smart Home is a system which provide not only more comfortable life standards for owners but also, they can feel safe by using these kinds of automated systems. Owners will be able to control all sensors which will inserted to all places in house by using their smartphones, tablets, etc. They will also be able to reach different data information. Shortly this kind of automated systems will provide better and safer life.

1.2. Why We Choose this project?

Firstly, we would like to explain that nowadays, technology is playing irreplaceable part in our lives and from day to day is improving very fast. Because of that, we though that smart home can be one of the main factors to prevent catastrophe like fire, gas leak etc. Despite of this, using these kinds of systems can make usage of electricity, water resources and this kind of important things which can affect people's budget in a proper way. Also, by making this project we will understand and improve our skills in many spheres such as computing, coding, decisions, IoT, teamwork etc.

1.3. How we will succeed?

Our first step will be to create a web service for a android applications. Data and statistics from a sensor will be stored in database system which will be shown in users' application.

After a web service by using Arduino and different kind of sensor we will create an example model of home to check workability and usage of it. We will divide our project to several systems like security management, temperature control, flood and gas leak control, curtain control and many other basic systems.

1.4. Project Objectives

Android-controlled Smart Home Automation should be able to control home appliances wirelessly effectively and efficiently.

Controlling Home via Application

An android application should be developed for that which includes with switches. After that application can be used by users with very simple interactions.

Device and WIFI Connection

Arduino based device should connect the ethernet or WIFI for data transmission. For that situation "Nodemcu" card can be useful for simple WIFI connection and also, "Nodemcu" card can be used for little server.

Disaster Controlling

In unexpected situation like flooding, gas leak, fire and robbery, Arduino should give reaction. Some sort of sensors and Arduino code interaction can be solving this problem.

Data Transition

System should keep some sort of data to show to the user like temperature, humidity, etc. For that situation MySQL Database can be built and after that, user can get the data that comes from Arduino sensors with interaction between database and services.

1.5. Project Scopes

The scopes of that project will be:

- ➔ To increase the life standards of people.
- ➔ To make people feel more secure in their homes.
- ➔ To ensure that people take control of their home when people are not in their homes.
- ➔ To ensure that their homes automatically protect themselves
- ➔ To provide access to historical data about their homes
- ➔ To provide immediate news about unexpected situations in their homes

2. Identification of constraints

During our project we realized serious constraints that restrained some of our abilities to make a real business world class project. As a students and lack of some skills our team faced many little but solvable challenges and problems. We learned many new things and improved our skills that we gained during university background.

One of the main constraints was economical constraint, we believe that with much better and expensive modules, sensors and tools we can improve our Smart Home system in better way. With non-free software quality of work and response times could be better in many specific ways. For example, we used free version of RStudio (Desktop version), but instead RStudio Server Pro +RStudio Connect which is non-free software can improve our statistical performance. Also, while modeling our Smart Home project we also can user more better models with different staff, instead using a box.

Second constraint we faced was environment. We finished our project in lab with a with a good workstation which were provided to us, but we didn't find any tools that can help us to finish a project in a proper way. For example, lack of tools such as digital multimeter, oscilloscopes and another important tools. During our project, in the last month we face with water problem in our lab, which forced us to stop working on some part of the project for 2 weeks, because we didn't reach our workstation.

Third constraint we faced were Health and Safety, because we worked with electrical staff some unexpected situation occurred during this year. The lack of medical center in our university, made a pressure to our team.

Another constraint we faced, was ethical constraint. Sometimes we forced to be one sided, so we could only present our project with a bright side of automated systems, we cannot tell the negative sides of our project. We should make sure that our customer can't see hidden sides of our project.

Last constraint that we found out, is a manufacturability. The biggest problem here was a lack of special tools and software as we described in economical section. If university provided to us more items, we could improve manufacturability and extend our project in many new ways, with more new functions and improvements. With low cost products we faced some problems during the project. But at the end of the day we finished it in proper way.

3. Literature Research

For our project, there are lots of resources that we can use and they are giving some good ideas for the future works. Also, in the industry, there are lots of different type of smart home systems. For example, some projects have provided Bluetooth connection to control the smart home. Karand [1] explained that Bluetooth based home automation systems are not useful nowadays because of explosion Internet.

Wahab [2] designed a prototype electrical appliance control system using the web. They also set the server to restart automatically if the server status was not available.

For the reverse fan part [3], typically, fire comes from a chemical reaction between oxygen in the atmosphere and some sort of fuel (wood or gasoline, for example). From here we can understand that if we can cut the reaction, we can stop the fire. Because there are too many options to choose for a one member of this reaction, we can choose oxygen to stop fire.

For the air quality part [4,5], there are five important element which affect quality of air. These are Sulfur dioxide, Nitrogen dioxide, Carbon Monoxide, Trioxide, Carbon dioxide. Also, mold, some bacteria types, humidity, flammable gasses, smoke etc. can be effective to decrease quality of air. If air in home is not good enough, it can cause immediate health problems including aggravated cardiovascular and respiratory illness, added stress to heart and lungs, which must work harder to supply the body with oxygen, damaged cells in the respiratory system. For long term, it can cause accelerated aging of the lungs, loss of lung capacity and decreased lung function, development of diseases such as asthma, bronchitis, emphysema, and possibly cancer and shortened life span.

Chan[6] claimed that a selection of leading smart home projects as well as placeable tracking systems and co-robotics technologies. When we look from psychological side, he mentioned that modern sensor embedded homes or smart homes not only help to reduce physical functions, but also help solve the social isolation people face who has living in smart home.

Khorov[7] showed that intelligent technologies like smart home play a key role in sustainable economic growth. Also, he aimed to explain and develop the organization between smart grids, smart meters, smart houses, smart healthcare systems and smart industry using WIFI technologies.

4. Requirements

4.1. Functional Requirements

User Requirements Specifications:

1. System should provide a registration page for new users
2. Only authorized users should be allowed to use a house
3. System should provide login page for regular users
4. User account should login to system at least one time in 24-hours, otherwise application will automatically logout the user account from application. (Necessary for security)
5. System should give warning message in case of emergency situations (example: fire, flooding, gas leak and robbery)
6. Users allowed to reach data sets from sensors
 - Temperature/Humidity value
 - Soil Humidity value
 - Fire, Flood, Gas Leak, Robbery alert status
 - Lights Status
 - Valves Status
 - Water Pump Status
 - Main door, Garage door, window status etc.
7. Users allowed to turn on/off sensors
 - Servo motors (main door, garage, window)
 - Gas and Water Valves
 - Water Pump
 - Alarm Status
 - Lights
 - Fan

4.2. Non-functional Requirements

1. Database should have enough capacity for a data that are comes from sensors of system.
2. Server should have only Database, Arduino (Nodemcu) and Mobile Application connections.
3. Server should be able to receive many inquiries.
5. None of quires should be lost.
6. None of data should be lost.

7. Response should be no more than 5 second.
8. All functions should work in proper way.
9. All buttons in applications should be clickable.
10. All data should be understandable.
11. Android app should take the data regularly.
12. Session handler in android app should logout the user if user didn't login to the system with his own account in 24-hours.
13. Nodemcu should connect to the WIFI properly.
14. Arduino electric power should have enough ampere value for sensor working.
15. Nodemcu should take the recent commands from MySQL database and should transmit to Arduino

4.3. System Specifications

Sensors Specifications:

Sensors	
SN-01	Sensor should sense the temperature
SN-02	Sensor should sense the humidity
SN-03	Sensor should sense light
SN-04	Sensor should sense if there is gas leak
SN-05	Sensor should sense water level
SN-06	Sensor should send data to server

Table 1: Sensors specifications

Mobile Application Specifications:

Mobile Application	
MA-01	Mobile application should allow only authorized users to login
MA-02	Mobile application should provide functions to authorized users
MA-03	Mobile application should provide data sets to authorized users
MA-04	Mobile application should allow users to configure their sensors
MA-05	Mobile application should provide reports about sensors
MA-06	Mobile application should provide alert reports

Table 3: Mobile Application specification

Database Specifications:

DATABASE	
DB-01	System should be able to send data from sensors to database

DB-02	System should be able to send data from database to application
DB-03	Database should organize a table for all sensors
DB-04	Database should store all data sets of all sensors for each unit of time
DB-05	System should create statistical analysis
DB-06	System should allow only administrator to modify database
DB-07	System should allow authorized users to reach their data sets

Table 4: Database specifications

4.4. Use Case Diagram

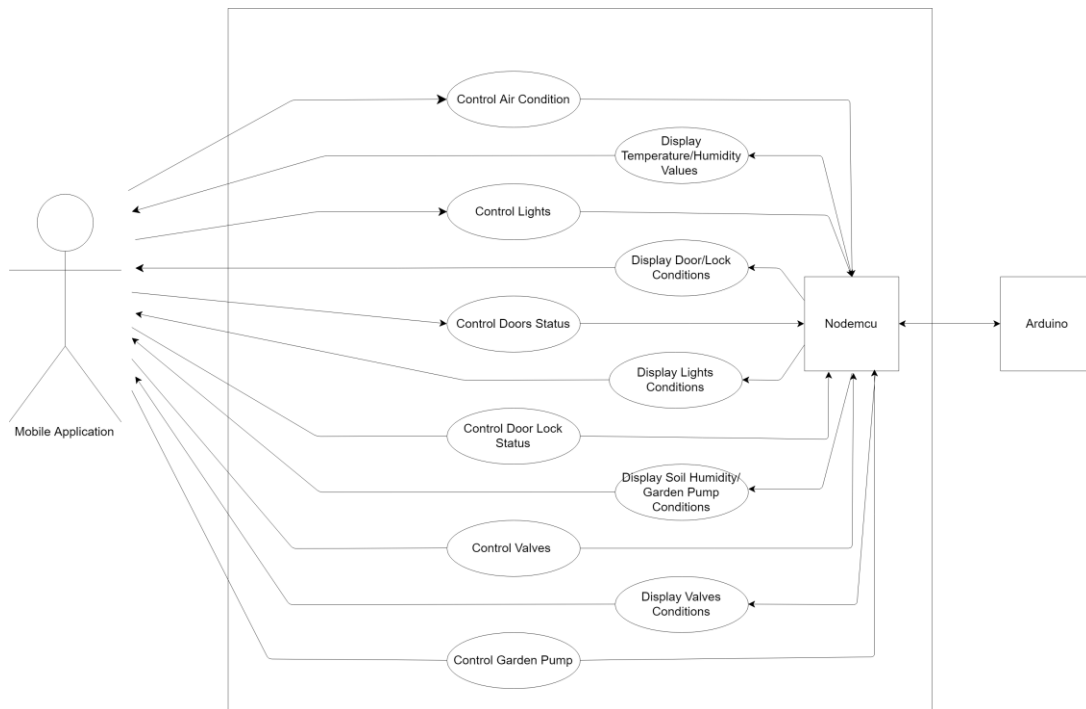


Figure 1: Use Case Diagram

This is the use case diagram of our project. As it seen from the figure 1, there is one user who can use the smart home. This user can control some properties of smart home and s/he can see some outputs of the system via interaction of server with applications, database and Arduino.

4.5. Context Diagram and Architecture Diagrams

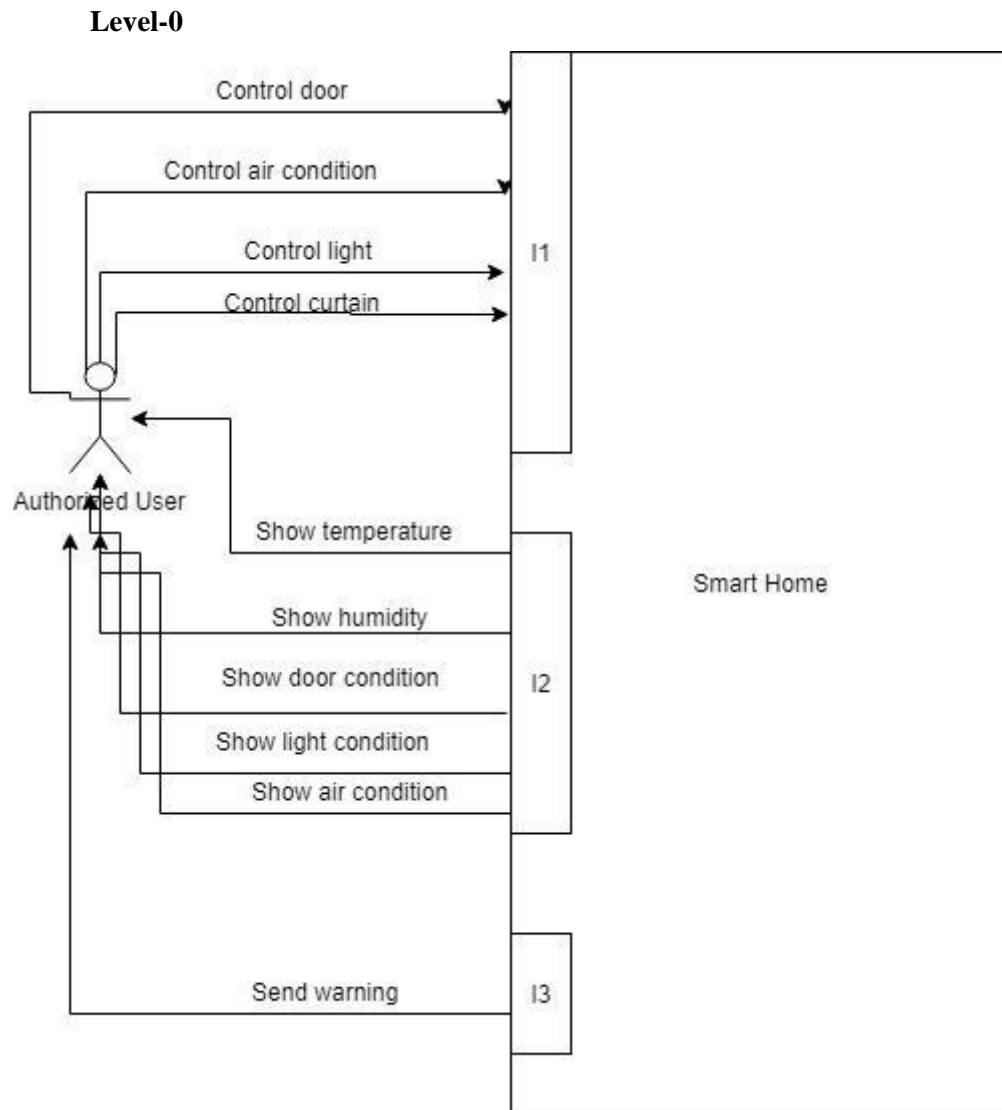


Figure 2: Context Diagram

Context diagram shows that interfaces which permit user to communicate with the system. First interface is for getting inputs from the user while others for giving outputs. Second interface is about giving information for the house and last interface will be used with giving message when house will have dangerous situation.

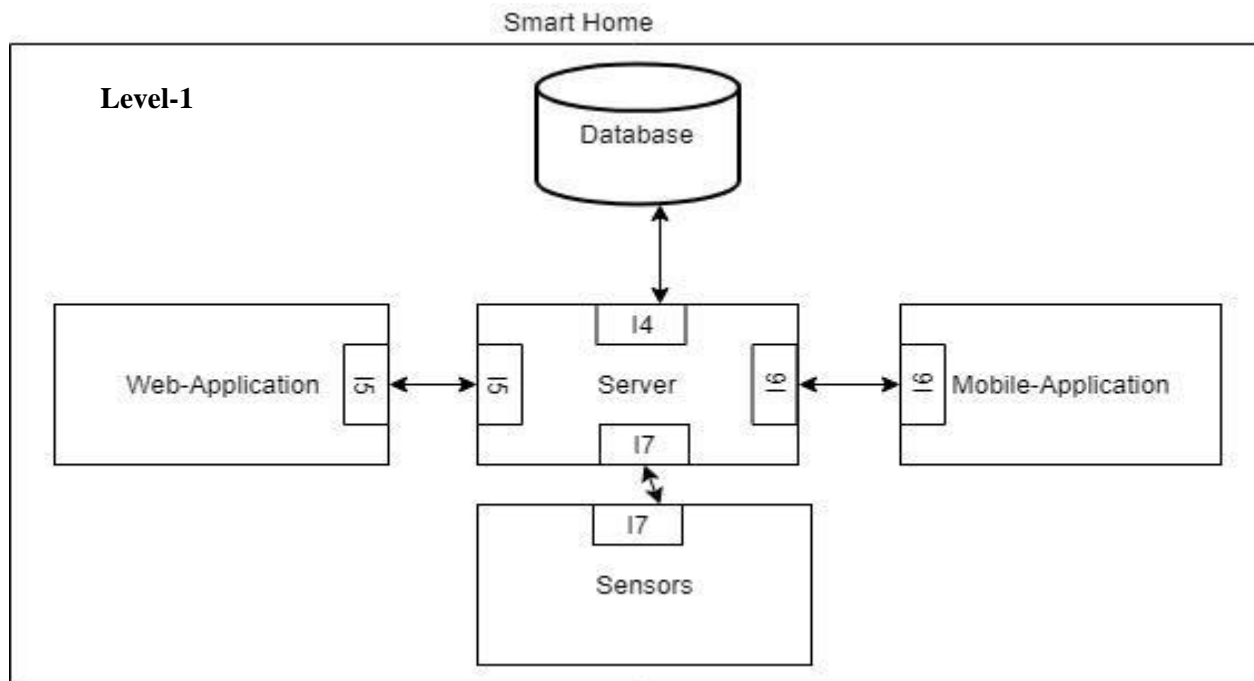


Figure 3: Architecture Diagram Level 1

This diagram determines how subsystems communicate with each other. There is one main server which provides interaction between applications, sensors and database. Shortly, all data should be stored in database and to store these data, system should have a server.

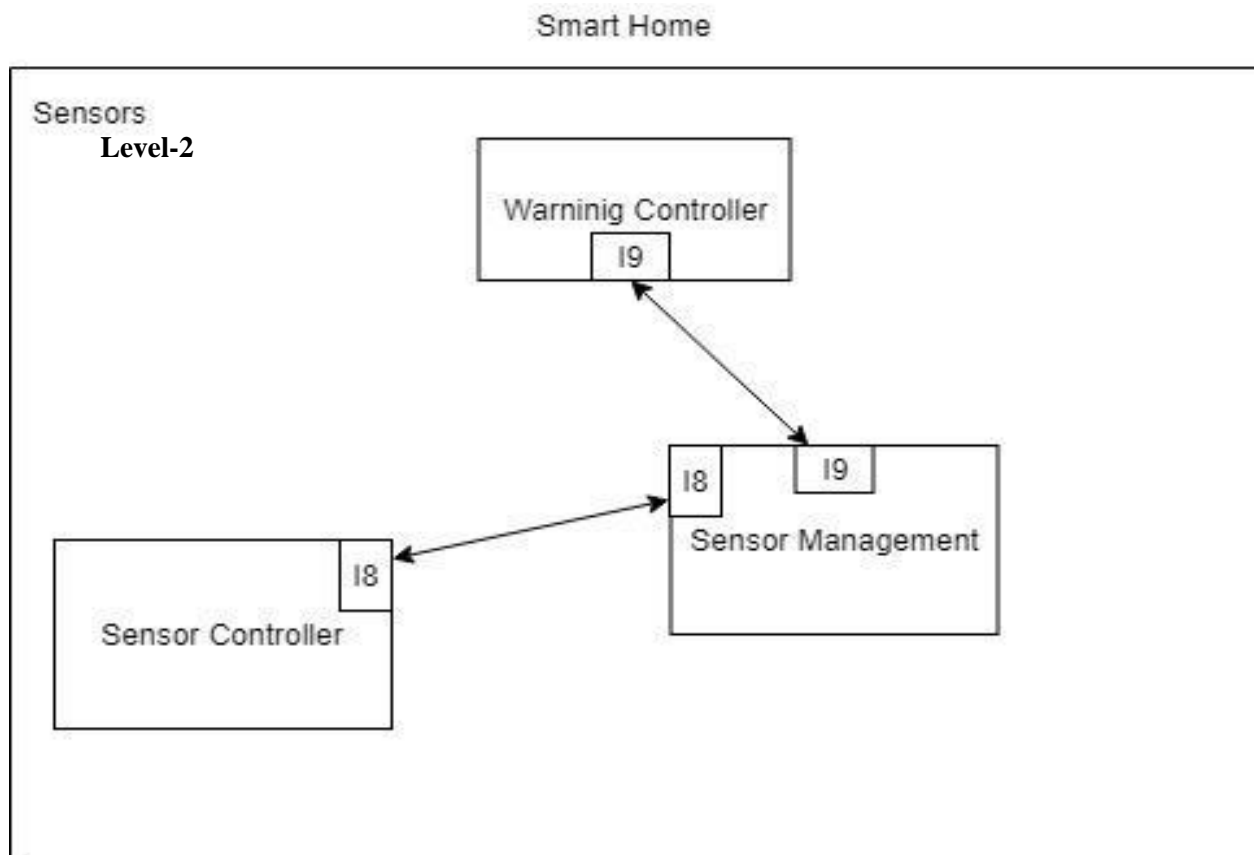


Figure 4: Architecture Diagram Level 2

Sensor subsystem has one sensor controller and sensor management interacts with it by using interface 8. When there is unexpected situation occurs, sensor management uses interface 9 to communicate warning controller.

Level-3

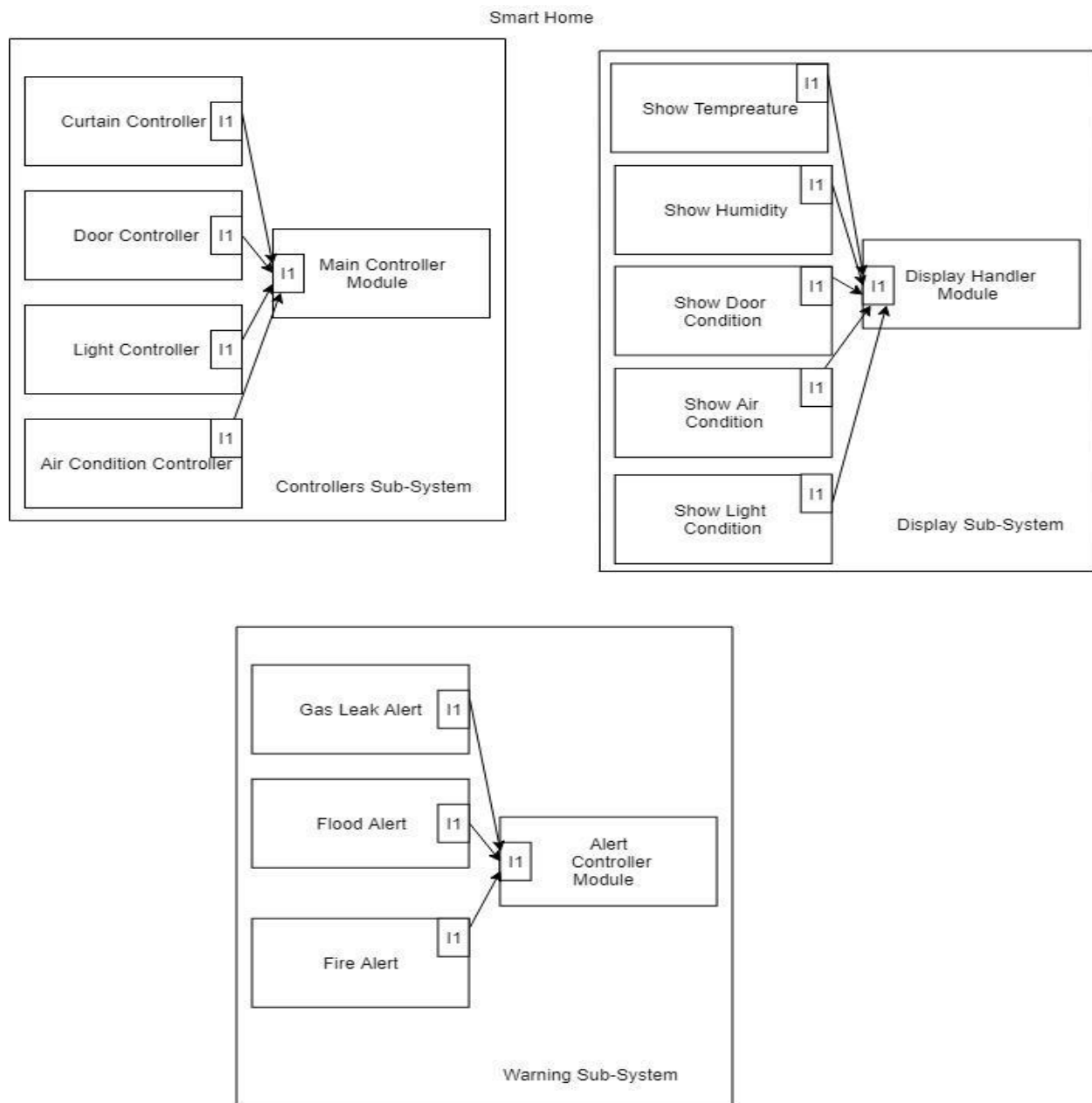


Figure 5: Architecture Diagram Level 3

It is seen that every subsystem has a controller module to have communication with other subsystems. As it is understandable from name of subsystems, controllers subsystem working for controlling inputs coming from user and display, display subsystem's job is showing outputs to user. If there is dangerous situation, warning subsystem's alert controller module shares information with the subsystem which needs this information, depends on the alert.

4.6. Process Model

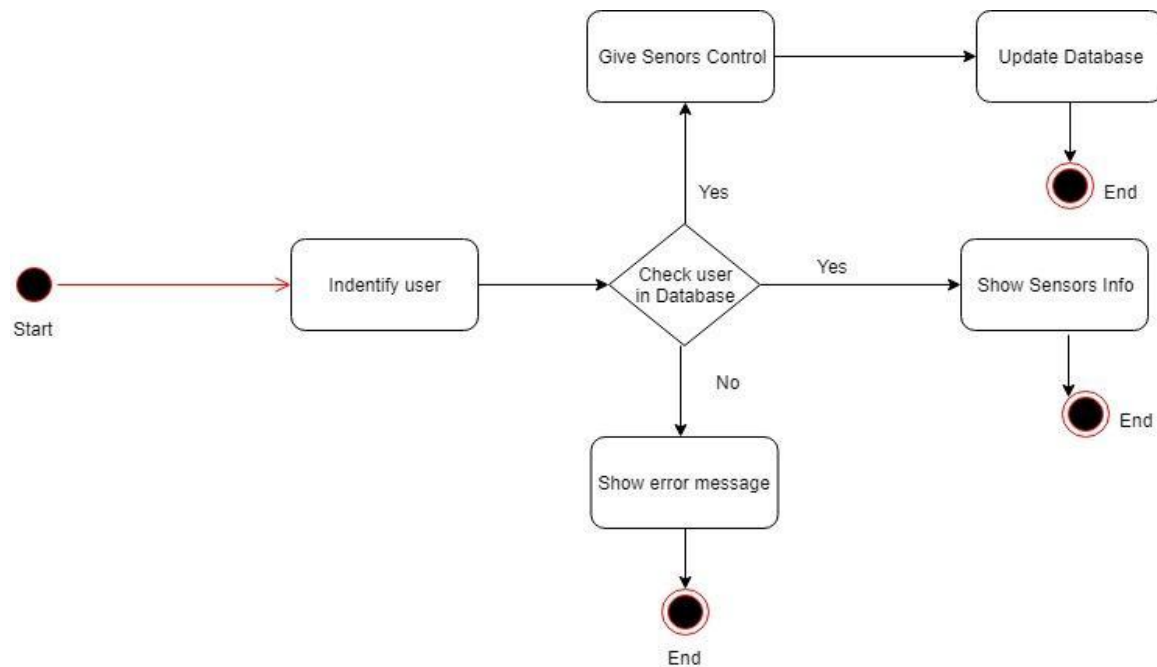


Figure 6:Process Diagram

Our project's process model is drawn above. If the user can reach database, s/he can update database with changing sensors' value or see sensors' information. If database is not reachable, user gets error message.

4.7. Sequence Diagram

Sequence diagram shows how our project works with the specification of simple runtime scenarios. User info is sending its own information to server via application and server is validating the information via database values. If it is true, user can enter to the application. After that, user can control some sensor data. And, user can see some other sensor data which is updated by sensors.

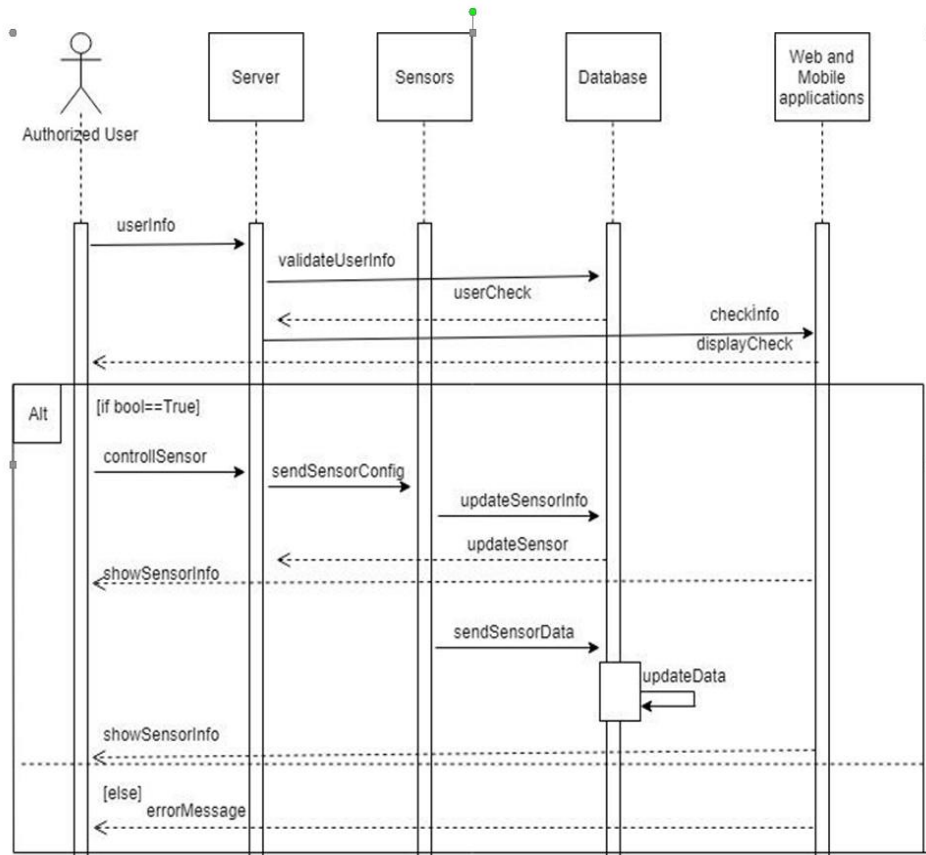


Figure 7: Sequence Diagram

4.8. Graphical User Interface

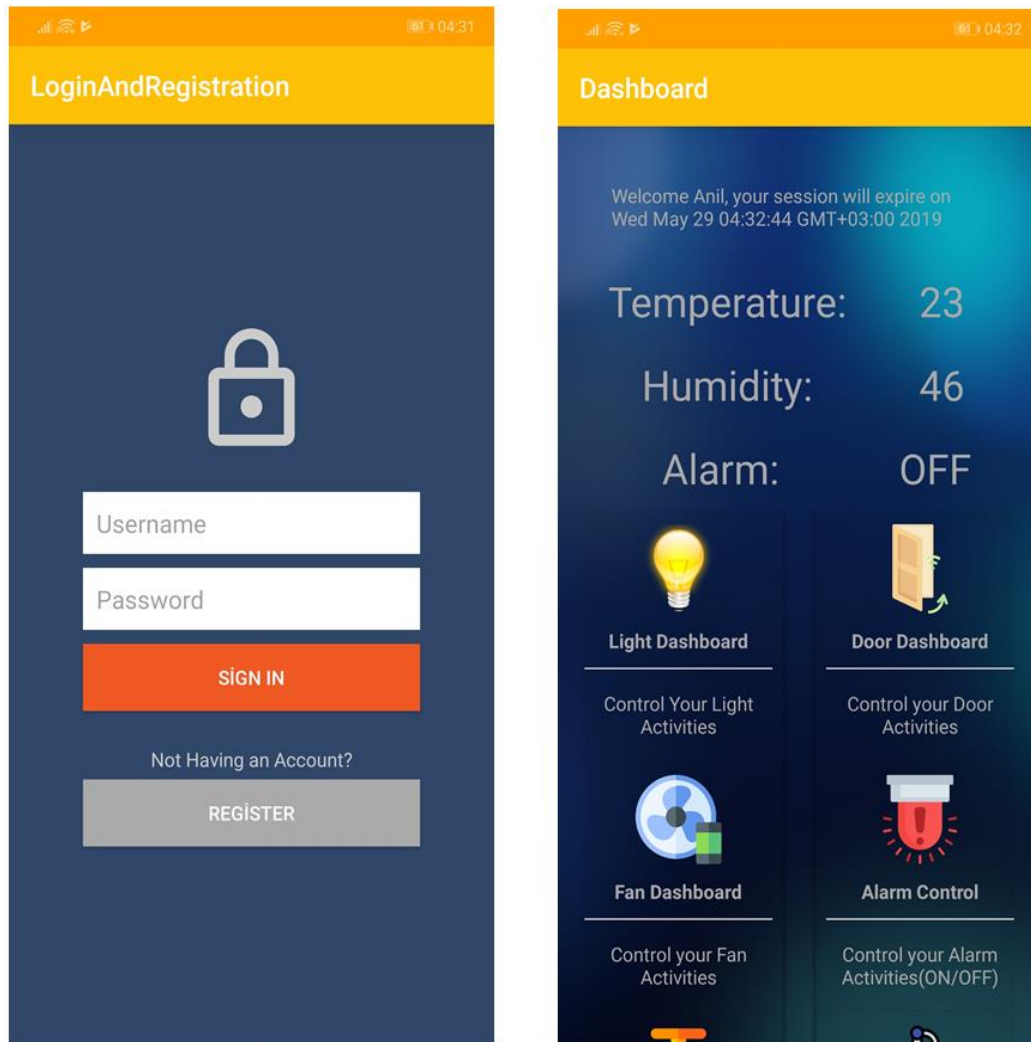


Figure 8: GUI of Smart Home Android APP

To control our smart home, we are using android controller app that we created. Application has login screen which is using for the entering to the system. In addition, login page contains register button which is sending you to registration page to register the application system. After user login, dashboard is appearing which is the main dashboard(menu) for our system. Main Dashboard contains some basic information about home like temperature, humidity values and alarm status. In addition, main dashboard is giving the path to go to deep journey in the application. Light dashboard contains light buttons which are using to turn on/off the lights of home. Door dashboard is giving control of the entrance of house. Fan Dashboard contains temperature, humidity values, fan control, and also, daily/monthly temperature and humidity charts. Alarm Control is a button which is using for alarm on/off. Valve dashboard is giving the full of control of home's valves and also, in this dashboard user can get information disasters statuses like gas leak, flooding and fire status. Pump dashboard contain soil-humidity value of the garden and manual control of water pump. In addition to main dashboard, which contains logout button.

4.9. Test Requirements

Use case	Requirement
Control Air Condition	<ol style="list-style-type: none"> 1.User should be authorized 2.User should specify sensor type 3.User should specify sensor 4.Dashboard should be shown 5.User should fill required blanks 6.System should process and show success message <ol style="list-style-type: none"> 6.1.In case error show error message go step 2 7.System should record changes 8.System should retransmit user to main page
Control Light	<ol style="list-style-type: none"> 1.User should be authorized 2.User should specify sensor type 3.User should specify sensor 4.Dashboard should be shown 5.User should fill required blanks 6.System should process and show success message <ol style="list-style-type: none"> 6.1.In case error show error message go step 2 7.System should record changes 8.System should retransmit user to main page
Control Door	<ol style="list-style-type: none"> 1.User should be authorized 2.User should specify sensor type 3.User should specify sensor 4.Dashboard should be shown 5.User should fill required blanks 6.User should chose open/close choice 7.System should process and show success message <ol style="list-style-type: none"> 7.1.In case error show error message go step 2 8.System should record changes 9.System should retransmit user to main page
Control curtain	<ol style="list-style-type: none"> 1.User should be authorized 2.User should specify sensor type 3.User should specify sensor 4.Dashboard should be shown 5.User should fill required blanks 6.User should chose open/close choice 7.System should process and show success message <ol style="list-style-type: none"> 7.1.In case error show error message go step 2 8.System should record changes 9.System should retransmit user to main page
Send Warning Message	<ol style="list-style-type: none"> 1.System should take details from sensor 2.System should verify problem 3.System should specify error type 4.System should send warning message type to user <ol style="list-style-type: none"> 4.1. Fire alarm message 4.2. Thief alarm message 4.3.Flood alarm message 4.4.Notification message 5.User should fill required blanks 6.User should chose open/close choice 7.System should process and show success message <ol style="list-style-type: none"> 7.1.In case error show error message go step 2 8.System should record changes 9.System should retransmit user to main page
Display Temperature	<ol style="list-style-type: none"> 1.System should specify sensor type 2.System should take details from sensors database 3.System should verify information 4.System should send information to server

	5.System should send information to application 6.System should display information to user 6.1.In case of error show error message go step 1
Display Humidity	1.System should specify sensor type 2.System should take details from sensors database 3.System should verify information 4.System should send information to server 5.System should send information to application 6.System should display information to user 6.1.In case of error show error message go step 1
Display Light Condition	1.System should specify sensor type 2.System should take details from sensors database 3.System should verify information 4.System should send information to server 5.System should send information to application 6.System should display information to user 6.1.In case of error show error message go step 1
Display Air Condition	1.System should specify sensor type 2.System should take details from sensors database 3.System should verify information 4.System should send information to server 5.System should send information to application 6.System should display information to user 6.1.In case of error show error message go step 1

4.10. Test cases

Test Case Id	Test Scenario	Test Steps	Test Data	Expected Results	Actual Result	Pass/Fail
TU01	Check user Login in Database	1.Go to Application 2.Enter user id 3.Enter user password 4.Click log in	User id User password	User should login to application	-	-
TU02	Application should show dashboard	1.Ask server for dashboard 2.Show options	Buttons Text Fields	User should see Menu and options bar	As expected	Pass
TU03	GUI check	1.Ask server for dashboard 2.Show options 3.All fields should be on page 4.All field should be shown correctly	Buttons Text Fields Visual Components	User should see Menu and options bar	As expected	Pass
TU04	Database Testing	1.Check if data stored correctly 2.Check data for integrity 3.Tables should have keys if required 4.Data should be rolled back in case of failure 5.Check if database fields stored with right values	Data	Database should work properly	As expected	Pass
TU05	Sensors Testing	1.Check sensor connections 2.Check sensor workability 3.Sensor should receive/send data to database	Sensor data	Sensor should work	As expected	Pass

4.11. Test Plan

4.11.1. Introduction

This report contains all the test plan that will be implemented to “Smart Home Prototype Senior Project”. The plan that we consider consists testing strategies, test items, features to be tested & not to be tested, criteria that is about pass/fail, test deliverables and test schedules.

4.11.2. Testing Strategy

We decided to use unit test, integration test, smoke testing, validation testing, stress testing, system testing.

Software Part:

- In that project unit test must be tested because there are many functions which is written with different software language.
- Project is containing many mobile application and backend process because of that, in these parts of this project, we will test that with system and integration testing.
- Stress testing will be used for the backend and servers.
- Smoke testing is related with Smart Home since whole system should be tested when new components are added.
- Validation testing will give information about Smart Home Prototype Senior Project are satisfied or not.

Hardware Part:

- Smoke testing will be used when new components (sensors) connected to the Arduino.
- Also, we will test compatibility of all of our sensors in the system.
- We will check functional and non-functional requirements for sensors and modules.

4.11.3. Test Items (Functions)

This section is giving information about test items that will be generate on the project.

1. Unit Test
2. System and Integration Testing
3. Smoke Testing
4. Stress and Performance Testing
5. Alpha/Beta Testing
6. Security Testing
7. Validation Testing

4.11.3.1. Unit Testing

- Participants: Temirlan Bayeshov, Anil Peker
- The Units: Functions which will compute statistical information correctly in R, functions that will send and retrieve data transaction in android, functions in backend about server abilities. Also, we will try the mocking Arduino sensor data before writing to database.
- Methodology: In R testthat library package and scripts will be used [8][9].After adding new codes we will be able to automatically re-run our testthat. Also it provides functions that will catch errors. Android will use JUNIT testing method, test folder will be created. PHPUnit framework will be used for server testing. We will mock our PHPUnit and JUNIT code using mockito framework. We will try to mock some sensor data like Temperature/Humidity sensor and control data reality.

4.11.3.2. System and Integration Testing

- Participants: Temirlan Bayeshov

- The Units: Data integration between database and Web server will tested.
- Methodology: Here participant start with bottom level units going to up, so bottom up strategy will be used. Before starting Integration Testing, Unit test should be successfully completed. We will check, if out data sets successfully arrived or delivered to Web Server from Database.

4.11.3.3. Smoke Testing

- Participants: Temirlan Bayeshov, Anil Peker
- The Units: Arduino Studio and Hardware.
- Methodology: QA team participants will manually test code written in Arduino Studio to verify work with sensors.

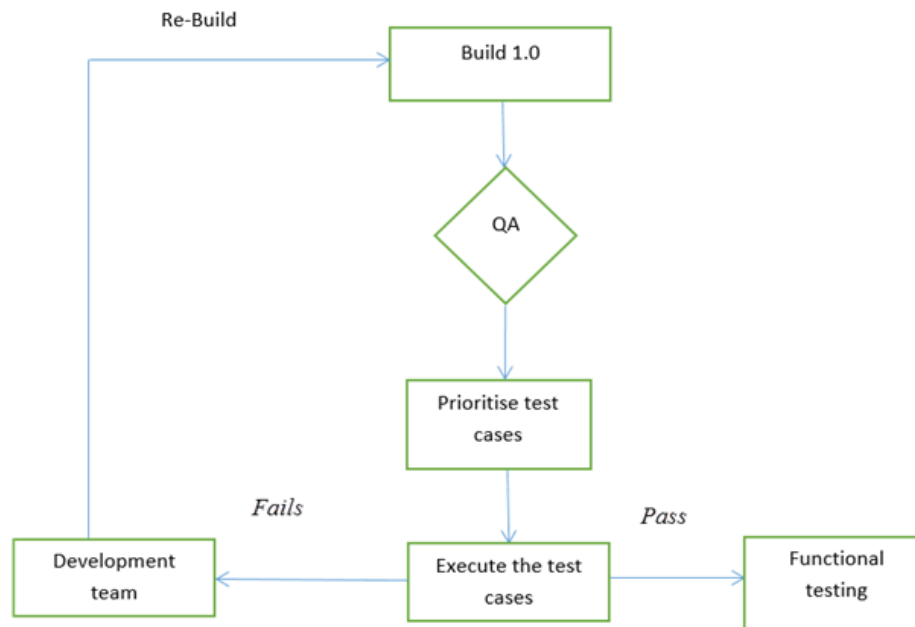


Figure 9: Smoke Testing Flowchart [10]

4.11.3.4. Performance and Stress Testing

- Participants: Temirlan Bayeshov, Anil Peker
- The Units: Mobile applications, Arduino.
- Methodology: Stress testing automation scripts will be created. NeoLoad [11] tool will be generate huge amount of users to evaluate and analyze mobile application performance under stress condition. Response time will be tested with help of time stamps in Arduino requests to the server.

4.11.3.5. Alpha/Beta Testing

- Participants: Anil Peker
- The Units: User Interface and Optimization for Android.
- Methodology: In that testing case, according to survey that will be done, Android and web page will be developed by the participant. For the Android App, Google Developer Console will be used.

4.11.3.6. Security Testing

- Participants: Anil Peker
- The Units: Android Authentication
- Methodology: For authentication, android app will generate token for every login. In that test case, participant will send request with random token and will check security of personal and smart home information reachable or not.

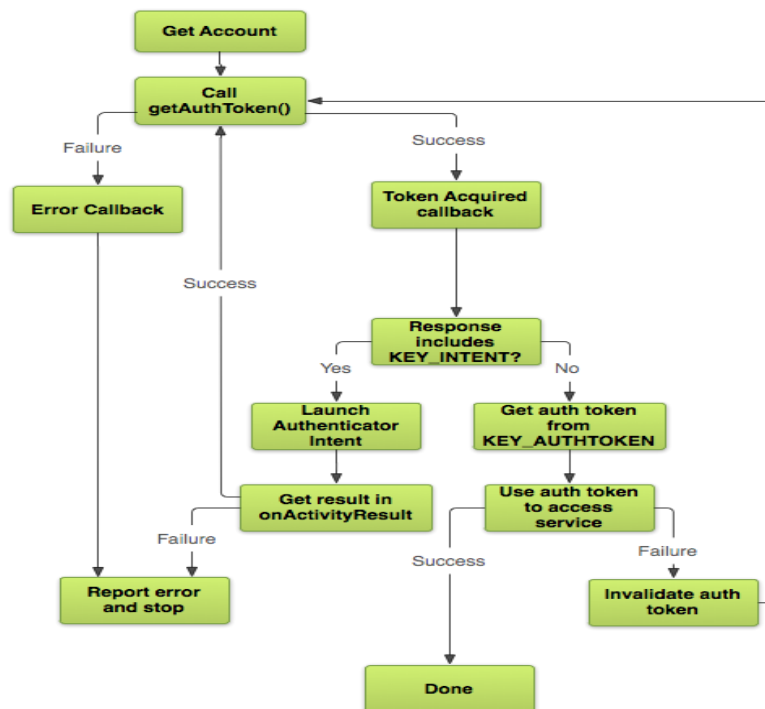


Figure 10: Token Generator Algorithm[12]

4.11.3.7. Validation Testing

- Participants: Anil Peker, Temirlan Bayeshov
- The Units: Unit Test Result, Integration Test Result, System Test Result, User Acceptance Testing Result, functional and non-functional requirements.
- Methodology: According to test case result, participants will validate the project using V-Model. In addition, we will check if our system validates with requirements.

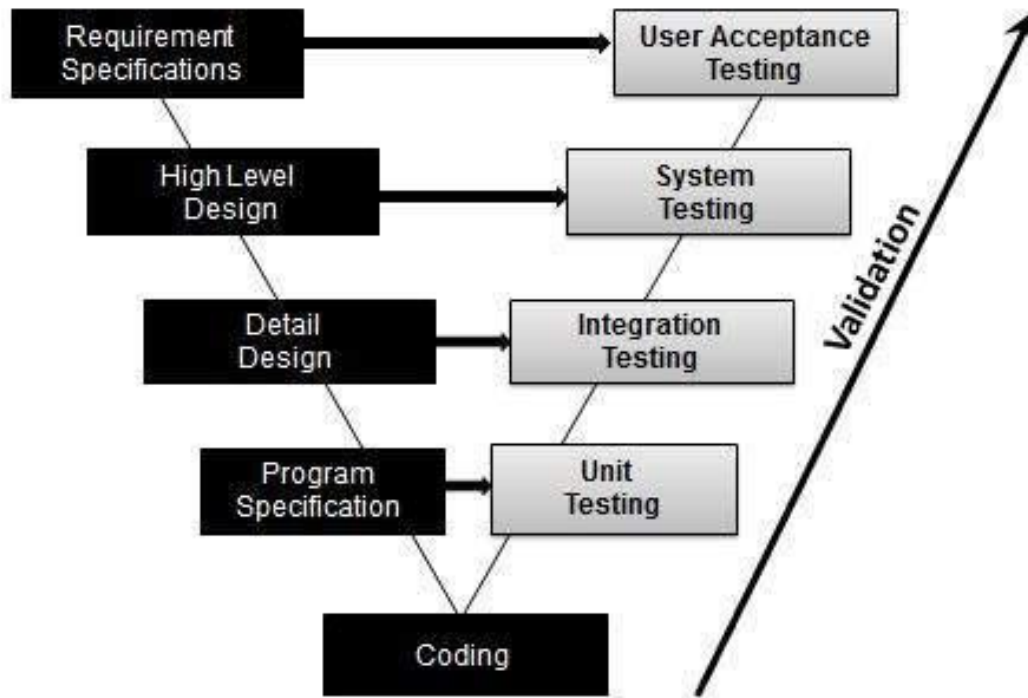


Figure 11: V-Model [13]

4.11.4. Summary of Features to Be Tested

- R statistical information in web server databases
- Database and Web Server integration
- Stress of mobile applications
- Response time of requests from servers and Arduino
- Sensors work with Arduino codes
- Android screens
- Login credentials and tokens generation for each user

4.11.5. Summary of Features Not to Be Tested

- Collection of data in sensors
- Database connection
- Internet connection of Arduino

4.11.6. Item Pass/Fail Criteria

- **Unit testing:** All test cases should be completed as expected. Unexpected results will lead to fail unit tests. All database sets should be sent and retrieved without any minor defects, because statistics should be computed in exact way from RStudio.
- **System and integration testing:** All modules and units should work together in proper way. All lower level plans such as Unit Testing should be completed without any defects. Data integration should be completed without any minor defects, otherwise statically computation will fail.
- **Performance and stress testing:** After performing Unit and Integration tests, Arduino hardware and web/mobile application will be tested. With help of NeoLoad tools stress test will be performed to find boundaries of how many users will be allow to enter server. Failure will be found depends on users' traffic. Arduino response time performance test should be completed without any defects, in case of long response time test will be failed.
- **Smoke testing:** Each sensor of Arduino should work independently of each other. In case of failure of one sensor minor defects will be found. Specific sensor test will be failed. If all sensors will work according the code in Arduino Studio test will be passed.
- **Security testing:** After performing performance and stress testing, security testing should be completed as expected without any failures or minor defects. In case of generating error token, test will fail. Each of verification data should be matched. Personal data should be reachable for each user. Otherwise security testing will fail.

4.11.7 Test Deliverables

Deliverable Name	Author(s)
Test plan	Anıl Peker
Test cases	Temirlan Bayeshov, Anıl Peker
Error Logs	Temirlan Bayeshov
Test closure report	Anıl Peker, Temirlan Bayeshov
Problem reports and corrective actions	Temirlan Bayeshov, Anıl Peker

4.11.8 Test Schedule

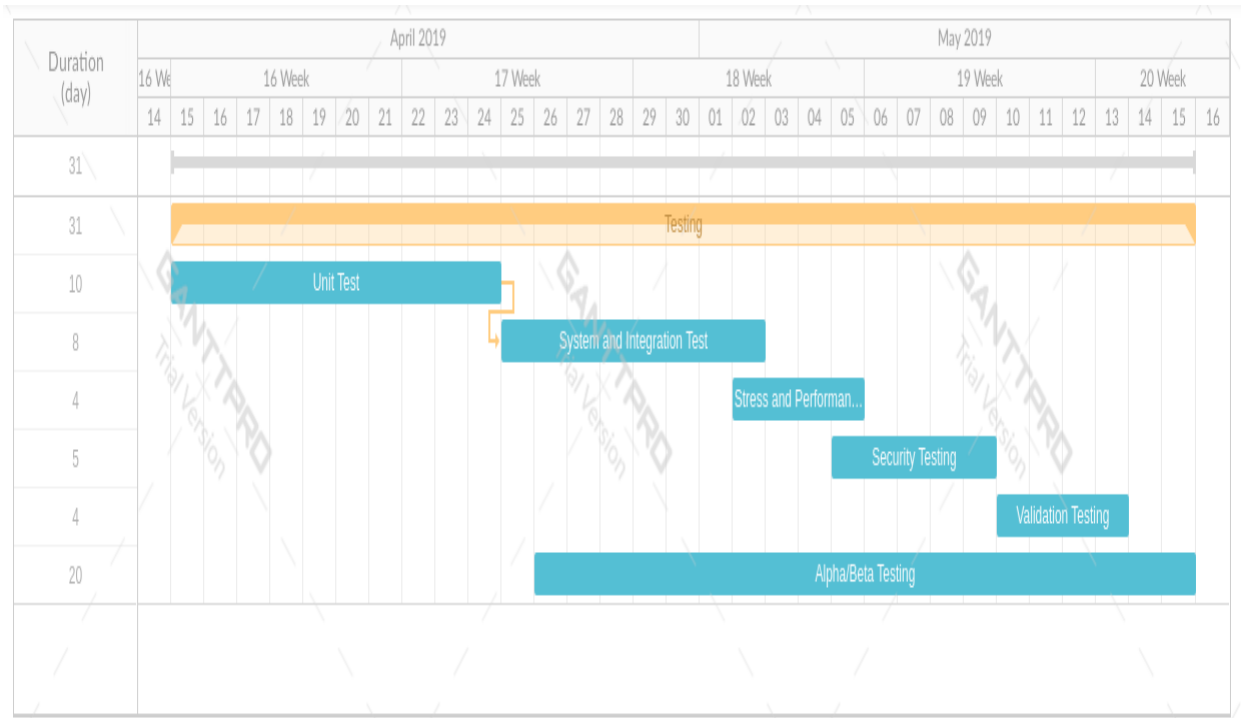


Figure 12: Test Schedule Gantt Chart

5. Design details

5.1. The architecture models- for all levels

You can see the all architecture models from [section 4.5](#).

5.2. Entity Relation Diagram Model

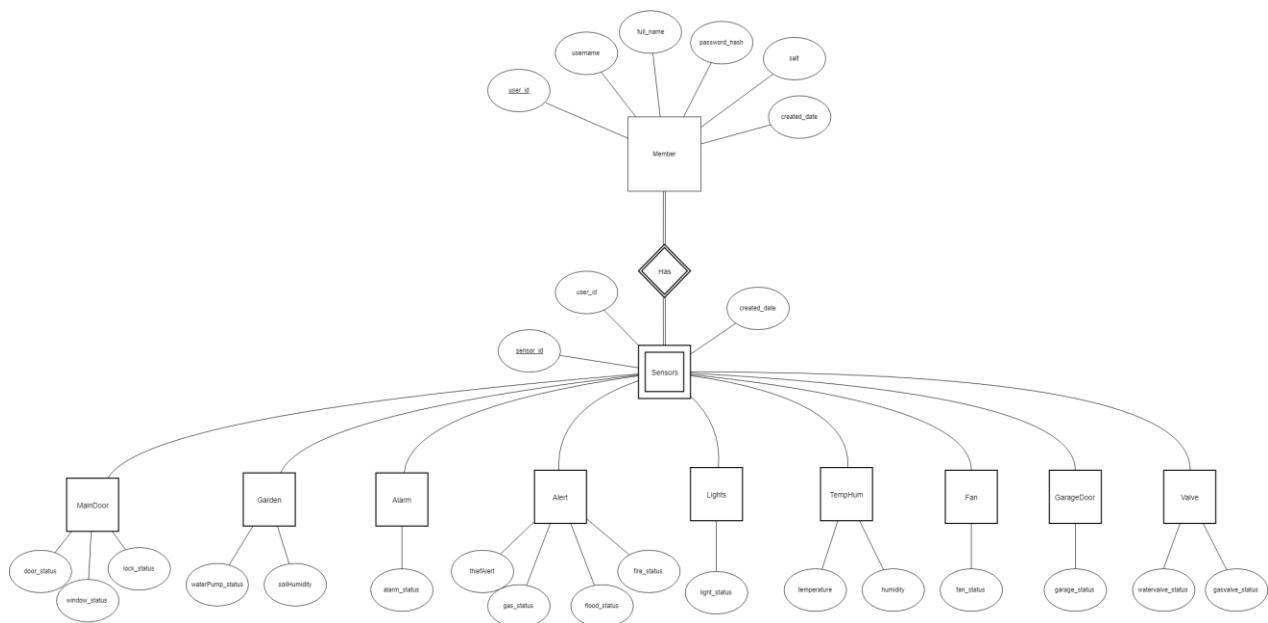


Figure 13: ERD Model

You can see the entity relation diagram which is in above. We have 11 table. It is an Enhanced Entity Relation Diagram. Sensor table is fully depending on the member of the system. Every sensor table contains user id as foreign key.

5.3. Data Flow Diagram

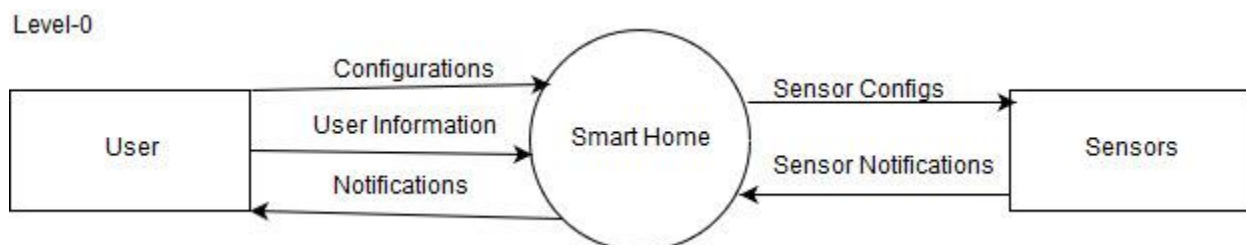


Figure 14: Data Flow Diagram Lever 0

For the data flow diagram, main part is smart home system(server, database, Arduino). User can control the manageable parts of system. Also, when user wants to enter to the system user data have been sending to system. Sensor part is controlling via user data and also some other sensors value is writing to the database. In unexpected situation, user is getting the warning message from the system.

Level-1

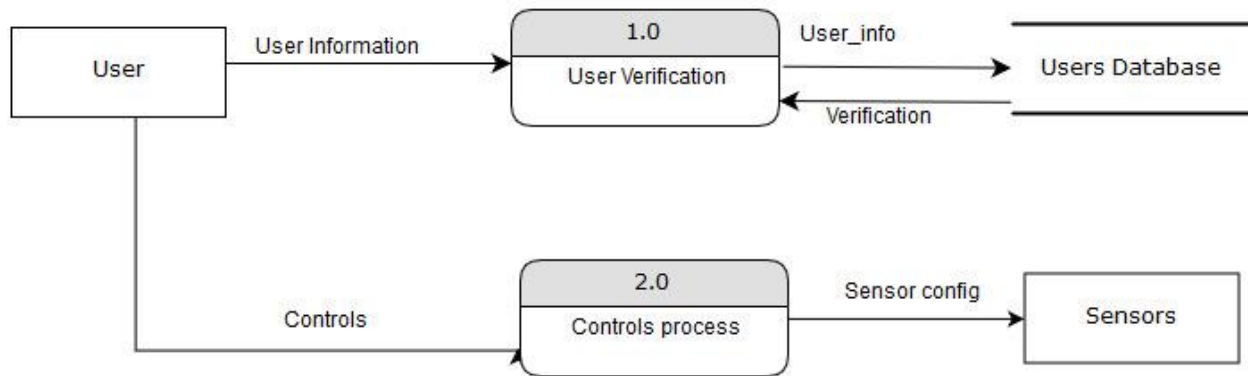


Figure 15: Data Flow Diagram Level 1

5.4. Sequence Diagram

You can see our sequence diagram in [section 4.7.](#)

5.5. List of Classes, Definitions and Hierarchy

5.5.1. Arduino Side (Nodemcu)

Mysql Connector: It is connecting the Arduino to MySQL Database. Actually, It is a library. To use it, firstly the object must be created with WIFI client and after that connector must be attach to cursor to work with SQL queries.

WifiClient: It is a library that used for connect the Arduino to wifi with WIFI name and password.

MQ-135 (Air Quality): It is a library which is using to get ppm value of air. After that, Arduino is doing some calculations with that library and returning ppm value which is comes from the sensor (MQ-135).

5.5.2. Android Side

Volley Framework: It is a framework which is using for Automatic scheduling of network requests. Multiple concurrent network connections.

Singleton Classes: It is a class which is entering to the interaction with volley framework to schedule the request with queuing system.

Session Handler: It is a class which is controlling the logged in user information and if the user did not enter to the system in 24-hours, it is automatically logout to the user account from application.

5.6. List of Interfaces Definitions

- ➔ I1 is using for login system, If user can enter the system, application will show the main dashboard of android application.

```
I1:UserAuthentication{  
    boolean postUserData(String,String)  
    void showMainDashboard()  
}
```

- ➔ I2 is using for the registration, If user can register correctly, user is going to main dashboard of his application page.

```
I2:Registration{  
    boolean postRegistarationData(String,String,String)  
}
```

- ➔ I3 is controlling the manageable sensors via user

```
I3: Control{  
    void open (string);  
    void close (string);  
}
```

- ➔ I4 is displaying the all of the datas in mobile application.

```
I4: Show{  
    void display ();  
}
```

- ➔ I5 is notification service method, which is warn the user the there is an unexpected situation apper in the home.

```
I5: Warning{  
    void notify ();  
}
```

- ➔ I6 is checking the database connection between server, application and database.

```
I6: DatabaseControl{
    int CheckDatabase (string);
}
```

➔ I7 is creating an json object and after it will be send to server.

```
I7: ServerMobile{
    void createJSON ();
}
```

➔ I8 is a server interface, which is taking the request which contain command datas and writing to database.

```
I8: ServerSensor{
    void command (string);
}
```

6. Implementation Details

6.1. Project Management

On this part we would like to explain what we are doing to manage our project. Project management is important to take firm steps forward and working well so from the date we started to project to now we tried to make plans such as creating schedule for working days and topics, estimations, assumptions, distribution of work, topics which we should give importance to learn etc. Depends on these plans, progress of our project became much more effective.

6.1.1. Project Estimation

Function Point Estimation

Inputs	Outputs	Inquiries	Logical Internal Files	External Interface Files
1.Control Temperature	1. Fire Alert Message	1. Temperature data query	1. Temperature data	1. User mobile login control
2.Control Door	2. Flood Alert Message	2.Humidity Data Query	2. Humidity Data	2. Statistical control data
3.Control Curtain	3. Gas Leak Alert Message	3.Light Data Query	3.Light Data	
4.Control Gas Valve	4. Temperature Status Message	4.Water Status Data Query	4.Door Data	
5.Control Water Valve	5. Humidity Status Message	5.Gas Data Query	5.Curtain Data	

	6.Fire Alert Status Message	6.Curtain Status Query	6.Water Valve Data	
	7.Theft Alert Message		7.Gas Valve Data	

Table 6: Function point estimation

Function Points			
Program Characteristic	Low Complexity	Medium Complexity	High Complexity
Num Of Inputs	1x3	1x4	3x6
Num of Outputs	7x4	0x5	0x7
Inquiries	2x3	2x4	2x6
Logical Internal Files	5x7	1x10	0x15
External Interface Files	0x5	0x7	3x10

Table 7: Function point complexity

UTFP=154

General System Characteristics (GSC)

1. Performance: 3
2. Data communication: 5
3. Transaction speed: 4
4. Reusability of the system: 2
5. Online data entry: 5
6. Sensor processing: 5

7. Installation structure: 4

8. Statistical processing: 4

9. Learning Algorithm processing: 3

$TDI = 35$ priority

$VAF = 35 * 0.01 + 0.65 = 1$

$ATFP = 154 * 1 = 154$

We are going to use 4 language such as R, PHP, C, Java and average of it :

$LOC = 154 * 62 = 9548$

Since the LOC is only applicable on Basic COCOMO.

COCOMO semi-detached mode:

$a = 3.0$ $b = 1.12$ $c = 0.35$

$KDSI = 154 * 62 / 1000 = 9.548$

$MM = 3 * 9.548^{1.12} = 37.56 \sim 38$ months

$TDEV = 2 * 38^{0.35} = 7$ months

6.1.2. Project Tasks and Durations Assumptions

- Literature Research and Background Information (40 Days)
 - Familiarization of Arduino → 10 Days
 - Familiarization of Sensors and Sensor Connections → 40 Days
 - Familiarization of Web Server Structure → 15 Days
 - Familiarization of Database Structure → 40 Days
 - Familiarization of Android → 5 Days
 - Familiarization of Statistical Information Solutions → 15 Days
 - Familiarization of Learning Algorithm → 20 Days

- Design Planning (25 Days)

- Web Server Design → 7 Days
 - Arduino and Sensors Circuit Design → 20 Days
 - Android Application Design → 8 Days
 - Database Design → 20 Days
 - Smart Home Construction Design → 3 Days
- Implementation (120 Days)
 - Server Implementation → 40 Days
 - Sensor Implementation and Coding → 80 Days
 - Mobile Application Implementation → 20 Days
 - Database Implementation → 50 Days
 - Tests(45 Days) → 30 Days

6.1.3. Milestones

- Literature Research and Background Information
- Design Planning
- Implementation
- Tests

6.1.4. Activity Diagram

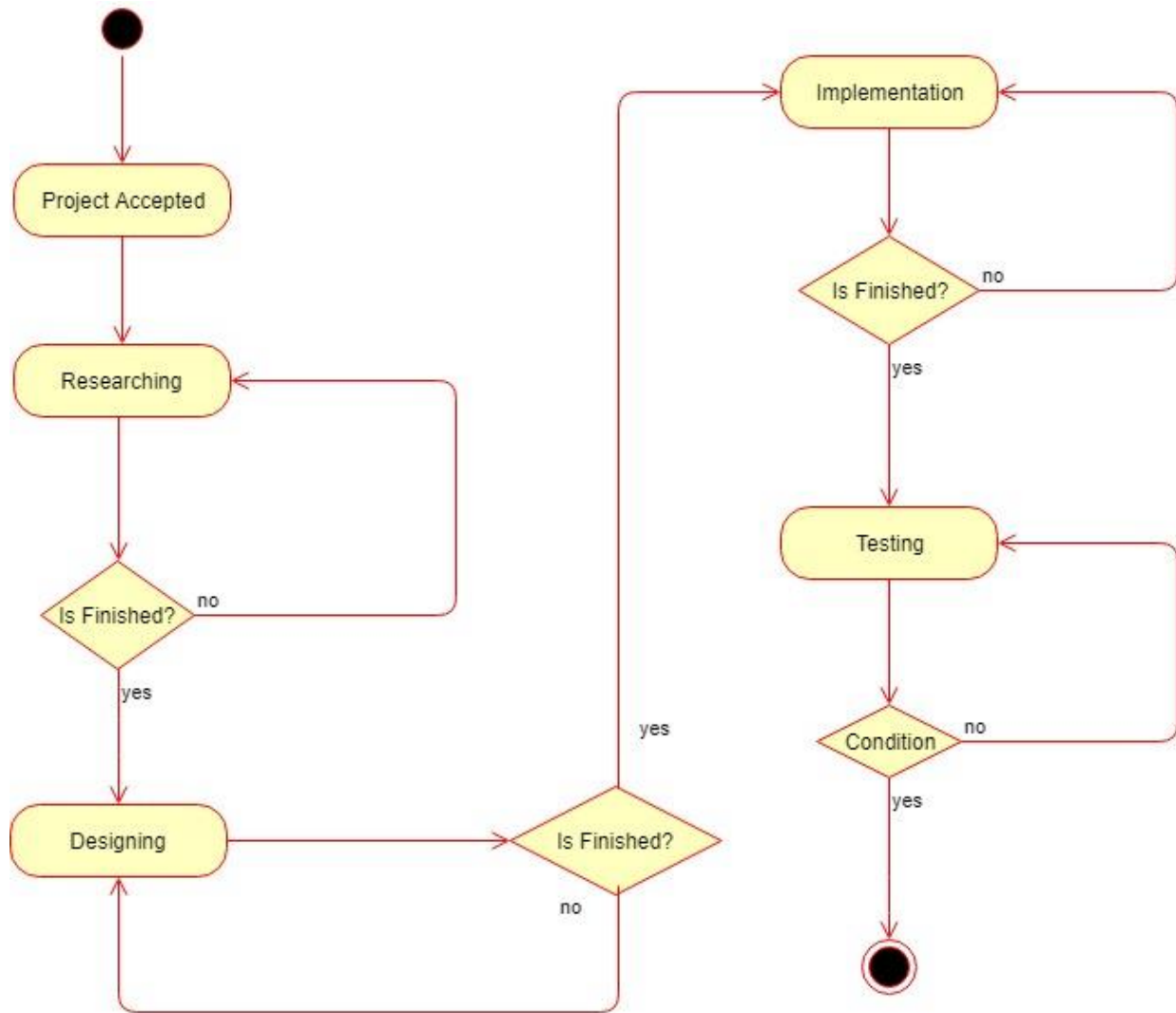


Figure 16: Activity Diagram

This diagram is showing out main activity algorithm and plans. Our plan is mainly, if a step didn't fully finished, then do not pass the next step.

6.1.5. Gantt Chart

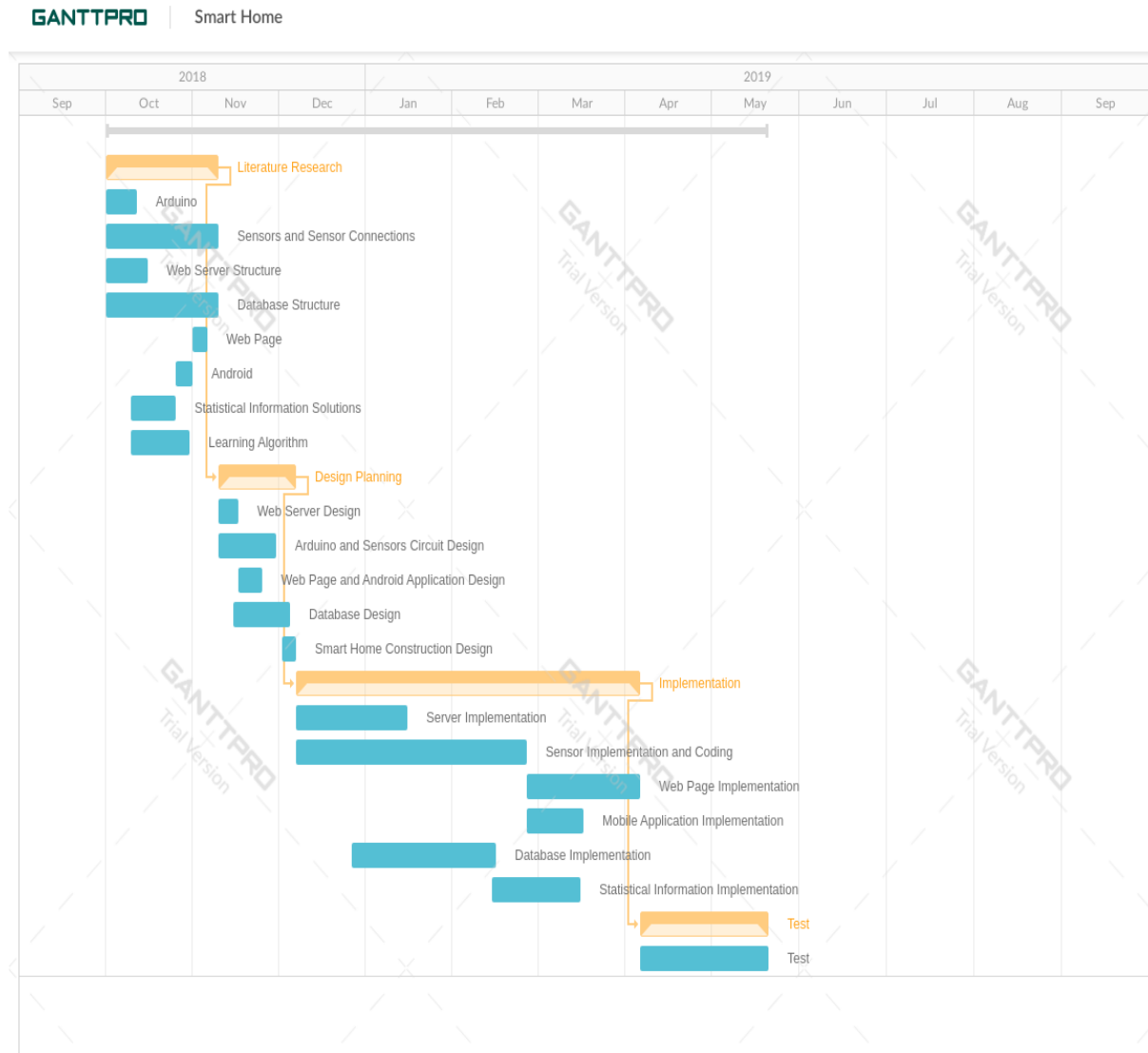


Figure 17: Gantt Chart

6.1.6. Task allocation chart

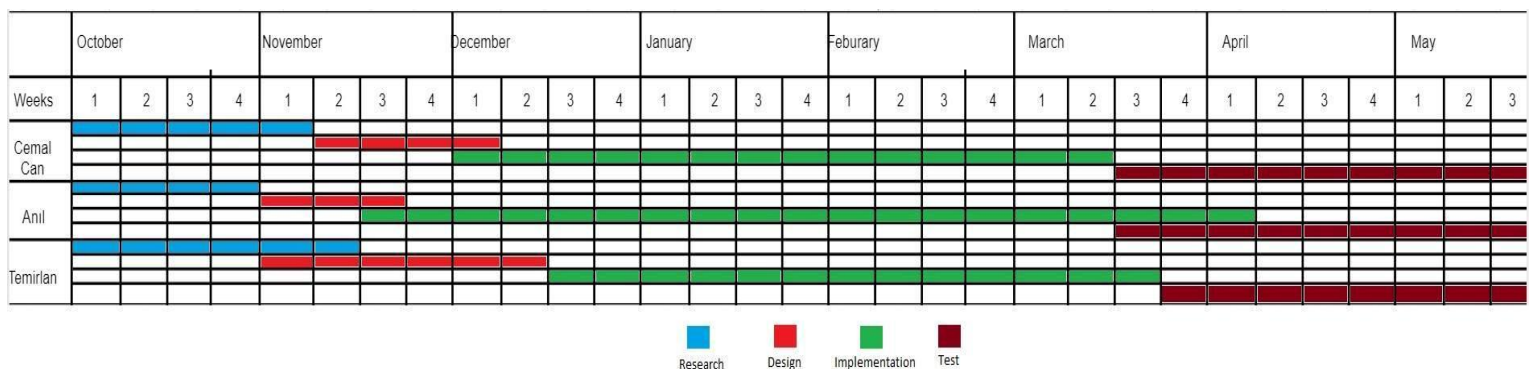


Figure 18: Task Allocation Chart

6.2. Testing section

In this part of the project we will perform:

- Unit tests
- System and Integration tests
- Alpha/Beta tests

Tests will be performed on our Smart-Home project, we will test separate units, modules, database, web server, applications to define errors or defects before final project will be released.

Unit testing for different functions on R passed successfully. As expected, all tests gave right values and we obtained that data computation to our project will be performed without any defects.

System and Integration testing for separate module connections passed successfully. We obtained expected results as interaction between database and computational modules. Also, data retrieving from server as expected done without any errors.

6.2.1 Unit Tests

The Unit Name: R function to sum up all temperature values

Participants: Temirlan Bayeshov

Pass/Fail criteria: testthat library will be used. First our main code will call calTempSum function, to take all temperature values from database and sum them up. After it will call test_Temp function to make unit test for a temperature final value. We will check if value type is data frame or not, because we should make a data frame to take values from database and sum them, and second test will check if there is only single value of temperature sum. In script we used expect_that function from the testthat library.

Results: We obtained success in both tests, because our final temperature value is single and data frame. So, as expected we the R unit test for temperature passed without any failures.

```
1 test_TempSum.R testing calTempSum function values 2 0 FALSE FALSE 0 0.01 0.01
  real passed
1 0.03 2
```

Figure 1: R calctemp sum

The Unit Name: R function to sum up all humidity values

Participants: Temirlan Bayeshov

Pass/Fail criteria: testthat library will be used. First our main code will call calHumiditySum function, to take all humidity values from database and sum them up. After it will call test_Humidity function to make unit test for a humidity final value. We will check if value type is data frame or not, because we should make a data frame to take values from database and sum them, and second test will check if there is only single value of humidity sum. In script we used expect_that function from the testthat library.

Results: We obtained success in both tests, because our final temperature value is single and data frame. So, as expected we the R unit test for humidity passed without any failures.

```
file context test nb failed skipped error warning user
1 test_HumiditySum.R testing calHumiditySum function values 2 0 FALSE FALSE 0 0.01
  system real passed
1 0 0.01 2
```

Figure 2: R calchumidity sum

The Unit Name: Android Credential Control with Non-User Person and Transition between the pages

Participants: Anil Peker

Pass/Fail criteria: With espresso framework in the android studio, test tried to connect to the system with non-registered information. The criteria are, system shouldn't allow to enter the user in the system without non-determined information and the application should inform the user. Also, we tried the transition between the dashboard and tried to see non-reachable dashboard with wrong path.

Results: The result that we obtain system is not allow to enter the user to the system and it is giving enough information which is comes from the Web service. We could reach to see the dashboards that we gave the wrong path.

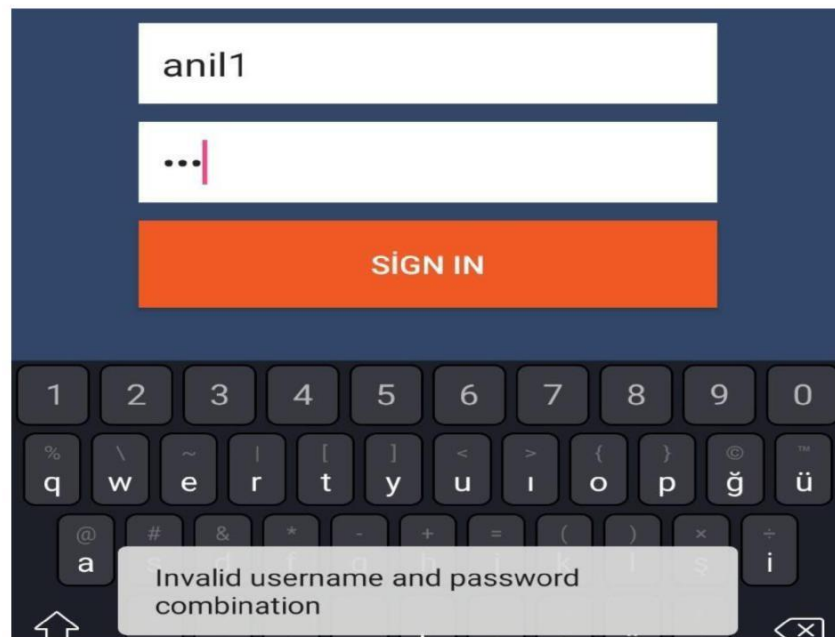
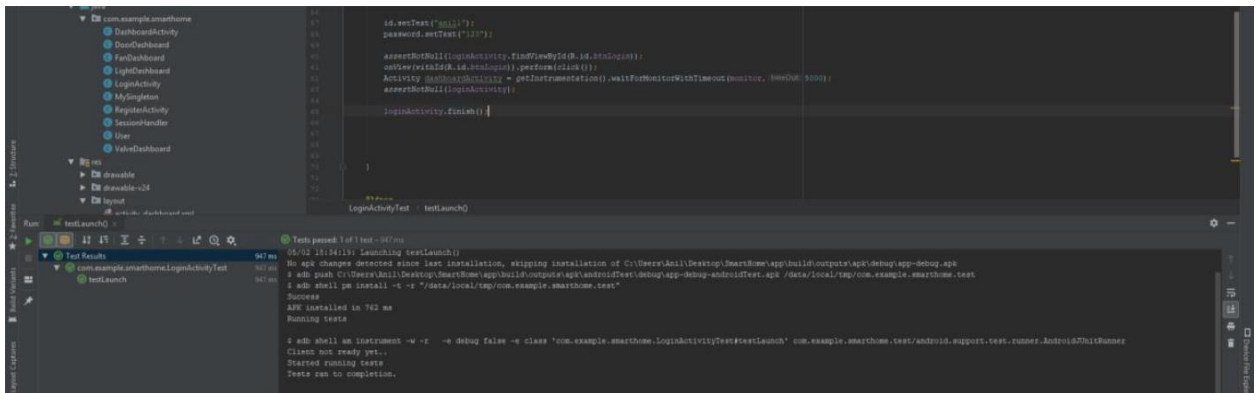


Figure 3: Andriod Credential Contr

The Unit Name: Arduino Mocking Sensor Datas

Participants: Anil Peker

Pass/Fail criteria: Sometimes, there is a asynchronous communication between Arduino and Nodemcu but also they are using serial communication. Because of this un-orchestrating organization sometimes the sensor data can be unrealistic. For example, at the fail case, temperature data can be “123412” type of data. We tried to prevent this error using mocking. Pass criteria is the data reality like temperature must be range in some numbers.

Results: After mocking and some optimization data is never writing wrong to database.



The image shows two side-by-side screenshots. The left screenshot displays Arduino code for mocking sensor data. The code includes three functions: `Mocking Temp Hum Data`, `Mocking Thief Alert Data`, and `Mocking Flood Alert Data`. Each function checks for specific conditions (e.g., temperature range, thief alert status) and inserts data into a database using `queryInsertTemp`, `queryInsertThief`, and `queryInsertFlood` respectively. The right screenshot shows a database table with columns: `sensor_id`, `user_id`, `temperature`, `humidity`, and `created_date`. The table contains 15 rows of data, all with a `temperature` value of 24 and a `humidity` value of 62. The `created_date` values range from 2019-05-22 21:24:34 to 2019-05-22 21:28:05.

sensor_id	user_id	temperature	humidity	created_date
20376	12327	24	62	2019-05-22 21:24:34
20377	12327	24	62	2019-05-22 21:24:35
20378	12327	24	62	2019-05-22 21:24:36
20379	12327	24	62	2019-05-22 21:24:36
20380	12327	24	62	2019-05-22 21:24:36
20381	12327	24	62	2019-05-22 21:24:36
20382	12327	24	62	2019-05-22 21:24:47
20383	12327	24	62	2019-05-22 21:25:11
20384	12327	24	61	2019-05-22 21:25:32
20385	12327	24	61	2019-05-22 21:25:54
20386	12327	24	60	2019-05-22 21:26:17
20387	12327	24	61	2019-05-22 21:26:35
20388	12327	24	62	2019-05-22 21:26:51
20389	12327	24	60	2019-05-22 21:27:12
20390	12327	24	60	2019-05-22 21:27:24
20391	12327	24	62	2019-05-22 21:27:47
20392	12327	24	60	2019-05-22 21:28:05

Figure 19: Arduino Mocking

6.2.2. System and Integration Testing

The Unit Name: Integration of R and MySQL database in server

Participants: Temirlan Bayeshov

Pass/Fail criteria: R scripts will be used to test if we can connect to MySQL and retrieve data from tables which stored in our server. Firstly, we will create connection Result function for connection to database and test it if our two modules connected successfully. We will use two methods for integration testing: `isIdcurrent` and `dbIsValid`. First method will show use if currently we connected to database while second one will show if our database valid. If both of the methods return TRUE than test successfully passed, otherwise in case of FALSE test will be failed. We expect that methods will return TRUE for both methods. After passing these criteria we will check if our data retrieved successfully.

Results: We obtained pass for integration test. Both of methods which checks connection to database throw R code showed successful results. We obtained that we can connect to our MySql database on our server, so now we will be able to retrieve data from server.

```
connectionResult<-function()
{
  con<-dbconnect(MySQL(),host="localhost",dbname="anildata",user="root",password="")
  return (con)
}

> test.connectionResult<-function()
+ {
+   dbIsValid(con)
+   isIdCurrent(con)
+   return(print(isIdCurrent(con)))
+ }

> con=connectionResult()

> conStatus=test.connectionResult()
[1] TRUE

> dbIsValid(con)
[1] TRUE
```

Figure 4: R connection test

The Unit Name: System testing retrieving of data from server

Participants: Temirlan Bayeshov

Pass/Fail criteria: R scripts will be used for testing if our data will be successfully retrieved from database. After successful integration test of connection to database, we are able to test if we can take data sets from it. We will use test.dataRetrieveResult function to test if we can access to temperature/humidity, etc. values. If we will obtain values, we will be able to see them in print method as a table. Otherwise, in case of fail, we will get an error message.

Results: After first passed integration test which are connection to database module, we tested if we can retrieve data sets from MySQL database. Our test passed successfully without any error messages. Now we able to connect and take a data for next steps of statistical performance.

```
> test.dataRetrieveResult<-function()
+ {
+   result=dbsendQuery (con,"select * from temphum")
+   data=fetch(result,n=-1)
+   temperature=subset(data,select=c(temperature))
+   return (temperature)
+ }
> temp=test.dataRetrieveResult()
> print(temp)
  temperature
1           24
2          123
3           26
4           40
> |
```

	temperature
1	24
2	123
3	26
4	40

Figure 5:R Data retrieving test

6.2.3. Alpha/Beta tests

The Unit Name: Project Survey

Participants: Anil Peker

Process: We ask questions about controlling of automated home via mobile application to find pros and cons and what should be added or removed from the project. In addition, we observed the most favorable function of the survey participants.

Pass/Fail criteria: Pass criteria is useful functions for customers that they choose most suitable for them. Fail criteria is non- useful functions for customers that they choose most non-suitable for them.

Results: In our survey, we found 10 participants for participating.

- i. 1 out of 10 participants didn't agree of session handler because user don't want to enter his/her credential in every 24 hours.
- ii. 1 out of 10 participants suggested to create a secondary application for children with restricted operation on home.
- iii. 3 out of 10 participants wanted to get more required information about the application interfaces and operations.
- iv. 1 out of 10 participants gave idea about water pumping system in the garden should be automated in Arduino according to soil moisture.
- v. For the security part, 2 out of 10 participants desired to have a camera to feel more secured.
- vi. In the interface part, only 1 participant out of 10 didn't like the graphical user interface of the application.
- vii. 3 out of 10 participants completely agreed with our application.

6.3.4. Test Requirements

6.3.4.1. Summary of the Tests Passed

6.3.4.1.1. Unit tests for R calculating functions

Test: Unit test

Library: testthat

Scripts: R language

Functions: calcTempSum, calcHumiditySum, test_TempSum, test_TempHumidity

- R function to sum up all temperature values passed. As we expected we made our values from INT type to Data Frame type. After getting all values from database we summed the up and we get only single value at the result. Both unit tests passed successfully.
- R function to sum up all humidity values passed. As we expected we made our values from INT type to Data Frame type. After getting all values from database we summed the up and we get only single value at the result. Both unit tests passed successfully.

6.3.4.1.2. Unit tests for Android Credential Control with Non-User Person

Test: Unit test

Library: Junit

Scripts: Java

Functions: Login interface and login php function

- The function that the interface does for login system tested for non-user registration information. We tried give dummy and non-correct information to login the system and we observed the required warning and the application continue to work properly.
- The user cannot reach every dashboard with the wrong paths information.

6.3.4.1.3. System and Integration Tests for connection of R and Database

Test: System and Integration test

Methods: dbIsValid, isIdcurrent

Scripts: R language

Functions: connectionResult, test.connectionResult, test.DataRetrieveResult

- Integration test passed as expected, both of methods return TRUE boolean values for connection from R to our MySql database in server. We used bottom up method, starting with unit tests, than we tested connection first before retrieving data sets.
- Test for retrieving data successfully passed, now we able to access tables and values inside of them for future statistical computations.

6.3.4.2. Summary of the Tests Failed

Test: Alpha/Beta Testing

Methods: Surveying

Functions: Mobile Application and Functionality

- 7 out of 10 participants didn't like some parts of application and its functionality fully.

- Some of the participants suggested new operations about session handler, security deficiency, GUI design and notifications.

7. User Manual

7.1. System Overview

Smart Home is a system, which allows collecting information about home status like temperature, humidity, etc. It provides to users' different options to control their home by using mobile application, notifies and performs necessary action when emergency situations appears, provides security. The application shows data collected from sensors and stored in database. Smart Home operates on mobile devices with Android operating system. Database and servers provided by the team.

7.2. Organization of the Manual

The user's manual consists of four sections: System Overview, System Summary, Getting Started, and Reporting. System Overview gives briefly information about Smart Home system. System Summary section provides a general overview of hardware and software requirements, system configuration, access levels and some risk conditions. Getting Started section explains how to get started with Smart Home system, register and installation performed by technical team. Reporting section describes how computational actions performed.

7.3. SYSTEM SUMMARY

System Summary describes and provides a general overview of the system to the user. Also, it outlines the hardware and software requirements, user's access to the system and system configurations. Also, user access levels discussed below.

7.3.1. System Configuration

Smart Home operates on mobile devices with Android operating system. It is compatible with all Android 4.0 or higher versions. The application requires connection to Internet for saving or retrieving data sets to database, which were collected by sensors installed in home. Information and data can be viewed by using Smart Home mobile application. After installation of sensors and mobile application on the device, Smart Home can be used immediately. No need for further configurations. In case of changes like adding or removing specific sensors, technical team will be sent. Database and Servers will be provided by us.

7.3.2. User Access Levels

Only registered users with installed sensors can use the system and can to save data to database. No interaction with database will be made manually, in case of losing statistical collection.

7.3.3. Contingencies

In case of power cutoff sensors will be stopped and no data collection will be performed but old data will be stored in database with old statistical information. In case of theirs no Internet connection on the mobile application, user will not be able to control the system or retrieving new statistical information, but all data will be collected at the background and when user will have internet connection, he/she will be able to retrieve data sets which collected at the background.

7.4. GETTING STARTED

This section will provide information how to be ready to use Smart Home system and what needed before using automated system.

7.4.1. Installation and Logging

First before using a system user should register to our system and fill specific preferred fields which sensors, he/she will need to monitor Smart Home. After registration to system, User ID and password will be delivered to owner. Depends on filled fields, technical team will be sent to configure and install sensors and Arduino parts. Database and Servers will be provided to user after installations.

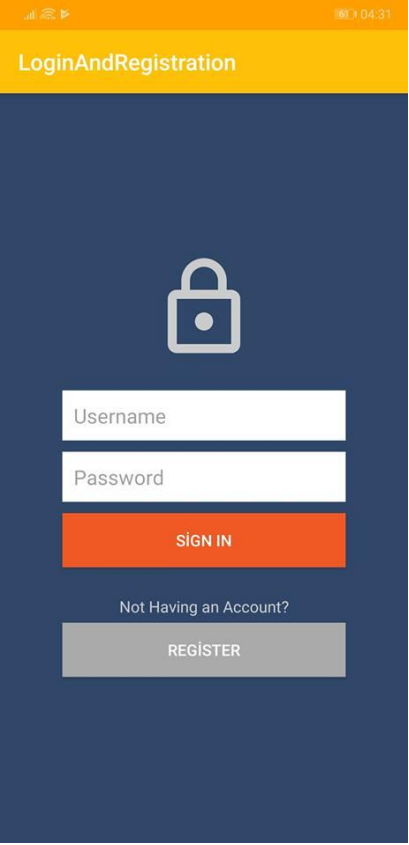
A mobile application screen titled "LoginAndRegistration" with a yellow header. The background is dark blue. In the center is a white padlock icon. Below it are two white input fields labeled "Username" and "Password". Under the "Password" field is an orange button labeled "SIGN IN". Below that is a link "Not Having an Account?" and a grey button labeled "REGISTER". The top status bar shows signal strength, Wi-Fi, and the time 10:04:31.

Figure 20: Login Screen

7.4.2. System Menu

Mobile application's main menu consists of 6 dashboards and three different values like temperature, humidity and alarm status. With different dashboards user will be allowed to control and monitor different functions like lights, valves, doors, fan status and alarm.

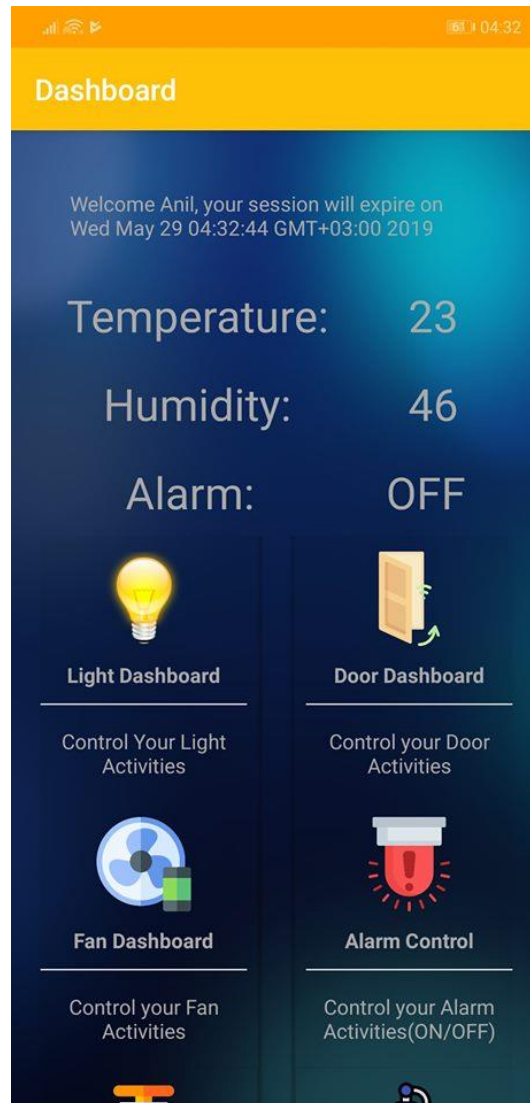


Figure 21: Main Dashboard

7.4.2.1 Sample Fan Dashboard

Sample Fan Dashboard will show temperature and humidity of current time, status of fan (open/close). Also in this dashboard statically computation performed and plotted by R to temperature and humidity values shown. User able to check temperature and humidity data by day, month and year. 4 different bar plots provided to user.



Figure 22: Fan Dashboard

7.4.2.2. Sample Door Dashboard

Sample Door Dashboard will show status of four main components: main door lock status, door, window and garage conditions. By using this components user allowed to control the given components, able to lock/unlock doors. Security is the main aim here.

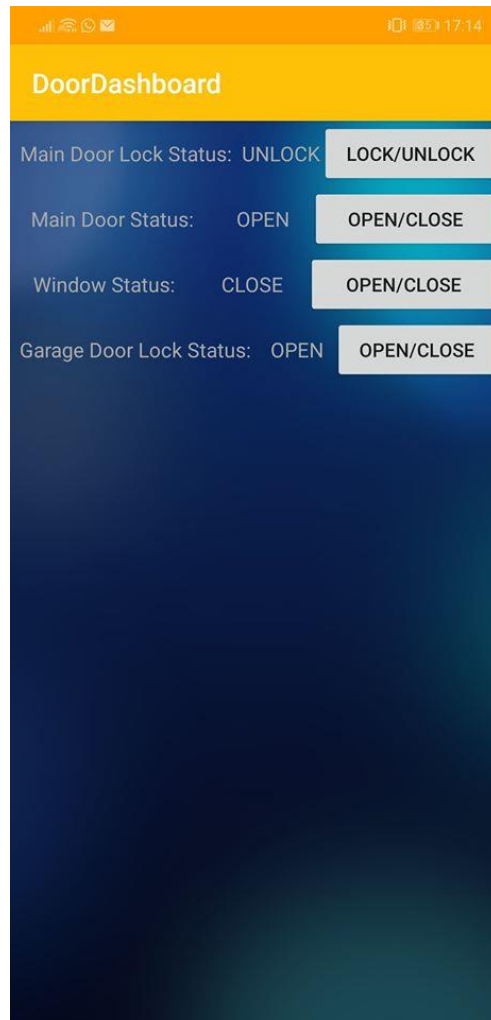


Figure 23: Door Dashboard

7.4.2.3. Sample Light Dashboard

Sample Light Dashboard will show status of lights in the house. User will be able to control all lights which installed during the installation process. Specific button ON/OFF provided for control function.

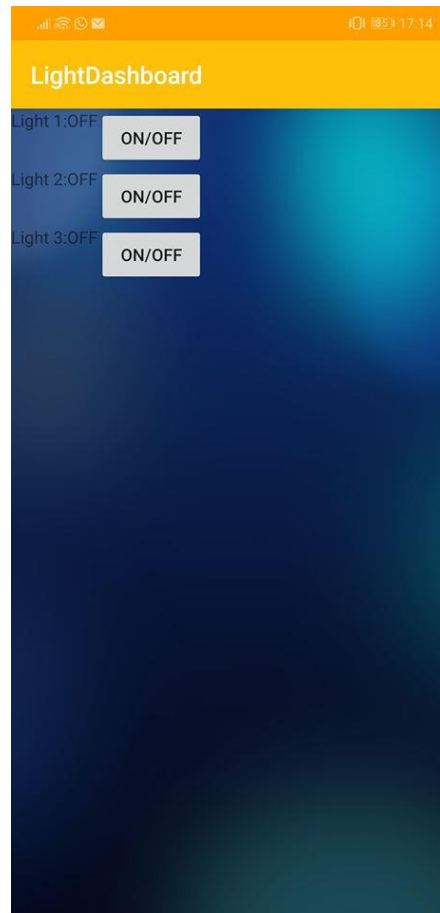


Figure 24: Light Dashboard

7.4.3.4. Sample Valve Dashboard

Sample Valve Dashboard will show 5 main statuses: flood, fire, gas leak, water valve and gas value. If some of provided condition will show not secure message, it means user in dangerous situation and another function to protect customer will be performed.

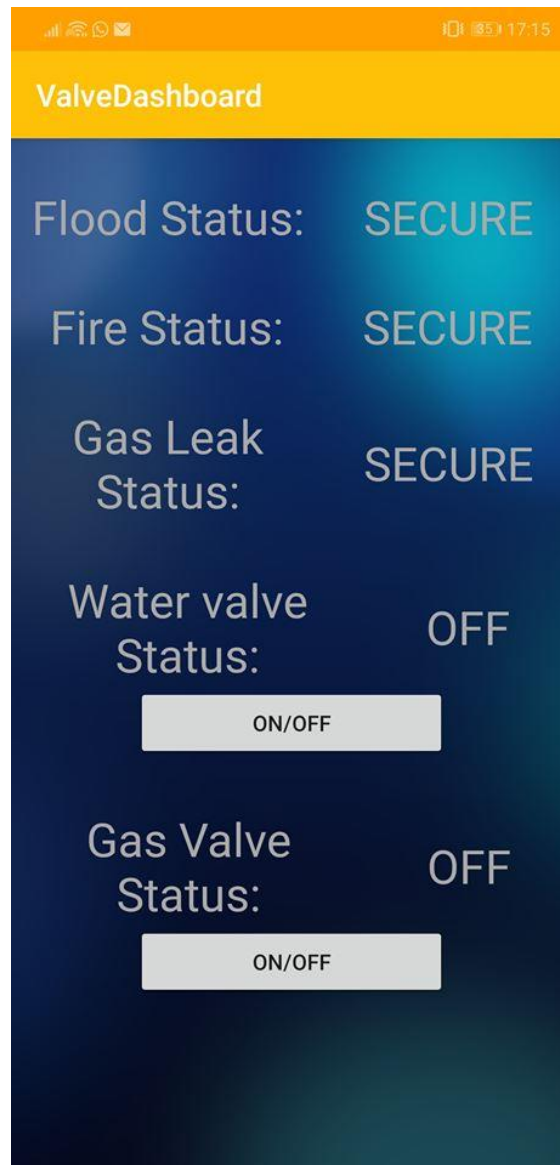


Figure 25: Valve Dashboard

7.4.3.5. Sample Pump Dashboard

Sample Pump Dashboard will show humidity of garden soil and water pump status. Depending on humidity values automated water pump will be performed.

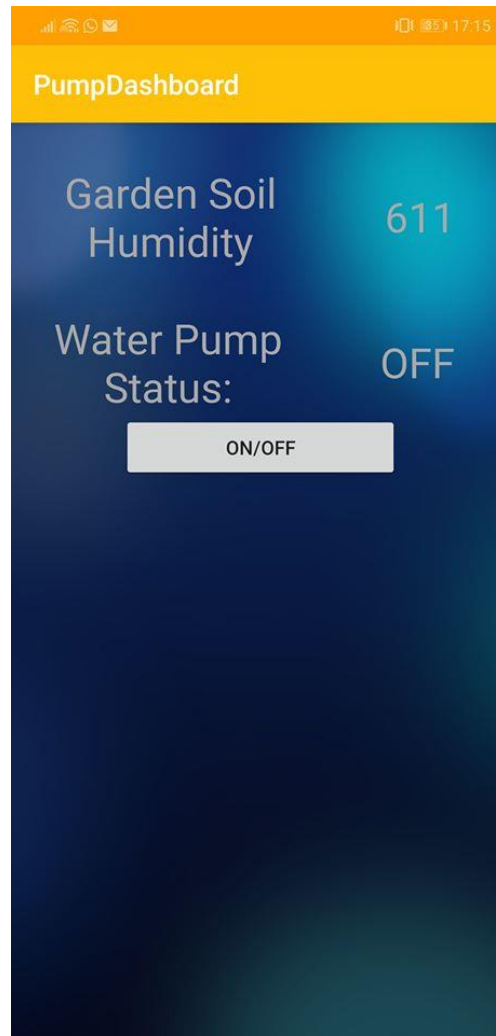


Figure 26: Pump Dashboard

7.4.4. Changing User ID and Password

User ID and password can be changed only by contacting producer.

7.5. REPORTING

Reporting section describes how statistical actions performed to a data and to reach them.

7.5.1. Bar plots

In fan dashboard, user can reach 4 different bar plots which provides statistical information for both temperature and humidity in the house. Data stored by sensors, sent to database and R connected to it will retrieve this data for further operations. User able to see average temperature/humidity values per day, month and year. All of this information computed by R sent back to database after necessary computations. Every 5 minutes data will be updated.

7.5.2. Sensors statuses

All the installed sensors statuses stored in database. Time stamps help user to understand at what time and which actions triggered.

8. Installation Manual and Deployment Details

8.1. Introduction

The purpose of this guide to describe how to install necessary software for a given project and make it operational.

8.2. Revision history

The Revision history table shows the date, changes, and authors who have worked on this document.

Version/Change request number	Version date	Description of changes	Author
	15/05/2019	Introduction	Temirlan Bayeshov
	18/05/2019	R part added	Anil Peker
	22/05/2019	Testing installation added	

8.3. Intended audience and reading suggestions

Installation and Deployment manual is intended to be used by project team and be used to describe planning, performing, or maintaining the installation or deployment. Project members and teachers will be able to reach this guide.

8.4. Server Configurations

8.4.1. Server 1 (Database)

Installation of this product is supported on the following operation systems and versions:

- **Wamp server:** Version 3.1.7 - 64bit

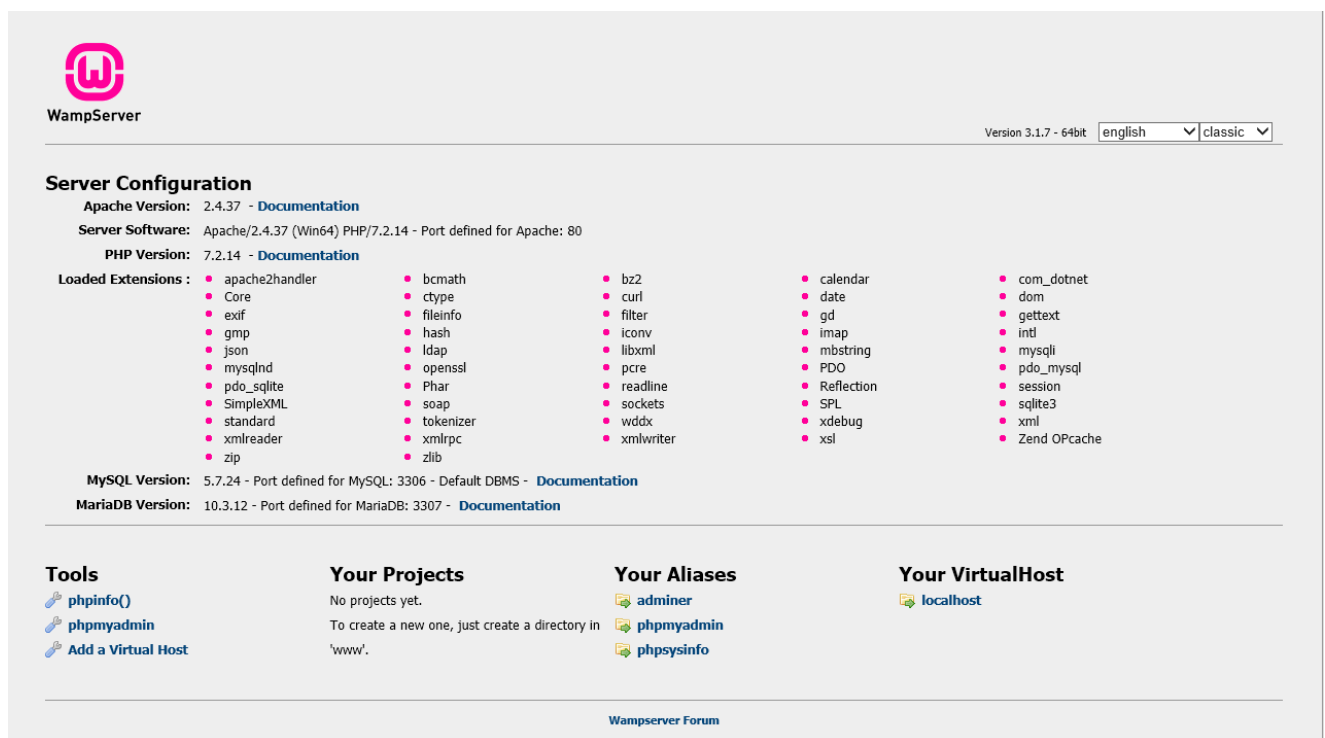
- **Apache Version:** 2.4.37
- **Server Software:** Apache/2.4.37 (Win64) PHP/7.2.14 - Port defined for Apache: 80
- **PHP Version:** 7.2.14
- **MySQL Version:** 5.7.24 - Port defined for MySQL: 3306 - Default DBMS
- **MariaDB Version:** 10.3.12 - Port defined for MariaDB: 3307

Configuration

SQL Server Network Configuration

TCP/IP should be enabled

Named Pipes should be enabled



The screenshot displays the WampServer configuration interface. At the top, the WampServer logo and version (3.1.7 - 64bit) are shown, along with language (english) and theme (classic) dropdowns. The main section is titled "Server Configuration" and lists the following details:

- Apache Version:** 2.4.37 - [Documentation](#)
- Server Software:** Apache/2.4.37 (Win64) PHP/7.2.14 - Port defined for Apache: 80
- PHP Version:** 7.2.14 - [Documentation](#)

Below these, a grid of "Loaded Extensions" is displayed, including:

- apache2handler, Core, exif, gmp, json, mysqlnd, pdo_sqlite, SimpleXML, standard, xmlreader, zip
- bcmath, ctype, fileinfo, hash, ldap, openssl, Phar, soap, tokenizer, xmlrpc, zlib
- bz2, curl, filter, iconv, libxml, pcre, readline, sockets, wddx, xmlwriter
- calendar, date, gd, imap, mbstring, PDO, Reflection, SPL, xdebug, xsl
- com_dotnet, dom, gettext, intl, mysqli, pdo_mysql, session, sqlite3, xml, Zend OPcache

At the bottom, the "MySQL Version" (5.7.24 - Port defined for MySQL: 3306 - Default DBMS - [Documentation](#)) and "MariaDB Version" (10.3.12 - Port defined for MariaDB: 3307 - [Documentation](#)) are listed.

The footer section contains four columns:

- Tools:** phpinfo(), phpmyadmin, Add a Virtual Host
- Your Projects:** No projects yet. To create a new one, just create a directory in 'www'.
- Your Aliases:** adminer, phpmyadmin, phpsysinfo
- Your VirtualHost:** localhost

The WampServer Forum link is located at the bottom center.

Figure 27: Wamp Server Configuration Page[14]

8.4.2. Server 2 (Mobile Application)

- **Java Version:** Version JRE 8 - 64bit
- **Android Version :** 3.4.1
- **R version:** R-3.5.3 64bit
- **Rstudio version :** 1.2.1335
-

8.4.2.1 Packages

The following software packages must be installed on the operating system prior to installation of the software:

- Volley Framework

Volley offers the following benefits:

- Automatic scheduling of network requests.
- Multiple concurrent network connections.
- Transparent disk and memory response caching with standard HTTP [cache coherence](#).
- Support for request prioritization.
- Cancellation request API. You can cancel a single request, or you can set blocks or scopes of requests to cancel.
- Ease of customization, for example, for retry and backoff.
- Strong ordering that makes it easy to correctly populate your UI with data fetched asynchronously from the network.
- Debugging and tracing tools.

Figure 28: Volley Framework Benefits[15]

8.5. Software Installation

8.5.1. Server 1 (Database)

Installation Steps

1. Download Wamp server from official Wamp website.
2. Install Wamp for a local server host.
3. Open localhost from browser settled during installation.
4. Open phpMyAdmin from the dashboard.
5. Initial username: “root” and password “ ”.
6. Create a new database catalog for different sensor values.
7. Create a new user configured for SQL Authentication named “nodemcu”.
8. Set the user mapping for the user created in previous step to add the database role membership for the catalog created in step 2 just to insert and update queries.
9. Create a new table for values like temperature, humidity, etc.
10. Create a new table for new users registered.
11. Create new tables for R calculations.
12. Ready for use.

8.5.2. R studio

Installation Steps

1. Download R-3.6.0 for a suitable operating system from <https://cran.r-project.org/>
2. Download RStudio Desktop or another version from <https://www.rstudio.com/products/rstudio/download/>
3. Follow installation process.
4. Open RStudio and download required packages:
 - `install.packages("DBI")`
 - `install.packages("RMySQL")`
 - `install.packages("ggplot2")`
 - `install.packages("lubridate")`
5. Use the libraries:
 - `library(ggplot2)`
 - `library(DBI)`

- library(RMySQL)
 - library(lubridate)
6. Connect to create MySql database created in section 3.1.1.
 7. Send a query to select datasets from temperature and humidity.
 8. Fetch retrieved data frames.
 9. Using date stamps divide data to day, month and year values.
 10. Calculate average, mean value for temperature both daily and monthly.
 11. Calculate average, mean value for humidity both daily and monthly.
 12. Create a new table for average temperature monthly.
 13. Create a new table for average temperature monthly.
 14. Create a new table for average temperature daily.
 15. Create a new table for average humidity monthly.
 16. Create a new table for average humidity daily.
 17. Write mean values of temperature to daily table.
 18. Write mean values of temperature to monthly table.
 19. Write mean values of humidity to daily table.
 20. Write mean values of humidity to monthly table.
 21. Create a png file for a daily average temperature.
 22. Create a png file for a monthly average temperature.
 23. Create a png file for a daily average humidity.
 24. Create a png file for a monthly average humidity.
 25. Create a barplot for a daily average temperature.
 26. Create a barplot for a monthly average temperature.
 27. Create a barplot for a daily average humidity.
 28. Create a barplot for a monthly average humidity.
 29. Clear results of fetched data.
 30. Close database connection.
 31. Create a batch file for R scripts by using CMD.
 32. Arrange windows scheduler for running R and repeat process daily with stamp of 5 minutes.

8.6. Testing the Installation

1. Navigate your web browser to localhost and enter to phpMyAdmin dashboard. Enter appropriate credentials to log in. Check if database updated.
2. Open cmd and make path to batch output file, check if png files updated every 5 minutes.
3. Check if R working as expected by checking values from database and bar plots.
4. Ensure that mobile application receives all plots and data needed.
5. Ensure that all components working as expected.
6. Ensure that database connection is working.
7. Ensure that bar plots updated every 5 minutes.
8. Ensure that Windows Scheduler working daily.

9. Possible extension details

We can extend our project by adding new sensors. For example, for security purpose we can add camera inside and outside of home. Actually, Kodali[16] is giving better ideas about security based home automated system and extensions. Also, we can extend our project by using laser in case of movements. Also, we can add better sensors for monitoring different kind of gas leakage.

Automated robot systems can be added to home project for making user life more comfortable. We can install water heater/cooler system.

10. References

1. N. Sriskanthan and Tan Karand. "Bluetooth Based Home Automation System". Journal of Microprocessors and Microsystems, Vol. 26, pp.281-289, 2002.
2. Muhammad Izhar Ramli, Mohd Helmy Abd Wahab, Nabihah, "TOWARDS SMART HOME: CONTROL ELECTRICAL DEVICES ONLINE" ,Nornabihah Ahmad International Conference on Science and Technology: Application in Industry and Education (2006)
3. Vaccum Fire Protection system. (n.d.) http://www.fireflex.com/datas/pdf/Anglais/Vactec/Leaflet/Fireflex_Vactec.pdf
4. Air Quality Information for the Sacramento Region (n.d.). <http://www.sparetheair.com/health.cfm>
5. Health effects of air pollution. (2019). <https://www.canada.ca/en/health-canada/services/air-quality/health-effects-indoor-air-pollution.html>
6. Chan, Marie, Daniel Estève, Christophe Escriba, and Eric Campo. "A review of smart homes—Present state and future challenges." Computer methods and programs in biomedicine 91, no. 1 (2008): 55-81.
7. Khorov, Evgeny, Andrey Lyakhov, Alexander Krotov, and Andrey Guschin. "A survey on IEEE 802.11 ah: An enabling networking technology for smart cities." Computer Communications 58 (2015): 53-69.
8. Testthat library, <https://www.utest.com/tools/testthat>
9. R test, <https://www.r-bloggers.com/unit-testing-with-r/>
10. NeoLoad, <https://www.neotys.com/>
11. Smoke Testing Flowchart, <https://mysullys.com/flow-chart-of-series-test/smoke-testing-learn-examples/>
12. Token Generator Algorithm, https://www.researchgate.net/figure/Flowchart-containing-the-steps-of-the-developed-compiler_fig5_236510290
13. V-Model, https://www.tutorialspoint.com/software_testing_dictionary/v_model.htm
14. Wamp Server, <http://www.wampserver.com/en/>
15. Volley Framework, <https://developer.android.com/training/volley>
16. Kodali, Ravi & Jain, Vishal & Bose, Suvadeep & Boppana, Lakshmi. (2016). IoT based smart security and home automation system. 1286-1289. 10.1109/CCAA.2016.7813916.