# 

NEF3002 Applied Project

Internet Of Things

Control of TP-Link E27 Light Bulb

Manikanda Sarma (s4578698)

Brendan McGuinness (s4579824)

Dang Nguyen (s4578996)

Nam Nguyen (s4579694)

Anthony Tran (s4551761)

# 

[**Contribution Table**](#_497tc75khyet)4

[**F1. Introduction Of The Project**](#_wvdry6h9kgmk)5

[1.1 Project Background](#_1qsl9k35l4tq) 6

[1.2 Project Description](#_1qsl9k35l4tq) 6

[1.3 Objective](#_1qsl9k35l4tq) 6

[1.4 Main Functions](#_rhhhgwwr4c0u) 6

[**F2. Introduction of Light Control Application**](#_loxzpmi54oyx)7

[2.1 Interface](#_60cwp5rjnka9) 7

[2.2 Details of Light Control Application Control Interface](#_ifw4bjm5d8fg) 8

[2.3 Structure of Application](#_qlpraqpnquav) 11

[**F3. Development Platform**](#_kewg4w4j0hdb)12

[3.1 Integrated Development Environment](#_yjr5nulso3m1) 12

[3.2 Languages Used](#_dn2sh944nvow) 13

[XML](#_u7b19n8gq0pv) 13

[JAVA](#_m6iyf47du35w) 14

[3.3 Postman](#_wyvdxzxvloa9) 14

[3.4 TPG API](#_97150zjz84fl) 16

[**F4 System Architecture**](#_g56mkt3b87ce)17

[4.1 Development View](#_qsnyhdfr119s) 17

[4.2 Logical View](#_q4l3ioypcy6f) 18

[4.3 Physical View and Process View](#_ad0lme3d57jp) 19

[4.4 Sequence Diagram](#_a5o3scg7bg4j) 21

[**F5 Linkages**](#_ll1c02adk9hj)22

[5.1 Connection between user, application, and smart bulb](#_94ta8uu8aclo) 22

[**F6 Detail Design**](#_awlscgxl26)25

[6.1 Sub-feature Detail](#_keygnqx4hxng) 25

[6.2 Data Flow Direction](#_lj46g7skk2k0) 25

[**F7 Test Result**](#_mnlqwbnd3mnz)27

[7.1 Metrics](#_10v8u6e04sl2) 27

[7.2 Test performance](#_5a0yrpvts3hn) 28

[7.3 Test environments](#_i5hff6bfjeec) 29

[**F8. The Difficulties Overcome**](#_d2pgxo25q21d)31

[**F9. Summary**](#_voy0rtuupx)33

[**F10 Conclusion**](#_im4gx3d0u1ra)35

[**F11. References**](#_3utgz6oct5cg)36

# 

# Contribution Table

# 

|  |  |  |  |
| --- | --- | --- | --- |
| Student Name(s) / Number | % Contribution | Description of Contribution | No of weeks worked for this assessment |
| Manikanda Sarma (s4578698) | 20% | User Manual, Introduction of the Project, The Difficulties Overcome | 5 weeks |
| Brendan McGuinness (s4579824) | 20% | Poster, Section 2, 3, 4, Summary and References of Final Report | 5 weeks |
| Dang Nguyen (s4578996) | 20% | Build Application, Testing and Deployment | 5 weeks |
| Nam Nguyen (s4579694) | 20% | Build Application, Testing and Deployment | 5 weeks |
| Anthony Tran (s4551761) | 20% | Test Report, Section 5, 6, 7 and Conclusion of Final Report | 5 weeks |

# 

# F1. Introduction Of The Project

## 1.1 Project Background

The term IoT (Internet of Things) refers to the devices that are interconnected through wireless networks. These devices can communicate with each other without human intervention. IoT devices are widely used in household applications, industrial applications, manufacturing and healthcare. There are numerous IoT devices such as smart lights, air conditioners and security systems that are being used in households around the world.

Smart lighting is one of the interesting applications of IoT. In recent years, the usage of IoT bulbs has increased exponentially due to the benefits associated with them. The main advantage of smart light bulbs is the fact that they are energy efficient and they have a longer lifespan as compared to traditional bulbs. Smart bulbs can minimize the wastage of energy as they enable the user to control features such as the brightness and saturation.

## 1.2 Project Description

The SmartLight application enables users to control the Kasa1 smart light bulb from their android mobile devices. The user would be able to modify different features of the bulb such as its color, brightness, temperature and saturation. The user can also turn the smart bulb on or off using the application.

## 1.3 Objective

The user can install the SmartLight application2 and proceed to create and register an account by using his/her email address. The user can then connect to the bulb using its ID and add it to the SmartLight application.

Following this process, the user can select the desired bulb and proceed to modifying its features.

# 

## 1.4 Main Functions

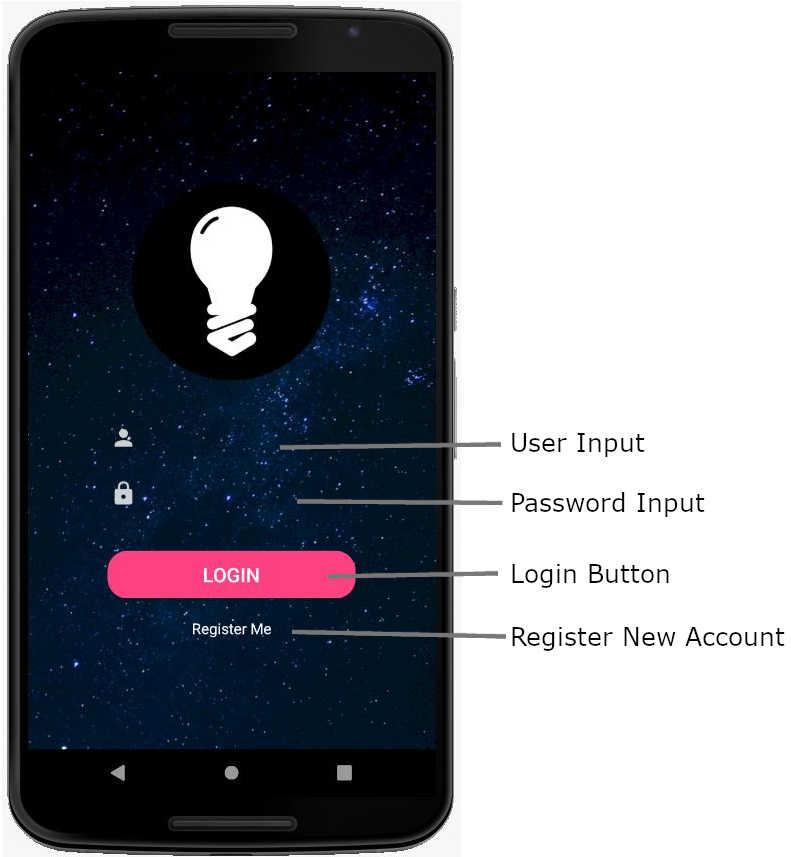
* The smart light bulb can be turned on/off.
* The brightness can be adjusted using the slider.
* The temperature of the IoT bulb can be set using the slider.
* The color of the bulb can be changed.
* The saturation of the bulb’s color can be changed using the slider.

# 

# 

# F2. Introduction of Light Control Application

## 2.1 Interface

 Fig 2.1

Shown in figure 2.1 is the introductory interface for when the app is loaded. The light bulb icon is the logo for the application. Beneath it there are 4 interactive fields.

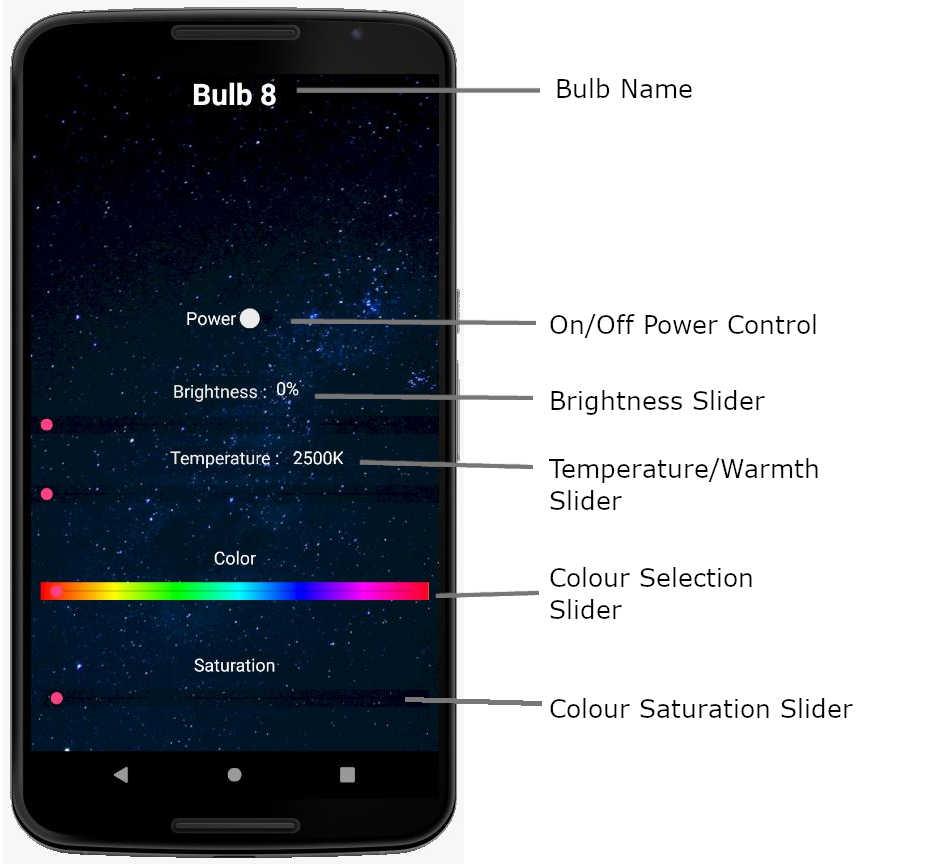
The first being the username input field which requires the email address that is registered to the account.

Second is the password field which is encrypted so as to not return a visual display of private information.

The third interactive field is the login button. This requires the first two input fields to be filled out with registered account information. When clicked the page is linked to the lightbulb display page.

The final field is a button containing the text “Register Me”. This is for new users to select. Once selected it will take the user to a page where they can register themselves an account.

## 2.2 Details of Light Control Application Control Interface

Fig 2.2.1

Shown in figure 2.2.1 is the controls interface. This is used to operate the features of the bulb. At the top of the interface is the name of the bulb which is being controlled. Beneath that are 5 interactive fields that work to control the 5 different features of lighting variance that the bulb offers. The second to fifth features use a slider to give a more accurate control.

First is the On/Off Power Control, a switch that will turn the bulb on or off depending on its state.

The second slider which controls the brightness of the bulb. Similar to a manual dimmer switch found in many homes, it slides left to right to increase the brightness by percentage. From 0% (completely off) to 100% (maximum brightness)5.

Third is the Temperature/Warmth Slider, this feature, measured in Kelvin, controls the intensity of the light emitted from the bulb. The range is from 2500k (a warmer, yellowish tone) to 9000k (a cool, super white tone). The ends of the spectrum represent traditional incandescent globes and the lighting tones offered with the opportunity to blend them to the users preference.

The fourth slider controls the ability to select which colour the user wishes the bulb to emit.

The fifth and final control for the bulb is colour saturation. Colour saturation is the intensity of the colour emitted. A low saturated colour gives the appearance of being washed out or pastel in tone. Increasing the saturation intensifies the colour making it appear more true to its tone.

HSL Colour Chart

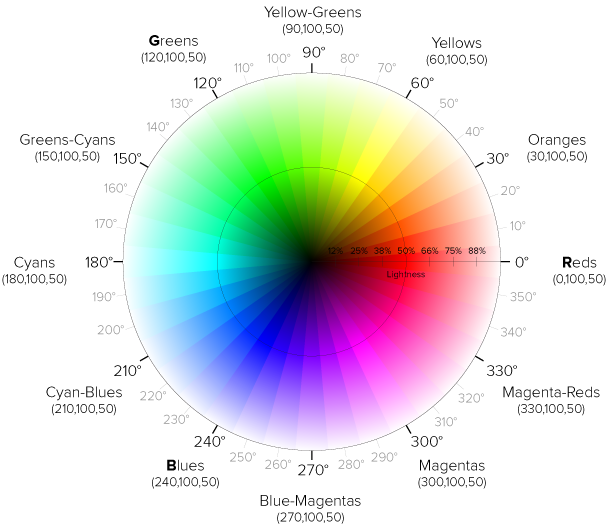
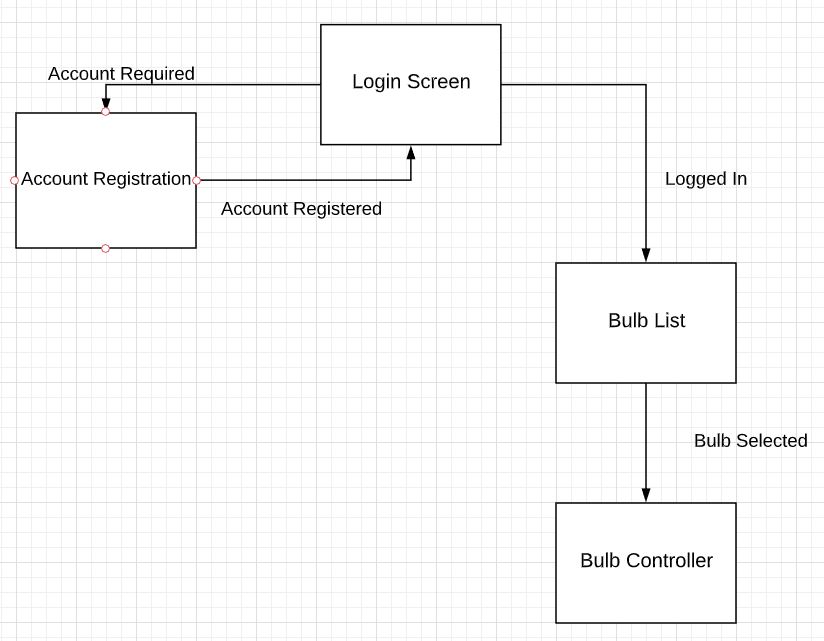
fig 2.2.2

Figure 2.2.2 shows the HSL colour chart. This is the chart that represents the spectrum of tones and colours available for the bulb control application. That blackest tone on the chart represents the light being switched off in a completely darkened room.

## 2.3 Structure of Application

Fig 2.3

The beauty of the structure of the application is in its simplicity. The power of the application is its ability to control the bulb.

The initial screen upon loading is the login screen. The two paths from this are:

1. To register a new account. Once the account details are registered, the user will be redirected back to the login screen to enter their registered email and password.
2. The bulb list, where the user can select which bulb they wish to operate.

Once the user has selected the bulb they wish to operate they are directed to the bulb controller where they can manipulate the features of the light.

# F3. Development Platform

## 3.1 Integrated Development Environment

The Integrated Development Environment used in the creation of the Light Bulb Control Application is Android Studio6.

*Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems.8*

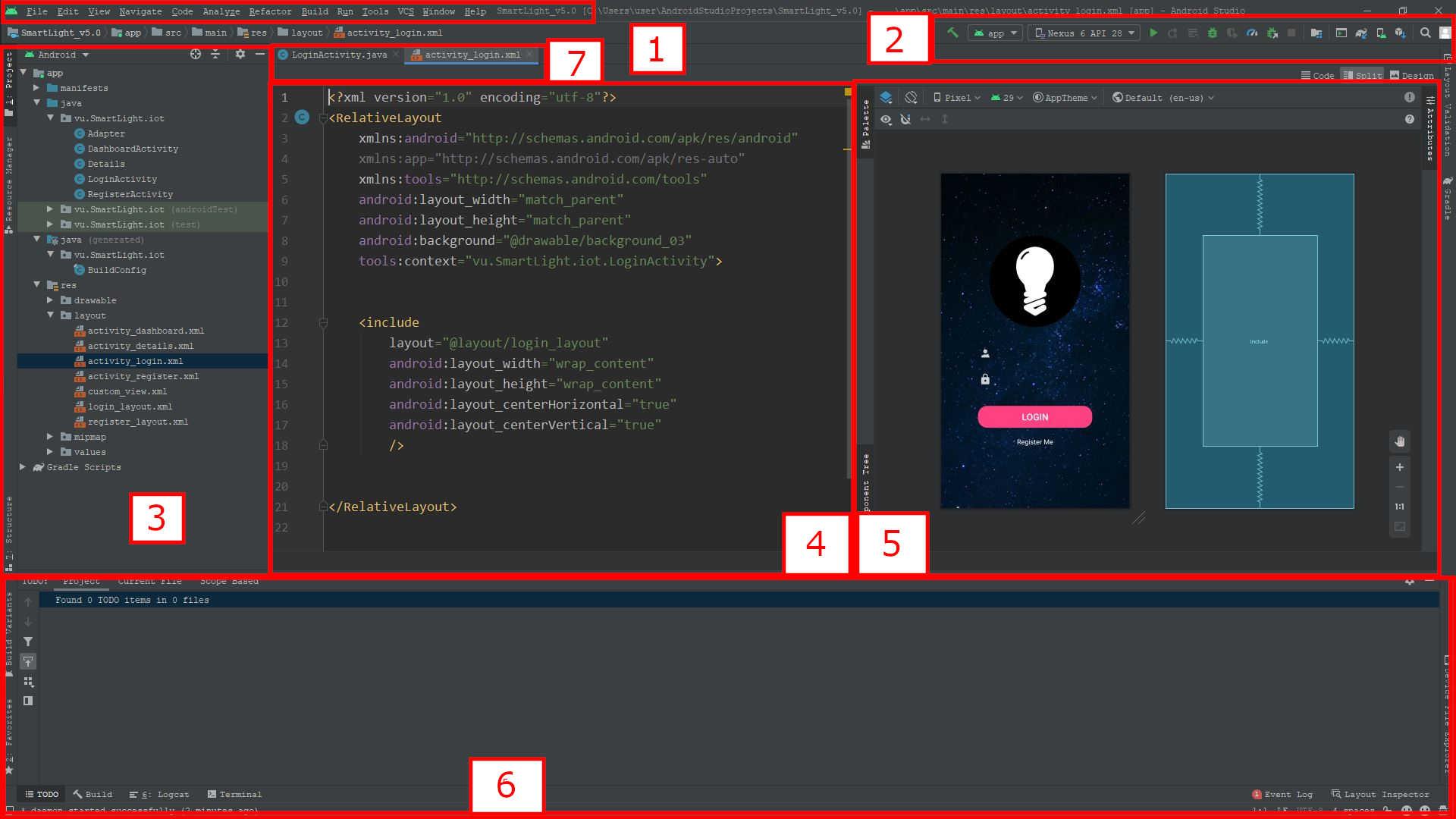


Fig 3.1.1

The interface in Fig 3.1.1 is divided into the following 7 parts:

1. File Task Bar - Accesses file controls and different features offered by Android studio that are not presented in the rest of the interface.
2. Debug/Run Tools - Options to run the application in an emulator, debug errors or run the application in stages with breakpoints inserted.
3. File layout manager - Displays the file hierarchy. This is where the user selects the file or component they choose to access
4. Code text editor - generally used for manually editing the .xml or .java files
5. GUI editor - mostly drag and drop, this is used for adding and aligning components such as buttons, text boxes, inputs etc
6. Error log/ TODO - Toggles between an error log that identifies errors in the build that need to be fixed or TODO which is a feature that breaks down the build into different steps that are removed when completed.
7. Working file list - toggles between the different files that the user is currently working on.

## 3.2 Languages Used

The two languages used in the construction and development of the Application are XML (eXtensible MarkUp Language) and Java.

### XML

XML10 is used similarly to HTML in web applications. It is the skeleton on which the GUI (Graphic User Interface) is built. It is used to build the layout of the different pages, it positions the different elements where the design decides that they want them. The power of XML is that there are no predefined tags unlike HTML. Which allows the developer more scope to create their own customised usage.

XML is predominantly used for layout is Android Studio but is also used for manifest files which define the components of the application or style files, similar to a CSS file used in web applications.

### JAVA

Java11 is one of the most widely used programming languages in the world. It is free to use, highly intuitive and very powerful. Java is used in the construction of Android Applications to make them dynamic, it increases the interaction between the user and the Application.

Features of Java include:

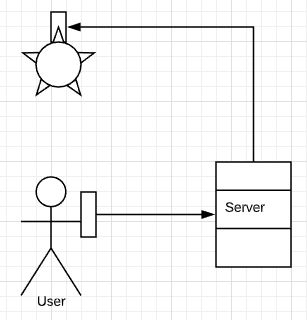
* Increased security - Java uses its own runtime environment meaning that it doesn’t rely on that of the operating system and doesn’t open or share its components in order to execute.
* Dynamic - It supports dynamic loading of classes, is interactive and creates an enhanced experience for users.
* High-Performance - Of the interpreted languages, Java is the fastest operating as it takes its cues from compiled languages like C++.
* Portability - due to using its own runtime environment Java can be implemented on any device and run just as efficiently.

## 3.3 Postman

Postman9 is a platform used to fetch data from an API (Application programming interface). It gets or posts data to the API.

The reason this is required is the controls for the Light Bulb Control Application are actually on the TP-Link server.

What happens is the user has the application on their device and uses the controls i.e. switches the bulb to the On mode. The application then sends a post request to the TP-Link API to switch the state of the bulb to on. It then returns this request to the bulb and switches it on.

Fig 3.4.1

The diagram in 3.4.1 shows the User with the handheld device. They send the request to the TP-Link server to turn the light on through a post request. The server then returns that request to the bulb, changing the state to that requested by the User.

## 

## 3.4 TPG API

Used in concert with Postmaster, the post requests are sent to the API via the application.

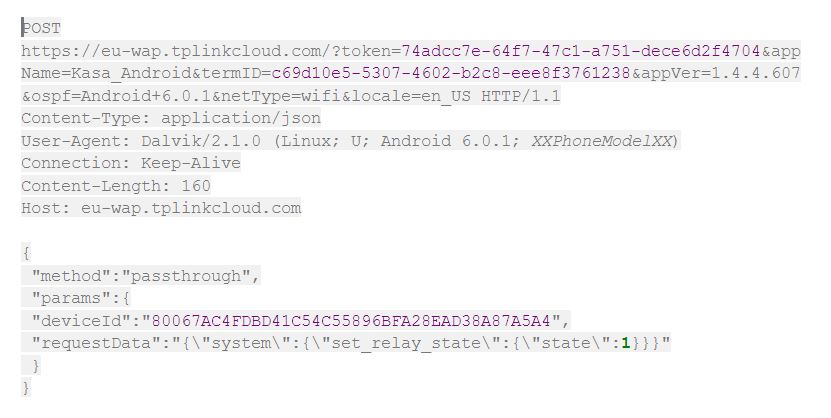


Fig 3.4.1

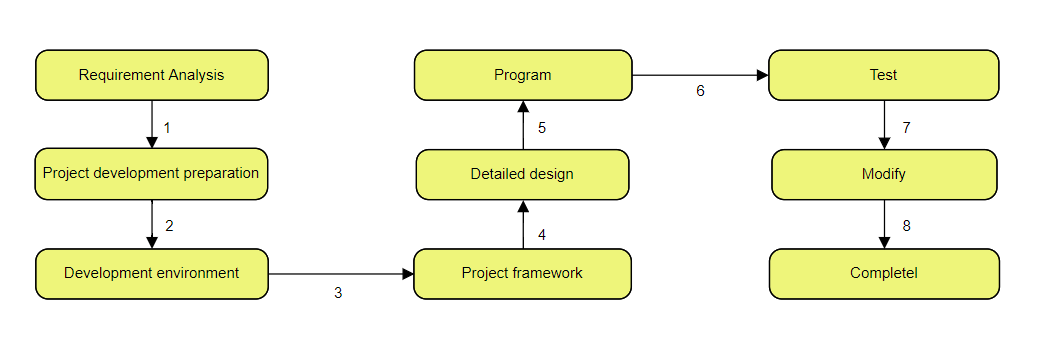
The app communicates by sending the code block in fig 3.4.1. The device details and the state that is requested to be changed is in the post request.

Fig 3.4.2

The code in fig 3.4.2 shows the code block returned by the API. Error code: 0 states that the request was returned without error. The response data is the command that alters the state of the bulb.

# F4 System Architecture

## 4.1 Development View



* Understand the concept of architecture in the construction of the Android Application and clarify the requirements, intentions and outcomes.
* Research Android functionality, what the requirements are to construct an application. The languages required and the ability to understand the workspace. In this case Java and XML, and Android Studio.
* Understand version control within Android. As the Android operating system has been updated, what is required to give the best functionality on the maximum number of devices. Does it only work for recent iterations? Is there support for it to be used on earlier versions?
* Identify what is required for the application. What is its function? What are the intentions of the client and stakeholders for it?
* As there is a User Interface, the User Experience is important. What can be done to fit in with the conventions of applications? It must be a clear layout that is easy to understand and interact with.
* Research the best methods for executing commands? Can it be done directly between the application and the device? Is a Wi-Fi connection required? What is the best means of executing commands?
* Work with the API to communicate with the device. Test the connectivity and the commands.

## 4.2 Logical View

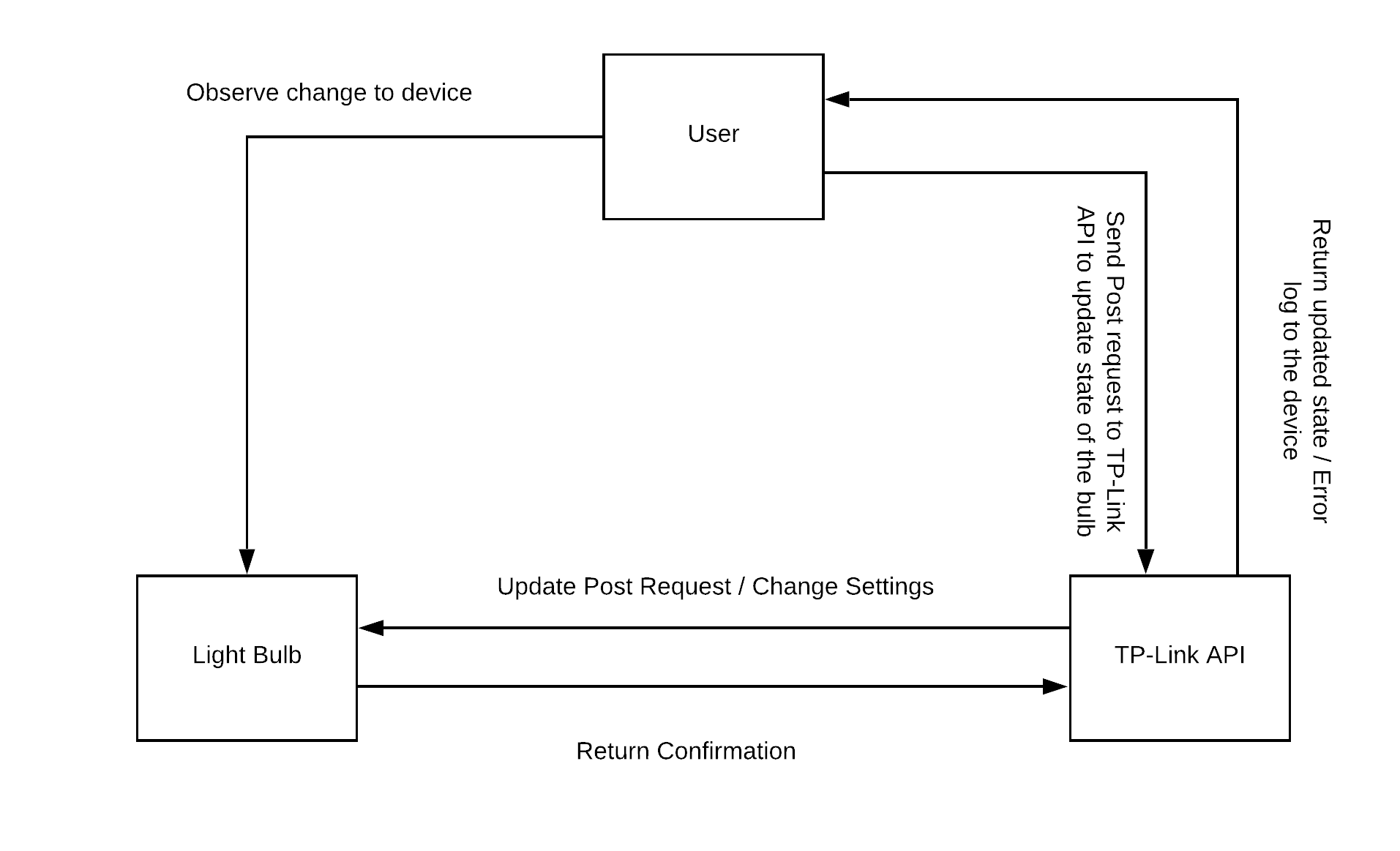
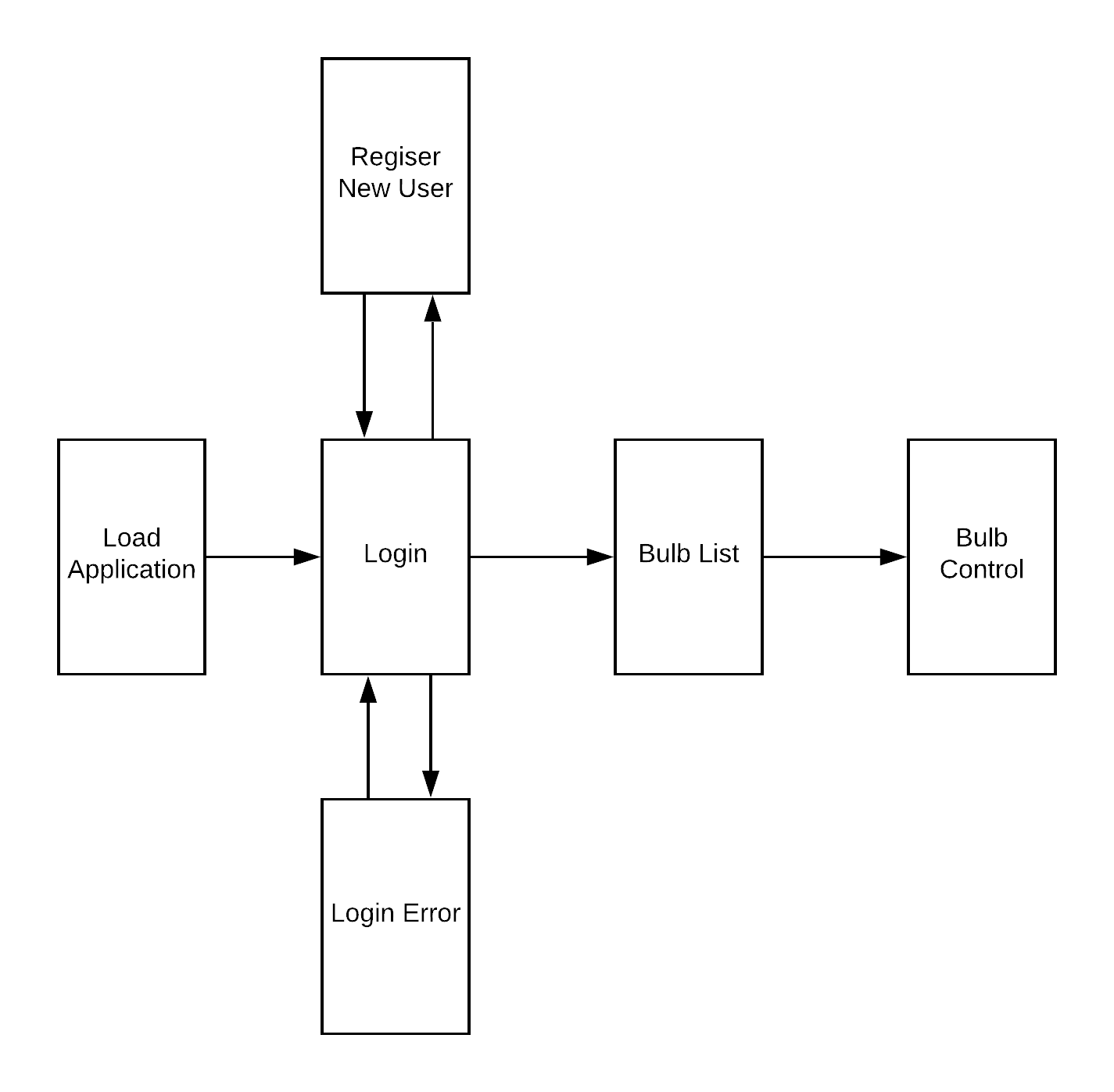
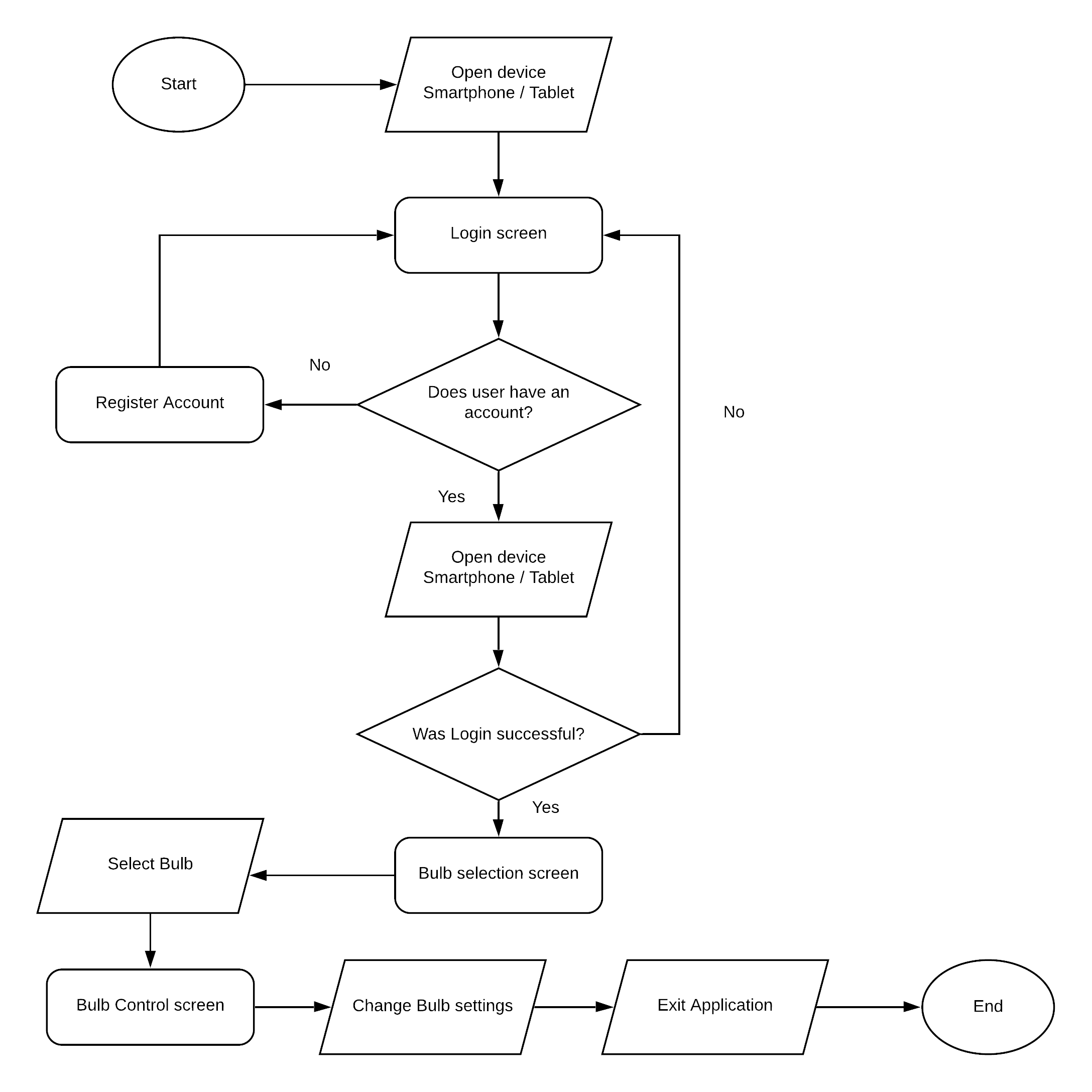
Fig 4.2.1

Fig 4.2.1 shows the relationship between the user interface, the light bulb and the TP-Link API. The user directly communicates with the interface which sends the request to the API which in turn communicates with the light bulb. The state changes in the light bulb and it sends back confirmation. The API will, in situations where the update of state change is successful, send confirmation back to the interface. In situations where there is an error the API will return an error code to the interface.

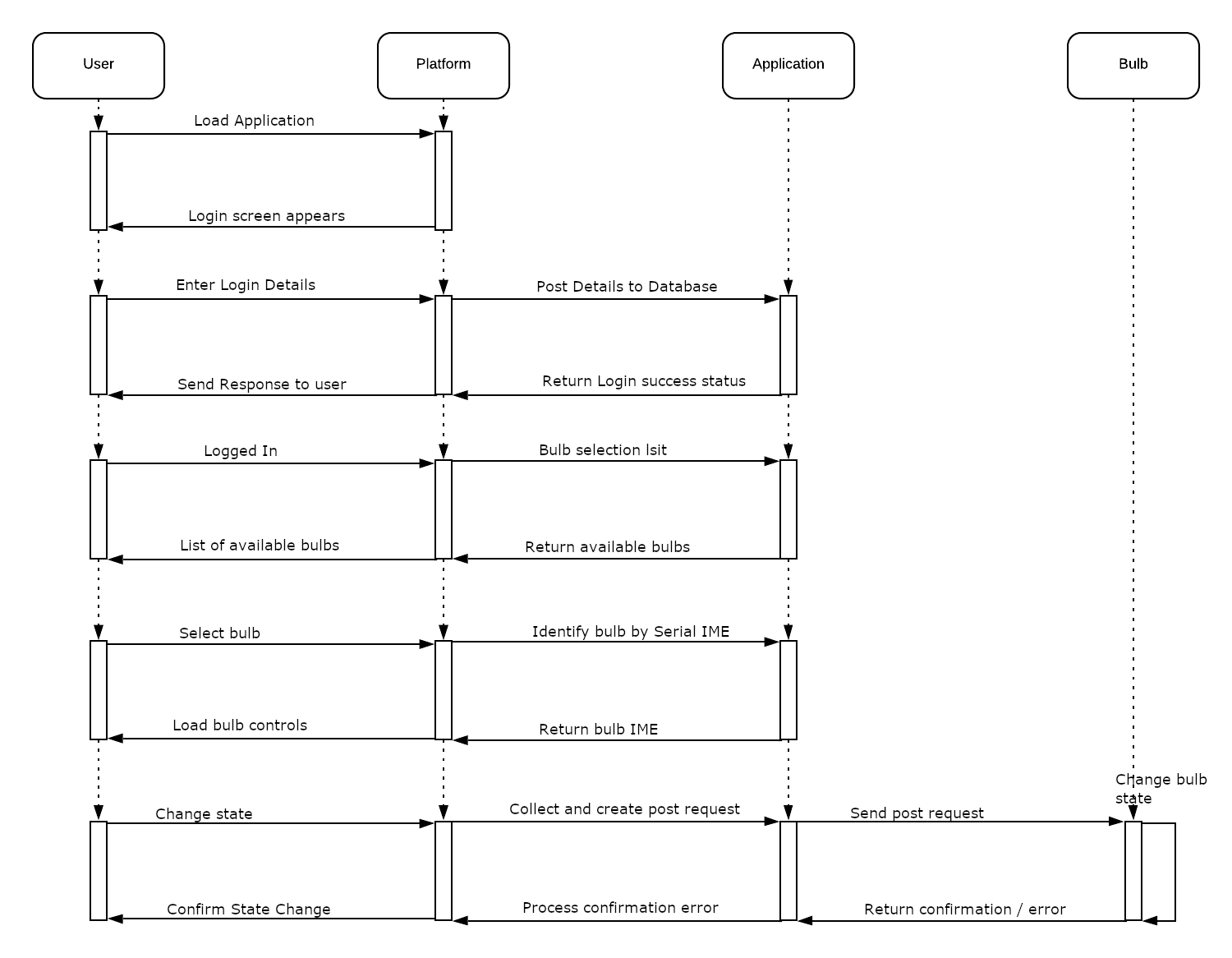
## 4.3 Physical View and Process View

Fig 4.3.1

Fig 4.3.2

* After the application is started, it first opens on the Login screen where it prompts the user to enter their registered email address and password. Beneath that is the Login button, to be executed when the Login details have been completed. The last input on the screen is the Register User button, this will be operated if the person using the application has not previously registered their details.
* If the user has previously registered an account, they will enter a username (email) and password and tap the Login button. The application will send a request to the database and compare the login details to existing accounts. If the details are correct then the user will move to the next screen.  
  If the user enters incorrect information that is not recognised by the database or is of an incorrect data type an error will be shown and the app will loop back to the Login page.
* If the user is operating the application for the first time they are directed to the registration screen where they enter their details and are redirected to the Login screen to Login to the Application.
* Once the Login process is completed, the next step for the user is to select the bulb from the list whose settings they wish to adjust.
* The final screen is the bulb controls. From here the user is able to adjust the settings of the light bulb they have selected.

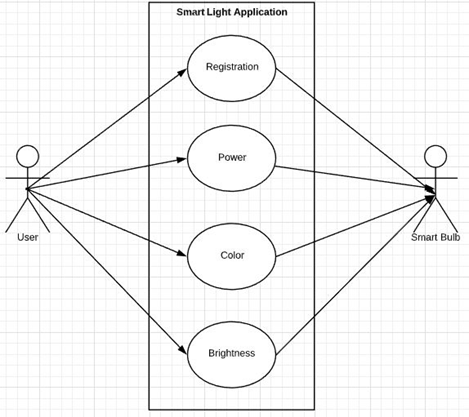
## 4.4 Sequence Diagram



* The first paragraph illustrates the users interaction with the handheld device and initiating the application to load.
* The second is the user accessing the application via the login process. The user may use an existing account or create a new one to register details. The details are posted to the database and compared for accessibility.
* The third paragraph is the actual functionality of the application once the users credentials have been confirmed. It checks on the available bulbs and identifies them before sending that back to the application in an itemised list from which the user may select their preferred bulb. Once the bulb is selected it moves to the bulb controller.
* The fourth paragraph shows the completed circuit. The user interacts with the controls to request a change of state to their preference. Once this is done, the application creates a post request in the background in real time. It sends the post request to the API which changes the state of the bulb. A confirmation or in the case of error a message is sent back to the application.

# F5 Linkages

## 5.1 Connection between user, application, and smart bulb



The user can directly turn on the bulb manually by turning the switch on/off. However if they want to turn it on remotely, then they have to use the Smart Light application to control it.

The user can create a new account and login to this system and connect to the bulb remotely and change brightness, color, temperature and saturation.

The Smart Light Application is the connection between the user and smart bulb for controlling remotely. It will automatically send a request to the server and automatically update the smart bulb settings in real time.

Use Case example:

|  |
| --- |
| Use case: Registration |
| ID: 1 |
| Description: Register a valid account for Smart Light App |
| Actor: User, testers, admins |
| Precondition: none |
| Flow of events:  1. Launch Smart Light Application  2. Click Register Me  3. Fill in name, email address and password  4. Click “Register” |
| Postcondition: A new account is created in database |
| Use case: Control Smart Bulb |
| ID: 1 |
| Description: Change colour of the bulb |
| Actor: User, testers, admins |
| Precondition: Connection is success between bulb and application |
| Flow of events:  1. Launch Smart Light Application  2. Fill in email and password  3. Click Login  4. Choose a specific Bulb  5. Turn power on  6. Choose a colour |
| Postcondition: The new setting will be changed automatically |

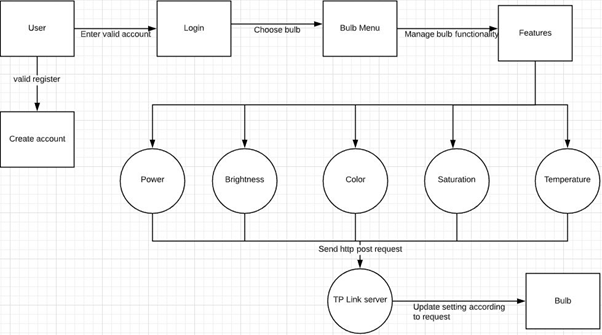
* Sub features are connected to TP link server. When a user chooses a specific feature, this will send an http post request to the TP link server, then the bulb will respond according to the request from the server and the bulb’s setting will be updated.
* A very useful thing to know that all these features can be re-done again with real time without any danger to other features.

# F6 Detail Design

## 6.1 Sub-feature Detail

* Although the Smart Light Application can provide main features like Power, rightness and Colour. It also gives users sub-features such as Temperature and Saturation.
* When a user wants to change the temperature of the bulb, they can just click on the slide bar and choose a specific level. This will be applied to Saturation as well. After the server receives requests, the bulb will be changed according to the adjustments. Check below chart for more information.

## 6.2 Data Flow Direction



* When the user successfully registers an account, they can use it to login to the Smart Light Application through the database (<https://databases.000webhost.com/>). After selecting a specific bulb then users can control its features.
* These features include main features (Brightness, Power, Colour) and sub features (Saturation, Temperature). The users only need to adjust/ choose the feature they would like to do by clicking on it.
* The Application system will send a http request to TP Link server and it automatically updates bulb settings based on the request then the bulb will be changed in seconds.

# F7 Test Result

## 7.1 Metrics

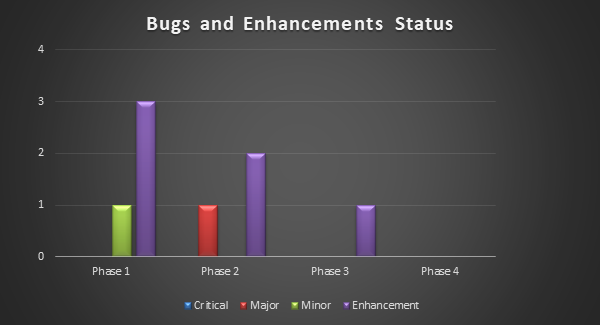
This will help understanding the tests execution results, the status of test cases and improvements.

* Number of tests cases planned and executed
* Number of tests cases Passed (OK, POK) and Failed (NR, NC, NOK)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test cases planned | Test cases executed | % OK | % NOK | %POK | %NR | %NC |
| 10 | 8 | 50 | 0 | 12.5 | 0 | 37.5 |

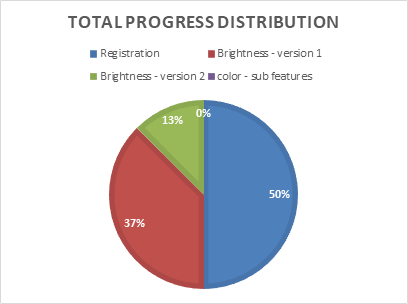
Number of Bugs and Enhancements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Phase 1** | **Phase 2** | **Phase 3** | **Phase 4** |
| **Critical** | 0 | 0 | 0 | 0 |
| **Major** | 0 | 1 | 0 | 0 |
| **Minor** | 1 | 0 | 0 | 0 |
| **Enhancement** | 3 | 2 | 1 | 0 |
| **Total** | 4 | 3 | 1 | 0 |



## 7.2 Test performance

* Phase 1: successfully launch the application. Test registration and login.
* Phase 2: Power on / off the light and change brightness (number input).
* Phase 3: Control light with slide bar (increased UI).
* Phase 4: Completed main features and sub features



## 7.3 Test environments

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Expected condition** | **Actual result** | **Comment** |
| Professionality | Poor experience testers, programmers | Outstanding testers, Programmers | First time we actually make an App to run a smart device and we actually success with our professionality. |
| Hardware and cost | Smart devices, computers, equipment | Unavailable hardware | We had to buy and use our own devices and equipment to test run the Application |
| Software | Control smart devices in lab room with smartlight app | Unable to control with our smartlight app | Cannot get DeviceID to get permission to use our SM App to control smart devices in the lab.  Must contact TPLINk support to remove the previous bound account to make further progress. |

# F8. The Difficulties Overcome

During the development phase, we faced numerous challenges that hindered

us from building the application. This section elaborates on the difficulties

that we encountered along with their respective solutions.

When the HTTP post request was sent to the TP-Link server for the token, it

was stated that the token was unable to be used as it had already expired.

The token acts as an authentication for the user. It was then discovered that

the URL (Uniform Resource Locator) of the HTTP post request was incorrect.

The URL was amended and a positive result was then achieved.

Another problem was that the token was refreshed each time the HTTP post request was sent to the TP-Link server. To resolve this issue, it was decided by the team that a single token of an account would be used throughout the project.

The Kasa smart light bulb had to be bound to a TP-Link account to add the

device in the application. This process could not be accomplished without

using the Kasa application, thereby leading to the team’s decision to use the

Application.

Another difficulty was that the team was unable to find the specific code that would enable us to change the smart bulb’s color. To overcome this issue, the team decided to implement the HSL (hue, saturation, lightness) color system. A major challenge was that in the HSL color system, the lightness value could not be adjusted as it was configured by default. This challenge was overcome by the use of the “SeekBar” function.

Another problem was that the code used to create the color wheel was outdated and thus, it could not be implemented in the project. To resolve this, the team created its own values and used the library function to change the specific hue value in order to be able to change the color of the smart bulb.

Finally, the application could not be used to control the IoT bulbs at the University because the “Remote Control” feature of the Kasa application could not be enabled and hence, the device ID of the bulbs could not be obtained. When an attempt to establish remote control was made, an error message stating “device bound to other account” was received. The reason behind this issue is that the bulbs have already been bound to another Kasa account. For security reasons, the remote control feature is exclusive to a unique user and thus, the devices must be deleted from the user’s account to enable someone else to utilize the feature. It is to be noted that factory resetting of the smart bulbs could not remove the account that was previously bound to the bulbs.

# F9. Summary

Originally the intention of this project was titled The Internet of Things: Control Lights Available in room D633. This was to involve the controlling of a lighting rig containing multiple LED light bulbs that had varied adjustable settings that are able to be controlled by a device such as a tablet or smartphone. The objective was to create a custom application that could access, identify and control the devices over the Victoria University Wi-Fi network.

The bulbs are made by TP-Link and use their native control application called Kasa. With the Kasa Application the user is able to control one or many bulbs.

Settings that are able to be manipulated via the application include:

* Turning the bulb on and off.
* Using the HSL colour spectrum to alter the colour emitted.
* Brighten or dimming the light
* Increasing or reducing the colour saturation, from a sharp colour to a softer tone.
* Adjusting the temperature to a warmer or cooler light tone, warmer being similar to the yellow hue emitted by traditional light globes or the cool white globes available as well.

Unfortunately due to mitigating factors including COVID-19 and the inaccessibility of Room D663 this was not a possibility and a change was required.

It was decided that the project would be changed to creating an application that would control a single TP-Link bulb that was ordered from Amazon.com. The bulb in question was the TP-LINK Kasa E27 10W RGBW Dimmable Smart Light. The team reviewed the different methods for creating the application and as all members owned smartphones with an Android operating system and that the majority of smartphones globally are also Android (72.6%) that it would be the best choice for the application.

Once the decision had been made, the next step was to determine the best way to build the application. The best method is to use an Integrated Development Environment (IDE). An IDE is required when the application you are constructing has its own runtime environment. As Android Applications are built using Java, and Java has its own runtime environment this was the best method. Other alternatives considered were Netbeans and Eclipse, which are Java IDEs, but they do not support EXtensible Markup Language (XML) which is the other programming language used in the construction of Android Applications.

Other things, such as the ability to use drag and drop functionality to assist in the construction of the Graphic User Interface (GUI) and the ability to use an emulator to conduct real time testing without requiring an actual handset were taken into account.

# F10 Conclusion

The team was assigned to programme a IoT Application described in [F1.2 – Project Description].

The project was divided into 4 versions. For each version we specified many features of implementation. Each member was given one of roles in the project. Had the trials and experiments been a complete success, all versions would have run correctly for specific features.

However our experiment indeed shows that the early state is very poor and struggling with limited time of 4 weeks, unavailable materials and the pandemic. But the limited capacity of our minds make us more efficient when our project depends on a small amount of knowledge and difficulties. So we decided to make a simple but efficient application where we can smartly invest our man power to the project.

We all believe this experience is truly valuable to us and has provided us with good insights into IoT application production. We recognised the challenges and difficulties of building Smart Light Application and surely we had learnt new technologies and new abilities to ourselves. Although Smart Light could be improved with some more features but we could say our application’s credibility of content does not suffer due to these new features which will be updated in the next opportunities.

# F11. References

[1] <https://community.tp-link.com/us/home/kb/detail/163>

(How to connect tp-link Smart Bulb to my home network via Kasa? | TP-Link Australia, 2020)

[2] <https://www.support.com/how-to/how-to-use-tp-link-smart-bulb-with-kasa-smart-app-12865>

(How to Use TP-Link Smart Bulb with Kasa Smart App | Support.com Tech Pro Team)

[3] <https://www.tp-link.com/us/support/faq/1718/>

(TP-Link Smart Bulb (Wi-Fi) | TP-Link FAQ, 2018)

[4] <http://www.freshyhome.com/tp-link-smart-bulb-troubleshooting/>

(TP Link Smart Bulb Troubleshooting | Author, Date Unknown)

[5] <https://www.kasasmart.com/us/products/smart-lighting/kasa-smart-light-bulb-multicolor-kl130>

(Kasa Smart Light Bulb, Multicolor Add Color to Your Life | KL130 | Author, Date Unknown)

[6] <https://guides.codepath.com/android/Architecture-of-Android-Apps#architectural-patterns>

(Architecture of Android Apps | N Sequena, 2018)

[7] https://developer.android.com/jetpack/docs/guide

(Guide to app architecture | Autor, Date Unknown)

[8] <https://developer.android.com/studio/>

(Android Studio Download | Author, Date Unknown)

[9] <https://www.postman.com/>

(Postman The Collaboration Platform for API Development | Author, Date Unknown)

[10] <https://www.w3schools.com/xml/>

(XML Tutorial | w3schools, ©1999-2020)

[11] <https://www.java.com/en/>

(Java + You. Download Today | Oracle, Date Unknown)