**Software Design Description Report**

**For**

**<Smart Water Level Detection System>**

**Software Engineering Project**

**Universiti Teknologi Malaysia**

**The members for our group are:**

* Farah Abdul Muttalib
* Zulfadli Kamaruzzaman
* Syahmi Khairudin
* Amirul Fahmi Abdullah
* Shahnizam Abd Saini
* Zulzhafri Zuki
* Wan Zahiruddin Wan Abd Kadir
* Atif Abd Razak

1. **Introduction**

This document comprehends the software design descriptions for Smart Water Level Detection System. This document is prepared according to the “IEEE Standard for Information Technology – Systems Design – Software Design Descriptions – IEEE 1016 – 2009”.

This document provides the details of how the Smart Water Level Detection software built and develop. The details are represented by graphical notation such as use case models, sequence diagrams, class diagram, object behavior models and other supporting design information.

* 1. **Project Scope**

Smart Water Level Detection System provide easy and efficient way to monitor and analysis the water level at places of these device being installed. The data can be monitored at either on any web browsers by accessing the web page or by using the smartphone apps. The water levels that being recorded by the device can be classified to several classes that represent different situation. The data also can be used to predict incoming flood or dam malfunction that will alert users so that safety measures can be taken.

Compared to the traditional way, users usually monitored by using estimation or on-site monitoring. These traditional data sometimes not accurate and wrong prediction might being made. The other advantage of using this system is that user can monitor and analyze the data from the device real-time anywhere there is internet connection. The device also will determine the critical water level and send alert to users either on the web page or on the smartphones apps. The data that being sent to the database (web page) will be stored and can be used to generate graph or data tabulation to prevent any related catastrophe. Another feature of the system is that the data inside the database can be downloaded into the computer for storage/analysis purpose.

* 1. **Purpose**

The purpose of this documentation is to specify the objectives, aim and requirement for the Smart Water Level Detection System. In this documentation, it also provide the introduction and some of the analysis that can be carried out by the Smart Water Level Detection System. This project implement the concept of IoT that uses the benefits of internet.

Smart Water Level Detection System consist of 3 main parts that’s are web monitoring (Django), smartphone application and Smart Water Level Detection System Device (Raspberry Pi.and water sensor).

* Smart Water Level Detection System Device – sensor and controller that bused to measure the water level at any environment it being installed
* Web monitoring – act as the database and monitoring the data that being send by the device. It also functions to create all the analysis for client to use for further actions. The web page can be access via any browser through internet. <link to the page>
* Smartphone apps – act as monitoring application that use data from the database to provide latest data/reading from the device. <name for the apps>

Smart Water Level Detection System Device consist of controller and sensors that will record the manipulated data. The controller that being used is Raspberry Pi and sensor that being used is water sensor. Raspberry Pi being programmed by using Python language. The controller contained build in Wi-Fi modules that enable the controller to connect to the internet. The device offers the function to record all the data and determine the output. The data is then being sent to the web server (Django) via internet. User can install this device at any place that need to be measured the water level. Some of the most suitable place to install the device are places with heavy rain (flood detection), dam, beaches and etc.

Web monitoring is used as the database for the project and also used to monitor/analysis the data received by the device. It receive the data from the device periodically and stored it inside its database. The data also being used to generate graph and provides information and alerts user if the water level reach critical condition. Users can monitor the data real-time anywhere by accessing the web page. Multi device can be integrated within the same web page that enable users to install multiple device at multiple places and monitor all the data at single web page.

Smartphone apps gives users the same advantage to monitor the data by using their smartphones. The apps can gives the latest data that being requested from the database and visualize it on the smartphones. It becomes more convenient for users that used their smartphones daily.

All the component mentioned above used the concept of IoT that enable them to send/receive data from each other using the internet. The data stored at the database also can be downloaded into the computer for further analysis.

* 1. **Intended Audience and Reading Suggestion**

This document is intended for any individual user, developer, tester, project manager or documentation writer that needs to understand the basic system architecture and its specifications. Here are the potential uses for each one of the reader types:

* Developer: The developer who wants to read, change, modify or add new requirements into the existing program, must firstly consult this document and update the requirements with appropriate manner so as to not destroy the actual meaning of them and pass the information correctly to the next phases of the development process.
* User: The user of this program reviews the diagrams and the specifications presented in this document and determines if the software has all the suitable requirements and if the software developer has implemented all of them.
* Tester: The tester needs this document to validate that the initial requirements of this programs actually corresponds to the executable program correctly.

For each one of the reader types to better understand this document, here is a suggestion of the chapters to read in this document:

Developer:(1.1 ,1.3 ,2.2 ,2.3 ,2.5 ,2.7 ,3 ,4 ,5 and rest)

User :( 1, 2.1, 2.2, 2.3, 4.1, 5.5)

Tester:(1.1 ,2.1 ,2.4 ,2.5 ,2.7 ,3 ,4 ,5)

* 1. **References**

This citation is used as a model of reference:

[IEEE Std 830-1998]

**1.5 Definition**

|  |  |
| --- | --- |
| Term | Definition |
| User | Someone who interacts with the application |
| Developer | Person of group that develop the application |
| Database | Collection of all the information monitored by this system |
| UML Diagram | Unified Modeling Language is a graphical visualization language. UML is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system. It consists of a series of symbols and connectors that can be used to create process diagrams and is often used to model computer programs and workflows. |
| Software Requirements Specification (SRS) | A complete description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. |
| Software Design Description (SDD) | Software Design Description which is the complete description of the design of system. |
| IEEE | Institute of Electrical and Electronics Engineers |

1. Conceptual Model

5. Design Viewpoints

5.1 Introduction

In this chapter, the viewpoints of the Water Level Detection System is explained in detail. During this section, UML diagram will be used to increase understandability.

5.2 Context Viewpoints

5.2.1 Design Concern

The main service category than concerning to our system is user. Users are people who wll use the application. They are going to interact with different clients of the final.

5.2.2 Design Elements

Design entities: Design entities are user and actions of the system. Below diagram, use case diagram with all functions can be seen.

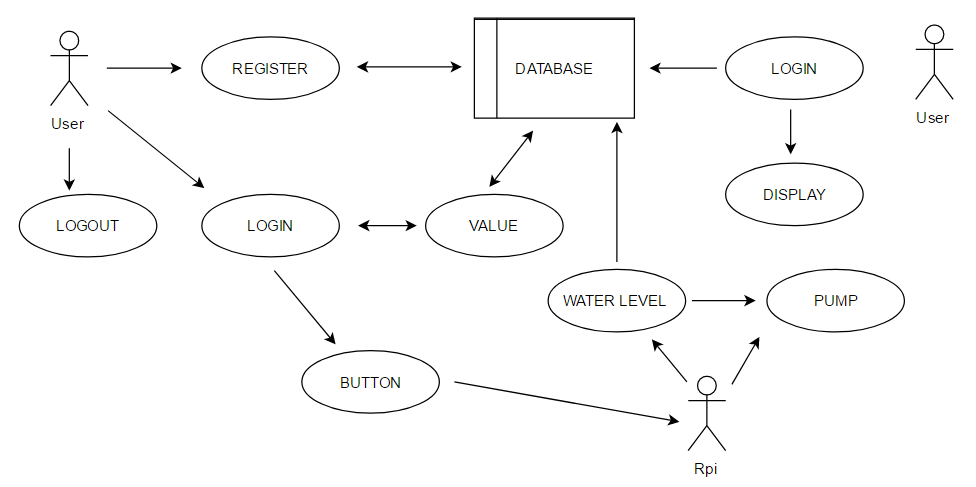
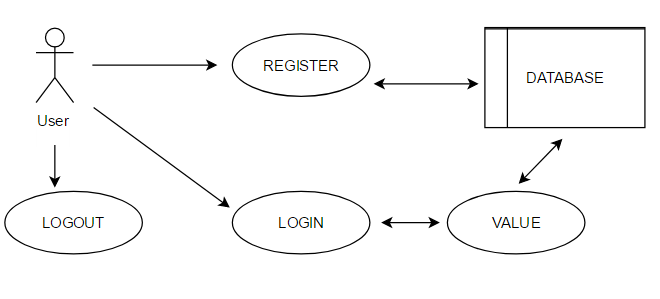


Figure : use case diagram with all functions

5.2.2.1 Android Application



5.2.2.1.1 Login

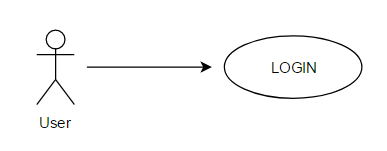


Figure : use case diagram login to android apps

|  |  |
| --- | --- |
| Use Case Number | 1 |
| Use Case | Login |
| Summary | User can login to the android application. |
| Actor | User |
| Trigger | Login Button |
| Primary Scenario | In order to login the application, user must registered to the system as describe in pre-condition. The account information must be same as registration information. After that, user can login to the system with filling required fields and press login button. |
| Exceptional Scenario | 1. Wrong username or password 2. Not register |
| Pre-Condition | User must register |
| Post- condition | User will enter the system and he/she can use it |
| Assumptions | User must be connected to the internet. |

5.2.2.1.2 Register

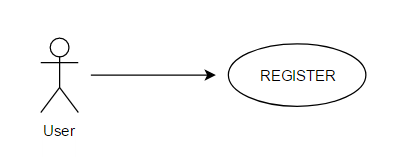


Figure : use case diagram login to android apps

|  |  |
| --- | --- |
| Use Case Number | 2 |
| Use Case | Register |
| Summary | User need to register to use this android application |
| Actor | User |
| Trigger | Register Button |
| Primary Scenario | After download this android application, user must register first in order to use it. In the login page, there will be register button. The register button will redirects user to the registration page. User need to filling the required field with information. After finish filling the information, user can press register button to register. |
| Exceptional Scenario | 1. Username has already taken 2. Password does not match |
| Pre-Condition | User must having e-mail address |
| Post- condition | After registering to the system, user can use the android application. |
| Assumptions | User must be connected to the internet and having valid e-mail address. |

5.2.2.1.3 Logout

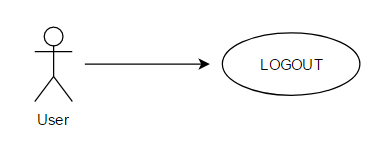


Figure : use case diagram login to android apps

|  |  |
| --- | --- |
| Use Case Number | 3 |
| Use Case | Logout |
| Summary | User can logout from the system. |
| Actor | User |
| Trigger | Logout Button |
| Primary Scenario | After login, user can logout at any time. |
| Exceptional Scenario | 1. None |
| Pre-Condition | Login |
| Post- condition | User will redirected to the login page. |
| Assumptions | User has already logged from system. |

5.2.2.1.4 Value

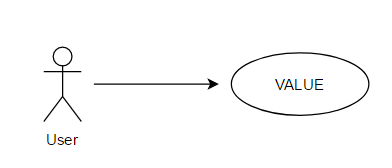


Figure : use case diagram login to android apps

|  |  |
| --- | --- |
| Use Case Number | 4 |
| Use Case | Value |
| Summary | User can request the data from server database. |
| Actor | User |
| Trigger | Value Button |
| Primary Scenario |  |
| Exceptional Scenario | 1. No database 2. Database breakdown |
| Pre-Condition | User must login in to system |
| Post- condition | User get the water level from database. |
| Assumptions | The water level. |

**3.0** System Features

Using the actor list we begin the analysis of the client-Rpi features that takes place in Section A and then for each modules that is being reported

**Section A**

**Raspberry Pi**

**3.1 System Feature Execute SensorModule::REQ1**

3.1.1 Description and Priority

A User executes the SensorModule and the startup setup script.

Priority level=High

3.1.2 Stimulus/Response Sequences

Preconditions: None

1. User execute auto.csh on Rpi terminal.

2. SensorModule setup script shows up.

Postconditions: auto.csh will run all the require setup script to run SensorModule

3.1.3 Functional Requirements

NA

**3.2 System Feature Connect to a Database (online mode)::REQ2**

3.2.1 Description and Priority

User sets Database name and port and connects to the server.

Priority level=High

3.2.2 Stimulus/Response Sequences

Preconditions: User access makeNet.py. User knows Database name and port. Rpi is online.

1. User replace variable <database> to the desired database name.

2. User replace variable <port number> to desired port.

Postconditions: User is connected to the database after auto.csh run. The database information is displayed.

Abnormal Paths: If the servers name and port is incorrect error will be print out to the terminal to inform the User that which one of the two is incorrect. auto.csh will halt

3.2.3 Functional Requirements

NA

**3.3 System Feature Setting Sensor Config::REQ3**

3.3.1 Description and Priority

User set the sensor configuration.

Priority level=High

3.3.2 Stimulus/Response Sequences

Preconditions: User access makeConfig.py. User knows detail of sensor mode.

1. User replace variable <mode> to desired mode.

2. User replace variable <unit> to change the SI unit of sensor reading.

Postconditions: Sensor mode change successfully and changes will appear on database upon running auto.csh.

Abnormal Path: If the mode and unit is incorrect error will be print out to the terminal to inform the User that which one of the two is incorrect. auto.csh will halt.

3.3.3 Functional Requirements

NA

**3.4 System Feature Connect to Devices::REQ4**

3.4.1 Description and Priority

User sets devices MAC address or IP to connect with RPI.

Priority level=High

3.4.2 Stimulus/Response Sequences

Preconditions: User access makeDevice.py. User knows MAC address or IP of the device. Device is online.

1. User replace variable <macaddress> to the device MAC address.

2. User replace variable <ip> to the device ip address.

Postconditions: User is connected to the device after auto.csh run. The device information is displayed.

Abnormal Paths: If the device MAC address or IP address is not available, error will be print out to the terminal to inform the User that which one of the two is incorrect. auto.csh will halt

3.4.3 Functional Requirements

NA

**3.5 System Feature Next Run Automation::REQ5**

3.5.1 Description and Priority

User can make auto.csh to run automatically on the next RPI boot.

Priority level=High

3.3.2 Stimulus/Response Sequences

Preconditions: User access auto.csh. User confirm that current configuration is the same on next run.

1. User execute auto.csh with variable <1/2>

Note: 1 is automate on the next boot

2. Do not automate

Postconditions: On the next Rpi boot, all the setup script will run by itself. If there is an error, a message will be display during Rpi boot.

3.5.3 Functional Requirements

NA

**DJANGO**

**3.6 System Feature Connect to a server (online)::REQ6**

3.6.1 Description and Priority

The user sets Server name based on the place that user want to monitor the water level.

Priority level=low

* + 1. Stimulus/Response Sequences

Preconditions: User knows servers name and the place to monitor.

1. User enters Servers name.

2. User enters Place Name.

3. User elects to Connect to the server.

4. User elects to view the water level information.

Post conditions: User is connected to the server. The water level information is displayed.

Abnormal Paths: If the servers name and place is incorrect SmartCity informs the user that which one of the two is incorrect.

* + 1. Functional Requirements

Specializes: NA

Specialized by: NA

**3.7 System Feature Connect to a server (offline)::REQ7**

3.7.1 Description and Priority

Server will get the data from rpie and store data inside Django.

Priority level=low

* + 1. Stimulus/Response Sequences

Preconditions: Rpie send data to server.

1. Rpie search server IP

2. Server receive data from rpie.

3. Server store data based on rpie place.

Post conditions: The information store in server and ready to display in SmartCity website.

Abnormal Paths: If the server no receive data from rpie, server will warning to technical engineer.

* + 1. Functional Requirements

Specializes: NA

Specialized by: NA

**3.8 System Feature User enters Place::REQ8**

3.8.1 Description and Priority

The user is obliged to choose a place after connected to the server and clicked next button.

Priority level=medium

* + 1. Stimulus/Response Sequences

Preconditions: Users clicked the next button from SmartCity startup screen.

1. User enters the place.

2. User clicks the start button.

Post condition: screen will show the information of water level based on place registered.

Abnormal Paths: if the place not registered in server, the screen will displayed “NO DATA” and user need to reenter the corrected place name.

* + 1. Functional Requirements

Specializes: NA

Specialized by: NA

**3.9 System Feature Display Graph::REQ9**

3.9.1 Description and Priority

The user choose a graph button and graph are displayed.

Priority level=low

* + 1. Stimulus/Response Sequences

Preconditions: Users clicked the graph button from screen.

1. User click the graph button.

2. Type of graph will appears.

3. User need to choose type of graph.

Post condition: screen will show the information with graph that user choose.

* + 1. Functional Requirements

Specializes: NA

Specialized by: NA

**3.10 System Feature View Information::REQ10**

3.10.1 Description and Priority

The result information are being forwarded to a printer

Priority level=medium

* + 1. Stimulus/Response Sequences

Preconditions: Users viewing the information of water level.

1. User enter the print button.

Post condition: Test information is printed using the default printer settings.

* 1. 3 Functional Requirements

Specializes: NA

Specialized by: NA

**Smartphone Apps**

**3.11 System feature install Smart Water Level Detection system.apk::REQ11**

3.11.1 Description and priority

User install the Smart Water Level Detection system.apk and the system wizard start up client screen appears

Priority level = high

3.11.2 Stimulus /response sequences

Precondition: User must have mobile internet connection

User already download Smart Water Level Detection system.apk

1. User mobile phone run on android 5.0 and above
2. User double click the Smart Water Level Detection system.apk
3. System wizard start up screen appear
4. User read and accept the term and condition
5. User close the system wizard

3.11.3 Functional requirement

  NA

**3.12 System feature Connect to a server (online Mode)::REQ12**

3.12.1 Description and Priority

User connect to Django server and allow data reading

3.12.2 Stimulus /response sequences

Preconditions: User must have mobile internet connection

1. User click on the installed application icon on the mobile home screen
2. System application user interface screen appear
3. "Connect to server" and "sign up" button appear
4. User click on the "connect to server" button
5. Include REQ4
6. System application home screen appear

Postconditions: None

**3.13 System feature Sign up (online Mode)::REQ13**

3.13.1 Description and Priority

User sign up in order to create the account before login to the system application

Priority level: medium

3.13.2 Stimulus /response sequences

Preconditions: User must have mobile internet connection

1. User click on "Sign up" button
2. User enter the email address
3. User enter password
4. User confirm the password
5. User click " Done" button
6. Connect to server user interface appear

Postconditions: None

**3.14 System feature login to the system (online Mode)::REQ14**

3.4.1 Description and Priority

User login to the system and get permission to access Django’s database

3.4.2 Stimulus /response sequences

Preconditions: None

1. User click "connect to server" button
2. User enter the email and password on the login section
3. User click "login" button
4. System application home screen appear

Postconditions: None

**3.15 System feature request data (online Mode)::REQ15**

3.5.1 Description and Priority

User request data from Django’s database and display on the application screen

3.5.2 Stimulus /response sequences

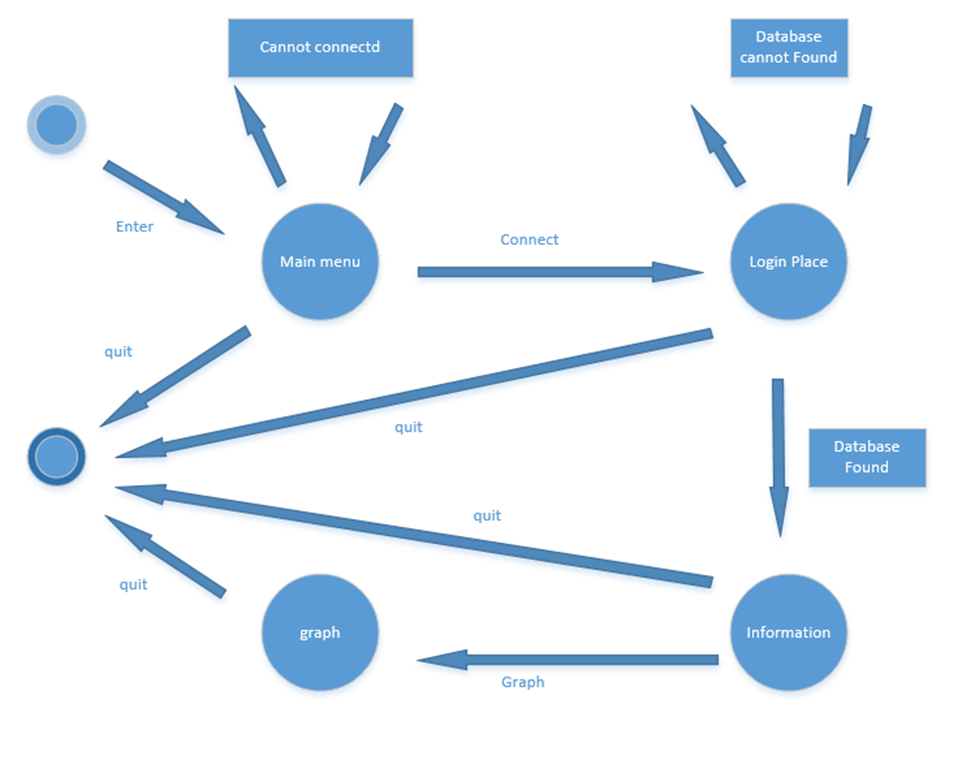
Preconditions: User already connect to Django’s server

1. User select " Location for System water level Detection "
   1. Location A
   2. Location B
   3. Location C

2. User select “Time detection"

1. User click " confirm' button
2. System application request data from server
3. Data from server appear on the user screen

**Section B**



State machine diagram of Smart Water Level Detection System