Bhuwan Karki

The Design of Short Message Server with Linux Machine

Information Technology

2017
ACKNOWLEDGEMENTS

First of all, I would like to thank my thesis advisor Jani Ahvonen from the department of Information technology at Vaasa university of Applied Sciences for giving me the opportunity to work on the thesis. His advice, guidance and insights offered throughout the thesis period has helped me in solving various obstacles. He was always available whenever I ran into a various problem or had a question about my research or writing.

I would also like to thank the experts who have helped me with the project. Mainly the “stack overflow” which has helped me with the various query and problem that I was unable to solve. Python and Django documentation was of a great help for most of the functions. Without the tutorials and explanation, it would not have been possible.

I would also like to thank my friends for accepting nothing less than excellence from me. Last but not the least, I would like to thank my family for supporting me spiritually throughout writing this thesis.

Vaasa 10.11.2017
ABSTRACT

The main objective of this thesis was to design a short message server with Linux machine for a Vakri school and Lyhty day care to send information to parents about different events occurring in a school and day care. A web based application was designed for the purpose of sending SMS.

To send a message from a mobile phone to each and every parent was not a favourable solution and as well as it was a time consuming method. To avoid a complication and to do it efficiently the web based SMS server was designed to achieve the required goal.

The Huawei mobile broadband E398 and Ubuntu, a Linux based operating system was used for the implementation of the SMS server. The whole project is done in Python and Django framework.

A simple web application was designed to search the data from the database. A database stores all the records of the students and the class group. On basis of their name and their class group the user can search the list of students and the group. This project will help the school to send bulk messages to the parents.

Keywords Python, Django, SMS gateway
CONTENTS

ACKNOWLEDGEMENTS
ABSTRACT

1 Introduction: ........................................................................................................ 16

1.1 Background .................................................................................................... 16
1.2 Motivation ..................................................................................................... 16
1.3 Objective ....................................................................................................... 17

2 Relevant Technologies ..................................................................................... 18

2.1 Global System for Mobile Communications ........................................... 18
  2.1.1 Mobile Station: ........................................................................................ 19
  2.1.2 Base Stations Subsystem: ........................................................................ 19
  2.1.3 Network and Switching Subsystem: ...................................................... 19

2.2 Short Message Service .............................................................................. 20

2.3 Short Message Service Center: ................................................................. 21

2.4 SMS Gateway: ............................................................................................ 21

2.5 Python .......................................................................................................... 23

2.6 Django .......................................................................................................... 24

2.7 NGINX .......................................................................................................... 24

2.8 Gunicorn ...................................................................................................... 24

2.9 Supervisor .................................................................................................... 24

2.10 MySQL ........................................................................................................ 25
2.11 Huawei E398 ............................................................................................................. 25

3 Application Description ......................................................................................... 27

3.1 General Description .................................................................................................. 27

3.2 Quality Function Deployment .................................................................................. 28

3.3 Functional Specification(FS) .................................................................................. 29

3.4 Use Case Diagram ................................................................................................... 33

3.4.1 User Use Case Diagram ....................................................................................... 33

3.4.2 Administrator Use Case Diagram ......................................................................... 34

3.5 Model Dependency Diagram ................................................................................... 36

3.6 Sequence Diagram .................................................................................................. 37

3.6.1 Administrator Login Sequence Diagram ............................................................... 37

3.6.2 Admin Add a User Sequence Diagram ................................................................. 38

3.6.3 Admin Delete User Sequence Diagram ............................................................... 39

3.6.4 Admin Add Student/Group: ............................................................................... 40

3.6.5 Admin Delete Student/Group .............................................................................. 41

3.6.6 User Search the Student /Group ......................................................................... 41

3.6.7 User /Admin Send the SMS ............................................................................. 43

3.7 Component Diagram ................................................................................................ 44

4 Database: .................................................................................................................. 45

4.1 Design of Database ................................................................................................ 45

5 Graphical user interface design .................................................................................. 48
6.3.2 Main URL Configuration ................................................................. 61
6.3.3 Login/Logout URL Configuration ...................................................... 61

6.4 Python Modules ................................................................................. 61

6.5 View ................................................................................................. 62
6.5.1 Student Search Result ................................................................. 62
6.5.2 Search Student ................................................................................ 62
6.5.3 Group Search Result ...................................................................... 63
6.5.4 Group Search .................................................................................. 63
6.5.5 Group Member List ......................................................................... 63
6.5.6 Group Display ................................................................................ 64
6.5.7 List Phone Number to Send SMS .................................................. 64
6.5.8 Sending SMS .................................................................................. 64

6.6 SMS with AT Command ...................................................................... 65

6.7 Models ............................................................................................... 65
6.7.1 Student Models ............................................................................... 66
6.7.2 Group Models ................................................................................ 66

6.8 Form .................................................................................................... 66
6.8.1 Student Search Form ..................................................................... 66
6.8.2 Group Search Form ....................................................................... 67

7 Deployment .......................................................................................... 68
7.1 Package Installation ........................................................................... 68
LIST OF FIGURES

Figure 1: GSM Architecture .............................................................................18

Figure 2: Text message application connects to SMSCs through SMS gateway 22

Figure 3: Application connects GSM/GPRS modems through an SMS gateway. ..................................................................................................................23

Figure 4: Huawei E398 .....................................................................................26

Figure 5: User Use Case Diagram ...................................................................34

Figure 6: Admin Use Case Diagram .................................................................35

Figure 7: Directed Orthogonal Class Diagram .................................................36

Figure 8: Admin Login Sequence Diagram .......................................................37

Figure 9: Admin Adding a User .........................................................................38

Figure 10: Admin Delete a User .........................................................................39

Figure 11: Adding Student/Group .....................................................................40

Figure 12: Delete Student/Group .......................................................................41

Figure 13: Search Query ....................................................................................42

Figure 14: Send SMS .........................................................................................43

Figure 15: Component Diagram ........................................................................44

Figure 16: ER Diagram .....................................................................................45

Figure 17: ER Diagram for Student and Group ..................................................46

Figure 18: Table View for Group ....................................................................47

Figure 19: Table view for student ....................................................................47
Figure 20: Login Page.................................................................48
Figure 21: Admin Page ..............................................................49
Figure 22: Window for Adding a User .........................................50
Figure 23: Adding a New User .........................................................50
Figure 24: Adding a Group ...............................................................51
Figure 25: Adding a Student ............................................................52
Figure 26: Search Interface .............................................................52
Figure 27: Search by Student ...........................................................53
Figure 28: Group Search .................................................................53
Figure 29: Student Search Result .......................................................54
Figure 30: Group Search Result .........................................................55
Figure 31: Send Message .................................................................55
Figure 32: Received SMS .................................................................56
Figure 33: Django Project Structure ................................................57
Figure 34: Template Structure .........................................................59
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Huawei E398 Specifications</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>Quality Functional Deployment</td>
<td>28</td>
</tr>
<tr>
<td>3</td>
<td>Login Activity</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>Add User</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>Add Student/Group</td>
<td>29</td>
</tr>
<tr>
<td>6</td>
<td>Delete User/Group</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>Delete Student/Group</td>
<td>30</td>
</tr>
<tr>
<td>8</td>
<td>Update Student/Group</td>
<td>31</td>
</tr>
<tr>
<td>9</td>
<td>Search Window</td>
<td>31</td>
</tr>
<tr>
<td>10</td>
<td>Search by Student</td>
<td>31</td>
</tr>
<tr>
<td>11</td>
<td>Search by Group</td>
<td>32</td>
</tr>
<tr>
<td>12</td>
<td>Send SMS</td>
<td>32</td>
</tr>
<tr>
<td>13</td>
<td>Logout</td>
<td>33</td>
</tr>
<tr>
<td>14</td>
<td>Testing</td>
<td>73</td>
</tr>
</tbody>
</table>
LIST OF CODE SNIPPET

Snippet 1: Database Configuration 59

Snippet 2: Application Definition 60

Snippet 3: Message URL Configuration 61

Snippet 4: Main URL Configuration 61

Snippet 5: Login/Logout URL Configuration 61

Snippet 6: Python Packages 61

Snippet 7: Student Search Result Function 62

Snippet 8: Student Search Function 62

Snippet 9: Group Search Result 63

Snippet 10: Group Search 63

Snippet 11: Group Member List 64

Snippet 12: Group Display Function 64

Snippet 13: Function to List all the Phone Number 64

Snippet 14: Send SMS 65

Snippet 15: Function to Send SMS 65

Snippet 16: USB Modem to Send SMS with AT Command 65

Snippet 17: Student Models 66

Snippet 18: Group Model 66

Snippet 19: Form for Searching Student 67
Snippet 20: Form for searching Group

Snippet 21: Configuration Script

Snippet 22: Supervisor Configuration Script

Snippet 23: NGINX Configuration File
LIST OF ABBREVIATIONS

SMS  Short Message Service
GSM  Global System for Mobile Communications
HTTP Hypertext Transfer Protocol
CDMA Code Division Multiple Access
TDMA Time Division Multiple Access
ETSI European Telecommunication Standards Institute
3GPP Third Generation Partnership Project
SIM Subscriber Identity Module
ME Mobile Equipment
MS Mobile Station
BSS Base Stations Subsystem
BSC Base Station Controller
BTS Base Transceiver Stations
HLR Home Location Register
VLR Visitor Location Register
EIR Equipment Identify Register
NSS Network and switching Subsystem
AuC Authentication Centre
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSC</td>
<td>Mobile Switching Centre.</td>
</tr>
<tr>
<td>IMEI</td>
<td>International Mobile Equipment Identity</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Application Protocol</td>
</tr>
<tr>
<td>SMSC</td>
<td>Short Message Service Centre</td>
</tr>
<tr>
<td>CIMD</td>
<td>Computer Interface to Message Distribution</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td>WSGI</td>
<td>Web Server Gateway Interface</td>
</tr>
<tr>
<td>LTE</td>
<td>Long Term Evolution.</td>
</tr>
<tr>
<td>3G</td>
<td>Third Generation</td>
</tr>
<tr>
<td>2G</td>
<td>Second Generation</td>
</tr>
<tr>
<td>4G</td>
<td>Fourth Generation</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>IMAP</td>
<td>Internet Message Access Protocol</td>
</tr>
<tr>
<td>POP3</td>
<td>Post Office Protocol version 3</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>AT</td>
<td>Attention</td>
</tr>
<tr>
<td>EDGE</td>
<td>Enhanced Data Rate for GSM Evolution</td>
</tr>
<tr>
<td>HSPA</td>
<td>High-Speed Packet Access</td>
</tr>
<tr>
<td>MIMO</td>
<td>Multiple-Input and Multiple-Output</td>
</tr>
<tr>
<td>ER</td>
<td>Entity Relationship</td>
</tr>
</tbody>
</table>
1 INTRODUCTION:

The introduction includes the background, motivation and objective of the thesis. The main target of the project is to develop a web application to send a bulk SMS.

1.1 Background

Nowadays, technologies have become a vital part of the means of communication. Communication plays an important role in each and every field to carry out a task effectively. Everyone is connected to each other with the help of technologies. Out of various technologies, smartphones and traditional phones are a means of communication. One of the features of the mobile device is to send an SMS to people which help in the conveyance of information.

According to the statistics 18.7 billion text messages are sent worldwide every day. 12,220,700 text are sent every minute of every day worldwide, not including app-to-app text messaging. /1/ 20 percentage of teenagers with cell phones actively text. Adults under 54 years send and receive 85 plus text messages every day on average. 77% of students want relevant information from colleges via text messages. Average response time for text messages is 90 seconds. /2/

Most people nowadays prefer SMS messages for communication purpose rather than phone calls. Many companies and organization use the SMS system for sending information to customers or to employees. Since it has a low-cost implementation and reliable.

1.2 Motivation

Ubiquitous access and low implementation costs make SMS the ideal platform for building this project. The goal of this project is to allow a school to send a bulk message to parents informing about different activities going on in the school. Going through the list of the student’s parent with a one cell phone and sending the message one by one was not an efficient solution. This project tries to overcome the problem with the web-based SMS sending application.
1.3 Objective

The main objective of the project is to have a simple web interface where the user can log in to the system and search the student based on their name or study group. After the search query result, the user can type the message and send it to the selected student’s parents. The application needs to be simple with a basic feature so that anyone using the system can use it without the need for the guidance.
2 RELEVANT TECHNOLOGIES

This section covers detailed information about the technologies that are being used in the project. Also, it tries to explain the basic information about the mobile system and the SMS.

2.1 Global System for Mobile Communications

The global system for mobile communication is a second generation standard for the mobile network based on the TDMA. Mobile services based on GSM technology were first launched in Finland in 1991. SMS was a popular implementation of the GSM standard. /3/

Figure 1 shows the GSM Architecture which is comprised of many functional units. These functional units are explained below.

Figure 1: GSM Architecture

The GSM Network can be divided into:

1. Mobile Station(MS)
2. The Base Station System (BSS)
3. Network Switching Subsystem (NSS)

2.1.1 Mobile Station:

The Mobile Station is comprised of mobile equipment i.e. mobile devices and Subscriber Identity Module (SIM). The mobile station communicates across the air interface with a base station transceiver in the same cell in which the mobile subscriber unit is located. /4/

2.1.2 Base Stations Subsystem:

It comprises of:

- Base Station Controller
- Base Transceiver Stations

The base station controller manages the channel allocation during a call, monitors and maintains the quality of call and generates a handover to another cell when required.

One or more base transceiver stations is responsible for the transmitting and receiving the radio frequency signal to and from the user terminal. /4/

The BSC connects to the BTS via the Abis interface and to the mobile switching center.

2.1.3 Network and Switching Subsystem:

The network and switching subsystem is responsible for the network operation which provides a link between the cellular network and the public switched tele-communicates Networks. The switching subsystem consists of: /4/

Mobile Switching Centre

The mobile switching Centre handles the calls establishment, release and holds the database subscriber information.
**Home Location Register**

The home location register is a database that handles information of the mobile subscriber account such as geographic position, administrative information on the subscribers registered in the area.

**Visitor Location Register**

The visitor location register is a database containing information on users other than the local subscribers. The VLR retrieves the data on a new user from the HLR of the user's subscriber zone. /5/

**Equipment Identify Register**

Equipment identify register is a database that keeps the information about the identity of mobile equipment such as IMEI. This information is used to prevent calls from being misused, to prevent unauthorized or defective MSs, to report stolen mobile phones. /4/

**Authentication Centre**

The database holds the authentication and encryption key for all the subscribers in both the home and visitor location register. /4/

**2.2 Short Message Service**

SMS is a short message service for sending and receiving of messages between mobile phone. SMS was first implemented in Europe in 1992 in GSM standard. later it was ported to wireless technologies like CDMA and TDMA. SMS was developed by ETSI. The 3gpp is responsible for the development and maintenance of the GSM and SMS standards, SMS can contain at most 140 bytes of data i.e. 160 characters if 7-bit encoding is used and 70 characters if 16-bit Unicode encoding is used. /6/
SMS is not just used to send texts but it can carry binary data which make it possible to send a picture, ringtones, logos, animation and WAP configuration to a mobile phone. /6/

2.3 Short Message Service Center:

An SMS center handles the SMS operation of a wireless network. An SMS message may need to pass through more than one network entity before reaching the destination. The main duty of an SMSC is to route SMS messages and regulate the process. If the recipient is unavailable, the SMSC will store the SMS message. It will forward the SMS message when the recipient is available. SMSCs developed by different companies use their own communication protocol and most of the protocol is proprietary for example Nokia has an SMSC protocol called CIMD. /7/

A network operator can also use a third-party SMSC that is located outside the wireless network system. Typically, an SMSC address is an ordinary phone number in the international format. the SMSC address is pre-set in the SIM card by the wireless network operator. /7/

2.4 SMS Gateway:

An SMS gateway acts as a relay between two SMSCs so it acts as a translator. We cannot connect two SMSCs if they do not support a common SMSC protocol to deal with the problem SMS gateway is paced between two SMSCs. /8/

To send and receive SMS text messages on your server you connect to the SMSCs of the wireless carrier. Having a different wireless carrier with the different SMSCs protocol the application needs to have a support for the multiple SMSC protocol this result in complexity in the development process.

To overcome multiple SMSC problem figure (2) shows a process where an SMS gateway can be setup to handle the connection to SMSC. Hence the text messaging application only needs to know a method to connect to the SMS gateway.
Therefore, it makes it easy only to change the setting of the SMS gateway to support different SMSCs.

**Figure 2**: Text message application connects to SMSCs through SMS gateway

Figure (3) shows the solution where you use a GSM/GPRS modem to send and receive SMS text. The SMS text messaging application has to know how to communicate with the modem using AT command. Some SMS gateways are capable of handling the connections to mobile phones and GSM/GPRS modems. The SMS text messaging application only needs to know how to talk to the SMS gateway and does not need to know anything about AT commands.
Figure 3: Application connects GSM/GPRS modems through an SMS gateway.

2.5 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics created by Guido van Rossum and first released in 1991. It has a high-level built-in data structures combined with dynamic typing and dynamic binding which make it very attractive for rapid application development and as well as for use as a scripting or glue language to connect existing components together. /9/ Python has a large module and packages, which makes programming more flexible. Python encourages reuse of the code which makes a development work more productive.

There are mainly two versions of Python Python2 and Python3. Linux and Mac have inbuilt Python 2.7 installed on it. Python 2 is the legacy version so most of the modules and package work for Python 2. Python3 is the future of Python language. Python 3 mainly focuses to clean up the codebase and remove redundancy making it clear that there was only one way to perform the given task. /10/
2.6 Django

Django is a free and open-source web framework written in Python. Django is a high-level Python web framework that encourages rapid development, clean and pragmatic design. Django is widely used Python web application framework with Django emphasizes reusability and "plug ability" of components and the principle of don't repeat yourself which means that the common functionality for building web application comes with the framework for example: authentication, URL routing, a template engine, an object-relational mapper (ORM), and database schema migrations are all included with the Django framework. /11/

2.7 NGINX

NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption. NGINX doesn’t rely on threads to handle requests. Instead, it uses a much more scalable event-driven (asynchronous) architecture. /12/

2.8 Gunicorn

A WSGI server implements the web server side of the WSGI interface for running Python web applications. Gunicorn is one of many WSGI server implementations, but it is particularly important because it is a stable, commonly-used part of web app deployments that's powered some of the largest Python-powered web applications in the world, such as Instagram. /13/

2.9 Supervisor

A Supervisor is a client/server system that allows its users to control a number of processes on UNIX-like operating systems. The server piece of supervisor is named supervisord. A Supervisord starts processes as its sub processes, and can be configured to automatically restart them in case of a crash. It can also automatically be configured to start processes on its own invocation. A Supervisor is simple, centralized, efficient, extensible, compatible system. /14/
2.10 MySQL

MySQL is a fast, easy to use RDBMS. It is being used in different sectors and businesses. It is popular because of many features:

- It is an open source license.
- It handles a large subset of the functionality.
- It supports many operating systems and many programming languages like C, C++, PHP, Java.
- It supports up to 50 million rows or more in table. /15/

2.11 Huawei E398

Huawei E398 is the latest 4G 100Mbps USB Modem. Huawei E398 is the first LTE multi-mode date card in the world which enables seamlessly switching between the use of 2G, 3G, and LTE networks, giving users the flexibility of choosing the optimal network connection speed. The technical specification of the modem is given below:/16/

Table 1: Huawei E398 Specifications

<table>
<thead>
<tr>
<th>Brand</th>
<th>Huawei</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>E398</td>
</tr>
<tr>
<td>Chipset</td>
<td>Qualcomm's MDM9200TM chipset</td>
</tr>
<tr>
<td>Frequency</td>
<td>800/900/1800/2100/2600MHz</td>
</tr>
<tr>
<td>Speed</td>
<td>LTE: DL 100Mbps, UL 50Mbps</td>
</tr>
<tr>
<td></td>
<td>Dual-Carrier HSPA+ and Receive Diversity, 42Mbit/s in Downlink, 11Mbit/s in Uplink</td>
</tr>
<tr>
<td>Network Band</td>
<td>GSM (2G), UMTS (3G) and LTE (proto-4G)</td>
</tr>
<tr>
<td><strong>Data Speed</strong></td>
<td>Download speeds of 100 Mb/s</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>USB 2.0</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>Micro SD card slot up to 32GB</td>
</tr>
<tr>
<td><strong>Supported Systems</strong></td>
<td>Support Win 7, Windows 2000/XP/Vista, Mac and Linux Operating Systems</td>
</tr>
<tr>
<td><strong>Function</strong></td>
<td>Support LTE: CAT3, DL 100Mbps / UL 50Mbps @ 20Mhz Bandwidth</td>
</tr>
<tr>
<td></td>
<td>2*2MIMO</td>
</tr>
<tr>
<td></td>
<td>Plug &amp; Play</td>
</tr>
<tr>
<td></td>
<td>Support work mode GPRS, EDGE, HSDPA, HSPA, HSPA + with MIMO and dual carrier, as well as LTE</td>
</tr>
</tbody>
</table>

*Figure 4: Huawei E398*
3 APPLICATION DESCRIPTION

3.1 General Description

The general description and overview of the web-based application are discussed which show how the application works and interacts with the user. The description of the application shows the use of its functionality and the application feature working as required.

The main aim of this project is to build an application which is self-explanatory, convenient and dynamic. The application is based on searching the student details from the local database and sending SMS to the selected student’s parents. The application includes the following basic services:

**Login:** A super user (admin) has the only right to login to the system.

**Add/remove:** From the administrator page an admin can add, update and delete the new user to give an access to the system. Additionally, an admin can add the student and group.

**Search:** Generally, there are two methods of searching:

1. By the student’s first and last name
2. By the group

**View and select:** The selected user and group can be viewed in a table form which gives the option for the admin to select only the student’s parents to whom the message to be sent.

**Send:** A message can be typed and send from the message dialogue box.

**Logout:** An admin can logout from the system.
3.2 Quality Function Deployment

Quality function deployment of the application can be classified into three types on the basis of their priorities:

1. must-have requirement
2. should have requirement
3. nice to have requirement.

QFD mainly helps to organize the development process and methods valuable for the project and its deployment

Table 2: Quality Functional Deployment

<table>
<thead>
<tr>
<th>Must have (normal requirement)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User must be able to login using the correct credential</td>
</tr>
<tr>
<td>User must be able to search the student based on their group name, student name, student id.</td>
</tr>
<tr>
<td>User must be able to view the lists of student and select a specific student</td>
</tr>
<tr>
<td>User must be able to send bulk SMS to parents</td>
</tr>
<tr>
<td>User must be able to add, update and delete student and group</td>
</tr>
<tr>
<td>User must be able to logout</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Should have (excepted requirement)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The application should use a 4g Huawei modem (not SMS API)</td>
</tr>
<tr>
<td>The application should be installed on the Linux server</td>
</tr>
<tr>
<td>The application should be basic and user friendly</td>
</tr>
<tr>
<td>The application should be able to use the functionality to its fullest</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nice to have (exiting requirement)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application may have an email functionality added</td>
</tr>
</tbody>
</table>
3.3 Functional Specification (FS)

Functional Specification present a framework of the complete application.

**Table 3**: Login Activity

<table>
<thead>
<tr>
<th>Case</th>
<th>Administrator’s login activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precondition</td>
<td>Application should be started</td>
</tr>
<tr>
<td>Action</td>
<td>Input administrator credential</td>
</tr>
<tr>
<td>Description</td>
<td>Check administrator’s credential</td>
</tr>
<tr>
<td>Expected Result</td>
<td>Administrator should be able to login</td>
</tr>
<tr>
<td>Exception</td>
<td>Incorrect credential</td>
</tr>
</tbody>
</table>

**Table 4**: Add User

<table>
<thead>
<tr>
<th>Case</th>
<th>Add user group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precondition</td>
<td>Administrator should be logged in</td>
</tr>
<tr>
<td>Action</td>
<td>On clicking add user admin should input the username and password</td>
</tr>
<tr>
<td>Description</td>
<td>New user will be written to database</td>
</tr>
<tr>
<td>Expected Result</td>
<td>New user will be saved to database</td>
</tr>
<tr>
<td>Exception</td>
<td>Incorrect input</td>
</tr>
</tbody>
</table>

**Table 5**: Add Student/Group

<table>
<thead>
<tr>
<th>Case</th>
<th>Add students/group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precondition</td>
<td>Admin should be logged in</td>
</tr>
<tr>
<td>Action</td>
<td>On clicking add students or groups admin should input the detail information</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Description</td>
<td>The student detail and group detail will be written to database</td>
</tr>
<tr>
<td>Expected Result</td>
<td>The data will be stored to database</td>
</tr>
<tr>
<td>Exception</td>
<td>Incorrect input</td>
</tr>
</tbody>
</table>

| Table 6: Delete User/Group                                             |
|------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| Case                                                                   | Delete user/group                                                               |
| Precondition                                                           | User account should be present                                                  |
| Action                                                                 | Administrator should select the user to delete                                  |
| Description                                                            | The user will be deleted from the database                                      |
| Expected Result                                                        | User will be deleted                                                            |
| Exception                                                              | Invalid option                                                                  |

| Table 7: Delete Student /Group                                         |
|------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| Case                                                                   | Delete student/group                                                            |
| Precondition                                                           | Admin should be logged in and student should be listed                          |
| Action                                                                 | Administrator should select the student and click on delete                      |
| Description                                                            | Student will be deleted from the database                                        |
| Expected Result                                                        | Student will get deleted                                                        |
| Exception                                                              | Invalid option                                                                  |
**Table 8 : Update Student/Group**

<table>
<thead>
<tr>
<th>Case</th>
<th>Update student/group</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precondition</strong></td>
<td>Students should be listed to be modify</td>
</tr>
<tr>
<td><strong>Action</strong></td>
<td>Administrator should select the student/group to update</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Student/group data will be written to its respective student name inside database</td>
</tr>
<tr>
<td><strong>Expected Result</strong></td>
<td>Student/group detail will be added</td>
</tr>
<tr>
<td><strong>Exception</strong></td>
<td>Invalid option</td>
</tr>
</tbody>
</table>

**Table 9 : Search Window**

<table>
<thead>
<tr>
<th>Case</th>
<th>Search window</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precondition</strong></td>
<td>User should be logged in</td>
</tr>
<tr>
<td><strong>Action</strong></td>
<td>On login the user will be redirected to homepage i.e. search option</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>User can choose option to search by student or by group</td>
</tr>
<tr>
<td><strong>Expected Result</strong></td>
<td>Student will be redirected to search option page</td>
</tr>
<tr>
<td><strong>Exception</strong></td>
<td>User does not click on any option</td>
</tr>
</tbody>
</table>

**Table 10 : Search by Student**

<table>
<thead>
<tr>
<th>Case</th>
<th>Search by student</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precondition</strong></td>
<td>User should choose search by the student option</td>
</tr>
<tr>
<td><strong>Action</strong></td>
<td>User should input either the first name, last name or student number to search</td>
</tr>
<tr>
<td>Description</td>
<td>On submit button user is redirected to the result</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Expected Result</td>
<td>Student search detail will be displayed</td>
</tr>
<tr>
<td>Exception</td>
<td>Invalid option</td>
</tr>
</tbody>
</table>

**Table 11: Search by Group**

<table>
<thead>
<tr>
<th>Case</th>
<th>Search by group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precondition</td>
<td>User chooses the group search</td>
</tr>
<tr>
<td>Action</td>
<td>User input the group code or group name to search</td>
</tr>
<tr>
<td>Description</td>
<td>On search the user is redirected to result page</td>
</tr>
<tr>
<td>Expected Result</td>
<td>Student detail will be displayed</td>
</tr>
<tr>
<td>Exception</td>
<td>Invalid option</td>
</tr>
</tbody>
</table>

**Table 12: Send SMS**

<table>
<thead>
<tr>
<th>Case</th>
<th>Message to selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precondition</td>
<td>Student should be selected</td>
</tr>
<tr>
<td>Action</td>
<td>User should write the message to send</td>
</tr>
<tr>
<td>Description</td>
<td>On submit message the message is sent</td>
</tr>
<tr>
<td>Expected Result</td>
<td>The sent message is received in a cell phone</td>
</tr>
<tr>
<td>Exception</td>
<td>Invalid phone number</td>
</tr>
</tbody>
</table>
Table 13: Logout

<table>
<thead>
<tr>
<th>Case</th>
<th>Logout from system</th>
</tr>
</thead>
<tbody>
<tr>
<td>precondition</td>
<td>User should be logged in to the system</td>
</tr>
<tr>
<td>Action</td>
<td>User should click the logout button</td>
</tr>
<tr>
<td>Description</td>
<td>On clicking the logout button the user should be logged out of the system</td>
</tr>
<tr>
<td>Expected Result</td>
<td>User is logged out from the system</td>
</tr>
<tr>
<td>Exception</td>
<td>User not logged in</td>
</tr>
</tbody>
</table>

### 3.4 Use Case Diagram

Use case diagrams are mainly used for the purpose of gathering the requirement of the system which includes the internal and external influences. When a system is analyzed to gather its functionalities use case are prepared and actors are identified. An actor can be a person, an organization or any external system. An actor interacts with the use cases.

The web-based application consists of an administrator and the user as an actor who can interact with the system. The admin has the right to add, update, delete and modify. Whereas the user can only send the SMS to the selected student.

#### 3.4.1 User Use Case Diagram

Figure 5 illustrates the use case diagram of the user where an actor (user) cannot register to the system directly. The admin has to give a right to the user to access the system. If the given credential for login is incorrect the application shows the error message. When the user is logged into the system, a user can search on the basis of the student’s name or number or by the study group. A user can view the list of the students and group to which the respective student is affiliated to on the
basis of the user search option. The user can select the student or group to whom
the user need to send an SMS. A user can logout from the system

![User Use Case Diagram](image)

**Figure 5**: User Use Case Diagram

### 3.4.2 Administrator Use Case Diagram

The use case diagram of the administrator differs from the use case diagram of the
user. Since the administrator is a super user it should only have the rights to
change the system. The administrator has its credential inside the system which
can be changed later if required. If the given credential does not match with the
system credential it shows the error message.

The main objective of having an administrator is to maintain the application so
that the admin can add, view, modify and delete the user and as well as the student
and groups. Apart from it the administrator can search for the student or group
and send bulk SMS or single SMS to the parents.
Figure 6: Admin Use Case Diagram
3.5 Model Dependency Diagram

Model dependency diagram enables us to get an overview of the models within Django application, and analyse their relationships. Figure (7) shows the directed orthogonal model dependency diagram. In the diagram, we have the student, group, user, profile and authentication model. The students have a list of fields that includes the email, first name, group set, group, last name, parent name, parent phone number Similarly, the figure shows the fields related to the models.

Figure 7: Directed Orthogonal Class Diagram
3.6 Sequence Diagram

The main purpose of a sequence diagram is to define event sequences that result in some desired outcome. The focus is less on the messages themselves and more on the order in which messages occur. /17/

3.6.1 Administrator Login Sequence Diagram

Figure 8 describes the administrator login to the system. The administrator should login with the correct credential to access the admin page, the login request is sent to the DB handler to check the credential in the database. If the credential does not match it return error message otherwise admin will be given access to the admin page.

![Interaction Diagram](image)

**Figure 8:** Admin Login Sequence Diagram
3.6.2 Admin Add a User Sequence Diagram

The user does not have the privilege to register themselves which means they cannot access the system without the permission of the administrator. So, to have a permission the admin should add the user to access the system. After login to the admin page, the admin should go to add a user button and fill up the required field to add a user to the system. The database handler handles the request and sent it to the database for update. If the admin does not give the correct input field error message is returned.

**Figure 9**: Admin Adding a User
3.6.3 Admin Delete User Sequence Diagram

Figure 10 describes the sequence diagram deleting the user. After login to the system, the administrator can go to the profiles to view the list of users currently active. The administrator can select the user which he/she wants to delete from the list. The application sends a query to the DB handler for the user selected to be removed from the database.

Figure 10: Admin Delete a User
3.6.4 Admin Add Student/Group:

Figure 11 describes the sequence diagram for adding student and group. After log-in to the system admin can go the admin page to add student and group. Here the admin can add the student to the details such as name, class, id, parent phone number. Also, a user can add to a group based on their class. So when the given detail is entered and save button is pressed the application sends a query to the DB handler which adds the detail information to the database. The required field must be filled up in order to put a data to database otherwise the error message is shown.

![Diagram](image)

**Figure 11:** Adding Student/Group
3.6.5 Admin Delete Student/Group

Figure 12 describes the sequence diagram for deleting a student or group. The administrator should be logged in to the system after login to the system admin can go the administrator page to select the student or group. The selected option shows the list of the student or group that admin wants to delete. So when the student or group is selected the application sends a query to the DB handler. This results in the removal of the student or group from the database.

![Interaction Sequence Diagram](image)

**Figure 12**: Delete Student /Group

3.6.6 User Search the Student /Group

Figure 13 describes the sequence diagram for searching the student and group. When the user login to the system it redirects the user to the main page where the user has the option to select search on the basis of student or group. When the user
searches for the student or group detail the DB handler send the query to the database and the result is fetched on the webpage. If the data does not exist it returns nothing. The user gets the list view of the student, group and detail information.

**Figure 13**: Search Query
3.6.7 User /Admin Send the SMS

Figure 14 describes when the user/admin searches the student they have the option to send the message to the selected list of the student or group. The user is redirected to the message page when the send SMS button is clicked. The user can type a message in the dialogue box and send the SMS to a single person or to a group.

![Diagram showing interaction sequence for sending SMS]

**Figure 14**: Send SMS
3.7 Component Diagram

Figure 15 illustrates the structural relationships between the components of a system.

The system is comprised of mainly four nodes:

- Server
- User Interface
- Database
- Hardware

The web browser sends the HTTP request to the NGINX server. Django accesses the data from the MySQL database and MySQL python connector enables us to develop Python applications that require secure, high-performance data connectivity with MySQL. Admin view is accessible to the user who has administrator right only and user view by the admin and user.

![Component Diagram](image)

**Figure 15:** Component Diagram
4 DATABASE:

4.1 Design of Database

MySQL database management system is used for the purpose of creating a database. The data are stored in a local database.

An entity relationship diagram is presented below for the whole project:

Figure 16: ER Diagram
Student and group ER diagram is shown in figure 17. The database consists of four tables message_group, message_student_group, message_student and message_group member. Message_group stores the group information when the admin adds a new group every time. The message_student table is created when every time administrator adds a new student. The other two tables hold the student_id and group_id key.

Figure 17: ER Diagram for Student and Group
Figure 18: Table View for Group

Figure 18 shows the table containing the list of field for the group:

- **Id**: the auto-increment of the record
- **Group code**: The group code example: I-IT
- **Name**: Name of the group
- **Email**: Group email

Figure 19: Table view for student

In figure 19 the table contains the view for the students which has the following fields:

- **Id**: the auto-increment of the record
- **First name**
- **Last name**
- **Middle name**
- **Student number**
- **Email**
- **Parent phone number**
- **Parents name**
5 GRAPHICAL USER INTERFACE DESIGN

The project is a web based application. The frontend is implemented using HTML, CSS, JavaScript and other web technologies. Django default template engine is used as a web framework. A template is the most common approach to generate HTML dynamically in Django.

5.1 Login Page

Figure 20 shows the login page for the user to login. The user can enter the credential to login to the system. The admin credential is created beforehand while developing the application, so that the admin can log in directly. The user does not have the option to register for the account. They can access the system only if the admin creates the account for them.

![Login Page](image.png)

**Figure 20**: Login Page
5.2 Admin Page

Figure 21 shows the admin page. The admin page is a Django automatic admin interface which reads metadata from the model that we created to provide an interface to manage the content of our site. The admin page has the option of creating and modifying the profile, students and groups.

![Admin Page](image)

**Figure 21** : Admin Page

5.2.1 Adding a New User

Figure 22 shows the page for creating a new user. On selecting the users from the authentication and authorization the admin is redirected to another page where a admin can add new a user.
**Figure 22**: Window for Adding a User

Figure 23 shows the page for adding a new user. The input text field for adding a new user must be filled to add a new user. On clicking the save button, a new user is added to the system.

**Figure 23**: Adding a New User
5.2.2 Adding a Group

Figure 24 shows the page for adding a group. From the main admin page, the user can select the group message under message option to add a group. To add a group, the input field needs to be filled where the email field can be left empty. Also from the add group page, you can add the students to the group or create a new student to be added to the group.

![Add group page](image)

**Figure 24:** Adding a Group

5.2.3 Adding the Student

Figure 25 shows the activity for adding the student. The input text field needs to be filled. The given fields: first name, last name, parent name, the mobile number should be filled and the student number, email field can be left blank. The created student can be directly added to the group by selecting the group to which the student is affiliated. If the group is not present, the admin can add the group from the add student page also.
Figure 25: Adding a Student

5.3 Data Search Interface

Figure 26 shows the search interface. On logging to the system user is redirected to the data search interface page. The user can choose the option of searching on the basis of:

- search by student
- search by group option.
5.3.1 Search by Student

Figure 27 shows the page for the student search query. When the user selects the option search by the student on the data search interface the user is redirected to student search page. The user can search on the basis of student first name or last name or by the student number. Also, the user has the feature to enable a quick messaging and to display a result in a place.

Figure 27: Search by Student

5.3.2 Search by Group

Figure 28 shows the search by group method in which the user can search on the basic of the group name or group code.

Figure 28: Group Search
5.4 Search Query Result

5.4.1 Student Search Result

Figure 29 shows the student search result on the basis of the user given search query. The search query gives the user detail information i.e. first name, last name, email and student number. On clicking HTML option under the groups a user can check the selected student group. The user has the option to select the student to whom a message to be sent. The user can select the message to selected option to send the message to the selected student.

Figure 29: Student Search Result

5.4.2 Group Search Result

Figure 30 shows the search result on the basis of the group search. The group search shows the group code, name of group and group email. The user can click on the members to check the student belongings in that group. The user can select the message to selected option to send the message to the whole group.
Figure 30: Group Search Result

5.4.3 SMS Send Page

Figure 31 shows the dialogue box where the user can type a message to be sent to the recipient number. Also, the user can add the new recipient SMS number and delete the SMS number. On clicking the submit button the SMS is sent to the recipient.

Figure 31: Send Message
Figure 32 shows the message received.

Figure 32: Received SMS
6 IMPLEMENTATION

6.1 Project Structure

Figure 33: Django Project Structure
6.1.1 Message Project Files

Figure 33 shows the Django project structure which has a message_project as the main project folder. The project folder consists of sub-projects and the Python files which are necessary for configuration of the application. The common function of these Python files are described below:

1. `message_project/` is a root directory which is a container for the whole project.
2. `manage.py`: A command-line utility that lets us interact with Django project in various ways.
3. The inner `message_project/` directory contains the actual Python package for our project.
4. `Message_project/__init__.py`: An empty file that tells Python that this directory is considered as a Python package.
5. `Message_project/settings.py`: It has all the settings/configuration required for the Django project to work.
6. `Message_project/urls.py`: The URL declarations for this Django project.
7. `Message_project/wsgi.py`: An WSGI-compatible web server to serve our project.

6.1.2 Message and Account Directory

The message and account directory contains the file that is important for our project to function properly. On creating a Django application these Python files are automatically generated. The specific uses of these files are described below:

1. `Admin.py`: In this file, we configure the admin panel which helps in the ease of management of the database and the content of our website.
2. `Forms.py`: A model form maps a model class fields to HTML form. Form allow the user to enter text, select options, manipulate objects or controls and then send that information back to the server.
3. *Models.py*: In this file we create code that represents database schema, allowing us to create tables without using SQL.

4. *urls.py*: It has a URL configuration which is generally, a set of patterns that Django will try to match the requested URL to find the correct view.

5. *Views.py*: Inside a view file, we simply have a function that takes a web request and returns the web response.

![Template Structure](image)

**Figure 34**: Template Structure

### 6.2 Settings

#### 6.2.1 Database Configuration

Inside the main project directory file settings.py we are adding a database configuration to work with our MySQL database.

```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'OPTIONS': {
            'read_default_file': BASE_DIR + '/db.cnf',
        },
    }
}
```

**Snippet 1**: Database Configuration
6.2.2 Application Definition

Message and accounts are the apps that we have created to interact with the project to handle the task. These apps should be defined inside an application for the app to work.

```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'accounts',
    'message',
]
```

Snippet 2: Application Definition

6.3 URL Configuration

A URL configuration is like a table of contents for Django-powered website. A simple example elaborate how it works:

```python
url(r'^student_search$', views.student_search, name='student_search').
```

Here, the first argument is a regular expression which is the pattern-matching string. ‘r’ character in front of the regular expression string tells Python that the string is a “raw string” and the second argument tell us the view function to use for the pattern URL. A request comes in to /student_search/. Django determines the root URL configuration by looking at the ROOT_URLCONF setting. Django looks at all of the URL patterns in the URL configuration for the first one that matches /student_search/. If it finds a match, it calls the associated view function. The view function returns an HTTP response.

6.3.1 Message URL Configuration

```python
from django.conf.urls import url
from . import views
app_name = "message"
urlpatterns = [
    # list of url and thier respective view modules
    url(r'^student_search$', views.student_search, name='student_search'),
]"
6.3.2 Main URL Configuration

```python
from django.conf.urls import include, url
from django.contrib import admin
from django.views.generic import TemplateView
from django.contrib.auth.decorators import login_required
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^accounts/', include('accounts.urls', namespace='accounts')),
    url(r'^$', login_required(TemplateView.as_view(template_name='home.html')), name='index'),
    url(r'^message/', include('message.urls', namespace='message'))
]
```

Snippet 4: Main URL Configuration

6.3.3 Login/Logout URL Configuration

```python
app_name = "accounts"
urlpatterns = [
    url(r'^login/$', auth_views.login, {'template_name': 'accounts/login.html'}, name='login'),
    url(r'^logout/$', auth_views.logout, {'template_name': 'accounts/logout.html'}, name='logout'),
    # url(r'^register/$', register, name='register'),
]
```

Snippet 5: Login/Logout URL Configuration

6.4 Python Modules

The application needs some of the Python packages for the HTTP responses and for database models. Python required packages are imported as follows.

```python
from django.shortcuts import render
from django.db.models import Q
from .forms import StudentSearch, GroupSearch
from .models import Student, Group
from django.contrib.auth.decorators import login_required
from django.http import HttpResponseNotFound, HttpResponse, HttpResponseRedirect
from django.views.decorators.http import require_http_methods
from django.core.mail import *
from django.core.urlresolvers import reverse
from .sms import *
```

Snippet 6: Python Packages
6.5 View

The main function is stored inside the View.py file.

6.5.1 Student Search Result

In the snippet 7 before the start of the function the login required decorator is used which help if the user is not logged in then redirect to login URL. If the user is logged in, executes the view normally.

The function returns an HTTP Response object of the template to use. Here, it accesses the student_search.html page.

```python
@login_required(login_url='/accounts/login/')
def student_search(request):
    # render template for student search
    return render(request, 'message/student_search.html', {'student_search':StudentSearch()})
```

Snippet 7: Student Search Result Function

6.5.2 Search Student

The student_search function is used for the searching the query from the database on the basis of first name or last name or student number. The function renders the student_result template from template folder.

```python
@login_required(login_url='/accounts/login/')
def student_result(request):
    # searching the query against the database
    if request.method == 'POST':
        lastname = request.POST['lastname'] if request.POST['lastname'] else "pqiouer-pouiaspdif"
        firstname = request.POST['firstname'] if request.POST['firstname'] else "pqiouer-pouiaspdif"
        student_number = request.POST['student_number'] if request.POST['student_number'] else 9182739
        all_in_one_page = True if ('all_in_one_page' in request.POST) else False
        quick_messaging = True if ('quick_messaging' in request.POST) else False
        result_students = Student.objects.filter(Q(first_name__icontains=firstname) | Q(last_name__icontains=lastname))
        # search results
        result_students = Student.objects.filter(Q(first_name__icontains=firstname) | Q(last_name__icontains=lastname) | Q(student_number__icontains=student_number))
        # templating for showing search results
        return render(request, 'message/student_result.html', {'result_students': result_students, 'quick_messaging': quick_messaging})
    # returning error if something wrong happens
    return HttpResponseNotFound('<h1>Page not found</h1>')
```

Snippet 8: Student Search Function
6.5.3 Group Search Result

This function returns the group search page.

```python
@login_required(login_url='/accounts/login/')
def group_search(request):
    # render template for group search
    return render(request, 'message/group_search.html', {'group_search': GroupSearch()})
```

Snippet 9: Group Search Result

6.5.4 Group Search

The group_result function is used for searching the query from the database on the basis of group code or group name. The function renders a group_display template from template folder.

```python
{'group_list': result_groups,'quick_messaging': quick_messaging}
```

The value in the dictionary is callable so the view will call it just before rendering the template.

```python
@login_required(login_url='/accounts/login/')
def group_result(request):
    # same as student result but for group search
    if request.method == "POST":
        group_code = request.POST["group_code"] if request.POST["group_code"] else "pqioerpuiaspdi"
        group_name = request.POST["group_name"] if request.POST["group_name"] else "pqioerpuiaspdi"
        all_in_one_page = True if ('all_in_one_page' in request.POST) else False
        quick_messaging = True if ('quick_messaging' in request.POST) else False
        result_groups = Group.objects.filter(Q(group_code__icontains=group_code) | Q(name__icontains=group_name))
    return render(request, 'message/group_display.html', {'group_list': result_groups,'quick_messaging': quick_messaging})
```

Snippet 10: Group Search

6.5.5 Group Member List

This function lists all the members of the group. The function renders a student_result template.

```python
@login_required(login_url='/accounts/login/')
def group_members(request, group_id):
    # getting all the members of group
    members = Group.objects.get(id = group_id).members.all()
    # returning the member list
```
return render(request, 'message/student_result.html', {'result_students': members, 'quick_messaging': True})

Snippet 11: Group Member List

6.5.6 Group Display

This function on clicking the student it shows the group or class the student is affiliated to. On clicking the student, it renders the template for group list display.

@login_required(login_url='/accounts/login/')
def group_display(request, student_id=None):
    if student_id:
        groups = Student.objects.get(id=student_id).groups.all()
        # getting grouping for displaying
        # rendering template for group list display
        return render(request, 'message/group_display.html', {'group_list': groups, 'quick_messaging': True})

Snippet 12: Group Display Function

6.5.7 List Phone Number to Send SMS

This Function appends the student parent phone number and stores it in a list. The function then returns the phone number that are stored in the list for sending the SMS.

@login_required(login_url='/accounts/login/')
@require_http_methods(['POST'])
def SMS_send(request):
    email_list = []
    sms_list = []
    for i in request.POST.getlist('recipient[]'):
        j = Student.objects.get(id=i)
        # getting email
        email_list.append(j.email)
        print(j.parent_phone_number)  # getting sms
        sms_list.append(j.parent_phone_number)
    # returning sms list for sending email and sms
    return render(request, 'message/email.html', {'recipient_list': email_list, 'sms_list': sms_list})

Snippet 13: Function to List all the Phone Number

6.5.8 Sending SMS

This function request to get all the recipient number and call the send_sms1 Function to send SMS. On sending the SMS the function returns to the index page.

def final_email(request):
    print(request.POST)
    # getting all email list
    for i in request.POST.getlist('recipient[]'):.
Snippet 14: Send SMS

This function takes two arguments phone number and message to be sent. Both the message should be given in string format.

```python
def send_sms1(email, message):
    # sending sms function
    send_sms(message=str(message), phonenumber=str(email.strip()))
    print("message sent")
```

Snippet 15: Function to Send SMS

6.6 SMS with AT Command

The sms.py file has a source code to interact with the modem for sending SMS where the modem works with AT commands. Inside a function we initialize a serial connection to USB modem then we use AT command to send the text message to the user given phone number.

```python
def send_sms(message, phonenumber):
    # Initialize serial connection to 3G USB Modem
    modem = serial.Serial(device, baudrate, timeout=5)
    print("Connected to ", modem.name)
    # Check modem status
    print("Modem Status: OK")
    modem.write(b"AT+CMGF=1\r\n".encode())
    print("Modem set to text mode!")
    print(modem.write(b"AT+CMGS="%s\r\n" % phonenumber.encode()))
    time.sleep(0.5)
    print(modem.write(b"%s\r\n" % message.encode()))
    time.sleep(0.5)
    print(modem.write(ascii.ctrl('z').encode()))
    time.sleep(0.5)
```

Snippet 16: USB Modem to Send SMS with AT Command

6.7 Models

Snippet 17 and 18 shows the description of the data in a database, represented by the Python code, instead of SQL. The model is used to execute the SQL code. From models, it is easy to maintain the database.
In the snippet 17 student model is defined which create the first name, last name, parents name, middle name, student number, parent phone number, the email inside a MySQL database. \texttt{__str__()} returns the representation of an object.

### 6.7.1 Student Models

```python
#model of Student table in the database
class Student(models.Model):
    first_name = models.CharField(max_length=200)
    last_name = models.CharField(max_length=200)
    middle_name = models.CharField(max_length=200, default='none')
    parents_name = models.CharField(max_length=200)
    student_number = models.IntegerField()
    parent_phone_number = models.CharField(max_length=30, blank=True, default='0000')
    email = models.EmailField()
    groups = models.ManyToManyField('group', blank=True)

    def __str__(self):
        return u'{} {} | {}'.format(self.first_name, self.last_name, self.student_number)
```

**Snippet 17:** Student Models

Similarly, in the snippet 18 group model is defined which create a group code, name, email, members in a MySQL database.

### 6.7.2 Group Models

```python
#model of group table in the database
class Group(models.Model):
    group_code = models.CharField(max_length=200)
    name = models.CharField(max_length=500)
    email = models.EmailField()
    members = models.ManyToManyField(Student, through='Student.groups', through=True)

    def __str__(self):
        return u'{} | {}'.format(self.name, self.group_code)
```

**Snippet 18:** Group Model

### 6.8 Form

Django has a form library called \texttt{Django.forms} which handles many issue with the HTML form display and validation.

#### 6.8.1 Student Search Form

We create a student search form on the basis of the student last name, first name and student number as a search field.

```python
#Form for Searching Student
class StudentSearch(forms.Form):
```
Snippet 19: Form for Searching Student

6.8.2 Group Search Form

Group search form include a search form on the reference to the group code and group name.

```python
#form for searching for group
class GroupSearch(forms.Form):
    group_code = forms.CharField(label = "Group Code", max_length = 200, required = False)
    group_name = forms.CharField(label = "Group Name", max_length = 200, required = False)
    quick_messaging = forms.BooleanField(label = "Enable Quick Messaging", required = False, initial = True)
    all_in_one_page = forms.BooleanField(label = "All in one page", required = False)
```

Snippet 20: Form for searching Group
7 DEPLOYMENT

For the project to work properly configuration needs to be done and this includes downloading and installing Python packages and configuring the server.

For the purpose of the project, Ubuntu is used as an operating system. The configuration of the project is as follows:

7.1 Package Installation

First of all, we need to install the Python version 3 to work on this project. Also, we need to install PIP which is a commonly used package management system for Python. And we will install Nginx for server management.

```bash
Sudo apt-get update
Sudo apt-get install python3 python3-pip python3-dev nginx
```

7.2 Virtual Environment Setup

To manage the project virtual environment is used which helps to work with the specific Python project with a complete isolation without worry of affecting other projects. The virtual environment creates a folder with necessary Python libraries which help in installing the required package for the specific project.

```bash
pip3 install virtualenv

// creating a virtual environment

sms@sms-OptiPlex-755:~$ virtualenv -p python3 smsserver

// Activating the Virtual environment

sms@sms-OptiPlex-755:~/$ smsserver/bin/activate

(smsserver) sms@sms-OptiPlex-755:~/sms$
```
7.2.1 Package Installation in Virtual Environment

The packages that are essential for the project are: Django, MySQLclient, pyserial, and gunicorn. They are installed in a virtual environment.

```
(smsserver) sms@sms-OptiPlex-755:~/sms$ pip3 install django gunicorn
(smsserver) sms@sms-OptiPlex-755:~/sms$ pip3 install mysqlclient
(smsserver) sms@sms-OptiPlex-755:~/sms$ pip3 install pyserial
```

7.3 Database Setup

The records are stored in a local database. So, for this purpose MySQL needs to be installed and configured to create a new database to store the project data.

```
// Installation of MySQL-server

apt-get install mysql-server

// login to MySQL:

mysql -u root #If you didn't set a password
mysql -u root -p If you set a Password

//create a database for the Project

mysql > CREATE DATABASE message_project;
```

7.4 Gunicorn

A configuration script is saved to make a Gunicorn more useful where we set the parameters and save them into a bash script.

```
(smsserver) sms@sms-OptiPlex-755:~/sms$ vim gunicorn_start.bash
```
#!/bin/bash

NAME="message_project"  # Name of the application
DJANGODIR=/home/sms/sms/message_project/  # Django project directory
SOCKFILE=/home/sms/sms/smsserver/run/gunicorn.sock  # Unix socket
USER=sms  # user
GROUP=sms  # group
NUM_WORKERS=3  # worker processes should Gunicorn spawn
DJANGO_SETTINGS_MODULE=message_project.settings  # Django file setting
DJANGO_WSGI_MODULE=message_project.wsgi  # WSGI module name

echo "Starting $NAME as `whoami`"

# Activate the virtual environment

cd $DJANGODIR
source /home/sms/sms/smsserver/bin/activate
export DJANGO_SETTINGS_MODULE=$DJANGO_SETTINGS_MODULE
export PYTHONPATH=$DJANGODIR:$PYTHONPATH

# Create the run directory if it doesn't exist
RUNDIR=$(dirname $SOCKFILE)
test -d $RUNDIR || mkdir -p $RUNDIR

# Start Django Unicorn
exec gunicorn ${DJANGO_WSGI_MODULE}:application \
  --name $NAME \
  --workers $NUM_WORKERS \
  --user=$USER --group=$GROUP \
  --bind=unix:$SOCKFILE \
  --log-level=debug \
  --log-file=-

Snippet 21: Configuration Script

7.5  Supervisor

The configuration script from the snippet 21 is now ready and working. To make
sure that it starts automatically with the system and it starts automatically if for
any reason it quit unexpectedly for this purpose, supervisor is installed which
handles these tasks. A configuration file is created in the directory
/etc/supervisor/conf.d/.

Snippet 22:  Supervisor Configuration Script
After the configuration file is saved for the program a supervisor can be asked to re-read the configuration file and update it so our new configuration file gets added by running the following commands.

```
(smsserver) sms@OptiPlex-755:~/sms$ sudo supervisorctl reread
(smsserver) sms@OptiPlex-755:~/sms$ sudo supervisorctl update
```

To check status:

```
(smsserver) sms@OptiPlex-755:~/sms$ sudo supervisorctl status message_project
```

### 7.6 NGINX

A nginx virtual server configuration file for our application and its static files is created inside `/etc/nginx/sites-available/` directory.

```conf
upstream message_project {
    server unix:/home/sms/sms/smsserver/run/gunicorn.sock fail_timeout=0;
}
server {
    listen 80;
    server_name example.com;
    client_max_body_size 4G;
    access_log /home/sms/logs/nginx-access.log;
    error_log /home/sms/logs/nginx-error.log;
    location /static/ {
        alias /home/sms/sms/message_project/staticfiles/;
    }
    location /media/ {
        alias /home/sms/media/;
    }
    location / {
        # an HTTP header important enough to have its own Wikipedia entry:
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;
    }
```
if (!-f $request_filename) {
    proxy_pass http://message_project;
    break;
}

# Error pages
error_page 500 502 503 504 /500.html;
location = /500.html {
    alias /home/sms/sms/message_project/staticfiles/;
}

**Snippet 23:** NGINX Configuration File

Now create the symbolic link in the sites-enabled folder and restart the server. This will deploy Django sites powered by Nginx and Gunicorn.

```
(smsserver) sms@sms-OptiPlex-755:~/sms$ sudo ln -s /etc/nginx/sites-available/sms.conf /etc/nginx/sites-enabled/sms.conf
(smsserver) sms@sms-OptiPlex-755:~/sms$ sudo service nginx restart
```
8 TESTING

Table 14: Testing

<table>
<thead>
<tr>
<th>#</th>
<th>Test case</th>
<th>Expected result</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Starting application</td>
<td>The application should start.</td>
<td>The application starts and redirect to login page.</td>
</tr>
<tr>
<td>2</td>
<td>Login to system</td>
<td>With the correct credential the user should be able to login to the system.</td>
<td>On giving the correct credential the user is redirected to home page.</td>
</tr>
<tr>
<td>4</td>
<td>Adding user</td>
<td>The admin should be able to add a non-exiting user from the admin page.</td>
<td>If the user is not registered the user is added to the user list.</td>
</tr>
<tr>
<td>5</td>
<td>Updating user</td>
<td>The admin should be able to update or delete a created user.</td>
<td>The changes were updated in the system.</td>
</tr>
<tr>
<td>6</td>
<td>Adding student/group</td>
<td>The admin should be able to add the student or group and the required filled should be filled.</td>
<td>When all the required field of input are filled the new student/group are added to the database or else error message is displayed for incorrect input.</td>
</tr>
<tr>
<td>7</td>
<td>Updating student/updating group</td>
<td>The admin should be able to update added list of student or group.</td>
<td>On modifying or deleting the list of student or group the changes are seen in the system.</td>
</tr>
<tr>
<td>8</td>
<td>Main search page</td>
<td>The user should be redirected to the main search page on login.</td>
<td>The user is directly redirected to the login page.</td>
</tr>
<tr>
<td>9</td>
<td>Search by student</td>
<td>The user should be redirected to search window for searching student</td>
<td>The user is redirected to student search window.</td>
</tr>
<tr>
<td>10</td>
<td>Search by group</td>
<td>The user should be redirected to group search window on selecting search by group option.</td>
<td>On clicking the search by group user is redirected to search by group window.</td>
</tr>
<tr>
<td>11</td>
<td>Search result</td>
<td>On giving a search query the user should be shown with a search result.</td>
<td>If the student/group is in the database then the list of student and group are displayed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Message page</td>
<td>The user should be redirected to message page on selecting send an SMS option.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>On selecting the student and groups to whom the message to be sent the user is redirected to message page.</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Add/delete recipient number</td>
<td>The user should be able to add/delete recipient number on selecting the add and delete option.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>On selecting add option the user is able to add a recipient number as well as delete on selecting delete option</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Send SMS</td>
<td>The user should be able to send a message typed on message box to the selected student or groups.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>On clicking send option the user send the message to the selected student parents.</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Logout</td>
<td>The user should be able to logout of the system on clicking logout.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The user get logged out of the system.</td>
<td></td>
</tr>
</tbody>
</table>
9 CONCLUSION

To overcome the problem faced by the school this project is carried out. The main aim of the project is to develop a web-based application where the user can search the student from the database and send the SMS to the selected student parents informing different activities. Since the application major purpose is just to send SMS taking this into account a simple web interface is developed which is easy to use and self-guided without any trouble. The super user has the right to give permission to other the user for the access to the application. The admin can add the new user, add the student and groups as well as update the information.

The goal of the project was clear. So, on completion of the project, the main goal has been fulfilled. The project has a future for the further development on the basis of requirement.

Over the course of the development of this application, I have come across various problem and solution method. This project has helped me to learn new things, tools, and working environment. For the development purpose, I learned various problem-solving method with the Python and Django. Learning Django was a new and interesting thing for me. The implementation with the Django was a little bit challenge. For the deployment of the project I used a NGINX and Gunicorn. Deployment of the application was not an easy task usually, there was always a problem with a server configuration or with the Gunicorn bash file which took time to debug.

The basic goal of the application was achieved in the completion of the project. However, there are much more possibilities in the project that can be implemented to make a better application.

- Improving the security measures to protect from malicious actions
- Improvement of the search functionality
- With the failure of SMS message sent the email to admin reporting the fail message.
10 REFERENCES

/1/ Text Message Statistics Accessed 25.11.2017


/2/ SMS Statistics Accessed 25.11.2017


/3/ Global System for mobile Communication Accessed 25.11.2017

https://www.techopedia.com/definition/5062/global-system-for-mobilecommunications-gsm


/5/ GSM Architecture Overview Accessed 25.11.2017

https://www.slideshare.net/abhishekshringi/gsm-architecture-11984082

/6/ SMS introduction Accessed 25.11.2017

http://www.developershome.com/sms/

/7/ About SMS center Accessed 25.11.2017


/8/ About SMS Gateway Accessed 25.11.2017


/9/ Introduction to Python Accessed 25.11.2017

https://www.python.org/doc/essays/blurb/
10/ Key difference Between Python 2 and python 3 Accessed 25.11.2017

https://www.digitalocean.com/community/tutorials/python-2-vs-python-3-practical-considerations-2#key-differences

11/ Introduction about Django Accessed 25.11.2017

https://www.djangoproject.com/

12/ About nginx Accessed 25.11.2017

https://www.nginx.com/resources/wiki/

13/ About Gunicorn Accessed 25.11.2017

https://www.fullstackpython.com/green-unicorn-gunicorn.html

14/ introduction to Supervisor Accessed 25.11.2017

http://supervisord.org/introduction.html#overview

15/ About MySQL database Accessed 25.11.2017

http://searchoracle.techtarget.com/definition/MySQL

16/ Huawei E398 Specification Accessed 25.11.2017

https://www.4gltemall.com/huawei-e398-4g-lte-tdd-fdd-100mbps-usb-surfstick.html

17/ Sequence Diagram Accessed 25.11.2017