SMS Organizer

Version 1.0

Prepared by

Muhammad Awais Rashid

Supervised by

Dr. Onaiza Maqbool

Acknowledgement

Firstly, thanks to Almighty Allah. I am very thankful to my parents and siblings. This projects is not the end, it’s actually the beginning towards success. I can’t forget to say thank you to my supervisor Dr. Onaiza Maqbool, she’s not an instructor but a real teacher, and I actually learnt a lot from her moral values. The whole program was tough and challenging but at the end I am quite happy. Thanks everyone who guided, gave me suggestions and especially who criticized.

Muhammad Awais Rashid

Abstract

SMS Organizer is an android based text messaging application. Users of this application can send SMS, create contact categories, block contacts and set auto-response for particular contacts. An important feature is that, system can put irrelevant SMS into a Spam Folder using machine learning algorithm.

Table of Contents

Table of Figures

List of Tables

1. Software Project Management Plan

1.1 Introduction

This chapter presents the Software Project Management Plan i.e. tasks, milestones and deliverables. Also describes tools and techniques used for development purpose.

1.1.1 Project Overview

It's an android based text messaging application. User of this application can create contact categories, block contacts and also set auto-response messages for particular contacts. An important feature is that, system uses machine learning algorithm to put irrelevant text messages into a Spam Folder.

1.1.2 Project Deliverables

1. Software Project Management Plan (SPMP)
2. Software Requirement Specifications (SRS)
3. Software Design Description (SDD)
4. Software Test Documentation (STD).

1.2 PROJECT ORGANIZATION

1.2.1 Software Process Model

Waterfall Model is used for the development of this project. This model is simple and easy to understand and use. It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.

1.2.2 Roles and Responsibilities

As I am a solo person doing this project. So all the roles and responsibilities are on my end. Gathering information, Analysis, Design, Implementation and Testing fall in this category.

1.2.3 Tools and Techniques

Following are the tools used for this project.

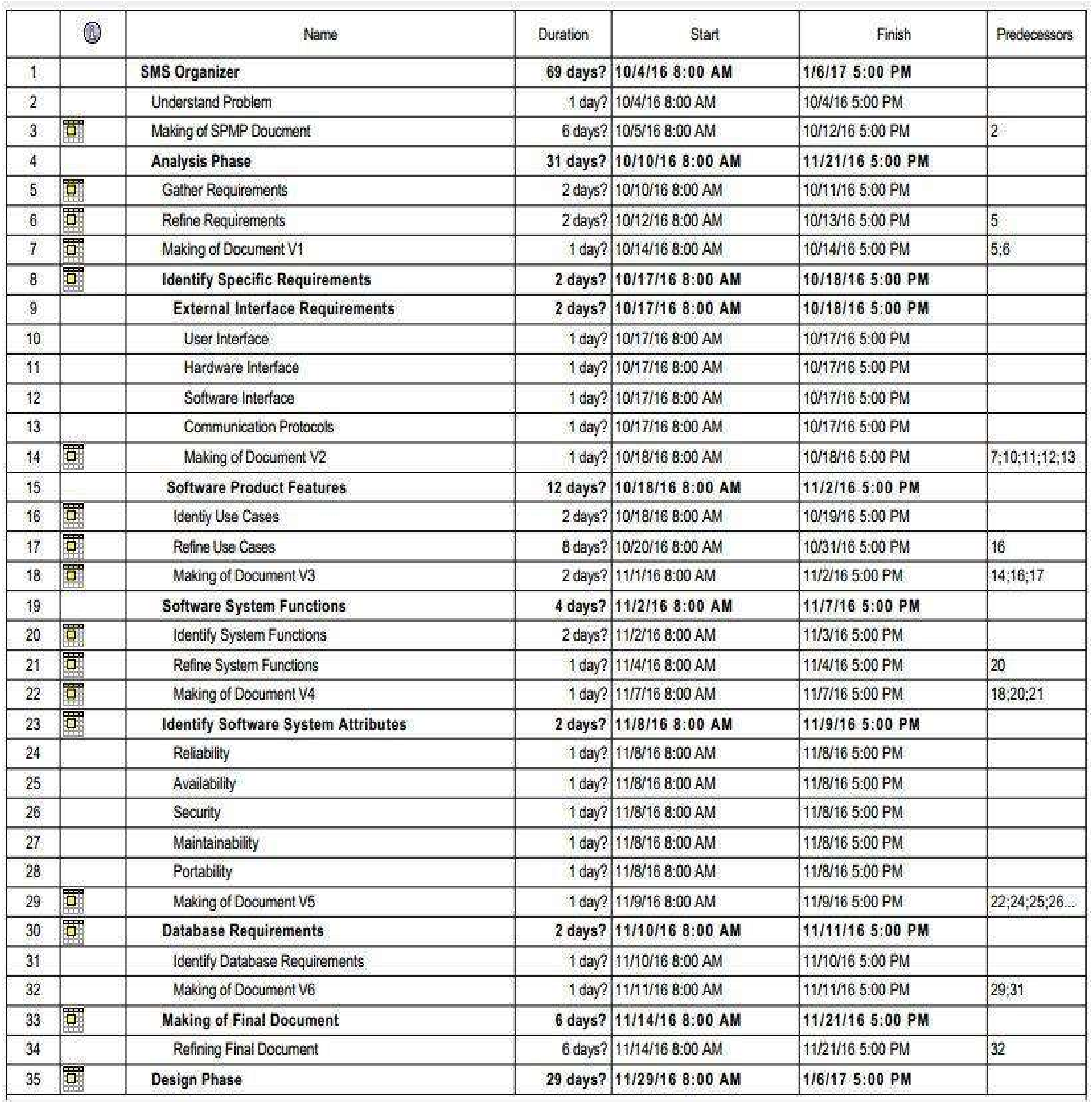
1. MS Word (For Documentation)
2. MS VISIO, Pencil Tool (For Designing)

1.3 PROJECT MANAGEMENT PLAN

This section describes software process model, stakeholders' roles and responsibilities along with tools & techniques for development purpose.

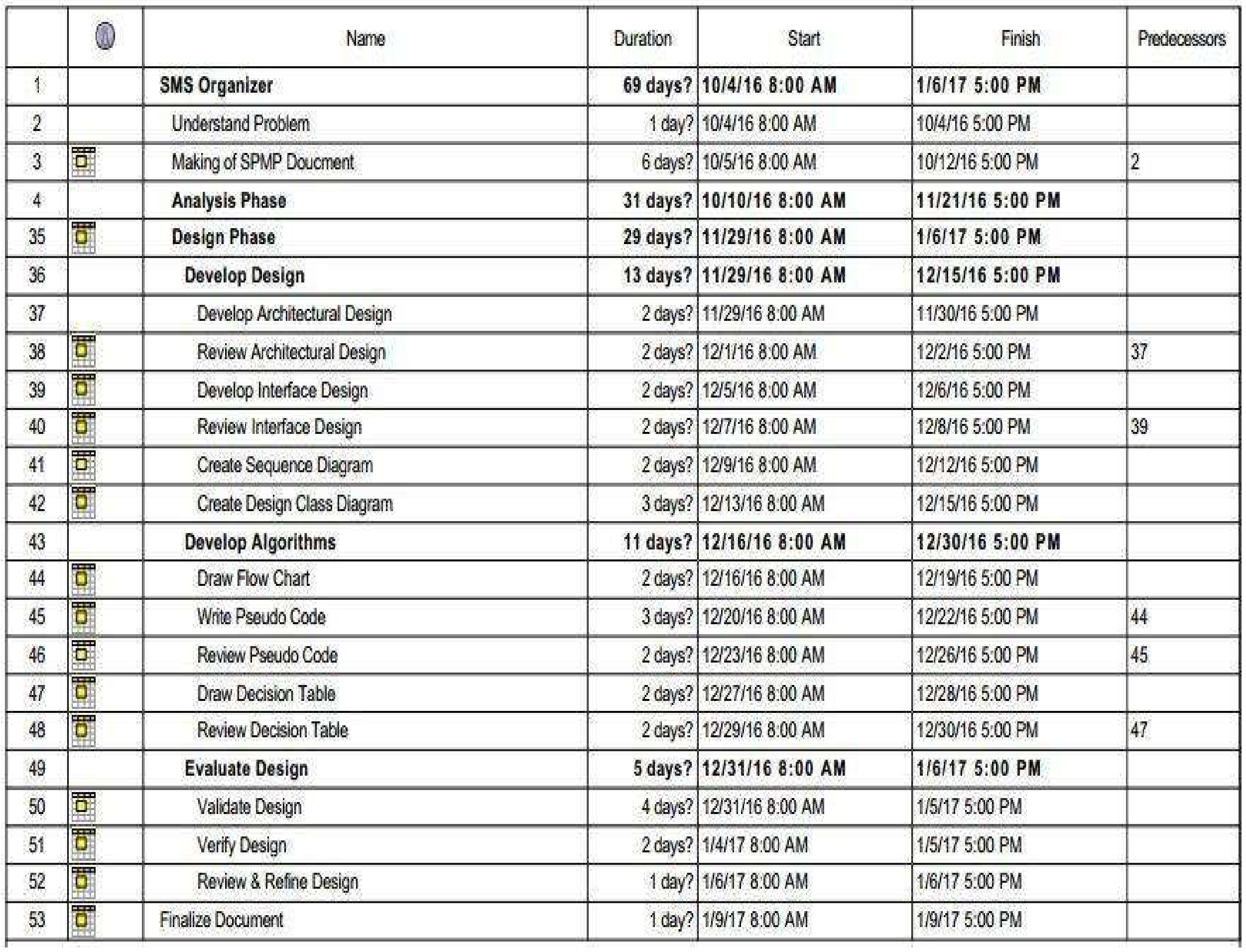
1.3.1 Tasks

Analysis and Design Phases tasks are divided into sub tasks:



	ⓘ	Name	Duration	Start	Finish	Predecessors
1		**SMS Organizer**	**69 days?**	**10/4/16 8:00 AM**	**1/6/17 5:00 PM**	
2		Understand Problem	1 day?	10/4/16 8:00 AM	10/4/16 5:00 PM	
3	▦	Making of SPMP Doucment	6 days?	10/5/16 8:00 AM	10/12/16 5:00 PM	2
4		**Analysis Phase**	**31 days?**	**10/10/16 8:00 AM**	**11/21/16 5:00 PM**	
5	▦	Gather Requirements	2 days?	10/10/16 8:00 AM	10/11/16 5:00 PM	
6	▦	Refine Requirements	2 days?	10/12/16 8:00 AM	10/13/16 5:00 PM	5
7	▦	Making of Document V1	1 day?	10/14/16 8:00 AM	10/14/16 5:00 PM	5;6
8	▦	**Identify Specific Requirements**	**2 days?**	**10/17/16 8:00 AM**	**10/18/16 5:00 PM**	
9		**External Interface Requirements**	**2 days?**	**10/17/16 8:00 AM**	**10/18/16 5:00 PM**	
10		User Interface	1 day?	10/17/16 8:00 AM	10/17/16 5:00 PM	
11		Hardware Interface	1 day?	10/17/16 8:00 AM	10/17/16 5:00 PM	
12		Software Interface	1 day?	10/17/16 8:00 AM	10/17/16 5:00 PM	
13		Communication Protocols	1 day?	10/17/16 8:00 AM	10/17/16 5:00 PM	
14	▦	Making of Document V2	1 day?	10/18/16 8:00 AM	10/18/16 5:00 PM	7;10;11;12;13
15		**Software Product Features**	**12 days?**	**10/18/16 8:00 AM**	**11/2/16 5:00 PM**	
16	▦	Identiy Use Cases	2 days?	10/18/16 8:00 AM	10/19/16 5:00 PM	
17	▦	Refine Use Cases	8 days?	10/20/16 8:00 AM	10/31/16 5:00 PM	16
18	▦	Making of Document V3	2 days?	11/1/16 8:00 AM	11/2/16 5:00 PM	14;16;17
19		**Software System Functions**	**4 days?**	**11/2/16 8:00 AM**	**11/7/16 5:00 PM**	
20	▦	Identify System Functions	2 days?	11/2/16 8:00 AM	11/3/16 5:00 PM	
21	▦	Refine System Functions	1 day?	11/4/16 8:00 AM	11/4/16 5:00 PM	20
22	▦	Making of Document V4	1 day?	11/7/16 8:00 AM	11/7/16 5:00 PM	18;20;21
23	▦	**Identify Software System Attributes**	**2 days?**	**11/8/16 8:00 AM**	**11/9/16 5:00 PM**	
24		Reliability	1 day?	11/8/16 8:00 AM	11/8/16 5:00 PM	
25		Availability	1 day?	11/8/16 8:00 AM	11/8/16 5:00 PM	
26		Security	1 day?	11/8/16 8:00 AM	11/8/16 5:00 PM	
27		Maintainability	1 day?	11/8/16 8:00 AM	11/8/16 5:00 PM	
28		Portability	1 day?	11/8/16 8:00 AM	11/8/16 5:00 PM	
29	▦	Making of Document V5	1 day?	11/9/16 8:00 AM	11/9/16 5:00 PM	22;24;25;26...
30	▦	**Database Requirements**	**2 days?**	**11/10/16 8:00 AM**	**11/11/16 5:00 PM**	
31		Identify Database Requirements	1 day?	11/10/16 8:00 AM	11/10/16 5:00 PM	
32		Making of Document V6	1 day?	11/11/16 8:00 AM	11/11/16 5:00 PM	29;31
33	▦	**Making of Final Document**	**6 days?**	**11/14/16 8:00 AM**	**11/21/16 5:00 PM**	
34		Refining Final Document	6 days?	11/14/16 8:00 AM	11/21/16 5:00 PM	32
35	▦	**Design Phase**	**29 days?**	**11/29/16 8:00 AM**	**1/6/17 5:00 PM**	

Figure 1.1 SPMP Analysis Phase



	ⓘ	Name	Duration	Start	Finish	Predecessors
1		**SMS Organizer**	**69 days?**	**10/4/16 8:00 AM**	**1/6/17 5:00 PM**	
2		Understand Problem	1 day?	10/4/16 8:00 AM	10/4/16 5:00 PM	
3	📅	Making of SPMP Doucment	6 days?	10/5/16 8:00 AM	10/12/16 5:00 PM	2
4		**Analysis Phase**	**31 days?**	**10/10/16 8:00 AM**	**11/21/16 5:00 PM**	
35	📅	**Design Phase**	**29 days?**	**11/29/16 8:00 AM**	**1/6/17 5:00 PM**	
36		**Develop Design**	**13 days?**	**11/29/16 8:00 AM**	**12/15/16 5:00 PM**	
37		Develop Architectural Design	2 days?	11/29/16 8:00 AM	11/30/16 5:00 PM	
38	📅	Review Architectural Design	2 days?	12/1/16 8:00 AM	12/2/16 5:00 PM	37
39	📅	Develop Interface Design	2 days?	12/5/16 8:00 AM	12/6/16 5:00 PM	
40	📅	Review Interface Design	2 days?	12/7/16 8:00 AM	12/8/16 5:00 PM	39
41	📅	Create Sequence Diagram	2 days?	12/9/16 8:00 AM	12/12/16 5:00 PM	
42	📅	Create Design Class Diagram	3 days?	12/13/16 8:00 AM	12/15/16 5:00 PM	
43		**Develop Algorithms**	**11 days?**	**12/16/16 8:00 AM**	**12/30/16 5:00 PM**	
44	📅	Draw Flow Chart	2 days?	12/16/16 8:00 AM	12/19/16 5:00 PM	
45	📅	Write Pseudo Code	3 days?	12/20/16 8:00 AM	12/22/16 5:00 PM	44
46	📅	Review Pseudo Code	2 days?	12/23/16 8:00 AM	12/26/16 5:00 PM	45
47	📅	Draw Decision Table	2 days?	12/27/16 8:00 AM	12/28/16 5:00 PM	
48	📅	Review Decision Table	2 days?	12/29/16 8:00 AM	12/30/16 5:00 PM	47
49		**Evaluate Design**	**5 days?**	**12/31/16 8:00 AM**	**1/6/17 5:00 PM**	
50	📅	Validate Design	4 days?	12/31/16 8:00 AM	1/5/17 5:00 PM	
51	📅	Verify Design	2 days?	1/4/17 8:00 AM	1/5/17 5:00 PM	
52	📅	Review & Refine Design	1 day?	1/6/17 8:00 AM	1/6/17 5:00 PM	
53	📅	Finalize Document	1 day?	1/9/17 8:00 AM	1/9/17 5:00 PM	

Figure 1.2 SPMP Design Phase

1.3.2 Analysis

1. **Description**
   The initial requirements are gathered and analyzed. After that use case, functional and nonfunctional requirements are extracted.
2. **Deliverables and Milestones**
   a. Software Project Management Plan (SPMP)
   b. Software Requirements Specification (SRS)
3. **Resources Needed**
   a. Workstation (Core processor minimum 4GB RAM)
   b. MS Office
   c. Internet

4. **Dependencies and Constraints**
    a. Listed in Figure 1.1
5. **Risks and Contingencies**
    a. Currently Not Available.

1.3.3 Design

1. **Description**
   Sequence Diagram and Class Diagram fall under design phase.
2. **Deliverables and Milestones**
    a. Software Design Description (SDD)
    b. Software Test Documentation (STD)
3. **Resources Needed**
    a. Workstation
    b. Internet Connection
    c. MS Office
    d. MS Visio, Pencil Tool (for design purpose)
4. **Dependencies and Constraints**
    a. Listed in Figure 1.2
5. **Risks and Contingencies**
    a. Currently not available

1.3.4 Assignments

All tasks are assigned to Muhammad Awais Rashid.

1.3.5 Timetable

Figure 1.1 & Figure 1.2 are showing the timetable.

2. System Requirements Specification

2.1 Introduction

It’s a mobile application used to organize SMS, create contact categories and set auto-response, while system can detect spam SMS using machine learning algorithm.

2.1.1 Product Overview

This product is an android based text messaging application that help users to organize their text messages by making categories. Also this application can control irrelevant messages by putting them in Others or Spam folder.

2.1.2 Background

Now a days, everybody have a mobile phone and the number of users are increasing day by day.

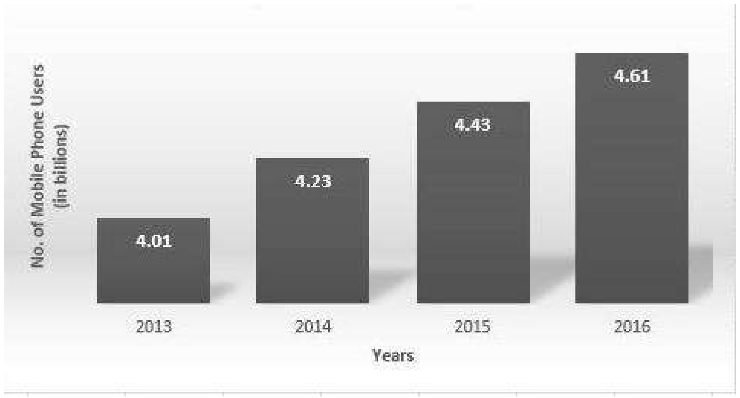


Figure 2.1 SMS Graph

In June, 2014, round about 561 billion text messages were sent worldwide. So roughly estimated daily text messages were 18.7 billion. By the end of 2011, mobile users in US sent 6 billion text messages every day, it means at that time US was sending out 180 billion text messages every month.

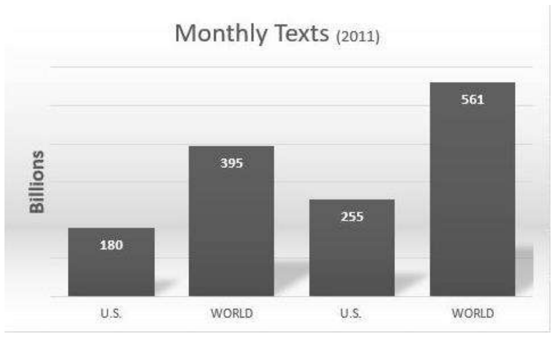


Figure 2.2 Monthly Texts

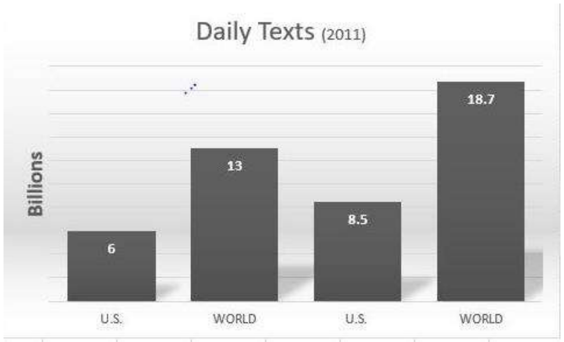


Figure 2.3 Daily Texts

2.1.3 Problem Statement

Many times irrelevant text messages from unknown numbers are present at the top and user gets confused between which to read and which to leave.
The purpose behind this project is to develop a text messaging mobile application. Users of this application can easily organize their text messages by making categories e.g. family, friends, and colleagues. Also there should be a pre-defined category for unknown numbers having name **Others**. In addition, this application can handle irrelevant messages (abusive, business marketing etc.), by placing them in a pre-defined category, similar to a spam folder present in email accounts by using machine learning algorithm.

2.1.4 Purpose

This mobile application will facilitate users to prioritize their contact directory.

2.2 Scope

This application is purely developed for android phones, the major functions are

2.2.1 Major Functions

User Functions

The major functions that a user can perform are

1. Read, forward and reply to Text Messages(s)
2. Modify message Categories
3. Delete text messages (incoming, outgoing)
4. Block Contact(s)
5. Group messaging

System Functions

The major functions on system end are

1. Put irrelevant text messages (abusive, business marketing) into a Spam folder using machine learning algorithm
2. Put text messages from unknown numbers (not in your contact directory) into Others folder
3. Put contact's text messages blocked by user in block category
4. Auto-reply text messages (user can make a specific text message for a specific person, present in contacts directory)

2.3 External Interface Requirements

2.3.1 User Interfaces

As described earlier, it's a smart phone application so all the android phones are compatible with this application. Display of this application is same on all phones, size of screen doesn't matter.

2.3.2 Hardware Interfaces

An android phone having 512MB RAM and 1.3 GHZ processor will be compatible with this application.

Note: these are the minimum requirements for this application

2.3.3 Software Interfaces

Android platform will be used for the development of this application. Android phone users from **Jellybean** (4.1) to **Nougat** (7.0) can easily install and use this application.

2.3.4 Communication Protocols

As described earlier, it is a text messaging application so SMPP (short message peer-to-peer protocol) will be used as a communication protocol. The SMPP is the true 'SMS Protocol' developed by the telecommunications industry specifically for SMS transmission.

2.4 Software Product Features

2.4.1 List of Use Cases

1. Send a Text Message
2. Reply to a Text Message
3. Forward a Text Message
4. Copy a Text Message
5. Create Category
6. Add Contact(s) to Category
7. Delete Contact(s) to Category
8. Delete Category
9. Rename Category
10. Delete a Text Message
11. Delete Whole Conversation
12. Block Contact
13. Unblock Contact
14. Set Auto Response
15. Change Theme

2.4.2 Use Case Model

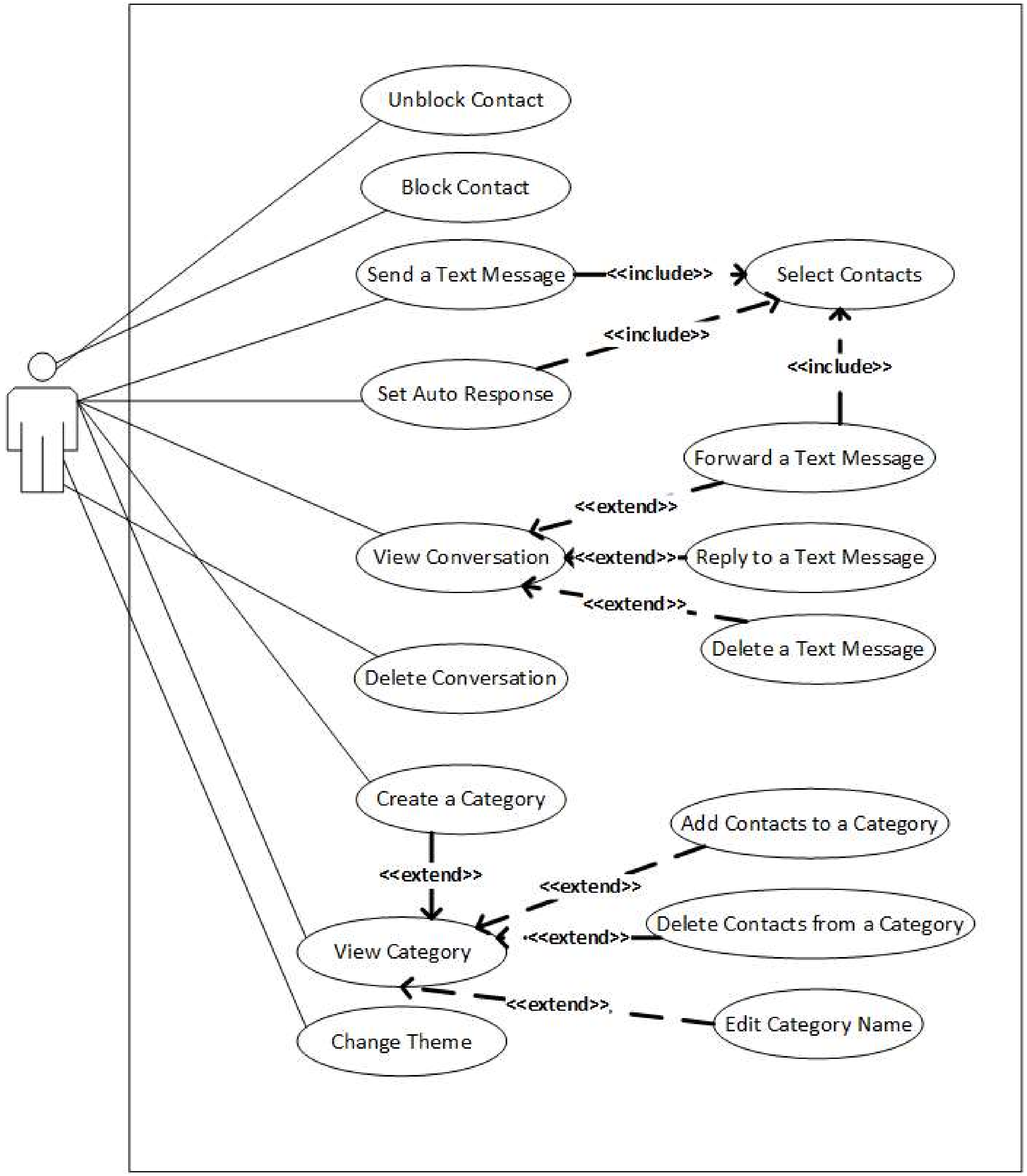


Figure 2.4 Use Case Model

2.5 Use Cases Detail

It’s very clear in all the use cases that whenever user wants to perform any function, installation of this application is a pre-condition and user have to open this application. Also first time, user have to authenticate via Gmail or can register its email.

2.5.1 Send a Text Message

Use Case ID	UC-01
Use Case Name	Send a Text Message
Primary Actor	User
Stakeholders & Interests	**User:** wants to create a new message and send to a particular person or a group
Pre-Conditions	
Post-Conditions	1. User has written the text message and sent to a particular person or a group 2. Text message is saved in application’s database
Main Success Scenario	1. User chooses option ‘Create Message’ 2. System shows ‘Create Message’ panel 3. User creates a new message 4. User clicks on select recipient(s) option 5. System displays recipients list present in contact directory 6. User selects recipient(s) 7. User clicks on send message option 8. System sends user’s text message 9. System confirms that ‘SMS send successfully’
Alternative Flows	2a. User don’t want to create a new message 1. User can use forward message option 6a. Required contact(s) are not available in directory 1. User can enter contact number itself 8-9a. System failed to send message 1. User hasn’t sufficient balance 2. Network error Note: In both cases, system will notify ‘Failed to Send SMS’
Frequency	Several times a day

Table 2.1 UC Send a Text Message

2.5.2 Reply to a Text Message

Use Case ID	UC-02
Use Case Name	Reply to a Text Message
Primary Actor	User
Stakeholders & Interests	**User:** wants to reply on a text message(s)
Pre-Conditions	To whom user wants to reply, its incoming text message should be found in application
Post-Conditions	User replied to a text message(s)
Main Success Scenario	1. To whom user wants to reply, user selects that contact by opening its chat head 2. System displays particular chat head to user 3. User type text for that contact 4. User clicks on Send button 5. System sends user's text message 6. System confirms user that SMS send successfully
Alternative Flows	1a. The text message of a person, user wants to reply, not found in application 1. User has to use 'Create Message' option rather than reply 3a. User want to send empty message 1. System prompts user 'Do you want to send empty message?' if yes, user will confirm otherwise user will cancel that option and type text 5-6a. System failed to reply message 1. User hasn't sufficient balance 2. Network error Note: In both cases, system will notify 'Failed to Send SMS'
Frequency	Several times a day

Table 2.2 UC Reply to a Text Message

2.5.3 Forward a Text Message

Use Case ID	UC-03
Use Case Name	Forward a Text Message
Primary Actor	User
Stakeholders & Interests	1. **User:** wants to forward stored messages(incoming or outgoing)
Pre-Conditions	1. A text which a user wants to forward should be present in application
Post-Conditions	1. Particular text message forwarded by the user
Main Success Scenario	1. User view text messages (incoming or outgoing) 2. User selects a particular text message to forward 3. System shows particular text message 4. User chooses option add recipient(s) 5. System shows recipient(s) list present in contact directory 6. User selects particular recipient(s) 7. User clicks on Send button 8. System sends user's message 9. System confirms that 'send SMS successfully'
Alternative Flows	2a. A particular text message which user wants to forward not found 6a. Particular recipient contact not found in contact directory 1. User will add contact number itself 8-9a. System failed to forward text message 1. Insufficient credit 2. Network error Note: In both cases, system will notify 'Failed to Send SMS'
Frequency	Several times a day

Table 2.3 UCD Forward a Text Message

2.5.4 Copy a Text Message

Use Case ID	UC-04
Use Case Name	Copy a Text Message
Primary Actor	1. User
Stakeholders & Interests	1. **User:** wants to use incoming or outgoing text messages for creation of new text message or other purpose
Pre-Conditions	1. A text which a user wants to copy should be present in application
Post-Conditions	1. Particular text copied by the user
Main Success Scenario	1. User view text messages (incoming or outgoing) 2. User selects copy message option 3. User copies particular text message 4. User use copied text for other operations (utilize in creating a new message)
Alternative Flows	3a. A text message which a user wants to copy not found
Frequency	1. Depends upon need of a user

Table 2.4 UCD Copy a Text Message

2.5.5 Create Category

Use Case ID	UC-05
Use Case Name	Create Category
Primary Actor	1. User
Stakeholders & Interests	1. **User:** wants to create a category of its own choice in order to add related contacts from phone directory
Pre-Conditions	1. Name of category should be unique
Post-Conditions	1. A new category is created 2. New category info is saved in system

Main Success Scenario	1. User chooses option ‘Create Category’ 2. User enters the name of category 3. User finalizes newly created category 4. System creates new category 5. System confirms that ‘New Category Created’
Alternative Flows	4-5a. System fails to create a new category 1. Category name is not unique
Frequency	1. Initially continuous i.e. when application is newly installed after that It will depend upon the need of a user

Table 2.5 UCD Create Category

2.5.6 Add Contact(s) to Category

Use Case ID	UC-06
Use Case Name	Add Contact(s) to Category
Primary Actor	User
Stakeholders & Interests	**User:** wants to add contacts to already created categories
Pre-Conditions	1. Particular category in which user wants to add contact(s) should be present in application 2. Adding contact(s) to one category shouldn’t be a part of any other category
Post-Conditions	Contact(s) added to category
Main Success Scenario	1. User selects particular category to add contact(s) 2. System opens category 3. User selects add contact(s) option 4. System displays contact list to the user 5. User selects contact(s) from directory 6. System add selected contacts to particular category
Alternative Flows	1a. Particular category not found 1. User have to make category first 6a. System failed to add contacts to category 1. May be particular contact present in other category
Frequency	Depends upon the need of a user

Table 2.6 UCD Add Contact(s) to Category

2.5.7 Delete Contact(s) from Category

Use Case ID	UC-07
Use Case Name	Delete Contact(s) from Category
Primary Actor	User
Stakeholders & Interests	**User:** wants to delete contact(s) from categories
Pre-Conditions	Particular category and contact(s) should be present in application
Post-Conditions	Particular contact deleted from category
Main Success Scenario	1. User selects particular category 2. System show all contacts present in category to the user 3. User selects contact(s) from directory to delete 4. System delete contact(s) from category
Alternative Flows	1a. Particular category not found 4a. System failed to delete contacts from category 1. Particular contact not found in category
Frequency	Depends upon the need of a user

Table 2.7 UCD Delete Contact(s) from Category

2.5.8 Delete Category

Use Case ID	UC-08
Use Case Name	Delete Category
Primary Actor	User
Stakeholders & Interests	**User:** wants to delete a particular category
Pre-Conditions	Particular category should be present in application
Post-Conditions	Particular category deleted
Main Success Scenario	1. User selects particular category 2. User selects delete option 3. System deletes particular category

	4. System confirms that category deleted
Alternative Flows	1a. Particular category not found
Frequency	Depends upon the need of a user

Table 2.8 UCD Delete Category

2.5.9 Rename Category

Use Case ID	UC-09
Use Case Name	Rename Category
Primary Actor	User
Stakeholders & Interests	**User:** wants to rename a particular category
Pre-Conditions	Particular category should be present in application
Post-Conditions	Particular category renamed
Main Success Scenario	1. User selects particular category 2. User selects rename option 3. User renames particular category 4. System confirms that category renamed
Alternative Flows	1a. Particular category not found 4a. System fails to rename category 1. New name of category is not unique 2. New name of category contains irrelevant characters or empty
Frequency	Depends upon the need of a user

Table 2.9 UCD Rename Category

2.5.10 Delete a Text Message

Use Case ID	UC-10
Use Case Name	Delete a Text Message
Primary Actor	User

Stakeholders & Interests	**User:** wants to delete text message (incoming or outgoing)
Pre-Conditions	Particular message which a user wants to delete should be present in application
Post-Conditions	Particular message deleted
Main Success Scenario	1. User selects particular text message to be deleted 2. User selects delete option 3. System asks for confirmation 4. User confirms delete option 5. System confirms that particular messaged deleted
Alternative Flows	1a. Particular text message not found
Frequency	Depends upon the need of a user

Table 2.10 UCD Delete a Text Message

2.5.11 Delete Whole Conversation

Use Case ID	UC-11
Use Case Name	Delete Whole Conversation
Primary Actor	User
Stakeholders & Interests	**User:** wants to delete whole conversation of a particular person
Pre-Conditions	Particular conversation, which a user wants to delete should be present in application
Post-Conditions	Particular conversation deleted
Main Success Scenario	1. User selects particular conversation to be deleted 2. User selects delete option 3. System asks for confirmation 4. User confirms delete option 5. System confirms that particular conversation deleted
Alternative Flows	1a. Particular conversation not found
Frequency	Depends upon the need of a user

Table 2.11 UCD Delete Whole Conversation

2.5.12 Block Contact

Use Case ID	UC-12
Use Case Name	Block Contact
Primary Actor	User
Stakeholders & Interests	**User:** wants to block a particular contact present in contact directory
Pre-Conditions	1. Particular contact which a user wants to block should be present in directory 2. Particular contact shouldn't be in block category
Post-Conditions	Particular contact is blocked
Main Success Scenario	1. User selects particular contact 2. User selects block option 3. System asks for confirmation 4. User confirms 5. System confirms that particular contact blocked
Alternative Flows	1a. Particular contact not found in contact directory
Frequency	Depends upon the need of a user

Table 2.12 UCD Block Contact

2.5.13 Unblock Contact

Use Case ID	UC-13
Use Case Name	Unblock Contact
Primary Actor	User
Stakeholders & Interests	**User:** wants to unblock a particular contact present in block category
Pre-Conditions	1. Particular contact which a user wants to unblock should be present in block category 2. Particular contact should be in block category
Post-Conditions	Particular contact is unblocked

Main Success Scenario	1. User selects block category 2. System shows all the contacts present in blocked category 3. User selects a particular contact to unblock 4. System asks for confirmation 5. User confirms 6. System confirms that particular contact is now unblocked
Alternative Flows	3a. Particular contact not found in a block category
Frequency	Depends upon the need of a user

Table 2.13 UCD Unblock Contact

2.5.14 Set Auto Response

Use Case ID	UC-14
Use Case Name	Set Auto Response
Primary Actor	User
Stakeholders & Interests	**User:** wants to set auto response message for a particular person
Pre-Conditions	Particular contact for which a user wants to set auto-response should be present in contacts directory
Post-Conditions	Auto-response for particular contact is activated
Main Success Scenario	1. User selects option 'Set Auto Response' 2. User chooses option select contact 3. System shows all the contacts present in contact directory 4. User selects particular contact 5. User write text message for that contact 6. System asks for confirmation 7. User confirms 8. System confirms that auto response message for that contact is set
Alternative Flows	1a. Particular contact for which user wants to set auto-response message not found in contact directory 1. Add particular contact to directory
Frequency	Depends upon the need of a user

Table 2.14 UCD Set Auto Response

2.5.15 Change Theme

Use Case ID	UC-15
Use Case Name	Change Theme
Primary Actor	User
Stakeholders & Interests	**User:** wants to change theme provided by the application
Pre-Conditions	Particular theme which a user wants to activate should be present in application
Post-Conditions	Theme of application changed
Main Success Scenario	1. User goes to the settings portion 2. User chooses option 'Change Theme' 3. System shows all the themes 4. User selects desired theme 5. System asks for confirmation 6. User confirms 7. System confirms that new theme activated
Alternative Flows	1a. Particular theme which a user wants to activate not found
Frequency	Depends upon the need of a user

Table 2.15 UCD Change Theme

2.6 Software System Functions

2.6.1 Handle Spam & Unknown Messages

Input	Incoming Text Message
Output	Incoming Text Message in Spam folder
Pre-Conditions	System should have received incoming message
Post-Conditions	System put incoming message into Spam or Other category
Logic	1. First of all system checks whether the incoming text message is from known or unknown number 2. If SMS is from an unknown number, system will use machine learning algorithm. 3. After applying this algorithm system will decide either the text from this particular number falls in spam category or not 4. If yes, system will put this text message into Spam category 5. If no, system will put this text message into Others category

Table 2.16 SSF Handle Spam & Unknown Messages

2.6.2 Handle Known Contacts

Input	Incoming Text Message
Output	Incoming Text Message in Block Category
Pre-Conditions	System should have received incoming message
Post-Conditions	System put incoming message into Block Category
Logic	1. First of all system checks whether the incoming text message is from known or unknown number 2. If message is from known number, system will check whether the number is blocked or not 3. If number is blocked by the user, system put this text message into Block category 4. If number is not blocked by the user, system will check either this contact number is present in any category or not. 5. If yes, system will put this text message into its particular category.

Table 2.17 SSF Handle Known Contacts

2.6.3 Handle Auto-reply Message

Input	Incoming Text Message
Output	Auto-reply Message send
Pre-Conditions	System should have received incoming message
Post-Conditions	System will send auto-reply message
Logic	1. First of all system checks whether the incoming text message is from known or unknown number 2. If message is from known number, system will check whether user has set auto-response message for that contact or not 3. If yes, system will auto-reply to that contact number

Table 2.18 SSF Handle Auto-reply Message

2.7 Software System Attributes

2.7.1 Availability

After installation, users of this application can use it round the clock. There aren't such constraints that can affect the availability service of this application.

2.7.2 Security

As described earlier, this application will handle irrelevant messages by putting them into a Spam folder. Now a days there are many text messages that contain linked based texts and having security threats. So this application can secure user.

2.7.3 Maintainability

During the development period, all the code will be properly documented so that we can easily make changes and upgrade our application.

2.7.4 Portability

Our application will run on an android phone which is itself a portable device so user can use this app anywhere, at any time.

2.8 Database Requirements

Application is maintaining categories, handling spam messages and dealing with contacts present in directory so a permanent storage is required. For that I chose Firebase.

Firebase stores and sync data with NoSQL cloud **database**. Data is synced across all clients in realtime, and remains available when your app goes offline. The **Firebase**Realtime **Database** is a cloud-hosted **database**. Data is stored as JSON and synchronized in realtime to every connected client.

2.9 Domain Model

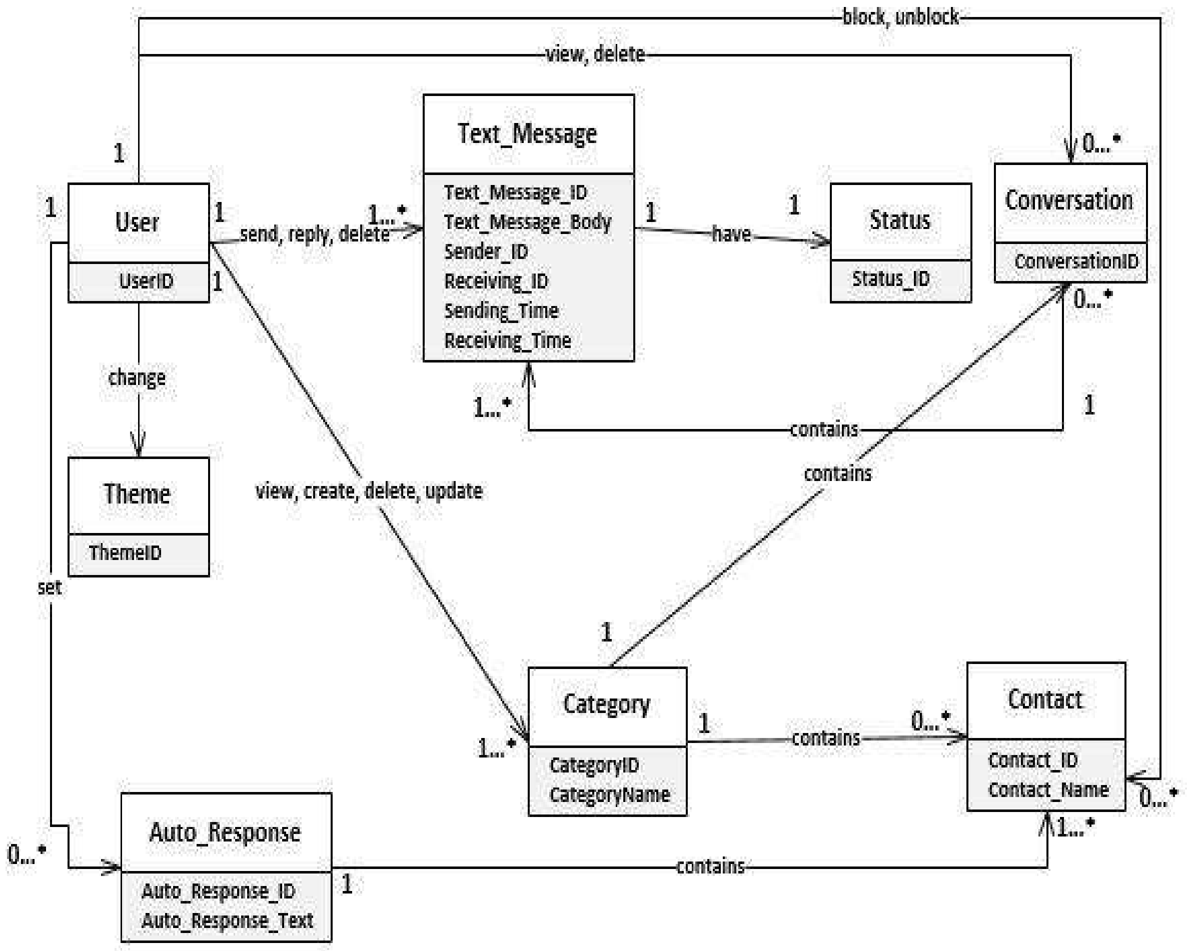


Figure 2.5 Domain Model

3. Software Design Description

3.1 Introduction

A software design description is a written description of a software product, that a software designer writes in order to give a software development team overall guidance to the architecture of the software project.

3.2 System Architecture Design

A system architecture is a conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

3.2.1 Chosen System Architecture

Three-tier architecture is a client–server software architecture pattern in which the user interface (presentation), functional process logic ("business rules"), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms.

3.2.2 Architecture Diagram

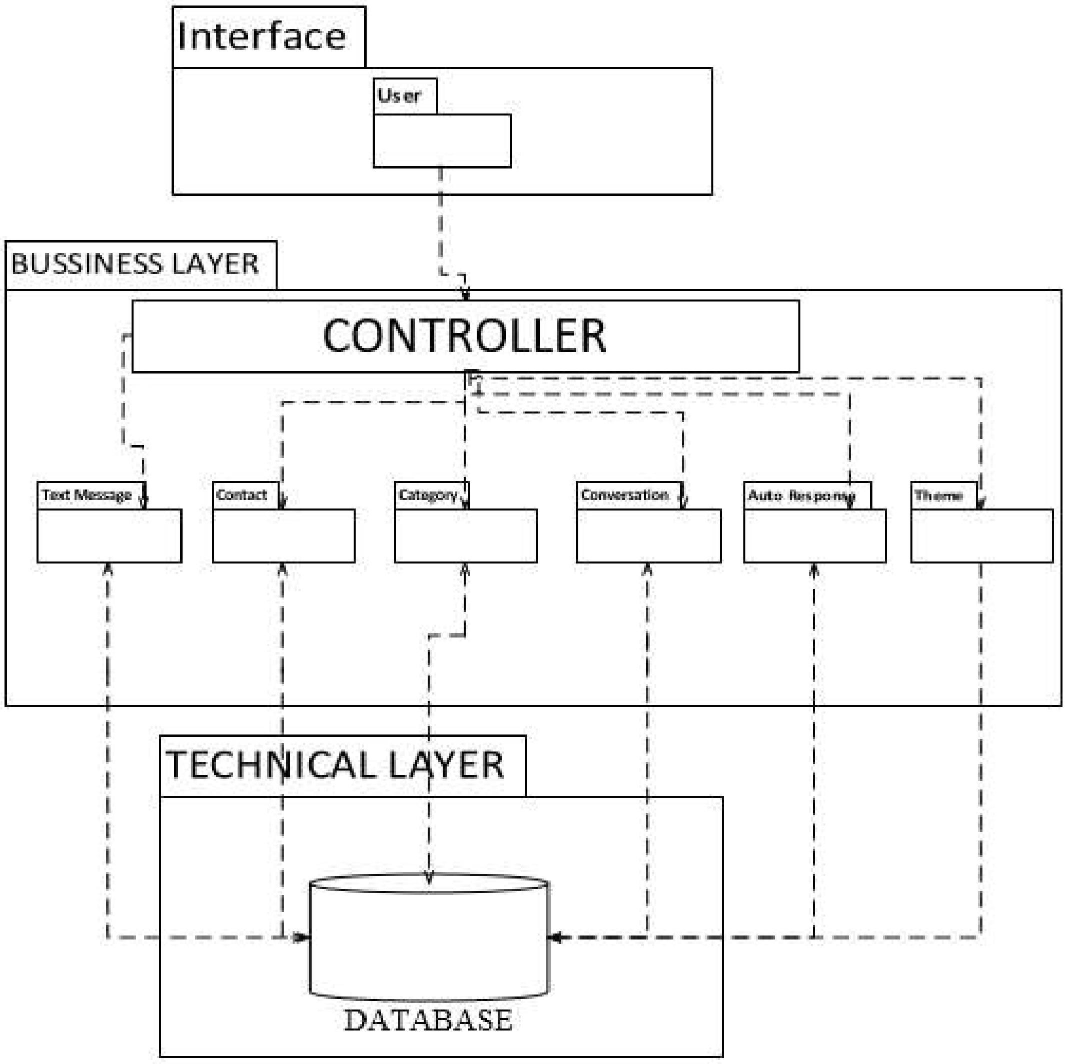


Figure 3.1 Architecture Diagram

3.3 Sequence Diagrams

It shows interaction between objects, arranged in time sequence. Also, it shows objects and classes involved in the scenario and the sequence of messages exchanged between the objects required to carry out the functionality of the scenario.

3.3.1 Send a Text Message

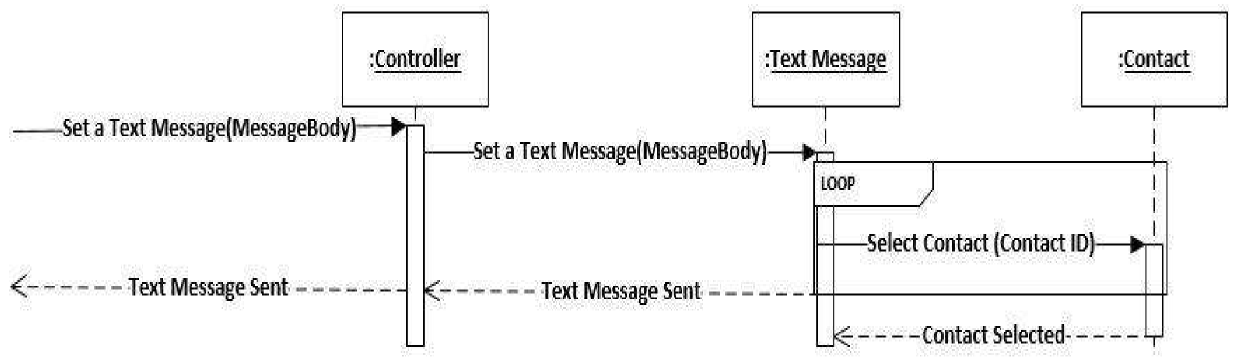

:Controller
:Text Message
:Contact
Set a Text Message(MessageBody)
Set a Text Message(MessageBody)
LOOP
Select Contact (Contact ID)
Text Message Sent
Text Message Sent
Contact Selected

Figure 3.2 Send a Text Message

3.3.2 Forward a Text Message

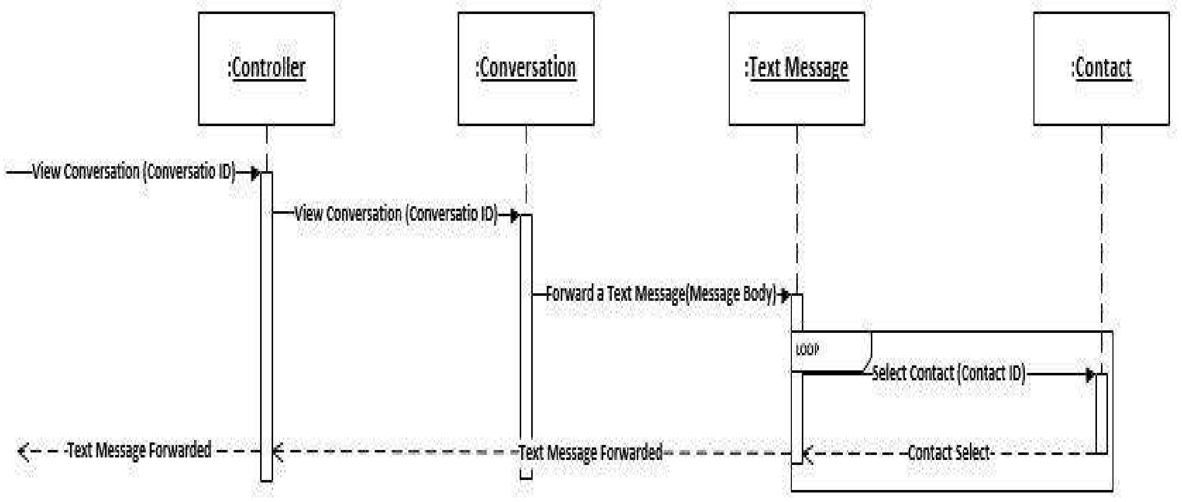

:Controller
:Conversation
:Text Message
:Contact
View Conversation (Conversatio ID)
View Conversation (Conversatio ID)
Forward a Text Message(Message Body)
LOOP
Select Contact (Contact ID)
Text Message Forwarded
Text Message Forwarded
Contact Select

Figure 3.3 SD Forward a Text Message

3.3.3 Reply to a Text Message

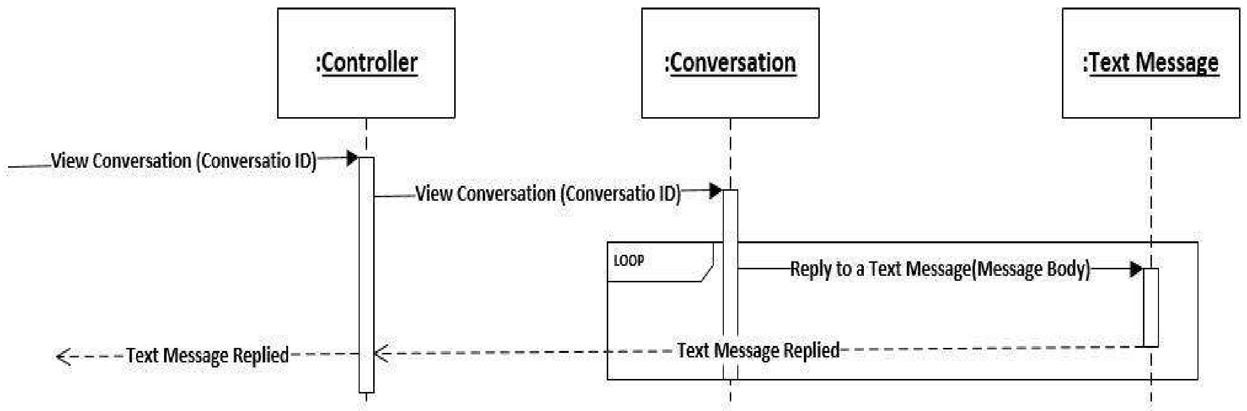

:Controller
:Conversation
:Text Message
View Conversation (Conversatio ID)
View Conversation (Conversatio ID)
LOOP
Reply to a Text Message(Message Body)
Text Message Replied
Text Message Replied

Figure 3.4 SD Reply to a Text Message

3.3.4 Delete a Text Message

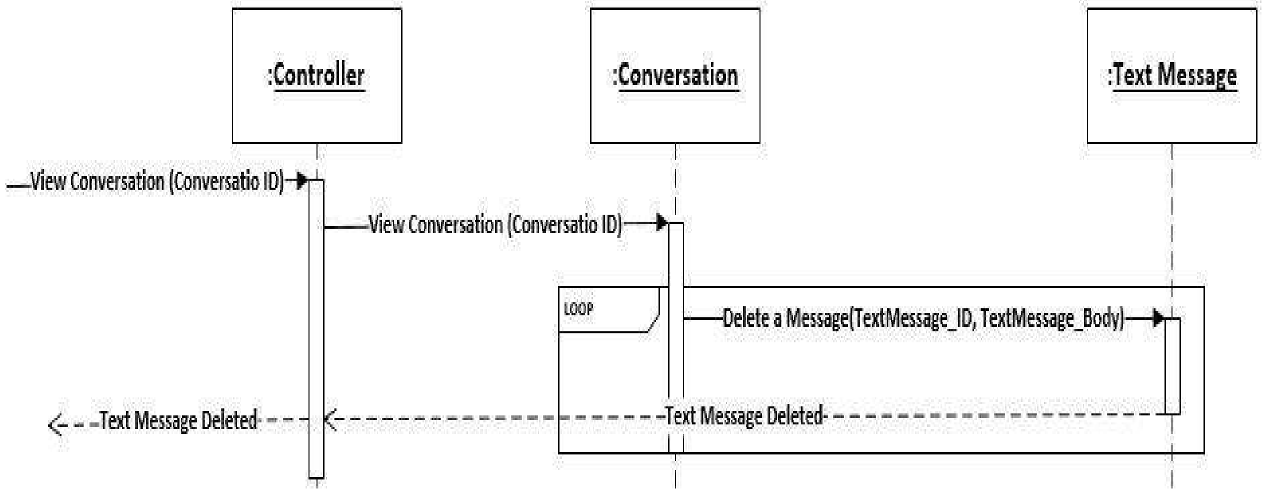

:Controller
:Conversation
:Text Message
View Conversation (Conversatio ID)
View Conversation (Conversatio ID)
LOOP
Delete a Message(TextMessage_ID, TextMessage_Body)
Text Message Deleted
Text Message Deleted

Figure 3.5 SD Delete a Text Message

3.3.5 Create a Category

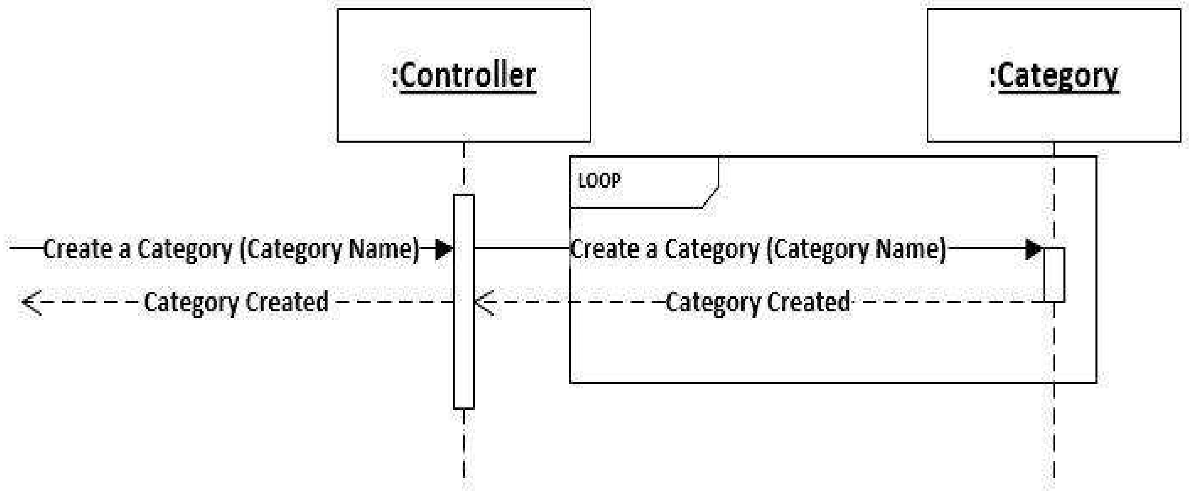

:Controller
:Category
LOOP
Create a Category (Category Name)
Create a Category (Category Name)
Category Created
Category Created

Figure 3.6 SD Create a Category

3.3.6 Add Contacts to a Category

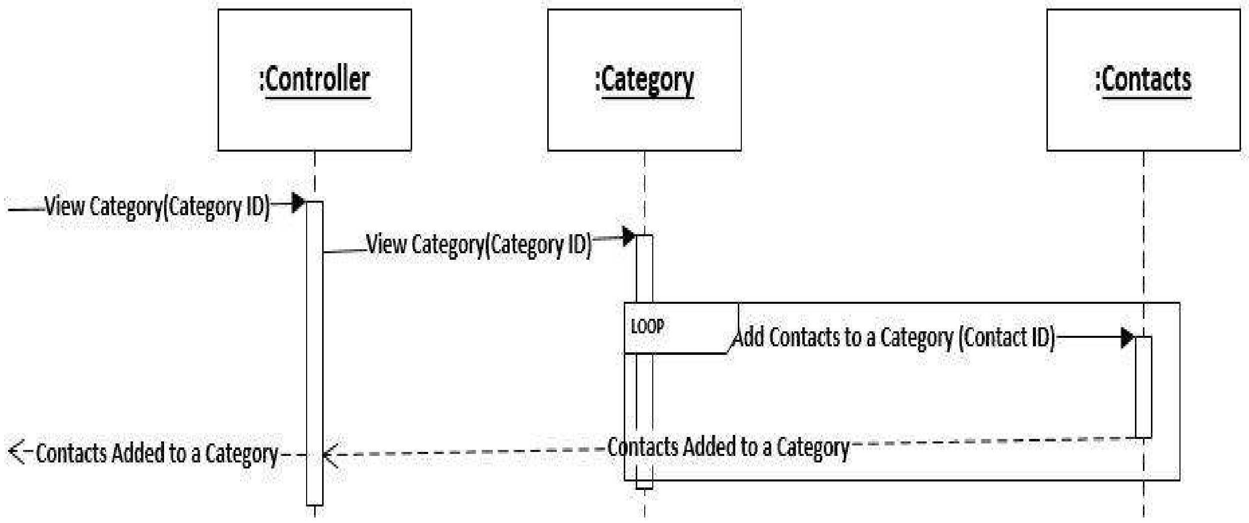

:Controller
:Category
:Contacts
View Category(Category ID)
View Category(Category ID)
LOOP
Add Contacts to a Category (Contact ID)
Contacts Added to a Category
Contacts Added to a Category

Figure 3.7 SD Add Contacts to a Category

3.3.7 Delete Contacts from a Category

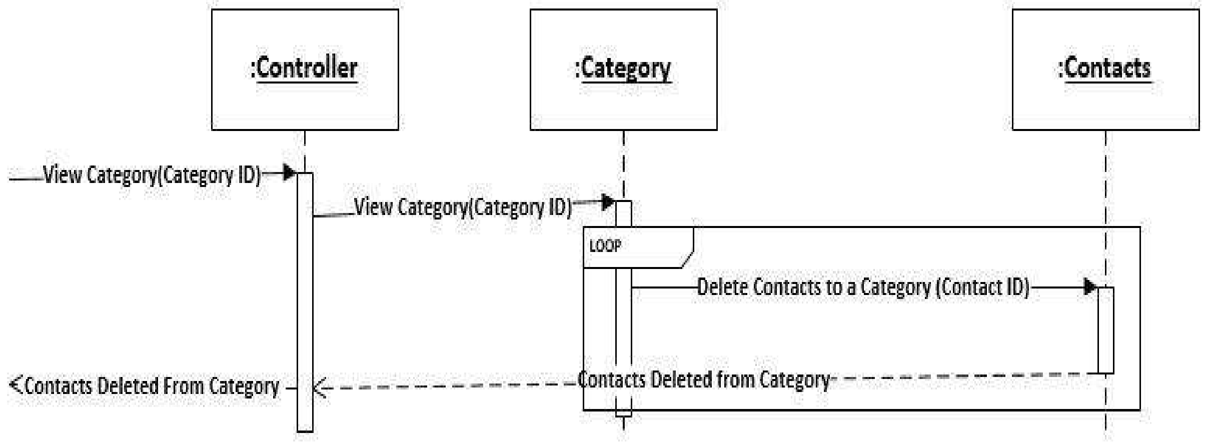

:Controller
:Category
:Contacts
View Category(Category ID)
View Category(Category ID)
LOOP
Delete Contacts to a Category (Contact ID)
Contacts Deleted from Category
Contacts Deleted From Category

Figure 3.8 SD Delete Contacts from a Category

3.3.8 View a Category

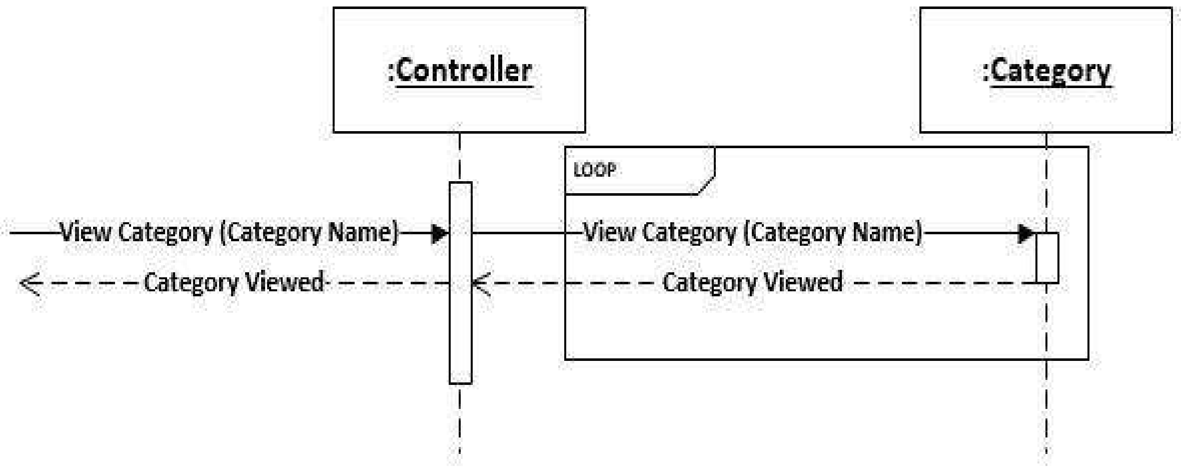


Figure 3.9 SD View a Category

3.3.9 Set Auto Response

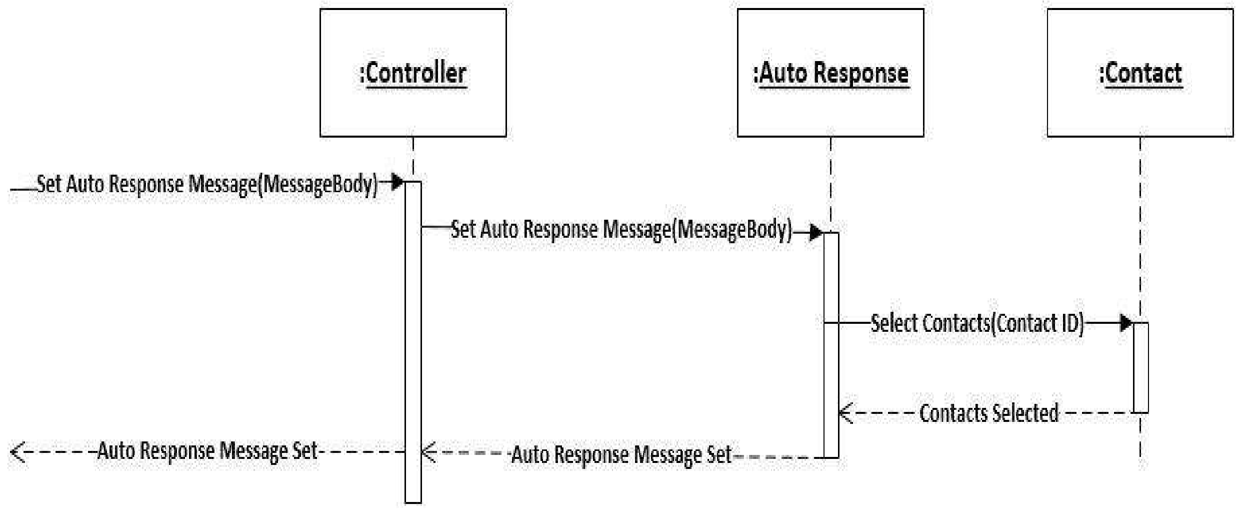

:Controller
:Auto Response
:Contact
Set Auto Response Message(MessageBody)
Set Auto Response Message(MessageBody)
Select Contacts(Contact ID)
Contacts Selected
Auto Response Message Set
Auto Response Message Set

Figure 3.10 SD Set Auto Response

3.3.10 Change Theme

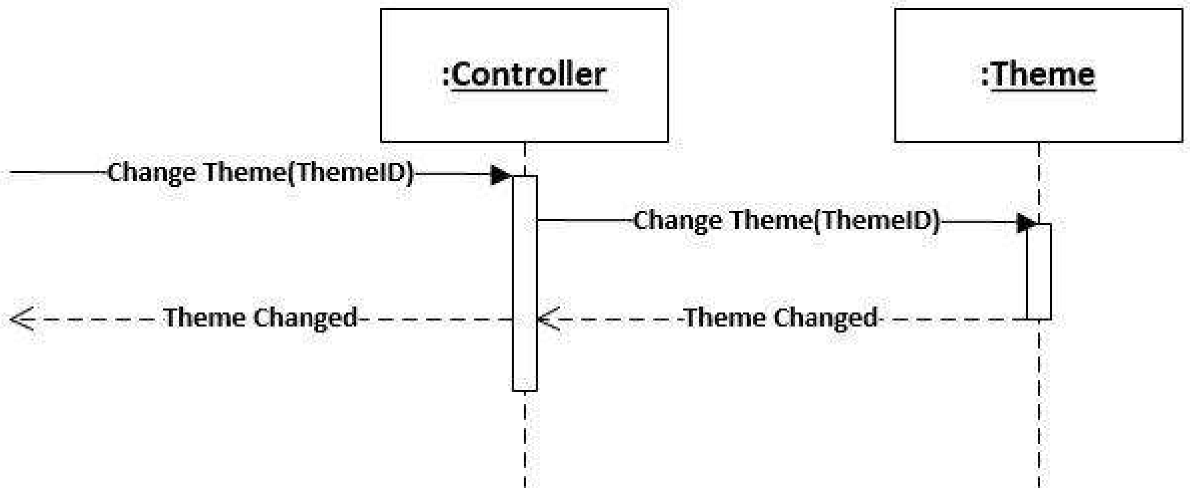


Figure 3.11 SD Change Theme

3.3.11 Block Contact

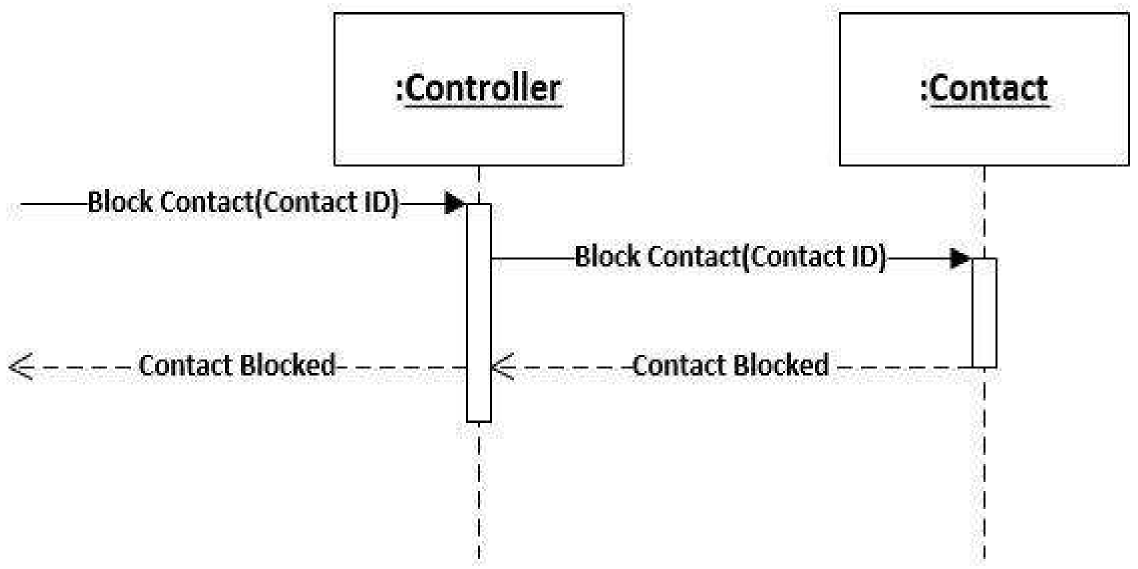


Figure 3.12 SD Block Contact

3.3.12 Unblock Contact

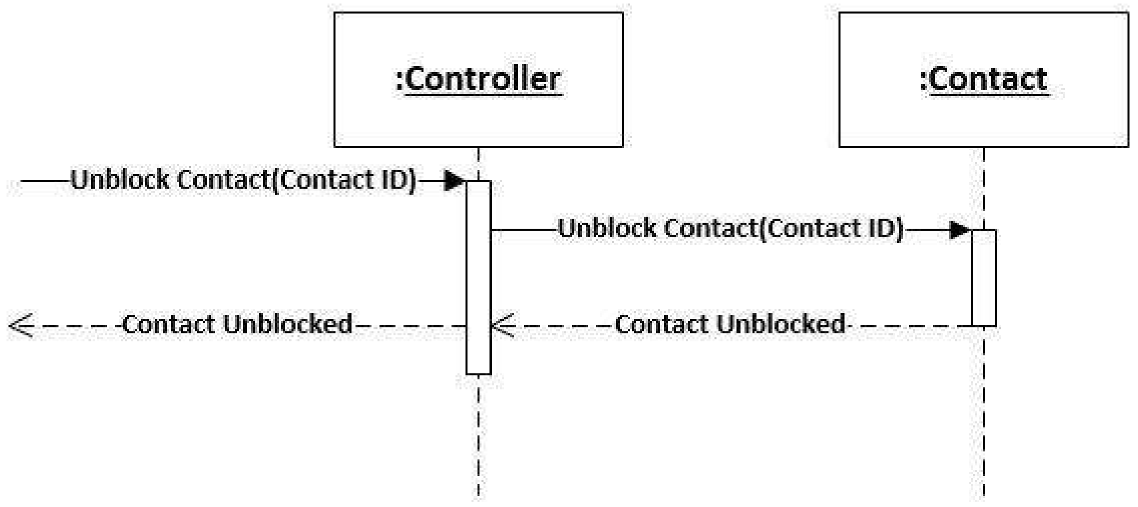


Figure 3.13 SD Unblock Contact

3.3.13 View Conversation

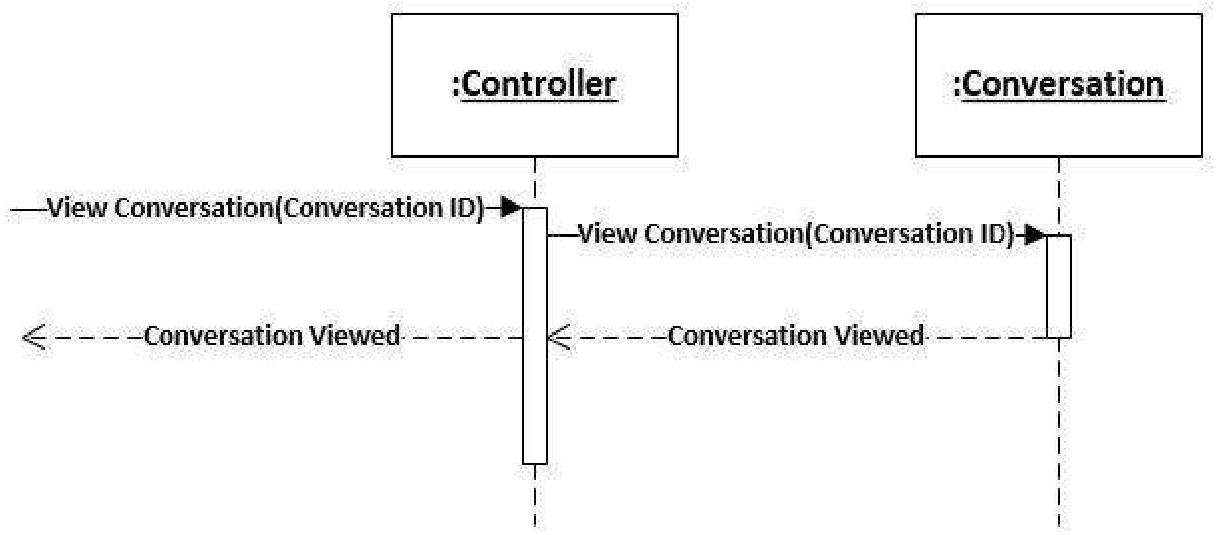

:Controller
:Conversation
View Conversation(Conversation ID)
View Conversation(Conversation ID)
Conversation Viewed
Conversation Viewed

Figure 3.14 SD View Conversation

3.3.14 Delete Conversation

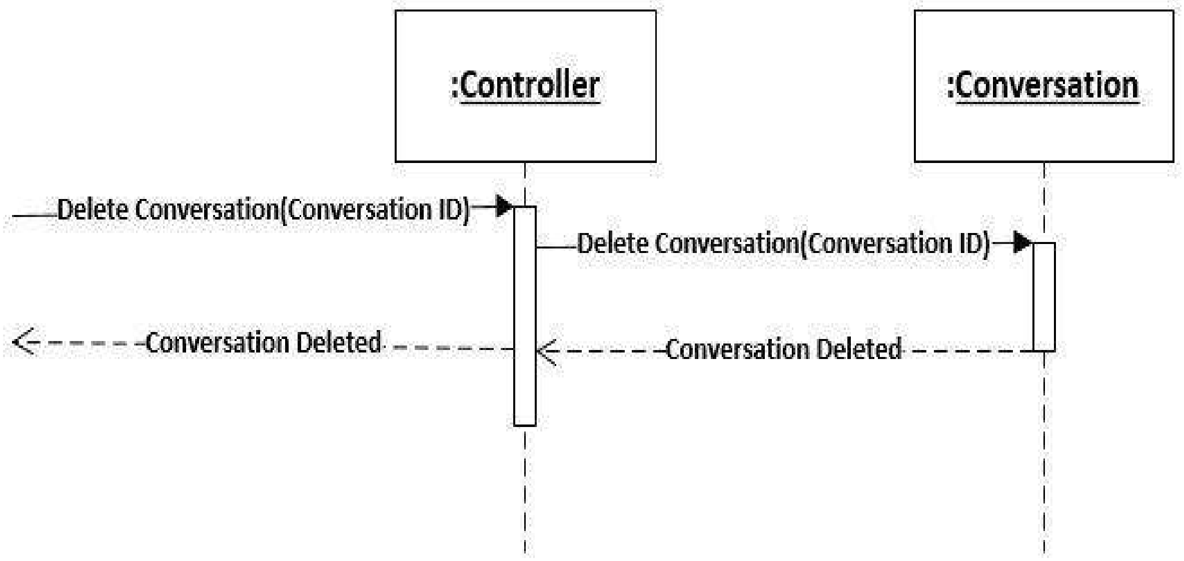

:Controller
:Conversation
Delete Conversation(Conversation ID)
Delete Conversation(Conversation ID)
Conversation Deleted
Conversation Deleted

Figure 3.15 SD Delete Conversation

3.4 Design Class Diagram

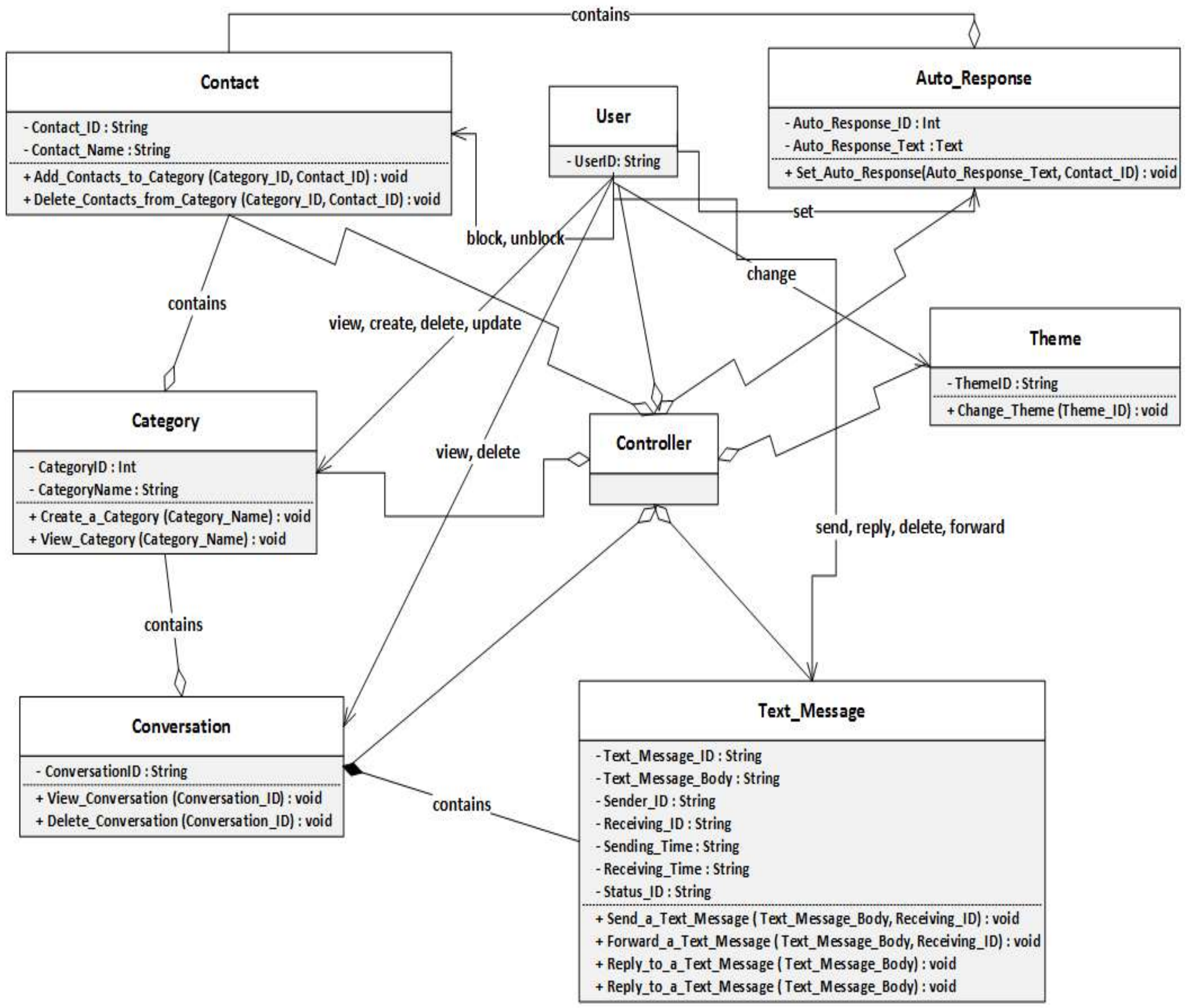

contains
Contact
- Contact_ID : String
- Contact_Name : String
+ Add_Contacts_to_Category (Category_ID, Contact_ID) : void
+ Delete_Contacts_from_Category (Category_ID, Contact_ID) : void
User
- UserID: String
Auto_Response
- Auto_Response_ID : Int
- Auto_Response_Text : Text
+ Set_Auto_Response(Auto_Response_Text, Contact_ID) : void
set
block, unblock
change
contains
view, create, delete, update
Theme
- ThemeID : String
+ Change_Theme (Theme_ID) : void
Category
- CategoryID : Int
- CategoryName : String
+ Create_a_Category (Category_Name) : void
+ View_Category (Category_Name) : void
Controller
view, delete
send, reply, delete, forward
contains
Conversation
- ConversationID : String
+ View_Conversation (Conversation_ID) : void
+ Delete_Conversation (Conversation_ID) : void
contains
Text_Message
- Text_Message_ID : String
- Text_Message_Body : String
- Sender_ID : String
- Receiving_ID : String
- Sending_Time : String
- Receiving_Time : String
- Status_ID : String
+ Send_a_Text_Message (Text_Message_Body, Receiving_ID) : void
+ Forward_a_Text_Message (Text_Message_Body, Receiving_ID) : void
+ Reply_to_a_Text_Message (Text_Message_Body) : void
+ Reply_to_a_Text_Message (Text_Message_Body) : void

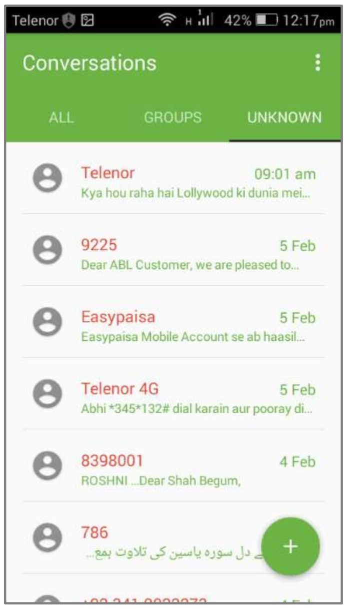
Figure 3.16 Design Class Diagram

3.5 User Interface Design

3.5.1 Description of the User Interface

As described earlier it is a mobile application so will show responsiveness on every android phone having variable screen sizes.

3.5.2 Screen Images



Telenor 09:01 am
Kya hou raha hai Lollywood ki dunia mei...

9225 5 Feb
Dear ABL Customer, we are pleased to...

Easypaisa 5 Feb
Easypaisa Mobile Account se ab haasil...

Telenor 4G 5 Feb
Abhi *345*132# dial karain aur pooray di...

8398001 4 Feb
ROSHNI ...Dear Shah Begum,

786
دل سورہ یاسین کی تلاوت بمع...

Figure 3.17 Unknown SMS

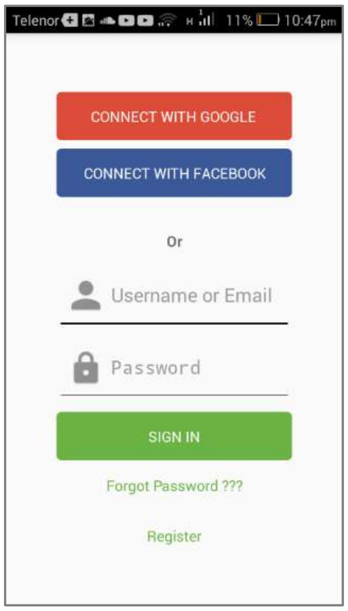


Figure 3.18 Login Screen

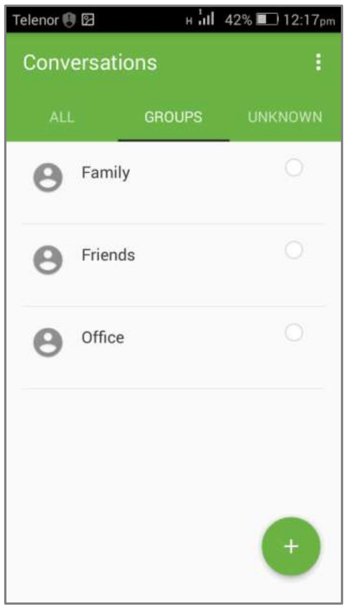
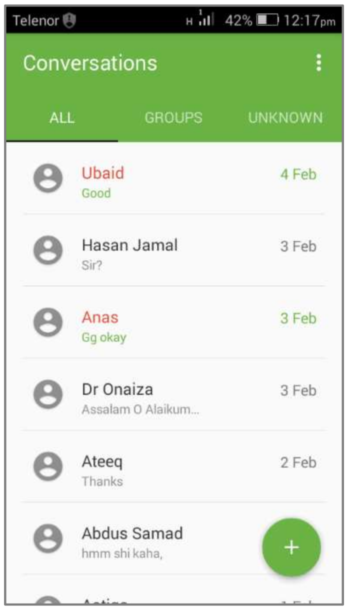


Figure 3.20 Contact Groups



Ubaid 4 Feb
Good

Hasan Jamal 3 Feb
Sir?

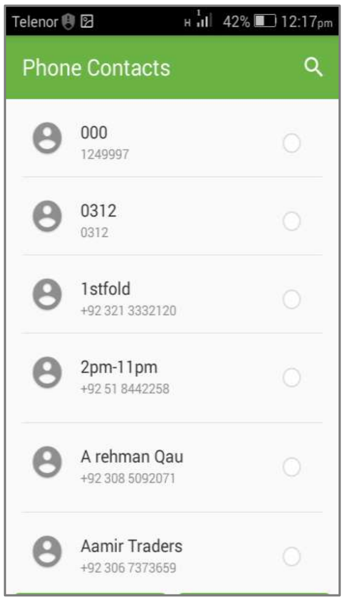
Anas 3 Feb
Gg okay

Dr Onaiza 3 Feb
Assalam O Alaikum...

Ateeq 2 Feb
Thanks

Abdus Samad
hmm shi kaha,

Figure 3.19 All SMS


Telenor 42% 12:17pm

Phone Contacts

000
1249997

0312
0312

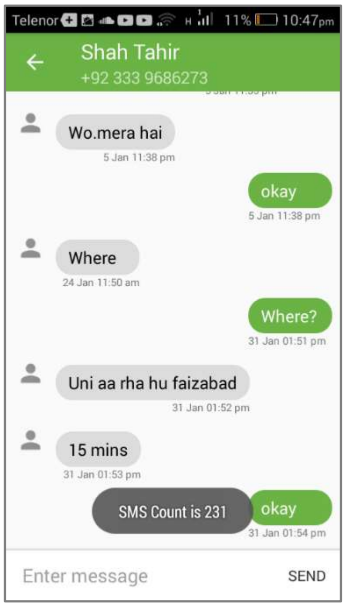
1stfold
+92 321 3332120

2pm-11pm
+92 51 8442258

A rehman Qau
+92 308 5092071

Aamir Traders
+92 306 7373659

Figure 3.22 Phone Contacts


Telenor 11% 10:47pm

Shah Tahir
+92 333 9686273

Wo.mera hai
5 Jan 11:38 pm

okay
5 Jan 11:38 pm

Where
24 Jan 11:50 am

Where?
31 Jan 01:51 pm

Uni aa rha hu faizabad
31 Jan 01:52 pm

15 mins
31 Jan 01:53 pm

SMS Count is 231

okay
31 Jan 01:54 pm

Enter message SEND

Figure 3.21 Detailed Conversation

4. Software Implementation & Testing Document

4.1 Introduction

This document actually describes the implementation details & testing of project.

4.1.1 Language Selection

Following are the programming languages used in the development of this project

1. **JAVA**
   Before 2016, Java was the only official language to develop native android apps, but in October 2017, Google has adopted Kotlin as its 2nd official programming language.
2. **Firebase**
   Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014.
   Firebase stores and sync data with NoSQL cloud **database**. Data is synced across all clients in real-time, and remains available when your app goes offline. The **Firebase Real-time Database** is a cloud-hosted **database**. Data is stored as JSON and synchronized in real-time to every connected client.
3. **XML**
   XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. XML tags are not predefined in XML. We must define our own Tags. Xml as itself is well readable both by human and machine. Also, it is scalable and simple to develop. In Android we use xml for designing our layouts because xml is lightweight language so it doesn’t make our layout heavy.

4.1.2 Tools Selection

Android Studio
As Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps.
It is developed by Google and JetBrains, its stable version is Android 3.0.1, written in Java. Windows, Linux & macOs are its compatible operating systems.

4.1.3 Resources

1. **Lib Phone Number (libphonenumber)**
   Google's common Java, C++ and JavaScript library for parsing, formatting, and validating international phone numbers.
2. **Fancy Buttons (Fancybuttons)**

Use to enhance android buttons

3. **Android Custom Checkbox (android-custom-checkbox)**
   Used to animate checkbox

4. **Leak Canary (leakcanary)**
   A memory leak detection library for Android and Java.

4.2 Algorithm Implementation

One of the major function of my application is to detect spam messages, i.e. if an incoming SMS is from unknown number, system will detect either that unknown SMS is a spam or not. For all that, I chose **Naïve Bayes Classifier**.

4.2.1 Naive Bayes Classifier

Naive Bayes is one of the simplest classifiers because very simple mathematical operations are involved and it's very easy to code with every standard programming language i.e. JAVA, PHP, C# etc.
This classifier is a simple probabilistic classifier based upon Bayes theorem having strong and naïve independence assumptions. It is one the most simple and basic text classification techniques with various applications in email spam detection, email sorting, documentation categorization, abusive content detection, language detection and sentiment detection. Also Naïve Bayes performs pretty well in many complex real-world problems.

Naïve Bayes classifier is very efficient because of the following facts

1. Requires small amount of training data
2. Training time is much smaller as compared to alternative methods
3. Less intensive in computation (in both CPU and memory)

Example

Examples	Text	Class
1.	I loved the movie	+
2.	I hated the movie	-
3.	A great movie, good movie	+
4.	Poor acting	-
5.	Great acting, great movie	+

Step 01:

Extract the unique words from the above examples

Unique Words

1.	I
2.	Loved
3.	The
4.	Movie
5.	Hated
6.	A
7.	Great
8.	Poor
9.	Acting
10.	Good

Step 02:

Convert the examples into feature sets, where the attributes are possible words, and the values are the number of times a word occurs in each example.

Sr.	I	Loved	The	Movie	Hated	A	Great	Poor	Acting	Good	Class
1.	1	1	1	1							+
2.	1		1	1	1						-
3.				2		1	1			1	+
4.								1	1		-
5.				1		1	1		1	1	+

Step 03:

Examples with positive outcomes

Sr.	I	Loved	The	Movie	Hated	A	Great	Poor	Acting	Good	Class
1	1	1	1	1							+
3				2		1	1			1	+
5				1		1	1		1	1	+

$p(+) = \frac{3}{5} = 0.6$ (Positive cases out of Total cases)

Let n be the number of words in the (+) case

In above case $n = 14$

Let n_k be the number of times a word k occurs in case (+)

$$p(w_k|+) = \frac{n_k + 1}{n + |Vocabulary|}$$

Using the above formula

$$p(I|+) = \frac{1+1}{14+10} = 0.0833 \qquad p(loved|+) = \frac{1+1}{14+10} = 0.0833$$

$$p(the|+) = \frac{1+1}{14+10} = 0.0833 \qquad p(movie|+) = \frac{5+1}{14+10} = 0.2083$$

$$p(a|+) = \frac{2+1}{14+10} = 0.125 \qquad p(great|+) = \frac{2+1}{14+10} = 0.125$$

$$p(acting|+) = \frac{1+1}{14+10} = 0.0833 \qquad p(good|+) = \frac{2+1}{14+10} = 0.125$$

$$p(hated|+) = \frac{0+1}{14+10} = 0.0417 \qquad p(poor|+) = \frac{1+1}{14+10} = 0.0417$$

Step 04:

Examples with Negative Outcomes

Sr.	I	Loved	The	Movie	Hated	A	Great	Poor	Acting	Good	Class
2	1		1	1	1						-
4								1	1		-

$p(-) = \frac{2}{5} = 0.4$ (Negative cases out of Total cases)

Let n be the number of words in the (-) case

In above case $n = 6$

Let n_k be the number of times a word k occurs in case (+)

$$p(w_k|-) = \frac{n_k + 1}{n + |Vocabulary|}$$

$$p(I|-) = \frac{1+1}{6+10} = 0.125 \qquad p(loved|-) = \frac{1+1}{6+10} = 0.125$$

$$p(the|-) = \frac{1+1}{6+10} = 0.0833 \qquad p(movie|-) = \frac{1+1}{6+10} = 0.125$$

$$p(a|-) = \frac{1+1}{6+10} = 0.125 \qquad p(great|-) = \frac{1+1}{6+10} = 0.125$$

$$p(acting|-) = \frac{0+1}{6+10} = 0.0625 \qquad p(good|-) = \frac{0+1}{6+10} = 0.0625$$

$$p(hated|-) = \frac{0+1}{6+10} = 0.0625 \qquad p(poor|-) = \frac{0+1}{6+10} = 0.0625$$

Step 05:

Now we have trained our classifier

Let's classify a new sentence

I hated the poor acting

$$v_+ = p(+)\, p(I|+)\, p(hated|+)\, p(the|+)\, p(poor|+)\, p(acting|+) = 6.03 \times 10^{-7}$$

$$v_- = p(-)\, p(I|-)\, p(hated|-)\, p(the|-)\, p(poor|-)\, p(acting|-) = 1.22 \times 10^{-5}$$

Here $v_- > v_+$

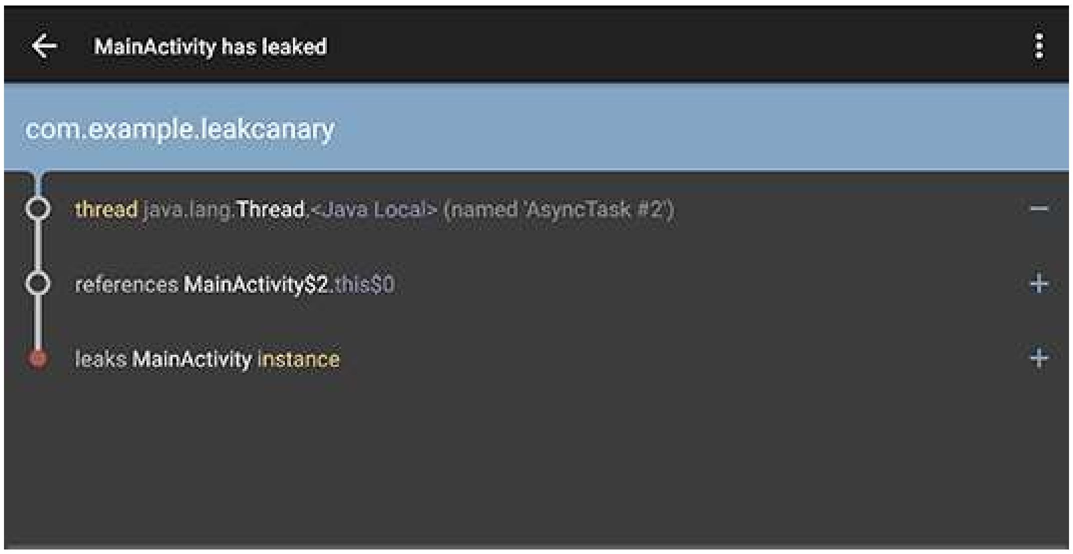
So, the above example is in (-) class

4.3 Memory Leakage

"*A small leak will sink a great ship.*" - Benjamin Franklin

Although Java has its own garbage collector but a garbage collector is not an insurance against memory leaks. On the other hand it's very difficult to keep track on memory leakage during development process for a newbie. So in order to avoid all memory leakages in android, an open-source library called **Leak Canary (LeakCanary)** is available on **GitHub**. All you need to do, is to just put the configuration code in **Application Class** (a class that loads first, when an app starts).

As you perform certain regular operations in your app, Leak Canary will notify you, which activity has a memory leakage and of what type.



```
MainActivity has leaked
com.example.leakcanary
thread java.lang.Thread.<Java Local> (named 'AsyncTask #2')
references MainActivity$2.this$0
leaks MainActivity instance
```

Figure 4.1 Memory Leakage (Leak Canary Notification)

5. Software Test Documentation

5.1 Introduction

A test case, in software engineering, is a set of conditions under which a tester will determine whether an application, software system or one of its features is working as it was originally established for it to do.

5.1.1 Test Strategy

A Test Strategy document is a high level document and normally developed by a project manager. This document defines "Software Testing Approach" to achieve testing objectives. Some companies include the "Test Approach" or "Strategy" inside the Test Plan, which is fine and it is usually the case for small projects.

5.2 Test Plan

A test plan is a document detailing the objectives, target market, internal beta team, and processes for a specific beta test for a software or hardware product. The plan typically contains a detailed understanding of the eventual workflow.

5.2.1 Testing Tools & Environment

A testing environment is a setup of software and hardware for the testing teams to execute test cases. In other words, it supports test execution with hardware, software and network configured. Test bed or test environment is configured as per the need of the Application under Test.

Here black box testing environment is used. Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance.

5.3 Test Cases

5.3.1 Send a Text Message

ID	TC 01
Description	User sends a text message
Tester	User
Setup	User with its user ID
Instructions:	1. Type a new text message 2. Enter recipients (from contact directory or itself) 3. Click on send button
Expected Results	Text message send with sender ID & recipient ID
Actual Results	As expected
Oracle	**Pass**

Table 5.1 TC Send a Text Message

5.3.2 Send a Text Message (Alternative Scenario)

ID	TC 02
Description	User can't send a text message
Tester	User
Setup	User with its user ID
Instructions:	1. Type a new text message 2. Enter recipients (from contact directory or itself) 3. Click on send button
Expected Results	1. Text message not sent 2. Either recipient's number is invalid or error in network
Actual Results	As Expected
Oracle	**Pass**

Table 5.2 TC Send a Text Message (Alternative Scenario)

5.3.3 Forward a Text Message

ID	TC 03
Description	User forwards a text message
Tester	User
Setup	User with its user ID
Instructions:	1. Select a text message which a user wants to forward 2. Enter recipients (from contact directory or itself) 3. Click on send button
Expected Results	Text message forwarded with sender ID & recipient ID
Actual Results	As Expected
Oracle	**Pass**

Table 5.3 TC Forward a Text Message

5.3.4 Forward a Text Message (Alternative Scenario)

ID	TC 04
Description	User can’t forwards text message
Tester	User
Setup	User with its user ID
Instructions:	1. Select a text message which a user wants to forward 2. Enter recipients (from contact directory or itself) 3. Click on send button
Expected Results	1. Text message not forwarded 2. Either recipient number is invalid or some network error
Actual Results	As Expected
Oracle	**Pass**

Table 5.4 TC Forward a Text Message (Alternative Scenario)

5.3.5 Delete a Text Message

ID	TC 05
Description	User deletes a text message
Tester	User
Setup	User with its userID
Instructions:	1. Select a text message which a user wants to delete 2. Select delete option 3. Confirm delete option
Expected Results	Text message deleted
Actual Results	As Expected
Oracle	**Pass**

Table 5.5 TC Delete a Text Message

5.3.6 Delete a Text Message (Alternative Scenario)

ID	TC 06
Description	User can’t delete a text message
Tester	User
Setup	User with its userID
Instructions:	1. Select a text message which a user wants to delete 2. Select delete option 3. Confirm delete option
Expected Results	Text message not deleted
Actual Results	As Expected
Oracle	**Pass**

Table 5.6 TC Delete a Text Message (Alternative Scenario)

5.3.7 Create a Category

ID	TC 07
Description	User creates a category
Tester	User
Setup	User with its userID
Instructions:	1. Select option ‘Create a Category’ 2. Enter category name 3. Click on ‘Create’ button
Expected Results	New category created
Actual Results	As Expected
Oracle	**Pass**

Table 5.7 TC Create a Category

5.3.8 Create a Category (Alternative Scenario)

ID	TC 08
Description	User can’t create a category
Tester	User
Setup	User with its userID
Instructions:	1. Select option ‘Create a Category’ 2. Enter category name 3. Click on ‘Create’ button
Expected Results	1. New category not created 2. Particular category already exists
Actual Results	As Expected
Oracle	**Pass**

Table 5.8 TC Create a Category (Alternative Scenario)

5.3.9 Add Contacts to a Category

ID	TC 09
Description	User add contacts to a category
Tester	User
Setup	User with its userID
Instructions:	1. Select particular category 2. Click on button 'Add Contacts' 3. Select contacts from contact directory 4. Click on confirm button
Expected Results	Contacts added to a particular category
Actual Results	As Expected
Oracle	**Pass**

Table 5.9 TC Add Contacts to a Category

5.3.10 Add Contacts to a Category (Alternative Scenario)

ID	TC 10
Description	User can't add contacts to a category
Tester	User
Setup	User with its user ID
Instructions:	1. Select particular category 2. Click on button 'Add Contacts' 3. Select contacts from contact directory 4. Click on confirm button
Expected Results	1. Contacts not added to a particular category 2. Particular contacts already exist in another category
Actual Results	As Expected
Oracle	**Pass**

Table 5.10 TC Add Contacts to a Category (Alternative Scenario)

5.3.11 Block Contact

ID	TC 11
Description	User block a particular contact
Tester	User
Setup	User with its user ID
Instructions:	1. Select option 'Bock Contact' 2. Click on button 'Add Contacts' 3. Select contacts from contact directory 4. Click on confirm button
Expected Results	Particular contact blocked
Actual Results	As Expected
Oracle	**Pass**

Table 5.11 TC Block Contact

5.3.12 Block Contact (Alternative Scenario)

ID	TC 12
Description	User can't block a particular contact
Tester	User
Setup	User with its user ID
Instructions:	1. Select option 'Bock Contact' 2. Click on button 'Add Contacts' 3. Select contacts from contact directory 4. Click on confirm button
Expected Results	1. Particular contact not blocked 2. Particular contact is already in block category
Actual Results	As Expected
Oracle	**Pass**

Table 5.12 TC Block Contact (Alternative Scenario)

6. Results, Conclusions & Future Enhancements

This chapter describes the results of a machine algorithm used for text classification and project conclusions.

For testing of Naive Bayes Classifier, I used K-Fold Cross Validation.

6.1 K-Fold Cross Validation

Cross-Validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.

In **K-Fold Cross-Validation**, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.

For classification problems, one typically uses stratified k-fold cross-validation, in which the folds are selected so that each fold contains roughly the same proportions of class labels.

In repeated cross-validation, the cross-validation procedure is repeated n times, yielding n random partitions of the original sample. The n results are again averaged (or otherwise combined) to produce a single estimation.

I took 100 sets of text messages, 33% are non-spam and 67% are spam.

For getting better results, I performed two types of testing

1. 10 Fold Testing (1 fold is for testing and 9 folds are for training)
2. 5 Fold Testing (1 fold is for testing and 4 folds are for training)

6.1.1 10 Fold Testing

	Testing Data (90 Sets)		**Training Data (10 Sets)**	
Folds	**Spam**	**Non Spam**	**Correctly Classified**	**Incorrectly Classified**
1.	60	30	9	1
2.	60	30	7	3
3.	60	30	10	0
4.	60	30	7	3
5.	60	30	6	4
6.	60	30	9	1
7.	60	30	8	2
8.	60	30	9	1
9.	60	30	8	2
10.	60	30	9	1

Average Results

- **Correctly Classified:** 82%
- **Incorrectly Classified:** 18%

6.1.2 5 Fold Testing

	Testing Data (80 Sets)		**Training Data (5 sets)**	
Fold	**Spam**	**Non Spam**	**Correctly Classified**	**Incorrectly Classified**
1.	53	27	80	20
2.	54	26	17	3
3.	54	26	15	5
4.	53	27	17	3
5.	54	26	16	4

Average Results

- **Correctly Classified :** 81%
- **Incorrectly Classified :** 19%

For testing, I used core java on Eclipse, as mobile processors are not so powerful & recommended for testing. I stored the obtained results obtained in a text file and used these results for testing unknown SMS on mobile phone.

6.2 Conclusions

So this application can perform the following functions

- Read, forward & reply to text message(s)
- Modify Categories
- Delete text messages (incoming, outgoing)
- Block Contacts
- Auto -response
- Detect Spam SMS

6.3 Future Enhancements

- As android has adopted **Kotlin** as a second official programming language after **JAVA**, so I have a plan to develop this application in **Kotlin** as well.
- Currently training of algorithm can’t be performed on a mobile device (as mobile processors aren’t so powerful) but in future training can be done on mobile phones also.

References

- C. Larman, Applying UML and Patterns 2nd edition
- Firebase Realtime Database CRUD Operation for Android https://www.simplifiedcoding.net/firebase-realtime-database-crud/
- Android Getting Started with Firebase – Login and Registration Authentication https://www.androidhive.info/2016/06/android-getting-started-firebase-simple-login-registration-auth/
- Android Login with Google Plus Account https://www.androidhive.info/2014/02/android-login-with-google-plus-account-1/
- Kenneth Burke, "How Many Texts Do People Send Every Day?" May 18, 2016, https://www.textrequest.com/blog/many-texts-people-send-per-day/
- 10 Fold Cross Validation (Naïve Bayes Classification) https://www.openml.org/a/estimation-procedures/1
- Fancy Buttons (Icons, Borders, Radius ... for Android buttons) https://github.com/medyo/Fancybuttons
- LeakCanary (A memory leak detection library for Android and Java) https://github.com/square/leakcanary
- Lib Phone Number (Google's common Java, C++ and JavaScript library for parsing, formatting, and validating international phone numbers) https://github.com/googlei18n/libphonenumber
- Fancy Buttons (Icons, Borders, Radius ... for Android buttons) https://github.com/medyo/Fancybuttons
- Android-custom-checkbox (Custom checkbox implementation for Android) https://github.com/iGenius-Srl/android-custom-checkbox