SMS SPAM DETECTION USING MACHINE LEARNING

A Main Project Report

submitted by

DEEPIKA BALAKRISHNAN C (MES20MCA-2015)

to

the APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the Degree

of

Master of Computer Applications

Department of Computer Applications

MES College of Engineering
Kuttippuram, Malappuram - 679 582

July 2022

DECLARATION

I undersigned hereby declare that the project report **SMS SPAM DETECTION USING MACHINE LEARNING**, submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala, is a bona fide work done by me under supervision of Muhammad Jabir C, Assistant Professor, Department of Computer Applications. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place:Kuttippuram

DEEPIKA BALAKRISHNAN C (MES20MCA-2015)

Date:06-07-2022

DEPARTMENT OF COMPUTER APPLICATIONS
MES COLLEGE OF ENGINEERING, KUTTIPPURAM

CERTIFICATE

This is to certify that the report entitled **SMS SPAM DETECTION USING MACHINE LEARNING** is a bona fide record of the Mini Project work carried out by **DEEPIKA BALAKRISHNAN C (MES20MCA-2015)** submitted to the APJ Abdul Kalam Technological University, in partial fulfillment of the requirements for the award of the Master of Computer Applications, under my guidance and supervision. This report in any form has not been submitted to any other University or Institution for any purpose.

Internal Supervisor(s)	External Supervisor(s)

Head Of The Department

Acknowledgements

My endeavor stands incomplete without dedicating my gratitude to a few people who have contributed towards the successful completion of my project.I pay my gratitude to the Almighty for his invisible help and blessing for the fulfillment of this work.At the outset I express our heartful thanks to our Head of the Department and guide Mr.Muhammad Jabir C, Assistant Professor of Computer Applications for his valuable guidance and supervision and project coordinator Mrs. Priya JD for her valuable support, timely advise and strict schedules to complete project. I also grateful to all teaching and non-teaching staff for their encouragement, guidance and whole-hearted support. Last but not least, I gratefully indebted to family and friends, who gave me a precious help in doing project.

DEEPIKA BALAKRISHNAN C (MES20MCA-2015)

Abstract

In the modern world where digitization is everywhere, SMS has become one of the most vital forms of communications, unlike other chatting-based messaging systems like Facebook, WhatsApp etc, SMS does not require active internet connection at all. As we all know that Hackers / Spammer tries to intrude in Mobile Computing Device, and SMS support for mobile devices had become vulnerable, as attacker tries to intrude to the system by sending unwanted link, with which on clicking those link the attacker can gain remote access over the mobile computing device. So, to identify those messages Authors have developed a system which will identify such malicious messages and will identify whether or not the message is SPAM or HAM (malicious or not malicious). Authors have created a dictionary using the TF-IDF Vectorizer algorithm, which will include all the features of words a SPAM SMS possess, based on content of message and referring to this dictionary the system will be classifying the SMS as spam or ham.

Keywords: SMS, SPAM and HAM, Machine Learning, TF-IDF Vectorizer, Text Classification

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

SMS is one of the most effective forms of communication. It is based on cellular communication systems, just the cell phone needs to be in the network coverage area in order to send or receive the message. Almost everyone is using this service for communication. Various organizations deal with SMS for communicating with their clients / customers, banks and other government organizations also use SMS for communication. Also, many business organizations use this service for advertising purposes. Thus, SMS is playing a vital role, as active internet connection is not required at all in this framework. So due to large usage of SMS, it has become one of the most favorite places for hackers and spammers. It is quiet easy for a hacker to compromise any one's cell phone just by passing or transmitting Malicious link to end user, the mobile device will automatically be compromised if end user click on the link or message being transmitted by hacker / spammer, and we can know the rest how a hacker can exploit the system if he gains control of the system. So it has become very much important to restrict the content which the end user is receiving. So there must be a system which could tell the end user whether the received message is SPAM or not, Non SPAM message is known as HAM. So by identifying the above mentioned problems and issues, authors have developed a system which can identify whether a Message is SPAM or HAM based on the content of the message using Machine Learning technique.

1.1.1 Motivation

For today's generation, usage of phones is not just confined to communication now but an array of different uses such as storing personal information like documents, notes, media, making financial transactions, shopping, etc. Owing to a wide range of information stored on devices some of which are personal and critical, hacking phones is of utmost interest to people having unethical intentions. SMS is an easy way to target people because it is used by people from all walks of life and all ages and they are not aware of the implications. Hackers access phone and all its information when a phone is hacked, and people have absolutely no idea about it. Consequently, there is a loss of critical data that can be exploited for illegal purposes. It can be traumatic for the victims causing psychological despair and financial loses.

1.2 Objective

The daily traffic of Short Message Service (SMS) keeps increasing. As a result, it leads to dramatic increase in mobile attacks such as spammers who plague the service with spam messages sent to the groups of recipients. Mobile spams are a growing problem as the number of spams keep increasing day by day even with the filtering systems. Spams are defined as unsolicited bulk messages in various forms such as unwanted advertisements, credit opportunities or fake lottery winner notifications. Spam classification has become more challenging due to complexities of the messages imposed by spammers. Hence, various methods have been developed in order to filter spams.

1.3 Report Organization

The project report is divided into five sections.
Section 2 describes literature survey.
Section 3 describes the methodology used for implementing the project.
Section 4 gives the results and discussions.
Finally Section 5 gives the conclusion.

Chapter 2

Literature Survey

A number of SMS Spam messages detection techniques are available these days like android apps to block spam messages, filtering spam messages using classification algorithms, etc. In this section, we will review the SMS Spam detection techniques by filtering spam messages based on feature selection using machine learning techniques.

El-Alfy and AlHasan. have proposed a model for filtering text messages for both email and SMS. They have analyzed different methods in order to finalize a feature set such that complexity can be reduced. They have used two classification algorithms i.e. Support Vector Machine (SVM) and Naïve Bayes and 11 features i.e. URLs, likely spam words, emotion symbols, special characters, gappy words, message metadata, JavaScript code, function words, recipient address, subject field and spam domain. They have evaluated their proposed model on five email and SMS datasets.

Jialin et al. have proposed a message topic model (MTM) for filtering Spam messages. Messages Topic Model (MTM) considers symbol terms, background terms and topic terms to represent spam messages and it is based on the probability guess of latent semantic analysis. They have used k-means algorithm to remove the sparse problem by training SMS spam messages into random irregular classes and then aggregating all SMS spam messages as a single file such that to capture word co-occurrence patterns.

Chan et al. have presented two methods for SMS Spam filtering i.e. feature reweighting method and good word attack. Both methods focus on the length of the message along with considering the weight of message. Good word attack focuses on deceiving the output of classifier by using least number of characters while for feature reweighting method they have introduced a new rescaling function for rescaling the weights. They have evaluated the experiment on two datasets i.e. SMS and comment.

Delany et al. discuss different approaches available for SMS Spam filtering and the problems associated with the dataset collection. They have analyzed a large dataset of SMS spam and used ten clusters i.e. ringtones, competitions, dating, prizes, services,finance, claims chat, voicemail and others.

Xu et al. have detected SMS Spam messages using content-less features. They have used 2 classification algorithms i.e. SVM and k-nearest neighbor (KNN) and feature set consisting of 3 features i.e. static, temporal and network for their experiment. They found that by combining temporal and network features SMS Spam messages can be detected more accurately and with good performance. Moreover, they also found the ways filter SMS Spam messages by using features that contain graph-topology and temporal information thus excluding the content of the message.

Nuruzzaman et al. used Text Classification techniques on independent mobile phones to evaluate their performance of filtering SMS spam. The training, filtering, and updating processes were performed on an independent mobile phone. Their proposed model was able to filter SMS spam with some good accuracy, less storage consumption, and good enough processing time without using a large amount of dataset or any support from computer.

Chapter 3

Methodology

3.1 Introduction

The project SMS Spam Detection Using Machine Learning two modules one is Admin module and second is User module. The admin can enter the data through the web into the database. Admin can add user, view users and manage users. The user can register through the application by giving the details. After logging into the application user moves to the homepage of the website. User can send friend requests, find friends and manage friend requests, chat with friends, view chats and reply the messages. User can view the chat messages is spam or ham through the status. When we read the chat messages, it checks whether the message is spam or ham by reading the complete data in the dataset. It read the data and place it in a data frame called trained data. Then we attach new message into the trained data. After that we preprocess the data in each rows in the data set .while preprocessing we remove the html tag and short notes, eliminate stop words, remove characters and numerical. After removing all these we get some keywords. we pass these keywords to a training section SVM to train it. After training we add currently sending messages into it and preprocess it.it checks the features taking from this message is matched with which all messages features is matching with this message in the dataset. Then we classify whether the message is ham or spam with the help of SVM. The output we are getting from the prediction is updated into the status field.

3.2 Work Flow

Data Collection:

In this phase authors have collected a dataset based on which they have performed the experimentation from Kaggle Repository.

Data Cleaning:

In this phase the authors have cleansed all the data which were taken into consideration. Authors have removed all the white-spaces, lowered the alphabet so that words like Equal and equal become the same, remove the remaining punctuation, like! is not that much important, tokenize each message, to represent the message as a list of words and done stemming, converting all the words to their root word, like floor, floored to floor.

Generating Testing and Training Data Sets:

Authors have created the testing and training data on the converted cleansed datasets.

Generating Word Cloud Vector:

Authors have used the TF-IDF vectorization for creating the word-vector. On the basis, the spam feature will be classified.

Prediction:

Authors have given input messages to check whether the message is SPAM or HAM.

Figure 3.1 shows the workflow or architectural layout of how authors have classified the SPAM.

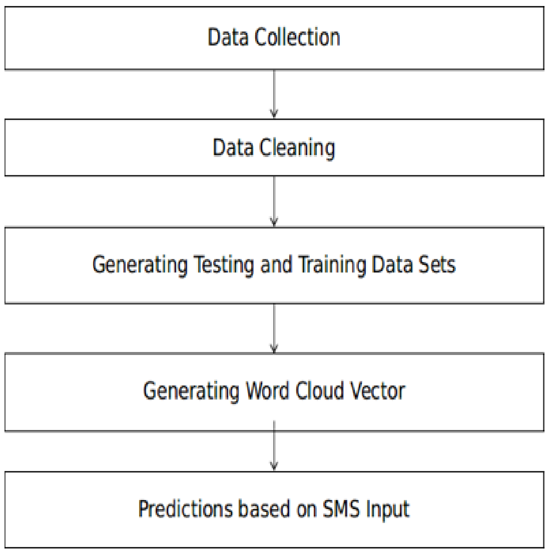

Data Collection
Data Cleaning
Generating Testing and Training Data Sets
Generating Word Cloud Vector
Predictions based on SMS Input

Figure 3.1: Proposed Process flow of the Entire Work

3.3 Machine Learning

Machine learning is a fascinating domain as it incorporates substantial parts of different fields namely statistics, artificial intelligence theory, data analytic and numerical methods. Machine learning can be defined as semi-automated extraction of knowledge from data sets or data.

Let's break down the definition into three component parts.

i) Firstly, machine learning always starts with data, with an objective to extract knowledge insight from the used data or data set.

ii) Secondly, machine learning involves a certain amount of automation rather than trying to gather insights from the data manually.

iii) Lastly, machine learning is not fully automated i.e. it requires human interventions to make many smart decisions for the process to be successful.

Simply we can put, machine learning is an application that can improve its prediction results with successive iterations or it improves with experience. The process of an application improving with experience is, naturally enough, called Training. It can take significant iterations to gradually improve results. During the process of training, data is given to a machine-learning algorithm, which then refines its internal representation, numerical parameters, as it encounters any deviations or Training errors. The purpose of this stage is to minimize cost function, error function or maximizing likelihood by adjusting the algorithm's internal weights. When the algorithm accuracy improves, we call this learning. Once the results are accurate enough also known as scoring, the machine-learning application can be deployed to solve the problem that it was supposed to.

Machine learning is broadly categorized into two categories:

a) Supervised Learning,

b) Unsupervised Learning.

3.3.1 Supervised Learning:

Supervised learning also known as predictive modelling, is the process of making predictions using data. Examples of Supervised learning are Classification and Regression. A supervised learning Training data set is pre labelled for classification problems or function values are known in case of regression. After training is done and the model has a minimum cost function for the training data set, later switch for scoring where we can predict values for new data.

Classification:

It identifies group membership. That means that if we have multiple events characterized by input parameters, which can be labelled differently, and we want our system to predict which label should be used.

Regression:

Regression is a combination of multi-dimensional power supply and function interpolation. The regression problem is used to find the approximation of the function with a minimum error deviation or a cost function. In other words, the regression technique simply tries to predict numeric dependence, a function value, for example, of a data set. Figure 1. Diagrammatically shows how supervised learning is to solve problems

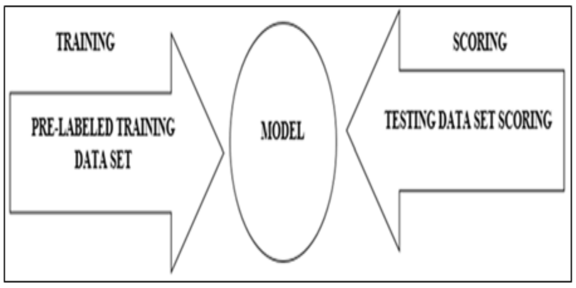


Figure 3.2: Supervised Learning

Example of supervised learning, if a system has a data set which is a series of email messages, supervised learning task is to predict whether each email message is spam or non-spam(ham). This is supervised learning because there is a specific outcome namely spam or ham.

3.3.2 Unsupervised Learning:

Unsupervised Learning is the process of extracting structure from data or how to best represent data. Examples of Unsupervised Learning are Clustering7 (is partitioning a data set into meaningful similar sub classes called cluster) and Association8 (method for discovering relations between existing attributes within a data set or data base). In an unsupervised learning situation, where the algorithm detects data features automatically, this depends on the purpose of the algorithm as well as the assumptions made on what the properties and observed values are. Figure 2. Describes how unsupervised learning is used to solve problems.

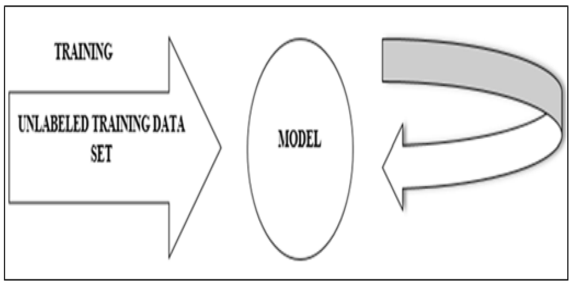


Figure 3.3: Unsupervised Learning

For example, if any data set was the characteristics and purchasing behavior of shoppers at grocery stores, the unsupervised learning task might be to segment the shoppers into groups or "clusters" that exhibit similar behaviors. Such learning methods might find that college students, parents with young children, and older adults have characteristic shopping behaviors that are similar within each group but dissimilar from the other. This is an unsupervised learning task because there is no right or wrong about how many clusters can be found in the data, which people belong in which cluster, or even how to describe each cluster.

Now after having a clear understanding of Machine Learning, authors have used the same in generating the rules, which will help in governing or identifying based on inputs whether or not the message is SPAM or HAM. For processing the document content authors have used TF-IDF.

3.4 Term Frequency Inverse Document Frequency (TF-IDF)

TF-IDF stands for Term Frequency Inverse Document Frequency, used in machine learning and text mining as a weighting factor for identifying words features. The weight increases as the word frequency in a document increases, i.e. weight increases, the more times a term occurs in the document, but that offset by the number of times the word appears, in the entire data set or this offset helps remove the importance from really common words like 'the' or 'a' appear quite often in all across the document. It is used very often in relevance ranking and scoring and to move stop words from ML Model, where these stop words don't give any relevant information about a particular document type or class. Figure represents the TF-IDF mathematical formula.

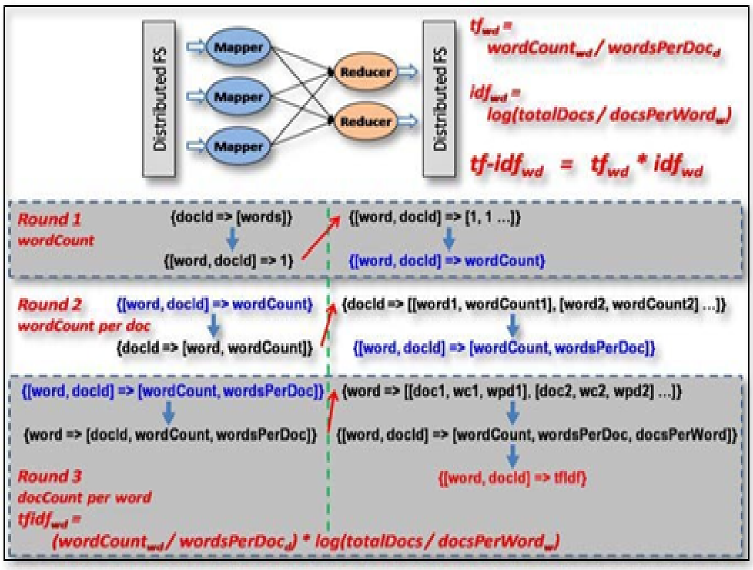

Distributed FS
Mapper
Mapper
Mapper
Reducer
Reducer
Distributed FS
tf_wd = wordCount_wd / wordsPerDoc_d
idf_wd = log(totalDocs / docsPerWord_w)
tf-idf_wd = tf_wd * idf_wd
Round 1
wordCount
{docId => [words]}
{[word, docId] => 1}
{[word, docId] => [1, 1 ...]}
{[word, docId] => wordCount}
Round 2
wordCount per doc
{[word, docId] => wordCount}
{docId => [word, wordCount]}
{docId => [[word1, wordCount1], [word2, wordCount2] ...]}
{[word, docId] => [wordCount, wordsPerDoc]}
{[word, docId] => [wordCount, wordsPerDoc]}
{word => [docId, wordCount, wordsPerDoc]}
{word => [[doc1, wc1, wpd1], [doc2, wc2, wpd2] ...]}
{[word, docId] => [wordCount, wordsPerDoc, docsPerWord]}
{[word, docId] => tfIdf}
Round 3
docCount per word
tfidf_wd = (wordCount_wd / wordsPerDoc_d) * log(totalDocs / docsPerWord_w)

Figure 3.4: Equation for calculating TF-IDF

3.5 Modules

There are two modules and a technical part in this application.

<u>1.ADMIN</u>

1.Login

2.View user list

3.Manage user

4.Block user

5.View blocked user

6.Remove Videos : Admin can remove videos that are uploaded.

7.Control Users : Admin can view and control users.

<u>2.USER</u>

1.Register

2.Login

3.Search friends

4.Send friend request

5.Manage friend request

6.View friend list

7.Chat with friends

8.View messages and replay message

9.View status (Spam / Ham)

3.TECHNICAL PART

1.Dataset

2.Training and model building

3.Data cleaning

4.Token generation

5.Spam core generation

6.Test accuracy

7.Output generation

3.6 Developing Environment

1.Hardware Requirements

- Processor : i3 and above
- RAM : 4GB
- Storage : 500GB Hard disk

2.Software Requirements

- Front End: HTML, CSS
- Back End: Python, MYSQL
- IDE: : JetBrains PyCharm
- FRAMEWORK USED: Flask

3.7 Agile Methodology

The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins,teams cycle through a process of planning,executing and evaluating. Continuous collaboration is vital.
There are two module in this project.

The module Admin has the userstories:
- Has Low Priority:
- Has Medium Priority: Login, View users
- Has High Priority: Manage Users, Block Users

The module User has the userstories:
- Has Low Priority:
- Has Medium Priority: Login, Registration, View Friends List
- Has High Priority: Chat With Friends, Manage Friends

This project is divided into 4 sprints:
Sprint 1
- Admin Login
- User Registration
- User Login

Sprint 2
- User Search friends
- User Send friend request
- User Manage friend request

Sprint 3

- User View friend list

- User Chat with friends

- Technical Part

Sprint 4

- User View messages and replay message

- User View status (Spam / Ham)

- Admin View user list

Sprint 5

- Admin Manage user

- Admin Block user

- Admin View blocked user

- Testing & Output generation

3.7.1 User Story

A user story is a tool used in agile software development to capture a description of a software feature from an end-user perspective. The user story describes the type of user, what they want and why. A user story helps to create a simplified description of a requirement. User stories are one of the core components of an agile program. They help provide a user-focused framework for daily work drives collaboration, creativity, and a better product overall. The user story is given below :

User StoryID	**As a <type of user>**	**I want to**	**So that I can**
1	Admin	Login	Login successful with correct username and password
2	Admin	View and Manage users	Admin can View all user details And manage users
3	Admin	Block users	Admin can Block users
4	Admin	View blocked users	Admin can View blocked users
5	User	Registration	User can register with this app
6	User	Login	Login successful with correct username and password
7	User	Search friends	User can Search friends profile
8	User	Send friend requests	User can Send requests to friends
9	User	Add and manage friend request	User can Accept or reject friend request
10	User	View friends list	User can View friends list
11	User	Chat	User can Chat with friends
12	User	View Chats and reply messages	User can View chats and reply friends messages
13	User	View status	User can View message is spam or ham through the status

Table 3.1: User Story

3.7.2 Product Backlog

A product backlog is a list of the new features, changes to existing features, bug fixes, infrastructure changes or other activities that a team may deliver in order to achieve a specific outcome.The priority of each user stories are given in the product backlog.

User Story ID	**Priority <High/ Medium/ Low>**	**Size (Hours)**	**Sprint <#>**	**Status <Planned/ In progress/ Completed**	**Release Date**	**Release Goal**
1	Medium	3	1	Completed	22/04/2022	Admin Login
2	High	4		Completed	27/04/2022	User Registration
3	Medium	2		Completed	3/05/2022	User Login
4	Medium	3	2	Completed	6/05/2022	User Search friends
5	High	5		Completed	13/05/2022	User Send friend request
6	High	3		Completed	18/05/2022	User Manage friend request
7	Medium	5	3	Completed	27/05/2022	User View friend list
8	Medium	2		Completed	31/05/2022	User Chat with friends
9	Medium	3		Completed	3/06/2022	Technical Part
10	High	5	4	Completed	10/06/2022	User View messages and replay message
11	Medium	2		Completed	14/06/2022	User View status (Spam / Ham)
12	Medium	3		Completed	17/06/2022	Admin View user list
13	High	3	5	Completed	22/06/2022	Admin Manage user
14	Medium	2		Completed	25/06/2022	Admin Block user
15	Medium	2		Completed	28/06/2022	Admin View blocked user
16	High	3		Completed	1/07/2022	Testing & Output generation

Table 3.2: Product Backlog

3.7.3 Project Plan

A project plan that has a series of tasks laid out for the entire project, listing task durations, responsibility assignments, and dependencies. Plans are developed in this manner based on the assumption that the Project Manager, hopefully along with the team, can predict up front everything that will need to happen in the project, how long it will take, and who will be able to do it.The project plan shows when each sprint starte

User StoryID	Task Name	Start Date	End Date	Days	Status
1	Sprint 1	20/04/2022	22/04/2022	3	Completed
2		24/04/2022	27/04/2022	4	Completed
3		2/05/2022	3/05/2022	2	Completed
4	Sprint 2	4/05/2022	6/05/2022	3	Completed
5		9/05/2022	13/05/2022	5	Completed
6		16/05/2022	18/05/2022	3	Completed
7	Sprint 3	23/05/2022	27/05/2022	5	Completed
8		30/05/2022	31/05/2022	2	Completed
9		1/06/2022	3/06/2022	3	Completed
10	Sprint 4	6/06/2022	10/06/2022	5	Completed
11		13/06/2022	14/06/2022	2	Completed
12		15/06/2022	17/06/2022	3	Completed
13	Sprint 5	20/06/2022	22/06/2022	3	Completed
14		24/06/2022	25/06/2022	2	Completed
15		27/06/2022	28/06/2022	2	Completed
16		29/06/2022	1/07/2022	3	Completed

Table 3.3: Project Plan

3.7.4 Sprint Plan

The sprint backlog is a list of tasks identified by the Scrum team to be completed during the Scrum sprint. During the sprint planning meeting, the team selects some number of product backlog items, usually in the form of user stories, and identifies the tasks necessary to complete each user story.Here the sprint backlog this project is given below :

Backlog Item	Status & completion date	Original estimate in hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14
UserStory #1,#2,#3																
Admin Login	22/04/2022	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
User Register	27/04/2022	4	0	0	0	0	1	1	1	1	0	0	0	0	0	0
User Login	3/05/2022	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1
UserStory #4,#5,#6																
Search friends	6/05/2022	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Send friend request	13/05/2022	5	0	0	0	0	0	0	1	1	1	1	1	0	0	0
Manage friends	18/05/2022	3	1	1	0	0	0	0	0	0	0	0	0	0	0	1
UserStory #7,#8,#9																
View friend list	27/05/2022	5	0	0	0	0	0	0	0	1	1	1	1	1	0	0
Chat with friends	31/05/2022	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Technical Part	1/06/2022	3	0	0	0	1	1	1	0	0	0	0	0	0	0	0
UserStory #10,#11,#12																
View & reply messages	10/06/2022	5	0	0	0	0	0	0	0	0	1	1	1	1	1	0
User View status	14/06/2022	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Admin view user list	17/06/2022	3	0	0	0	1	1	1	0	0	0	0	0	0	0	0
UserStory #13,#14,#15, #16																
Admin manage user	22/06/2022	3	0	0	0	0	0	0	0	0	1	1	1	0	0	0
Admin block user	25/06/2022	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Admin view blocked user	28/06/2022	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Testing & Output generation	1/07/2022	3	0	0	0	1	1	1	0	0	0	0	0	0	0	0
Total		50	3	6	5	3	4	3	2	3	4	4	4	2	3	3

Table 3.4: Sprint Plan

3.7.5 Sprint Actuals

The sprint actual is done based on the sprint plan. The sprint actual is divided into 4 sprints in which each of the tasks specified in the sprint backlog are done based on each sprint.Each sprint needs to be completed based on the dates in the sprint backlog.

Backlog Item	Status & completion date	Original estimate in hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14
UserStory #1,#2,#3																
Admin Login	22/04/2022	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
User Register	27/04/2022	4	0	0	0	0	1	1	1	1	0	0	0	0	0	0
User Login	3/05/2022	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Total		9	1	1	1	0	1	1	1	1	0	0	0	0	1	1

Table 3.5: Sprint 1 Actual

Backlog Item	Status & completion date	Original estimate in hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14
UserStory #1,#2,#3																
Admin Login	22/04/2022	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
User Register	27/04/2022	4	0	0	0	0	1	1	1	1	0	0	0	0	0	0
User Login	3/05/2022	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1
UserStory #4,#5,#6																
Search friends	6/05/2022	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Send friend request	13/05/2022	5	0	0	0	0	0	0	1	1	1	1	1	0	0	0
Manage friends	18/05/2022	3	1	1	0	0	0	0	0	0	0	0	0	0	0	1
Total		20	3	3	2	0	1	1	2	2	1	1	1	0	1	2

Table 3.6: Sprint 2 Actual

Backlog Item	Status & completion date	Original estimate in hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14
UserStory #1,#2,#3																
Admin Login	22/04/2022	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
User Register	27/04/2022	4	0	0	0	0	1	1	1	1	0	0	0	0	0	0
User Login	3/05/2022	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1
UserStory #4,#5,#6																
Search friends	6/05/2022	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Send friend request	13/05/2022	5	0	0	0	0	0	0	1	1	1	1	1	0	0	0
Manage friends	18/05/2022	3	1	1	0	0	0	0	0	0	0	0	0	0	0	1
UserStory #7,#8,#9																
View friend list	27/05/2022	5	0	0	0	0	0	0	0	1	1	1	1	1	0	0
Chat with friends	31/05/2022	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Technical Part	1/06/2022	3	0	0	0	1	1	1	0	0	0	0	0	0	0	0
Total		30	3	4	3	1	2	2	2	3	2	2	2	1	1	2

Table 3.7: Sprint 3 Actual

Backlog Item	Status & completion date	Original estimate in hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14
UserStory #1,#2,#3																
Admin Login	22/04/2022	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
User Register	27/04/2022	4	0	0	0	0	1	1	1	1	0	0	0	0	0	0
User Login	3/05/2022	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1
UserStory #4,#5,#6																
Search friends	6/05/2022	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Send friend request	13/05/2022	5	0	0	0	0	0	0	1	1	1	1	1	0	0	0
Manage friends	18/05/2022	3	1	1	0	0	0	0	0	0	0	0	0	0	0	1
UserStory #7,#8,#9																
View friend list	27/05/2022	5	0	0	0	0	0	0	0	1	1	1	1	1	0	0
Chat with friends	31/05/2022	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Technical Part	1/06/2022	3	0	0	0	1	1	1	0	0	0	0	0	0	0	0
UserStory #10,#11,#12																
View & reply messages	10/06/2022	5	0	0	0	0	0	0	0	0	1	1	1	1	1	0
User View status	14/06/2022	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Admin view user list	17/06/2022	3	0	0	0	1	1	1	0	0	0	0	0	0	0	0
Total		40	3	5	4	2	3	3	2	3	3	3	3	2	2	2

Table 3.8: Sprint 4 Actual

Backlog Item	Status & completion date	Original estimate in hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14
UserStory #1,#2,#3																
Admin Login	22/04/2022	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
User Register	27/04/2022	4	0	0	0	0	1	1	1	1	0	0	0	0	0	0
User Login	3/05/2022	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1
UserStory #4,#5,#6																
Search friends	6/05/2022	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Send friend request	13/05/2022	5	0	0	0	0	0	0	1	1	1	1	1	0	0	0
Manage friends	18/05/2022	3	1	1	0	0	0	0	0	0	0	0	0	0	0	1
UserStory #7,#8,#9																
View friend list	27/05/2022	5	0	0	0	0	0	0	0	1	1	1	1	1	0	0
Chat with friends	31/05/2022	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Technical Part	1/06/2022	3	0	0	0	1	1	1	0	0	0	0	0	0	0	0
UserStory #10,#11,#12																
View & reply messages	10/06/2022	5	0	0	0	0	0	0	0	0	1	1	1	1	1	0
User View status	14/06/2022	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Admin view user list	17/06/2022	3	0	0	0	1	1	1	0	0	0	0	0	0	0	0
UserStory #13,#14,#15, #16																
Admin manage user	22/06/2022	3	0	0	0	0	0	0	0	0	1	1	1	0	0	0
Admin block user	25/06/2022	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Admin view blocked user	28/06/2022	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Testing & Output generation	1/07/2022	3	0	0	0	1	1	1	0	0	0	0	0	0	0	0
Total		50	3	6	5	3	4	3	2	3	4	4	4	2	3	3

Table 3.9: Sprint 5 Actual

Chapter 4

Results and Discussions

4.1 Datasets

There are a dataset were used, for SPAM messages and HAM messages. For SPAM message comparison is based on dataset called SPAM and HAM.

4.2 Experimentation

As a part of experimentation, authors after creating the vector set, passed 2 inputs to test whether or not the model (including the word vectors) is able to check whether the message is SPAM or HAM. Input I: given to the developed system: “Hello, how are you?” Input II: given to the developed system: “Congratulations!!! You have won 5000”

4.3 Results

The Output were generated

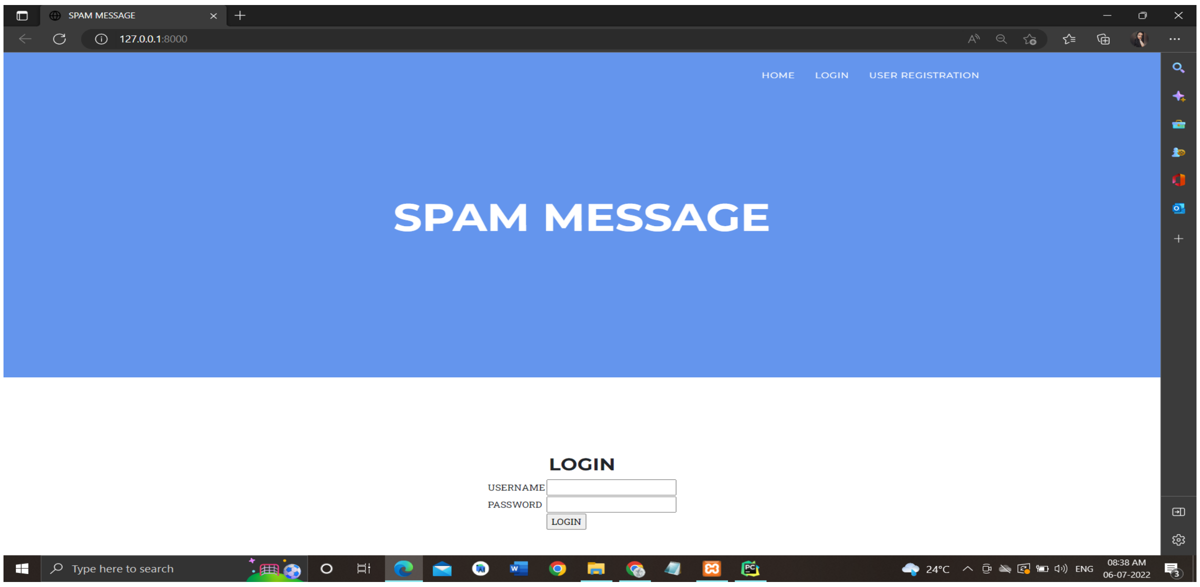


Figure 4.1: Login Page

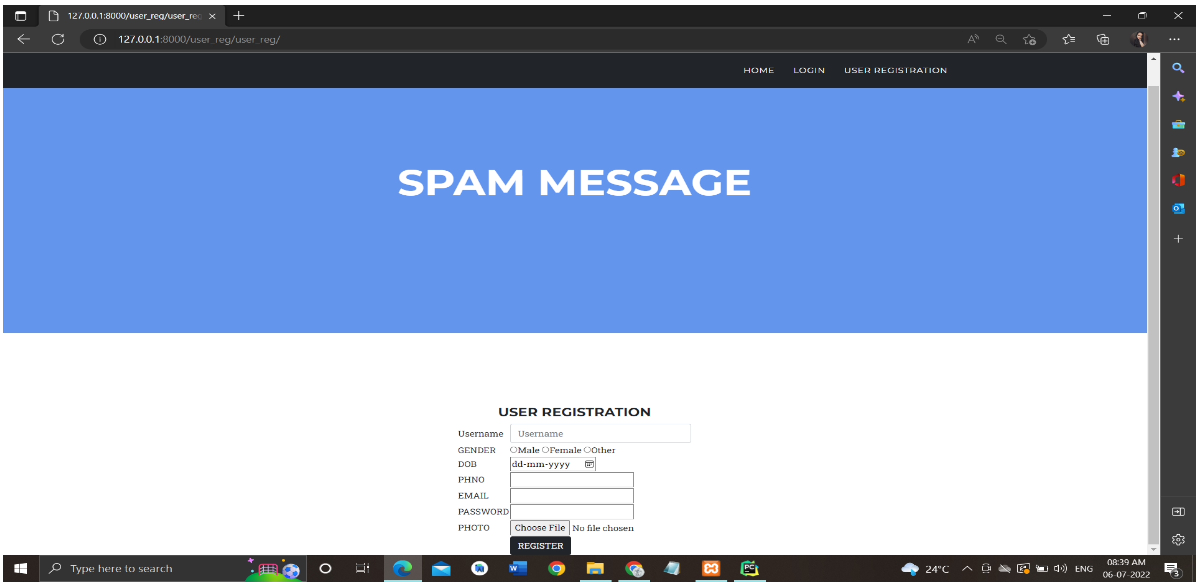


Figure 4.2: Registration

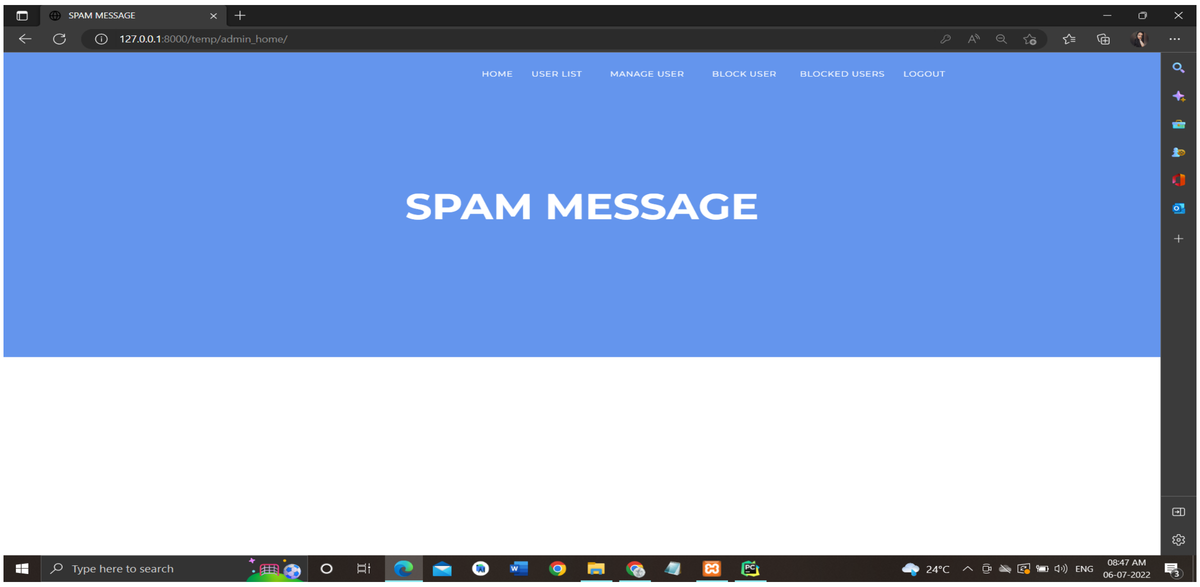
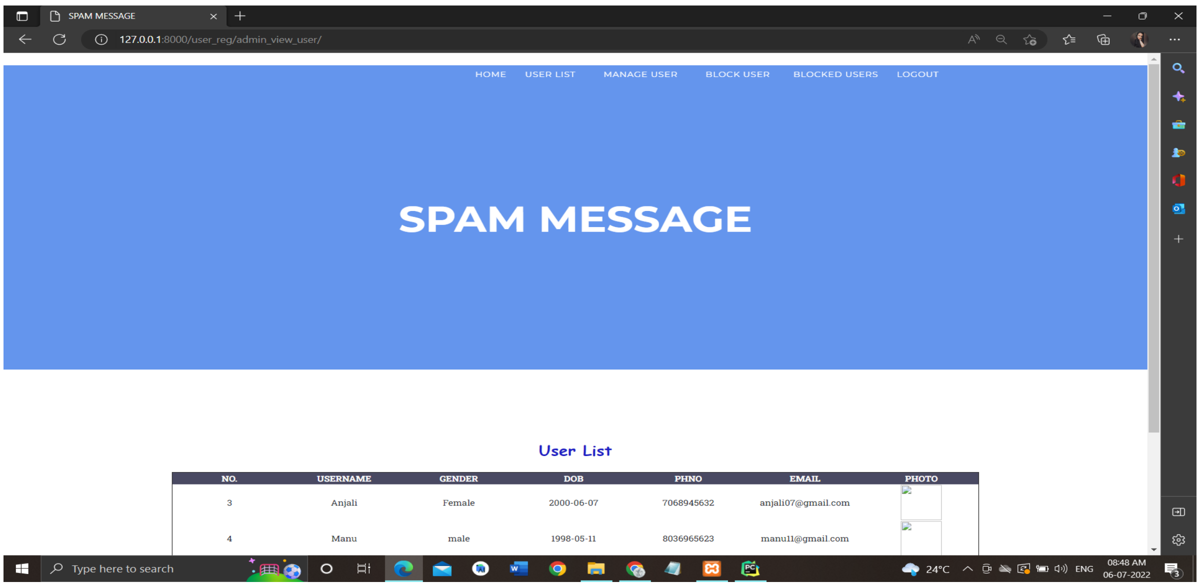


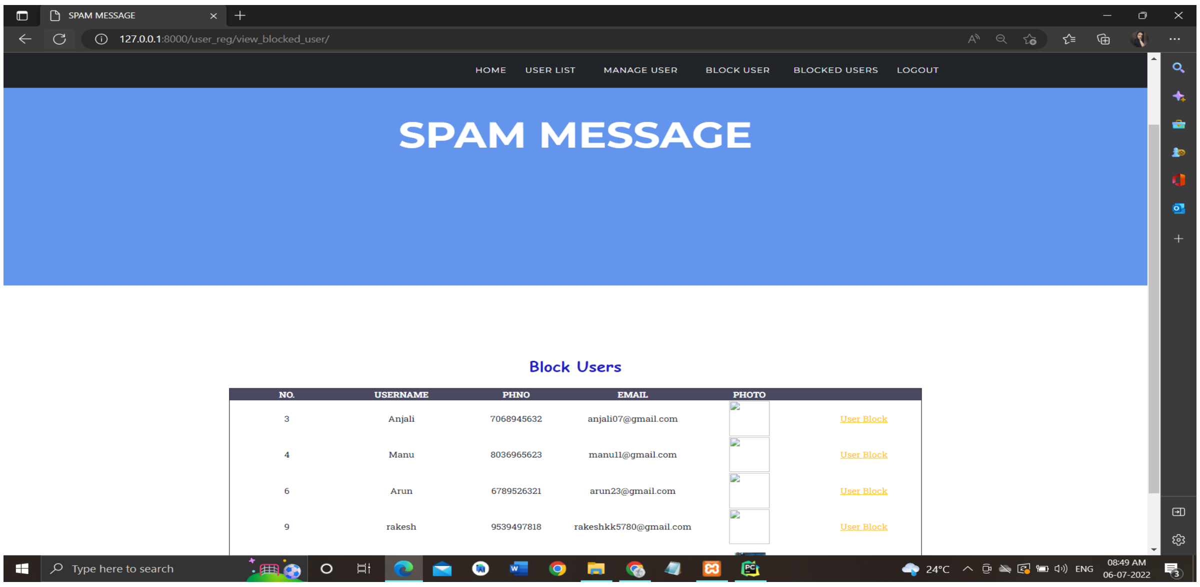
Figure 4.3: Admin Home



User List

NO.	USERNAME	GENDER	DOB	PHNO	EMAIL	PHOTO
3	Anjali	Female	2000-06-07	7068945632	anjali07@gmail.com	
4	Manu	male	1998-05-11	8036965623	manu11@gmail.com	

Figure 4.4: User List



Block Users

NO.	USERNAME	PHNO	EMAIL	PHOTO	
3	Anjali	7068945632	anjali07@gmail.com		User Block
4	Manu	8036965623	manu11@gmail.com		User Block
6	Arun	6789526321	arun23@gmail.com		User Block
9	rakesh	9539497818	rakeshkk5780@gmail.com		User Block

Figure 4.5: Block User

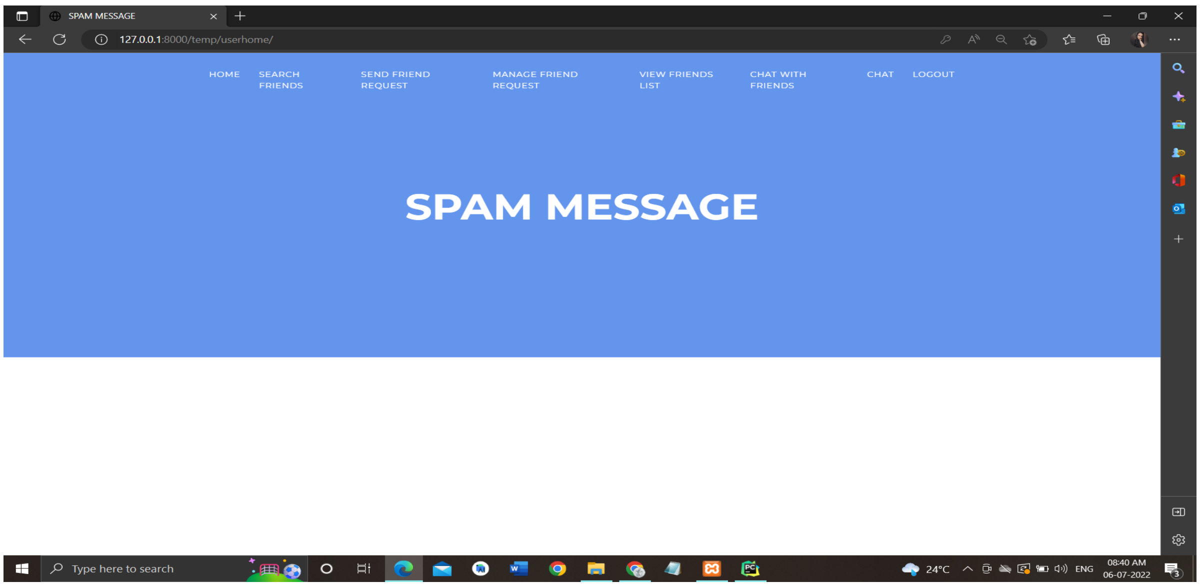
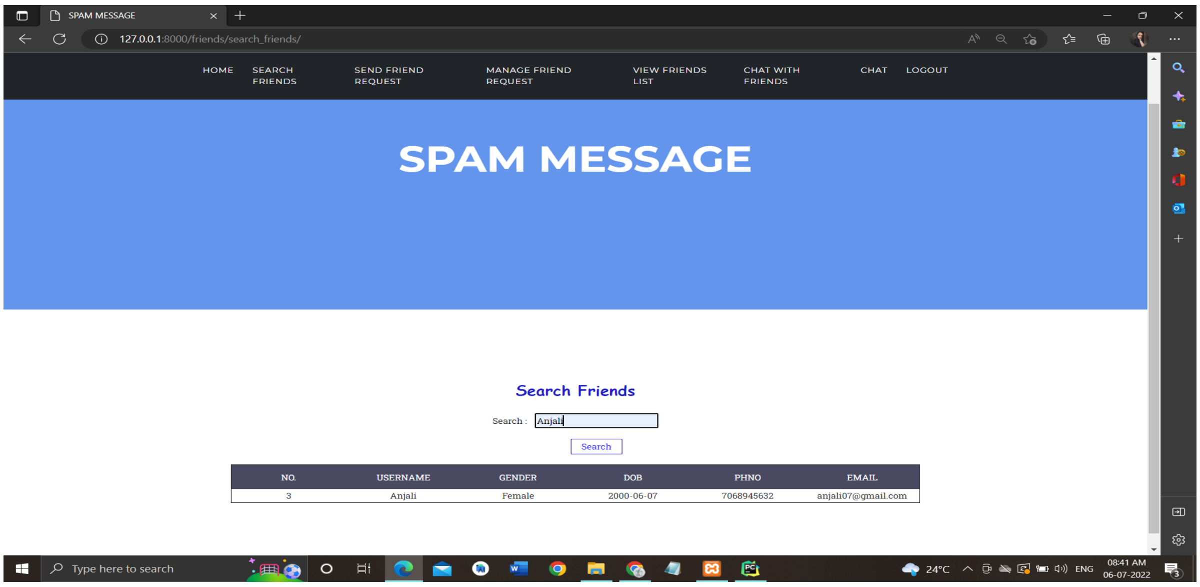


Figure 4.6: User Home

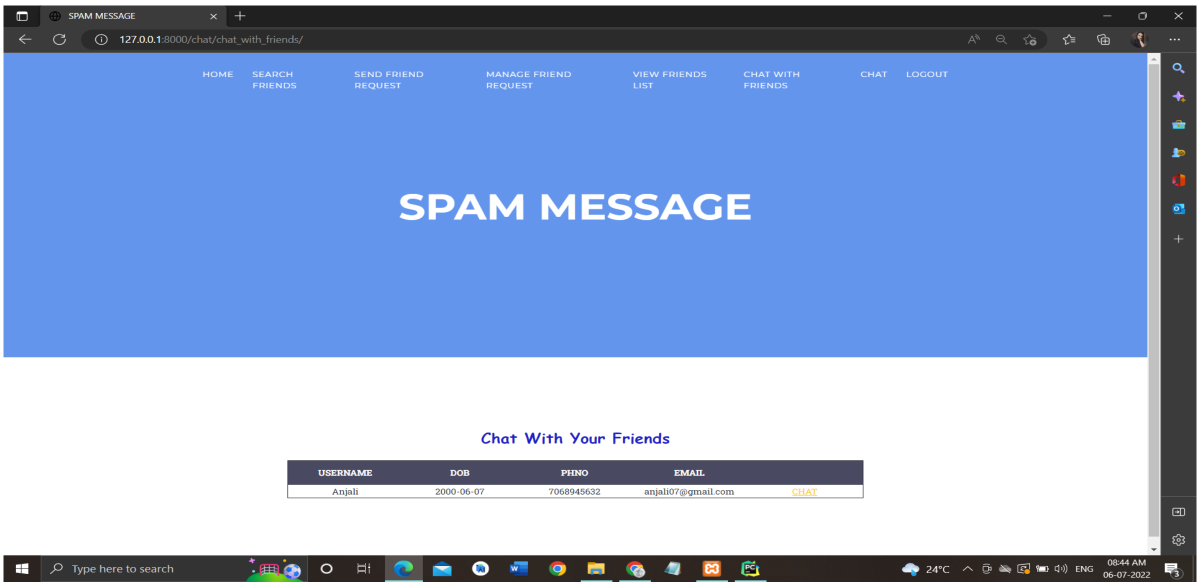


Search Friends

Search : Anjali

NO.	USERNAME	GENDER	DOB	PHNO	EMAIL
3	Anjali	Female	2000-06-07	7068945632	anjali07@gmail.com

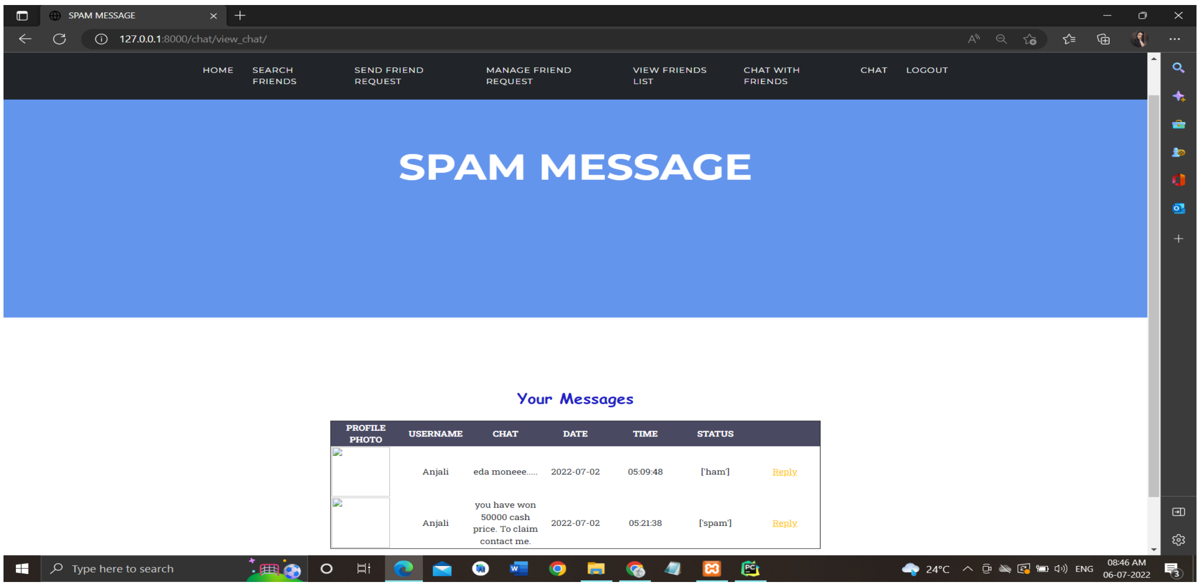
Figure 4.7: Search Friends



Chat With Your Friends

USERNAME	DOB	PHNO	EMAIL	
Anjali	2000-06-07	7068945632	anjali07@gmail.com	CHAT

Figure 4.8: Chat With Friends



Your Messages

PROFILE PHOTO	USERNAME	CHAT	DATE	TIME	STATUS	
	Anjali	eda moneee.....	2022-07-02	05:09:48	['ham']	Reply
	Anjali	you have won 50000 cash price. To claim contact me.	2022-07-02	05:21:38	['spam']	Reply

Figure 4.9: Chat

Chapter 5

Conclusions

In this project describes an automated SPAM message detection system. The proposal uses natural language processing techniques, machine learning approaches in order to infer whether a SMS belongs to SPAM or HAM. The implementation of minimum support facilitates the problems dealing with limited features due to the limited number of characters in SMS; it therefore produces the new features to differentiate between spam SMS and ham SMS. The use of datasets with varied training data is agreeable to be applied by using the SVM. By implementing the SVM for feature extraction, it can elevate the score of precision. Thus, the system becomes more precise in providing the information requested by the users in response to the SMS classification.

References

[1] Njoku, Mary Gloria. (2015). The use of short message service in post-secondary education.

[2] http://troindia.in/journal/ijcesr/vol8iss1/41-47.pdf

[3] https://www.hindawi.com/journals/scn/2020/8873639/

[4] L. Zhang, J. Zhu, and T. Yao, "An evaluation of statistical spam filtering techniques," ACM Transactions on Asian Language Information Processing (TALIP), vol. 3, no. 4, pp. 243–269, 2004.

[5] M. Bassiouni, M. Ali, and E. A. El-Dahshan, "Ham and spam E-mails classification using machine learning techniques," Journal of Applied Security Research, vol. 13, no. 3, pp. 315–331, 2018.

[6] I. Alsmadi and I. Alhami, "Clustering and classification of email contents," Journal of King Saud University—Computer and Information Sciences, vol. 27, no. 1, pp. 46–57, 2015.

[7] B. Yu and Z.-B. Xu, "A comparative study for content-based dynamic spam classification using four machine learning algorithms," Knowledge-Based Systems, vol. 21, no. 4, pp. 355–362, 2008.

Appendix

Source Code

```
from django.shortcuts import render
from chat.models import Chat
from friends.models import Friends
from user_reg.models import UserReg
from django.core.files.storage import FileSystemStorage

import datetime

from chat.models import Chat
# Create your views here.
import pandas as pd
from pandas import read_excel

from bs4 import BeautifulSoup
import re
import pickle

def html_tag(phrase):
   http_remove = re.sub(r"http\S+", "",phrase)
   html_remove = BeautifulSoup(http_remove, 'lxml').get_text()
   return html_remove

#remove words with numbers python: https://stackoverflow.com/a/18082370/4084039
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
def decontracted(phrase):
   # specific
   phrase = re.sub(r"won't", "will not", phrase)
   phrase = re.sub(r"can\'t", "can not", phrase)

   # general
   phrase = re.sub(r"n\'t", " not", phrase)
   phrase = re.sub(r"\'re", " are", phrase)
   phrase = re.sub(r"\'s", " is", phrase)
   phrase = re.sub(r"\'d", " would", phrase)
   phrase = re.sub(r"\'ll", " will", phrase)
   phrase = re.sub(r"\'t", " not", phrase)
   phrase = re.sub(r"\'ve", " have", phrase)
   phrase = re.sub(r"\'m", " am", phrase)
   return phrase

stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
         "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
         'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
         'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
         'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
         'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
         'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
         'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
         'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
         'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
         's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
         've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
         "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
         "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
         'won', "won't", 'wouldn', "wouldn't"])

from tqdm import tqdm
from spam_message import settings
from sklearn.feature_extraction.text import TfidfVectorizer
def chat(request,abc):
   uid = request.session["uid"]
   objj = UserReg.objects.filter(user_id=uid) # name
   gm = Chat.objects.filter(friend_id=uid)
   context = {
      'name': objj,
      'chat':gm
   }

   if request.method == "POST":
      obj = Chat()
      uid = request.session["uid"]
      obj.friend_id= abc
      obj.user_id= uid
      obj.date = datetime.date.today()
      try:
         myfile = request.FILES["photo"]
         fs = FileSystemStorage()
         filename = fs.save(myfile.name, myfile)
         obj.photo = myfile.name
      except:
         pass

      obj.chat = request.POST.get('message')

      dspath=str(settings.BASE_DIR)+str(settings.STATIC_URL)+"spam.xls"
      trainData = read_excel(dspath) # DATA FRAME -> COL, ROWS
      df2 = {'Label': 'ham', 'Content': obj.chat}
      trainData = trainData.append(df2, ignore_index=True)
      processed_review = []
      for i in trainData["Content"].values:
         sentance = html_tag(i)
         sentance = decontracted(sentance)
         sentance = re.sub("\S*\d\S*", "", sentance)
         sentance = re.sub('[^A-Za-z]+', ' ', sentance)
         sentance = " ".join(i.lower() for i in sentance.split() if i.lower() not in stopwords)
         processed_review.append(sentance)
      trainData["Cleantext"] = processed_review
      vectorizer = TfidfVectorizer(min_df=5,
                            max_df=0.8,
                            sublinear_tf=True,
                            use_idf=True)
      train_vectors = vectorizer.fit_transform(trainData['Cleantext'])
      X_test1 = train_vectors[train_vectors.shape[0] - 1]

      mpath = str(settings.BASE_DIR) + str(settings.STATIC_URL) + "model_dtc.sav"
      model = pickle.load(open(mpath, 'rb'))
      # y_score = model.predict(X_test)
      y_score = model.predict(X_test1)
      res = "This Message is :" + y_score[0]
      obj.status=y_score
      print(res)


      # obj.time = datetime.datetime.now().time()
      obj.time = datetime.datetime.now().strftime("%I:%M:%S")
      obj.save()


   return render(request,'chat/chat.html',context)


def chat_with_friends(request):
   uid = request.session["uid"]
   objj = UserReg.objects.filter(user_id=uid) # name

   print(uid)
   cd=UserReg.objects.filter(status='cyberbullying detected').values_list('user_id',flat=True)
   print(cd)
   # obj = Friends.objects.filter(request_status='accepted', f_user_id=uid)
   obj = Friends.objects.filter(request_status="accepted",user_id=uid)
   gm = Chat.objects.filter(friend_id=uid)
   chat_friend = {
         'chatfriend': obj,
          'name':objj,
          'chat':gm
      }
   return render(request,'chat/chat_with_friends.html',chat_friend)

def view_chat(request):
   uid = request.session["uid"]
   # print(uid)
   objj = UserReg.objects.filter(user_id=uid) # name

   obj = Chat.objects.filter(friend_id=uid)
   gm = Chat.objects.filter(friend_id=uid)
   # gm = Chat.objects.filter(user_id=uid).exclude(friend_id=uid)
   print(gm)
   chatt = {
      'cha': obj,
      'name': objj,
      'chat':gm
   }

   return render(request,'chat/view_chat.html',chatt)



   from django.db import models
from user_reg.models import UserReg
# from friends.models import Friends
class Chat(models.Model):
   chat_id = models.AutoField(primary_key=True)
   # friend_id = models.IntegerField()
   # friend=models.ForeignKey(UserReg,to_field='user_id',on_delete=models.CASCADE,related_name='abcd')
   friend=models.ForeignKey(UserReg,to_field='user_id',on_delete=models.CASCADE,related_name='ff1')
   # user_id = models.IntegerField()
   user=models.ForeignKey(UserReg,to_field='user_id',on_delete=models.CASCADE,related_name='ff2')
   chat = models.CharField(max_length=160)
   photo = models.CharField(max_length=100, blank=True, null=True)
   date = models.CharField(max_length=20)
   time = models.CharField(max_length=20)
   status = models.CharField(max_length=50)

   class Meta:
      managed = False
      db_table = 'chat'
```

```
      from django.conf.urls import url
from chat import views

urlpatterns = [
   url('chat/(?P<abc>\w+)',views.chat,name='chat'),
   url('chat_with_friends/',views.chat_with_friends),
   url('view_chat/',views.view_chat)

]
```

```
from django.shortcuts import render
from user_reg.models import UserReg
from login.models import Login
from chat.models import Chat
from django.core.files.storage import FileSystemStorage
from django.http import HttpResponse,HttpResponseRedirect

# from user_reg.models import UserRegCreate your views here.
def user_reg(request):
   if request.method == "POST":
      obj=UserReg()
      obj.username=request.POST.get('username')
      obj.gender = request.POST.get('gender')
      obj.dob = request.POST.get('dob')
      obj.phno = request.POST.get('phno')
      obj.email = request.POST.get('email')
      obj.password = request.POST.get('password')
      obj.status="pending"

      myfile = request.FILES["photo"]
      fs = FileSystemStorage()
      filename = fs.save(myfile.name, myfile)
      obj.profile_photo = myfile.name


      obj.save()

      ob =Login()
      ob.username=request.POST.get('email')
      ob.password=request.POST.get('password')
      ob.type="user pending"
      ob.user_id=obj.user_id
      ob.save()



   return render(request,'user_reg/user_reg.html')

def manage_user(request):
   ob=UserReg.objects.filter(status='pending')
   context={
      'objval':ob,
   }
   return render(request,"user_reg/manage_user.html",context)
def view_blocked_user(request):
   obj=UserReg.objects.filter(status='approved')
   context={
      'objval':obj,
   }
   return render(request,"user_reg/view_blocked_user.html",context)

def blckuser(request,idd):
   obj=UserReg.objects.get(user_id=idd)
   obj.status='blocked'
   obj.save()

   objj = Login.objects.get(user_id=idd)
   objj.type = 'user pending'
   objj.save()


   return view_blocked_user(request)


def admin_view_user(request):
   obj=UserReg.objects.filter(status='approved')
   user_details = {
      'user': obj
   }


   return render(request,'user_reg/admin_view_user.html',user_details)




def approve_user(request,abc):
   obj=UserReg.objects.get(user_id=abc)
   obj.status="approved"
   obj.save()

   ob=Login.objects.get(user_id=abc, type = 'user pending')
   ob.type="user"
   ob.save()
   # return render(request,'user_reg/manage_user.html')
   # return HttpResponse("yesss")
   return HttpResponseRedirect("/user_reg/manage_user/")
def reject_user(request,cdf):
   obj = UserReg.objects.get(user_id=cdf)
   obj.status = "rejected"
   obj.save()
   return render(request,'user_reg/manage_user.html')

def edit_profile(request):
   uid = request.session["uid"]
   objj = UserReg.objects.filter(user_id=uid) #name
   gm = Chat.objects.filter(friend_id=uid)

   context = {
      'name': objj,
      'chat': gm
   }
   if request.method == "POST":
      obj = UserReg.objects.get(user_id=uid)
      myfile = request.FILES["profile"]
      fs = FileSystemStorage()
      filename = fs.save(myfile.name, myfile)
      obj.profile_photo = myfile.name

      # obj.profile_photo = request.POST.get('profile_photo')
      obj.save()

      ob = Login.objects.get(type="user", user_id=uid)
      ob= request.POST.get('user_id')
      # ob.save()
   return render(request,'user_reg/edit_profile.html',context)


def edit_username(request):
   uid = request.session["uid"]
   objj = UserReg.objects.filter(user_id=uid) # name
   gm = Chat.objects.filter(friend_id=uid)
   context = {
      'name': objj,
      'chat':gm
   }
   if request.method == "POST":
      obj = UserReg.objects.get(user_id=uid)
      obj.username= request.POST.get('username')
      obj.save()

      # ob = Login.objects.get(type="user", user_id=uid)
      # ob.user_id = request.POST.get('user_id')
      # ob.save()
   return render(request,'user_reg/edit_username.html',context)
def edit_email(request):
   uid = request.session["uid"]
   objj = UserReg.objects.filter(user_id=uid) # name
   gm = Chat.objects.filter(friend_id=uid)
   context = {
      'name': objj,
      'chat':gm
   }
   if request.method == "POST":
      obj = UserReg.objects.get(user_id=uid)
      obj.email = request.POST.get('email')
      obj.save()

      ob = Login.objects.get(type="user", user_id=uid)
      ob.username = request.POST.get('email')
      ob.save()
   return render(request,'user_reg/edit_email.html',context)

def edit_password(request):
   uid=request.session["uid"]
   objj = UserReg.objects.filter(user_id=uid) # name
   gm = Chat.objects.filter(friend_id=uid)
   context = {
      'name': objj,
      'chat':gm
   }
   if request.method == "POST":
      obj=UserReg.objects.get(user_id=uid)
      obj.password=request.POST.get('password')
      obj.save()

      ob=Login.objects.get(type="user",user_id=uid)
      ob.password=request.POST.get('password')
      ob.save()
   return render(request,'user_reg/edit_password.html',context)
def edit_phno(request):
   uid=request.session["uid"]
   objj = UserReg.objects.filter(user_id=uid) # name
   gm = Chat.objects.filter(friend_id=uid)
   context = {
      'name': objj,
      'chat':gm
   }
   if request.method == "POST":
      obj=UserReg.objects.get(user_id=uid)
      obj.phno=request.POST.get('phno')
      obj.save()

      # ob=Login.objects.get(type="user",user_id=uid)
      # ob.=request.POST.get('password')
      # ob.save()
   return render(request,'user_reg/edit_phno.html',context)

def admin_view_blocked_users(request):
   uid = request.session["uid"]
   objj = UserReg.objects.filter(user_id=uid) # name

   obj = UserReg.objects.filter(status='blocked')

   blk = {
      'bl': obj,
      'name': objj,
   }


   return render(request, 'user_reg/admin_view_blocked_users.html',blk)


def unblock(request,aa):
   # uid = request.session["uid"]
   # objj = UserReg.objects.filter(user_id=uid) # name

   obj = UserReg.objects.get(user_id=aa)
   obj.status = 'approved'
   obj.save()

   objj = Login.objects.get(user_id=aa)
   objj.type = 'user'
   objj.save()



   # unbl={
   #    'name':objj
   # }

   # return render(request,'user_reg/admin_view_blocked_users.html',unbl)


   return admin_view_blocked_users(request)
```

Database Design

Attribute Name	**Datatype**	**Width**	**Description**
login id	Integer	10	Primary Key
username	varchar	50	
password	Varchar	50	
type	varchar	30	
user id	Integer	10	

Table A.1: login

Attribute Name	**Datatype**	**Width**	**Description**
user id	Integer	10	Primary Key
username	Varchar	50	
dob	Varchar	20	
gender	Varchar	20	
phno	Varchar	20	
email	Varchar	50	
password	Varchar	20	
profile photo	Varchar	100	
status	Varchar	50	

Table A.2: user registration

Attribute Name	Datatype	Width	Description
friend id	Integer	10	Primary Key
user id	Integer	10	
f user id	Integer	10	
request status	varchar	50	

Table A.3: friends

Attribute Name	Datatype	Width	Description
chat id	Integer	10	Primary Key
friend id	Integer	10	
user id	Integer	10	
chat	Varchar	160	
photo	Varchar	100	
date	Varchar	20	
time	Varchar	20	
status	Varchar	50	

Table A.4: chat

Attribute Name	Datatype	Width	Description
report id	Integer	10	Primary Key
user id	Integer	10	
chat id	Integer	10	
report	Varchar	50	
date	Varchar	20	
time	Varchar	20	

Table A.5: report user

DataFlow Diagram

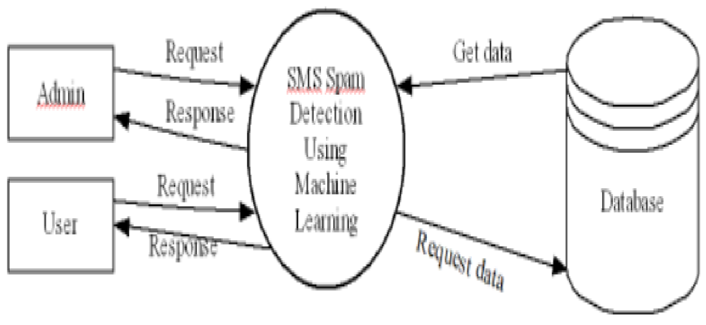


Figure A.1: LEVEL 0

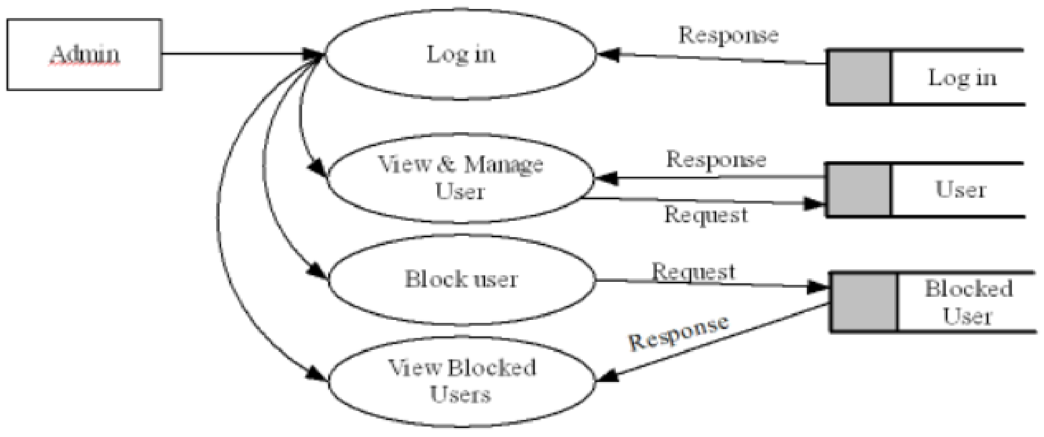


Figure A.2: LEVEL 1

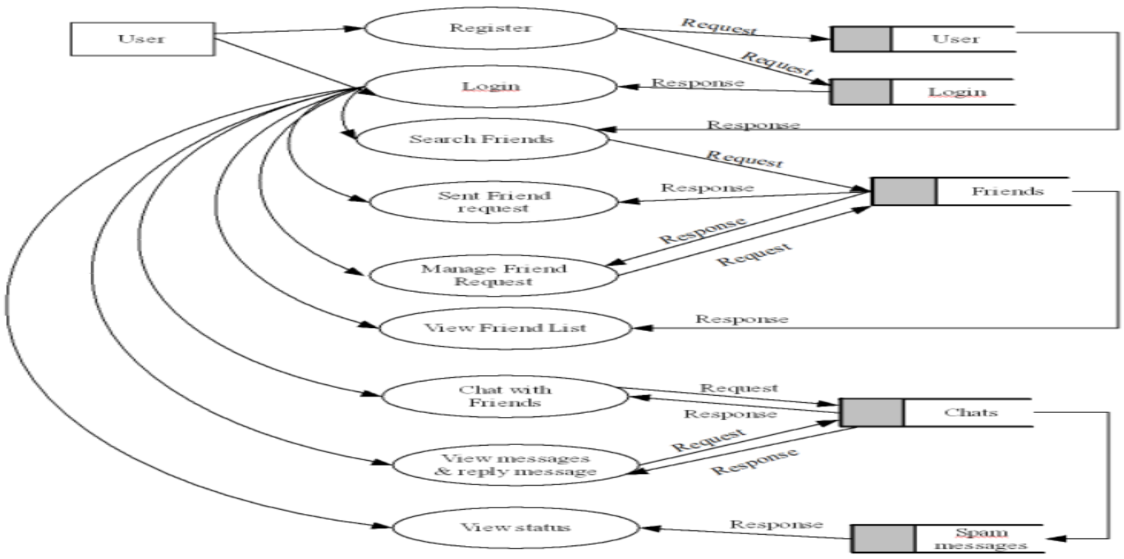


Figure A.3: LEVEL 2