**A PROJECT REPORT ON**

# CHAT APPLICATION WITH SPAM DETECTION

SUBMITTED TO THE
CUMMINS COLLEGE OF ENGINEERING FOR WOMEN,
KARVENAGAR, PUNE
(An autonomous institute affiliated to Savitribai
Phule Pune university),

IN THE PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE AWARD OF THE
DEGREE

OF

## BACHELOR OF TECHNOLOGY

## (COMPUTER ENGINEERING)

**SUBMITTED BY**


| | | |
|---|---|---|
| **PRAJAKTA DEOKULE** | **C22019221332** | **4329** |
| **MAITREYEE JOSHI** | **C22019221355** | **4351** |
| **TANVI KATAKKAR** | **C22020222305** | **4376** |
| **LAVANNYA PATIL** | **C22020222306** | **4377** |


## DEPARTMENT OF COMPUTER ENGINEERING


**MKSSS'S CUMMINS COLLEGE OF ENGINEERING FOR WOMEN**

**KARVENAGAR, PUNE 411052**
**SAVITRIBAI PHULE PUNE UNIVERSITY**


**A.Y. 2022 -2023**

# CERTIFICATE

This is to certify that the project report entitles

**“ CHAT APPLICATION WITH SPAM DETECTION”**

Submitted By

| | | |
|---|---|---|
| **PRAJAKTA DEOKULE** | **C22019221332** | **4329** |
| **MAITREYEE JOSHI** | **C22019221355** | **4351** |
| **TANVI KATAKKAR** | **C22020222305** | **4376** |
| **LAVANNYA PATIL** | **C22020222306** | **4377** |

is a bonafide student of this institute and the work has been carried out by her under the supervision of **Prof. Rakhi A. Dongaonkar** and it is approved for the partial fulfillment of the requirement of Cummins college of engineering for women, karvenagar, Pune
(An autonomous institute affiliated to Savitribai Phule Pune University.),

for the award of the degree of **Bachelor of Technology** (Computer Engineering).

**(Prof. Rakhi Dongaonkar)**
Guide,
Department of Computer Engineering

**(Dr. S. M. Kelkar)**
Head,
Department of Computer Engineering

**(Dr. M.B. Khambete**)
Director,
Cummins College of Engineering for Women

Place : Pune

Date :

# ACKNOWLEDGMENT

Our profound gratitude goes to our wonderful guide, **Prof. Rakhi Dongaonkar** for her invaluable support, patience, time and guidance in seeing us to the completion of this project work. Also our gratitude goes to **Dr. Supriya Kelkar**, Head, and Department of Computer Engineering for top of the state labs for project work.

We also extend gratitude and appreciation to **Dr. M.B. Khambete**, Principal and Cummins College of Engineering for Women, Pune.
We also wish to acknowledge the great support and advice given to us by our **BlinkAds** mentor **Mr. Prabhat Mishra**.

We also wish to acknowledge the great support of our parents, siblings, who have been a source of inspiration towards our academic pursuit.

**PRAJAKTA DEOKULE**
**MAITREYEE JOSHI**
**TANVI KATAKKAR**
**LAVANNYA PATIL**

# ABSTRACT

SMS messaging is now more common than ever thanks to recent advancements in mobile technology. As a result, there are now more spam communications are happening via mobile devices. Even though emails are the primary source of spam worldwide, SMS services are now not far behind in their contribution to the problem. No one wants their mobile devices to be inundated with spam messages. Numerous methods have been developed to identify and minimize spam messages, and research on the subject is constantly ongoing.

Spam classification in SMS messages is a difficult task. A lot of research has been done in this area using machine learning approaches including Naive Bayes (NB), Random Forest (RF), and Support Vector Machine (SVM). These techniques, however, perform only to a limited extent, thus failing to classify a diverse variety of spam messages correctly. So a thorough investigation is required to discover a more reliable and accurate way. To solve this, we put forth a technique called Long Short-Term Memory (LSTMs), an enhanced Recurrent Neural Network (RNN) structure with memory cells as part of its gating mechanism. In this project, we're creating a web and mobile application that will weed out spam messages. This report covers all aspects of system design, project implemenatation and the technology used along with the functional and non-functional requirements of the project.

# TABLE OF CONTENTS

# LIST OF ABBREVIATIONS

1. AI - Artificial Intelligence
2. ML - Machine Learning
3. SRS - Software Requirements Specification
4. LAN - Local Area Network
5. OS - Operating System
6. HTML - Hypertext Markup Language
7. LSTM - Long Short Term Memory
8. DFD - Data Flow Diagram
9. UI - User Interface
10. RN - React Native
11. DRY - Don’t Repeat Yourself
12. CRUD - Create, Read, Update, Delete
13. API - Application Programming Interface
14. NLP - Natural Language Processing

# LIST OF FIGURES

# LIST OF TABLES

# 01. INTRODUCTION

Messaging is a form of interpersonal communication, and there are billions of individuals that use mobile devices every day. However, the absence of adequate message filtering methods makes this sort of communication insecure. Spam contributes to this risk by making mobile SMS communication insecure.

One of the major issues with instant message systems is regarded to be spam. Spam is a junk message. Spam messages are distributed to users without their consent, and receivers find them annoying. It includes a variety of information, including phishing messages, advertisements for goods and services. These days, more mobile devices are being used in the environment for message communication, which has led to an increase in spam messages.

The users may occasionally suffer financial loss as a result of these spam message. The cost of sending mail and messages is very low for senders but highly expensive for the recipients. Spam's cost can be calculated as the loss of human time and the loss of vital messages or communications. The important communications are impacted by these spam messages.

## 1.1 MOTIVATION

The emergence of technology has transformed the way we interact in profound ways. We communicate seamlessly while being remote using various devices and platforms like chat application that keep coming out. Spam messages have dramatically increased since chat applications first appeared. Spam is electronic messaging that is unsolicited, unwelcomed, and may contain malicious content.

The risks associated with spam communications for users are numerous: unwanted advertising, disclosure of the user's personal information, falling prey to fraud or financial schemes, being seduced into malware and phishing websites, unintentional exposure to offensive content, etc. Therefore, spam identification is urgently needed in order to enhance user experience, message quality, and to protect users from fraud.

## 1.2 PROBLEM DEFINITION

Chat Application with Spam detection – the problem statement deals with creation of real-time chat application with spam detection capabilities that uses machine learning algorithms.

The chat application offers features including group and instant messaging, profile personalization, emoji support, slash commands, and presence and status tracking for users. The chat application has a spam detection feature driven by AI. The messages can be filtered out as spam or ham using the machine learning technique.

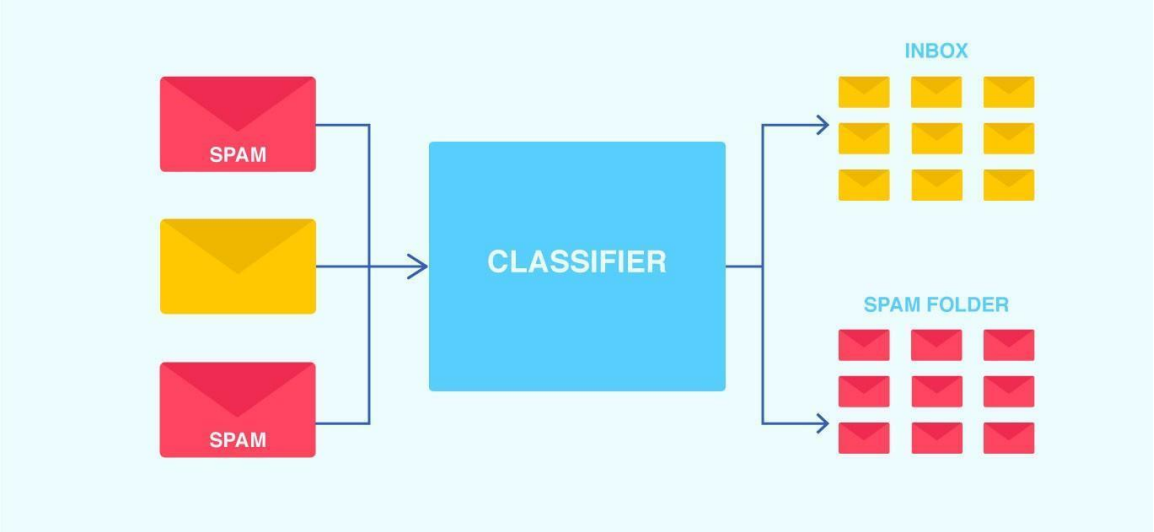



Fig 1.1 :- Classification of Spam and Ham Messages

## 1.3 OBJECTIVES

a) To develop a real time chat application:

   It consists of offering a group chat and instant messaging function that facilitates mass communication.

b) To integrate spam detection algorithm:

   To incorporate the most effective spam detection method.

c) To provide user friendly interface:

   Providing users easy to access, easy to operate and easy to understand interface.

d) To classify the message as spam or ham:

   Using machine learning algorithm to classify the messages as spam or not spam (ham).

# 02.LITERATURE SURVEY

## 2.1 BACKGROUND OF DOMAIN

The growth of mobile phone users has led to a dramatic increase in SMS spam messages. Despite the fact that a variety of information channels are currently seen as "spotless" and reliable in many parts of the world, ongoing data clearly demonstrate that the amount of cell phone spam is dramatically increasing over time. It is a growing catastrophe, especially in the Middle East and Asia.

Separating SMS spam is a similarly late task to solve this problem. It gains several concerns and practical fixes from SMS spam separation. In any case, it brings up its own unique problems. By including Indian messages in the entire available SMS dataset, this research aims to address the challenge of classifying flexible messages as Ham or Spam for the Indian Users. The research examines various machine learning classifiers on a sizable dataset of individual SMS texts.

The entire research of Julis et al [1] was broken up into various iterations. Each iteration was finished by working through the following four stages: inception, where the project's idea was discovered; elaboration, where the system's architecture was designed; construction, where the existing code was put into use; and transition, where the developed portion of the project was validated. There are still certain areas that can be improved, such as by incorporating more filtering methods or altering certain features of the ones that already exist. Changes like increasing or decreasing the message's intriguing word count and rearranging the formula for determining interesting rate can be made afterwards.This research however does not solve the problem of analysis and management of report in spam sms filter storing, which is far more challenging.

Spam has developed into a significant problem for computer security since it serves as a primary channel for the spread of dangers including viruses, worms, and phishing scams. Presently, a significant portion of received emails and messages are spam. Different strategies are in place to address this issue, including challenge response models, whitelisting, blacklisting, email signatures, and various machine learning techniques.

These options are available to consumers, however owing to the dynamic nature of the Web, there aren't any worldwide 100% secure methods that can address this issue. Machine learning techniques are typically used by spam detectors to filter web traffic. Muhammed Iqbal et al's[3] research focuses on methodically examining the benefits and drawbacks of existing technology for spam detection, and a taxonomy of accepted methods is presented. The research claims that there is no way to completely eradicate the spam problem and propose a solution of merely combining various techniques of finding a probability score to determine whether the given text is spam or not. However, they don't tell "How to combine those techniques".

Spam detection is crucial for protecting email and message communication. A significant problem is the accurate identification of spam, and numerous detection techniques have been put forth by various researchers. However, these techniques fall short in their ability to correctly and effectively detect spam. GuangJun et al [4] have suggested a technique for spam identification using machine learning predictive models to address this problem. The technique is used in order to identify spam. The testing findings demonstrate the great capability of the suggested strategy to detect spam. 99% accuracy was attained using the suggested strategy, which is high compared to other systems already in use. Thus, the findings imply that the suggested approach is more trustworthy for precise and prompt identification of spam and will secure messaging and email systems.

The spammers target those people who are unaware of these frauds and target them by easily creating phony profiles and email accounts. In their spam emails, they pose as a real person. Therefore, it is necessary to identify spam emails that are fraudulent. Nikhil Kumar et al's [5] project will do this by using machine learning techniques. Their paper will discuss machine learning algorithms and show how to apply them to their data sets. The best algorithm for email spam detection is then chosen because it has the highest precision and accuracy.

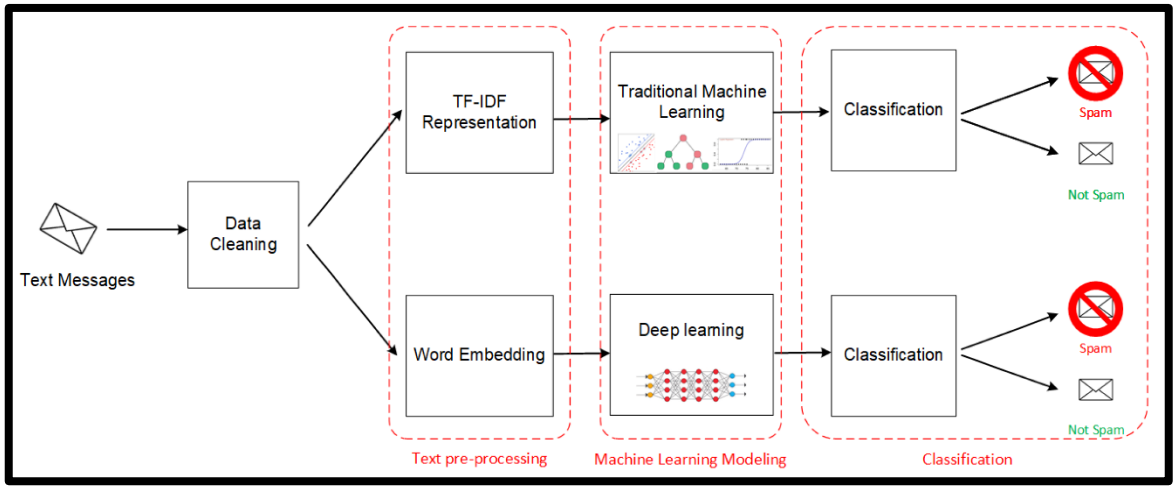



Fig 2.1: Process to Classify the Messages

## 2.2 COMPARISON OF RESEARCH PAPERS STUDIED

| Sr. No | Research Paper | Description | Merits | Demerits |
|---|---|---|---|---|
| 1 | Sridevi Gadde,A .Lakshmanarao,S.S atyanarayana, "SMS Spam Detection using Machine Learning and Deep Learning Techniques", 2021 | In this paper, authors proposed a deep learning model for SMS spam detections. The accuracy is compared of the proposed model with previous work. | Author applied various classification algorithms and achieved an accuracy of 99% with the LSTM model. Experimental results showed that LSTM outperforms previous models for spam detection. | The dataset which is being used is UCI with labels, So they applied various supervised learning algorithms for SMS spam detection. The model is tested on only this dataset |
| 2 | M.Rubin Julis, S.Alagesan, "Spam Detection In Sms Using Machine Learning Through | The research examines various machine learning classifiers on a | The research work involved a careful study on the different filtering algorithms and existing anti-spam tools. | This research however does not solve the problem of analysis and management of report in spam sms |

| | | | | |
|---|---|---|---|---|
| | Text Mining", 2020 | sizable dataset of individual SMS texts | | filter storing, which is far more challenging. |
| 3 | Muhammad Iqbal, Malik Muneeb Abid, Mushtaq Ahmad, Faisal Khurshid, "Study on the Effectiveness of Spam Detection Technologies", 2016 | This paper states different strategies are in place to address spam detection issue, including challenge response models, whitelisting, blacklisting, pattern detection, and various machine learning techniques. | Research focuses on methodically examining the benefits and drawbacks of existing technology for spam detection, and a taxonomy of accepted method is presented. | They proposed a solution of merely combining various techniques of finding a probability score to determine whether the given text is spam or not. However, they don't tell "How to combine those techniques". |
| 4 | Luo GuangJun, Shah Nazir, Habib Ullah Khan, Amin Ul Haq, "Spam Detection Approach for Secure Mobile Message Communication Using Machine Learning Algorithms", 2020 | GuangJun et have suggested a technique for spam identification using machine learning predictive models to address this problem. The technique is used in order to identify spam | 99% accuracy was attained using the suggested strategy, which is high compared to other systems already in use | The model is test on only one dataset. |

# 03.REQUIREMENTS

## 3.1 DESCRIPTION OF REQUIREMENT

Developing a real time chat-application with spam detection capabilities using machine learning methods. The chat application has a spam detection feature driven by AI. The messages can be filtered out as spam or ham using the machine learning technique.

## 3.2 SRS

### 3.2.1 Scope

The scope of this project includes group chat, emoji support, push notifications, AI powered spam detection, profile customization and slash commands. Using the chat application, the user would also be able to react to messages, preview the URL's and videos, and monitor presence and status indications while utilizing the chat application.

The scope of this project does not include features like video calling, voice calling, audio recording, or disappearing messaging.

### 3.2.2 Features

- Register user
- Login for web app
- Group chat
- Emoji support
- Push notifications for messages
- AI powered spam detection
- User settings and profile customization
- User role configuration
- Typing indicators
- Message status indicators
- Message reactions
- URL Preview
- File upload and preview

- Video playback
- Auto complete enabled search
- Slash commands

### 3.2.3 Functional Requirements

Requirement 1: User registration

| | |
|---|---|
| Function | User Registration |
| Pre-condition | User should not be registered already |
| Steps | User fills all the relevant details |
| Post-condition | The user successfully registers for the application |

Requirement 2: Convey a Message

| | |
|---|---|
| Function | User should be able to send and receive messages instantly. |
| Pre-condition | User should be already registered |
| Steps | 1) User access the application<br>2) To send a message, the user clicks on the contact.<br>3) After entering the message, the user clicks the Send button. |
| Post-condition | The message is successfully sent. |

Requirement 3: User's message should be sent to Backend Server.

| | |
|---|---|
| Function | User's message should be delivered to the backend to check for spam/ham. |
| Pre-condition | User should have already sent a message. |
| Steps | User types in the message and hits the "send" button.<br>User's message is then sent to the backend server via the frontend server. |
| Post-condition | The user's message is received at the backend server. |

Requirement 4: Backend server must send the user message for Spam detection.

| | |
|---|---|
| Function | Backend sends the user message for spam detection. |
| Pre-condition | User's message should be delivered at the backend server. |
| Steps | User's message is passed as an argument to the function that uses the trained model to make prediction on spam detection. |
| Post-condition | The user's message has been classified as spam/ham. |

Requirement 5: Backend Server sends the prediction back to the user

| | |
|---|---|
| Function | Backend provides spam/ham response back to the frontend once prediction is complete. |
| Pre-condition | Spam Detection Model has already made a prediction on the received user messages. |
| Steps | Prediction from trained model is sent to backend server.<br>Backend server gives a response to frontend server with spam/ham classification on the current user message.<br>Frontend Server works appropriately depending on the spam/ham classification of the user message |
| Post-condition | The user gets to know if his message was spam or not. |

### 3.2.4 External Interface Requirements

3.2.4.1 User Interfaces

- Front-End Software: React-Native for Android/iOS and ReactJS for WebApp
- Back-End Software: PostgreSQL
- Framework: Django

3.2.4.2 Hardware Interfaces

- Android/ iOS Mobile Phone
- 128 Minimum Ram Required

- Internet/LAN
- Laptop/PC with any operating system

3.2.4.3 Software Interfaces

- Operating System:
  For Mobile Application: Android, iOS
  For Web Application: Windows/MacOS
- Browser: Browser that support Javascript, CGI and HTML.

**3.2.5 Non-Functional Requirements**

- **Performance:**

The following performance requirements are being taken into account:

a) Instant message sending

b) Simultaneous access

c) Instant application loading

d) Registration procedure time of less than one minute

- **Security:**

The user will have access to his personal profile. To protect privacy, the user will be able to manage the ways that people can read his profile and his status indications.

- **Software quality attributes:**

  - **Availability:**

The application needs to be accessible to users around-the-clock.

  - **Portability:**

The application can be accessed through any web application and mobile application (Android and iOS).

  - **Scalability:**

The application should be able to provide service to over 1000 users simultaneously.

- **Usability:**

The application should be user-friendly, easy to operate and should provide a pleasant experience for the users.

### 3.2.6 System Requirements

- **Database Requirement**

PostgreSQL-With over 35 years of continuous development, the durable, open-source PostgreSQL object-relational database system has built a solid reputation for dependability, feature robustness, and performance.

The official documentation has a lot of information outlining how to set up and use PostgreSQL. There are several of helpful resources in the open source community to learn about PostgreSQL, understand how it functions, and locate job possibilities. Learn more about ways to interact with the neighbourhood.

- **Software Requirements**

| Sr. No | Tools/Technology | Use |
|---|---|---|
| 1 | Python | Backend Development |
| 2 | Python Libraries (Scikit-learn, Pandas, etc.) | Spam detection Model |
| 3 | ReactJS | Frontend Development(Web) |
| 4 | ReactNative | Frontend Development(mobile) |
| 5 | Selenium | Web testing |
| 6 | Postman | API Testing |

- **Hardware Requirements**

1. Android/ iOS Mobile Phone

2. 128 Minimum Ram Required
3. Internet/LAN
4. Laptop/PC with any operating system

### 3.3 Analysis Model: SDLC Model to be applied-

The **Agile methodology** is used for implementation of this project.It is a way to manage a project by breaking it up into several phases and 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, both with team members and project stakeholders. In Agile processes, there is constant feedback, allowing team members to adjust to challenges as they arise and it also gives the stakeholders an opportunity to oversee the development of the product and offer continuous feedback. This approach is useful as the requirements change with time and the development is done under the stakeholder's supervision. Unlike traditional approaches for development, this leaves very little room for major errors in the project and is of great benefit to all parties involved.

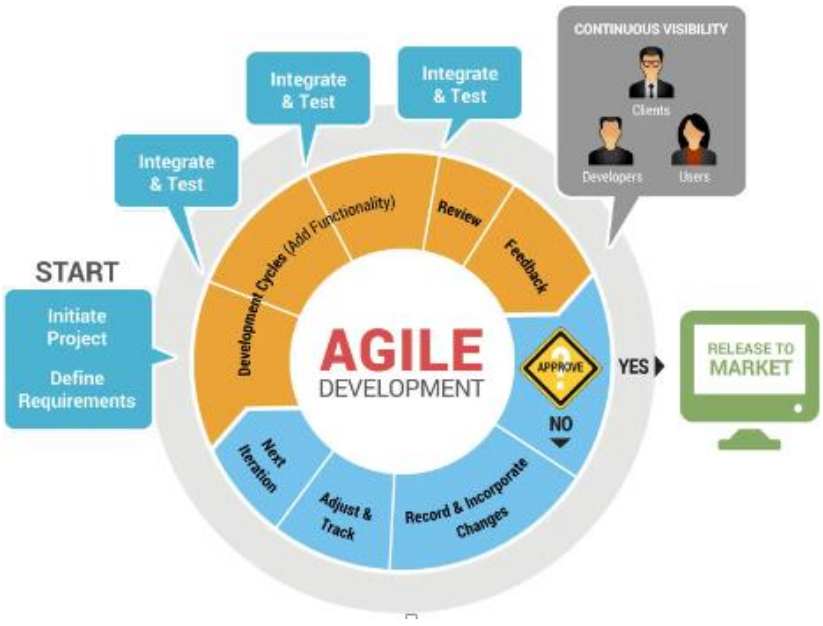



Fig 3.3: Agile Development Methodology

# 04.SYSTEM DESIGN

## 4.1 System Architecture

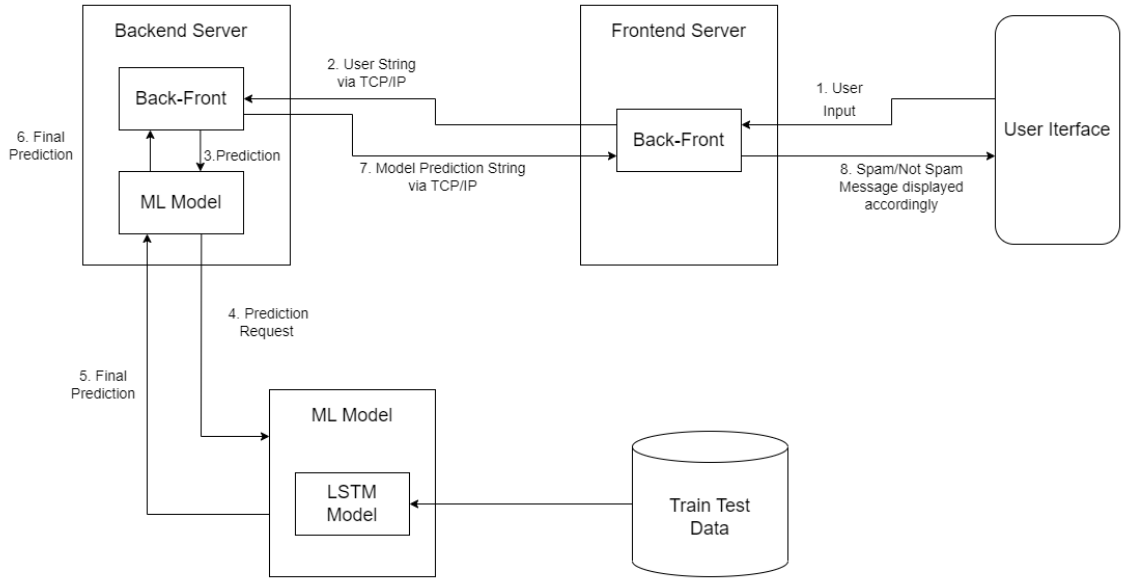



Fig 4.1 System Architecture

## 4.2 Algorithms

**Create a Backend Server:**

1. Host the SMS Spam-Detection trained model on the server.
2. Open a port on which the frontend of the chatbot can communicate on.
3. Accept string/user messages from the frontend and give it as an input to the trained SMS Spam-Detection Model.
4. The model will return its predictions to the backend server, which will then send the predictions back to the frontend.

**Create a Frontend Server:**

1. Connect the frontend server to the frontend of the Chatbot (Mobile App/Web App)
2. For each client, create a thread to keep each client's request isolated from the other clients.

3. Accept user messages when the user hits the “send” button.
4. Send the messages to the backend server via the communicable port established at the backend server and wait for a response.
5. If the response is “spam”, exit.
6. If the response is “ham”, send an appropriate response to the user.

### 4.3 Entity Relationship Diagram


Fig 4.2: ER Diagram

## 4.4 UML Diagrams

- **Use Case Diagram**

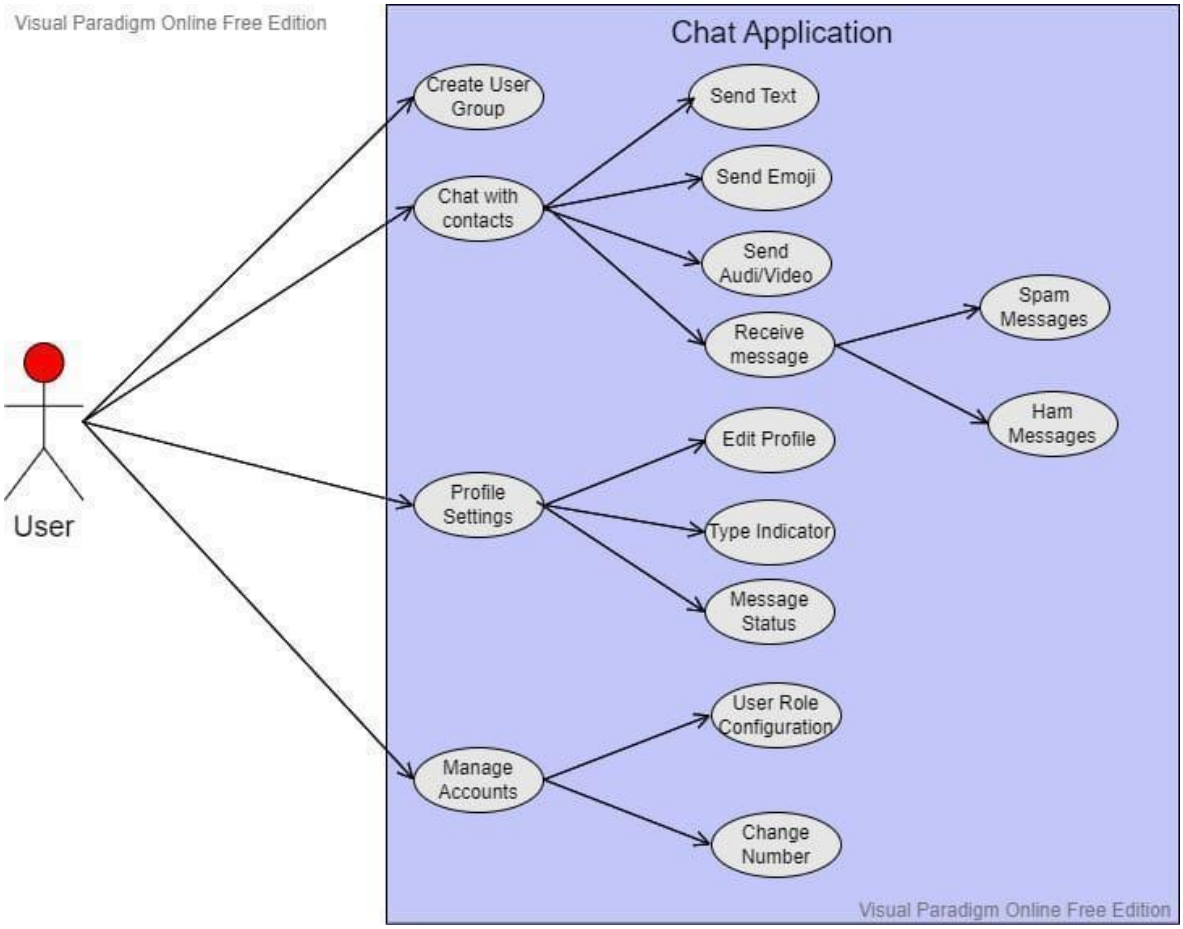



Fig 4.3: Use Case Diagram

- **Activity Diagram**

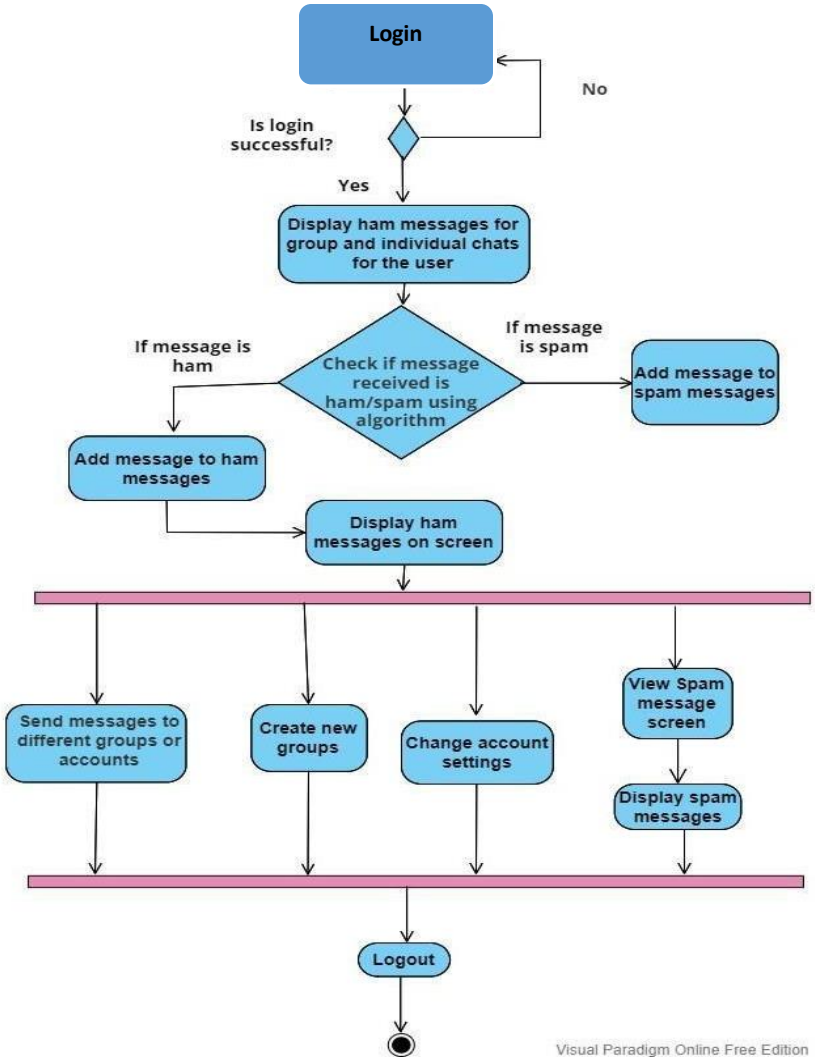



Fig 4.4: Activity Diagram

- **Class Diagram**

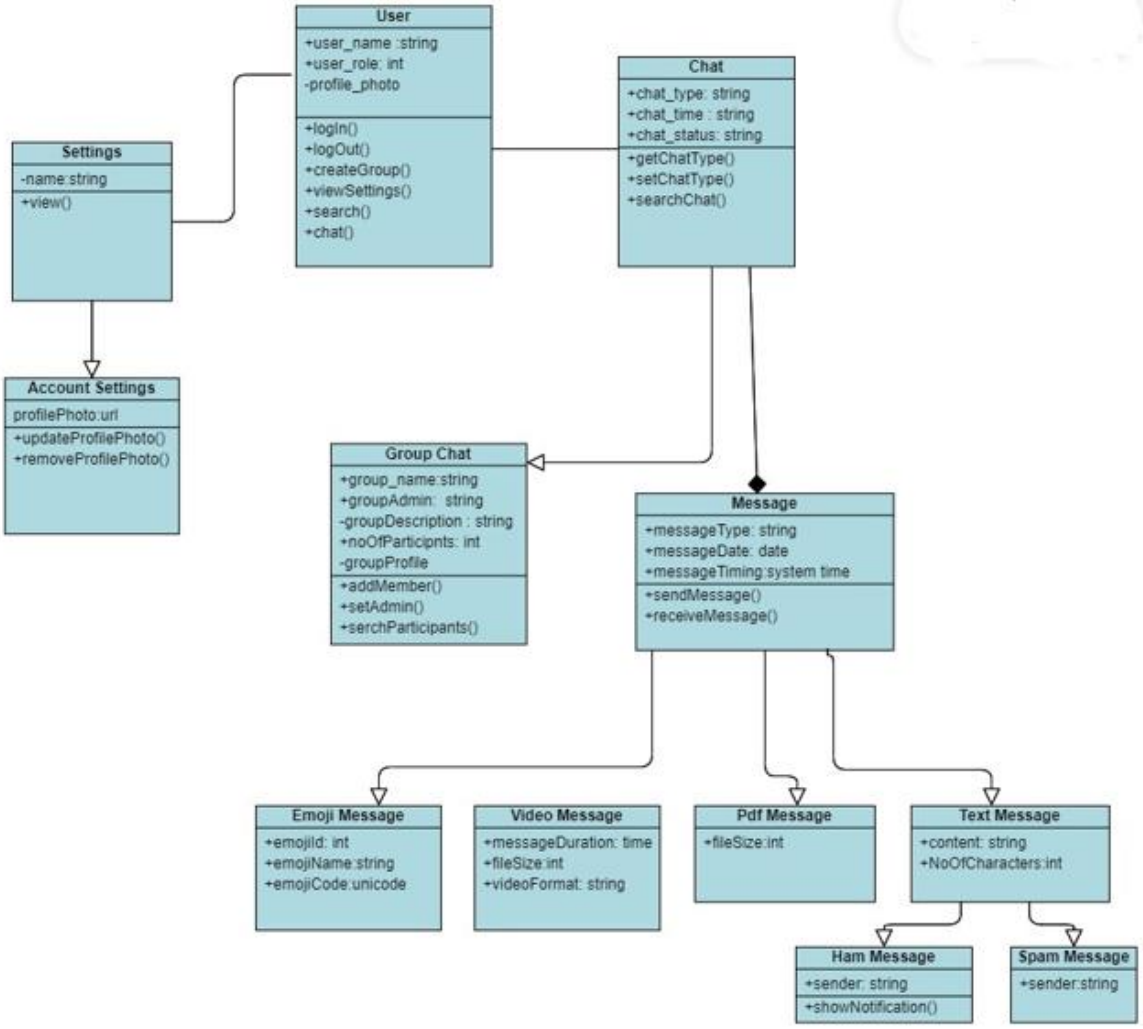



Fig 4.5 Class Diagram

## 4.5 Data Flow Diagrams

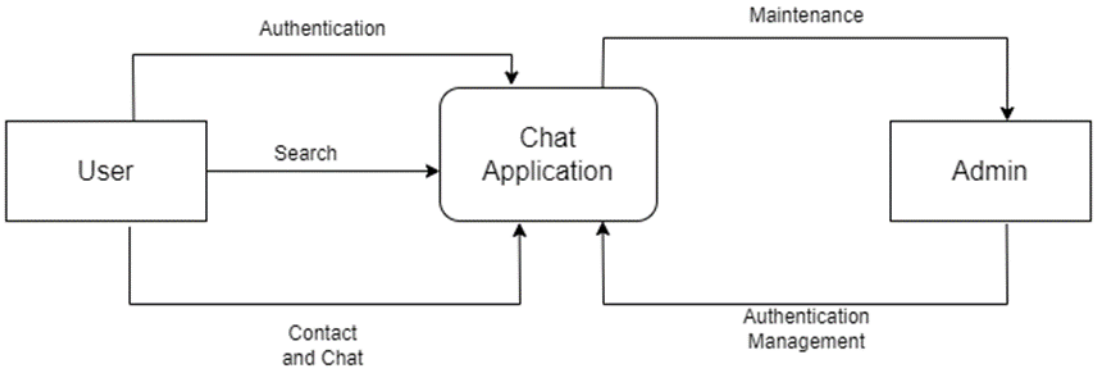



Fig 4.8: Level 0 Data Flow Diagram

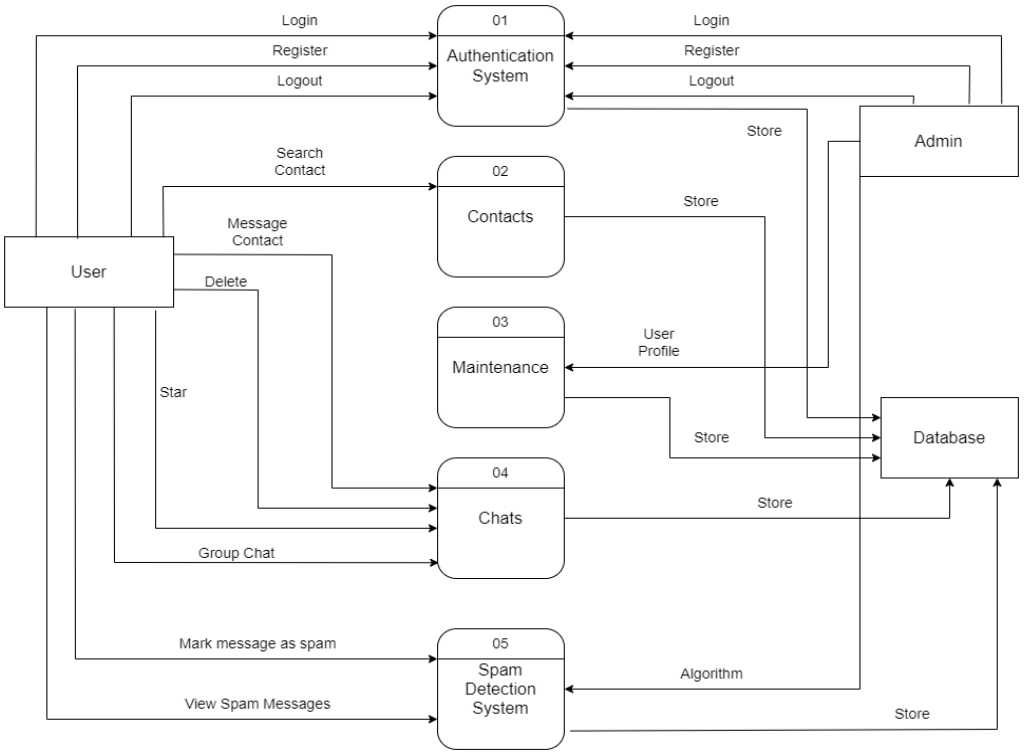
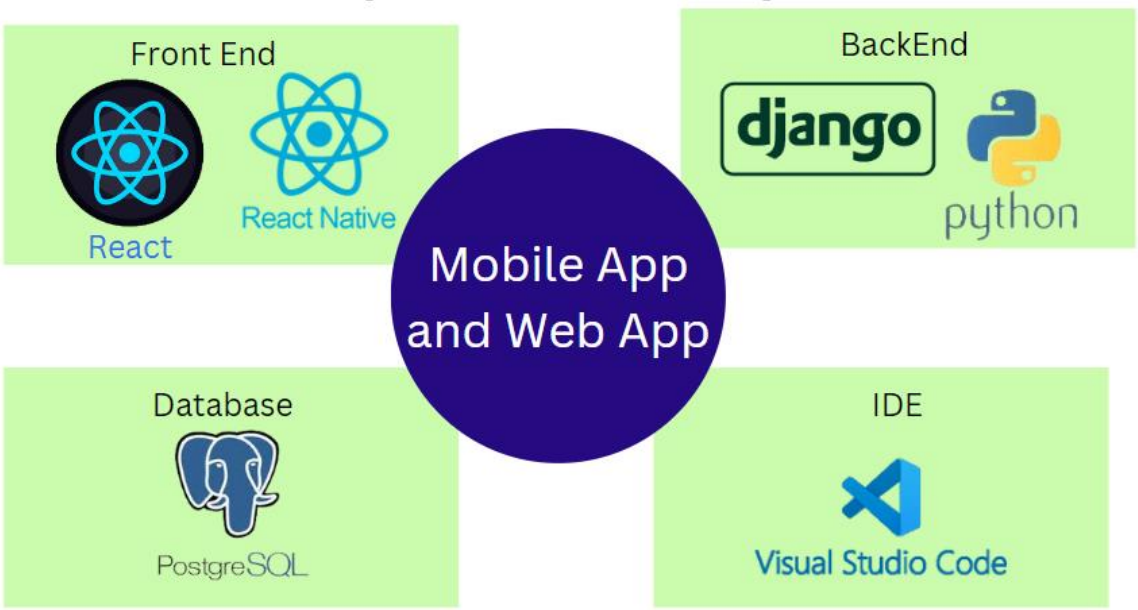



Fig 4.9: Level 1 DFD

# 05.TECHHNOLOGY

## 5.1 TECHNOLOGY STACK




**React:**

React (React.js/ReactJS) is a front-end JavaScript library that is free and open-source for creating user interfaces based on UI components. It is kept up-to-date by Meta (previously Facebook) and a group of independent programmers and businesses. With frameworks like Next.js, React can be the foundation for single-page, mobile, or server-rendered applications. Beyond only UI, React includes a number of extensions for supporting the architectural design of complete applications, including Flux and React Native.

**React Native**

You may create natively rendered mobile apps for iOS and Android using the well-liked JavaScript-based mobile app framework known as React Native (RN). Using the same codebase, the framework enables us to develop applications for numerous platforms. Some of the most popular mobile applications in the world, such as Instagram, Facebook, and Skype, are powered by React Native development. Companies may utilize React

Native to write code once and use it to power their iOS and Android apps. Significant time and resource savings result from this. React, a JavaScript library that was already quite well-liked when the mobile framework was announced, served as the foundation for React Native. The framework allowed frontend developers, who had previously been restricted to working with web-based technologies, to produce comprehensive programmes for mobile platforms.

**Postgresql**

An advanced, enterprise-class, open-source relational database system is PostgreSQL. Both SQL (relational) and JSON (non-relational) querying are supported by PostgreSQL. A highly stable database, PostgreSQL has been developed by the open-source community for more than 20 years. Numerous web, mobile, and analytics applications all use PostgreSQL as their main database.

**Django**

A Python framework called Django makes it simpler to develop websites using Python. It handles the challenging aspects so you can focus on creating your web applications. Don't Repeat Yourself (DRY) principles are emphasized, and it includes ready-to-use features like a login system, database connection, and CRUD (Create, Read, Update, Delete) operations.

**Python**

Python is a popular high-level, general-purpose programming language. Guido van Rossum invented it in 1991, and the Python Software Foundation continued to advance it. Programmers can express their ideas in less code thanks to its syntax, which was designed with code readability in mind. Python is a programming language that enables quick work and more effective system integration.

**Standard tools, APIs and Libraries used:**

1. Pandas
2. Numpy

3. NLTK
4. Tensorflow
5. Keras

## 5.2 TEST PLANS

1. Check that a user can register with a new mobile number. Make sure that if the mobile number is already registered in the application then new account registration must not be allowed with the same mobile number.

2. Check that the chat application follows the design specification and all the sections (spam messages, regular messages, search section and contacts section) are easy to use.

3. The ham messages and spam messages are correctly being classified and the user is receiving notifications for only the ham messages. And that the user can view the spam messages by viewing the spam message screen.

4. Check that the ham messages are received in the order of their arrival time with the latest message being on top.

5. Check the compatibility of the application on different platforms like android, IOS and windows.

6. Test if the user can create a new group by adding multiple people from his contact list. And he/she can give the group name and set group profile photo. Also the user can update the group by adding or removing multiple people from his contact list.

7. Verify that the user can change his/her profile information and also his contact list.

8. Test if the user is able to chat with other users who are offline.

9. Test if the search feature is able to search for accounts, groups and past messages.

10. Test if the application supports emojis and the user can send and receive all types of messages like audio, video, images and text in chat.

11. Verify if the slash commands feature is working correctly.

12. Test if the message status of sent, delivered and read is working correctly and typing indicators and the last seen feature is also working.

| Test Scenario Description | Test Case ID | Test Case Description | Test Steps | Prerequisite | Test Data | Excepted Result | Actual Result | Status |
|---|---|---|---|---|---|---|---|---|
| Verify the login functionality of chat application | TC_01 | Login with valid username and valid password | 1.Launch the app<br>2.Enter valid username<br>3.Enter valid password<br>4.Click on login button | User should have network connection | username:sneha<br>password:P@19wrd | Successful login | Successful login | Pass |
| | TC_02 | Login with valid username and invalid password | 1.Launch the app<br>2.Enter valid username<br>3.Enter invalid password<br>4.Click on login button | User should have network connection | username:rama<br>password:xxxxxxxxx | A popup message box to show "invalid username/password" | Successful login | Fail |
| | TC_03 | Login with invalid username and valid password | 1.Launch the app<br>2.Enter invalid username<br>3.Enter valid password<br>4.Click on login button | User should have network connection | username:xxxxxxx<br>password:123wwHd@# | A popup message box to show "invalid username/password" | A popup message box to show "invalid username/password" | Pass |
| | TC_04 | Login with invalid username and invalid password | 1.Launch the app<br>2.Enter invalid username<br>3.Enter invalid password<br>4.Click on login button | User should have network connection | username:xxxxxxxx<br>password:xxxxxxxxx | A popup message box to show "invalid username/password" | A popup message box to show "invalid username/password" | Pass |
| Verify the register functionality of chat application | TC_05 | Register with valid mobile number,username and password | 1.Launch the app<br>2.Enter valid mobile number not registered in the app<br>3.Enter otp<br>4.Set valid username<br>5.Set valid password<br>6.Click on register | User should have network connection and mobile | mobile:7865456786<br>username:niharika<br>password:Ps239@com | Successful registration | A popup message box to show "invalid password" | Fail |

| Test Scenario Description | Test Case ID | Test Case Description | Test Steps | Prerequisite | Test Data | Excepted Result | Actual Result | Status |
|---|---|---|---|---|---|---|---|---|
| Verify the register functionality of chat application | TC_06 | Register with valid mobile number,username and invalid password | 1.Launch the app<br>2.Enter valid mobile number<br>3.Enter otp<br>4.Set valid username<br>5.Set invalid password<br>6.Click on register | User should have network connection and mobile | mobile:9789787866<br>username:sanket<br>password:xxxxxxxxx | A popup message box to show "invalid password" | A popup message box to show "invalid password" | Pass |
| | TC_07 | Register with valid mobile number,password and invalid username | 1.Launch the app<br>2.Enter valid mobile number<br>3.Enter otp<br>4.Set invalid username<br>5.Set valid password<br>6.Click on register | User should have network connection and mobile | mobile:9789787866<br>username:xxxxxxxx<br>password:user456@80 | A popup message box to show "invalid username" | A popup message box to show "invalid username" | Pass |
| | TC_08 | Register with invalid mobile number | 1.Launch the app<br>2.Enter invalid mobile number | User should have network connection and mobile | mobile:xxxxxxxxxxx | A popup message box to show "invalid mobile number" | A popup message box to show "invalid mobile number" | Pass |
| Create new group | TC_09 | Create new group for users | 1.Launch the app<br>2.Go to Settings and click on create new group<br>3.Give a group name<br>4.Add contacts to the group | User should have network connection and at least 2 contacts to add the group | Group name "Friends"created with 3 users | Group created successfully | Group created and opened | Pass |

| Test Scenario Description | Test Case ID | Test Case Description | Test Steps | Prerequisite | Test Data | Excepted Result | Actual Result | Status |
|---|---|---|---|---|---|---|---|---|
| Test the classification of ham and spam messages | TC_10 | Check if the ham messages are identified and user receives notification on receiving these messages | 1.Launch the app<br>2.Send a ham message to another user using app | User should have network connection | message="Hey, what's up?" | Message correctly identified as ham and receiver gets a notification | Message correctly identified as ham and receiver gets a notification | Pass |
| | TC_11 | Check if the spam messages are identified correctly and user does not receive notification | 1.Launch the app<br>2.Send a spam message to another user using app | User should have network connection | message="Chance to win an IPhone! Hurry and claim now." | Message correctly identified as spam and is shown on spam screen | Message correctly identified as spam and is shown on spam screen | Pass |
| Change account settings | TC_12 | Test if the user can change his/her username to a valid username | 1.Go to settings<br>2.Click on account_info<br>3.Delete the existing username<br>4.Enter new valid username<br>5.Click on update button | User should have network connection | username="Ritika" | Username gets updated with a popup message "username updated" | Username gets updated with a popup message "username updated" | Pass |
| | TC_13 | Test if the user cannot change his/her username to an invalid username | 1.Go to settings<br>2.Click on account_info<br>3.Delete existing username<br>4.Enter invalid username<br>5.Click on update button | User should have network connection | username="777" | Username is not updated and popup message box showing "invalid username" | Username gets updated with a popup message "username updated" | Fail |

# 06.IMPLEMENTATION ASPECTS

## 6.1 OVERVIEW

**Naive-Bayes Classifier**

Simple classifier construction method. Assumes that, given a class variable, the value of one feature is unrelated to the value of any other feature.

**Logistic Regression**

For classification tasks, logistic regression employs the sigmoid and logit functions. S-shaped curves are used to anticipate the output variable.

**K-nearest Neighbors**

Based on distance calculations, this classifier is easy to use and effective. The new data point is categorized based on the count of neighbors class and the k-nearest neighbors that are identified.

**Decision Tree Classifiers**

Uses a system of rules in the same way as humans do to make judgments. The idea is to build yes/no questions using the dataset's attributes, split the dataset repeatedly, and then isolate all the data points that correspond to each class.

**Random Forest**

To determine the class of a new data point, the Random Forest classifier consults with several different decision trees. It is an ensemble approach, which means that it creates a final output by combining the results of multiple trees.

**SVM (Support Vector Machine)**

A hyperplane is built in SVM based on the classification process. To locate the hyperplane, some data points are used as support vectors.

**LSTM (Long Short Term Memory)**

Artificial neural networks include recurrent neural networks. The previous state output in RNNs is used to create the current state input. But vanishing gradient descent is a concern for conventional RNNs. The gradients of the loss function in neural networks approach zero when more layers with

specific activation functions are added, making the network challenging to train. LSTM addresses the issues with conventional RNNs. Text mining issues are best suited for LSTMs.

## 6.2 IMPLEMENTATION BASIC LOGIC

### DataSet

We make use of an SMS spam dataset that Almeida and Hidalgo suggested. The number of records in this dataset is roughly 5,574. It has SMS text messaging in it. Talks in the English language that use text and numbers in phrases of varying lengths. This dataset's records are all already labelled. The normal communications are labelled as 0 and the spam ones as 1 (747 records) (4,827 records).

### Data Preparation

Natural Language Processing (NLP) is used in this procedure to pre-process natural language data. NLP is a process that enables computers to comprehend natural language in a manner that is comparable to that of a human [15]. It uses a variety of methods to preprocess data into an understandable format for computers. In this study, we employ NLP approaches to convert SMS text data into sequence data so that we may use it to create SMS classification models using the LSTM and GRU algorithms. Additionally, we pre-process data using word tokenization, padding, truncating, and word embedding algorithms. The following is a description of each approach we employ for data pre-processing in detail.

### Word Tokenization

The process of turning the words in a sentence into index values denoted by a number is called word tokenization. In this procedure, we generate a word tokenizer using a specified number of interesting vocabulary terms. Word tokenizer is used to turn the words of a sentence into sequence data after being created. Tokenization converts words into indexes and sets the index to 0 for terms that are unknown.

**Padding**

In this procedure, we uniformly lengthen every sequence in the dataset for LSTM and GRU training.We determine the message length that is optimised using (1). After the message's length has been optimised, we pad data that is shorter than the ideal length by inserting a zero at the beginning of the sequence until it is the same length as the ideal one.

**Word Embedding**

This method transforms a word sequence that has already undergone preprocessing into a vector representation known as an embedding space that has more dimensions than the standard word data used to train LSTM and GRU algorithms. After padding and truncating the data, we employ the word embedding technique, setting the embedding size to 32, to add more dimensions for the data in order.

**Modeling**

We create categorization models for SMS spam based on LSTM and GRU algorithms, two deep learning techniques.

**LSTM**

Hochreiter and Schmidhuber created the LSTM in 1997. By including cell states for remembering or forgetting data, it enhances the fundamental RNN method that resolves the vanishing problem. Cell gates are a type of structure found in the cell states. The input gate, forget gate, memory-cell state gate, and output gate are the four components that make up the cell gates. The input is utilised as a gate to control whether the input data is valuable enough to maintain or not. The prior hidden state that needs to be preserved in the memory cell of the current hidden state is controlled by the forget gate. Based on the data from the input gate and the forget gate, the memory-cell state gate is utilised to update the data. Based on the state of the memory cells, the output gate computes the network's output data.

A memory cell known as a "cell state" that preserves its state over time plays a key role in an LSTM model. The horizontal line that passes through the top of the diagram below represents the cell state. It can be pictured as a conveyor belt across which data simply and

unaltered passes. The cell state may contain important information while the sequence is processed. As a result, even knowledge from earlier time steps might reach later time steps, diminishing the impact of short-term memory. Information is added to or withdrawn from the cell state via gates as the cell state travels. The gates, which determine which information is permitted on the cell state, are various neural networks.

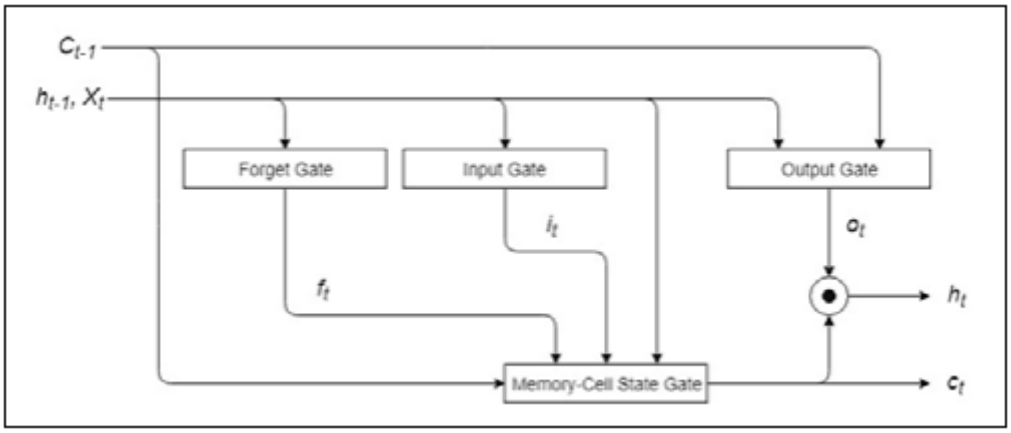



Fig 6.1 LSTM Memory Cell

**Gated Recurrent Unit (GRU)**

By utilizing update gates and reset gates, the GRU is a deep learning algorithm that is modified from the LSTM method to reduce the complexity of the algorithm's structure . The update gate is used to regulate how much of the concealed state is sent on to the following state. The importance of the prior hidden state information is determined by the reset gate.

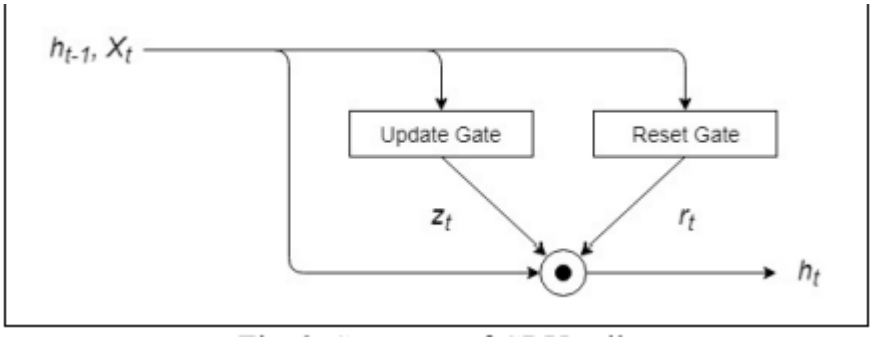



Fig 6.2 Gated Recurrent Unit

## 6.3 ALGORITHM

### Creating SMS Spam-Detection Model using LSTM:

1. Import SMS spam detection dataset from kaggle.

   https://www.kaggle.com/uciml/sms-spam-collection-dataset

2. Label each SMS with spam or ham accordingly.
3. Perform pre-processing operations on the dataset.
4. Remove unnecessary columns
5. Remove stopwords
6. Vectorize inputs
7. Tokenization of inputs
8. Pad each input with 0s so that each input is of equal length
9. Create a 6-layer LSTM model with Dense and Dropout layers in between to avoid overfitting.
10. Use Adam optimizer and binary-cross entropy loss function to adjust weights of the model during training optimally.
11. Train the model using model.fit() function.
12. Store the trained weights.

### Create a Backend Server:

1. Host the SMS Spam-Detection trained model on the server.
2. Open a port on which the frontend of the chatbot can communicate on.
3. Accept string/user messages from the frontend and give it as an input to the trained SMS Spam-Detection Model.
4. The model will return its predictions to the backend server, which will then send the predictions back to the frontend.

### Create a Frontend Server:

1. Connect the frontend server to the frontend of the Chatbot (Mobile App/Web App)

2. For each client, create a thread to keep each client's request isolated from the other clients.
3. Accept user messages when the user hits the "send" button.
4. Send the messages to the backend server via the communicable port established at the backend server and wait for a response.
5. If the response is "spam", exit.
6. If the response is "ham", send an appropriate response to the user.

# 07.RESULTS

## 7.1 SCREENSHOTS

```
# check_spam = ["Free entry in 2 a wkly comp to win FA Cup finals"]
# check_spam = ["Nah I don't think he goes to usf"]
check_spam_msg = input()
check_spam_msg = [check_spam_msg]
sms_spam_tokenizer.fit_on_texts(X)
check_spam_tokenized = sms_spam_tokenizer.texts_to_sequences(check_spam_msg)
print(prediction(check_spam_tokenized, sms_spam_tokenizer))
check_ham_msg = input()
check_ham_msg = [check_ham_msg]
# print(check_spam_ham_msg)
check_ham_tokenized = sms_spam_tokenizer.texts_to_sequences(check_ham_msg)
# print("Text               : ", check_spam_ham_msg[0] )
# print("Numerical Sequence : ", check_spam_tokenized[-1])

print(prediction(check_ham_tokenized, sms_spam_tokenizer))
```

```
Free entry in 2 a wkly comp to win FA Cup finals
1/1 [==============================] - 0s 18ms/step
SPAM!
Nah I don't think he goes to usf
1/1 [==============================] - 0s 17ms/step
HAM!
```

Fig 7.1: LSTM Model for Spam Detection

# 08.CONCLUSION AND FUTURE WORK

## 8.1 CONCLUSION

We provide SMS spam classification models based on LSTM and GRU, two deep learning algorithms. NLP methods were utilised to pre-process SMS.

Word tokenization, padding, truncating, and word embedding techniques are used to put text data into sequence. Additionally, using deep learning algorithms like LSTM and GRU, we created a model. Last but not least, we assessed models using a test set taken from the SMS spam dataset.

## 8.2 FUTURE WORK

Classifying the messages as SPAM messages or HAM messages is done with the help of deep learning technique LSTM.

We will host this spam detection algorithm on the backend server so that the messages from the chat application will be sent to the server to check whether it is HAM or SPAM.

We will start working on the UI of the chat application which is both web as well as mobile application. We will store the messages with the help of PostgreSQL database.

We will add the various functionalities of the chat application. Frontend server will do all the preprocessing of the messages and the data will be store on the server.

We will test the applications created and will host them on the appropriate platform for the end-users to use them.

# APPENDIX B: PLAGIARISM REPORT

turnitin

**Similarity Report ID:** oid:8054:36081493

PAPER NAME

**REPORT_SEM1 (1) (4).docx**

| | |
|---|---|
| WORD COUNT | CHARACTER COUNT |
| **4455 Words** | **24153 Characters** |
| PAGE COUNT | FILE SIZE |
| **32 Pages** | **1.4MB** |
| SUBMISSION DATE | REPORT DATE |
| **May 24, 2023 7:51 PM GMT+5:30** | **May 24, 2023 7:52 PM GMT+5:30** |

**5% Overall Similarity**

The combined total of all matches, including overlapping sources, for each database.

- 4% Internet database
- 3% Publications database
- Crossref database
- Crossref Posted Content database

**Excluded from Similarity Report**

- Submitted Works database
- Bibliographic material
- Quoted material
- Cited material

# REFERENCES

[1] M.Rubin Julis, S.Alagesan, “Spam Detection In Sms Using Machine Learning Through Text Mining”, 2020

[2] . F. Kai Petersen, Shahid Mujtaba, Michael Mattsson, "Systematic Mapping Studies in Software Engineering," 2008.

[3] Muhammad Iqbal, Malik Muneeb Abid, Mushtaq Ahmad, Faisal Khurshid, “Study on the Effectiveness of Spam Detection Technologies”, 2016

[4] Luo GuangJun, Shah Nazir, Habib Ullah Khan, Amin Ul Haq, “Spam Detection Approach for Secure Mobile Message Communication Using Machine Learning Algorithms”, 2020

[5] Nikhil Kumar, Sanket Sonowal, Nishant, “Email Spam Detection Using Machine Learning Algorithms”, 2020

[6] Pumrapee Poomka, Wattana Pongsena, Nittaya Kerdprasop, and Kittisak Kerdprasop,―SMS Spam Detection Based on Long Short-Term Memory and Gated Recurrent Unit , 2019

[7] https://www.analyticsvidhya.com/blog/2021/05/sms-spam-detection-using-lstm-a-hands-on-guide/