PROJECT REPORT

ON

ONLINE CHATTING WEBSITE WITH SPAM MESSAGE DETECTION

Submitted in partial fulfillment of the requirement for the award of degree in

MASTER OF COMPUTER APPLICATIONS

of the

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Submitted by

ANOOP B K

(NCE21MCA-2009)

Under the guidance of

Mr. PRAMOD K, MCA

ASSOCIATE PROFESSOR

DEPARTMENT OF MCA

NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE,

(NAAC Re-Accredited with "A" grade)

PAMPADY, THIRUVILWAMALA, THRISSUR-680567

MAY 2023

PROJECT REPORT

ON

ONLINE CHATTING WEBSITE WITH SPAM MESSAGE DETECTION

Submitted in partial fulfillment of the requirement for the award of degree in

MASTER OF COMPUTER APPLICATIONS

of the

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Submitted by

ANOOP B K

(NCE21MCA-2009)

Semester 4 MCA (2021-23)

Under the guidance of

Mr. PRAMOD K, MCA

ASSOCIATE PROFESSOR

DEPARTMENT OF MCA

NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE,

MAY 2023

NEHRU COLLEGE OF ENGINEERING AND RESEARCHCENTRE

DEPARTMENT OF MCA

COLLEGE VISION

To mold true citizens who are millennium leaders and catalysts of changethrough excellence in education.

COLLEGE MISSION

NCERC is committed to transform itself into a center of excellence inLearning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values. We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

DEPARTMENT VISION

To create a school of distinction for the PG students, prepare them tobe industry-ready, and achieve Academic excellence by continuous endorsement of the faculty team in terms of Academics, Applications & Research.

DEPARTMENT MISSION

The Department of Computer Applications strives to provide quality and competency-based education and fine-tune the younger generation through Curricular, Co-Curricular and Extra-curricular activities so as to encounter the Professional and Personnel challenges ahead with Pragmatic skills & courage , thereby ‘Creating the True Citizens’.

DECLARATION

I hereby declare that the project report entitled "**ONLINE CHATTING WEBSITE WITH SPAM MESSAGE DETECTION**" Submitted to the **DEPARTMENT OF MCA at Nehru College of Engineering and Research Centre** in partial fulfilment of the requirement for the award of degree in **MASTER OF COMPUTER APPLICATIONS** from **APJ ABDUL KALAM TECHNLOGICAL UNIVERSITY**, is a record of original work done by me under the guidance of **Mr. PRAMOD K**, Associate Professor of theDepartment of MCA, during my Fourth Semester MCA course period 2023.

PAMPADY **ANOOP B K**

DATE:

NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE, PAMPADY

CERTIFICATE

This is to certify that, the project work entitled **“ONLINE CHATTING WEBSITE WITH SPAM MESSAGE DETECTION”** has been presented by **ANOOP B K, NCE21MCA-2009** of Fourth Semester MCA in Partial Fulfillment of the requirement for the award degree **MASTER OF COMPUTER APPLICATIONS, APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY.**

We also certify that the work done is original.

Project Guide **Head of the Department**

Principal **External Examiner**

ACKNOWLEDGEMENT

First and most, I thank the **God Almighty** for showing me the path to the completion of seminar work. I thank **Prof. Dr. Karibasappa Kwadiki**, The Principal of **NCERC**, for providing good atmospherefor project completion and presentation. I thank **Dr. Sudheer S Marar**, Head of MCA Department, NCERC for his valuable suggestions and support throughout the project work. I wish to acknowledge my deep sense of gratitude and heartfelt thanks to My project Guide **Mr. PRAMOD K**, Associate professor, MCA Department for his valuable suggestions, precious time, kind hearted motivations throughout the project work period. I express my immense gratitude to all my friends,without whom I would have never been able to do my project well.

ABSTRACT

Spam is a hot topic for decades. After all the tech advancements still all of us stumble upon spam messages then and now. Spam has become more realistic where a normal people could not distinguish whether it is real or not. Random job offers appearing on your WhatsApp messenger from unknown source, Spam messages on every comment section of almost all the social medias. Occasionally, individuals may receive spam messages from their friends. Tech giants are spending millions to keep these spammers from their application. The project titled “ONLINE CHATTING WEBSITE WITH SPAM MESSAGE DETECTION” is aimed to provide a spam free chatting system to users. To detect and avoid spam messages we can use natural language processing and machine learning. In this system we developed a machine learning model using TF-IDF Vectorizer algorithm and Decision Tree Classifier to predict a SMS is a spam or non-spam(ham) and it was discovered that the model out performs existing models. The TF-IDF algorithm is a common technique in natural language processing (NLP) for text data preprocessing and feature extraction. This algorithm is used for feature extraction in this system. A decision tree classifier is a type of supervised learning algorithm that is commonly used in machine learning for classification tasks. Decision trees can be trained quickly and can handle large datasets with many features. This is important in the context of SMS spam detection, where there may be a large number of messages to classify and many features to consider. This model is used to predict the spam messages in this online chatting website. In this website user can register and login if his request of registration is accepted by the admin. The user can send friend request to other users and can send SMS to their friends. The system evaluates messages and predicts whether they are spam or ham.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

In today’s digital age, online communication has become an essential part of our daily lives. With the rise of social media platforms and online messaging apps, people now prefer to communicate with each other online rather than through traditional methods. However, while online communication has its benefits, it also has its drawbacks – spam messages. Spam messages are unwanted and irrelevant messages that fill up our inboxes and make it difficult to find and read important messages. To address this problem, we are proposing an online chatting website called Spam Free, which incorporates spam message detection. Our goal is to provide users with a safe and secure online chatting experience by automatically detecting spam messages. To achieve this, we are developing a highly effective machine learning model that has a high accuracy rate for predicting spam messages, and implementing this model into the chat system. By using this model, users can determine whether a message is spam or not, allowing them to have a safe chat with their friends and protecting them from fraudulent activities and security risks. Overall, this website provides a spam-free chatting experience to users.

1.1 Background

The proliferation of online communication channels has made it easier for people to stay in touch with each other. Online chatting websites have become increasingly popular as they allow users to communicate with their friends and family in real-time, regardless of their location. However, as the popularity of these platforms has increased, so has the problem of spam messages. Spam messages are unsolicited messages that are sent to a large number of users, often with the intention of promoting a product or service. These messages can be annoying and disruptive, making it difficult for users to find and read important messages. Moreover, spam messages can also be malicious, containing links to phishing websites or malware. It is quite easy for a hacker to compromise any one’s device 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 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, we propose a chatting 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

The motivation behind our proposed system is to address the growing problem of spam messages in online communication channels. Spam messages are a major concern for online users as they not only disrupt the flow of communication but also pose a security threat. The impact of spam messages can be significant, as they can prevent users from receiving important messages and make it difficult for them to find and read the messages that matter. Furthermore, some spam messages may contain malicious links that can lead to the compromise of the user’s personal information, such as login credentials or financial information. Our proposed project aims to provide users with a safe and secure online chatting experience by automatically detecting and filtering out spam messages. This will not only improve the user experience but also enhance the security of online communication. Overall, the motivation behind our proposed project is to create a reliable and secure online chatting experience for users. By integrating spam message detection, we aim to improve the user experience and reduce the security risks associated with online communication.

1.2 Objective

The idea of this project came into existence because of the growing problem of spam messages in online communication channels, which disrupts the user experience and poses a security threat.

The objectives of this project are:

- Develop an online chatting website: We aim to create a user-friendly website that allows users to communicate with their friends in real-time.
- Implement spam message detection: We will integrate various techniques such as natural language processing and machine learning to automatically detect and filter out

spam messages.

- Another objective is to conclude the best model which is more efficient and gives fast and accurate result by using Decision Tree Classifier.
- Enhance security: By filtering out spam messages, we aim to reduce the security risks associated with online communication.

1.3 Contribution

The major contributions in this project are:

➢ Designed and developed a system that aims to provide spam-free chatting for users.

➢ Designed and developed a machine learning model for detecting spam messages with a high level of accuracy.

1.4 Report Organization

The project report is divided into six sections. Section 2 describes literature survey. Section 3 describes the methodology and section 4 describes agile methodology used for implementing the project. Section 5 gives the results and discussions. Finally, Section 6 gives the conclusion.

Chapter 2

Literature Survey

Spam message detection has been a topic of extensive research in the field of natural language processing and machine learning. Various techniques and algorithms have been proposed for identifying and filtering out spam messages from the vast amount of online communication. In this literature survey, we review some of the previous studies that have been conducted in this area.

Li and Yang were done research on “Content-based spam message detection using deep learning”. They proposed a content-based approach for spam message detection using deep learning algorithms. They used a neural network model to classify messages as spam or non-spam based on their content. The model was trained on a large dataset of messages and achieved high accuracy in detecting spam messages. The study highlights the effectiveness of deep learning algorithms for spam message detection, particularly in identifying complex and novel spam messages. However, the approach requires a large dataset for training and may not be suitable for small datasets.

Alzahrani and Alshammari were done research on “A hybrid approach for spam detection in social media”. They proposed a hybrid approach for spam message detection, combining rule-based and content-based filtering techniques. The rule-based filter used a set of predefined rules to identify spam messages, while the content-based filter used machine learning algorithms to analyze the content of messages. The study found that the hybrid approach outperformed either technique alone, highlighting the importance of using multiple techniques for spam message detection. However, the approach may require significant computational resources and may not be efficient for real-time spam detection.

Srivastava et al. were done research on “Performance comparison of machine learning algorithms for spam detection”. In this study they evaluated the effectiveness of machine learning algorithms for spam message detection. They compared the performance of Naive Bayes, Decision Tree Classifier, and Support Vector Machine on a dataset of spam and non-spam messages. The study found that all three algorithms achieved high accuracy in detecting spam messages, with Decision Tree Classifier performing the best. The study highlights the importance of choosing the right machine learning algorithm for spam message detection.

Zhang et al. were done research on “A personalized spam message filter based on user preference.” They proposed a personalized spam message filter that allows users to customize their filter settings based on their preferences. Users could specify keywords and domains to block or allow, and the system would adapt to their preferences over time. The study found that the personalized filter improved user satisfaction and reduced the likelihood of false positives in spam message detection. The study highlights the importance of user customization and adaptability in spam message detection. However, the approach relies heavily on user input and may not be effective for users with limited knowledge of spam messages.

Luo et al. were done research on “A spam message detection method based on user behavior analysis.” They proposed a spam message detection model based on user behavior analysis. Their approach leveraged user feedback and behavior patterns to identify and filter out spam messages. The study found that their approach achieved high accuracy in detecting spam messages and reduced the likelihood of false positives. The study highlights the importance of user feedback and behavior analysis in spam message detection. However, the approach may require a significant amount of data and user participation to be effective.

Overall, these studies provide valuable insights into the development of effective spam message detection systems, including the use of machine learning algorithms, content-based filtering techniques and user behavior analysis. However, each approach has its strengths and limitations, and choosing the right approach depends on the specific needs and requirements of the system.

Chapter 3

Methodology

3.1 Introduction

The project online chatting website with spam message detection, consisting of two major modules – admin and user – and implementation modules. Users can register and log in, with their requests being accepted by the admin. Once logged in, users can send and receive friend requests and chat with their friends by sending SMS messages, which may be spam or ham. Users can view the status of the messages to determine if they are spam or ham. To detect spam messages, a machine learning model was developed and implemented in the chat system.

Spam messages are unsolicited messages sent in bulk to a large number of recipients, often with the intent of advertising a product or service. These messages can be sent via email, text message, social media, or any other communication medium. Spam messages are usually unwanted and intrusive, and can be harmful if they contain links to malicious websites or attempt to steal personal information. In the context of this project, spam messages refer specifically to unwanted and potentially harmful messages sent through an online chatting system. To detect the spam messages, we use the following methodologies:

Natural language processing

Natural language processing (NLP) is a branch of artificial intelligence (AI) that deals with the interaction between human language and computers. It involves the use of computational algorithms to process and analyze human language, including speech and text. NLP techniques are used in a wide range of applications, such as sentiment analysis, machine translation, chatbots, and information retrieval. In the context of spam detection, NLP can be used to analyze the content of messages and identify patterns that are characteristic of spam. This can include features such as specific keywords or phrases, unusual message formatting, or other factors that suggest the message is not legitimate. Natural language processing (NLP) techniques can be used for data cleaning. In text data, there can be various types of noise or inconsistencies that can affect the quality of data, such as misspellings, abbreviations, contractions, punctuation, and more. NLP can be used to identify and correct such errors, as well as to standardize the format of text data. For example, NLP techniques such as stemming or lemmatization can be used to reduce words to their base form, and stop words can be removed to eliminate common words that do not contribute to the meaning of the text. In summary, NLP can be an effective tool for data cleaning and preparation in text-based applications.

Term Frequency Inverse Document Frequency (TF-IDF)

In this system we are using the TF-IDF for feature extraction. 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 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 do not give any relevant information about a particular document type or class. Figure represents the TF-IDF mathematical formula**.**

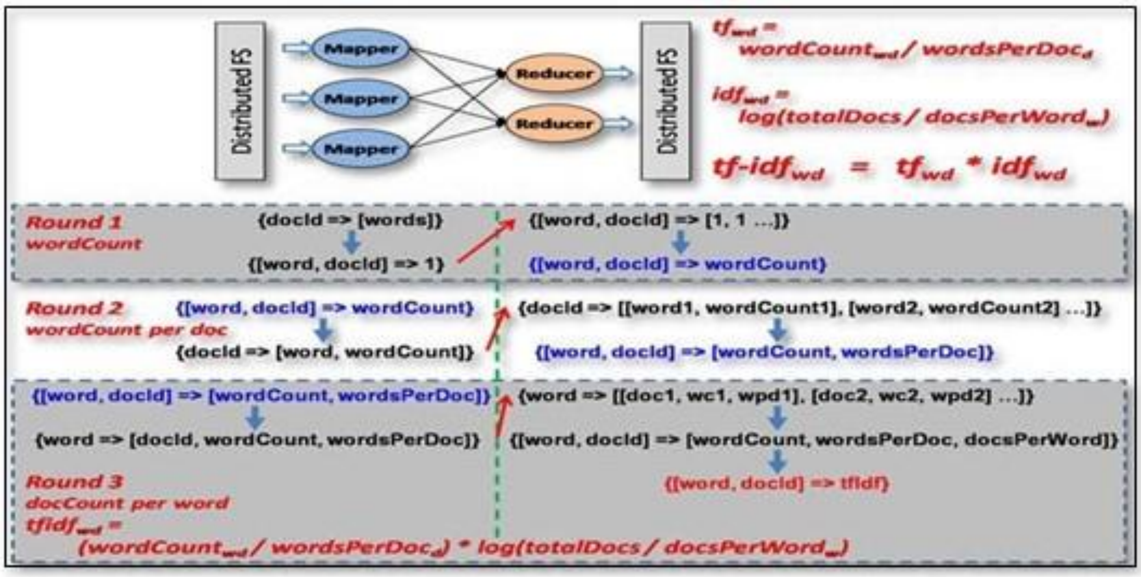

Distributed FS
Mapper
Mapper
Mapper
Reducer
Reducer
Distributed FS
tf_wd = wordCount_wd / wordsPerDoc_d
idf_wd = log(totalDocs / docsPerWord_w)
tf-idf_wd = tf_wd * idf_wd
Round 1
wordCount
{docId => [words]}
{[word, docId] => 1}
{[word, docId] => [1, 1 ...]}
{[word, docId] => wordCount}
Round 2
wordCount 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)

Fig 3.1.1: Equation for calculating TF-IDF

Decision Tree Classifier

The machine learning technique used in this system is Decision Tree Classifier. Decision Tree is a popular machine learning algorithm used for classification and regression tasks. In a Decision Tree, the data is split recursively into subsets based on the values of one or more features, and a tree-like model is created that maps the features to their target labels. When used for classification, the Decision Tree algorithm builds a model that predicts the class label of a new instance based on the features of the instance. The tree consists of nodes, where each node represents a decision based on a feature value, and edges that connect the nodes represent the outcome of that decision. The leaves of the tree represent the class labels or target values.

3.2 Modules and Descriptions

The system comprises of 2 major modules and implementation modules. The two major modules and their sub modules are as follows:

1. Admin

- Login – Admin can login to the site by giving the username and password.
- View user list – Admin can view all users list.
- Manage user – Admin can accept and reject user registrations under manage user.
- Block user – Admin can block and unblock users.
- View blocked users – Admin can view the list of blocked users under this section.

2. User

- Register: A user can register to the website by entering their credentials like username gender, DOB, PHNO, email, password and uploading profile picture. After registering a request will send to the admin. The user can successfully register if the admin approves the request.
- Login: After the acceptance of the request user can login to the website by entering their username and password.
- Search friends: In this section user can search their friends by entering their username.
- Send friend requests: Under this section user can view all the users who have registered in the site and send friend request.
- Manage friend requests: Under this section user can view all the friend request from other users and accept or reject the friend requests.
- Chat with friends: Under this section user can view the friends and can chat with them. View chats and reply the messages. User can view the chat messages is spam or ham through the status.
- View messages and replay messages: Under the chat section user can view the

messages send by their friends and send replay.

- View status: Under the chat section user can view the received message is spam or ham.

Implementation Modules

The system comprises of the following implementation modules:

- **Data Collection:** This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get, the better our model will perform. There are several techniques to collect the data, like web scraping, manual interventions etc. In this phase authors have collected a dataset based on which they have performed the experimentation from Kaggle Repository.
- **Dataset:** The dataset that was collected includes two columns: "Label" and "Content." The "Content" column is a collection of text messages, while the "Label" column specifies whether the corresponding message is spam or ham (non-spam). The dataset contains a total of 5572 messages, of which 749 are spam messages and 4851 are ham messages.
- **Data Cleaning:** Data cleaning is the process of identifying and correcting or removing inaccurate, incomplete, or irrelevant data in a dataset. It involves several steps such as removing duplicate records, filling missing values, correcting spelling errors, and removing outliers. The main goal of data cleaning is to ensure that the data is accurate, consistent, and ready for analysis. It is an important step in the data preparation process, as inaccurate data can lead to incorrect conclusions and decisions. While preprocessing we remove the html tag and short notes, eliminate stop words, remove characters and numerical.
- **Feature extraction:** Feature extraction is the process of selecting and transforming the most relevant and informative features from a dataset. In machine learning, features are the input variables or attributes that are used to train a model to make predictions. The process of feature extraction involves reducing the dimensionality of the dataset by selecting only the most important features or creating new features from the existing ones. This is done to improve the performance and efficiency of the model by reducing noise, improving accuracy, and reducing overfitting. In this system we are using natural language processing as feature extraction. We are using TF-IDF for vectorization.
- **Model Training:** After the feature extraction the data is used for training the machine

learning model. In this system we are using Decision Tree Classifier.

- **Model Evaluation:** After the model is trained, it needs to be evaluated to determine its performance on unseen data. This may involve splitting the data set into training and validation sets. We can evaluate the performance of the model by using techniques such as accuracy, precision, recall, F1 score and confusion matrix.
- **Saving the Model:** Once the model is evaluated and its performance is satisfactory, it is saved for future use. The saved model is then deployed to the chatting system where it can be used to predict whether messages sent by the user are spam or ham in real-time. This deployment involves integrating the model with the chatting system's backend so that it can process incoming messages and return the appropriate prediction. The predictions can then be used to flag potential spam messages for further review.
- **Prediction:** After the model is evaluated and saving model it can be used for spam message prediction. The new messages send by the user is preprocessed and predicted using this model. The predicted result will be updated into the status field.

3.3 Workflow

A workflow diagram, also known as a flowchart, is a visual representation of the steps involved in completing a task or achieving a specific goal in a software project. It shows the sequence of steps that need to be performed, as well as the decision points and actions that occur at each step. The purpose of a workflow diagram is to provide a clear and concise overview of the software project, helping project teams to identify potential issues and optimize the workflow.

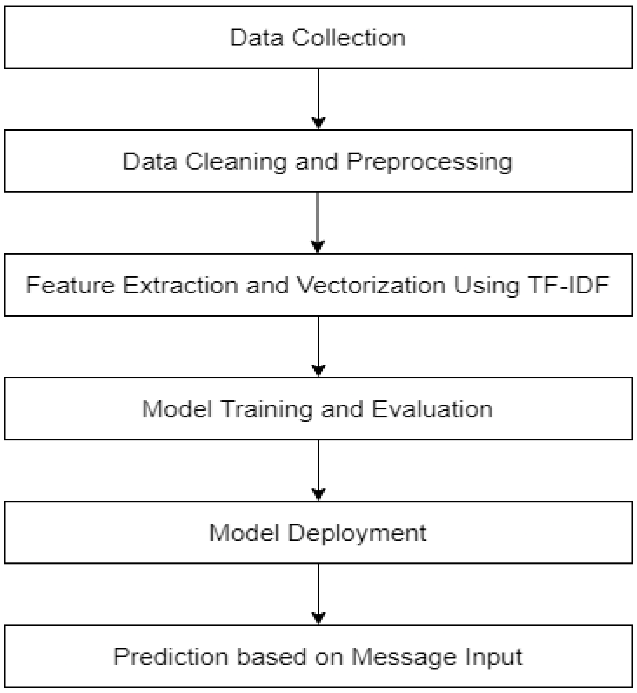

Data Collection
Data Cleaning and Preprocessing
Feature Extraction and Vectorization Using TF-IDF
Model Training and Evaluation
Model Deployment
Prediction based on Message Input

Fig 3.3.1: Workflow Diagram

3.4 UML Diagrams

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts, or classes, in order to better understand, alter, maintain, or document information about the system. UML is an acronym that stands for Unified Modelling Language. Simply put, UML is a modern approach to modelling and documenting software. In fact, it is one of the most popular business process modelling techniques. It is based on diagrammatic representations of software components. As the old proverb says: “a picture is worth a thousand words.” By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

The Primary goals in the design of the UML in this project is:

1. Provide users a ready-to-use, expressive visual modeling Language so that they candevelop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices

1.1.1 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

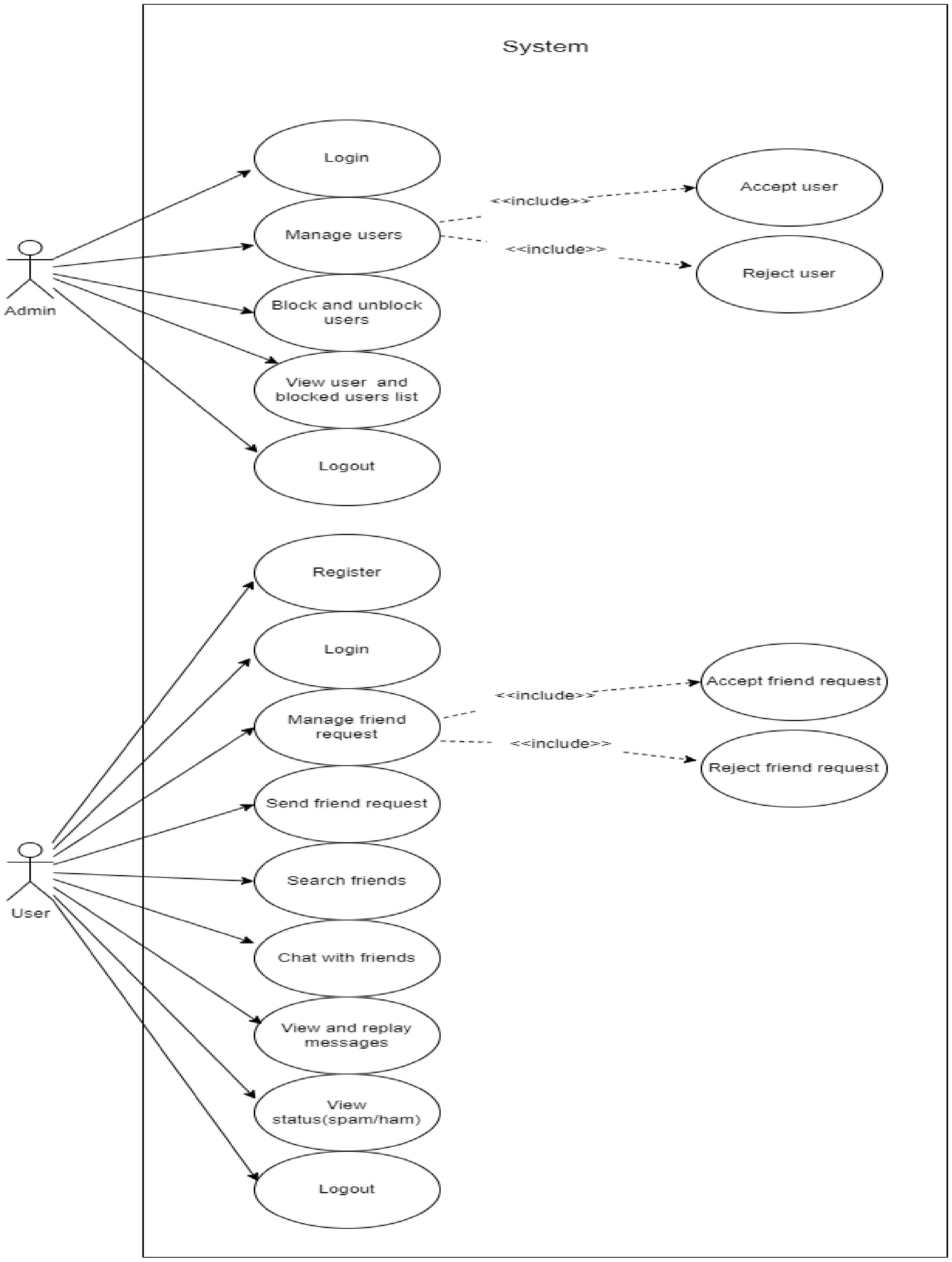


Fig 3.4.1: Use case Diagram for Proposed System

3.4.2 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a

Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

1. Admin

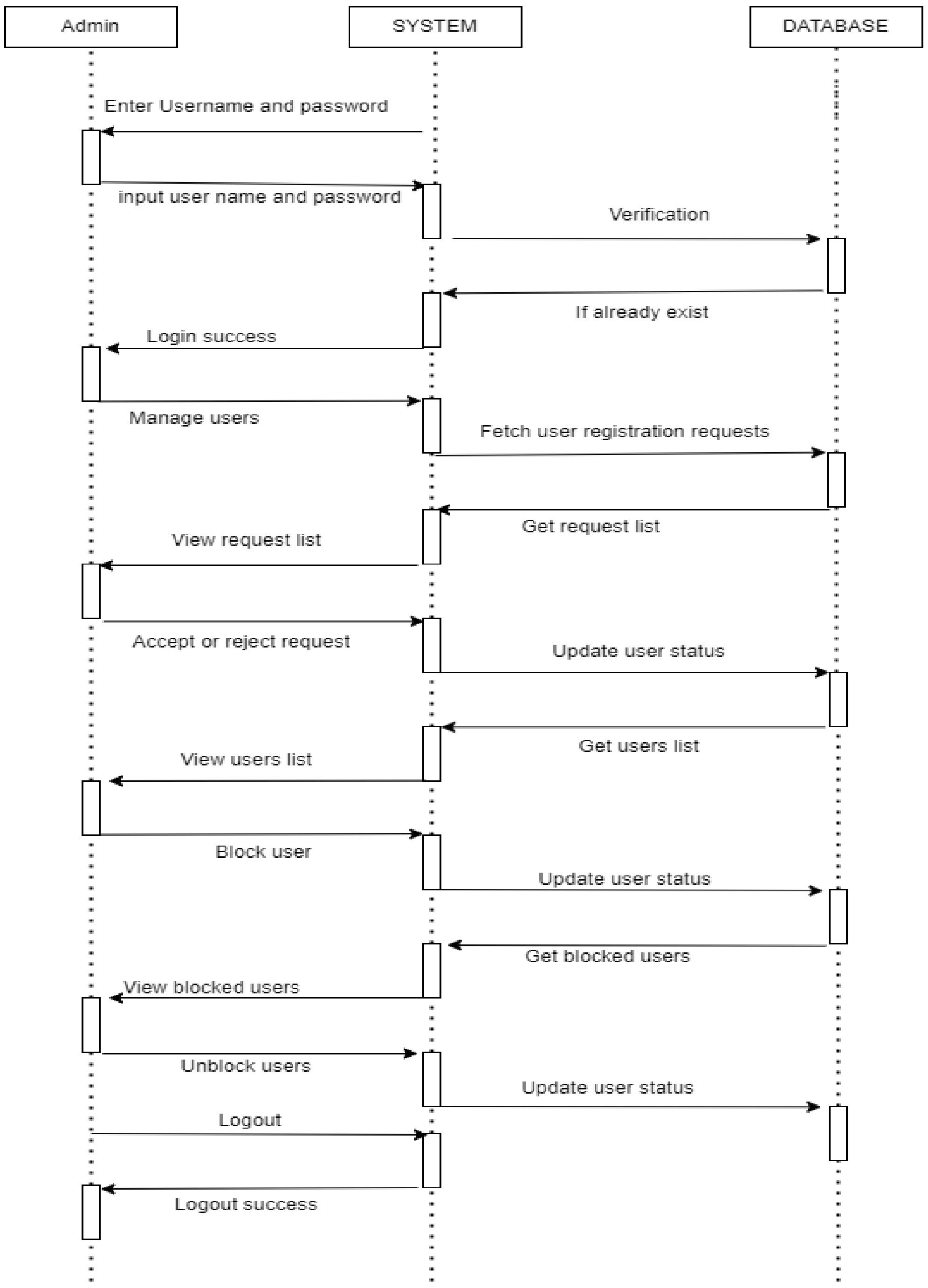


Figure 3.4.2: Admin Sequence Diagram

2. User

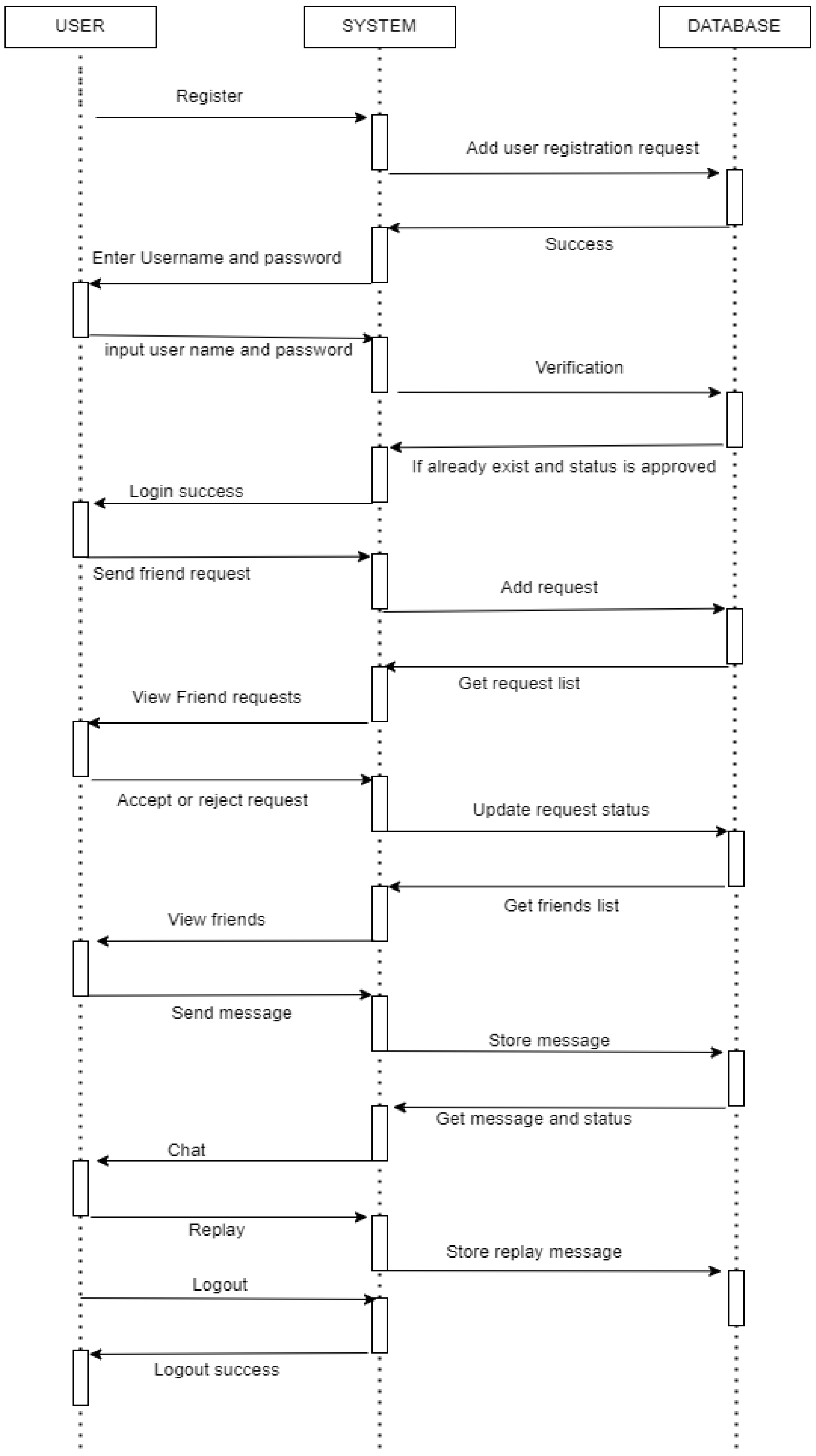


Figure 3.4.3: User Sequence Diagram

3.5 Hardware and Software Requirements

3.5.1 Hardware Requirements

- i3 Processor Based Computer or higher
- Memory: 1 GB
- Hard Drive:50 GB
- Keyboard: Standard 120 keys
- Monitor
- Internet Connection

3.5.1 Software Requirements

- OS: Windows 7 or higher
- Database System: WAMP Server, My SQL 5.6
- Documentation Tool: MS – Word
- Language: Python
- Visual Studio Code

3.5.2 Technologies Used

Python

Python is a high-level, interpreted programming language that emphasizes code readability and simplicity. It was created in the late 1980s by Guido van Rossum and has since become one of the most popular programming languages in the world. Python’s syntax is designed to be easy to read and write, making it a popular choice for beginners and experienced programmers alike. It has a large and active community that contributes to a vast ecosystem of libraries and frameworks, making it versatile and suitable for a wide range of applications such as web development, data analysis, machine learning, and scientific computing.

Readability: Python’s syntax is designed to be easy to read and understand, which makes it easier for developers to write code quickly and accurately.

Simplicity: Python’s syntax is straightforward and easy to learn, making it an ideal language for beginners.

Interpreted: Python is an interpreted language, which means that code is executed line by line, rather than being compiled into machine code before execution.

Dynamic typing: Python is dynamically typed, which means that variable types are determined at runtime rather than being declared explicitly.

Strong typing: Despite being dynamically typed, Python has strong typing, which means that types are enforced during runtime, making it less prone to errors.

Large standard library: Python has a vast standard library that includes modules for a wide range of tasks such as web development, scientific computing, and data analysis.

Object-oriented: Python supports object-oriented programming, making it easier to create reusable and modular code.

Cross-platform: Python runs on a wide range of platforms, including Windows, Mac, and Linux.

Extensible: Python can be extended with modules and packages, making it possible to add new functionality and features to the language.

Open-source: Python is open-source, which means that its source code is freely available to the public, making it accessible to anyone who wants to use or modify it.

Django

Django is a high-level web framework for building web applications quickly and easily. It follows the Model-View-Controller (MVC) architectural pattern, but implements it differently with a Model-View-Template (MVT) pattern. It is written in Python, and provides many prebuilt components and tools for common web development tasks, such as URL routing, form handling, authentication, database ORM, and more. Django also encourages clean, reusable, and modular code, and has a large and active community of developers who contribute to its growth and maintenance. It is widely used by many popular websites, such as Instagram, Pinterest, and Mozilla.

Sklearn

Scikit-learn, commonly known as sklearn, is a popular open-source machine learning library for Python. It provides a wide range of tools for implementing various machine learning algorithms, including classification, regression, clustering, and dimensionality reduction.

Sklearn is built on top of other popular scientific computing libraries in Python, such as NumPy, SciPy, and matplotlib. This makes it easy to integrate into existing Python workflows and leverage the power of these other libraries. Sklearn includes a wide range of algorithms and tools for pre-processing, feature selection, and model selection, as well as tools for evaluating the performance of machine learning models. It also includes a number of datasets for practice and experimentation. One of the advantages of using sklearn is its ease of use and accessibility. Its API is consistent and well-documented, making it easy for beginners to get started with machine learning. Additionally, its extensive documentation and active community make it easy to find help and guidance when needed. Sklearn is a powerful and versatile tool for implementing machine learning algorithms in Python, and is widely used in both academia and industry.

Pandas

A data manipulation library that provides tools for data cleaning, transformation, and analysis, which can be useful for preparing and preprocessing datasets for machine learning tasks.

Matplotlib

Matplotlib is a popular open-source plotting library for Python. It provides a wide range of tools for creating visualizations, including line plots, scatter plots, bar charts, histograms, and many others. Matplotlib is built on top of NumPy and provides a simple and easy-to-use interface for creating high-quality visualizations. Matplotlib provides a range of customization options for visualizations, including customizing colors, markers, labels, axes, and legends. It also supports a range of plot styles and formatting options, making it easy to create professional-looking visualizations. In addition to its basic plotting capabilities, Matplotlib also provides tools for creating more complex visualizations, such as subplots, grids, and 3D plots. It also supports interactive visualizations and can be integrated with other libraries, such as Seaborn and Pandas. Matplotlib is widely used in the scientific and data analysis communities, and has a large and active community of users and developers. It is well-documented and provides many examples and tutorials to help users get stared and learn how to create effective visualizations.

Seaborn

Seaborn is a popular open-source data visualization library for Python, built on top of matplotlib. It provides a high-level interface for creating statistical graphics, including heatmaps, time series visualizations, distribution plots, and many others.

Pickle

In Python, pickle is a built-in module that provides a way to serialize and deserialize Python objects. Serialization is the process of converting a Python object into a byte stream, and deserialization is the process of recreating a Python object from a byte stream. Pickle can be used to store Python objects in a file or send them over a network. It is commonly used in machine learning applications to save trained models to a file and load them back later for predictions. Pickle supports many Python data types, including integers, floats, strings, lists, tuples, dictionaries, and more complex objects such as user-defined classes and functions. However, it cannot serialize certain types of objects, such as file handles, network sockets, and some third-party library objects.

BeautifulSoup

BeautifulSoup is a Python library used for web scraping purposes to extract data from HTML and XML documents. It provides an easy-to-use interface for parsing and navigating through HTML and XML documents, allowing users to extract specific information from websites. BeautifulSoup works by converting an HTML or XML document into a parse tree, which can then be searched and manipulated using Python code. It provides a range of methods for finding specific tags and attributes within the document, as well as for extracting text and other data from those tags.

Bootstrap

Bootstrap is a popular open-source front-end web development framework that provides a collection of pre-built HTML, CSS, and JavaScript components and utilities. It is designed to make it easier and faster to create responsive and mobile-first websites. Bootstrap includes a wide range of components, such as navigation menus, forms, buttons, tables, and more, which can be easily customized and combined to create a custom design. It also includes a powerful grid system that allows developers to create responsive layouts that adapt to different screen sizes and devices. In addition to its core components and utilities, Bootstrap also provides a range of plugins and extensions that add additional functionality, such as carousels, modals, and tooltips. Bootstrap is widely used in web development and is supported by a large and active community of developers. It is well-documented and provides a range of examples and templates to help developers get started and learn how to use its features effectively.

Wamp Server

WampServer is a popular open-source web development environment for Windows that allows developers to create and test dynamic web applications locally. It includes Apache web server, PHP scripting language, and MySQL database, and can be easily installed and configured on a Windows machine. WampServer provides an easy-to-use graphical user interface for managing server settings, such as PHP extensions, MySQL databases, and virtual hosts. It also includes tools for debugging and testing PHP applications, such as Xdebug and phpMyAdmin. Overall, WampServer provides a simple and convenient way for Windows users to set up a local web development environment.

MySQL

MySQL is a popular open-source relational database management system (RDBMS) that is widely used for web applications and other data-driven applications. It is written in C and C++, and provides a robust and scalable database engine that can handle large amounts of data and high traffic volumes. MySQL supports a wide range of features, including support for multiple storage engines, such as InnoDB, MyISAM, and Memory, transactions, triggers, views, and more. It also supports a variety of programming languages, including PHP, Python, Java, and more, and can be easily integrated with web applications through its native drivers and connectors. MySQL is free and open-source software and is widely used by many popular websites, including Facebook, Twitter, and Wikipedia.

Visual Studio Code

Visual Studio Code (VS Code) is a free and open-source integrated development environment (IDE) developed by Microsoft. It is available for Windows, macOS, and Linux, and provides support for a wide range of programming languages, including Python, JavaScript, TypeScript, C++, and more. VS Code is designed to be lightweight and customizable, and provides a range of features to help developers write and debug code more efficiently. It includes a powerful code editor with support for syntax highlighting, code completion, and intelligent code analysis. It also provides a range of debugging tools, including breakpoints, variable inspection, and step-by-step execution. In addition to its core features, VS Code provides a rich ecosystem of extensions that can be used to add additional functionality, such as support for specific programming languages, code snippets, and more. It also provides integrated support for Git version control and a range of other tools and services, such as GitHub and Azure. One of the key benefits of VS Code is its flexibility and customizability. It can be customized with a range of themes, extensions, and settings to suit the needs of individual developers and projects. This makes it a popular choice for both beginners and experienced developers alike.

Chapter 4

Agile Methodology

4.1 Introduction

Agile methodology is a set of values, principles, and practices for software development that emphasizes flexibility, collaboration, and continuous improvement. It was developed in response to the limitations of traditional software development methodologies, which often resulted in delayed delivery, budget overruns, and unsatisfied customers. Agile methodology is based on the Agile Manifesto, which values individuals and interactions, working software, customer collaboration, and responding to change over processes and tools, comprehensive documentation, contract negotiation, and following a plan. Scrum is a process framework that has been used to manage complex product development. It is not a process or technique for building products rather it is a framework within which various processes can be employed.

Agile methodology emphasizes short iterations or sprints, typically lasting two to four weeks, during which a small portion of the software system is developed, tested, and delivered. The team meets regularly to discuss progress, identify issues, and plan the next iteration. It also emphasizes close collaboration between the development team and the customer or product owner. The customer or product owner provides feedback on each iteration, allowing the development team to quickly respond to changing requirements or priorities.

Key practices in agile methodology include:

- **Continuous integration:** The practice of integrating new code changes into the main codebase as soon as they are ready.
- **Test-driven development:** The practice of writing tests before writing code, ensuring that the code meets the specified requirements.
- **Pair programming:** The practice of having two programmers work together on the same codebase, allowing for better collaboration, knowledge sharing, and error detection.

- **Agile planning:** The practice of planning the project in short iterations, with the focus on delivering working software that meets the customer’s needs.
- **Retrospectives:** The practice of holding regular team meetings to reflect on what worked well and what needs to be improved.

Major roles in scrum methodology includes:

- **Product Owner:** The Product Owner is responsible for maximizing the value of the product by managing the Product Backlog, which is a prioritized list of features or requirements. The Product Owner ensures that the Product Backlog is up-to-date, well-defined, and represents the customer’s needs.
- **Scrum Master:** The Scrum Master is responsible for ensuring that the Scrum process is understood, implemented, and followed by the Scrum team. They facilitate the Scrum ceremonies such as Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective, and remove any impediments that are hindering the team’s progress. The Scrum Master also acts as a coach and mentor to the team, helping them to continuously improve their processes and practices.
- **Development Team:** The Development Team is responsible for delivering the product increment at the end of each Sprint.

Major Artifacts in scrum methodology includes:

- **Product Backlog:** The Product Backlog is a prioritized list of user stories or product requirements. The Product Owner is responsible for maintaining the Product Backlog and ensuring that it reflects the customer’s priorities.
- **Sprint Backlog:** The Sprint Backlog is a list of the tasks that the Development Team plans to complete during the current Sprint. The Sprint Backlog is created during the Sprint Planning meeting and is updated throughout the Sprint as progress is made. The Development Team is responsible for managing the Sprint Backlog and ensuring that the Sprint goal is met.
- **Product Increment:** The Increment is the sum of all the completed Product Backlog items at the end of each Sprint. The Increment is a working version of the product that is potentially releasable and adds value to the customer.

Major Events in scrum methodology includes:

- **Sprint:** A Sprint is a time-boxed iteration of the software development process. Typically, a Sprint lasts 2-4 weeks, and at the end of each Sprint, the team delivers a potentially shippable product increment.
- **Sprint Planning:** At the beginning of each Sprint, the Scrum Team holds a Sprint Planning meeting to determine the Sprint Goal and select the Product Backlog items that will be worked on during the Sprint.
- **Daily Scrum:** The Daily Scrum is a 15-minute meeting that is held every day during the Sprint.
- **Sprint Review:** At the end of each Sprint, the Scrum Team holds a Sprint Review meeting to demonstrate the completed work to stakeholders and receive feedback. The Sprint Review provides an opportunity for the Scrum Team to reflect on their progress and identify areas for improvement.
- **Sprint Retrospective:** The Sprint Retrospective is a meeting that is held at the end of each Sprint to reflect on the Sprint and identify areas for improvement in the next Sprint. The Scrum Team uses this meeting to discuss what went well, what could be improved, and what actions they will take in the next Sprint to improve their process.

The three pillars of scrum are transparency, inspection and adaptation. In scrum everyone has a role.

4.2 User Story

A user story is a simple, one-sentence description of a feature or requirement used to capture the user’s needs and help the team understand what they should be building. User stories are a lightweight and flexible way of communicating requirements that can be easily understood and prioritized by the development team. The user story describes the type of user, what they want and why.

User Story ID	**As a <Type of user>**	**I want to perform <some task>**	**So that I can <achieve some goal>**
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 and unblock them
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 4.2: User Story

4.3 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 product backlog is the single authoritative source for things that a team works on. That means that nothing gets done that isn't on the product backlog. Conversely, the presence of a product backlog item on a product backlog does not guarantee that it will be delivered. It represents an option the team has for delivering a specific outcome rather than a commitment.

It should be cheap and fast to add a product backlog item to the product backlog, and it should be equally as easy to remove a product backlog item that does not result in direct progress to achieving the desired outcome or enable progress toward the outcome. The Scrum Product Backlog is simply a list of all things that needs to be done within the project. It replaces the traditional requirements specification artifacts. These items can have a technical nature or can be user-centric e.g., in the form of user stories. The product backlog of the system is given in Table 4.2.

PRODUCT BACKLOG			
ID	**Name**	**Priority**	**Estimate [Hrs]**
1	Admin Login	1	8
2	User Registration	2	12
3	User Login	3	8
4	User Search friends	4	12
5	User Send friend request	5	16
6	User Manage friend request	6	19
7	User View friend list	7	12
8	User Chat with friends	8	20
9	User View messages and replay message	9	20
10	User View status (Spam / Ham)	10	8
11	Admin View user list	11	12
12	Admin Manage user	12	14
13	Admin Block user	13	11
14	Admin View blocked user	14	9

Table 4.3: Product Backlog

4.4 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. Project plan is given in Table 4.3

User story ID	Task Name	Start Date	End Date	Days	Status
Sprint 1		**25/01/2023**	**02/02/2023**	**7**	**Completed**
1	Design Admin login	25/01/2023	27/01/2023	2	Completed

2	Design User Registration and login	28/01/2023	31/01/2023	3	Completed
3	Connection to database	01/02/2023	02/02/2023	2	Completed
Sprint 2		**03/02/2023**	**17/02/2023**	**12**	**Completed**
4	Develop features to manage user friend requests and send friend requests	03/02/2023	06/02/2023	3	Completed
5	Develop and execute a search and viewing feature for user's friends	07/02/2023	10/02/2023	4	Completed
6	Connection to database	13/02/2023	17/02/2023	5	Completed
Sprint 3		**20/02/2023**	**10/03/2023**	**15**	**Completed**
7	Design and implement chatting facility	20/02/2023	22/02/2023	3	Completed
8	Design and implement feature to view and replay messages	23/02/2023	01/03/2023	5	Completed
9	Database connection and testing	02/03/2023	10/03/2023	7	Completed
Sprint 4		**13/03/2023**	**24/03/2023**	**10**	**Completed**
10	Creating features for admin to view users and manage users	13/03/2023	17/03/2023	5	Completed
11	Creating features for admin to block and unblock user	20/03/2023	21/03/2023	2	Completed
12	Database connection and testing	22/03/2023	24/03/2023	3	Completed
Sprint 5		**27/03/2023**	**17/04/2023**	**14**	**Completed**
13	Model creation and evaluation for spam detection	27/03/2023	30/03/2023	4	Completed
14	Model implementation	31/03/2023	03/04/2023	3	Completed
15	Create feature for user to view message status(spam/ham)	04/04/2023	10/04/2023	3	Completed
16	Testing & Output generation	11/04/2023	17/04/2023	4	Completed

Table 4.4: Project plan

The Project has five sprints:

1. **Sprint 1**

Three tasks are planned at this stage. They are Design Admin login page, Design User Registration and login page and connect to the database.

2. **Sprint 2**

   Three tasks are planned at this stage. They are Develop features to manage user friend requests and send friend requests, Develop, and execute a search and viewing feature for user's friends and Connection to database.

3. **Sprint 3**

   Three tasks are planned at this stage. They are Design and implement chatting facility, Design and implement feature to view and replay messages and Database connection and testing.

4. **Sprint 4**

   Three tasks are planned at this stage. They are Creating features for admin to view users and manage users, creating features for admin to block and unblock user, Database connection and testing.

5. **Sprint 5**

   Four tasks are planned at this stage. They are Model creation and evaluation for spam detection, Model implementation, Create feature for user to view message status(spam/ham) and Testing & Output generation.

4.5 Sprint Backlog (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. Most teams also estimate how many hours each task will take someone on the team to complete.

2. **Sprint 1**

   Three tasks are planned at this stage. They are Design Admin login page, Design User Registration and login page and connect to the database. Sprint backlog (planning) for sprint 1 is given in Table 4.4.

3. **Sprint 2**

   Three tasks are planned at this stage. They are Develop features to manage user friend requests and send friend requests, Develop, and execute a search and viewing feature for user's friends and Connection to database. Sprint backlog (planning) for sprint 2 is given in Table 4.5.

4. **Sprint 3**

   Three tasks are planned at this stage. They are Design and implement chatting facility, Design and implement feature to view and replay messages and Database connection and testing. Sprint backlog (planning) for sprint 3 is given in Table 4.6.

5. **Sprint 4**

   Three tasks are planned at this stage. They are Creating features for admin to view users and manage users, creating features for admin to block and unblock user, Database connection and testing. Sprint backlog (planning) for sprint 4 is given in Table 4.6.

6. **Sprint 5**

   Four tasks are planned at this stage. They are Model creation and evaluation for spam detection, Model implementation, Create feature for user to view message status(spam/ham) and Testing & Output generation. Sprint backlog (planning) for sprint 5 is given in Table 4.7.

Backlogitems	User story#1 Hours	Design Admin login	Design User Registration and login	Connection to database
Completion date		Hours	Hours	Hours
Original estimate in hrs		8	12	8
Day 1 25/01/2023	Hours	4	0	0
Day 2 27/01/2023	Hours	4	0	0
Day 3 28/01/2023	Hours	0	4	0
Day 4 30/01/2023	Hours	0	4	0
Day 5 31/02/2023	Hours	0	4	0
Day 6 01/02/2023	Hours	0	0	4
Day 7 02/02/2023	Hours	0	0	4

Table 4.5.1: Sprint Backlog (Plan) – Sprint 1

Backlogitems	User story#1 Hours	Develop features to manage user friend requests and send friend requests	Develop and execute a search and viewing feature for user's friends	Connection to database
Completion date		Hours	Hours	Hours
Original estimate in hrs		12	16	19
Day 1 03/02/2023	Hours	4	0	0
Day 2 04/02/2023	Hours	4	0	0
Day 3 06/02/2023	Hours	4	0	0
Day 4 07/02/2023	Hours	0	4	0
Day 5 08/02/2023	Hours	0	4	0
Day 6 09/02/2023	Hours	0	4	0
Day 7 10/04/2023	Hours	0	4	0
Day 8 13/02/2023	Hours	0	0	4
Day 9 14/02/2023	Hours	0	0	4
Day 10 15/02/2023	Hours	0	0	4
Day 11 16/02/2023	Hours	0	0	4
Day 12 17/02/2023	Hours	0	0	3

Table 4.5.2: Sprint Backlog (Plan) – Sprint 2

Backlogitems	User story#1 Hours	Design and implement chatting facility	Design and implement feature to view and replay messages	Database connection and testing

Completion date		Hours	Hours	Hours
Original estimate in hrs		12	20	25
Day 1 20/02/2023	Hours	4	0	0
Day 2 21/02/2023	Hours	4	0	0
Day 3 22/02/2023	Hours	4	0	0
Day 4 23/02/2023	Hours	0	4	0
Day 5 24/02/2023	Hours	0	4	0
Day 6 27/02/2023	Hours	0	4	0
Day 7 28/02/2023	Hours	0	4	0
Day 8 01/03/2023	Hours	0	4	0
Day 9 02/03/2023	Hours	0	0	4
Day 10 03/03/2023	Hours	0	0	4
Day 11 06/03/2023	Hours	0	0	4
Day 12 07/03/2023	Hours	0	0	4
Day 13 08/03/2023	Hours	0	0	4
Day 14 09/03/2023	Hours	0	0	4
Day 15 10/03/2023	Hours	0	0	1

Table 4.5.3: Sprint Backlog (Plan) – Sprint 3

Backlogitems	User story#1 Hours	Creating features for admin to view users and manage users	Creating features for admin to block and unblock user	Database connection and testing
Completion date		Hours	Hours	Hours
Original estimate in hrs		20	8	12
Day 1 13/03/2023	Hours	4	0	0
Day 2 14/03/2023	Hours	4	0	0
Day 3 15/03/2023	Hours	4	0	0
Day 4 16/03/2023	Hours	4	0	0
Day 5 17/03/2023	Hours	4	0	0
Day 6 20/03/2023	Hours	0	4	0
Day 7 21/03/2023	Hours	0	4	0
Day 8 22/03/2023	Hours	0	0	4
Day 9 23/03/2023	Hours	0	0	4
Day 10 24/03/2023	Hours	0	0	4

Table 4.5.4: Sprint Backlog (Plan) – Sprint 4

Backlog items	User story #1 Hours	Model creation and evaluation for spam detection	Model implementation	Create feature for user to view message status(spam/ham)	Testing & Output generation
Completion date		Hours	Hours	Hours	Hours
Original estimate in hrs		14	11	9	14

Day 1 27/03/2023	Hours	4	0	0	0
Day 2 28/03/2023	Hours	4	0	0	0
Day 3 29/03/2023	Hours	4	0	0	0
Day 4 30/03/2023	Hours	2	0	0	0
Day 5 31/03/2023	Hours	0	4	0	0
Day 6 01/04/2023	Hours	0	4	0	0
Day 7 03/04/2023	Hours	0	3	0	0
Day 8 04/04/2023	Hours	0	0	4	0
Day 9 05/04/2023	Hours	0	0	1	0
Day 10 10/04/2023	Hours	0	0	4	0
Day 11 11/04/2023	Hours	0	0	0	4
Day 12 12/04/2023	Hours	0	0	0	4
Day 13 13/04/2023	Hours	0	0	0	2
Day 14 17/04/2023	Hours	0	0	0	4

Table 4.5.5: Sprint Backlog (Plan) – Sprint 5

4.6 Sprint Backlog (Actual)

Actual sprint backlog is what adequate sprint planning is actually done by project team there may or may not be difference in planned sprint backlog. The detailed sprint backlog (Actual) is given below.

Backlogitems	User story#1 Hours	Design Admin login	Design User Registration and login	Connection to database
Completion date		Hours	Hours	Hours
Original estimate in hrs		7	12	7
Day 1 25/01/2023	Hours	3	0	0
Day 2 27/01/2023	Hours	4	0	0
Day 3 28/01/2023	Hours	0	4	0
Day 4 30/01/2023	Hours	0	4	0
Day 5 31/02/2023	Hours	0	4	0
Day 6 01/02/2023	Hours	0	0	3
Day 7 02/02/2023	Hours	0	0	4

Table 4.6.1: Sprint Backlog (Actual) – Sprint 1

Backlogitems	User story#1 Hours	Develop features to manage user friend requests and send friend requests	Develop and execute a search and viewing feature for user's friends	Connection to database
Completion date		Hours	Hours	Hours
Original estimate in hrs		12	15	15
Day 1 03/02/2023	Hours	4	0	0
Day 2 04/02/2023	Hours	4	0	0
Day 3 06/02/2023	Hours	4	0	0
Day 4 07/02/2023	Hours	0	3	0

Day 5 08/02/2023	Hours	0	4	0
Day 6 09/02/2023	Hours	0	4	0
Day 7 10/04/2023	Hours	0	4	0
Day 8 13/02/2023	Hours	0	0	4
Day 9 14/02/2023	Hours	0	0	3
Day 10 15/02/2023	Hours	0	0	3
Day 11 16/02/2023	Hours	0	0	2
Day 12 17/02/2023	Hours	0	0	3

Table 4.6.2: Sprint Backlog (Actual) – Sprint 2

Backlogitems	User story#1 Hours	Design and implement chatting facility	Design and implement feature to view and replay messages	Database connection and testing
Completion date		Hours	Hours	Hours
Original estimate in hrs		12	16	21
Day 1 20/02/2023	Hours	4	0	0
Day 2 21/02/2023	Hours	4	0	0
Day 3 22/02/2023	Hours	4	0	0
Day 4 23/02/2023	Hours	0	2	0
Day 5 24/02/2023	Hours	0	3	0
Day 6 27/02/2023	Hours	0	4	0

Day 7 28/02/2023	Hours	0	3	0
Day 8 01/03/2023	Hours	0	4	0
Day 9 02/03/2023	Hours	0	0	3
Day 10 03/03/2023	Hours	0	0	4
Day 11 06/03/2023	Hours	0	0	3
Day 12 07/03/2023	Hours	0	0	4
Day 13 08/03/2023	Hours	0	0	2
Day 14 09/03/2023	Hours	0	0	3
Day 15 10/03/2023	Hours	0	0	2

Table 4.6.3: Sprint Backlog (Actual) – Sprint 3

Backlogitems	User story#1 Hours	Creating features for admin to view users and manage users	Creating features for admin to block and unblock user	Database connection and testing
Completion date		Hours	Hours	Hours
Original estimate in hrs		18	8	12
Day 1 13/03/2023	Hours	4	0	0
Day 2 14/03/2023	Hours	4	0	0
Day 3 15/03/2023	Hours	4	0	0
Day 4 16/03/2023	Hours	2	0	0
Day 5 17/03/2023	Hours	4	0	0
Day 6 20/03/2023	Hours	0	4	0

Day 7 21/03/2023	Hours	0	4	0
Day 8 22/03/2023	Hours	0	0	4
Day 9 23/03/2023	Hours	0	0	4
Day 10 24/03/2023	Hours	0	0	4

Table 4.6.4: Sprint Backlog (Actual) – Sprint 4

Backlog items	User story #1 Hours	Model creation and evaluation for spam detection	Model implementati on	Create feature for user to view message status(spam/ham)	Testing & Output generatio n
Completion date		Hours	Hours	Hours	Hours
Original estimate in hrs		16	11	9	14
Day 1 27/03/2023	Hours	4	0	0	0
Day 2 28/03/2023	Hours	4	0	0	0
Day 3 29/03/2023	Hours	4	0	0	0
Day 4 30/03/2023	Hours	4	0	0	0
Day 5 31/03/2023	Hours	0	4	0	0
Day 6 01/04/2023	Hours	0	3	0	0
Day 7 03/04/2023	Hours	0	3	0	0
Day 8 04/04/2023	Hours	0	0	4	0
Day 9 05/04/2023	Hours	0	0	3	0
Day 10 10/04/2023	Hours	0	0	2	0

Day 11 11/04/2023	Hours	0	0	0	4
Day 12 12/04/2023	Hours	0	0	0	2
Day 13 13/04/2023	Hours	0	0	0	4
Day 14 17/04/2023	Hours	0	0	0	4

Table 4.6.5: Sprint Backlog (Actual) – Sprint 5

4.7 Product Backlog Review

REVIEW FORM

SPRINT 1

Version: 1.0 **Date:02/02/2023**

User story ID	Comments from scrum master if any	Comments from product owner if any
1	Admin should have an effective login page	User friendly login
2	User should have an easy registration process	User friendly registration
3	Should check database connectivity	Check result

Table 4.7.1: Product backlog review (sprint 1)

SPRINT 2

Version: 1.0 **Date: 17/02/2023**

User story ID	Comments from scrum master if any	Comments from product owner if any
4	Should maintain a page to control the friend request	Must list the friend requests in a good order
5	Should have a good interface to view friend list registration process	Easy to search friends
6	Should check database connectivity	Check result

Table 4.7.2: Product backlog review (sprint 2)

SPRINT 3

Version: 1.0 **Date: 10/03/2023**

User story ID	Comments from scrum master if any	Comments from product owner if any
7	Should have a good chatting facility for communication	Good inter face for communication
8	-	Feature for replay and view messages
9	Should check database connectivity and test the current version	Check result

Table 4.7.3: Product backlog review (sprint 3)

SPRINT 4

Version: 1.0 **Date: 24/03/2023**

User story ID	Comments from scrum master if any	Comments from product owner if any
10	Admin module should have authority to manage users	-
11	-	Admin must have the facility to block the users
12	Should check database connectivity and test the current version	Check result

Table 4.7.4: Product backlog review (sprint 4)

SPRINT 5

Version: 1.0 **Date: 17/04/2023**

User story ID	Comments from scrum master if any	Comments from product owner if any
13	Model creation and evaluation	Evaluate and visualize the test results
14	Implement the model	-
15	-	A good facility to warn user about spam messages
16	Generate final output	Satisfied

Table 4.7.5: Product backlog review (sprint 5)

4.8 Sprint Review

At the end of each sprint a Sprint Review meeting is held. During this meeting the Scrum Team shows which Scrum Product Backlog items they completed (according to the Definition of Done) during the sprint. This might take place in the form of a demo of the new features. Backlog items that are not completed shall not be demonstrated. Otherwise, this might suggest that these items are finished as well. Instead, incomplete items/remaining activities shall be taken back into the Scrum Product Backlog, re-estimated and completed in one of the following sprints. The Sprint Review meeting should be kept very informal. No PowerPoint slides should be used and time for preparation and performing the meeting should be limited. During the meeting the Scrum Product Owner inspects the implemented backlog entries and accepts the solution or adds new stories to the Scrum Product Backlog to adapt the functionality. Participants in the sprint review typically include the Scrum Product Owner, the Scrum Team and the Scrum Maste.Additionally, management, customers, and developers from other projects might participate as well.

REVIEW FORM

SPRINT 1

Version: 1.0 **Date:02/02/2023**

User story ID	Comments from scrum master if any	Comments from product owner if any
1	Admin should have an effective login page	User friendly login
2	User should have an easy registration process	User friendly registration
3	Should check database connectivity	Check result

Table 4.8.1: Sprint review (sprint 1)

SPRINT 2

Version: 1.0 **Date: 17/02/2023**

User story ID	Comments from scrum master if any	Comments from product owner if any
4	Should maintain a page to control the friend request	Must list the friend requests in a good order
5	Should have a good interface to view friend list registration process	Easy to search friends

6	Should check database connectivity	Check result

Table 4.8.2: Sprint review (sprint 2)

SPRINT 3

Version: 1.0 **Date: 10/03/2023**

User story ID	Comments from scrum master if any	Comments from product owner if any
7	Should have a good chatting facility for communication	Good inter face for communication
8	-	Feature for replay and view messages
9	Should check database connectivity and test the current version	Check result

Table 4.8.3: Sprint review (sprint 3)

SPRINT 4

Version: 1.0 **Date: 24/03/2023**

User story ID	Comments from scrum master if any	Comments from product owner if any
10	Admin module should have authority to manage users	-
11	-	Admin must have the facility to block the users
12	Should check database connectivity and test the current version	Check result

Table 4.8.4: Sprint review (sprint 4)

SPRINT 5

Version: 1.0 **Date: 17/04/2023**

User story ID	Comments from scrum master if any	Comments from product owner if any
13	Model creation and evaluation	Evaluate and visualize the test results
14	Implement the model	-
15	-	A good facility to warn user about spam messages

16	Generate final output	Satisfied

Table 4.8.5: Sprint review (sprint 5)

4.9 Testing and Validation

Sprint 1:

Test #	Date	Action	Expected result	Actual Result	Pass? (Yes/No)
1	27/01/2023	Design Admin login	Complete Admin Login design	Complete Admin Login design	Yes
2	31/01/2023	Design User Registration and login	Complete User Registration and login design	Complete User Registration and login design	Yes
3	02/02/2023	Connection to database	Successful Connection to database	Successful Connection to database	Yes

Table 4.9.1: Testing and Validation – Sprint 1

Sprint 2:

Test #	Date	Action	Expected result	Actual Result	Pass? (Yes/No)
1	06/02/2023	Develop features to manage user friend requests and send friend requests	Complete Developing features to manage user friend requests and send friend requests	Complete Developing features to manage user friend requests and send friend requests	Yes
2	10/02/2023	Develop and execute a search and viewing feature for user’s friends	Complete Developing and execute a search and viewing feature for user’s friends	Complete Developing and execute a search and viewing feature for user’s friends	Yes
3	17/02/2023	Connection to database	Successful Connection to database	Successful Connection to database	Yes

Table 4.9.2: Testing and Validation – Sprint 2

Sprint 3:

Test #	Date	Action	Expected result	Actual Result	Pass? (Yes/No)
1	18/05/2022	Design and implement chatting facility	Complete Designing and implementation chatting facility	Complete Designing and implementation chatting facility	Yes
2	25/05/2022	Design and implement feature to view and replay messages	Complete Designing and implementation feature to view and replay messages	Complete Designing and implementation feature to view and replay messages	Yes
3	30/05/2022	Database connection and testing	Success full connection to Database and testing	Success full connection to Database and testing	Yes

Table 4.9.3: Testing and Validation – Sprint 3

Sprint 4:

Test #	Date	Action	Expected result	Actual Result	Pass? (Yes/No)
1	24/03/2023	Creating features for admin to view users and manage users	Complete Creating features for admin to view users and manage users	Complete Creating features for admin to view users and manage users	Yes
2	17/03/2023	Creating features for admin to block and unblock user	Complete Creating features for admin to block and unblock user	Complete Creating features for admin to block and unblock user	Yes
3	21/03/2023	Database connection and testing	Success full connection to Database and testing	Success full connection to Database and testing	Yes

Table 4.9.4: Testing and Validation – Sprint 4

Sprint 5:

Test #	Date	Action	Expected result	Actual Result	Pass? (Yes/No)
1	17/04/2023	Model creation and evaluation for spam detection	Complete Model creation and evaluation for spam detection	Complete Model creation and evaluation for spam detection	Yes

2	30/03/2023	Model implementation	Complete Model implementation	Complete Model implementation	Yes
3	03/04/2023	Create feature for user to view message status(spam/ham)	Complete Creating feature for user to view message status(spam/ham)	Complete Creating feature for user to view message status(spam/ham)	Yes
4	10/04/2023	Testing & Output generation	Complete Testing & Output generation	Complete Testing & Output generation	Yes

Table 4.9.5: Testing and Validation – Sprint 5

4.10 Git

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. The Git is used as the version control system for this project. Version control is a system that records changes to a file or set of files over time so that a specific version can be recalled later. Version control systems are a category of software tools that help a software team for managing changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.. To show the continuous development of the project the Git lab histories are shown in Appendix

Chapter 5

Results and Discussions

SYSTEM DESIGN

5.1 DATABASE

Table name: login

Primary Key: login id

FIELD NAME	TYPE	SIZE	DESCRIPTION
login id	Integer	10	Login id
username	varchar	50	Username
password	Varchar	50	Password
type	varchar	30	User type (Admin/User)
user id	Integer	10	User id

Table 5.1.1: login

Table name: user

Primary Key: user id

FIELD NAME	TYPE	SIZE	DESCRIPTION
user id	Integer	10	User id
username	Varchar	50	Username
dob	Varchar	20	Date of Birth
gender	Varchar	20	Gender(male/female)

phno	Varchar	20	Phone number
email	Varchar	50	Email
password	Varchar	20	Password
profile photo	Varchar	100	Profile photo
status	Varchar	50	User status(Blocked/accepted)

Table 5.1.2: User

Table name: friends

Primary Key: friend id

FIELD NAME	TYPE	SIZE	DESCRIPTION
friend id	Integer	10	Friend id
user id	Integer	10	User id
f user id	Integer	10	f user id
request status	varchar	50	Friend request status

Table 5.1.3: Friends

Table name: chat

Primary Key: chat id

FIELD NAME	TYPE	SIZE	DESCRIPTION
chat id	Integer	10	Chat id
friend id	Integer	10	Friend id
user id	Integer	10	User id
chat	Varchar	160	Message
photo	Varchar	100	Photo
date	Varchar	20	Date

time	Varchar	20	Time
status	Varchar	50	Message Status(spam/ham)

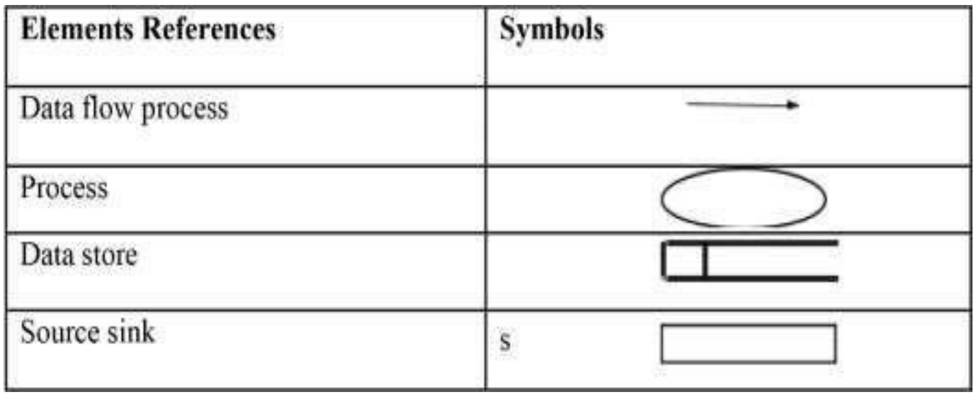
Table 5.1.4: Chat

5.2 DATA FLOW DIAGRAM

The database may be defined as an organized collection of related information. The organized information serves as a base from which further recognizing can be retrieved desired information or processing the data. The most important aspect of building an application system is the design of tables. The data flow diagram is used for classifying system requirements to major transformation that will become programs in system design. This is thestarting point of the design phase that functionally decomposes the required specifications down to the lower level of details. It consists of a series of bubbles joined together by lines.

Dataflow analysis studies the use of data in each activity. It documents this finding in DFD‟s. Dataflow analysis give the activities of a system from the viewpoint of data where it originates and how they are used or hanged or where they go, including the stops along the way from their destination. The components of dataflow strategy span both requirements determination and system’s design. The first part is called dataflow analysis. As the name suggests, we didn’t use the dataflow analysis tools exclusively for the analysis stage but also in the designing phase with documentation. Notations used in Dataflow Diagrams

The logic data flow diagrams can be drawn using only four simple notations i.e., special symbols or icons and the annotation that associates them with a specific system. Since the choice of notation we follow, does not affect impede or catalyse the system process; we used three symbols from YOURDON notation and one from Gain and Sarson notation as specified below.



Elements References	Symbols
Data flow process	→
Process	⬭
Data store	⊏
Source sink	s ▭

Process: describes how input data is converted to output Data Data Store: Describes the repositories of data in a system

Data Flow: Describes the data flowing between process, Data stores and external entities. Sources: An external entity causing the origin of data.

Sink: An external entity, which consumes the data

Several rules of thumb are used in drawing DFDs:-

- Process should be named and numbered for easy reference.
- The direction of flow is from source to destination, although they may flow back to a source. One way to indicate this is to draw a long flow line back to the source. An alternative way is to repeat the source symbol as a destination.
- When a process is exploded into lower-level details, they are numbered.
- The names of data stores, sources, and destinations are written in capital letters. Process and data flow names have the first letter of each word capitalized.

A level 0 DFD, also called a context level, represents the entire software elements as a single bible with input and output indicated by incoming and outgoing arrows respectively. Additional process and information flow parts are represented in the next level i.e. Level 1 DFD. Any process, which is complex in Level 1, will be further represented into sub functions in the next level .i.e

The DFD is designed to aid communication. DFD shows the minimum contents of data stores. In order to show what happens within a given process, then the detailed explosion of that process is shown. The DFD methodology is quite effective, especially when the required design is unclear and the user and the analyst need a notational language for communication.

LEVEL 0

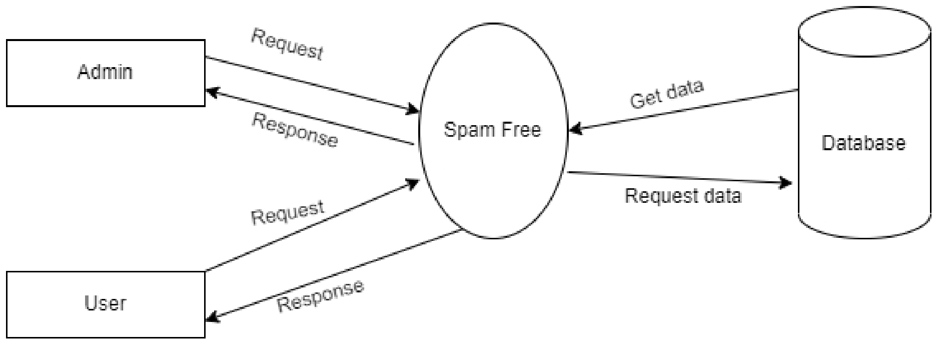


Fig 5.2.1: DFD level 0

LEVEL 1

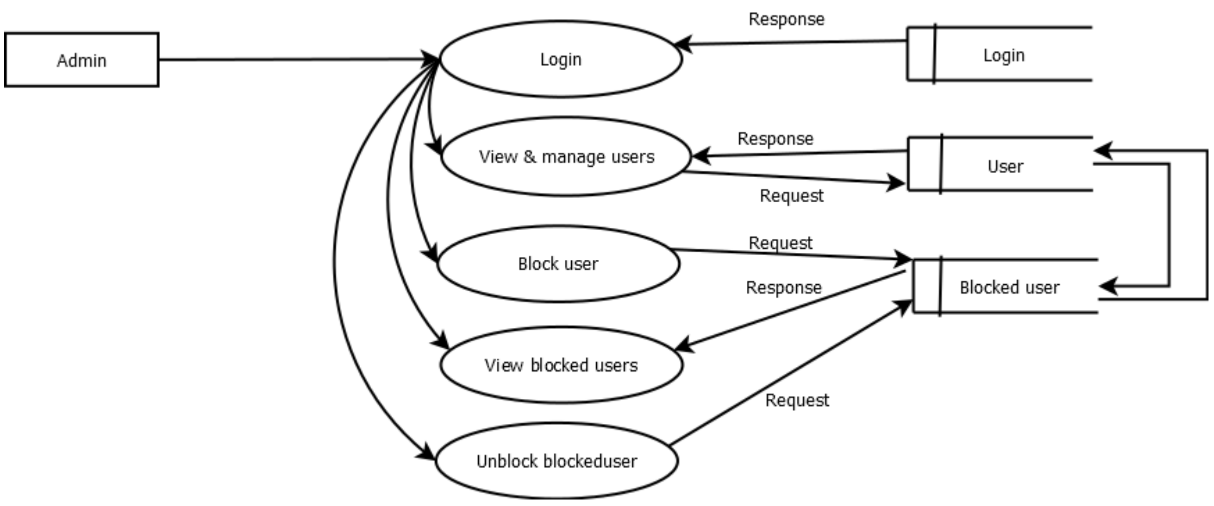


Fig 5.2.2: DFD level 1

LEVEL 2

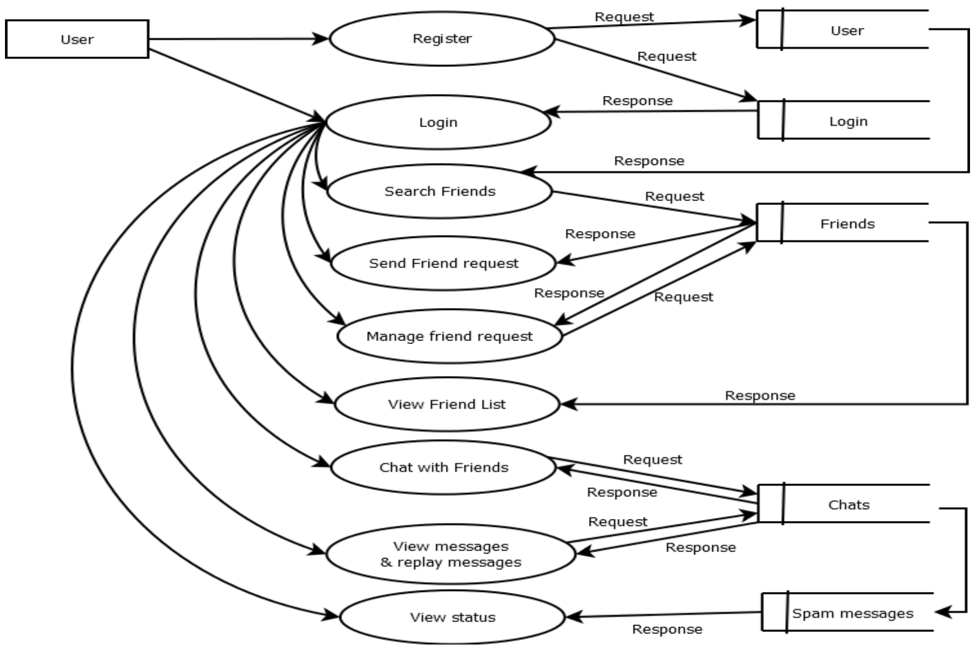
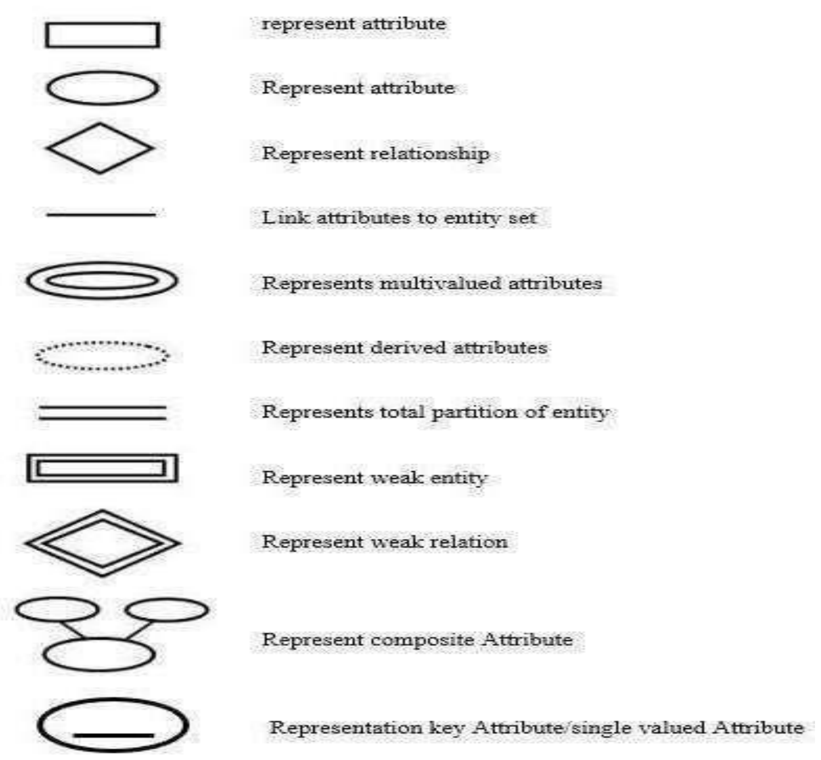


Fig 5.2.3: DFD level 2

5.3 ER DIAGRAM

ENTITY-RELATIONSHIP DIAGRAM (ERD) displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases. At first look, an ER diagram looks very similar to the flowchart.However, ER Diagram includes many specialized symbols, and its meanings make this modelunique. The purpose of ER Diagram is to represent the entity framework infrastructure.



▭	represent attribute
⬭	Represent attribute
◇	Represent relationship
—	Link attributes to entity set
◎	Represents multivalued attributes
⬭	Represent derived attributes
=	Represents total partition of entity
▣	Represent weak entity
◈	Represent weak relation
⬭	Represent composite Attribute
⬭	Representation key Attribute/single valued Attribute

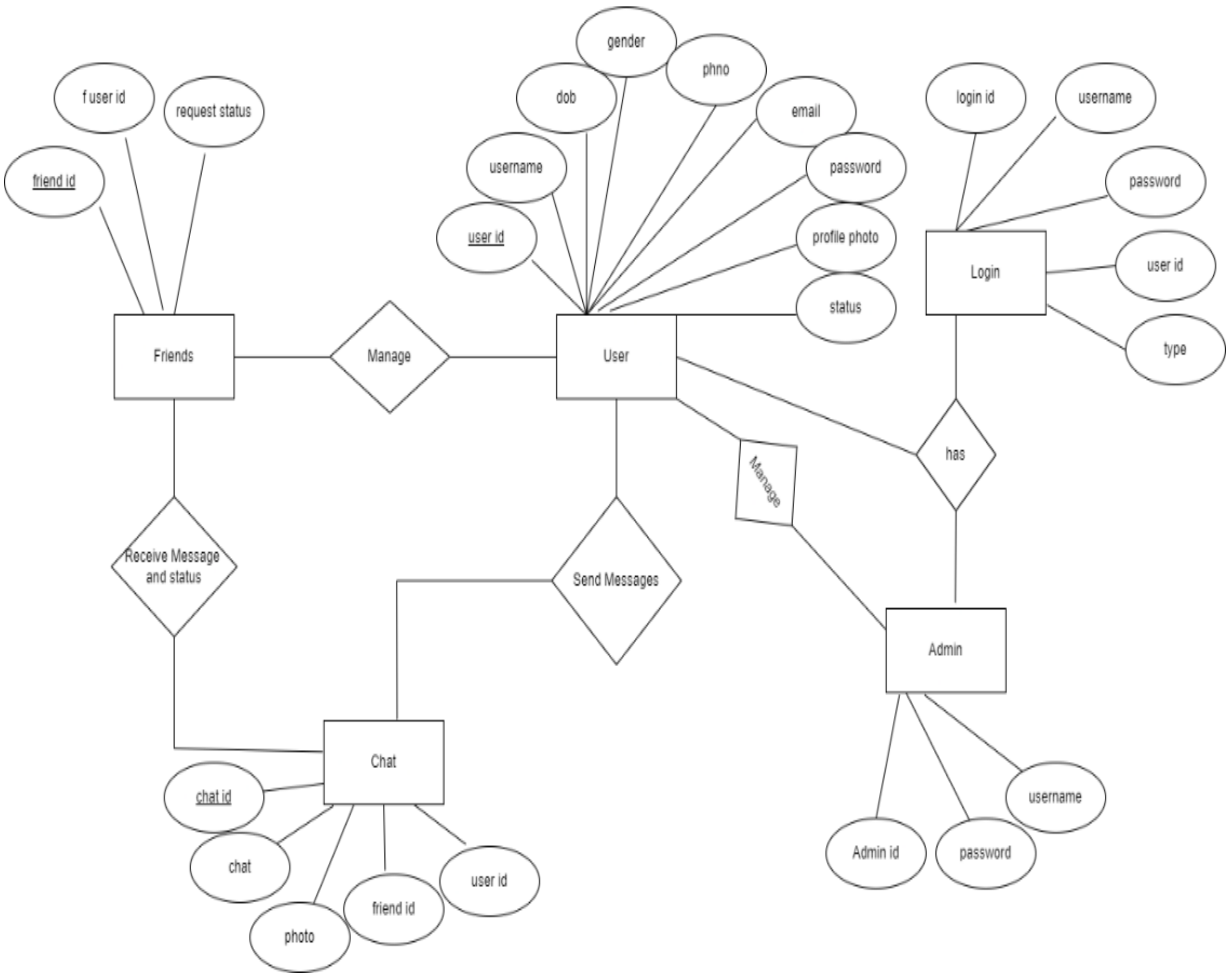


Figure 5.3: ER diagram

5.4 Model Evaluation

In machine learning, model evaluation is the process of assessing the performance of a trained model on a dataset. It is an essential step in the machine learning pipeline as it helps us determine whether the model is accurate and generalizable enough to make predictions on new, unseen data. There are various evaluation metrics used to evaluate a model. The evaluation techniques used in this system are accuracy, precision, recall, F1-score and confusion matrix.

1. **Confusion matrix:** A confusion matrix is a table that is used to evaluate the performance of a machine learning model for a binary or multi-class classification problem. It shows the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for a given set of predictions. In a binary classification problem, a confusion matrix has two classes: positive and negative. The true positives (TP) represent the number of samples that were correctly predicted as positive, while true negatives (TN) represent the number of samples that were correctly predicted as negative. False positives (FP) represent the number of samples that were incorrectly predicted as positive, and false negatives (FN) represent the number of samples that were incorrectly predicted as negative.

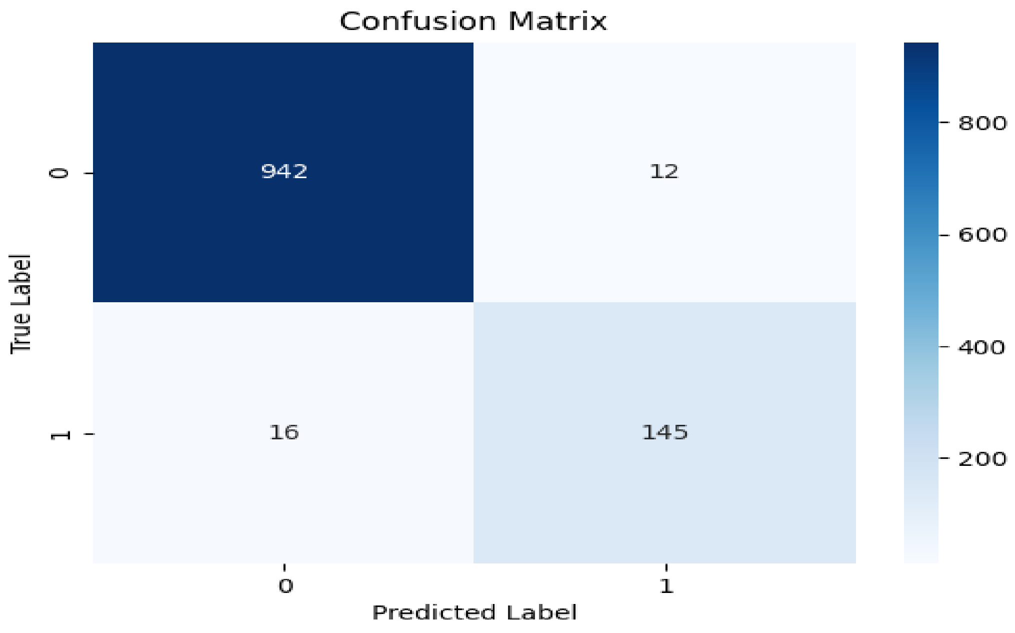


Figure 5.4: Confusion Matrix of the proposed model

2. **Accuracy:** Accuracy is the most common evaluation metric and is defined as the number of correct predictions made by the model divided by the total number of predictions. It gives the overall performance of the model.

Accuracy = (TP + TN) / (TP + TN + FP + FN)

3. **Precision:** Precision is the proportion of true positives out of all the positive predictions made by the model. It measures how precise the model is when predicting positive samples.

   Precision = TP / (TP + FP)

4. **Recall:** Recall is the proportion of true positives out of all the actual positive samples in the dataset. It measures how well the model can identify positive samples.

   Recall = TP / (TP + FN)

5. **F1-score:** F1-score is the harmonic mean of precision and recall. It gives a balance between precision and recall and is often used when both precision and recall are equally important.

   F1-score = 2 * (Precision * Recall) / (Precision + Recall)

The proposed model's performance in terms of accuracy, precision, recall, and F1-score is presented in the following table. The support for the model is 1115.

	Result
Accuracy	97.49%
Precision	97.47%
Recall	97.49%
F1-score	97.48%

Table 5.4: Performance Metrics of Proposed Model

5.5 Implementation

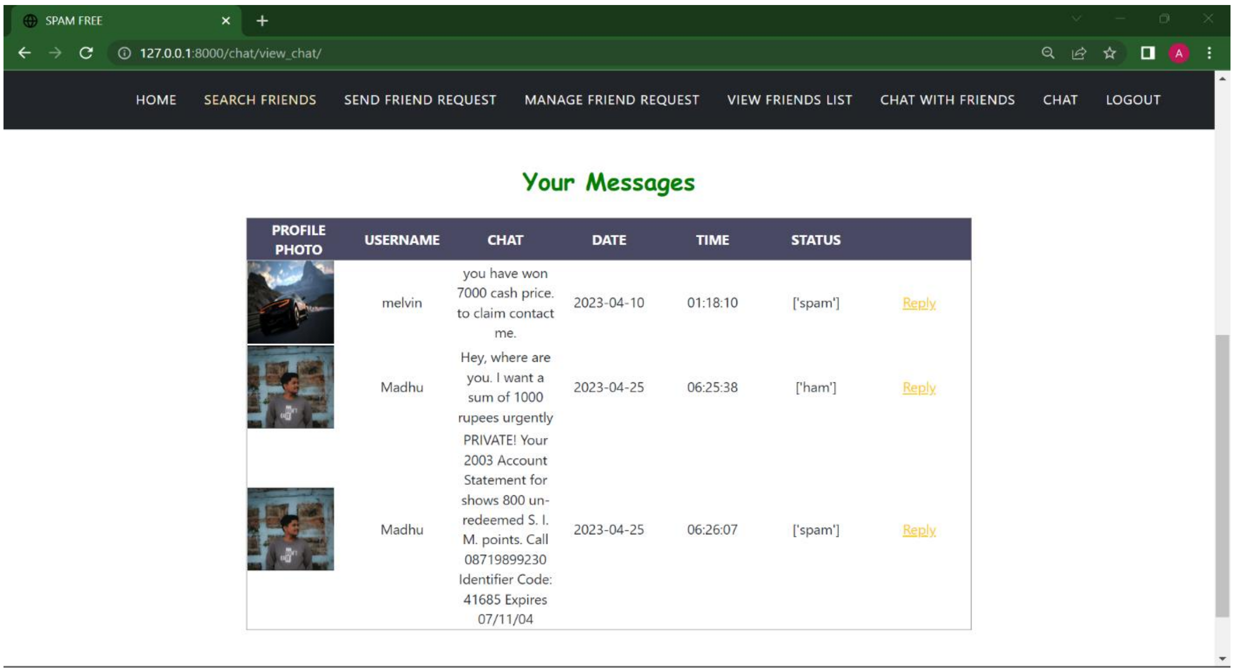
The project online chatting website with spam message detection, consisting of two major modules - admin and user - and implementation modules. Users can register and log in, with their requests being accepted by the admin. Once logged in, users can send and receive friend requests and chat with their friends by sending SMS messages, which may be spam or ham. Users can view the status of the messages to determine if they are spam or ham. To detect spam messages, a machine learning model was developed and implemented in the chat system. To create the model, a dataset containing spam and ham messages was collected and preprocessed, with natural language processing used to extract features. The model was then trained using a decision tree classifier and saved for implementation. When a new message is received, the model reads the complete data in the dataset and places it in a data frame called "trained data." The new message is then attached to this data frame, and the data in each row is preprocessed to remove html tags, short notes, stop words, characters, and numbers. The result is a set of keywords that are vectorized using a TF-IDF vectorizer. After vectorization the result is given to the implemented model. The decision tree model then checks which features from the current message match with the features of other messages in the dataset. Finally, the message is classified as ham or spam using the decision tree classifier, with the output being updated in the status field.

5.6 Experimentation

As a part of experimentation after implementing the model, we passed 3 inputs to test whether the model is able to check whether the message is SPAM or HAM. Input I: given to the developed system: “you have won 7000 cash price. to claim contact me.” Input II: given to the developed system: “Hey, where are you. I want a sum of 1000 rupees urgently” Input III: given to the developed system: “PRIVATE! Your 2003 Account Statement for shows 800 un-redeemed S. I. M. points. Call 08719899230 Identifier Code: 41685 Expires 07/11/04”.

5.7 Experimentation Result

The Output were generated as follows.



Your Messages

PROFILE PHOTO	USERNAME	CHAT	DATE	TIME	STATUS	
	melvin	you have won 7000 cash price. to claim contact me.	2023-04-10	01:18:10	['spam']	Reply
	Madhu	Hey, where are you. I want a sum of 1000 rupees urgently	2023-04-25	06:25:38	['ham']	Reply
	Madhu	PRIVATE! Your 2003 Account Statement for shows 800 un-redeemed S. I. M. points. Call 08719899230 Identifier Code: 41685 Expires 07/11/04	2023-04-25	06:26:07	['spam']	Reply

Fig 5.7: Experimentation result

Chapter 6

Conclusion

6.1 Summary

In conclusion, this project aimed to design and develop an online chatting website with spam message detection using machine learning techniques. We collected SMS data, preprocessed it, and extracted relevant features. We trained a Decision Tree Classifier model on the data and evaluated its performance. The trained model was then deployed to the website for real-time spam detection. We found that the model performed well in detecting spam messages with high accuracy of 97.49%. The website's backend successfully classified incoming messages as spam or ham based on the model's predictions, providing users with a safe and secure chatting environment with their friends. This project has practical applications in preventing unwanted spam messages from reaching users and maintaining the website's reputation.

6.2 Limitations

1. The accuracy of the model can be affected by the quality and quantity of the training data, and new types of spam messages that were not present in the training data may reduce the model's performance.
2. The model's performance may be impacted by other factors that contribute to spam messages, such as the sender's behavior or previous history.
3. The model's performance may decrease over time as spammers develop new techniques to evade spam detection.
4. The lack of other communication options, such as voice or video, may limit the richness and effectiveness of communication, especially in cases where non-verbal cues are important.

6.3 Future Scope

1. Integration of more advanced natural language processing techniques, such as deep learning-based models, to improve the accuracy and efficiency of spam detection.
2. Expansion of the chat system to support other forms of communication, such as voice or video, to enhance the richness and effectiveness of communication.

3. Development of a feedback system that allows users to report misclassified messages and improve the accuracy of the spam detection model.
4. Integration of more comprehensive data sources, such as social media or email, to improve the coverage and diversity of the training data.
5. Implementation of a real-time learning system that continuously updates the model based on new spam messages and user feedback.

Bibliography

Reference

[1] Li, Z., & Yang, X. (2020). Content-based spam message detection using deep learning. Journal of Ambient Intelligence and Humanized Computing, 11(5), 1925-1936.

[2] Alzahrani, F., & Alshammari, R. (2019). A hybrid approach for spam detection in social media. Journal of Ambient Intelligence and Humanized Computing, 10(6), 2179-2191.

[3] Srivastava, R., Khare, N., & Singh, A. (2017). Performance comparison of machine learning algorithms for spam detection. International Journal of Computer Applications, 159(8), 25-29.

[4] Zhang, S., Wu, J., & Li, Y. (2018). A personalized spam message filter based on user preference. Journal of Ambient Intelligence and Humanized Computing, 9(6), 2059-2067.

[5] Luo, C., Chen, X., & Guo, X. (2019). A spam message detection method based on user behavior analysis. Journal of Ambient Intelligence and Humanized Computing, 10(3), 969-977.

[6] Yang, H., Wang, Y., & Wang, Y. (2020). SMS Spam Detection with Deep Learning. IEEE Access, 8, 56632-56639.

[7] Scikit-learn developers. (2021). scikit-learn documentation: Feature extraction. Retrieved September 22, 2021, from https://scikit-learn.org/stable/modules/feature_extraction.html

[8] Kaggle: https://www.kaggle.com/

Appendix

Source Code

```
import pandas as pd
import pickle
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data from Excel file
data = pd.read_excel('spam.xls')


# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data['Content'], data['Label'], test_size=0.2,
random_state=42)

# Create a TF-IDF vectorizer with stopwords removal
vectorizer = TfidfVectorizer(stop_words='english')

# Fit the vectorizer on the training data
X_train_vec = vectorizer.fit_transform(X_train)

# Train a Decision Tree classifier
clf = DecisionTreeClassifier()

# Fit the classifier on the training data
clf.fit(X_train_vec, y_train)

# Save the trained model to file
with open('model_dtc.sav', 'wb') as f:
    pickle.dump(clf, f)

# Load the saved model from file
with open('model_dtc.sav', 'rb') as f:
    clf = pickle.load(f)

# Transform the test data using the trained vectorizer
X_test_vec = vectorizer.transform(X_test)

# Predict the labels of the test data using the loaded model
y_pred = clf.predict(X_test_vec)

# Calculate the accuracy, precision, recall, f1-score, and support of the model
```

```
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=['ham', 'spam'], output_dict=True)
precision = report['weighted avg']['precision']
recall = report['weighted avg']['recall']
f1_score = report['weighted avg']['f1-score']
support = report['weighted avg']['support']

# Print the accuracy, precision, recall, f1-score, and support of the model
print("Accuracy: {:.2%}".format(accuracy))
print("Precision: {:.2%}".format(precision))
print("Recall: {:.2%}".format(recall))
print("F1-score: {:.2%}".format(f1_score))
print("Support: {}".format(support))

# Create a pie chart of the test set labels
labels = ['Ham', 'Spam']
sizes = [len(y_test[y_test=='ham']), len(y_test[y_test=='spam'])]
fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
ax1.axis('equal')
plt.title("Distribution of Test Set Labels")
plt.show()

# Create a confusion matrix of the test data using the loaded model
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()



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_free 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_n
ame='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.shortcuts import render
from friends.models import Friends
from django.http import HttpResponseRedirect,HttpResponse
from user_reg.models import UserReg
from chat.models import Chat
from django.db import connection
from django.db.models import Q
from reported_comments.models import ReportedComments

# Create your views here.
def search_friends(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":
        search_friend=request.POST.get('search')
        request.session['name']=search_friend
        # return search_result(request)
        # return HttpResponseRedirect('/friends/search_result/')
        obj=UserReg.objects.filter(username__icontains=search_friend,status='approved')
        print(obj)
        searchf={
            'search':obj,
            'name':objj
        }
        return render(request,'friends/search_friends.html',searchf)
    return render(request,'friends/search_friends.html',context)

def search_send_request(request,df):
    fid=UserReg.objects.get(user_id=df)
    print(df)
    uid = request.session["uid"]
    objj = UserReg.objects.filter(user_id=uid)  # name
    context={
        'name':objj
    }
    if Friends.objects.filter(f_user_id=uid,user_id=fid.user_id).exists():
        pass
    else:
        ob=Friends()
        ob.user_id=uid
        ob.f_user_id=df
        ob.request_status="requested"
        ob.save()
        return HttpResponseRedirect('/friends/search_friends/')

    return render(request,'friends/search_friends.html',context)

def send_friend_request(request):
    uid = request.session["uid"]
    objj = UserReg.objects.filter(user_id=uid)  # name
    ff=Friends.objects.filter(f_user_id=uid).values_list('user_id',flat=True)
    rr=Friends.objects.filter(user_id=uid).values_list('f_user_id',flat=True)
    # fff=ff+str(uid)
    print(rr)
    print(ff)
    obj                                                                    =
UserReg.objects.filter(status='approved').exclude(user_id__in=ff).exclude(user_id=uid).exclu
de(user_id__in=rr)
    gm = Chat.objects.filter(friend_id=uid)
    request_friends = {
        'friends': obj,
        'name':objj,
        'chat':gm
    }

    return render(request,'friends/send_friend_request.html',request_friends)

def send_request(request,abc):
    uid = request.session["uid"]
    if Friends.objects.filter(user_id=uid,f_user_id=abc).exists():
        pass
    else:
        obj = Friends()
        obj.user_id = uid
        obj.f_user_id = abc
        obj.request_status = "requested"
        obj.save()
    return HttpResponseRedirect('/friends/send_friend_request/')

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

    obj=Friends.objects.filter(request_status='requested',f_user_id=uid)
    manage_friend = {
        'friend': obj,
        'name':objj,
        'chat':gm
    }


    return render(request,'friends/manage_friend_request.html',manage_friend)
```

```
def accept_friend(request,abc):
    uid = request.session["uid"]
    obj=Friends.objects.get(friend_id=abc)
    obj.request_status="accepted"
    obj.save()

    ob=Friends()
    ob.f_user_id=obj.user_id
    ob.user_id=uid
    ob.request_status="accepted"
    ob.save()
    # return manage_friend_request(request)
    return HttpResponseRedirect('/friends/manage_friend_request/')
    # return render(request,'friends/manage_friend_request.html')


def reject_friend(request,cdf):
    obj=Friends.objects.get(friend_id=cdf)
    obj.request_status="rejected"
    obj.save()

    return manage_friend_request(request)


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

    cd=UserReg.objects.filter(status='cyberbullying detected').values_list('user_id',flat=True)
    print(cd)


obj=Friends.objects.filter(user_id=uid,request_status='accepted').exclude(f_user_id__in=cd)
    print(obj)

    gm = Chat.objects.filter(friend_id=uid)


    friends = {
        'friend': obj,
        'name':objj,
        'chat':gm,
        # 'images': o
        # bjlist.fetchall()

    }


    return render(request,'friends/view_friends_list.html',friends)
```

```
def chat_with_friends(request):
    return render(request,'friends/chat_with_friends.html')



def search_result(request):
    uid = request.session["uid"]
    objj = UserReg.objects.filter(user_id=uid)  # name
    # gm = Chat.objects.filter(user_id=uid).exclude(friend_id=uid)
    ff=request.session['name']
    al=Friends.objects.filter(f_user_id=uid).values_list('user_id',flat=True)
    print(al)
    # ff=Friends.objects.filter(f_user_id=uid).values_list('user_id',flat=True)
    context={
        'name':objj,

    }
    if
UserReg.objects.filter(username__icontains=ff,status='approved').exclude(user_id=uid).exclu
de(user_id__in=al):


obj=UserReg.objects.filter(username__icontains=ff,status='approved').exclude(user_id=uid).e
xclude(user_id__in=al)
        print(obj)
        gm = Chat.objects.filter(friend_id=uid)
        searchf={
            'search':obj,
            'name':objj,
            'chat':gm
        }
    # return render(request,'friends/search_friends.html',searchf)
        return render(request,'friends/search_result.html',searchf)
    return render(request,'friends/search_result.html',context)

def already_friend(request):



     return render(request,'friends/already_friend.html')

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

    objlist = connection.cursor()
    # objlist.execute( "SELECT * FROM user_reg,reported_comments WHERE
user_reg.user_id=%s AND reported_comments.user_id=user_reg.user_id",[uid])
    objlist.execute("SELECT user_reg.*,reported_comments.*,friends.* FROM friends INNER
JOIN reported_comments on friends.f_user_id=reported_comments.user_id INNER JOIN
user_reg on user_reg.user_id=reported_comments.user_id WHERE
friends.request_status='accepted' and friends.user_id=%s",[uid])
    gm = Chat.objects.filter(friend_id=uid)

    fr=Friends.objects.filter(user_id=uid).values_list('f_user_id',flat=True)
    print(fr)

    ob = UserReg.objects.filter(user_id__in=fr,status='blocked').values_list('user_id',flat=True)
    print(ob)
    # obj = ReportedComments.objects.all()

    blocked = {
        # 'block': obj,
         'bl':ob,
        'name':objj,
        'images': objlist.fetchall(),
        'chat':gm
    }
    return render(request,'friends/view_blocked_friends.html',blocked)
```

```
from django.db import models
from user_reg.models import UserReg
from reported_comments.models import ReportedComments
# Create your models here.
class Friends(models.Model):
    friend_id = models.AutoField(primary_key=True)
    # user_id = models.IntegerField()
    user                                                                        =
models.ForeignKey(UserReg,to_field='user_id',on_delete=models.CASCADE,related_name=
'f1')
    #       report=       models.ForeignKey(ReportedComments,       to_field='user_id',
on_delete=models.CASCADE, related_name='f1')
    # f_user_id = models.IntegerField()

f_user=models.ForeignKey(UserReg,to_field='user_id',on_delete=models.CASCADE,related
_name='f2')
    request_status = models.CharField(max_length=50)

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

Output

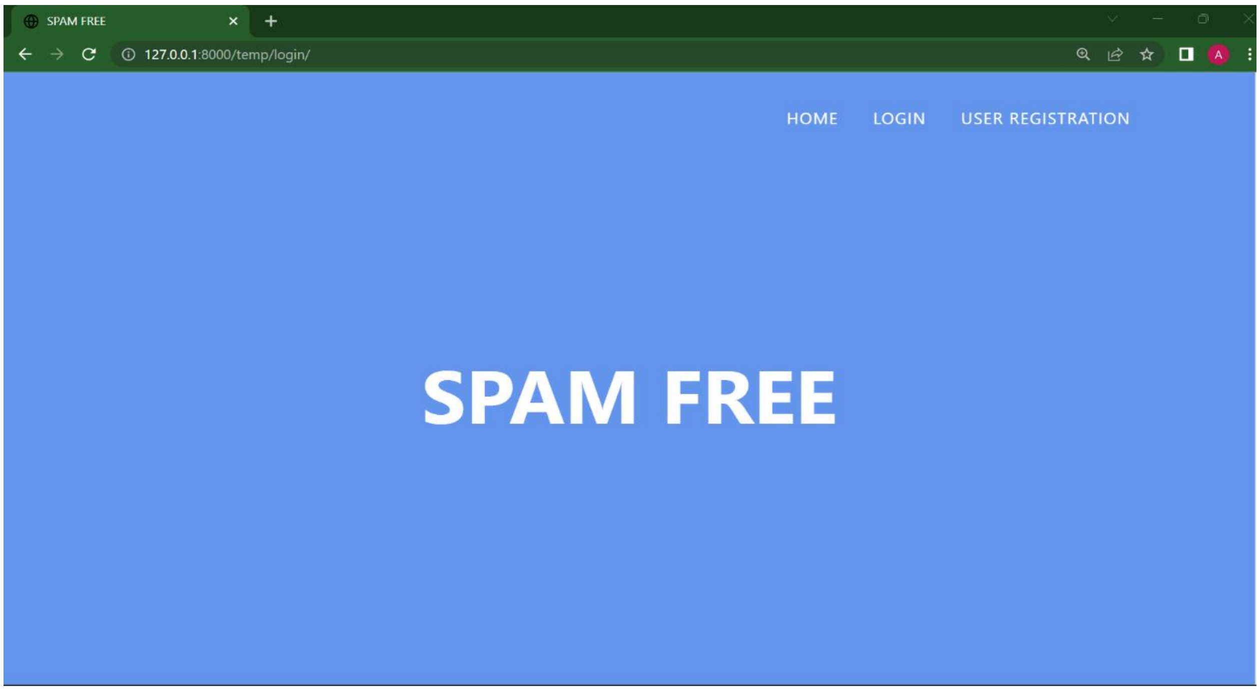


Fig A.1: Home Page

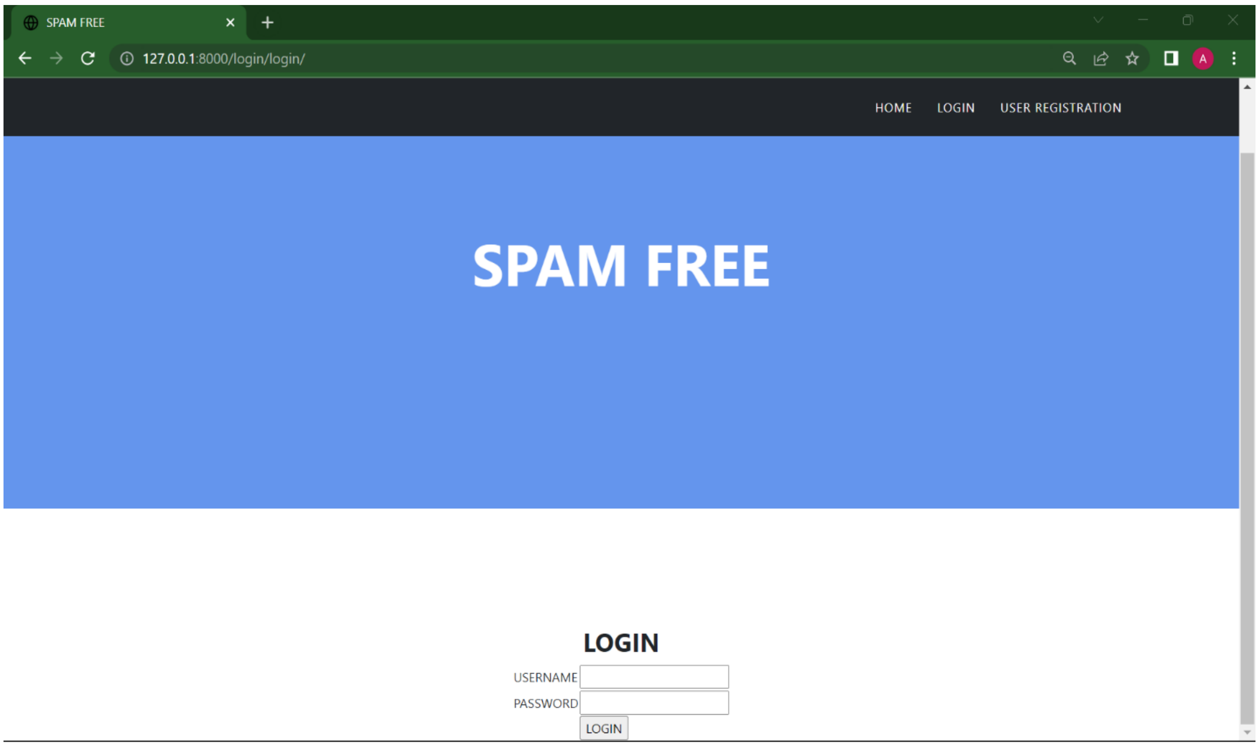
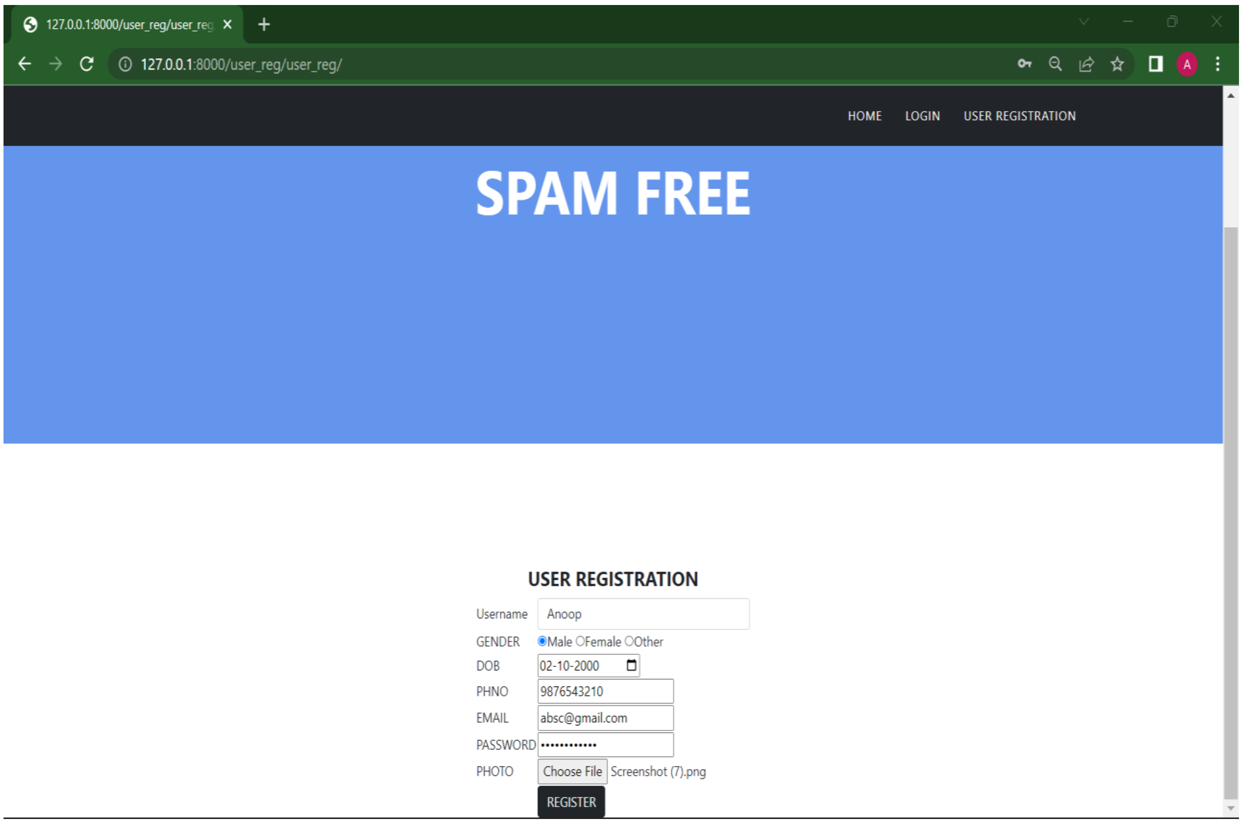


Fig A.2: Login Page



Username Anoop
GENDER ●Male ○Female ○Other
DOB 02-10-2000
PHNO 9876543210
EMAIL absc@gmail.com
PASSWORD ••••••••••••
PHOTO Screenshot (7).png

Fig A.3: User registration Page

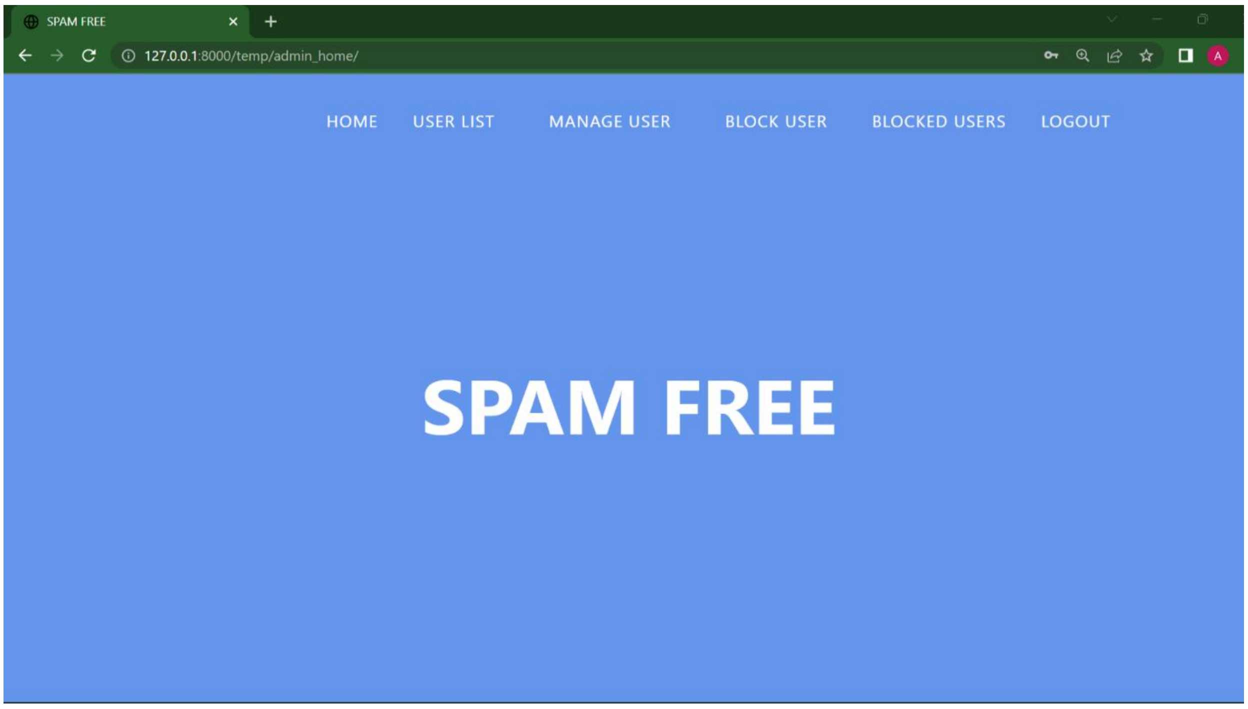
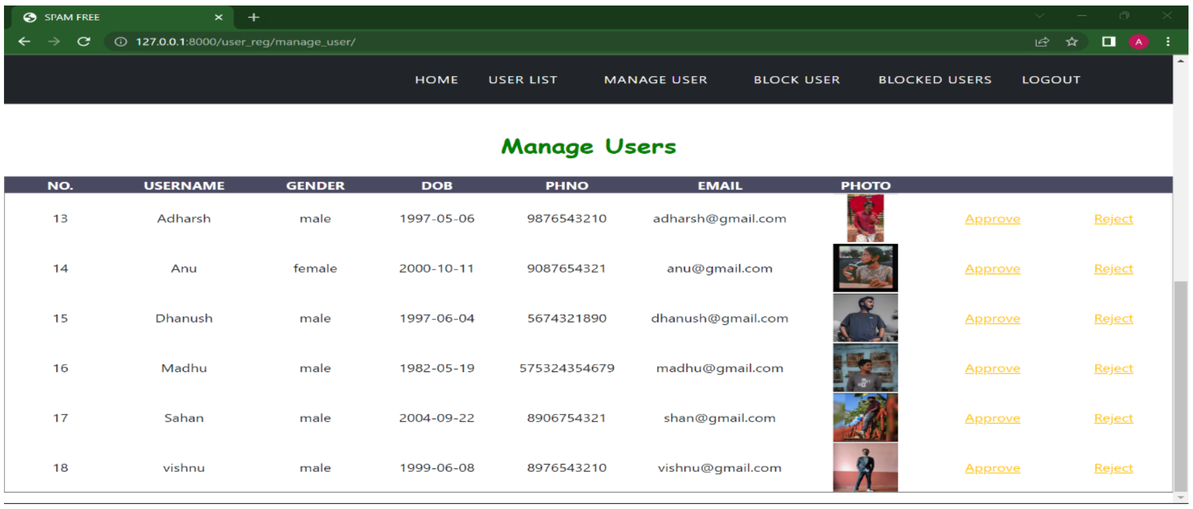


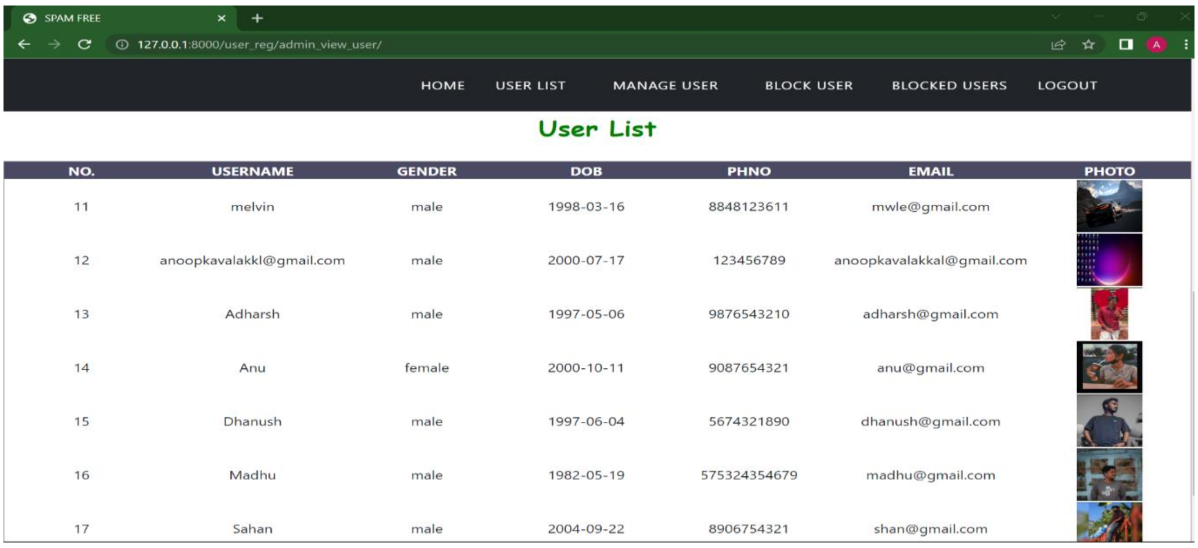
Fig A.4: Admin Dashboard



Manage Users

NO.	USERNAME	GENDER	DOB	PHNO	EMAIL	PHOTO		
13	Adharsh	male	1997-05-06	9876543210	adharsh@gmail.com		Approve	Reject
14	Anu	female	2000-10-11	9087654321	anu@gmail.com		Approve	Reject
15	Dhanush	male	1997-06-04	5674321890	dhanush@gmail.com		Approve	Reject
16	Madhu	male	1982-05-19	575324354679	madhu@gmail.com		Approve	Reject
17	Sahan	male	2004-09-22	8906754321	shan@gmail.com		Approve	Reject
18	vishnu	male	1999-06-08	8976543210	vishnu@gmail.com		Approve	Reject

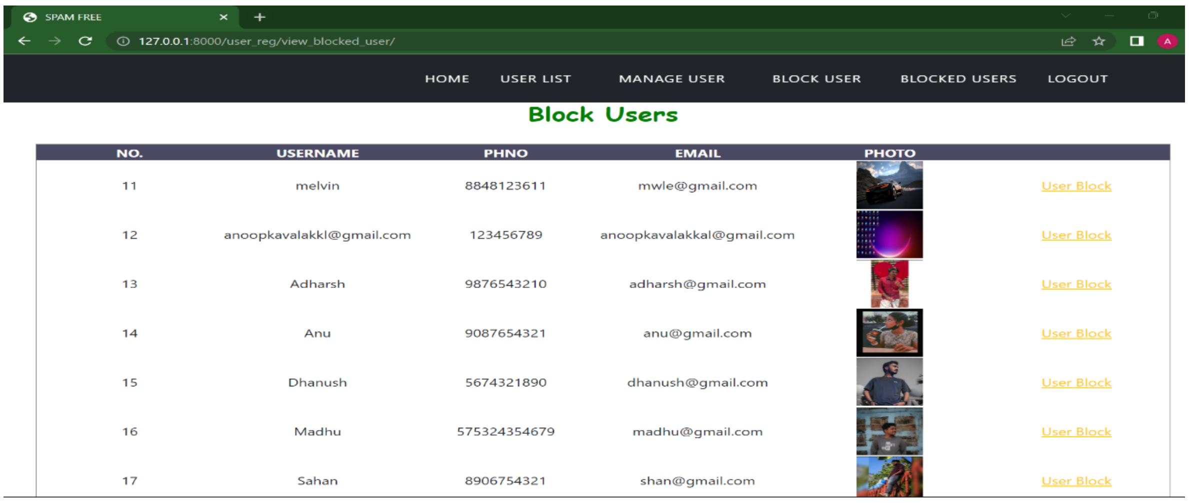
Fig A.5: Admin Manage users page



User List

NO.	USERNAME	GENDER	DOB	PHNO	EMAIL	PHOTO
11	melvin	male	1998-03-16	8848123611	mwle@gmail.com	
12	anoopkavalakkl@gmail.com	male	2000-07-17	123456789	anoopkavalakkal@gmail.com	
13	Adharsh	male	1997-05-06	9876543210	adharsh@gmail.com	
14	Anu	female	2000-10-11	9087654321	anu@gmail.com	
15	Dhanush	male	1997-06-04	5674321890	dhanush@gmail.com	
16	Madhu	male	1982-05-19	575324354679	madhu@gmail.com	
17	Sahan	male	2004-09-22	8906754321	shan@gmail.com	

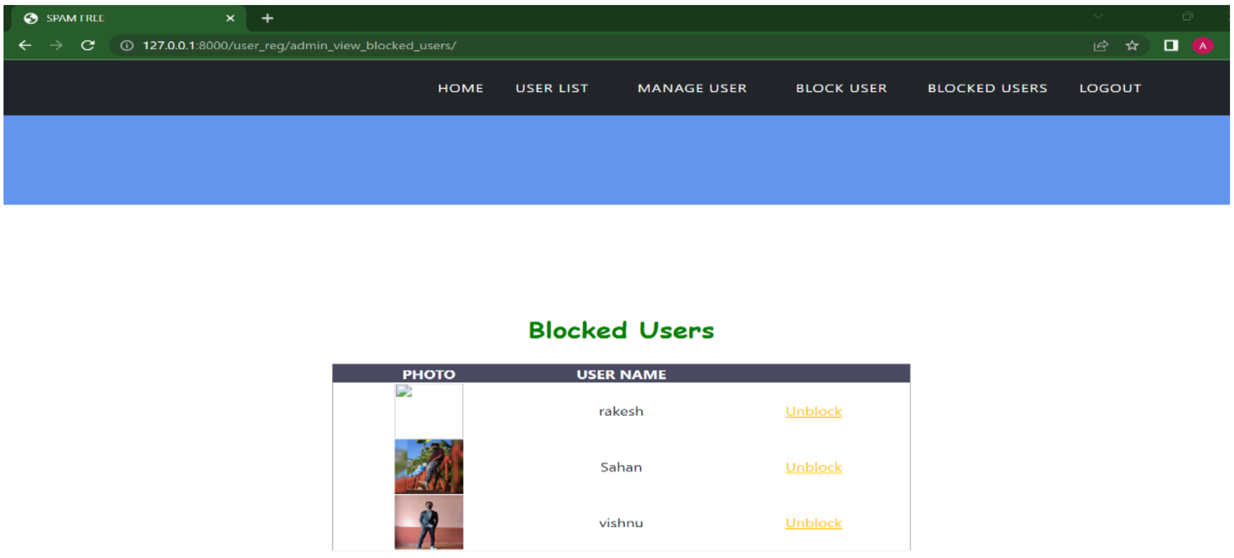
Fig A.6: User list



Block Users

NO.	USERNAME	PHNO	EMAIL	PHOTO	
11	melvin	8848123611	mwle@gmail.com		User Block
12	anoopkavalakkl@gmail.com	123456789	anoopkavalakkal@gmail.com		User Block
13	Adharsh	9876543210	adharsh@gmail.com		User Block
14	Anu	9087654321	anu@gmail.com		User Block
15	Dhanush	5674321890	dhanush@gmail.com		User Block
16	Madhu	575324354679	madhu@gmail.com		User Block
17	Sahan	8906754321	shan@gmail.com		User Block

Fig A.7: Admin block users



Blocked Users

PHOTO	USER NAME	
	rakesh	Unblock
	Sahan	Unblock
	vishnu	Unblock

Fig A.8: Admin blocked users page

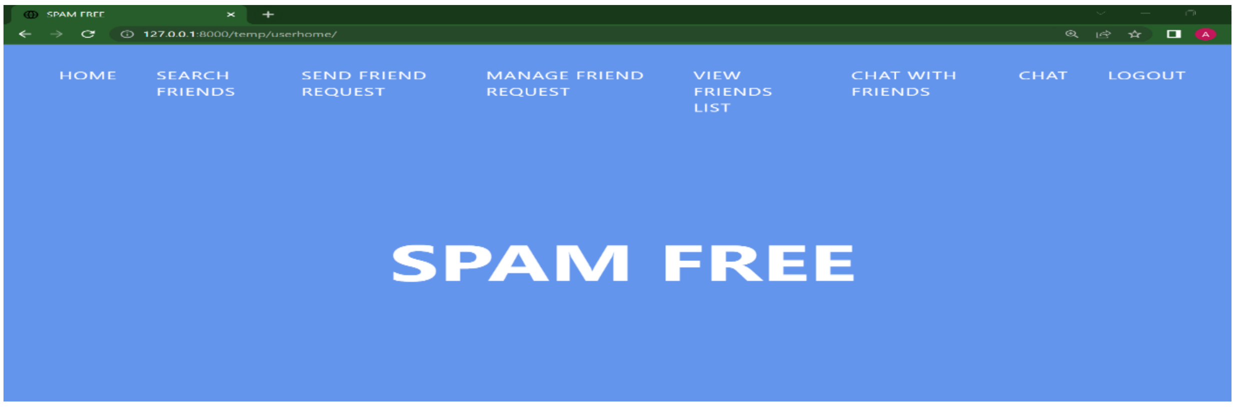
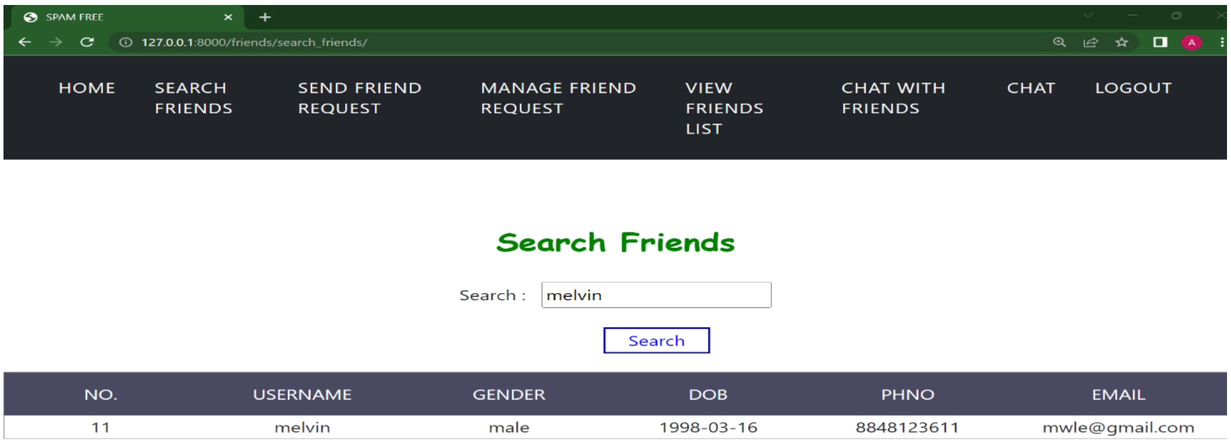


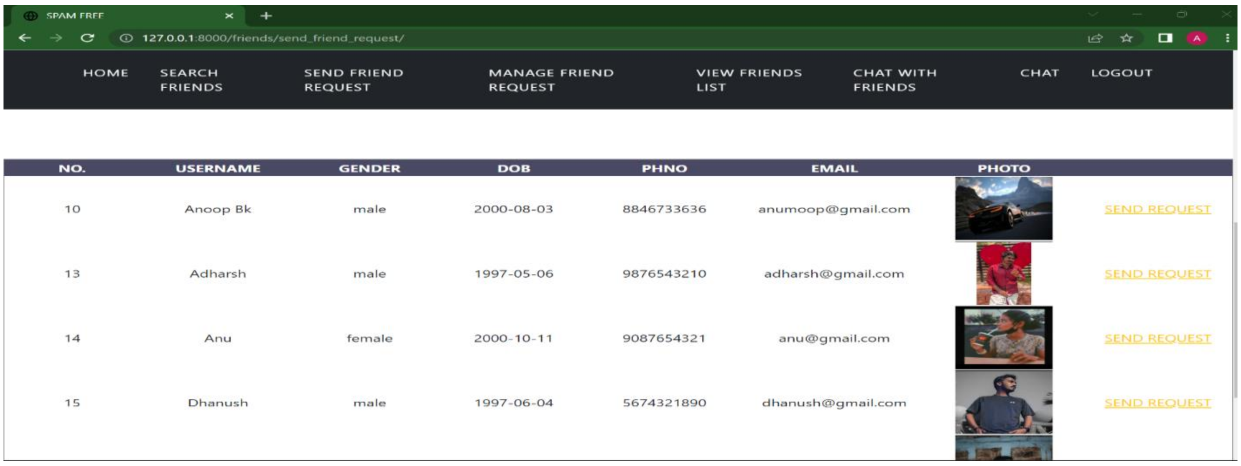
Fig A.9: User Home page



Search Friends

NO.	USERNAME	GENDER	DOB	PHNO	EMAIL
11	melvin	male	1998-03-16	8848123611	mwle@gmail.com

Fig A.10: Search Friends



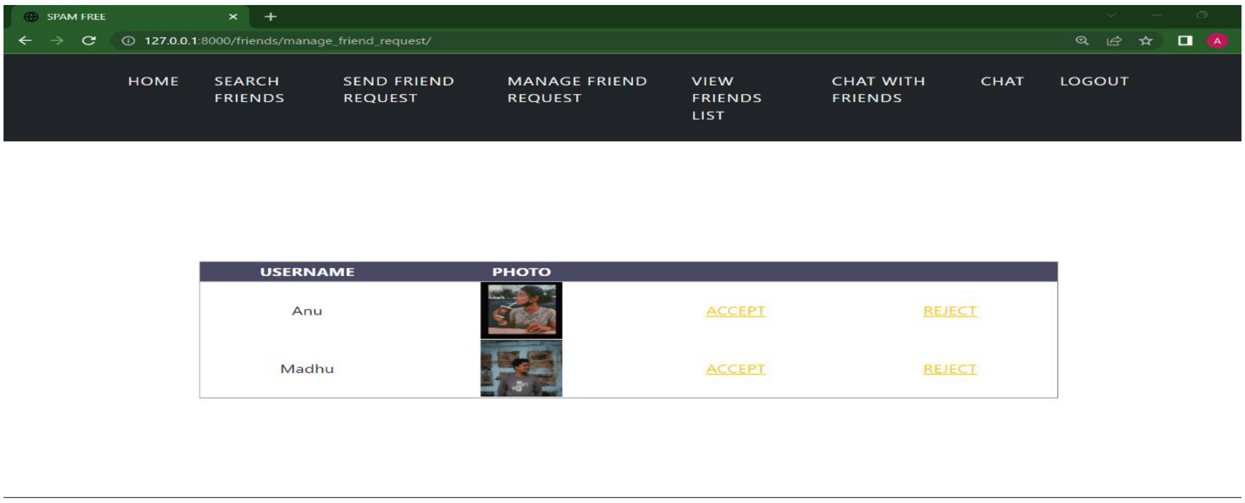
SPAM FREE

127.0.0.1:8000/friends/send_friend_request/

HOME | SEARCH FRIENDS | SEND FRIEND REQUEST | MANAGE FRIEND REQUEST | VIEW FRIENDS LIST | CHAT WITH FRIENDS | CHAT | LOGOUT

NO.	USERNAME	GENDER	DOB	PHNO	EMAIL	PHOTO	
10	Anoop Bk	male	2000-08-03	8846733636	anumoop@gmail.com		SEND REQUEST
13	Adharsh	male	1997-05-06	9876543210	adharsh@gmail.com		SEND REQUEST
14	Anu	female	2000-10-11	9087654321	anu@gmail.com		SEND REQUEST
15	Dhanush	male	1997-06-04	5674321890	dhanush@gmail.com		SEND REQUEST

Fig A.11: Send Friend request



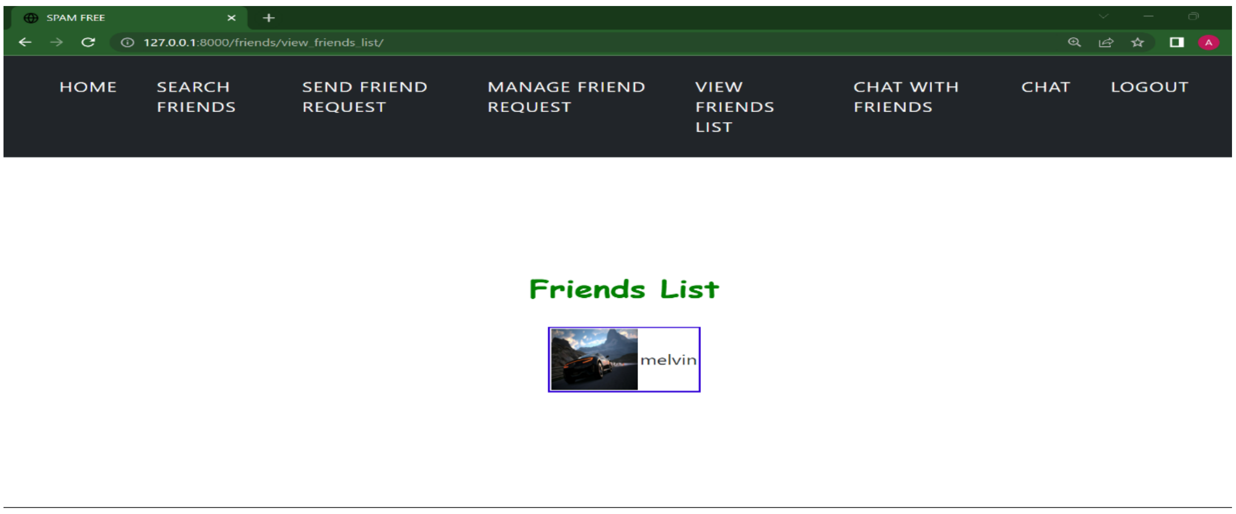
SPAM FREE

127.0.0.1:8000/friends/manage_friend_request/

HOME | SEARCH FRIENDS | SEND FRIEND REQUEST | MANAGE FRIEND REQUEST | VIEW FRIENDS LIST | CHAT WITH FRIENDS | CHAT | LOGOUT

USERNAME	PHOTO		
Anu		ACCEPT	REJECT
Madhu		ACCEPT	REJECT

Fig A.12: Manage Friend request



SPAM FREE

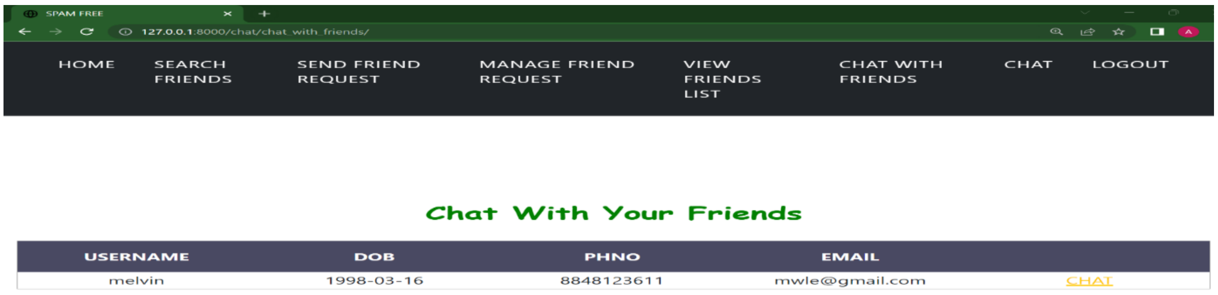
127.0.0.1:8000/friends/view_friends_list/

HOME | SEARCH FRIENDS | SEND FRIEND REQUEST | MANAGE FRIEND REQUEST | VIEW FRIENDS LIST | CHAT WITH FRIENDS | CHAT | LOGOUT

Friends List

melvin

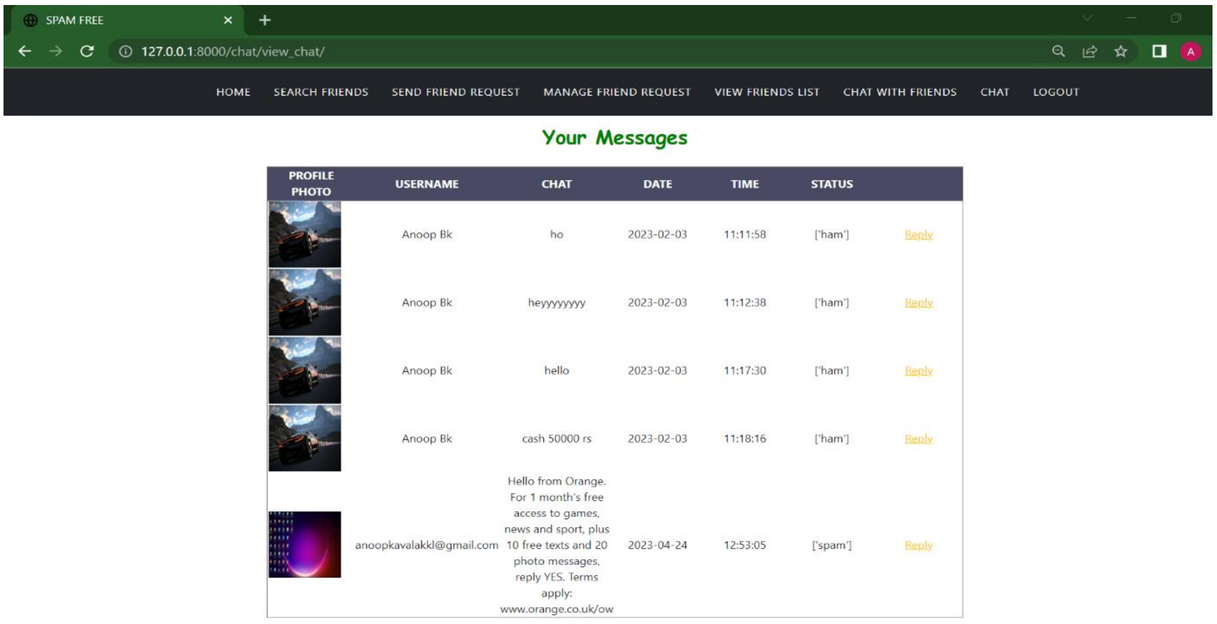
Fig A.13: View Friends list



Chat With Your Friends

USERNAME	DOB	PHNO	EMAIL	
melvin	1998-03-16	8848123611	mwle@gmail.com	CHAT

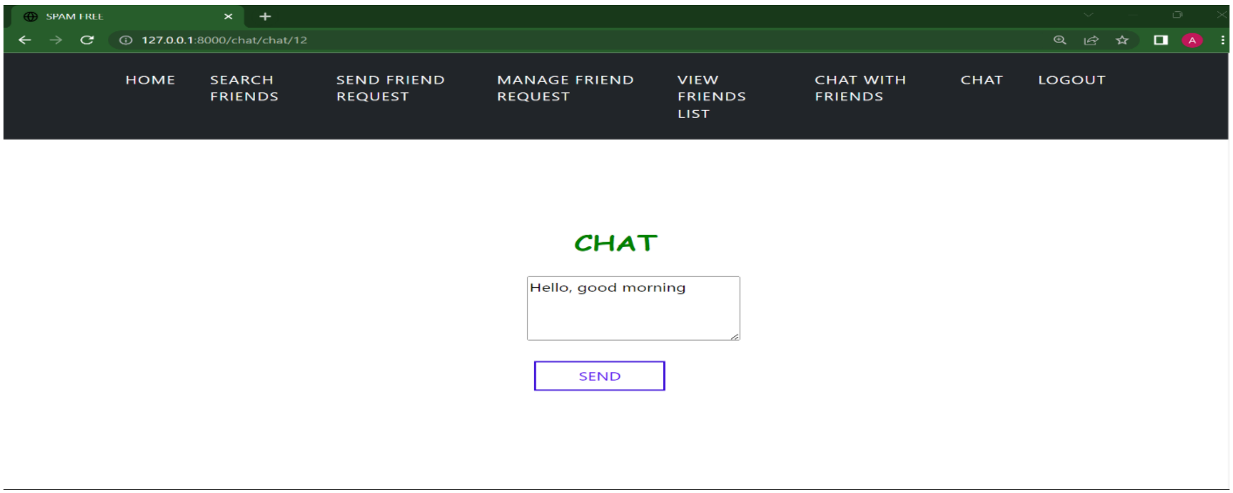
Fig A.14: Chat with friends Friends



Your Messages

PROFILE PHOTO	USERNAME	CHAT	DATE	TIME	STATUS	
	Anoop Bk	ho	2023-02-03	11:11:58	['ham']	Reply
	Anoop Bk	heyyyyyyyy	2023-02-03	11:12:38	['ham']	Reply
	Anoop Bk	hello	2023-02-03	11:17:30	['ham']	Reply
	Anoop Bk	cash 50000 rs	2023-02-03	11:18:16	['ham']	Reply
	anoopkavalakkl@gmail.com	Hello from Orange. For 1 month's free access to games, news and sport, plus 10 free texts and 20 photo messages, reply YES. Terms apply: www.orange.co.uk/ow	2023-04-24	12:53:05	['spam']	Reply

Fig A.15: Chat Page

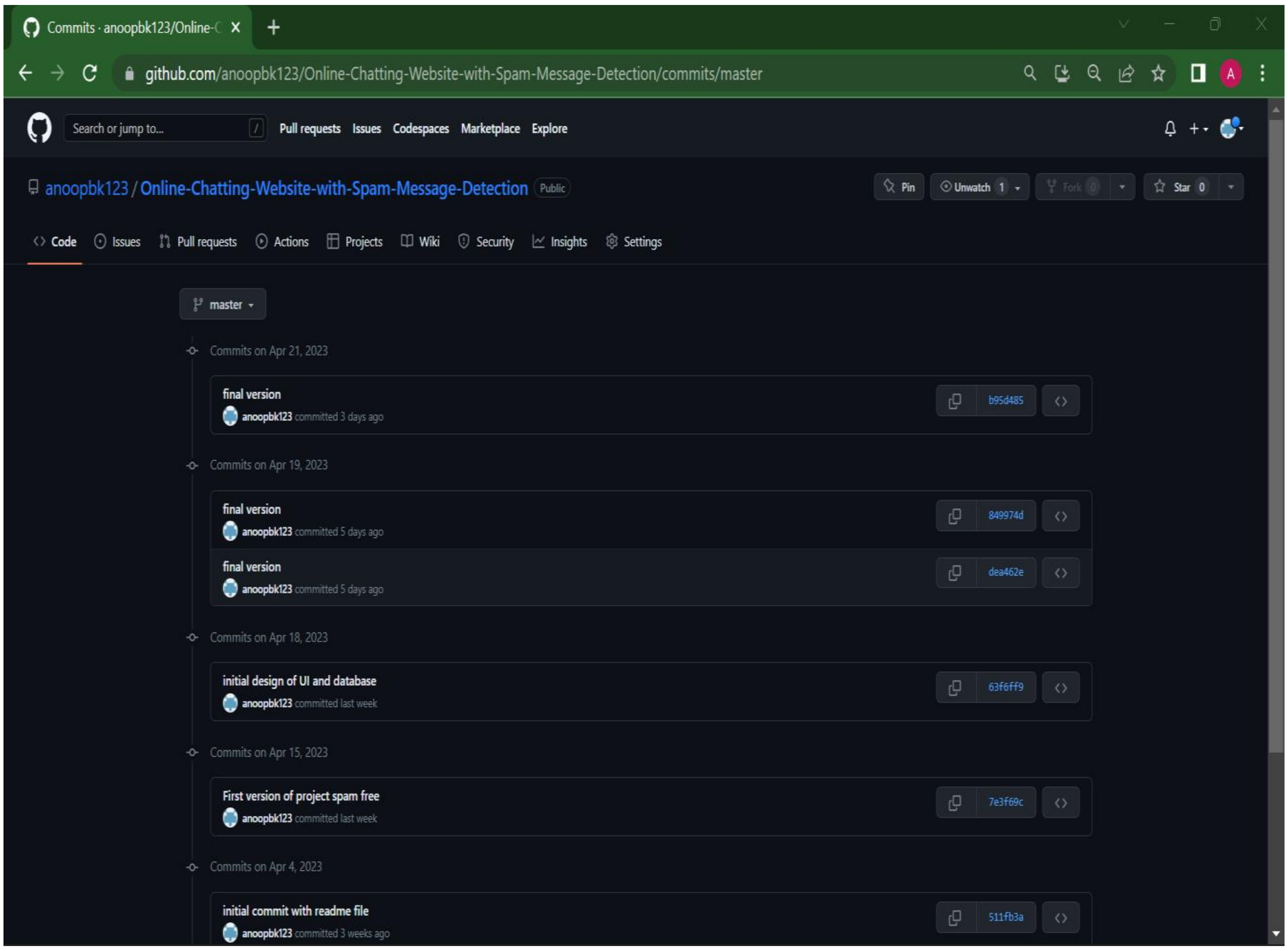


Hello, good morning

Fig A.16: Chat Box

Git History

Git Commit History



github.com/anoopbk123/Online-Chatting-Website-with-Spam-Message-Detection/commits/master

anoopbk123 / Online-Chatting-Website-with-Spam-Message-Detection Public

master

Commits on Apr 21, 2023

final version
anoopbk123 committed 3 days ago — b95d485

Commits on Apr 19, 2023

final version
anoopbk123 committed 5 days ago — 849974d

final version
anoopbk123 committed 5 days ago — dea462e

Commits on Apr 18, 2023

initial design of UI and database
anoopbk123 committed last week — 63f6ff9

Commits on Apr 15, 2023

First version of project spam free
anoopbk123 committed last week — 7e3f69c

Commits on Apr 4, 2023

initial commit with readme file
anoopbk123 committed 3 weeks ago — 511fb3a

Fig A.17: Git Commit History

PUBLICATION

ONLINE CHATTING WEBSITE WITH SPAM MESSAGE DETECTION

Anoop B K, Student

Mr Pramod K, Associate Professor

Department of MCA, Nehru College of Engineering and Research Centre, Thrissur, Kerala

ABSTRACT

Spam is a hot topic for decades. After all the tech advancements still all of us stumble upon spam messages then and now. Spam has become more realistic where a normal people could not distinguish whether it is real or not. Random job offers appearing on your WhatsApp messenger from unknown source, Spam messages on every comment section of almost all the social medias. Occasionally, individuals may receive spam messages from their friends. Tech giants are spending millions to keep these spammers from their application. The project titled "ONLINE CHATTING WEBSITE WITH SPAM MESSAGE DETECTION" is aimed to provide a spam free chatting system to users. To detect and avoid spam messages we can use natural language processing and machine learning. In this system we developed a machine learning model using TF-IDF Vectorizer algorithm and Decision Tree Classifier to predict a SMS is a spam or non-spam(ham) and it was discovered that the model out performs existing models. This model is used to predict the spam messages in this online chatting website. In this website user can register and login if his request of registration is accepted by the admin. The user can send friend request to other users and can send SMS to their friends. The system evaluates messages and predicts whether they are spam or ham.

INTRODUCTION

In today's digital age, online communication has become an essential part of our daily lives. With the rise of social media platforms and online messaging apps, people now prefer to communicate with each other online rather than through traditional methods. However, while online communication has its benefits, it also has its drawbacks – spam messages. Spam messages are unwanted and irrelevant messages that fill up our inboxes and make it difficult to find and read important messages. To address this problem, we are proposing an online chatting website called Spam Free, which incorporates spam message detection. Our goal is to provide users with a safe and secure online chatting experience by automatically detecting spam messages.

LITERATURE SURVEY

Spam message detection has been a topic of extensive research in the field of natural language processing and machine learning. Various techniques and algorithms have been proposed for identifying and filtering out spam messages from the vast amount of online communication. In this literature survey, we review some of the previous studies that have been conducted in this area.

Li and Yang were done research on "Content-based spam message detection using deep learning". They proposed a content-based approach for spam message detection using deep learning algorithms. They used a neural network model to classify messages as spam or non-spam based on their content. The study highlights the effectiveness of deep learning algorithms for spam message detection, particularly in identifying complex and novel spam messages.

Alzahrani and Alshammari were done research on "A hybrid approach for spam detection in social media". They proposed a hybrid approach for spam message detection, combining rule-based and content-based filtering techniques. The study found that the hybrid approach outperformed either technique alone, highlighting the importance of using multiple techniques for spam message detection. However, the approach may require significant computational resources and may not be efficient for real-time spam detection.

Srivastava et al. were done research on "Performance comparison of machine learning algorithms for spam detection." In this study they compared the performance of Naive Bayes, Decision Tree Classifier, and Support Vector Machine on a dataset of spam and non-spam messages. The study found that all three algorithms achieved high accuracy in detecting spam messages, with Decision Tree Classifier performing the best.

METHODOLOGY

The project online chatting website with spam message detection, consisting of two major modules – admin and user – and implementation modules. Users can register and log in, with their requests being accepted by the admin. Once logged in, users can send and receive friend requests and chat with their friends by sending SMS messages, which may be spam or ham.

Users can view the status of the messages to determine if they are spam or ham. To detect spam messages, a machine learning model was developed and implemented in the chat system. The work flow of the proposed system is given below:

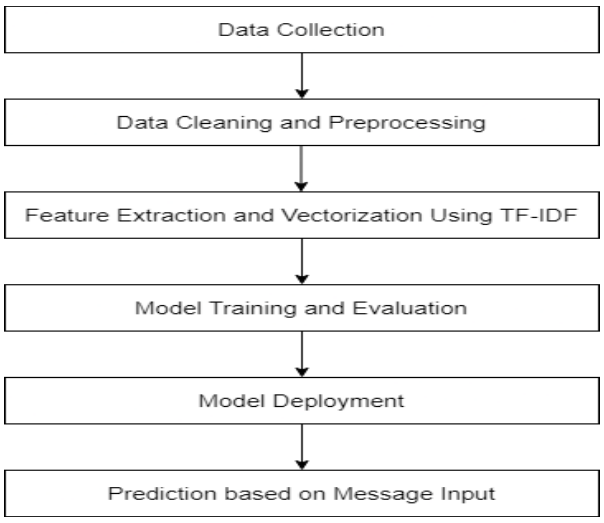

Data Collection
Data Cleaning and Preprocessing
Feature Extraction and Vectorization Using TF-IDF
Model Training and Evaluation
Model Deployment
Prediction based on Message Input

Fig 1: Flowchart of the Proposed System

To create the model, a dataset containing spam and ham messages was collected and pre-processed, with natural language processing used to extract features. The model was then trained using a decision tree classifier and saved for implementation. When a new message is received, the model reads the complete data in the dataset and places it in a data frame called "trained data." The new message is then attached to this data frame, and the data in each row is pre-processed to remove html tags, short notes, stop words, characters, and numbers. The result is a set of keywords that are vectorized using a TF-IDF vectorizer.

In this system we are using the TF-IDF for feature extraction. 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 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 do not give any relevant information about a particular document type or class. The following Figure represents the TF-IDF mathematical formula.

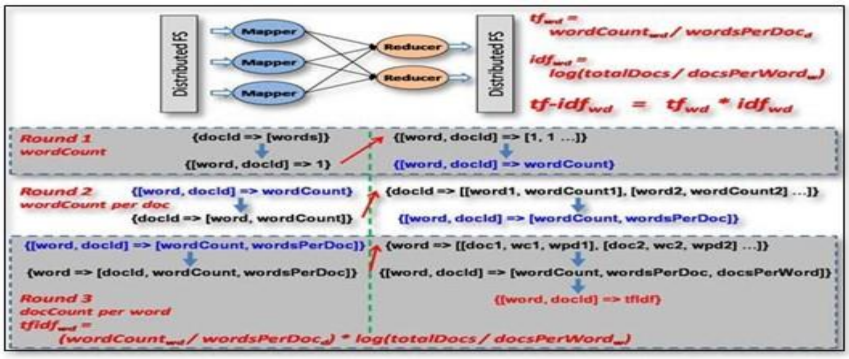
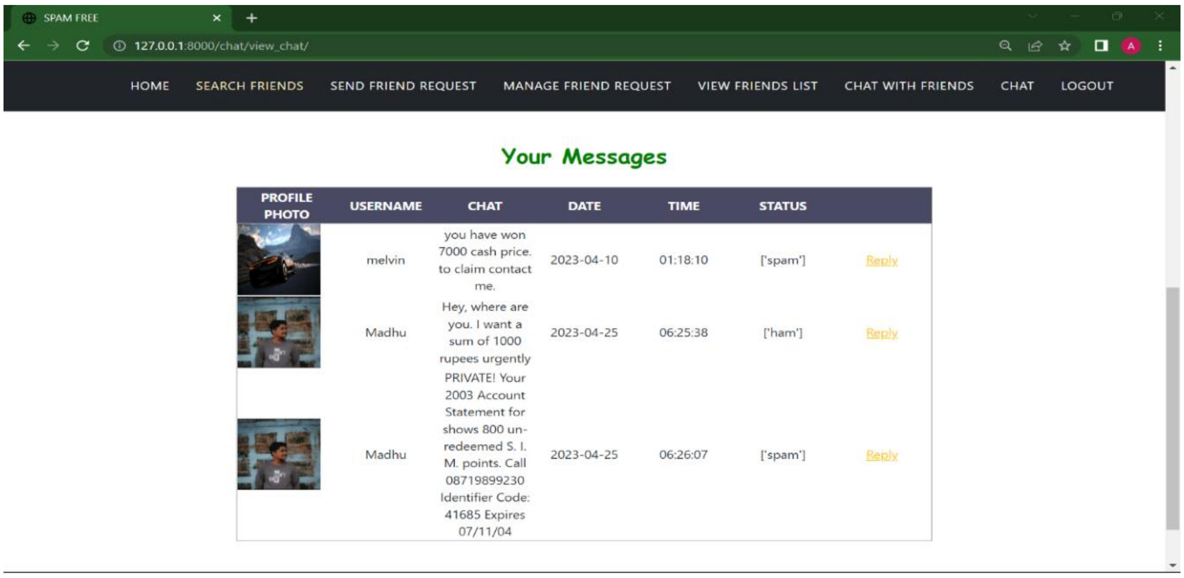

Distributed FS
Mapper
Mapper
Mapper
Reducer
Reducer
Distributed FS
tf_wd = wordCount_wd / wordsPerDoc_d
idf_wd = log(totalDocs / docsPerWord_w)
tf-idf_wd = tf_wd * idf_wd
Round 1 wordCount
{docId => [words]}
{[word, docId] => 1}
{[word, docId] => [1, 1 ...]}
{[word, docId] => wordCount}
Round 2 wordCount 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)

Fig 2: Equation for calculating TF-IDF

After vectorization the result is given to the implemented model. The decision tree model then checks which features from the current message match with the features of other messages in the dataset. Finally, the message is classified as ham or spam using the decision tree classifier, with the output being updated in the status field.

RESULT ANALYSIS

After evaluating the proposed model, an accuracy of 97.49% was obtained. This high level of accuracy indicates that the proposed model is effective, and therefore, it has been implemented into our chatting system. As a part of experimentation after implementing the model, we passed 3 inputs to test whether the model is able to check whether the message is SPAM or HAM. The Output were generated as follows:



Your Messages

PROFILE PHOTO	USERNAME	CHAT	DATE	TIME	STATUS	
	melvin	you have won 7000 cash price. to claim contact me.	2023-04-10	01:18:10	['spam']	Reply
	Madhu	Hey, where are you. I want a sum of 1000 rupees urgently	2023-04-25	06:25:38	['ham']	Reply
	Madhu	PRIVATE! Your 2003 Account Statement for shows 800 un-redeemed S. I. M. points. Call 08719899230 Identifier Code: 41685 Expires 07/11/04	2023-04-25	06:26:07	['spam']	Reply

Fig 3: Output

CONCLUSION

In conclusion, this project aimed to design and develop an online chatting website with spam message detection using machine learning techniques. We collected SMS data, preprocessed it, and extracted relevant features. We trained a Decision Tree Classifier model on the data and evaluated its performance. The trained model was then deployed to the website for real-time spam detection. We found that the model performed well in detecting spam messages with high accuracy of 97.49%. The website's backend successfully classified incoming messages as spam or ham based on the model's predictions, providing users with a safe and secure chatting environment with their friends. This project has practical applications in preventing unwanted spam messages from reaching users and maintaining the website's reputation. Even though there some limitations for this system. The accuracy of the model can be affected by the quality and quantity of the training data, and new types of spam messages that were not present in the training data may reduce the model's performance. Integration of more advanced natural language processing techniques, such as deep learning-based models, to improve the accuracy and efficiency of spam detection.

REFERENCES

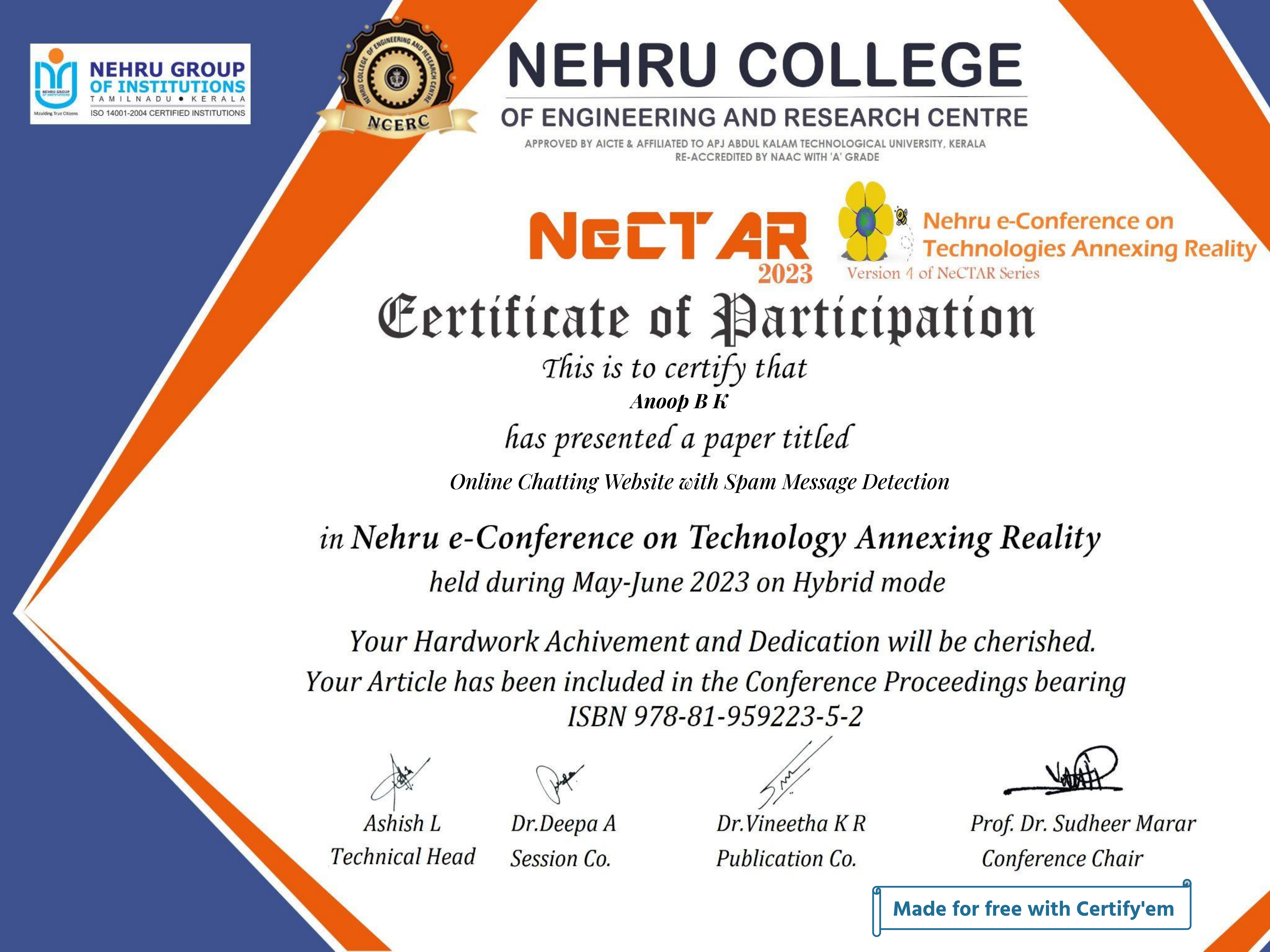
[1] Li, Z., & Yang, X. (2020). Content-based spam message detection using deep learning. Journal of Ambient Intelligence and Humanized Computing, 11(5), 1925-1936.

[2] Alzahrani, F., & Alshammari, R. (2019). A hybrid approach for spam detection in social media. Journal of Ambient Intelligence and Humanized Computing, 10(6), 2179-2191.

[3] Srivastava, R., Khare, N., & Singh, A. (2017). Performance comparison of machine learning algorithms for spam detection. International Journal of Computer Applications, 159(8), 25-29.

[4] Zhang, S., Wu, J., & Li, Y. (2018). A personalized spam message filter based on user preference. Journal of Ambient Intelligence and Humanized Computing, 9(6), 2059-2067.

[5] Luo, C., Chen, X., & Guo, X. (2019). A spam message detection method based on user behavior analysis. Journal of Ambient Intelligence and Humanized Computing, 10(3), 969-977.



NEHRU GROUP OF INSTITUTIONS
TAMILNADU • KERALA
Moulding True Citizens
ISO 14001-2004 CERTIFIED INSTITUTIONS

NCERC

NEHRU COLLEGE
OF ENGINEERING AND RESEARCH CENTRE

APPROVED BY AICTE & AFFILIATED TO APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY, KERALA
RE-ACCREDITED BY NAAC WITH 'A' GRADE

NeCTAR 2023
Nehru e-Conference on Technologies Annexing Reality
Version 4 of NeCTAR Series

Certificate of Participation

This is to certify that

Anoop B K

has presented a paper titled

Online Chatting Website with Spam Message Detection

in **Nehru e-Conference on Technology Annexing Reality**
held during May-June 2023 on Hybrid mode

Your Hardwork Achivement and Dedication will be cherished.
Your Article has been included in the Conference Proceedings bearing
ISBN 978-81-959223-5-2

Ashish L	Dr.Deepa A	Dr.Vineetha K R	Prof. Dr. Sudheer Marar
Technical Head	Session Co.	Publication Co.	Conference Chair