SMART EXPENSE TRACKER APPLICATION USING NAIVE BAYES

The Project submitted to

Sant Gadgebaba Amravati University, Amravati

Towards partial fulfilment of the Degree of

Bachelor of Engineering

In

Information Technology

Guided by

Dr. A S Manekar

Submitted by

Mr. Raj Thakare
Mr. Ninad Thakare
Mr. Raj Sangtani
Mr. Shubham Bondre

DEPARTMENT OF INFORMATION TECHNOLOGY
SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGINEERING, SHEGAON (M.S.)
2022- 2023

(02)

SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGINEERING, SHEGAON

EST.1983

2022-2023

CERTIFICATE

This is to certify that **Mr. Raj Thakare, Mr. Ninad Thakare, Mr. Raj Sangtani, Mr. Shubham Bondre** students of final year B.E. (Information Technology) in the year 2022-2023 of the Information Technology Department of this institute have completed the project work entitled "**Smart Expense Tracker Application Using Naive Bayes**" based on syllabus and has submitted a satisfactory account of his/her work in this report which is recommended for the partial fulfilment of the degree of Bachelor of Engineering in Information Technology.

[illegible]

Dr. A S Manekar
(Project Guide)

[illegible]

Dr. A S Manekar
Head of the Department
SSGMCE, Shegaon

[illegible]

Dr. S B Somani
Principal
SSGMCE, Shegaon

SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGINEERING, SHEGAON

2022-2023

<u>CERTIFICATE</u>

This is to certify that the project work entitled “**Smart Expense Tracker Application Using Naive Bayes**” submitted by **Mr. Raj Thakare, Mr. Ninad Thakare, Mr. Raj Sangtani, Mr. Shubham Bondre** students of final year B.E. (Information Technology) in the year 2022-2023 of the Information Technology Department of this institute, is a satisfactory account of his work based on the syllabus which is approved for the award of the degree of Bachelor of Engineering in Information Technology.

Internal Examiner	External Examiner
Date:	Date:

ACKNOWLEDGEMENT

The real spirit of achieving goals through the way of excellence and lustrous discipline. We would have never succeeded in completing our task without the cooperation, encouragement and help provided to us by various personalities.

*We would like to take this opportunity to express our heartfelt thanks to our guide **Dr. A.S.Manekar** for his esteemed guidance and encouragement, especially through difficult times. Her suggestions broaden our vision and guide us to succeed in this work. We are also very grateful for her guidance and comments while studying part of our project and learning many things under her leadership.*

*We would also like to extend our sincere thanks to **Prof. F.I.Khandwani**, Project-In-Charge for his valuable support and feedback during the entire course of the project.*

*We also extend our thanks to **Dr. A. S. Manekar**, Head of Information Technology Department, Shri Sant Gajanan Maharaj College of Engineering, Shegaon for providing us with a variety of opportunities and inspirations to gather professional knowledge and material that made us consistent performers.*

*We also extend our thanks to **Dr. S. B. Somani**, Principal, Shri Sant Gajanan Maharaj College of Engineering, Shegaon for providing us the infrastructure and facilities without which it was impossible to complete this work.*

Also, we would like to thank all teaching and non-teaching staff of the department for their encouragement, cooperation and help. Our greatest thanks to all those who wished us success, especially parents and friends.

Student Names

1. *Raj Thakare*
2. *Ninad Thakare*
3. *Raj Sangtani*
4. *Shubham Bondre*

ABSTRACT

In the era of this fast paced world, it is very important to take care of personal finance management. But keeping track of finanaces manually is not pracically possible and ideal. This is where Expense Tracker Application comes into picture. This android application helps users track their daily expenses, by allowing them to manually enter their expenses or automatically detect and read expense messages from the bank using a Naive Bayes based machine learning model.The proposed application has the potential to help people manage their finances better and make informed decisions about their spending habits. By automatically detecting and reading expense messages from the bank, the system can help users keep track of their expenses more accurately and efficiently. Expense Tracker allows users to add expenses manually with details like amount, category, date and description. And these expenses will shown with the help of pie charts so that users can have quick and clear picture of their expenses. Addtional functionalities such as adding goals or warning notification when spending limit exceeded are also there.

Keywords: Machine Learning; Personal Finance Management; Expense Tracking; Predictive Modeling; User Interface.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

1. INTRODUCTION

1.1 Preface

The "Expense Tracker" is a mobile application for android users. The app is developed using Kotlin and XML languages in android studio. "Expense Tracker" helps users to keep track of their daily expenses so that they can manage their finances properly and have complete idea of their spending habits. All the expenses are shown with help of Pie Chart and Bar Graph so that users get quick and clear idea of their spendings. Not only this app allows user to enter the expense manually along with amount, category and description of the expense, but also it can automatically detect the bank SMS from your mobile phone and read the credited or debited amount. Firebase is used as the online database for the app, so that even if you uninstall the app, still your data will be saved. To use the "Expense Tracker" app the user will have to sign up using email address and password. Text classification technique is used to detect Bank SMS accurately. In text classification a category is assigned to a document. SMS are text messages and we want to classify SMS as "Bank SMS" or "Normal SMS". We use supervised machine learning in text classification and that is why the labelled dataset is required. So, to classify Bank SMS we have used NB (Naive Bayes) supervised ML algorithm, which is mostly used in textual classification [1]. Also, in this study we analysed the performance of the NB (Naive Bayes) algorithm on the basis of “precision” and “accuracy”. Also compared performance of NB (Naive Bayes) with other ML algorithms such as SVC (Support Vector Classifier), LR (Logistic-Regression), ETC (Extra-Trees-Classifier), KNC (K-Neighbours-Classifier), RFC (Random-ForestClassifier) and DTC (Decision-Tree-Classifier).

1.2 Statement of Problem

In today's fast-paced world, managing personal expenses has become a crucial task. Many individuals struggle to keep track of their daily expenses, leading to financial instability and poor budgeting decisions [2]. Additionally, manually recording expenses is time-consuming and prone to errors, making it challenging to maintain accurate financial records. To address this issue, there is a need for an expense tracker application that utilizes machine learning to help users track their expenses efficiently. The application should enable users to record expenses manually or automatically and generate insightful charts to visualize their financial data. The objective is to create an intuitive and user-friendly interface that helps individuals manage their expenses effectively and make informed financial decisions.

1.3 Objectives of Project

The objectives for this project are to create expense tracker application which keeps track of user expenses automatically with the help of machine learning. And also user can add offline expenses manually with date, category, amount and description. User can set maximum spending limit and whenever the spending limit exceeded app will send the warning notification to notify user [3]. User statistics will be shown with help of charts which will help user get idea of his spendings in different areas, these stats are shown with different timelines such as weekly charts, monthly charts, yearly charts and all-time charts. The overall objective of this application is to help user in keeping track of his/her expenses and avoid the financial mistakes.

1.4 Scope and Limitations of the Project

i. **Scope:**

The expense tracker application will allow users to add expenses manually and automatically with the help of machine learning algorithms.

The application will have a user-friendly interface that enables easy tracking of expenses.

The application will provide detailed reports on expenses, including charts and graphs that show expense patterns and trends over time.

The application will allow users to set goals and alerts for specific expense limits to help them manage their spending.

ii. **Limitations:**

The application may not be suitable for users who prefer to track expenses using only manual system, as it relies heavily on automated tracking.

The accuracy of the automated categorization of expenses may be limited by the quality and quantity of data provided by the user.

The application may not be able to provide real-time updates on expenses, as some transactions may take time to be processed by financial institutions.

The machine learning algorithms may require substantial computational resources and may not work well on older or less powerful devices.

1.5 Organization of the Project

The project is organized as follows:

1. Chapter 1 gives Introduction about the project in which we discuss problem statement and solution to be given. It gives the objectives of project.
2. Chapter 2 gives Literature survey of the project. While making this report and developing this project we refer the list of literature.
3. Chapter 3 provides analysis of project. The hardware and software required for this project are analysed in this chapter. Along with that feasibility study, use case representations of the project.
4. Chapter 4 provides design phase of project in which we discuss the design goals, design strategies, various uml diagrams representation including class diagram, sequence diagram, collaboration diagram.
5. Chapter 5 provides how project is implemented in which we discuss the implementation strategies and testing strategies which we used to test our project module working.
6. Chapter 6 gives conclusion of the project.
7. Future Work of the project gives the brief of the new technologies which we can add in the future to make it more effective project.

2. LITERATURE SURVEY

Author in [1] proposes a developed email spam classification system using two popular machine learning methods, Naive Bayes Classifier and Support Vector Machine, to classify emails as spam or ham. The system focuses on the body or content of the emails to detect and filter out spam emails before entering the user's inbox. The machine learning approach has been shown to have better efficiency in email spam filtering. The system was tested and compared in terms of precision, recall, and F-measure performance metrics to find the best method. The results showed that Support Vector Machine had higher accuracy than Naive Bayes Classifier in different sizes of training emails. The study highlights the problem of spam emails, which causes significant losses to email users and servers. With the increasing usage of emails, the volume of spam or unwanted emails has also increased. An automatic email spam filtering system using machine learning algorithms is necessary to protect email users from malicious emails. The study concludes by suggesting future studies to improve the system's performance by considering richer email contents such as images, attachment files, links, routing, and meta information, and by applying other supervised learning algorithms.

Study [2] focuses on the classification of spam and not-spam emails using machine learning algorithms such as Naïve Bayes and Support Vector Machine (SVM). The aim is to compare the performance of both algorithms in terms of accuracy, precision, recall, and F-measure. The study highlights the importance of email communication and the prevalence of spam emails. The study also discusses the negative impact of spam emails on users, such as phishing attacks, financial loss, and inconvenience. The study compares two techniques implemented by other researchers, one using Naïve Bayes Algorithm and the other using SVM Algorithm. The results show that Naïve Bayes Algorithm has a higher accuracy rate when using Spam Data, but the precision rate is higher when using SPAMBASE. SVM Algorithm is recommended over Naïve Bayes Algorithm as it is one hundred percent precise in classifying non-spam emails in one case. However, the study acknowledges that no algorithm can classify both spam and not-spam emails without making a mistake.

Research study [3] discusses the problem of SMS spam and its potential dangers, such as data breaches and privacy invasion. The authors compare different machine learning techniques for detecting spam SMS, including traditional classifiers and deep learning methods. They evaluate the techniques on different datasets and measure their accuracy, precision, recall, and CAP curve. The authors find that the Convolutional Neural Network (CNN) Classifier achieves the highest accuracy of 99.19% and 98.25% for the two datasets, and they conclude that CNN has the potential for wider application in text-related classification problems. The authors also note that SMS spam detection can be challenging due to the limited availability of datasets and the informal language used in text messages. Despite these challenges, the authors believe that their work provides significant results that can be applied to real-world situations. This study contributes to the growing body of research on SMS spam detection, highlighting the need for effective filtering techniques to protect against cyber attacks and data breaches. The comparison of different machine learning techniques provides insights into the strengths and weaknesses of each approach and can inform future research on this topic.

The study [4] discusses the importance of automated management of emails, particularly in the context of spam classification using machine learning algorithms. With email being the most extensively used official communication mechanism, the volume of emails continues to grow, with more than 55% identified as spam. The study reviews and analyzes different machine learning techniques and email features used in various approaches for spam classification, providing insights into the current state of research, future research directions, and challenges in the field. The study finds that supervised machine learning approaches generate higher accuracy results with less variation and high consistency, with Naïve Bayes and SVM being in high demand compared to other algorithms. The use of multi-algorithm systems is more common for better outcomes. Future research opportunities lie in developing systems to detect spam using other email features beyond Bag-of-Words and Body text.

The study [5] presents a survey of existing email spam filtering systems that use machine learning techniques such as Naive Bayes, SVM, K-Nearest Neighbor, Bayes Additive Regression, KNN Tree, and rules. The study highlights the increasing popularity of email communication and the need for effective spam filtering due to the growing number of unsolicited messages. The various techniques used for filtering email spam include Knowledge-based technique, Clustering techniques, Learning-based technique, and Heuristic processes. The study provides a detailed classification, evaluation, and comparison of different email spam filtering systems and summarizes the overall scenario regarding the accuracy rate of different existing approaches. It explores several methods and concludes with an overview of several Spam Filtering techniques and summarizing the accuracy of different proposed approaches regarding several parameters. The study highlights the effectiveness of existing methods for email spam filtering but also acknowledges their shortcomings.

Authors in [6] discuss the issue of Twitter spam and how it has become a major problem in recent times. The authors focus on the use of machine learning techniques to detect spam tweets, which make use of statistical features of tweets. They identify the problem of Twitter spam drift, where the statistical properties of spam tweets change over time, making it difficult for existing classifiers to perform well. To address this issue, the authors propose the Lfun scheme, which can detect changed spam tweets from unlabelled tweets and incorporate them into the training process. The authors also discuss the need for protecting user privacy on social networking sites, especially on Twitter, where users interact with each other through tweets. They highlight the importance of detecting spamming activities, as spammers not only interfere with the privacy of users but also damage the internet as a whole.

Study [7] proposes a novel hybrid machine learning technique for email spam detection called GADT, which combines decision tree and genetic algorithm. The genetic algorithm is used to optimize the decision tree's performance by finding the optimal value for a parameter called the confidence factor, which controls the pruning of the decision tree. Additionally, principle component analysis (PCA) is used to reduce the high dimensionality of the feature vector and eliminate irrelevant noise, improving the performance of all text classifiers used for spam detection. The experimental results demonstrate that GADT outperforms traditional decision trees in terms of accuracy for detecting spam emails. The authors suggest that future work should focus on improving the time complexity of the proposed technique and exploring its potential for other applications beyond email spam detection. Overall, this study contributes to the ongoing research efforts in automatic text classification, specifically in the context of email spam detection, by presenting a new hybrid approach that improves the accuracy and efficiency of traditional machine learning techniques.

The study [8] presents an actual experiment implemented for the classification of Hindi poem documents into three classes, namely Shringar, Karuna, and Veera. The study discusses the challenges of text classification in Indic languages due to their morphological richness and the fusion of too much information in words. The experiment involved the manual collection of 122 documents from the web, preprocessing the data, extracting features using the Bag of Words Model, and converting those features into numeric representation for training the model. Five machine-learning classification algorithms were used for classification, and the model was tested with 20% of test data. Naïve Bayes and Random Forest algorithms performed better with 64% and 56% accuracy, respectively, as compared to other algorithms for Hindi Poem Classification. The study shows that document classification helps to assign labels to unlabeled documents, which is crucial for information management and easy retrieval.

Study [9] explores the use of machine learning and natural language processing techniques to classify occurrence reports in the aviation industry. The study discusses the challenges of analyzing the large volume of safety data collected through these reports and proposes a Random Forest algorithm for automatic text classification. The classifier was trained on the ICAO Occurrence Category and tested on over 45,000 reports, achieving an accuracy between 80-93%. The study highlights the importance of efficient categorization of reports for identifying safety trends and monitoring safety performance indicators. The authors suggest that future research should explore the use of multi-label classifiers for occurrence reports that fit multiple categories and analyze the possibilities of additional training data. The study concludes that the Random Forest classifier is a viable option for the classification of occurrence reports, and the use of data integration techniques could further improve the accuracy of the classifier. Overall, this study provides valuable insights for researchers and practitioners in the aviation industry who are interested in using machine learning techniques to analyze occurrence reports for improving safety.

The study [10] discusses the problem of email spam and the various techniques used to address this issue. Specifically, the Naive Bayesian classifier is used to classify spam and non-spam emails in the Ling spam dataset. The increasing number of spam emails is a major concern, as it affects the computer resources and bandwidth of email servers, and can lead to financial losses and fraud for users. Spammers collect data from emails, websites, and social media platforms, and use this information to send spam emails to unsuspecting users. To combat this issue, machine learning algorithms such as supervised and unsupervised learning are used to detect and filter spam emails. The study suggests that the use of deep learning and neural networks can also be explored in the future.

The research study [11] discusses the issue of spam emails and its negative impact on internet users. The authors state that email is essential for communication in today's world, but spam mail is a major problem for researchers to analyze and reduce. The study proposes the use of different classification techniques to identify spam mail and remove it. The Naive Bayes classifier and decision tree algorithms such as Random Tree, REPTree, Random Forest, and J48 are used to classify incoming emails as spam or ham. The UCI spambase dataset, which contains 58 attributes and 4601 instances, is used for analysis and implementation of results using Weka software. Feature selection is applied on the dataset for training set and cross-validation using Cfs Subset evaluation method. The results indicate that Random Tree classifier using the training set gives the best result for the classification of spam mail with 100% accuracy and only 0.01 second needed. The study concludes that different algorithms such as non-machine learning algorithms, dynamic algorithms, and hybrid algorithms can be used in the future to identify spam mail and give perfect results with measures of accuracy and efficiency. The study's findings have important implications for removing viruses, Trojans, malware, and websites, including phishing attacks and fraudulent attempts in emails, and it can benefit various fields such as education, banking, social networking sites, and offices.

The article [12] discusses the issue of spam categorization and the use of machine learning algorithms to differentiate spam from legitimate messages. The dataset used for the study contains 9324 records and 500 characteristics, and the optimal classification technique was determined using the Random Forest method. The article highlights the vulnerability of email and text messaging to abuse, and how spam has become a significant security issue in recent years, serving as a primary method for phishing critical information and distributing dangerous malware. The study compares the effectiveness of different classification techniques such as Neural Networks, Support Vector Machines, and Naive Bayesian, and concludes that Random Forests deliver greater accuracy. The article concludes by discussing the limitations of current spam filtering systems and the need for developing next-generation algorithms that can examine vast quantities of multimedia data and filter spam material more prominently. The Random Forests Technique achieves 92% average accuracy on the given dataset, making it a good choice for text classification. The study suggests that the application of Random Forests to cope with the problem of unbalanced classification in spam classification will be part of future research.

The study [13] information provided describes the growing trend of communication through short messages, specifically SMS. However, this trend has also led to an increase in spam messages, which can be harmful and result in cybercrime, theft, and fraud. To address this issue, machine learning algorithms such as Support Vector Machines (SVM) and Naïve Bayes have been used to detect spam messages from a tagged dataset. The study found that SVM outperformed Naïve Bayes with an accuracy of 94.32%. The article highlights the importance of detecting spam SMS to prevent cyber theft and protect the common man's interest. It also suggests that this model can be integrated into mobile phones as an alternative to identify received SMS as spam or ham. The study proposes the use of different datasets containing various types of spam messages to improve the classifier's performance and scalability. Finally, the study suggests using ensemble classifiers to further improve performance.

According to study [14] The above information discusses the importance of online media cooperation and how it is a major source of information for people today. However, with the rise of fake news, it has become essential to automate the process of detecting fake news on Twitter, which is one of the most popular sources of news. The study proposes a model for detecting fake news messages from Twitter posts by using machine learning algorithms such as Support Vector Machine, Naïve Bayes Method, Logistic Regression, and Recurrent Neural Network models. The experimental results show that SVM and Naïve Bayes classifier outperform the other algorithms. The study also highlights the importance of sentiment analysis in understanding people's opinions towards various topics and how it can be used in predicting stock prices and election results. The sentiment analysis of user-generated content can be useful in many applications such as product review analysis, customer interest mining, personalized recommendation, social marketing, and crisis management.

This study [15] proposes a model for identifying fake news on Twitter using machine learning algorithms such as Support Vector Machine, Naive Bayes Method, Logistic Regression, Recurrent Neural Network and Long Short Term Memory. The study aims to automate the detection of fake news to maintain a robust online media and social network. The dataset used in this research is expected to be utilized for machine learning-based statistical algorithms. The research found that SVM and Naive Bayes classifiers outperformed the other algorithms in terms of classification performance. The study highlights the increasing use of social media as a primary news source, which necessitates the need for strategies to filter out low-quality content. It also emphasizes that fake news spreads faster and more widely than genuine news, and the effects can be dangerous and horrifying. The study suggests that even basic algorithms in AI and machine learning can provide good results in detecting fake news, making it an effective tool in handling this critical issue. Overall, this study demonstrates the importance of identifying fake news in social media, and the need to develop automated models to improve credibility decisions.

The article [16] discusses the issue of Twitter spam and proposes a new set of features to improve spam detection mechanisms. The proposed features include both graph-based and tweet content-based features, and are evaluated using various machine learning classification algorithms, including k-Nearest Neighbor, Decision Tree, Naive Bayesian, Random Forest, Logistic Regression, Support Vector Machine, and eXtreme Gradient Boosting. The performance of these algorithms is compared based on different evaluation metrics, and the proposed approach is also compared with four latest state-of-the-art approaches. The results show that the proposed set of features provides better performance than existing state-of-the-art approaches, with an accuracy of 91%, precision of 92%, and F1-score of 91%. The study also highlights the limitations of using traditional account-related features, such as the number of followers/followings, for spam detection, as spammers can easily buy followers. The study provides insights into the nature of Twitter spam and proposes a more effective and robust approach for spam detection. In the future, the authors plan to extend their approach to other social networks and explore modifications to the machine learning algorithms.

The study [17] discusses the increasing problem of SMS and email spam and the need to secure systems from vulnerabilities. The authors use a SMS spam dataset from UCI Machine Learning repository and apply preprocessing and machine learning techniques such as Naive Bayes and Support Vector Machine to compute the performance of these algorithms. They propose a machine learning technique for SMS spam filtering based on Naive Bayes and SVM. The dataset used in their work consists of 5574 observations of two variables, the content of the messages and the target variable which is the class to be predicted (ham or spam). The authors highlight the frustration caused by SMS and email spam which not only impacts user performance but also leads to the consumption of longer resources, money, and network bandwidth. They also note the misclassification problem that arises when legitimate messages are blocked as spam. The study acknowledges the limitations of existing spam detection techniques and proposes a solution based on machine learning.

The work [18] focuses on the challenge of identifying and filtering spam emails using machine learning techniques. Traditional spam filtering methods such as black and white lists have become obsolete due to the increasing variety and volume of spam messages. The study evaluates the performance of two kernel functions, linear and Gaussian, for non-linear SVM-based classifiers on the SpamAssasin Public Corpus Dataset. The study compares the training and testing accuracy of these two kernels and identifies which kernel is more suitable for which dataset. The proposed method for email spam detection is capable of effectively identifying spam emails and blocking them while retaining genuine emails. The study concludes that the linear kernel-based scheme outperforms the Gaussian kernel in terms of accuracy and speed of operation. The study highlights that spam emails come in a variety of types, including advertisements, business promotions, objectionable services, and unrelated posts in Usenet groups, instant messaging platforms, and mobile phone messaging services. The decision tree classifier is not suitable for spam filtering due to the huge memory requirement, and the number of features/attributes for spam filtering can vary greatly. SVM is considered an important kernel method that can handle sparse data formats with a large number of features and acceptable recall and precision values.

The study [19] proposes a novel method for email spam detection using Support Vector Machine (SVM) and feature extraction. The study highlights the negative consequences of spam emails, including reduced productivity, wasted mailbox space and time, increased risk of viruses and malware, and potential harm to internet users. The proposed method achieved a high accuracy of 98% with test datasets, indicating its effectiveness in identifying and blocking spam emails. The study also discusses the different types of spam, including Usenet spam, instant messaging spam, mobile spam, and email spam. Usenet spam is the presenting of commercial advertisements on newsgroups, while instant messaging and mobile spam target users through messaging systems. The study identifies email spam as the most common form of spam and provides a comprehensive overview of its negative impacts on users. The proposed method can help reduce the negative consequences of email spam and improve the efficiency and effectiveness of email communication for internet users. Overall, the study offers valuable insights into the problem of email spam and provides a promising solution for detecting and blocking spam emails.

The study [20] discusses the problem of spam emails and proposes an integrated approach of Naïve Bayes (NB) algorithm and Particle Swarm Optimization (PSO) for email spam detection. The Naïve Bayes algorithm is used for learning and classification of email content as spam and non-spam, while the PSO algorithm optimizes the parameters of the NB approach. The Ling spam dataset is used for experimentation, and the performance is evaluated in terms of precision, recall, f-measure, and accuracy. The proposed approach outperforms the individual NB approach. The study provides a comprehensive literature survey of different techniques used by researchers for email spam classification, including machine learning-based and computational intelligence-based approaches. The study also discusses the properties of the Naïve Bayes and Particle Swarm Optimization algorithms used in the proposed approach. The results of the proposed approach show that it is an effective method for email spam detection, and the authors suggest future work on integrating NB with other swarm optimization-based concepts.

According to study [21] Email spam has become a major problem with the rapid growth of internet users. People are using spam emails for illegal and unethical conducts, phishing and fraud. Machine learning algorithms are being used for identifying spam mails which are fraud. The study discusses the machine learning algorithms and applies these algorithms on data sets to select the best algorithm for email spam detection with the best precision and accuracy. The study discusses the major approaches adopted for spam filtering including text analysis, white and blacklists of domain names, and community-based techniques. Naive Bayes is one of the most well-known algorithms applied in these procedures. However, rejecting sends based on content examination can be a challenging issue in the event of false positives. Ensemble methods have proven to be useful as they use multiple classifiers for class prediction. The study suggests that filtering of spams can be done on the basis of trusted and verified domain names, and the spam email classification is significant in categorizing e-mails and to distinguish e-mails that are spam or non-spam. The study concludes that there is a wide possibility of improvement in email spam detection, and subsequent improvements can be made by using the method for differentiating decent mails that are the emails they wish to obtain.

In work [22] use of email as a communication medium is increasing due to its effectiveness in saving time and money. However, spam emails are a major concern as they flood the internet with unwanted messages. Spam emails are used to send bulk emails to recipients who do not wish to receive them. The main challenge in spam classification is to accurately classify emails and unwanted threats. To address this challenge, researchers are working to find the best classifier to detect spam. In this study, the researchers analyzed different classification algorithms and found that incorporating feature selection approach into the classification process can significantly improve accuracy. Specifically, they used Naïve Bayes Classifier and word count algorithm to extract features and found that Naïve Bayesian Classifier produces better results than Support Vector Machine. Overall, this study provides insights into the use of data mining techniques for spam classification and can be useful for researchers and practitioners working in this area.

The Study [23] The increasing number of scientific publications has made it challenging to classify and organize them effectively. This study aims to analyze the performance of classification algorithms, namely Naïve Bayes (NB) and K-Nearest Neighbor (K-NN), on the Scopus dataset. The major issues in text classification are classification and feature extraction from the document using extracted features. The performances of classification algorithms are analyzed using Bayesian boost and bagging. The selected dataset is preprocessed and cleaned, and class imbalance issues are analyzed to increase the performance of text classification algorithms. The experimental results showed that K-NN performed better than NB, and boosting and bagging significantly increased the overall accuracy using NB. However, the performance of K-NN remained unchanged and better than NB. The study concluded that K-NN with bagging and boosting is a better classifier for classifying scientific publications.

The study [24] describes a proposed system for image classification and text extraction using machine learning techniques. Specifically, the system uses a Convolutional Neural Network (CNN) for image classification and Tesseract with Long Short-Term Memory (LSTM) for text extraction. The system has been shown to perform better than existing systems based on accuracy, due to the use of CNN which helps to overcome the problem of overfitting. The study also provides a brief overview of various supervised techniques for image classification, including Naive Bayes, K-Nearest Neighbor (KNN), Support Vector Machines (SVM), Decision Trees, Random Forests, and CNNs. It highlights the limitations of Naive Bayes, which assumes that features are independent of each other, whereas in reality, features may depend on each other. The proposed system can be further improved by incorporating a larger dataset and increasing the number of epochs. In addition, developing it into a full-stack application with a user interface for image uploading could make it more user-friendly. Overall, the study provides valuable insights into the use of machine learning techniques for image classification and text extraction.

This study [25] presents a study on hate speech target classification in Indonesian Twitter using machine learning techniques. The study aims to identify the targets of hate speech in social media to prevent its harmful impact, such as exclusion, discrimination, and violence. The study compares the performance of three classification algorithms (Naïve Bayes, SVM, and Random Forest Decision Tree) and two-term weighting schemes (Bag-of-Words and TF-IDF) using word n-grams as feature representations. The study found that SVM with TF-IDF and word unigram as features achieved the best F1-score of 0.84772 for hate speech target classification. The study concludes that SVM is the best algorithm for hate speech target classification, and word unigram is the best feature representation, while TF-IDF is the best term weighting scheme. The study also highlights the increasing prevalence of hate speech in social media, especially Twitter, in Indonesia, where 150 million people are active users. The National Human Rights Commission in Indonesia defines hate speech as actions based on hatred committed against individuals or groups through any means. The study suggests using deep learning for future studies to obtain better performance in hate speech target classification. However, increasing the size of the hate speech target dataset is necessary for effective implementation of deep learning.

The study [26] discusses the problem of unwanted spam emails and the need for more effective anti-spam filters. The authors propose a density-based clustering approach using the kNN algorithm for email classification. Relevant features are extracted from the study to act as an anti-spam filter, generating a successful corpus list for detecting spam emails. The proposed approach is compared to other methods on various email datasets, and the results show improved performance. The study also discusses the increasing use of email as a key tool for communication and the problem of unwanted spam emails. Ethical hacking techniques are proposed as a mechanism to identify spammers and their unsolicited emails. The study also mentions different types of spam emails, including image spam and malware spam. Text analysis is used to extract the most common spam words in a user's inbox for content-based analytics. The study proposes a framework for improving clustering quality by selecting correct attributes and removing redundant information. Finally, the authors show that the kNN classification approach improves the ability to classify datasets compared to other methods.

The work [27] discusses the increasing usage of email communication, particularly in the Urdu language, and the corresponding rise in spam content, which poses a security threat to users. The authors propose using machine learning algorithms, including Naive Bayes, CNN, SVM, and LSTM, to detect and categorize spam email content in Urdu. They generate a translated email dataset containing spam and ham emails from Kaggle and preprocess it for analysis. The study finds that deep learning models, specifically the LSTM model, outperform other models, achieving a high accuracy score of 98.4%. The authors highlight the importance of detecting and filtering spam emails, as they contain fake content and links for phishing attacks and other threats, with the intention of stealing personal information and using it for materialistic gain. Despite the availability of anti-spam services, there is still no definitive way to distinguish between legitimate and malicious emails due to the ever-changing content of such emails. The study concludes that deep learning models are more successful in classifying Urdu spam emails and can be used as an automated way to detect such emails.

The study [28] the systematic literature review discussed in this study aims to provide an overview of existing research on SMS spam detection techniques. SMS is a widely used messaging medium and has become a platform for advertising and marketing, including SMS marketing. However, spam SMS can be a cause of concern for users, and their detection is crucial for preventing damage. The review analyzed 13 research studys and reviewed their techniques, approaches, algorithms, advantages and disadvantages, evaluation measures, and datasets used. The study highlighted that existing research on SMS spam detection has not addressed challenges like the use of local contents, shortened words, and incomplete slogan information. Therefore, there is a significant scope for future research in this field. The study also discussed various categories of SMS spam filtering, such as white-list and black-list, content-based, non-content-based, and challenge-response techniques. The publicly available dataset information was also highlighted as a prior need for a spam filtering algorithm. Overall, the review can serve as a reference for the upcoming direction of research in this field.

The article [29] discusses the challenging task of multi-label classification in the Bengali language, which has limited resources. The authors propose a solution utilizing a large dataset of over 400,000 news articles from a popular Bengali newsstudy, categorized into six categories. They use the ML-KNN algorithm and Neural Network for supervised learning and Count Vectorizer for word embedding. The authors perform extensive statistical analysis and experiments on a small scale to analyze the impact of the feature vector size, words per document, and the number of labels on the model's performance. They compare the results of both approaches and suggest future directions, including exploring different word embedding techniques and considering unsupervised or semi-supervised learning for more feasible and cost-effective solutions. This research contributes to the less explored research domain of Bengali language processing and provides a sophisticated and standard solution for multi-label classification, which can be utilized by Bengali newsstudy portals to improve their recommendation system and reduce manual labor. Overall, this study presents a valuable contribution to the field of natural language processing and text mining for the Bengali language.

The research study of [30] discusses the task of text classification, which involves assigning predefined class labels to text documents. The authors analyze the performance of various machine learning and deep learning algorithms for text classification on the 20 newsgroups dataset. They use three different vectorization techniques, including Bag of Words (BOW), TF-IDF, and word embeddings, to preprocess the text data. The study finds that Logistic Regression outperforms other machine learning algorithms, while a bi-channel Convolutional Neural Network (CNN) model produces the best results among deep learning models. Additionally, the authors observe that TF-IDF is the best method for vectorization compared to word embeddings like Word2Vec and BERT.

The work of [31] focuses on the problem of email spam detection and proposes the use of machine learning algorithms to build an effective model for classifying whether an email is spam or not. The authors use the UCI Machine Learning Repository Spambase Data Set to experiment with five popular machine learning classification algorithms: Logistic Regression, Decision Tree, Naïve Bayes, KNN, and SVM. The performance of each algorithm is evaluated based on various metrics, such as accuracy, precision, recall, and F1-score. The authors highlight the importance of preventing email spam, as it poses serious threats to internet users by causing security vulnerabilities and potentially revealing users' identities or destroying vital information. While knowledge engineering methods for spam classification are available, they require constant updating of rules, making them difficult to maintain. Machine learning approaches, on the other hand, can analyze large amounts of data more efficiently and accurately, making them a suitable solution for email spam detection.

The study [32] analyzes and compares the relative strengths of various machine learning algorithms in detecting spam messages on mobile devices. The authors acquired data from an open public dataset and prepared two datasets for testing and validation purposes. The accuracy in detecting spam messages was the first priority in ranking these algorithms. The study shows that different machine learning algorithms under different features tend to perform differently in classifying spam messages. Naïve Bayes algorithm outperformed Random Forest and Logistic Regression algorithms with a high accuracy of 98.445% just with the information gain matrix and easily classified the text as either spam/non-spam. The authors proposed some feature sets that could be utilized by different machine learning algorithms in order to classify those text messages. The study also discusses the problem of mobile spam messages, which has been a consistent problem in Far East countries since 2001, and the legal and technical measures that can be taken to control this widespread disturbing abuse of mobile spam messages. The study concludes that technical and legal measures need to be taken in order to control this problem and that Bayesian filters have been the most prominent and widely accepted method for discriminating and classifying legitimate messages from the pile of normal and SMS spam messages with a clever use of machine learning algorithms.

According to work [33] the growth of short message services has led to an increase in spam messages, which are irritating for consumers and pose a challenge for spam detection techniques. Most research in this field has focused on manually found features, but machine learning techniques are now being used to automatically process spam SMS messages. The goal of this study is to develop an accurate and responsive classification model to differentiate between ham and spam messages, with a low false positive rate. Spam detection is a relatively new issue for mobile phone users, and poses a threat to both individuals and businesses. Existing spam detection methods face challenges in distinguishing between spam and legitimate messages, such as the use of regional and shortcut words. The study highlights the scope for further research in this field, with the aim of improving accuracy and minimizing the number of deciding features and factors. The authors conducted a literature survey and found that while many research studys have used naive Bayes theorem for spam detection, they have low validation scores and accuracy. They tried a Python library called EVALML and found that logistic regression pipeline provided higher accuracy and validation score. Overall, the study suggests that more research is needed to address the challenges posed by spam messages and improve the accuracy of spam detection techniques.

In work [34] the prevalence of SMS spam messages has increased with the development of cell phone clients. To address this issue, accurate and effective spam solutions are needed. In this literature survey, the authors propose treating SMS spam detection as a two-class document classification problem and compare the performance of different classifiers, including K Nearest Neighbour, Naïve Bayes, and Decision Tree. Due to the limited dataset available for SMS spam filtering, feature extraction and classification algorithms were used to improve performance while minimizing computational requirements. The study found that Naïve Bayes classifier showed the highest accuracy among others classifier. However, future work is needed to further improve feature selection and add more relevant features such as length limits and learning curves. Overall, the proposed techniques can be used to develop an application for mobile phones to protect against spam messages in the future.

The study of [35] presents a study on detecting spam and ham messages in SMS using various supervised machine learning algorithms such as naïve Bayes, support vector machines, and maximum entropy. The study aims to compare the performance of these algorithms in filtering out spam and ham messages, given the rising prevalence of SMS spam as people engage more in web-based activities and share their private data with different companies. The authors found that the support vector machine algorithm gave the most accurate results compared to the other two algorithms tested. The study suggests that developing a classification algorithm that filters SMS spam would be a useful tool for mobile phone providers, as SMS spam can be particularly bothersome for users who often pay a fee for each SMS received. However, SMS spam presents additional challenges for automated filters, as messages are often limited to 160 characters, and users tend to use shorthand notations and slang, making it difficult to distinguish between ham and spam.

The work [36] discusses the problem of fake online reviews and the need for effective methods to detect and eliminate them. The authors propose an AI-based approach that considers both the features of the reviews and the behavior of the reviewers to detect fake reviews. They use the Yelp dataset to evaluate the proposed approach and apply various techniques from KNN. The importance of online reviews is highlighted, as they are often the primary source of information for customers when making purchasing decisions. However, fake reviews are deliberately written to manipulate customers and build a false reputation. The study emphasizes the need for ongoing research in detecting fake reviews, which depends not only on the content but also on the behavior of the reviewers. The authors suggest that machine learning can provide a significant contribution to identifying fake reviews, particularly through web mining techniques such as sentiment analysis. They note that the development of meaningful features extraction for reviewers is crucial for successful fake reviews detection. Overall, the study provides insights into the problem of fake online reviews and presents a novel approach to detecting them using AI-based techniques.

The article [37] discusses the increasing importance of opinion mining in the digital world, where user reviews and comments play a crucial role in evaluating products, services, and policies. However, spammers take advantage of this trend and flood the internet with spam messages that misguide users. The study focuses on using a Naive Bayes classifier to efficiently detect spam reviews by training words and analyzing further sentences. The study highlights the need to accurately detect spam messages, which can contain malicious links, fraudulent reviews, insults, blackmail letters, fake jobs, viruses, and personal information. The spam messages are free of cost and time, and their bulk inundation affects the performance and accuracy of the system. The study proposes a spam classification system that identifies spam and non-spam messages using the Naive Bayesian Classifier and word-count algorithm. The results show that the Naive Bayesian Classifier has a low error rate and more accurate results. The study concludes that spam classification is a crucial matter in social networks, and accurate detection of spam messages is necessary to improve the quality of public opinions and prevent spam-related problems.

The study [38] discusses the issue of spam SMS that has become a problem for mobile phone users and how it can lead to the skipping of important and genuine messages, as well as making users vulnerable to phishing and fraud. The study proposes the use of Naive Bayes algorithm for detecting spam SMS, which is a supervised machine learning algorithm. The algorithm is implemented on a dataset and its performance on detecting spam SMS on test data is shown. The study highlights the importance of separating spam SMS from genuine SMS to avoid potential cyber attacks and leakage of confidential information stored in mobile phones. Text classification techniques are widely used for spam filtering, and in this case, supervised machine learning approach is used, which requires already labeled dataset for constructing a classifier. The study notes the similarity in email spam filtration and SMS spam filtration and suggests that techniques used for spam email filtration can be used for spam SMS filtration. The Naïve Bayes algorithm is popularly used for spam email filtration and the study uses it for spam SMS detection. The results of the implementation of the algorithm on the test dataset are presented in a confusion matrix and statistics, which show that the algorithm correctly classified most of the ham and spam messages, with only a small number of wrongly classified messages. Overall, the study provides useful insights and a potential solution to the problem of spam SMS.

The authors of [39] discuss the problem of spam on Twitter, which includes the spread of malicious software, fake URLs, and pornography advertisements. The authors note that Twitter users are experiencing data stealing malware by accessing or visiting unnecessary spam messages or tweets, and it has become a significant concern. Additionally, they discuss the use of keywords by digital marketers to manipulate Twitter trends, leading to the spread of unwanted information, which is considered spam. The study emphasizes the need for spam control measures to suppress the actions of spammers. The authors review several studies that have been conducted to detect spammers in social networks, and they conclude that combining multiple features for spam detection leads to better performance. Finally, the study notes that Twitter uses API for mobile application with several spam filters that use machine learning algorithms to control spam, but the accuracy and performance of these filters depend on the training and algorithm efficiency. Overall, the study highlights the importance of spam detection and suppression to ensure a safe and pleasant user experience on Twitter.

Authors in [40] discuss that the exponential growth of unstructured textual data has led to the need for automated classification of documents into predefined categories. Text classification has become an important area of research, with many different approaches being explored. The dominant approach is machine learning, which offers efficiency, accuracy, performance, and usability to different domains. K-Nearest Neighbors (KNN) is a popular and efficient algorithm for text classification, but finding the optimal value of k, which represents the number of neighbors, can be a challenging task. This study presents an approach for building a machine learning system in R that uses KNN for the classification of textual documents. The study highlights the importance of high accuracy in government documents, as they improve the quality of provision of public services to the population. The future work will be devoted to the large amount of documents by integrating R with Hadoop.

3. ANALYSIS

3.1 Detailed Statement of the Problem

Expense tracking is a crucial aspect of personal finance management. It helps individuals and businesses keep track of their spending habits, identify areas of overspending, and make informed financial decisions. However, keeping track of expenses can be time-consuming and challenging, especially if done manually. Therefore, there is a need for an expense tracker application that can automate the process and provide users with real-time insights into their spending. The proposed expense tracker application aims to provide users with a user-friendly interface that simplifies the process of tracking expenses. The application will enable users to categorize their expenses, set budgets, and receive notifications when they exceed their spending limits.

The application will also provide users with graphical representations of their spending habits, making it easier to identify areas where they can cut back on expenses. The primary challenge in developing an expense tracker application is designing a user-friendly interface that is easy to navigate. The application should also be secure, with features such as two-factor authentication and encryption to ensure that user data is protected. Overall, the expense tracker application will address the need for a convenient and automated way to track expenses, providing users with real-time insights into their spending habits and helping them make informed financial decisions. The problem that an expense tracker application for common people is trying to solve is the difficulty in managing personal finances. Many people struggle with keeping track of their expenses, sticking to a budget, and planning for their financial future. This can lead to overspending, debt, and financial stress. The lack of a comprehensive and effective system for tracking expenses can have several negative consequences [4]. People may not realize how much they are spending, which can make it difficult to make ends meet. Additionally, not tracking expenses can make it difficult to plan for future expenses, such as saving for a vacation or a down payment on a home.

The application should make it easy for users to input their expenses quickly and easily. Also user should be able to input input more details such as amount, category (for example food, education, travel, sport etc), date of the expense, description (short detail about the expense which can help later for understanding).

The application should allow user to set goal that he want to achieve in future and user should add goal which details like amount required to achieve the goal, the date by which user want to achieve it , and a short description about the goal. Expense tracker should allow user to set the maximum monthly spending limit which if exceeded then the user will be notified by sending a warning notification.

User should be able to authenticate into the app using email and password and this authentication will be implemented with the help of firebase authentication service. To store the data of the user we are using firebase realtime database which is very fast, so whenever user deletes or uninstalls the app, if he again log in using his details all of user’s details will be available.

Charts and graphs provide a visual representation of data, making it easier for users to understand and interpret information [6]. However, creating charts and graphs manually can be time-consuming and require specialized skills, such as data visualization and graphic design. An application that shows user data with the help of charts is designed to help users better understand and interpret their data. By addressing the key challenges of easy data import, chart customization, data filtering, real-time updates, sharing and collaboration, and security, the application will provide users with a comprehensive and effective system for visualizing and analyzing their data. This can help users identify trends, patterns, and insights that might otherwise be difficult to see, and make more informed decisions based on their data.

3.2 Requirement Specifications

Requirement Specifications for smart expense tracker application is given as follows with explanation:

i. User Authentication: The application should allow users to create an account by providing their email address and password. The authentication process should be secure and protect users' personal information.

ii. Expense Tracking: The application should allow users to manually enter their expenses with relevant details like category, description, and amount. Additionally, the app should be able to automatically detect and read bank SMS messages to track expenses.

iii. Data Visualization: The application should show all expenses in an easily understandable graphical representation [5]. The graphical representation should be in the form of a pie chart or a bar graph, which gives the users an idea of their spending habits and patterns. Categorization: The application should have pre-defined categories of expenses, but it should also allow users to create their own categories for better expense tracking.

iv. Online Database: The application should have an online database to store users' expense data securely. The database should also have the ability to synchronize data across multiple devices. Data Security: The application should ensure that users' data is secure and protected from unauthorized access. All communication between the application and the server should be encrypted.

v. Notification System: The application should have a notification system that informs users about their daily, weekly or monthly expenses.

vi. Budgeting: The application should allow users to set a budget for each category of expenses. The application should notify users when they are about to exceed their budget.

vii. Machine Learning Algorithm: The application should use a Naive Bayes algorithm for text classification to accurately detect bank SMS messages [37]. The algorithm should be trained using supervised machine learning to ensure high accuracy and precision.

viii. Performance: The application should be able to handle a large amount of data and provide real-time analysis of users' expenses. The application should also be responsive and fast to provide a seamless user experience.

ix. Compatibility: The application should be compatible with Android devices of different screen sizes and operating systems.

x. User-friendly Interface: The application should have an easy-to-use interface that is intuitive and easy to navigate. The application should be designed to provide a user-friendly experience to the users. Overall, the proposed expense tracker application should be able to help users manage their finances better by allowing them to track their expenses accurately, visualize their spending habits, and make informed decisions about their spending.

3.3 Functional Requirements

The proposed system should be able to perform the following functions:

i. The system should be able to detect Bank SMS and read user expenses automatically.
ii. The system should allow users to manually enter their expenses along with the amount, category, and description of the expense.
iii. The system should show all the expenses with the help of Pie Chart and Bar Graph to give users a clear idea of their spending habits.
iv. The system should allow users to sign up using an email address and password.
v. The system should use Firebase as the online database to store user data.
vi. The system should be able to categorize expenses based on user preferences.
vii. The system should be able to generate monthly or weekly expense reports.

3.4 Non-Functional Requirements

The proposed system should have the following non-functional requirements:

i. The system should have a simple and user-friendly interface.
ii. The system should be highly accurate in detecting Bank SMS and categorizing expenses.
iii. The system should be able to process a large volume of data.
iv. The system should be able to work offline and sync data when the user goes online.
v. The system should be highly secure and protect user data from unauthorized access.
vi. The system should be highly available and provide uninterrupted service to users.

3.5 Performance Requirements

The proposed system should have the following performance requirements:

i. The system should be able to detect Bank SMS with a high degree of accuracy, preferably above 95%.
ii. The system should be able to classify expenses accurately, preferably with a precision and accuracy of above 90%.

iii. The system should be able to handle large amounts of data and provide fast response times to users.
iv. The system should be able to generate expense reports quickly and efficiently.
v. The system should be able to handle concurrent user requests without compromising performance.

3.6 Usability Requirements

The proposed system should have the following usability requirements:

i. The system should have a simple and intuitive user interface that is easy to navigate.
ii. The system should be able to provide users with clear and concise information about their expenses.
iii. The system should provide users with real-time updates on their expenses.
iv. The system should provide users with personalized recommendations on how to save money and manage their expenses [8].
v. The system should provide users with the ability to customize the app to suit their individual preferences and needs.

In summary, the proposed Expense Tracker system should be able to accurately detect Bank SMS, classify expenses, and provide users with real-time updates on their expenses. The system should be highly accurate, secure, and user-friendly, with fast response times and the ability to handle large amounts of data [9]. The system should also be compatible with a range of mobile devices and networks and provide users with personalized recommendations and the ability to customize the app to suit their individual needs.

3.7 Feasibility Study

Feasibility study of expense tracker application is explained as follows with proper explanation for various types of feasibility studies.

i. **Technical Feasibility:** The proposed expense tracker application is built using Kotlin and XML languages in Android Studio, which are widely used and have a vast community of developers [11]. The Android platform also provides a comprehensive set of APIs that allow developers to build complex applications with ease. Additionally, the application uses Firebase as the online database, which is a reliable and scalable cloud-based database service [10]. The Naive Bayes algorithm used in the application is a widely-used machine learning algorithm for text classification, which is also technically feasible to implement.

ii. **Operational Feasibility:** The proposed expense tracker application aims to help users manage their finances better by allowing them to track their expenses. The application offers several features such as manual expense entry, automatic detection and reading of bank SMS messages, and visualization of expenses through pie charts and bar graphs [12]. These features make the application operationally feasible as it saves users time and effort in managing their finances.

iii. **Economic Feasibility:** The proposed expense tracker application is free to use, which makes it economically feasible for users. However, the application may require ongoing server and database maintenance, which may require some financial investment. The cost of maintaining the application can be offset by using monetization strategies such as in-app advertisements or premium features.

iv. **Legal Feasibility:** The proposed expense tracker application is compliant with data protection laws and regulations as it requires users to sign up using email address and password. The application does not store any sensitive financial information such as bank account details or credit card numbers. The Naive Bayes algorithm used in the application is a widely-used machine learning algorithm and does not violate any legal rights.

In conclusion, the proposed expense tracker application using Naive Bayes algorithm is technically, operationally, economically, and legally feasible [13]. The application has the potential to help users manage their finances better and make informed decisions about their spending habits. However, it is important to note that the success of the application depends on its adoption rate and user engagement.

3.8 Use Case Diagram

The figure 3.1 is a use case diagram for the expense tracker android app which involves two main actors: the user and the system [14]. The user is the person who uses the app to track their expenses, and the system is the app itself. There are three primary use cases involved in the app:

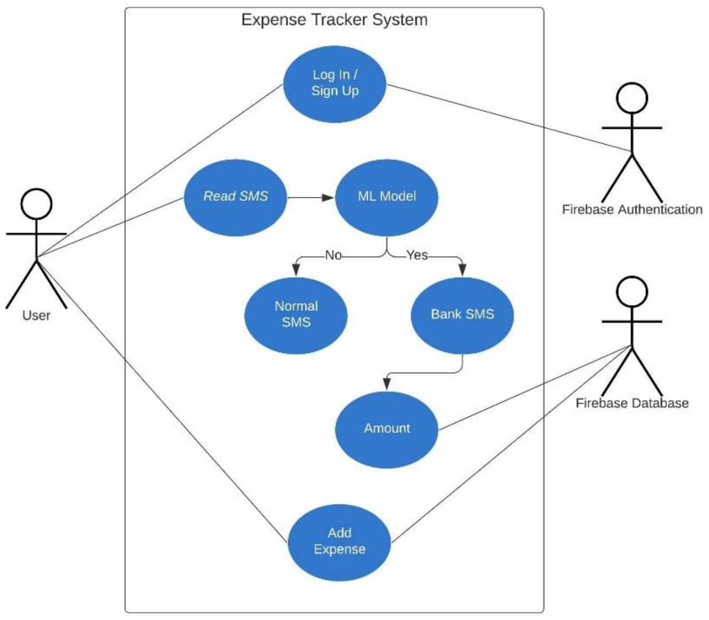


Fig 3.1 Use Case Diagram

i. **Login/Signup using Firebase Authentication:** The first use case involves the user logging into the app or signing up for a new account. The app uses Firebase Authentication to handle user authentication. When the user tries to log in, they will be prompted to enter their credentials [15]. The app will then authenticate the user's identity by checking with Firebase Authentication if the entered credentials match an existing user [16]. If there is a match, the user is logged in, and the app proceeds to the next screen. If not, the app displays an error message.

ii. **Add Expense and store on Firebase Realtime Database:** The second use case involves the user adding a new expense to the app, with the amount of the expense, category of expense, date of the expense and the description [18]. And this will be stored in Firebase Realtime Database.

iii. **Automatically Read Expenses:** The app first reads all SMS messages on the user's device. The app then makes an API call to an ML model to determine if the SMS message is related to an expense or not [17]. If the ML model confirms that the SMS message is an expense, the app creates a new expense object and stores it in Firebase Realtime Database.

The expense object will contain the following fields:

a. **Description:** A brief description of the expense.
b. **Amount:** The total cost of the expense.
c. **Date:** The date the expense was incurred.
d. **Category:** The category to which the expense belongs (e.g., food, transportation, entertainment).

Once the expense object is stored in Firebase Realtime Database, it can be retrieved and displayed to the user in various ways, such as a list or a chart.

Overall, the use case diagram illustrates how the app uses Firebase Authentication and Firebase Realtime Database to provide users with an easy way to track their expenses [19]. The app reads SMS messages to automatically detect expenses and then stores them in Firebase Realtime Database for easy access and analysis.

3.9 Use Case Specifications

Use case specifications for the expense tracker application:

i. **Use Case Name:** Login/Signup with Firebase Authentication

Primary Actor: User

Goal in Context: The user wants to login or signup to the app using Firebase Authentication [20].

Preconditions:

The user has installed the app on their Android device.

The user has an active internet connection.

The user has not already logged in.

Trigger: The user taps on the login/signup button.

Scenario:

The system displays the login/signup screen.

The user enters their email and password or chooses to sign up.

The system validates the input and makes an API call to Firebase Authentication.

Firebase Authentication verifies the user's credentials and sends a response to the system.

The system logs in the user and displays the main screen of the app.

Postconditions:

The user is logged in and can use the app's features.

Alternate Flows:

If the user enters invalid login credentials, the system displays an error message and prompts the user to try again [21].

If the user chooses to sign up, the system prompts the user to enter their details and create a new account.

ii. **Use Case Name:** Add Expense

Primary Actor: User

Goal in Context: The user wants to add a new expense to their list.

Preconditions:

The user has logged in to the app.

Trigger:

The user taps on the "Add Expense" button.

Scenario:

The system displays the "Add Expense" screen.

The user enters the details of the expense, such as the amount, category, and date.

The system validates the input and creates a new expense object.

The system stores the expense object in the Firebase database.

The system updates the user's expense list with the new expense.

The system displays a confirmation message to the user.

Postconditions:

The user's expense list is updated with the new expense.

Alternate Flows:

If the user enters invalid input, the system displays an error message and prompts the user to try again.

iii. **Use Case Name:** ML Model Expense Creation

Primary Actor: System

Goal in Context:

The system wants to use an ML model to automatically create an expense based on an SMS message.

Preconditions:

The user has given permission for the app to access their SMS messages.

The user has not disabled the ML model feature.

Trigger: The system receives a new SMS message.

Scenario:

The system reads the content of the SMS message.

The system makes an API call to the ML model with the SMS message content.

The ML model analyzes the message and determines if it is related to an expense.

If the ML model determines that the message is related to an expense, the system creates a new expense object.

The system stores the expense object in the Firebase database.

The system updates the user's expense list with the new expense.

The system displays a notification to the user informing them of the new expense.

Postconditions:

A new expense object is created and added to the user's expense list.

Alternate Flows:

If the ML model determines that the message is not related to an expense, the system does not create a new expense object.

iv. **Use Case Name:** View Expense List

Primary Actor: User

Goal in Context: The user wants to view their list of expenses.

Preconditions:

The user has logged in to the app.

The user has added at least one expense.

Trigger: The user navigates to the expense list screen.

Scenario:

The system displays the user's list of expenses.

The user can scroll through the list and view each expense's details, such as the amount, category, and date.

The user can choose to edit or delete an expense by tapping on the expense item.

If the user chooses to edit an expense, the system displays the "Edit Expense" screen and allows the user to make changes to the expense details.

If the user chooses to delete an expense, the system removes the expense from the user's expense list and Firebase database.

Postconditions:

The user can view their list of expenses and make changes as needed.

Alternate Flows:

If the user has not added any expenses yet, the system displays a message informing the user to add expenses to see the list.

4. DESIGN

4.1 Design Goals

Design goals are an essential aspect of any project as they provide a clear roadmap for the project's development and help ensure that the final product meets the needs and expectations of its users. Here are some design goals for smart expense tracker application explained below:

i. **Guide project development:** Design goals provide a clear direction for the development team, outlining what needs to be achieved and how it should be achieved. This helps the team stay focused and on track throughout the development process, reducing the risk of scope creep or unnecessary features being added.

ii. **Ensure user needs are met:** By defining design goals that are focused on user needs, the development team can ensure that the final product meets the needs of its intended audience. This can help increase user satisfaction and engagement with the product, leading to greater success and adoption rates.

iii. **Enhance usability and functionality:** Design goals can help guide decisions about the product's usability and functionality, ensuring that the final product is intuitive, easy to use, and provides the necessary functionality required by its users.

iv. **Increase efficiency and productivity:** By defining design goals that prioritize efficiency and productivity, the development team can create a product that streamlines workflows and reduces time and resource consumption, ultimately leading to a more effective and efficient product.

v. **Facilitate communication and collaboration:** Design goals provide a common language and understanding for the development team, making it easier for team members to communicate and collaborate effectively [22]. This can help reduce misunderstandings and ensure that everyone is working towards the same end goal.

In summary, design goals are an essential aspect of any project as they help guide development, ensure user needs are met, enhance usability and functionality, increase efficiency and productivity, and facilitate communication and collaboration. By defining clear design goals, the development team can create a product that is both successful and impactful.

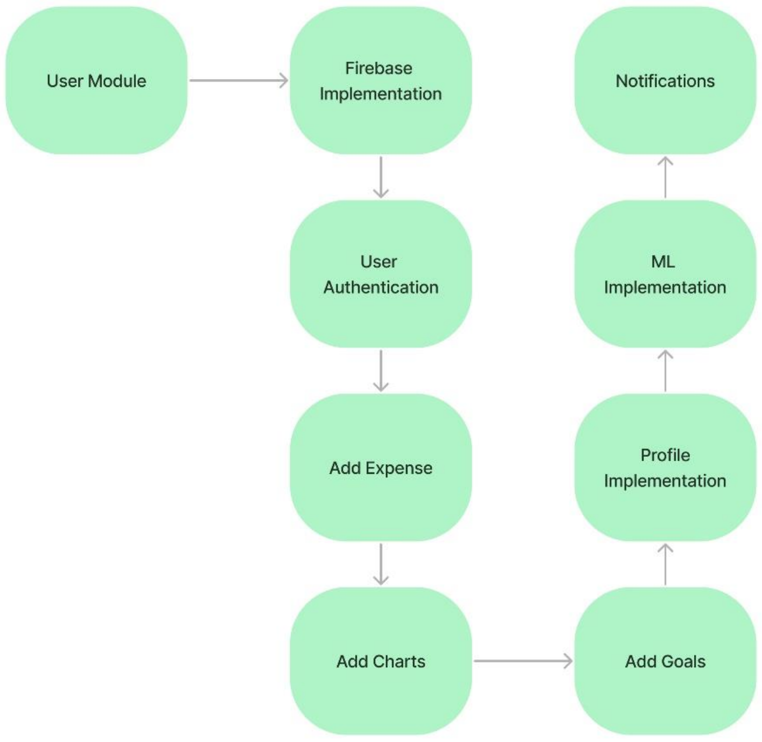


Fig 4.1 Design Goals Diagram

Figure 4.1 shows design goals diagram for smart expense tracker application and has the following elements:

i. User Module: The first step is to create a user module, which allows users to register, log in, and manage their account information. This module can be implemented using Firebase Authentication, which provides a secure and easy-to-use authentication system that supports multiple authentication methods, such as email/password.

ii. Firebase Implementation: After creating a user module, the next step is to implement Firebase into the application. Firebase is a mobile and web application development platform that provides a range of tools and services, including hosting, database, storage, and authentication [23]. Firebase can be used to store and manage user data, as well as to provide real-time updates and notifications.

iii. User Authentication: Once the Firebase implementation is complete, the next step is to add user authentication to the application. This involves adding login and registration screens, as well as implementing Firebase Authentication to handle user authentication.

iv. Add Expense: The next step is to add the ability for users to add expenses to the application. This can be implemented by adding a screen where users can enter the details of their expenses, such as the amount, category, date, and description. This expense will be saved on firebase database.

v. Add Charts: After implementing the ability to add expenses, the next step is to add charts to the application. This allows users to visualize their expenses and gain insights into their spending habits. Charts can be implemented using libraries such anychart.

vi. Add Goals: Once the basic functionality of the application is in place, the next step is to add goals. Goals allow users to set targets for their spending and track their progress towards those targets. Goals can be implemented by adding a screen where users can set their goals and a dashboard that displays their progress towards those goals.

vii. Profile Implementation: After implementing the main features of the application, the next step is to implement a user profile. This allows users to view and manage their account information, as well as to customize their preferences and settings. User can add name, salary , max monthly spending limit etc.

viii. ML Implementation: Machine learning is used to read all device messages and then feed them to api of machine learning model [24]. this machine learning model will give us bank related messages. And then with the help of programming libraries we will extract amount from the message [25].

ix. Notifications: Finally, notifications can be added to the application to alert users when they have reached their spending goals, when they have exceeded their budgets, or when there are other important updates or reminders.

Overall, this sequence for goal design for an expense tracker application provides a framework for building a comprehensive and user-friendly application that can help users manage their finances more effectively.

4.2 Design Strategy

We use design strategy for a expense tracker for several reasons:

i. Helps to define project goals and objectives:
A design strategy helps to define the goals and objectives of a project. By identifying what needs to be achieved and why, a design strategy provides clarity and direction for the project team.

ii. Guides decision-making:
A design strategy provides a framework for decision-making throughout the project lifecycle. It helps to prioritize design decisions based on their impact on project goals and objectives, and ensures that design decisions align with the overall project strategy.

iii. Reduces risks:
A well-defined design strategy helps to reduce risks by identifying potential issues early on in the project lifecycle. It allows the project team to anticipate and address potential problems before they become major issues.

iv. Improves communication:
A design strategy facilitates communication among project stakeholders. It ensures that everyone is on the same page and working towards the same goals, which helps to prevent misunderstandings and improve collaboration.

v. Enhances user experience:
A design strategy ensures that user needs and preferences are taken into account

throughout the project lifecycle. It helps to create a user-centered design approach that focuses on improving the user experience and creating products that are more user-friendly and intuitive.

Overall, a design strategy is an essential part of any project as it provides a roadmap for achieving project goals and objectives, reduces risks, improves communication, and enhances user experience [26]. It helps to ensure that the project team is working towards a common goal and that all design decisions are aligned with the overall project strategy.

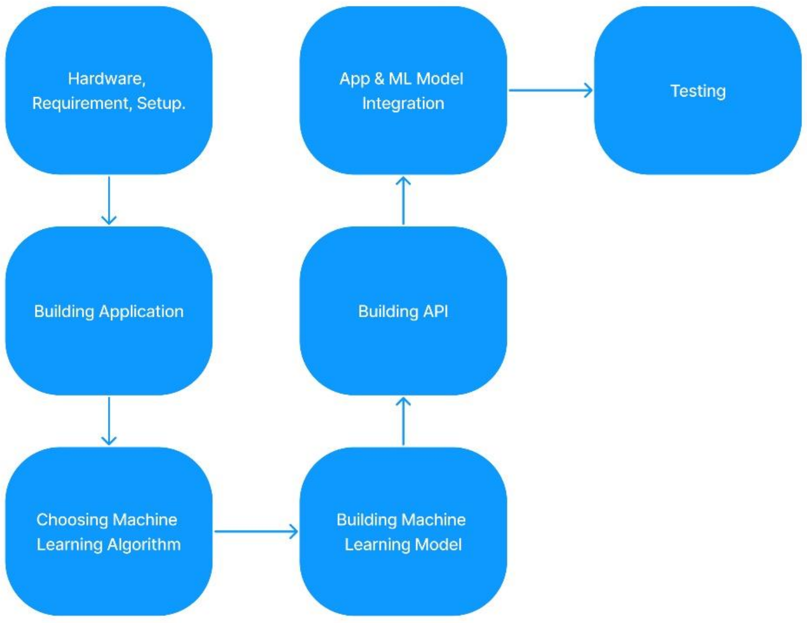

Hardware, Requirement, Setup.
App & ML Model Integration
Testing
Building Application
Building API
Choosing Machine Learning Algorithm
Building Machine Learning Model

Fig 4.2 Design Strategy Diagram

Figure 4.2 shows design strategy for a smart expense tracker application that includes machine learning.

i. Hardware and Requirement Setup:

The first step in the design strategy for an expense tracker application with machine learning is to define the hardware and software requirements for the project. This involves selecting the appropriate hardware and software tools and setting up the development environment.

ii. Building Application:
Once the hardware and software requirements are defined, the next step is to build the application. This involves creating the user interface, implementing user authentication, and adding functionality to the application to enable users to add expenses, set goals, view charts, and receive notifications.

iii. Choosing Machine Learning Algorithm:
The next step is to choose the appropriate machine learning algorithm that will be used to build the expense prediction model [27]. This involves evaluating various machine learning algorithms based on their suitability for the project, such as decision trees, random forests, or neural networks.

iv. Building Machine Learning Model:
Once the machine learning algorithm is chosen, the next step is to build the machine learning model. This involves preparing the data for training, selecting features that are relevant to expense prediction, and training the model using the selected algorithm.

v. Building API for ML Model:
After building the machine learning model, the next step is to build an API that will allow the application to communicate with the machine learning model [28]. This involves setting up a web server, defining the API endpoints, and integrating the machine learning model into the API [29].

vi. App and ML Model Integration:
The next step is to integrate the machine learning model into the application. This involves making API calls from the application to the server to get expense predictions based on user input [30]. The machine learning model can be used to provide personalized insights and recommendations to users based on their spending habits.

vii. Testing:

The final step is to test the application to ensure that it meets the requirements and performs as expected. This involves testing the user interface, testing the machine learning model, and testing the API to ensure that it can handle the expected traffic and provide accurate predictions [31].

Overall, this design strategy provides a framework for building an expense tracker application that includes machine learning. It ensures that the hardware and software requirements are defined, the application is built to meet user needs, the appropriate machine learning algorithm is selected and used to build the machine learning model, the API is set up to communicate with the model, and the application is tested to ensure that it meets the requirements and performs as expected.

4.3 Module Diagram

Module diagrams are a type of structural diagram used in software engineering to represent the architecture of a software system of expense tracker. A module diagram provides a high-level view of the software system, showing how the different components of the system are organized and how they interact with each other [32].

There are several reasons why module diagrams are useful in software engineering:

i. Communication: Module diagrams are a great way to communicate the architecture of a software system to stakeholders, such as project managers, developers, and clients. They provide a clear and concise visual representation of the system, making it easier for stakeholders to understand how the system works and how different components fit together.

ii. Design: Module diagrams can be used to design the software system by breaking it down into smaller, more manageable modules. This helps developers to focus on individual modules, making it easier to develop and test each component independently.

iii. Testing: Module diagrams can be used to plan and organize testing efforts. By breaking the software system down into modules, it is easier to identify which modules need to be tested, which modules are dependent on others, and which modules can be tested in isolation.

iv. Maintenance: Module diagrams can be used to help maintain the software system by providing a clear overview of the system's architecture. This makes it easier to identify areas that need to be updated or modified, as well as to understand how those changes will affect other parts of the system.

In summary, module diagrams are a valuable tool in software engineering because they help to communicate, design, test, and maintain software systems [33]. By providing a high-level view of the system's architecture, module diagrams make it easier for developers to understand and work with complex software systems.

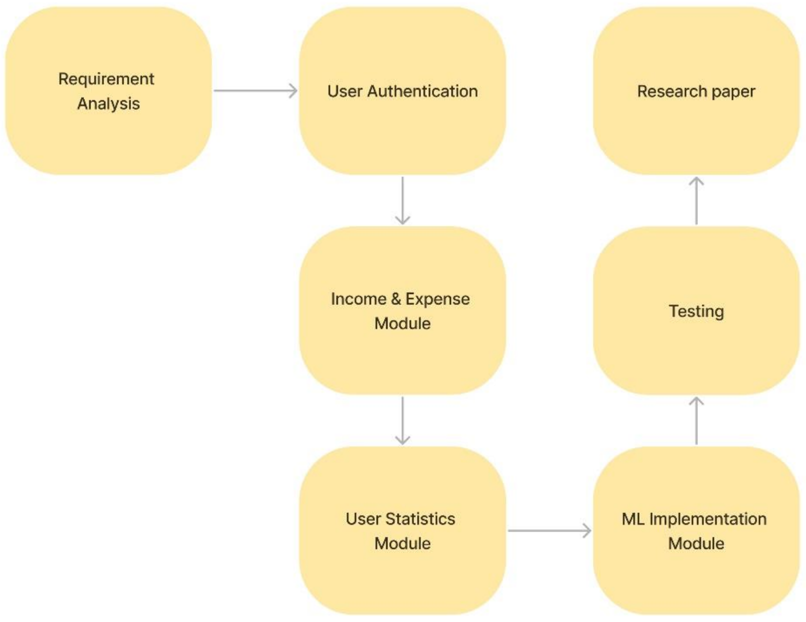


Fig 4.3 Module Diagram

Figure 4.3 gives module diagram for a smart expense tracker application and it is explained below:

i. Requirement Analysis:

The first step in developing an expense tracker application is to perform requirement analysis. This involves identifying the user's needs and requirements for the application, as well as the features and functionality that will be included in the application.

ii. User Authentication:

The next module in the sequence is user authentication. This module is responsible for handling user registration, login, and logout functionality. It ensures that only authorized users can access the application.

iii. Income & Expense Module:

The income & expense module is the core of the expense tracker application. It provides users with the ability to add and manage their income and expenses. This module includes features such as adding income, adding expenses, categorizing expenses, and viewing income and expense reports.

iv. User Statistics Module:

The user statistics module provides users with an overview of their financial situation. It displays various statistics and insights related to their income and expenses, such as total income, total expenses, monthly income and expense trends, and budget tracking.

v. ML Implementation Module:

Machine learning is used to read all device messages and then feed them to api of machine learning model. this machine learning model will give us bank related messages. And then with the help of programming libraries we will extract amount from the message.

vi. Testing:

Testing is an important module in the development process. It ensures that the application is functioning correctly and that it meets the user's needs and requirements. Testing can include functional testing, unit testing, integration testing, and user acceptance testing.

vii. Research Study:

The research study module involves writing a research study on the expense tracker application. This module focuses on the development process, the features and functionality of the application, and the user's experience with the application. It can include topics such as the effectiveness of machine learning algorithms in providing financial insights and the impact of the application on the user's financial behavior.

In summary, the module diagram for an expense tracker application includes modules for requirement analysis, user authentication, income & expense management, user statistics, machine learning implementation, testing, and research study. This sequence provides a comprehensive framework for developing an expense tracker application that meets the user's needs and provides valuable financial insights and recommendations.

4.4 Architecture Diagram

An architecture diagram is a visual representation of the high-level design of a software system, including its components, modules, and their interactions. There are several reasons why we use architecture diagrams of a smart expense tracker:

i. Communication: An architecture diagram provides a common language and understanding for all stakeholders involved in a project, including developers, managers, and clients. It allows everyone to see the big picture of the system and how its various components interact with each other.

ii. Planning and Design: An architecture diagram helps in planning and designing the software system by identifying the key components, their interactions, and their dependencies. It enables developers to make informed decisions about the system's design and helps them to identify potential issues before they arise.

iii. Scalability and Maintainability: An architecture diagram helps in ensuring the scalability and maintainability of the software system. It enables developers to identify potential bottlenecks and performance issues and design the system in a way that can handle future growth and changes.

iv. Documentation: An architecture diagram serves as a useful reference for documenting the system's design, which can be helpful for future developers and stakeholders who may need to understand or modify the system.

v. Quality Assurance: An architecture diagram helps in quality assurance by enabling developers and testers to verify that the system has been built as per the design specifications and that all components are functioning as expected.

In summary, an architecture diagram is an essential tool for planning, designing, and communicating the high-level design of a software system. It enables all stakeholders to have a common understanding of the system's components and their interactions, which helps to ensure that the system is scalable, maintainable, and meets the requirements of its users.

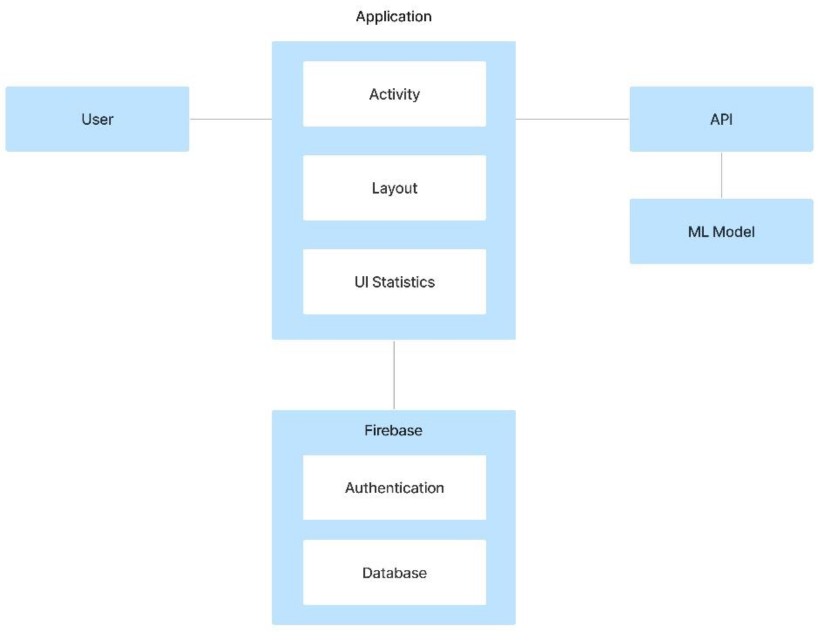


Fig 4.4 Architecture Diagram

Figure 4.4 gives architecture diagram for a smart expense tracker application:

The architecture diagram for an expense tracker application consists of three main components:

i. User: The user is the main actor of the system, who interacts with the application to manage their expenses.

ii. Application: The application is the software system that provides the user interface and functionality for managing expenses. It consists of three sub-components:
    1. Activities: Activities are the screens or windows that the user interacts with, such as the login screen, expense entry screen, and dashboard screen.
    2. Layout: The layout defines the visual appearance and organization of the application screens, such as the placement of buttons, text, and images.
    3. UI Statistics: The UI statistics component collects data on user interactions with the application, such as the number of times each screen is visited and the duration of each session.

iii. Firebase: Firebase is a cloud-based platform that provides two key services for the expense tracker application:
    1. Authentication: Firebase Authentication provides a secure and easy-to-use authentication system that allows users to sign up, sign in, and manage their account information securely.
    2. Database: Firebase Database provides a cloud-hosted NoSQL database that allows developers to store and sync data in real-time. In the case of the expense tracker application, the database is used to store user expenses.

Machine learning is used to read all device messages and then feed them to api of machine learning model. this machine learning model will give us bank related messages. And then with the help of programming libraries we will extract amount from the message.

In summary, the architecture diagram for an expense tracker application consists of a user, an application that provides activities, layout, and UI statistics, Firebase for authentication and database, and an optional API and ML model component for additional functionality. This architecture provides a scalable and secure foundation for building a comprehensive and user-friendly expense tracking application.

4.5 Class Diagram

Class diagrams are an important modeling tool in software engineering that help developers visualize and organize the structure of a software system. They provide a high-level overview of the classes and objects in the system, their relationships, and their attributes and methods. Here are some reasons why we use class diagrams for expense tracker:

i. Understanding the System: Class diagrams provide a clear view of the structure of the software system, which helps stakeholders to understand the overall architecture and design of the system.

ii. Communication: Class diagrams are a common language for communicating the design and structure of a software system to developers, stakeholders, and other interested parties. They help ensure everyone is on the same page when it comes to how the software is organized and how its components interact.

iii. Design and Planning: Class diagrams can be used to plan and design a software system before implementation begins. They help developers identify key components of the system and their relationships to each other, as well as to identify potential problems with the design before the system is built.

iv. Implementation: Once the design is finalized, class diagrams can be used to guide the implementation of the software system. Developers can use class diagrams to understand how the classes and objects in the system are related, and how they should be implemented.

v. Maintenance: Class diagrams can also be useful for maintaining and modifying a software system over time. They help developers understand the structure of the system and how changes to one component might impact other parts of the system.

In summary, class diagrams are an essential tool for software development because they provide a visual representation of the structure of a software system, which can be used to understand, communicate, plan, implement, and maintain the system.

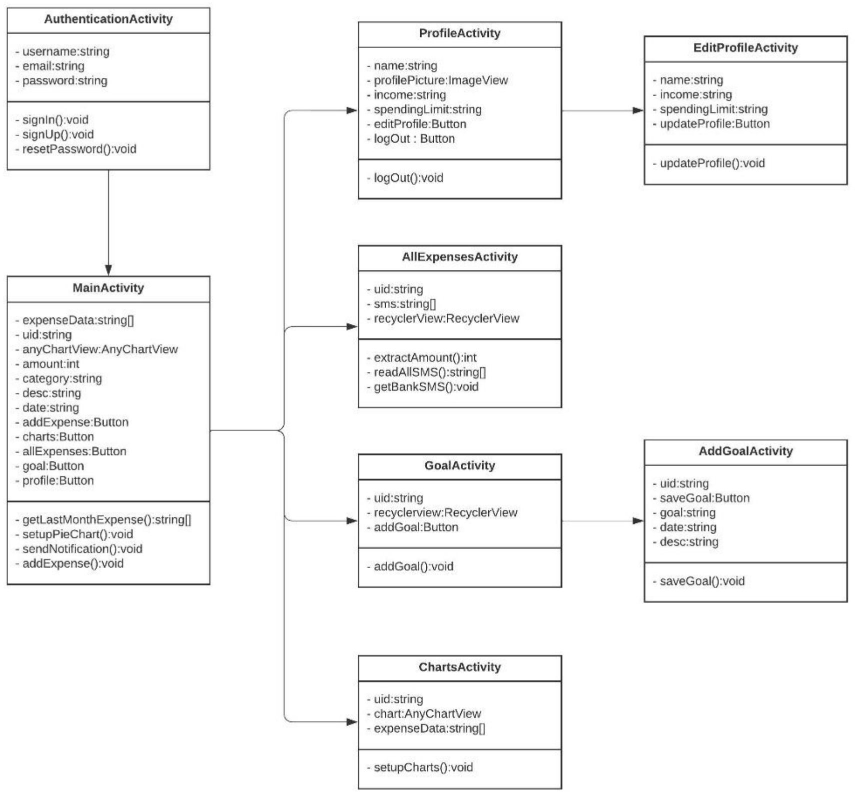


Fig 4.5 Class Diagram

In above figure 4.5 which is class diagram for expense tracker application we can see that it starts with Authentication Activity first, this activity will have the attributes such as username, email and password. With the help of methods such as signIn(), signUp() and resetPassword() user will be able to authenticate into the application.

After Authentication Activity the app will take it to the Main Activity which is also the home of the app. This activity will have attributes such as expense Data, uid of firebase, anyChartView, amount, category, description of the expense, date etc. This class has multiple methods such as getLastMonthExpense() method will get all previous month methods. setupPieChart() method will inflate the pie chart with values, sendNotification() method will notify user if there was any limit exceeded and at last

there will be addExpense() method and this method adds manual expense on the firebase database with values such as amount, category, date and description.
From home user can go to multiple classes depending on the UI element he clicks on. These classes are Profile, All Expenses, Goal and Charts.
In Profile Activity class there are attributes such as name, profile Picture, income, spending limit etc. In this class you also have options to log out and edit profile.
In Edit Profile activity there are attributes such as name, income, spending limit. With the help of update profile method user can save new information in profile.
All Expenses activity has attributes such uid, sms array and recycler view. It has the methods namely extractAmount(), realAllSMS() and getBankSMS().
Goal Activity class has the fields such as uid, recycler view. It has the method addGoal(). This add goal method takes us to Add Goal Activity which has the attributes such as uid, goal, date and description. It also has the method saveGoal.
And at last there is Charts Activity with fields such as uid, chart and expense data. It also has the setupCharts() method.

4.6 Sequence Diagram

Sequence diagrams are a type of UML diagram that are used to model the interactions between objects or components in a system over time. They are particularly useful for modeling complex interactions, such as those that occur in software systems or other technical processes.

There are several reasons why sequence diagrams are commonly used in software development projects:

i. Visualize the flow of information: Sequence diagrams allow developers to visualize the flow of information and messages between different components in a system. This can be particularly useful for understanding the behavior of complex systems, as it allows developers to see how different components interact and communicate with one another.

ii. Identify errors and inefficiencies: By examining the sequence diagram, developers can identify errors, bottlenecks, or inefficiencies in the system. This can help them to optimize the performance of the system and improve its overall functionality.

iii. Improve communication: Sequence diagrams can also be useful for improving communication between developers and stakeholders. By providing a visual representation of the system's behavior, developers can more easily explain the system's workings to non-technical stakeholders, such as managers or clients.

iv. Aid in testing: Sequence diagrams can be used to aid in testing, as they provide a clear and detailed view of the interactions between components in a system. This can help testers to design more effective test cases and identify potential issues more easily.

Overall, sequence diagrams are an important tool for software developers as they allow them to model and visualize the behavior of complex systems. By using sequence diagrams, developers can improve the performance and functionality of their systems, as well as improve communication and collaboration with stakeholders.

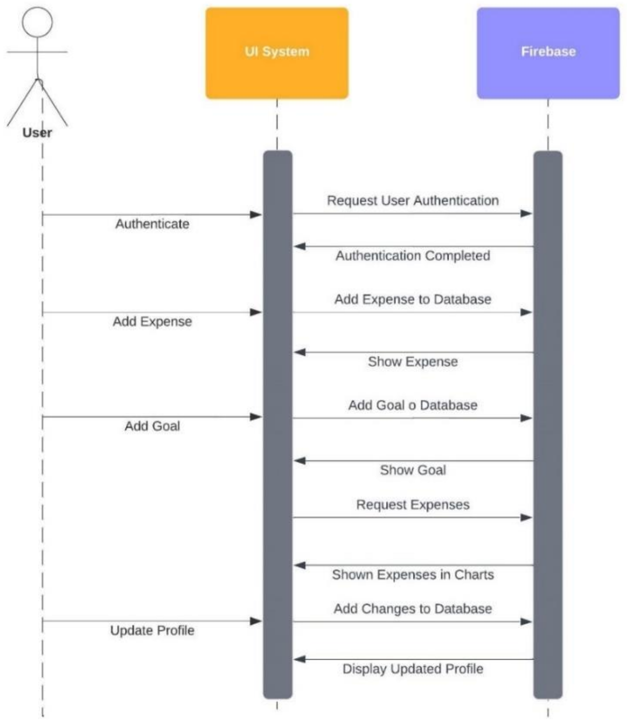


Fig 4.6 Sequence Diagram

This is the sequence diagram in figure 4.6 is for the expense tracker application. At first user will add credentials on the app to authenticate, then the app will validate the user credentials from firebase authentication. After authentication is successful the user will be redirected to the home.

Then the user can add expense with value such as amount, category, date and description. This expense will be stored on the firebase database by the app, and then it will be show on the screen.

User can add goal from the app with values like amount, date and description and then this goal will be saved on the firebase database by the app, after that it will be shown on the screen.

From the profile user has the option to update the profile, user can update the profile by entering new values such as name, salary max monthly spending limit. App will save these values on the firebase and then updated profile will be shown to the user.

4.7 Collaboration Diagram

Collaboration diagrams, also known as communication diagrams, are a type of UML diagram that shows the interactions and relationships between objects in a system. These diagrams are useful in project development for several reasons:

i. Visualizing Object Interactions: Collaboration diagrams provide a visual representation of how objects in a system interact with each other. By showing the message exchanges between objects, these diagrams help to clarify the flow of information and the responsibilities of each object.

ii. Identifying Design Issues: Collaboration diagrams can help identify design issues early in the development process. By analyzing the interactions between objects, developers can identify potential issues with the system's design, such as dependencies or coupling between objects.

iii. Improving Communication: Collaboration diagrams can help improve communication between developers by providing a common understanding of the system's architecture and design. These diagrams can help developers to communicate complex ideas more effectively and can help to avoid misunderstandings and miscommunications.

iv. Supporting Collaboration: Collaboration diagrams can also support collaboration between team members. By visualizing the interactions between objects, team members can more easily understand each other's work and can collaborate more effectively on the development of the system.

Overall, collaboration diagrams are a useful tool for project development because they help to clarify the interactions between objects in a system, identify design issues, improve communication, and support collaboration among team members.

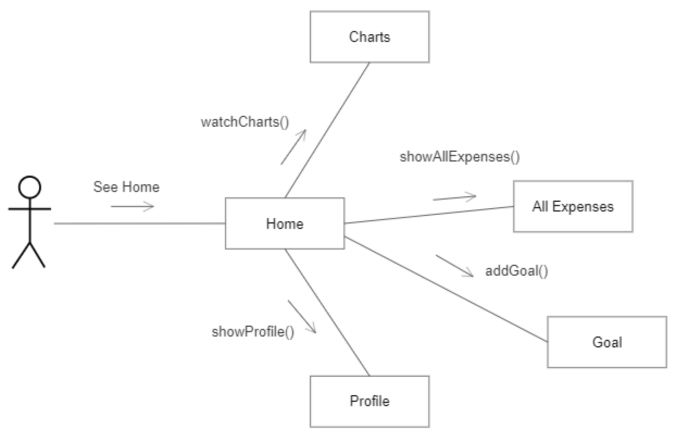


Fig 4.7 Collaboration Diagram

The figure 4.7 is collaboration diagram for the expense tracker application. In the diagram, how the entire app system works is shown. At first after the user authentication the user will go to the home. From home he can go to charts activity where charts will details will be shown with the help of methods such as watchCharts(). Then user can go to the all expenses activity where all the expenses are shown with date, amount category and description. User has the option to add the goals in goal activity where goal has the attributes such as date, amount, description etc. Goal will be saved on the database with the help of method addGoal(). And at last user can go to the profile activity where user’s personal information is shown such as profile picture, name, salary and max monthly spending limit. And this information is displayed using method showProfile().

4.8 State Chart Diagram

State chart diagrams are a type of behavioral diagram that model the behavior of a system or a component over time. They are useful in software development for modeling the state of a system or a component, as well as the events that cause the system or component to transition from one state to another.

There are several reasons why state chart diagrams are useful in software development projects:

i. Clarify Requirements: State chart diagrams can help to clarify the requirements of the system by modeling the behavior of the system in a visual way. By creating a state chart diagram, developers and stakeholders can gain a better understanding of how the system should behave under different conditions.

ii. Design and Analysis: State chart diagrams can be used to design and analyze the behavior of a system. By modeling the system's behavior in a state chart diagram, developers can identify potential issues and design a more robust system.

iii. Maintainability: State chart diagrams can help to make the system more maintainable by providing a clear understanding of the system's behavior. This can help developers to make changes to the system without introducing unintended consequences.

iv. Testability: State chart diagrams can be used to test the system by identifying the different states of the system and the events that trigger state transitions. This can help developers to create more effective tests that cover all possible scenarios.

In summary, state chart diagrams are useful in software development projects because they help to clarify requirements, design and analyze the behavior of a system, improve maintainability, and increase testability.

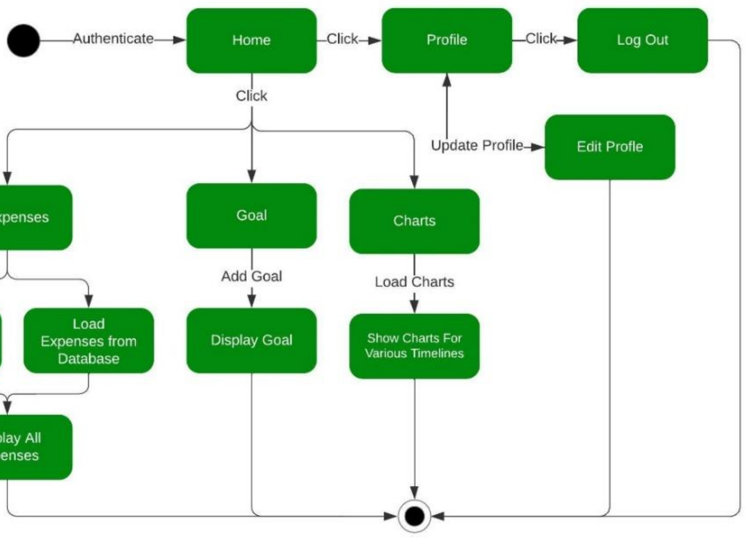


Fig 4.8 State Chart Diagram

State chart diagram of a smart expense tracker application shown in figure 4.8 and its explanation is given below:

The sequence starts with the user starting the application and being directed to the home screen, where they have several options to choose from. The user can either access their profile by clicking on the profile icon or choose to view their expenses, goals, or charts.

If the user clicks on the profile icon, they are directed to the profile screen where they can view their personal information. From the profile screen, the user can choose to update their profile by clicking on the update profile button. This will take the user to the edit profile screen where they can make changes to their personal information.

Once the user has finished editing their profile, they can save the changes and the application will return to the profile screen. From here, the user can choose to log out of the application by clicking on the log out button.

If the user chooses to view their expenses, they will be directed to a screen where they can view all of their expenses. The application will make an API call to load the user's expenses and display them on the screen. Once the expenses are loaded, the user can scroll through them and view the details of each expense.

If the user chooses to view their goals, they will be directed to a screen where they can add a new goal or view their existing goals. If the user chooses to add a new goal, they can enter the details of the goal and save it. The application will then display the new goal on the screen.

If the user chooses to view their charts, they will be directed to a screen where they can view charts for various timelines. The application will load the charts and display them on the screen, allowing the user to view their expenses over time.

Once the user has finished using the application, they can simply close the application, which will end the sequence.

In summary, the state chart diagram for an expense tracker application includes several states, such as start, home, profile, update profile, edit profile, all expenses, goals, and charts. These states are interconnected and allow the user to navigate through the application to view and manage their expenses, goals, and personal information.

4.9 Activity Diagram

Activity diagrams are a type of UML diagram that are used to model the flow of activities or processes within a system. They can be used to illustrate how different actors (users, systems, etc.) interact with a system, and how data flows through the system.

Activity diagrams are useful for project design and development for several reasons:

i. Visualization of Workflow: Activity diagrams provide a clear visualization of the workflow or processes within a system, making it easier to understand and communicate how the system works.

ii. Identifying System Requirements: By modeling the activities and processes involved in a system, activity diagrams can help identify system requirements and ensure that all necessary activities are included.

iii. Clarification of System Functionality: Activity diagrams can help clarify the functionality of a system and ensure that all necessary activities are included in the system design.

iv. Communication: Activity diagrams provide a common visual language for communicating system behavior and processes to stakeholders, including developers, project managers, and end-users.

v. Validation and Verification: Activity diagrams can be used to validate and verify system requirements by simulating the flow of activities and processes in the system and identifying potential issues or errors.

Overall, activity diagrams are an important tool for project design and development as they provide a clear and visual representation of system behavior and processes, helping to ensure that the system meets the requirements and functions as intended.

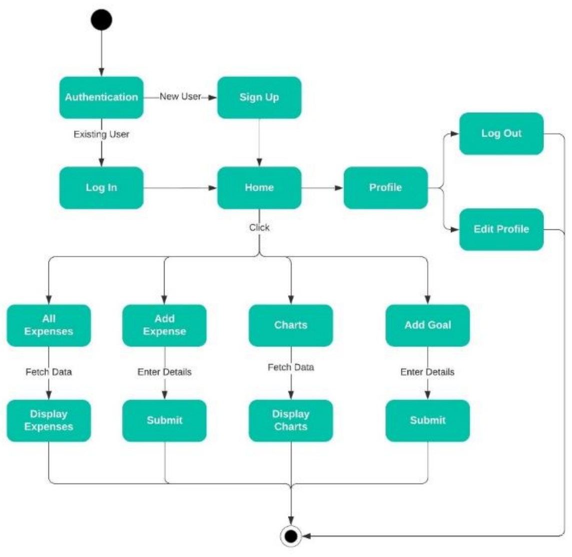


Fig 4.9 Activity Diagram

In the above activity diagram (figure 4.9), we can see that when user starts the app first requirement is to authenticate, if the user is already existing user then he will log in and enter home otherwise user registration is required. From home user can go to the profile, in profile there are two options which are log out and edit profile. In edit profile user can update the profile with new information. From home by clicking on various UI elements user can perform various actions such as by clicking on all expenses button, user can see all the expenses with information such as date, amount, category and description etc. By clicking on the add expense button user can add expense on the firebase by putting value such as amount, date, category and description. After saving this expense it will be displayed in all expenses activity. Clicking on charts user can see overview of his/her expenses with help of pie chart. Pie chart shows all the expenses for all the categories.

5. IMPLEMENTATION

5.1 Implementation Strategy

The "Expense Tracker" is a mobile application for android users. The app is developed using Kotlin and XML languages in android studio. "Expense Tracker" helps users to keep track of their daily expenses so that they can manage their finances properly and have complete idea of their spending habits. All the expenses are shown with help of Pie Chart and Bar Graph so that users get quick and clear idea of their spendings. Not only this app allows user to enter the expense manually along with amount, category and description of the expense, but also it can automatically detect the bank SMS from your mobile phone and read the credited or debited amount. Firebase is used as the online database for the app, so that even if you uninstall the app, still your data will be saved. To use the "Expense Tracker" app the user will have to sign up using email address and password.

Text classification technique is used to detect Bank SMS accurately. In text classification a category is assigned to a document. SMS are text messages and we want to classify SMS as "Bank SMS" or "Normal SMS"[34]. We use supervised machine learning in text classification and that is why the labelled dataset is required. So, to classify Bank SMS we have used NB (Naive Bayes) supervised ML algorithm, which is mostly used in textual classification [35]. Also, in this study we analysed the performance of the NB (Naive Bayes) algorithm on the basis of “precision” and “accuracy”. Also compared performance of NB (Naive Bayes) with other ML algorithms such as SVC (Support Vector Classifier), LR (Logistic-Regression), ETC (Extra-Trees-Classifier), KNC (K-Neighbours-Classifier), RFC (Random-Forest-Classifier) and DTC (Decision-Tree-Classifier) [36].

We need to build an app which can read device messages automatically and extracts the expense amount from that and an app with with features such as adding offline expense with amount, category, date and description, viewing expense details in chart, adding goal and viewing all expenses.

To build such app first we need to first integrate firebase with app to add features of user authentication. Then we need to add other features such as adding offline expense and storing on firebase database, adding goals, viewing all expenses in a list, viewing expenses with the help of charts and adding functionality for profile [37].

After building an application which can do basic things and store data on firebase database, then to add features of reading expenses automatically we need to build a machine learning model which classifies the bank related messages correctly. The process of building this machine learning model is given as follows.

We need to understand the workflow of the system. Initially, the expense tracker app reads the SMS messages from the user's device & feeds them to the ML model. This ML model accurately identifies bank-related messages and returns them to the app [38]. The app then extracts the credited or debited amount from the user's bank messages and adds the entry of the expense into the database. Finally, the app displays the user's data using various charts.

To build a ML model that detects the bank SMS accurately we need to perform following steps:

I. Importing Libraries:
NumPy: It’s a library from python used to deal with arrays, and in addition it provides features of matrices, linear algebra, etc.
Pandas: It’s a python library used to deal with datasets.
Matplotlib: This library is used for creating animated, static and interactive visualizations in Python.
Sklearn: Python library with various effective methods for statistical modelling and machine learning.

II. Data Cleaning:
At first, we imported the dataset with the help of pandas’ library. We removed all the unnecessary columns from the dataset so that we have only two columns left, one is the text column containing SMS text and other one is target column containing only two values either the text is “Bank SMS" or "Normal SMS". Now as a next step we encoded the target column so that it contains value as

either 0 or 1. We checked if there were any duplicate values present in our dataset and then we deleted the duplicate values.

III. Data Preprocessing:

To improve the accuracy of the ML model we preprocessed the data and then fed it to the model [39]. To preprocess the data, we created a function which converted the text of the SMS to the lower case, also it performed the tokenization of the text along with removal of the special characters present in the SMS.

IV. Model Building:

In this step we will build naïve bayes model which is based on Bayes' theorem (Explained in 4. Naïve Bayes Classifier) [40]. After data preprocessing, we first imported the TfidfVectorizer (an algorithm that assesses the relevance of words to a given document using their frequency) and applied that on the text column of the dataset [1]. Next, we created the dependent & the independent variables and then dataset was split into the training set & test set in the ratio given as 80 : 20 respectively. The NBC (Naïve-Bayes-Classifier) was imported & trained on the training dataset.

V. Evaluation:

After training of the naive bayes classifier, we evaluated the model on "accuracy" and "precision". At the end we achieved accuracy of 99.33% and precision of 97.67%.

After building this machine learning model we will need to deploy it on any cloud platform, so that it can be accessed by an android app with API call. In this case we have used railway platform to deploy our machine leaning model and accessed it from android app with the help of volley library.

5.2 Hardware Platform Used

To build android application and machine learning model we required following hardware platforms.

i. **Laptop/PC:** A laptop or PC is required to develop the Android application and machine learning model. The laptop or PC should have a powerful processor and sufficient memory to handle the development tools and software libraries required for the project. A modern laptop or PC with an Intel i3/i5/i7 processor would be suitable.

ii. **System:** Intel Processor i3/i5/i7: The processor is the brain of the computer, and a powerful processor is required to handle the computational demands of the project. An Intel processor i3/i5/i7 would be suitable for developing the Android application and machine learning model.

iii. **RAM:** 2GB or above: Random Access Memory (RAM) is the memory that the computer uses to store data that is currently being used by the processor. A minimum of 2GB of RAM is required for developing the Android application and machine learning model. However, for optimal performance, it is recommended to have more than 2GB of RAM.

iv. **Hard Disk:** 256 GB or above: A hard disk is used to store the operating system, development tools, and project files. A minimum of 256 GB of hard disk space is required to develop the Android application and machine learning model. However, having more than 256 GB of hard disk space is recommended.

v. **Android Smartphone:** An Android smartphone is required to test the application on a real device. It is essential to have a smartphone that meets the minimum requirements for the Android version that the application is targeting.

vi. **USB Cable:** A USB cable is required to connect the Android smartphone to the computer to test the application on the device. It is essential to use a reliable USB cable that supports data transfer to avoid any connection issues.

In summary, the hardware platform requirements for building an Android application and machine learning model are a powerful laptop or PC with an Intel processor i3/i5/i7, a minimum of 2GB of RAM, at least 256 GB of hard disk space, an Android smartphone that meets the minimum requirements, and a reliable USB cable to connect the smartphone to the computer.

5.3 Software Platform Used

To build an android app and machine learning model we required following software platforms.

i. **Operating System:** An operating system is required to run the software tools and development environment. Windows, macOS, or Linux are the most commonly used operating systems for developing Android applications and machine learning models.

ii. **Android Studio:** Android Studio is the official integrated development environment (IDE) for Android app development. It provides a complete environment for developing, testing, and debugging Android applications.

iii. **Jupyter Notebook:** Jupyter Notebook is an open-source web application that allows developers to create and share documents that contain live code, equations, visualizations, and narrative text. It is often used for data analysis and machine learning development.

iv. **Android OS:** Android OS is the operating system that runs on Android smartphones and tablets. It is necessary to have a basic understanding of the Android OS to develop an Android application.

v. **Railway Platform:** Railway Platform is a cloud platform that provides various tools and services for developers, including a database, API management, and authentication. It can be used to deploy and manage the backend of an Android application.

vi. **PyCharm:** PyCharm is a powerful IDE for Python development. It can be used for developing machine learning models and integrating them with Android applications.

vii. **Volley:** Volley is an open-source networking library for Android applications. It provides a fast and easy way to make HTTP requests and handle responses.

viii. **AnyChart:** AnyChart is a JavaScript charting library that can be used to create interactive charts and graphs for web and mobile applications.

ix. **Firebase:** Firebase is a mobile and web application development platform that provides various services, including hosting, authentication, and database management.

x. **Glide:** Glide is an open-source image loading and caching library for Android applications. It provides an easy way to load and display images in an Android application.

xi. **GitHub:** GitHub is a web-based hosting service for version control using Git. It can be used to store and manage the source code for an Android application and machine learning model.

xii. **Numpy:** Numpy is a Python library for scientific computing. It provides support for arrays, matrices, and mathematical functions, making it useful for data analysis and machine learning development.

xiii. **Pandas:** Pandas is a Python library for data manipulation and analysis. It provides support for data structures and tools for data analysis, making it useful for data preprocessing in machine learning development.

In summary, the software platform requirements for building an Android application and machine learning model are an operating system, Android Studio, Jupyter Notebook, Android OS, Railway Platform, PyCharm, Volley, AnyChart, Firebase, Glide, GitHub, Numpy, and Pandas [23]. These tools and platforms provide a complete environment for developing, testing, and deploying an Android application and integrating it with a machine learning model [33].

5.4 Deployment Diagram

A deployment diagram is used in software engineering to depict how software components and hardware resources are deployed in a system or application. It shows the physical architecture of the system and the distribution of software components across hardware nodes.

The main reason for using a deployment diagram in a project is to help visualize and plan the physical deployment of software components and hardware resources. It can help to identify potential issues with hardware or software dependencies, and ensure that the system can be deployed in a way that meets performance, scalability, and reliability requirements.

Deployment diagrams can also help to communicate the deployment plan to stakeholders who may not be familiar with the technical details of the system. This can be particularly important in larger projects where there are multiple teams or vendors involved in the deployment process.

In summary, a deployment diagram can be a valuable tool for project planning, communication, and ensuring the successful deployment of a software system or application.

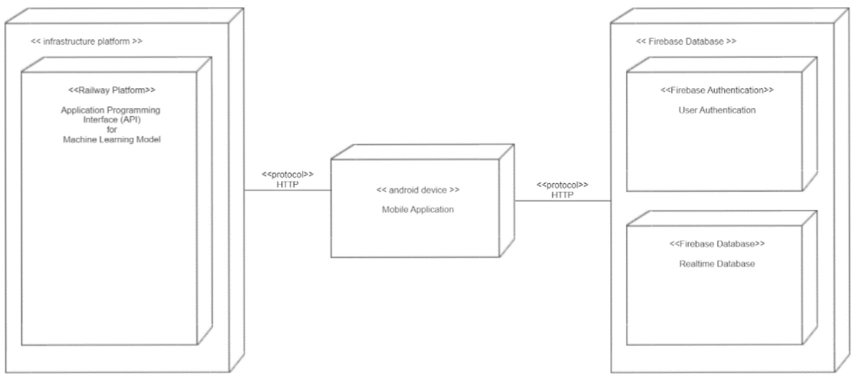


Fig 5.1 Deployment Diagram

In above figure 5.1 which gives deployment diagram for smart expense tracker application, there are various elements which explained below:

i. Railway Platform (API of ML model): The machine learning model API is hosted on the Railway Platform. The API is responsible for processing incoming messages from the user's device and returning a prediction of whether the message contains an expense or not [37]. It uses machine learning algorithms to analyze the message content and extract relevant information such as the expense amount, date, and description.

ii. Mobile Application: The mobile application is the interface for the user to interact with the expense tracker app. It allows the user to manually add expenses and view their expense history. Additionally, it communicates with the Firebase authentication and database services to manage user authentication and store user data.

iii. Firebase Authentication: Firebase Authentication is a service provided by Firebase that allows the app to authenticate users using various methods such as email and password, Google Sign-In, or Facebook Login. It ensures that only authorized users can access the app's features and data.

iv. Firebase Database: Firebase Database is a cloud-hosted NoSQL database provided by Firebase. It is used to store user data, including the user's expense history. The app communicates with the Firebase Database to save newly added expenses and retrieve existing expenses.

The overall structure of the app involves the mobile application communicating with Firebase to manage user authentication and store user data, and with the Railway Platform to use the machine learning model API to analyze incoming messages for expenses. When the user adds an expense manually, the app sends the expense data to Firebase, which saves it to the Firebase Database. When a new message arrives on the user's device, the app sends the message content to the machine learning model API on the Railway Platform to determine if it contains an expense. If the API returns a positive prediction, the app extracts the relevant expense information and sends it to Firebase, which saves it to the Firebase Database.

In summary, this app structure combines the functionality of a traditional expense tracker with the added convenience of automatically extracting expenses from the user's messages with the help of a machine learning model. The Firebase services handle user authentication and data storage, while the machine learning model API on the Railway Platform provides automated expense detection.

5.5 Implementation Level Details (Algorithms)

To build a machine learning model there are top two candidate algorithms which can provided high accuracy to classify bank related messages correctly. First is Naïve Bayes Algorithm and second is Support Vector Classifier. We have implemented naïve bayes algorithm for this project. These two algorithms are discussed below in detail.

1. **Naive Bayes Algorithm:**

   Naive Bayes is a probabilistic algorithm that is widely used for classification tasks in machine learning [37]. It is based on Bayes' theorem, which states that the probability of a hypothesis (in this case, a class label) given some observed evidence (in this case, a set of input features) is proportional to the product of the probability of the evidence given the hypothesis and the prior probability of the hypothesis [1].

   In the context of classification, the Naive Bayes algorithm works as follows:

   a. Input Data: The algorithm is trained on a dataset consisting of labeled examples. Each example is a set of input features, along with a corresponding class label.

   b. Feature Extraction: The algorithm extracts relevant features from the input data. It assumes that the features are independent of each other given the class label (hence the name "Naive").

   c. Probability Estimation: The algorithm estimates the probabilities of each class label given the input features. It does this by applying Bayes' theorem, using the prior probabilities of each class label (which are estimated from the training data) and the conditional probabilities of each input feature given each class label (which are estimated from the training data using techniques such as maximum likelihood estimation or Bayesian estimation).

d. Prediction: Once the probabilities of each class label have been estimated, the algorithm selects the class label with the highest probability as the predicted class for the input data.

One of the key advantages of Naive Bayes is its simplicity and speed. It requires only a small amount of training data and can be trained quickly. Additionally, it is well-suited for text classification tasks, such as spam detection or sentiment analysis, where the input features are often binary or categorical.

However, Naive Bayes has some limitations. One of the main assumptions of the algorithm is that the input features are independent of each other given the class label, which may not always be true in practice. Additionally, it may not perform as well as more complex algorithms on datasets with many input features or complex dependencies between the input features and the class label.

Overall, Naive Bayes is a useful and widely-used algorithm for classification tasks in machine learning, particularly for text classification tasks where the input features are binary or categorical.

2. **Support Vector Classifier:**

Support Vector Classifier (SVC) is a type of supervised learning algorithm that is used for classification tasks. It is based on the concept of Support Vector Machines (SVMs), which are a set of algorithms used for both classification and regression analysis. The main idea behind SVC is to find the hyperplane that maximally separates the different classes in the training data.

The SVC algorithm works by first mapping the input data into a high-dimensional feature space using a kernel function. It then finds the hyperplane that maximally separates the different classes in this feature space. The hyperplane is chosen such that it has the largest margin, which is the distance between the hyperplane and the closest data points from both classes.

One of the strengths of SVC is that it can handle non-linearly separable data by using non-linear kernel functions such as polynomial or radial basis function (RBF) kernels [2]. These kernels allow the SVC algorithm to project the data into a higher dimensional space where it becomes linearly separable.

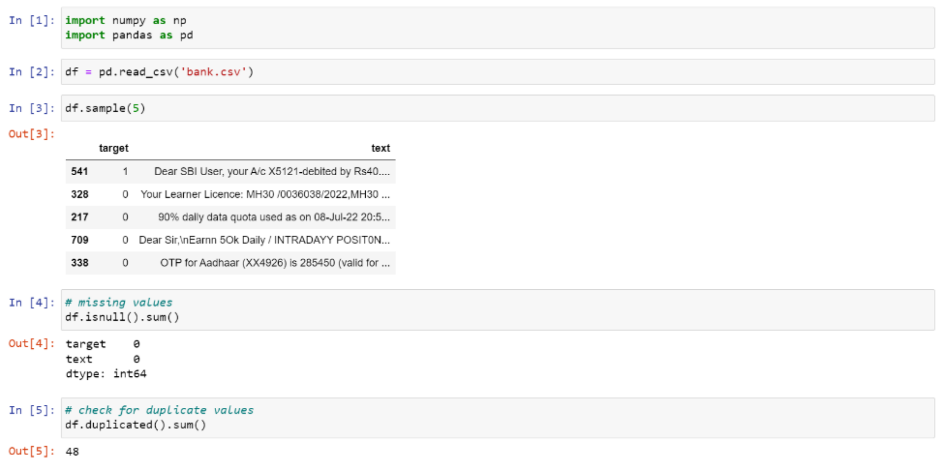
The SVC algorithm also has several important parameters that need to be tuned for optimal performance. These include the kernel type, kernel parameters (such as the degree of the polynomial kernel or the gamma parameter for the RBF kernel), and the regularization parameter C, which controls the tradeoff between achieving a large margin and allowing some misclassifications in the training data.

The training process of SVC involves finding the optimal values of these parameters that minimize the error on the training data. Once the model is trained, it can be used to predict the class labels of new data points by simply computing which side of the hyperplane they lie on.

SVC has been successfully applied to a wide range of classification tasks in various fields, including image classification, text classification, and bioinformatics. Its ability to handle non-linearly separable data and its robustness to noise and outliers make it a popular choice for many machine learning applications.

In summary, SVC is a powerful supervised learning algorithm used for classification tasks. It works by finding the hyperplane that maximally separates the different classes in the training data using a kernel function. Its ability to handle non-linearly separable data and its robustness to noise and outliers make it a popular choice for many machine learning applications.

To create naïve bayes classification model we first imported numpy and pandas libraries as shown in figure 5.2. Then we imported the dataset and checked for inconsistencies in the database and removed them



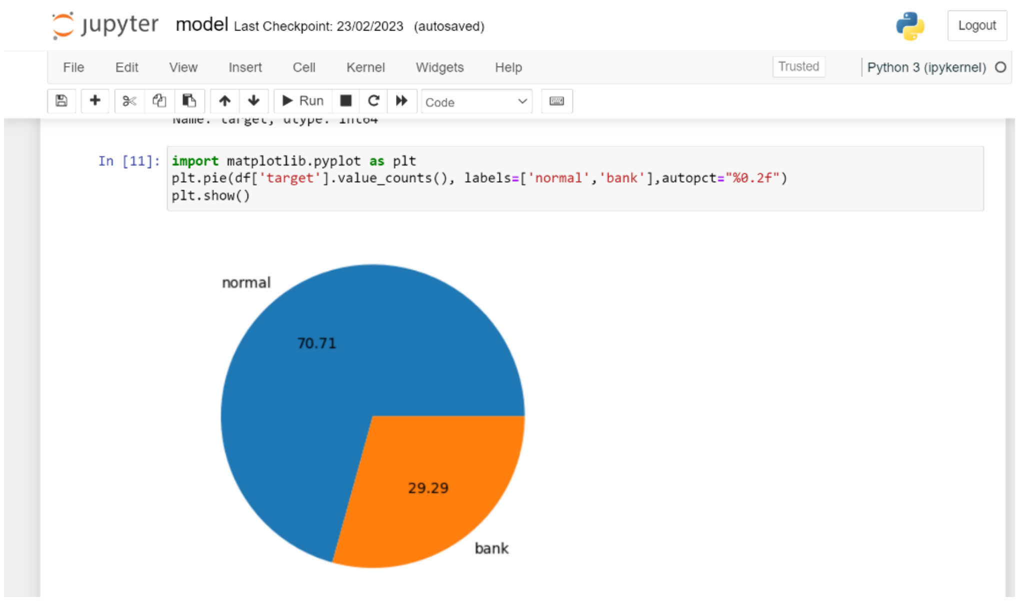
```
In [1]: import numpy as np
        import pandas as pd

In [2]: df = pd.read_csv('bank.csv')

In [3]: df.sample(5)
Out[3]:
```

	target	text
541	1	Dear SBI User, your A/c X5121-debited by Rs40....
328	0	Your Learner Licence: MH30 /0036038/2022,MH30 ...
217	0	90% daily data quota used as on 08-Jul-22 20:5...
709	0	Dear Sir,\nEarnn 5Ok Daily / INTRADAYY POSITON...
338	0	OTP for Aadhaar (XX4926) is 285450 (valid for ...

```
In [4]: # missing values
        df.isnull().sum()

Out[4]: target    0
        text      0
        dtype: int64

In [5]: # check for duplicate values
        df.duplicated().sum()

Out[5]: 48
```

Fig 5.2 Importing Libraries & Dataset

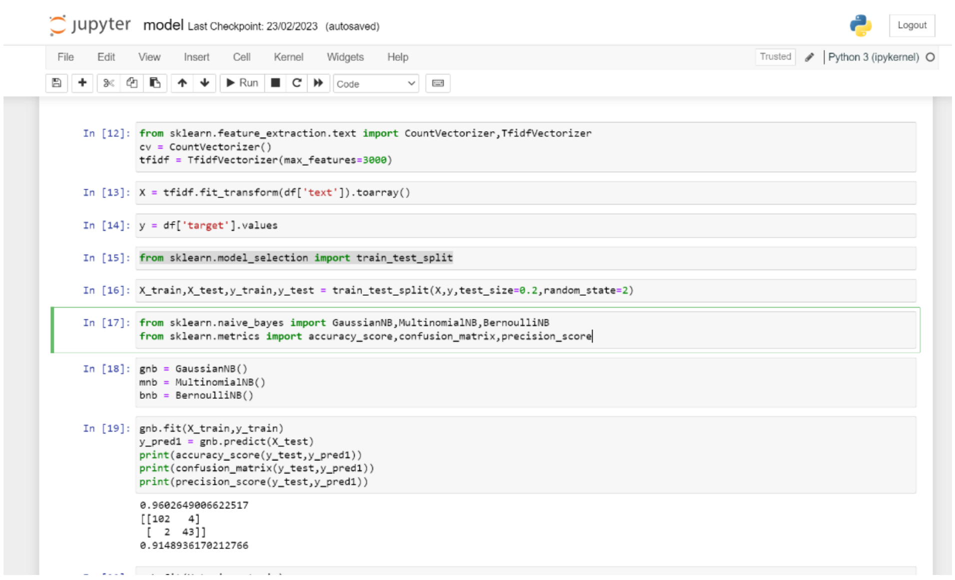
Then we imported matplotlib library to get visual idea of the dataset using pie chart as shown in figure 5.3.



```
In [11]: import matplotlib.pyplot as plt
         plt.pie(df['target'].value_counts(), labels=['normal','bank'],autopct="%0.2f")
         plt.show()
```

Fig 5.3 Pie Chart Implementation

Next step is to create Naïve Bayes model for classification of bank messages accurately(shown in figure 5.4).


jupyter model Last Checkpoint: 23/02/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [12]: from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
         cv = CountVectorizer()
         tfidf = TfidfVectorizer(max_features=3000)

In [13]: X = tfidf.fit_transform(df['text']).toarray()

In [14]: y = df['target'].values

In [15]: from sklearn.model_selection import train_test_split

In [16]: X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)

In [17]: from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
         from sklearn.metrics import accuracy_score,confusion_matrix,precision_score

In [18]: gnb = GaussianNB()
         mnb = MultinomialNB()
         bnb = BernoulliNB()

In [19]: gnb.fit(X_train,y_train)
         y_pred1 = gnb.predict(X_test)
         print(accuracy_score(y_test,y_pred1))
         print(confusion_matrix(y_test,y_pred1))
         print(precision_score(y_test,y_pred1))

         0.9602649006622517
         [[102   4]
          [  2  43]]
         0.9148936170212766
```

Fig 5.4 Naïve Bayes Model

This code snippet imports the necessary modules from the Scikit-learn library to perform text classification using different types of Naive Bayes classifiers. The steps of the code are as follows:

Import the CountVectorizer and TfidfVectorizer classes from the sklearn.feature_extraction.text module.

Create an instance of CountVectorizer using the default settings, which will be used to transform the text data into a matrix of word counts.

Create an instance of TfidfVectorizer with a maximum of 3000 features, which will be used to transform the text data into a matrix of term frequency-inverse document frequency (TF-IDF) values.

Transform the text data in the df['text'] column into a matrix of features using the TfidfVectorizer, and convert it to an array using the toarray() method. The target variable is stored in the df['target'] column.

Import the train_test_split function from sklearn.model_selection to split the data into training and testing sets, with a test size of 0.2 and a random state of 2.

Create the training and testing sets by calling the train_test_split function on the feature matrix and target variable arrays.

Import the GaussianNB, MultinomialNB, and BernoulliNB classes from sklearn.naive_bayes, which represent different variants of Naive Bayes classifiers.

Import the necessary metrics from sklearn.metrics to evaluate the performance of the classifiers.

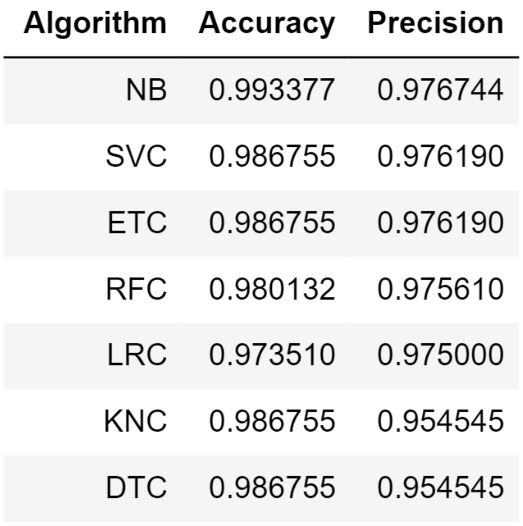
Create an instance of BernoulliNB and fit it to the training data using the fit method.

Use the trained model to make predictions on the testing data and store the predicted labels in y_pred3.

Print the accuracy score, confusion matrix, and precision score of the predictions.

Overall, this code is performing text classification using a Naive Bayes classifier with TF-IDF features, and evaluating the performance of the classifier using various metrics.

We evaluated naïve bayes algorithm & we got accuracy of 99.33% and precision of 97.67%.



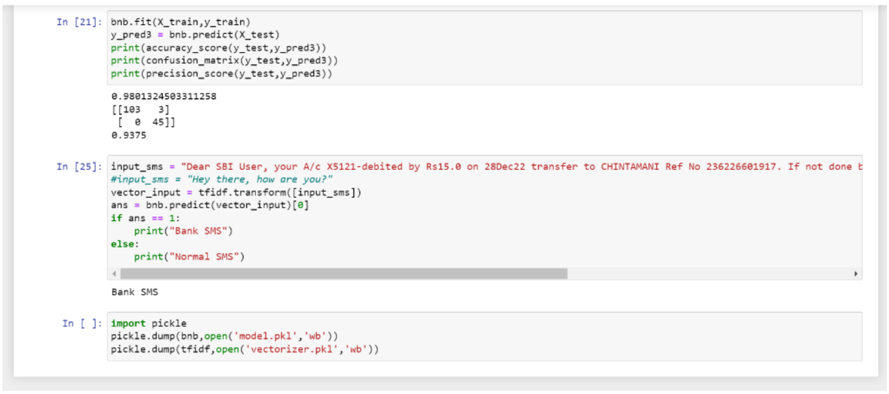
Algorithm	Accuracy	Precision
NB	0.993377	0.976744
SVC	0.986755	0.976190
ETC	0.986755	0.976190
RFC	0.980132	0.975610
LRC	0.973510	0.975000
KNC	0.986755	0.954545
DTC	0.986755	0.954545

Table 5.1 Performance Metrics of Algorithms

Table 5.1 shows performance metrics of various algorithms. After evaluating the Naïve bayes model we wanted to check if there is any other algorithm that can perform better than current so we compared its performance with multiple other ML algorithms such as SVC (Support-VectorClassifier), ETC (Extra-Tree-Classifier), RFC (Random-Forest-Classifier), LRC (Logistic-Regression-Classifier), KNC (K-Neighbors Classifier) and DTC (Decision Tree Classifier). From the figure you can see that after comparing all the algorithms with Naïve bayes, the NB (Naïve bayes) still gives results better that all other algorithms in terms of both "accuracy" and "precision" with accuracy of 99.33% and Precision of 97.67%. Second best performer is Support Vector Classifier with accuracy of 98.67% and precision of 97.61%. Furthermore, the Extra

Tree Classifier gives accuracy of 98.67% and precision of 97.61, Random Forest gives accuracy of 98.01% and precision of 97.56%, Logistic Regression gives accuracy of 97.35% and precision of 97.50%, KNC gives accuracy of 98.67% and precision of 95.45% and DTC gives accuracy of 98.67% and precision of 95.45%.

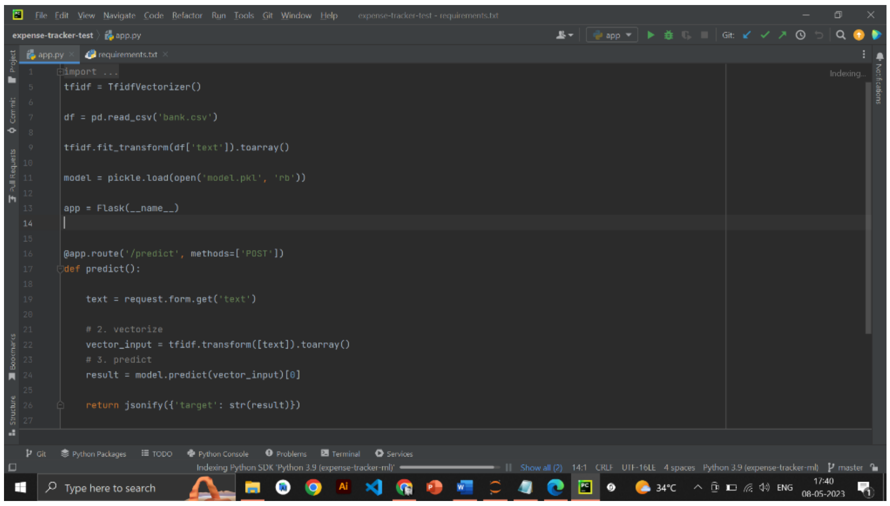
In figure 5.5 after the comparison of the naïve bayes algorithm with other algorithms we predicted a message with naïve bayes model and then exported pickle file of that model to build API.



```
In [21]: bnb.fit(X_train,y_train)
         y_pred3 = bnb.predict(X_test)
         print(accuracy_score(y_test,y_pred3))
         print(confusion_matrix(y_test,y_pred3))
         print(precision_score(y_test,y_pred3))

         0.9801324503311258
         [[103   3]
          [  0  45]]
         0.9375

In [25]: input_sms = "Dear SBI User, your A/c X5121-debited by Rs15.0 on 28Dec22 transfer to CHINTAMANI Ref No 236226601917. If not done b
         #input_sms = "Hey there, how are you?"
         vector_input = tfidf.transform([input_sms])
         ans = bnb.predict(vector_input)[0]
         if ans == 1:
             print("Bank SMS")
         else:
             print("Normal SMS")

         Bank SMS

In [ ]:  import pickle
         pickle.dump(bnb,open('model.pkl','wb'))
         pickle.dump(tfidf,open('vectorizer.pkl','wb'))
```

Fig 5.5 Model Prediction

After exporting the pickle file we create flask API with python language in PyCharm as shown in figure 5.6


File Edit View Navigate Code Refactor Run Tools Git Window Help
expense-tracker-test - requirements.txt
expense-tracker-test
app.py
requirements.txt
Git Python Packages TODO Python Console Problems Terminal Services
Indexing Python SDK 'Python 3.9 (expense-tracker-ml)'
14:1 CRLF UTF-16LE 4 spaces Python 3.9 (expense-tracker-ml) master
Type here to search
34°C ENG 17:40 08-05-2023

```
import ...
tfidf = TfidfVectorizer()

df = pd.read_csv('bank.csv')

tfidf.fit_transform(df['text']).toarray()

model = pickle.load(open('model.pkl', 'rb'))

app = Flask(__name__)


@app.route('/predict', methods=['POST'])
def predict():

    text = request.form.get('text')

    # 2. vectorize
    vector_input = tfidf.transform([text]).toarray()
    # 3. predict
    result = model.predict(vector_input)[0]

    return jsonify({'target': str(result)})
```

Fig 5.6 PyCharm Flask API

After creating flask API successfully, we deployed it on the railway cloud platform as shown in figure 5.7.

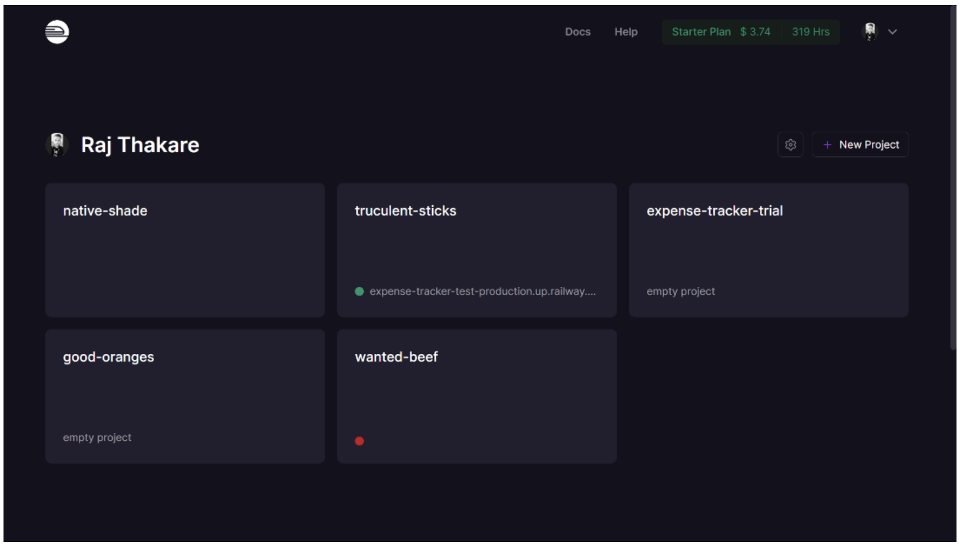
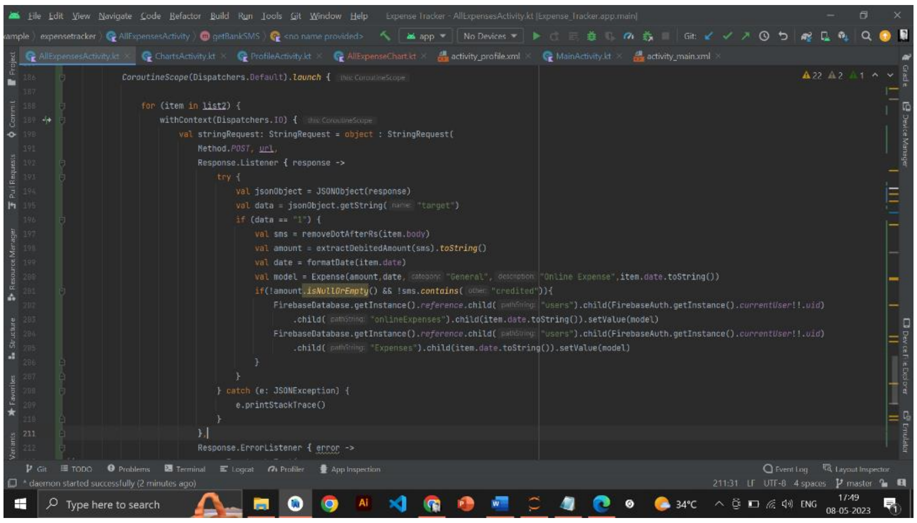


Fig 5.7 Railway Platform

With the help of volley library in android studio we accessed the ml model (shown in figure 5.8) deployed on railway platform. Using the URL from railway platform we categorized all bank related messages and then extracted amount from them.



```
CoroutineScope(Dispatchers.Default).launch {

    for (item in list2) {
        withContext(Dispatchers.IO) {
            val stringRequest: StringRequest = object : StringRequest(
                Method.POST, url,
                Response.Listener { response ->
                    try {
                        val jsonObject = JSONObject(response)
                        val data = jsonObject.getString("target")
                        if (data == "1") {
                            val sms = removeDotAfterRs(item.body)
                            val amount = extractDebitedAmount(sms).toString()
                            val date = formatDate(item.date)
                            val model = Expense(amount,date,"General","Online Expense",item.date.toString())
                            if(!amount.isNullOrEmpty() && !sms.contains("credited")){
                                FirebaseDatabase.getInstance().reference.child("users").child(FirebaseAuth.getInstance().currentUser!!.uid)
                                    .child("onlineExpenses").child(item.date.toString()).setValue(model)
                                FirebaseDatabase.getInstance().reference.child("users").child(FirebaseAuth.getInstance().currentUser!!.uid)
                                    .child("Expenses").child(item.date.toString()).setValue(model)
                            }
                        }
                    } catch (e: JSONException) {
                        e.printStackTrace()
                    }
                },
                Response.ErrorListener { error ->
```

Fig 5.8 Accessing API

In this manner we first created machine learning model in jupyter notebook using naïve bayes algorithm and then using that model we create a flask API which we deployed on railway platform. And at the end with the help of volley library we accessed that API in our android app.

5.6 Testing

Testing an expense tracker application ensures that it functions correctly and accurately tracks and manages user expenses, which ultimately leads to better financial management for the user.

i. **GUI testing:** Graphical User Interface (GUI) testing is the one of the mechanism in which user interface developed System Under some graphical rules. GUI testing includes checking various controls- menus, buttons, icons, dialog boxes and windows etc. Proposed system is tested for user inputs against different modules, validations are done.

ii. **Unit Testing**: It is the testing of individual software units of the application .it is done after the complexion of an individual unit before integration. Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results .

iii. **Integration Testing:** Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

iv. **System testing:** System testing tests a completely integrated system to verify that the system meets its requirements. For example, a system test might involve testing a logon interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff.

Our smart expense tracker application was tested for various functionalities and it passed these tests successfully. At first we tested the user interface of the app and checked whether all the buttons and views are working or not. Then we tested the user authentication functionality to register new user. After that we tested the app for adding expense and storing it on firebase database. We also tested whether app is categorizing the bank messages correctly or not. After that the expense tracker was tested for showing pie charts with correct information. Loading the data from firebase database and feeding it to proper views was also tested. Functionality test of the application was conducted for all the function of the app and it passed successfully.

6. CONCLUSION

We developed an Android application that helps users track their daily expenses, by allowing them to manually enter their expenses or automatically detect and read expense messages from the bank using a Naive Bayes algorithm. The algorithm was trained using supervised machine learning to accurately classify SMS as "Bank SMS" or "Normal SMS". The performance of the algorithm was evaluated using precision and accuracy metrics, which were found to be 97.67% and 99.33% respectively.

Overall, the proposed system has the potential to help people manage their finances better and make informed decisions about their spending habits. By automatically detecting and reading expense messages from the bank, the system can help users keep track of their expenses more accurately and efficiently. Moreover, the ability to classify SMS messages as "Bank SMS" or "Normal SMS" can help users filter out irrelevant messages and focus on the ones that are relevant to their financial management.

In conclusion, the system described in the text is a promising application that has the potential to improve financial awareness and responsibility among users. By leveraging modern software development tools and machine learning algorithms, the system provides a practical and efficient way for users to track their expenses and make informed financial decisions.

FUTURE WORK

Here are some potential future work suggestions for this expense tracking Android application:

1. **Expand the feature set:** Currently, the application only allows users to track expenses by entering them manually or by automatically detecting and reading expense messages from the bank. To make the application more comprehensive, additional features could be added, such as the ability to categorize expenses, track income, and set budget limits for different categories.

2. **Improve the algorithm's performance:** While the Naive-Bayes algorithm used in the project achieved high precision and accuracy rates, there is always room for improvement. The algorithm could be further optimized by trying different variants of Naive-Bayes or other classification algorithms.

3. **Incorporate machine learning for categorization:** In addition to detecting bank SMS messages, the application could also use machine learning to automatically categorize expenses based on their type, such as food, travel, or entertainment. This could be achieved by training a supervised machine learning model on a labeled dataset of expenses.

4. **Add data visualization:** To help users better understand their spending patterns and identify areas where they can cut back on expenses, the application could incorporate data visualization features. This could include graphs and charts that display expense data over time or by category.

5. **Expand the database options:** Currently, the application uses Firebase as its online database. However, other database options could be explored to offer users more choices in how their data is stored and accessed.

USER MANUAL

Step 1: Install and open the expense tracker app in your android device.

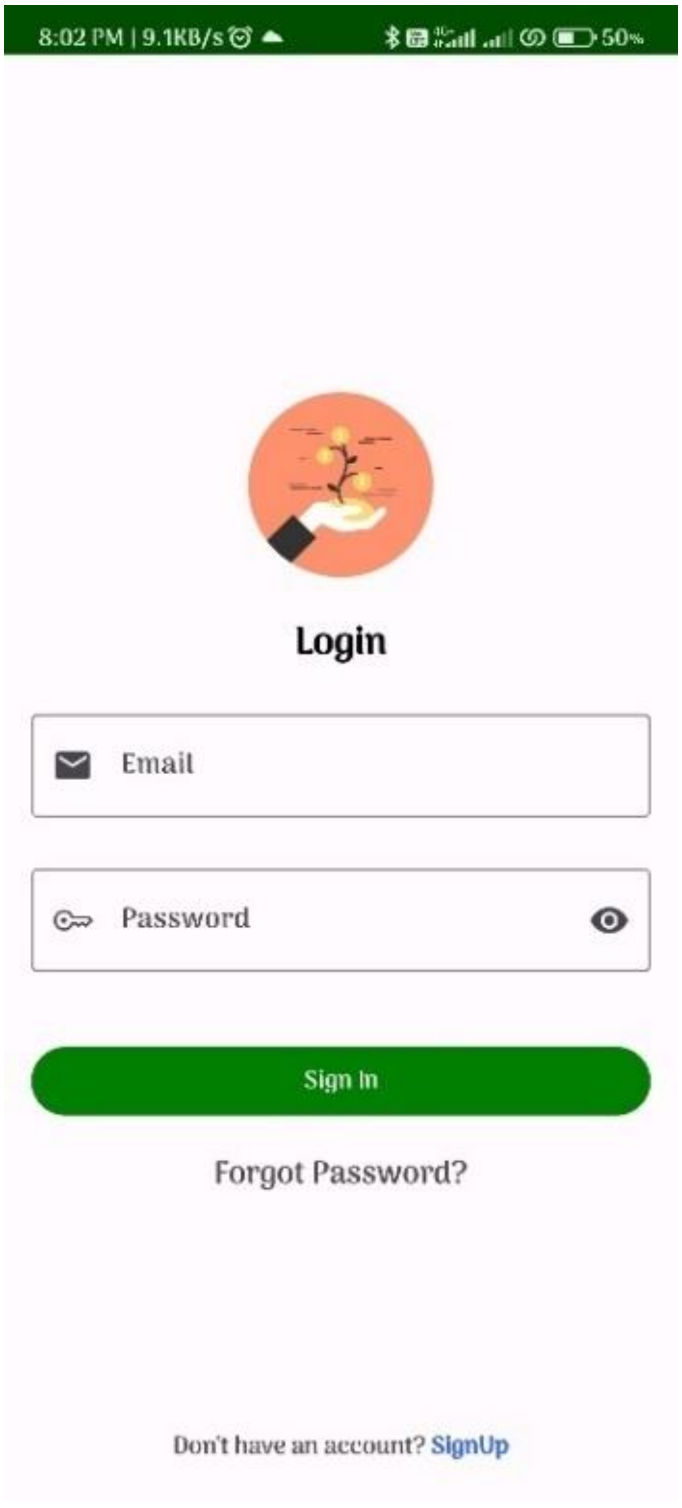


Image 1. App Installed

Step 2: Register yourself on the app with email and password.

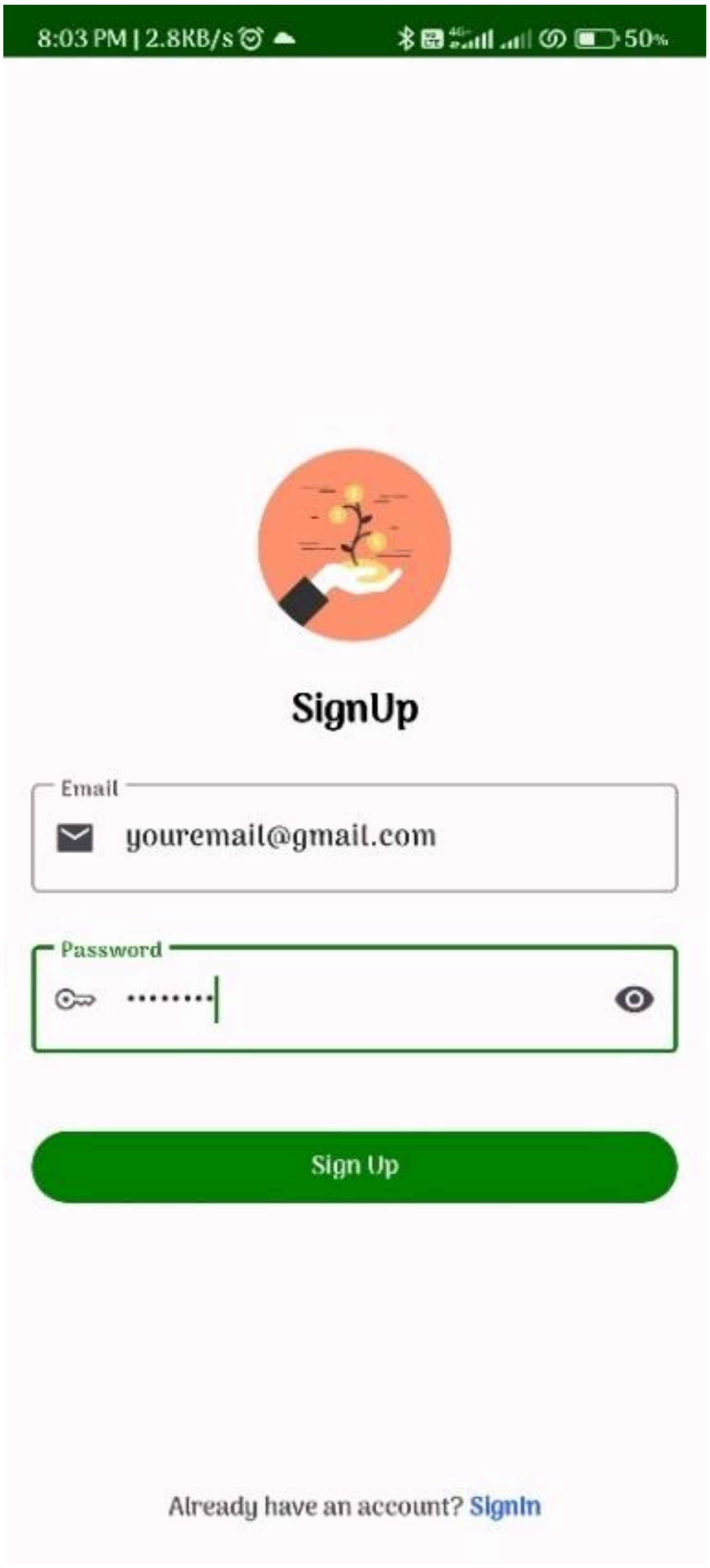
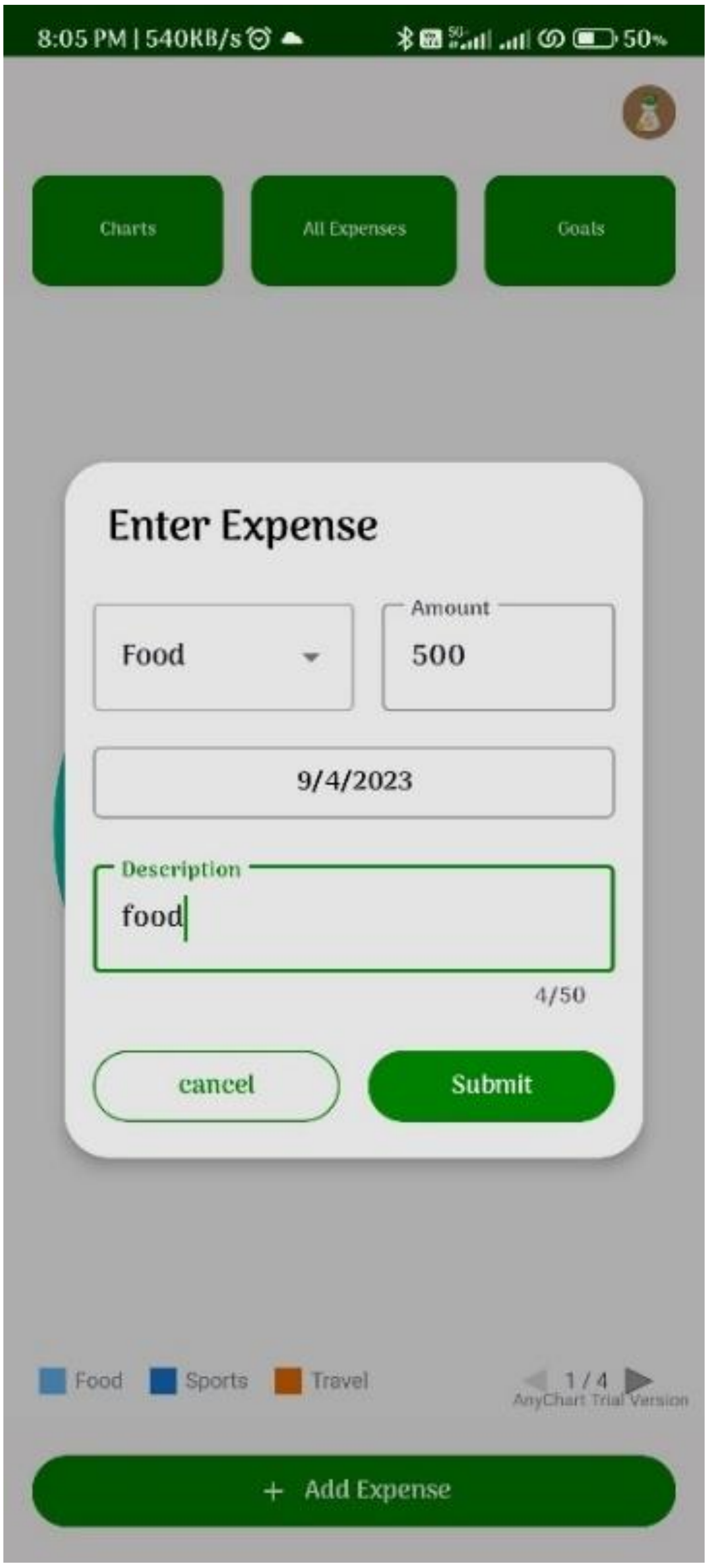


Image 2. User Authentication

Step 3: After successful registration you will be redirected to the Home. By clicking on add expense button user can add new expense.



Enter Expense

Food

Amount 500

9/4/2023

Description food

4/50

Image 3. Add Expense

Step 4: By clicking on charts button from home screen user can see expenses for week, month, year, and of all time.

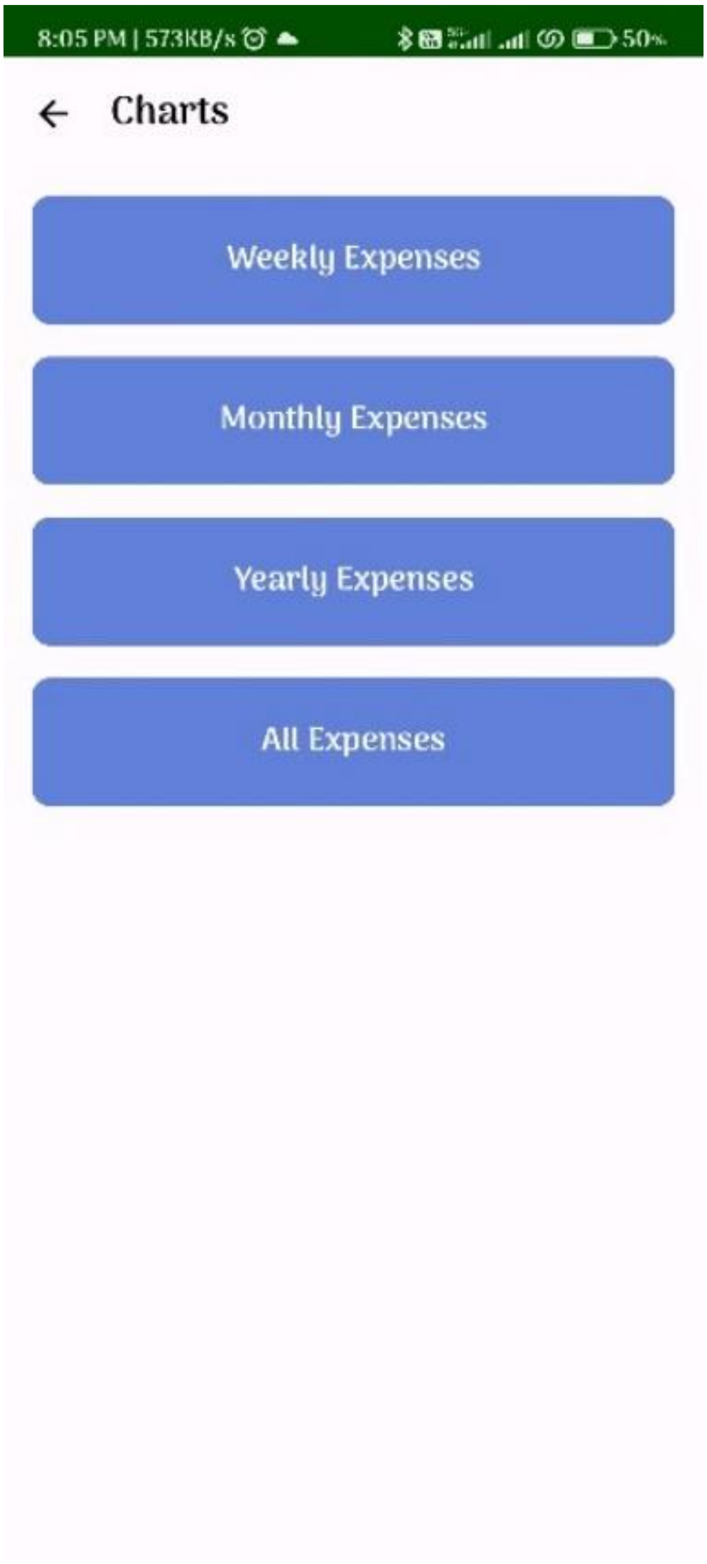


Image 4. View Charts

Step 5: User can see pie charts for various categories.

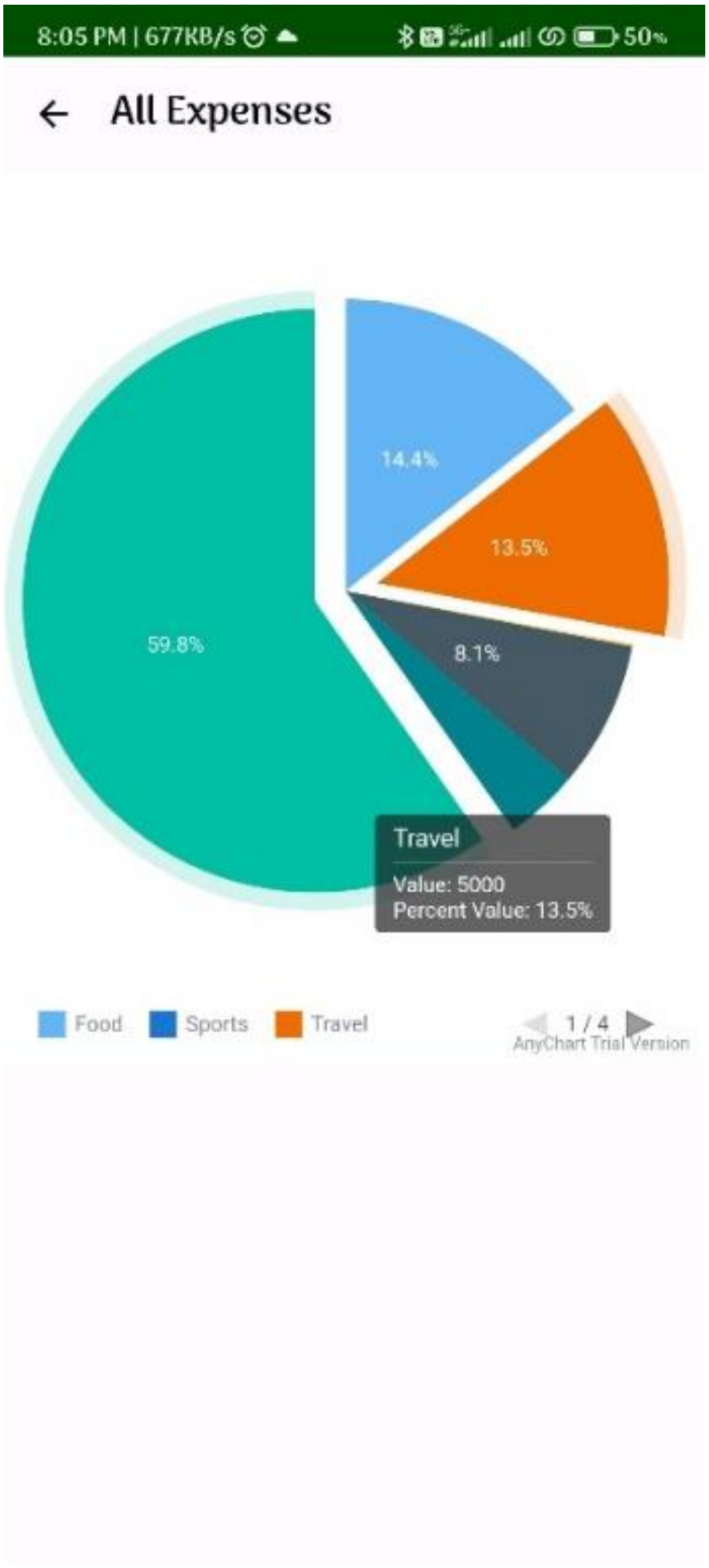
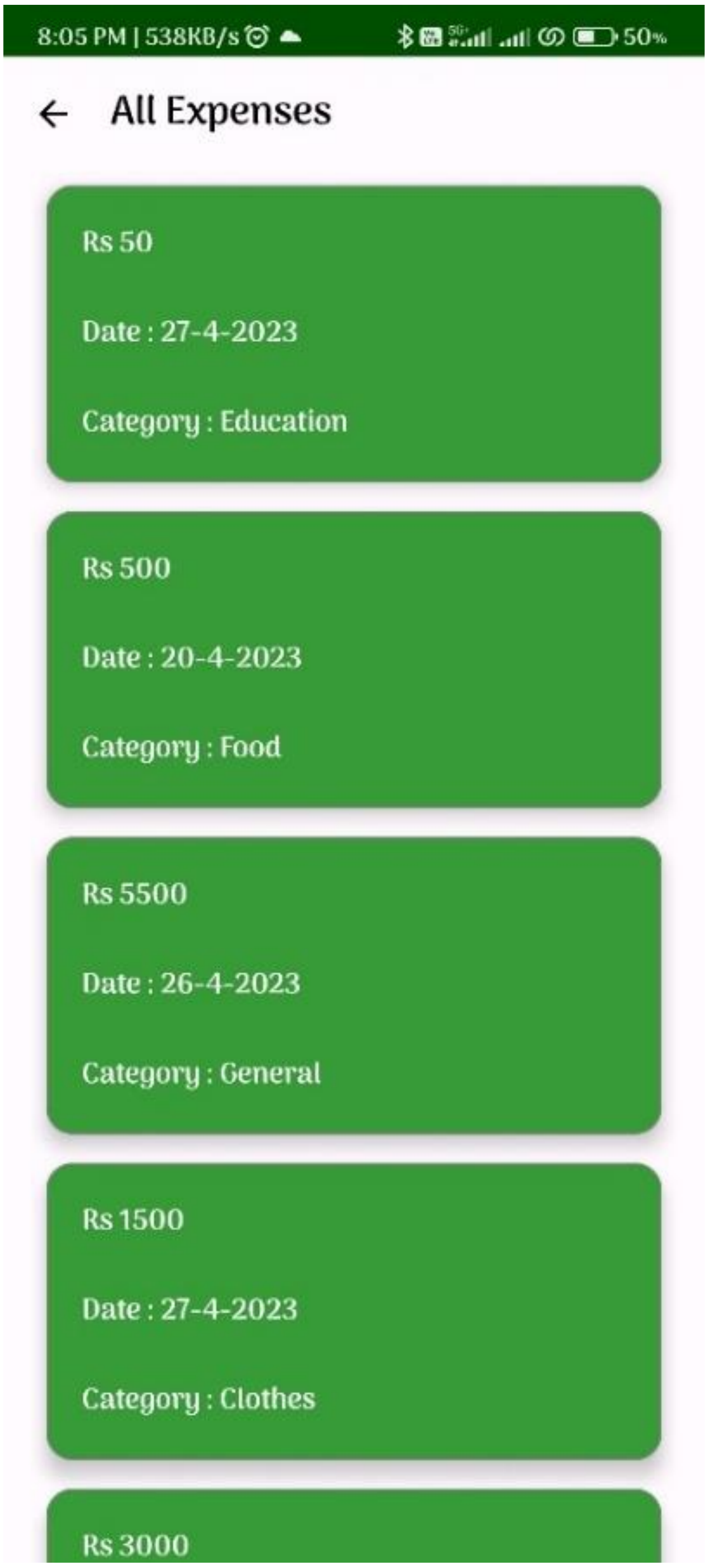


Image 5. View Pie Chart

Step 6: View all expenses by clicking on all expenses button from home screen.


8:05 PM | 538KB/s 50%

All Expenses

Rs 50

Date : 27-4-2023

Category : Education

Rs 500

Date : 20-4-2023

Category : Food

Rs 5500

Date : 26-4-2023

Category : General

Rs 1500

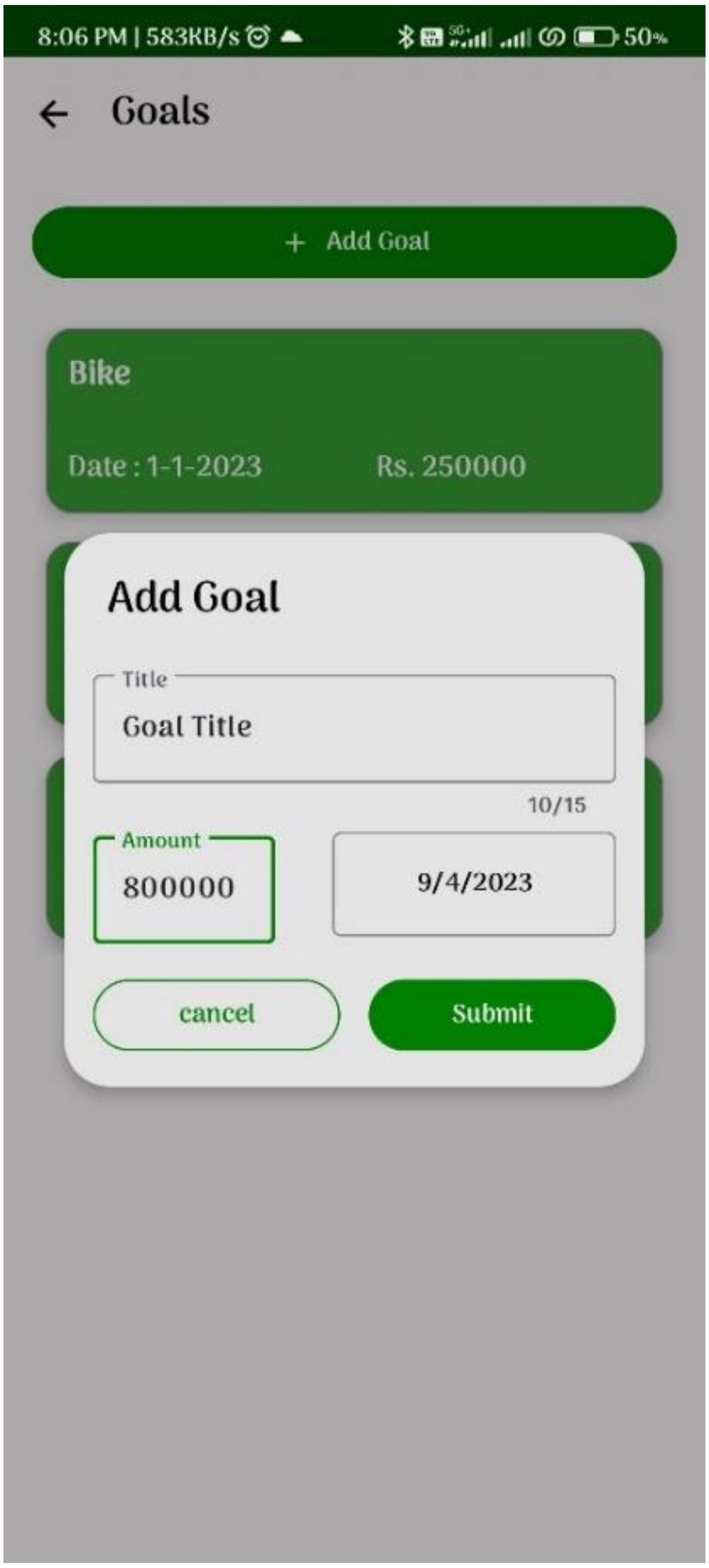
Date : 27-4-2023

Category : Clothes

Rs 3000

Image 6. View All Expenses

Step 7: By clicking on goals button from home screen user can see goals. And by clicking on add goal button user can add goal.

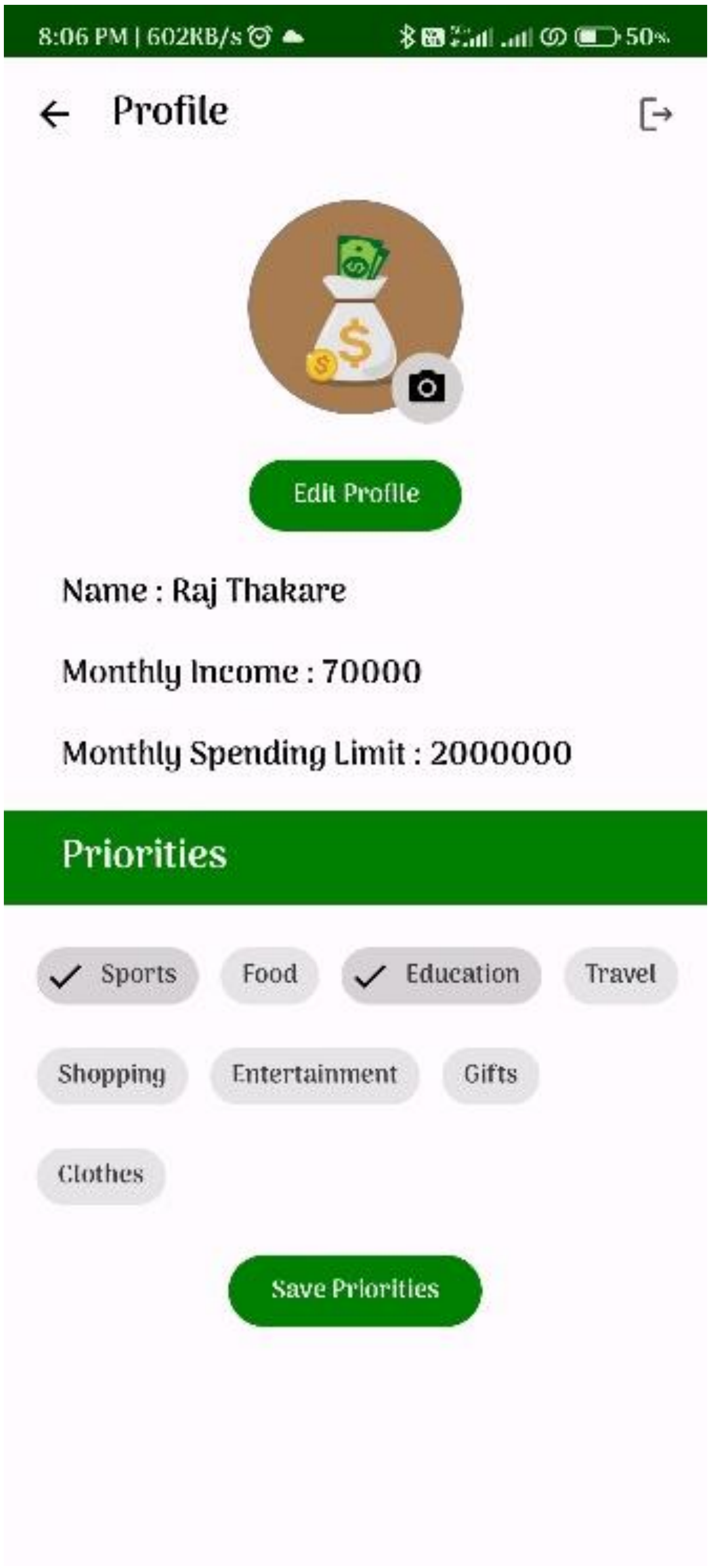


Bike
Date : 1-1-2023 Rs. 250000

Add Goal
Title: Goal Title
10/15
Amount: 800000
9/4/2023

Image 7. Add Goal

Step 8: By clicking on profile icon at top right corner user can go to profile.


8:06 PM | 602KB/s 50%

Profile

Edit Profile

Name : Raj Thakare

Monthly Income : 70000

Monthly Spending Limit : 2000000

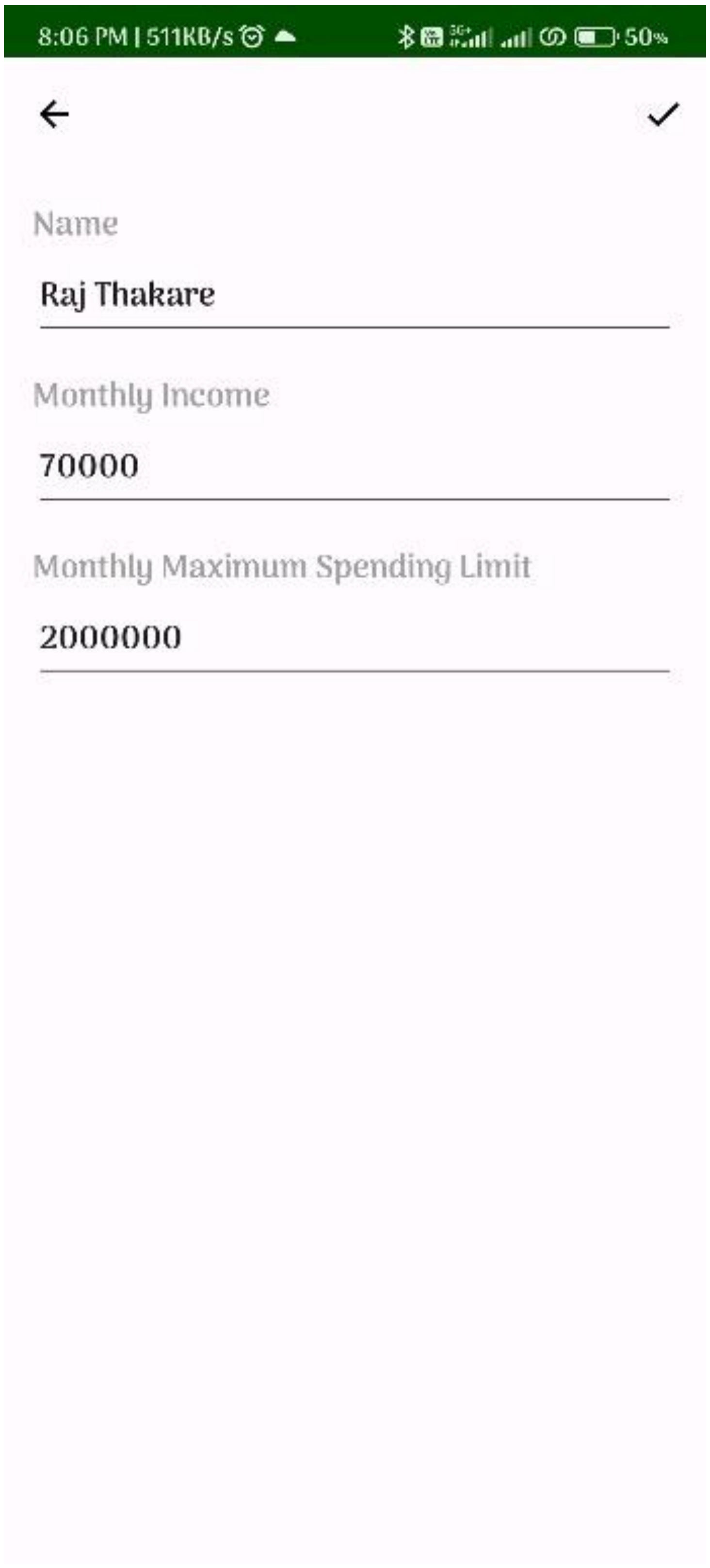
Priorities

✓ Sports | Food | ✓ Education | Travel | Shopping | Entertainment | Gifts | Clothes

Save Priorities

Image 8. Profile

Step 9: Clicking on edit profile button users can update their profile.



Name

Raj Thakare

Monthly Income

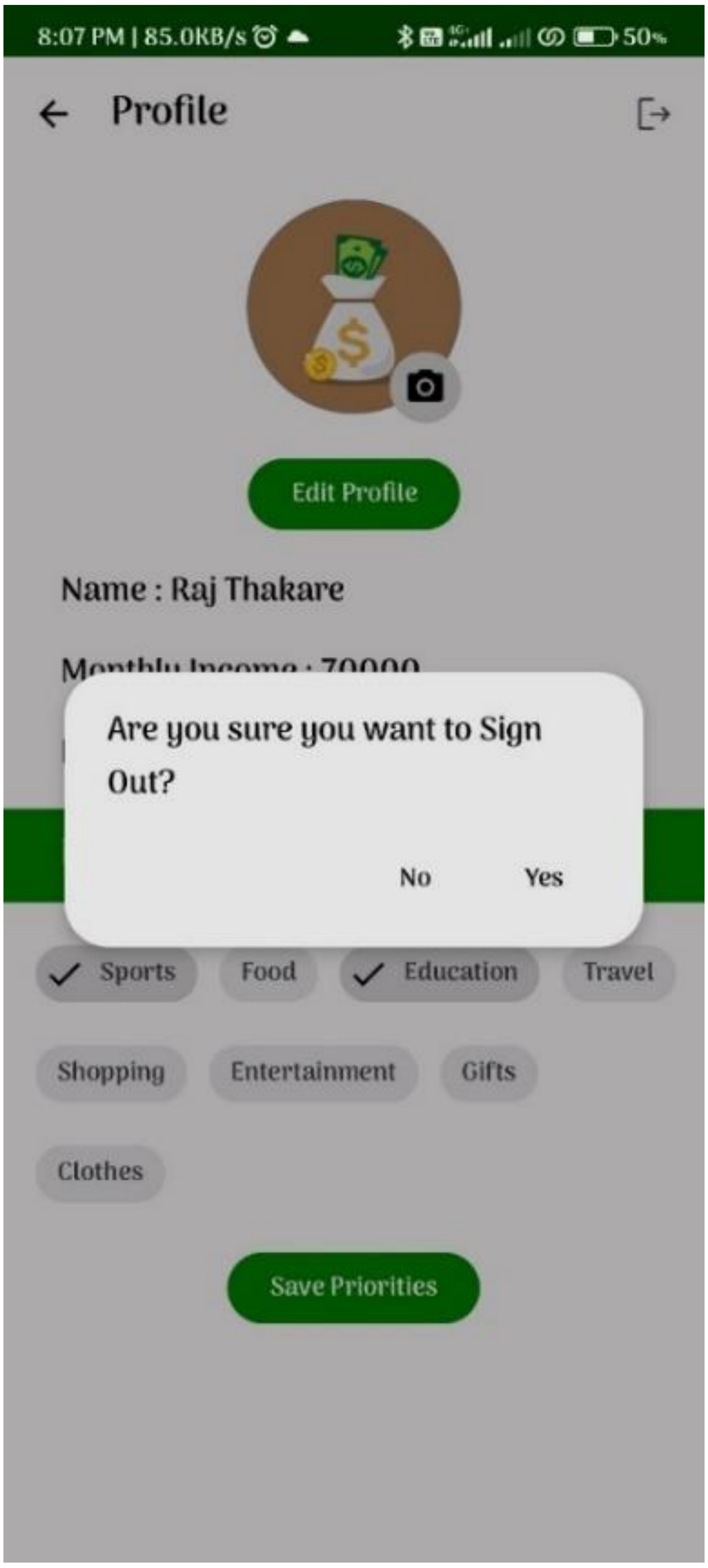
70000

Monthly Maximum Spending Limit

2000000

Image 9. Edit Profile

Step 10: Log out button at top right corner can be used to sign out from app.


8:07 PM | 85.0KB/s

Profile

Edit Profile

Name : Raj Thakare

Are you sure you want to Sign Out?

No Yes

Image 10. Log Out

REFERENCES

[1] Ma, T., Yamamori, K., & Thida, A. (2020). A comparative approach to Naive Bayes classifier and Support Vector Machine for email spam classification. In 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE). IEEE.

[2] Ziyan Mohammed, Mohammed Farhaz J A, Mohammed Irshad M P, Mustafa Basthikodi, and Ahmed Rimaz Faizabadi. "A Comparative Study for Spam Classifications in Email Using Naïve Bayes and SVM Algorithm." Journal of Emerging Technologies and Innovative Research (JETIR) 6, no. 5 (May 2019).

[3] Gupta, M., Bakliwal, A., Agarwal, S., & Mehndiratta, P. (2018). A Comparative Study of Spam SMS Detection using Machine Learning Classifiers. In Proceedings of 2018 Eleventh International Conference on Contemporary Computing (IC3) (pp. 1-5). Noida, India: IEEE.

[4] Raza, M., Jayasinghe, N.D., & Muslam, M.M.A. (2021). A Comprehensive Review on Email Spam Classification using Machine Learning Algorithms. In 2021 International Conference on Information Networking (ICOIN) (pp. 1-6). IEEE.

[5] Bhuiyan, H., Ashiquzzaman, A., Juthi, T. I., Biswas, S., & Ara, J. (Year of publication). A Survey of Existing E-Mail Spam Filtering Methods Considering Machine Learning Techniques. Journal Name, Volume Number(Issue Number), Global Journals, 2018.

[6] Katpatal, R. & Junnarkar, A. (2018). An Efficient Approach of Spam Detection in Twitter. In Proceedings of the International Conference on Inventive Research in Computing Applications (ICIRCA 2018).

[7] Taloba, A. I., & Ismail, S. S. I. (2019). An Intelligent Hybrid Technique of Decision Tree and Genetic Algorithm for E-Mail Spam Detection. In 2019 IEEE Ninth International Conference on Intelligent Computing and Information Systems (ICICIS) (pp. 1-6). IEEE.

[8] Pal, K., & Patel, B. V. (2020). Automatic Multiclass Document Classification of Hindi Poems using Machine Learning Techniques. Study presented at the International Conference for Emerging Technology (INCET), Belgaum, India, June 5-7, 2020.

[9] De Vries, V. (2020). Classification of Aviation Safety Reports using Machine Learning. Aerospace Operations Safety Institute, Royal Dutch Aerospace Centre, Amsterdam, The Netherlands. Vincent.de.Vries@nlr.nl. 2020 IEEE.

[10] Phani Prasad, J., & Venkatesham, T. (2021). Classification of E-mail Spam with Supervised Machine Learning-Naïve Bayesian Classification. Advances and Applications in Mathematical Sciences, 2021.

[11] Shrivastava, A., & Dubey, R. (2018). Classification of Spam Mail using different machine learning algorithms. Study presented at the 2018 IEEE.

[12] Reddy, K.N. and Kakulapati, V. (Year). Classification of Spam Messages using Random Forest Algorithm./Journal of Xidian University, December 2021

[13] Krishnaveni, N., & Radha, V. (2021). Comparison of Naive Bayes and SVM classifiers for detection of spam SMS using natural language processing. International Journal of Scientific Research in Computer Science and Engineering, 9(2), 2260-2267. DOI: 10.21917/ijsc.2021.0323.

[14] Bharath, G., Manikanta, K.J., Bhanu Prakash, G., Sumathi, R., & Chinnasamy, P. (2021). Detecting Fake News Using Machine Learning Algorithms. Study presented at the 2021 International Conference on Computer Communication and Informatics (ICCCI-2021), Jan. 27 – 29, 2021.

[15] Tanvir, A.-A., Mahir, E.M., Akhter, S., & Huq, M.R. (2019). Detecting Fake News using Machine Learning and Deep Learning Algorithms. In 2019 7th International Conference on Smart Computing & Communications (ICSCC). East West University, Dhaka, Bangladesh.

[16] Alom, Z., Carminati, B., & Ferrari, E. (2018). Detecting spam accounts on Twitter. In 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) (pp. 856-863). IEEE.

[17] Gupta, P., Dubey, R. K., & Mishra, S. (2019). Detecting Spam Emails/Sms Using Naive Bayes and Support Vector Machine. International Journal of Scientific & Technology Research, 8(11), ISSN 2277-8616.

[18] Singh, M., Pamula, R., & Shekhar, S. K. (2018). Email Spam Classification by Support Vector Machine. In 2018 International Conference on Computing, Power and Communication Technologies (GUCON) (pp. 556-560). IEEE.

[19] Shradhanjali, Verma Toran. "E-Mail Spam Detection and Classification Using SVM and Feature Extraction." International Journal of Advance Research, Ideas and Innovations in Technology, vol. 3, no. 3, 2017, pp. 1491. ISSN: 2454-132X. Available online at www.ijariit.com.

[20] Agarwal, K., & Kumar, T. (2018). Email Spam Detection using integrated approach of Naïve Bayes and Particle Swarm Optimization. In Proceedings of the Second International Conference on Intelligent Computing and Control Systems (ICICCS 2018)

[21] Kumar, N., Sonowal, S., & Yadav, N. (2020). Email Spam Detection Using Machine Learning Algorithms. In Proceedings of the Second International Conference on Inventive Research in Computing Applications (ICIRCA-2020)

[22] Tope, M. (2019). Email Spam Detection using Naive Bayes Classifier. International Journal of Scientific Development and Research (IJSDR), 4(6), 1-3. 2019.

[23] Azam, M., Ahmed, T., Sabah, F., & Hussain, M. I. (2018). Feature Extraction based Text Classification using K-Nearest Neighbor Algorithm. IJCSNS International Journal of Computer Science and Network Security, 18(12), 95. Manuscript received December 5, 2018; Manuscript revised December 20, 2018.

[24] Deepa, R., & Lalwani, K. N. (2019). Image classification and text extraction using machine learning. In Proceedings of the Third International Conference on Electronics Communication and Aerospace Technology [ICECA 2019] (pp. 218-222). IEEE Conference Record # 45616; IEEE Xplore ISBN: 978-1-7281-0167-5. Loyola-ICAM College of Engineering and Technology, Chennai, India.

[25] Kurniawan, S., & Budi, I. (n.d.). Indonesian Tweets Hate Speech Target Classification using Machine Learning. Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia. Retrieved May 21, 2021, from IEEE Xplore database.

[26] Murugavel, U. and Santhi, R. (2020). K-Nearest Neighbor Classification of E-Mail Messages for Spam Detection. Indian Journal of Science and Technology, 13(22), 2218. doi: 10.21917/ijsc.2020.0316.

[27] Siddique, Z. B., Khan, M. A., Ud Din, I., Almogren, A., Mohiuddin, I., & Nazir, S. (2021). Machine Learning-Based Detection of Spam Emails. Scientific Programming, 2021, 6508784. 2021.

[28] Nagre, S. (2018). Mobile SMS Spam Detection using Machine Learning Techniques. Journal of Emerging Technologies and Innovative Research (JETIR), 5(12), 548. Retrieved from http://www.jetir.org (ISSN-2349-5162), 2018.

[29] Akanda, W. and Uddin, A. (2021), "Multi-label Bengali article classification using ML-KNN algorithm and Neural Network," in Proceedings of the International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD).

[30] C. M. Suneera and Jay Prakash, "Performance Analysis of Machine Learning and Deep Learning Models for Text Classification," 2020 IEEE 17th India Council International Conference (INDICON), pp. 1-5, 2020.

[31] Nandhini, S. & Jeen Marseline, K. S. (2020). Performance Evaluation of Machine Learning Algorithms for Email Spam Detection. Study presented at the 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE). IEEE. ISBN: 978-1-7281-4142-8.

[32] Sethi, P., Bhandari, V., & Kohli, B. (Year). SMS spam detection and comparison of various machine learning algorithms. Journal/Conference Name, Volume(Issue), 2017 IEEE.

[33] Chaurasia, N., Bharali, P., & Naresh, R. (2021). SMS Spam Detection using Supervised Learning. Turkish Journal of Computer and Mathematics Education, 12(11), 2021.

[34] Selvapattu, P. and Patil, P.V. (2020). SMS Spam Detection. IJISET - International Journal of Innovative Science, Engineering & Technology, 7(6), 194. ISSN (Online) 2348 – 7968. Impact Factor (2020).

[35] Navaney, P., Dubey, G., & Rana, A. (Year). SMS Spam Filtering using Supervised Machine Learning Algorithms. IEEE, 2018.

[36] Deshmukh, N., Dhumal, V., Gavasane, N., & Jadhav, S. V. (2021). SPAM DETECTION BY USING KNN ALGORITHM TECHNIQUES. International Journal of Advance Scientific Research and Engineering Trends, 6(5), 128. doi:10.51319/2456-0774.2021.5.0029.

[37] Pooja and Komal Kumar Bhatia. "Spam Detection using Naive Bayes Classifier." International Journal of Computer Sciences and Engineering Open Access, vol. 6, no. 7, July 2018, pp. 1-6, E-ISSN: 2347-2693.

[38] Vijay and Shubham Kumar. "Spam SMS Detection Using Naive Bayes Classifier". International Journal of Scientific Research and Engineering Development. Volume 4, Issue 1, Jan-Feb 2021, pp. 561. ISSN: 2581-7175.

[39] Santoshi, K.U., Bhavya, S.S., Sri, Y.B., & Venkateswarlu, B. (2021). Twitter Spam Detection Using Naïve Bayes Classifier. In Proceedings of the Sixth International Conference on Inventive Computation Technologies [ICICT 2021], (pp. 223-227).

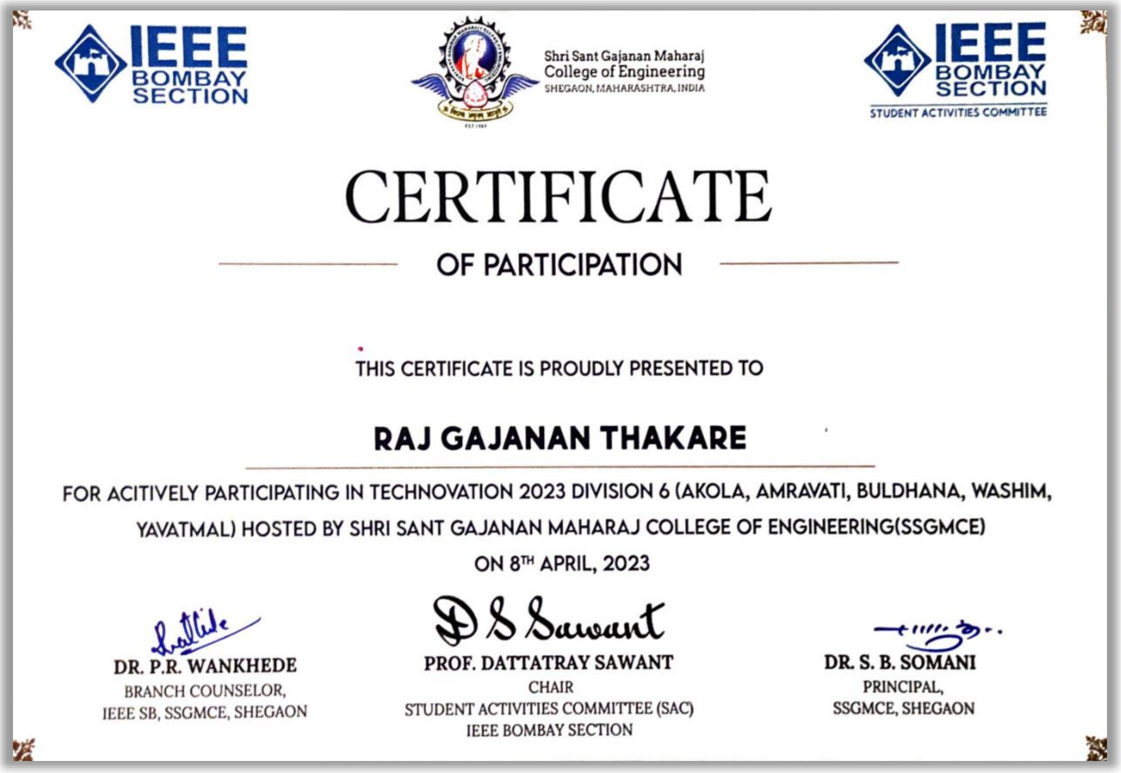
[40] Moldagulova, A., & Sulaiman, R.B. (2017). Using KNN Algorithm for Classification of Textual Documents. In 2017 8th International Conference on Information Technology (ICIT) (pp. 1-5). IEEE. ISBN 978-1-5090-6332-1.

DISSEMINATION OF WORK

Title:	Machine Learning Based Expense Tracker Application For Personal Finance Management.
Authors:	Mr. Raj Thakare, Mr. Ninad Thakare, Mr. Raj Sangtani, Mr. Shubham Bondre, Dr. A S Manekar.
Publisher:	SGGVS-AIMT Spring International Conference on Advancement in Science, Management, Technology, Social Science and Humanities (ICSMTSH)
ISSN No:	
Volume:	
DOI:	
Issue:	

Project Competition: IEEE Bombay Technovation 2023 Division 6 hosted by SSGMCE Shegaon.

Certificates:



IEEE BOMBAY SECTION

Shri Sant Gajanan Maharaj College of Engineering SHEGAON, MAHARASHTRA, INDIA

IEEE BOMBAY SECTION STUDENT ACTIVITIES COMMITTEE

CERTIFICATE

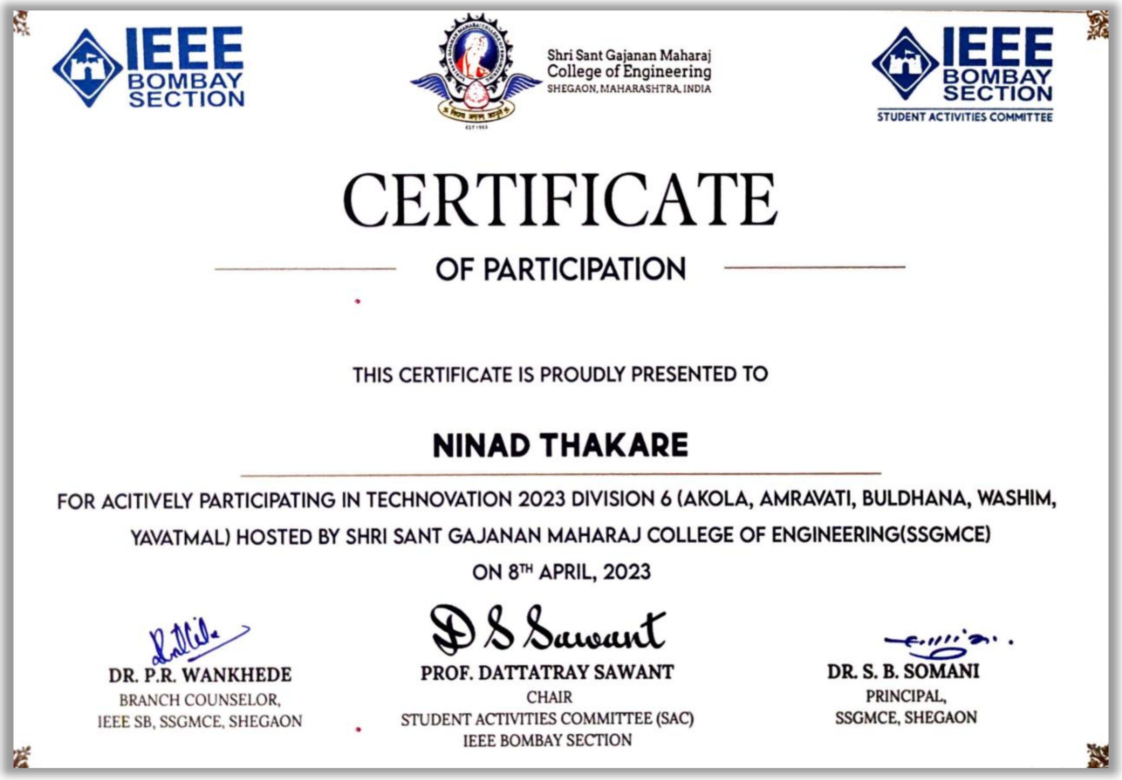
OF PARTICIPATION

THIS CERTIFICATE IS PROUDLY PRESENTED TO

RAJ GAJANAN THAKARE

FOR ACITIVELY PARTICIPATING IN TECHNOVATION 2023 DIVISION 6 (AKOLA, AMRAVATI, BULDHANA, WASHIM, YAVATMAL) HOSTED BY SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGINEERING(SSGMCE) ON 8TH APRIL, 2023

[illegible]	D S Sawant	[illegible]
DR. P.R. WANKHEDE BRANCH COUNSELOR, IEEE SB, SSGMCE, SHEGAON	PROF. DATTATRAY SAWANT CHAIR STUDENT ACTIVITIES COMMITTEE (SAC) IEEE BOMBAY SECTION	DR. S. B. SOMANI PRINCIPAL, SSGMCE, SHEGAON



IEEE BOMBAY SECTION

Shri Sant Gajanan Maharaj College of Engineering
SHEGAON, MAHARASHTRA, INDIA

IEEE BOMBAY SECTION
STUDENT ACTIVITIES COMMITTEE

CERTIFICATE

OF PARTICIPATION

THIS CERTIFICATE IS PROUDLY PRESENTED TO

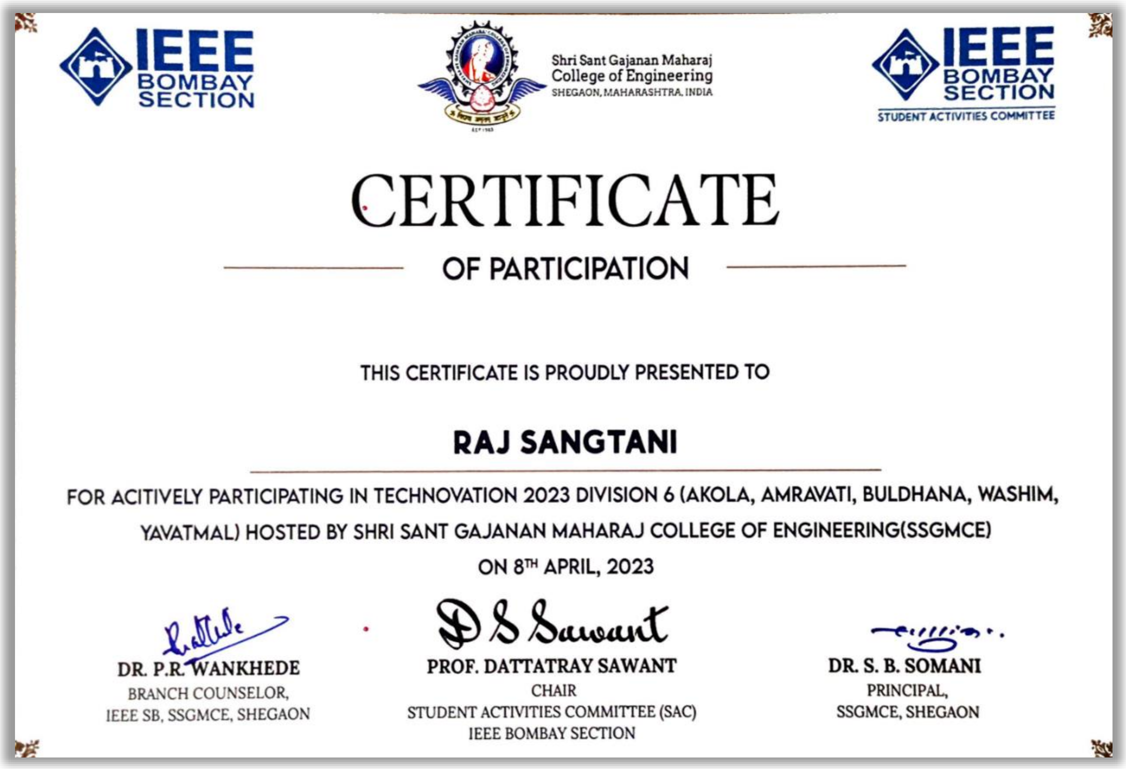
NINAD THAKARE

FOR ACITIVELY PARTICIPATING IN TECHNOVATION 2023 DIVISION 6 (AKOLA, AMRAVATI, BULDHANA, WASHIM, YAVATMAL) HOSTED BY SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGINEERING(SSGMCE) ON 8TH APRIL, 2023

[illegible]
DR. P.R. WANKHEDE
BRANCH COUNSELOR,
IEEE SB, SSGMCE, SHEGAON

D S Sawant
PROF. DATTATRAY SAWANT
CHAIR
STUDENT ACTIVITIES COMMITTEE (SAC)
IEEE BOMBAY SECTION

[illegible]
DR. S. B. SOMANI
PRINCIPAL,
SSGMCE, SHEGAON



IEEE BOMBAY SECTION

Shri Sant Gajanan Maharaj College of Engineering
SHEGAON, MAHARASHTRA, INDIA

IEEE BOMBAY SECTION
STUDENT ACTIVITIES COMMITTEE

CERTIFICATE

OF PARTICIPATION

THIS CERTIFICATE IS PROUDLY PRESENTED TO

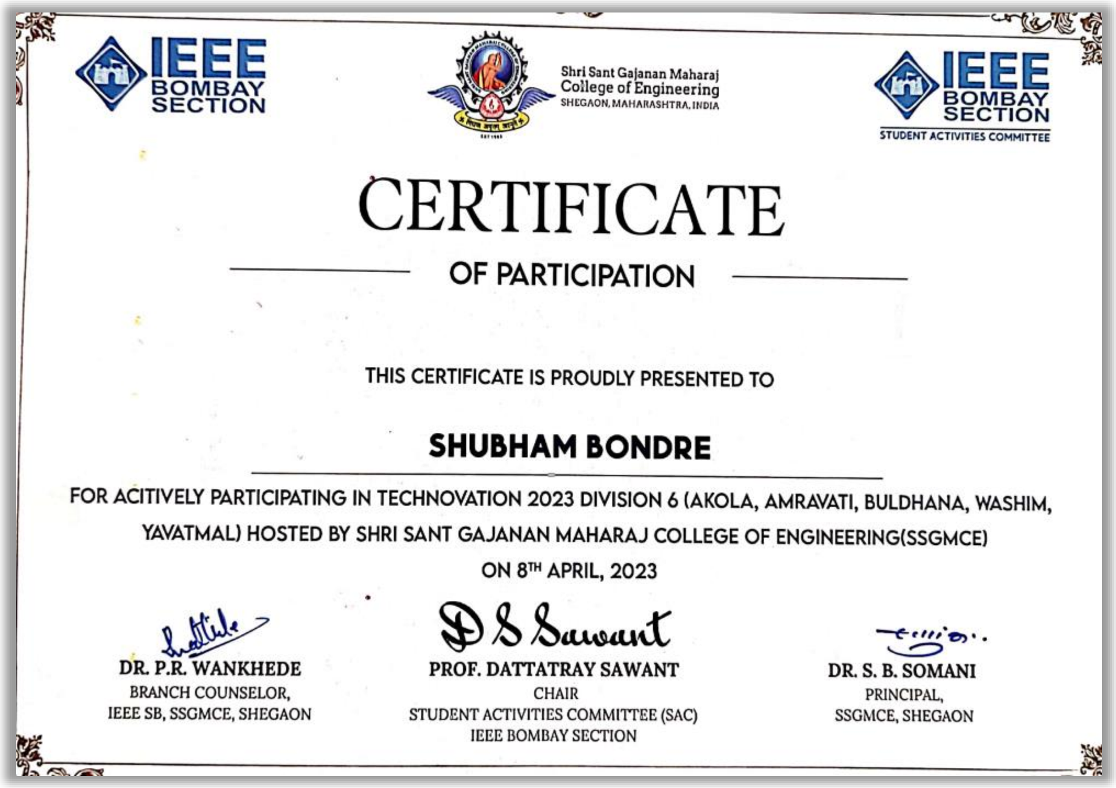
RAJ SANGTANI

FOR ACITIVELY PARTICIPATING IN TECHNOVATION 2023 DIVISION 6 (AKOLA, AMRAVATI, BULDHANA, WASHIM, YAVATMAL) HOSTED BY SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGINEERING(SSGMCE) ON 8TH APRIL, 2023

[illegible]
DR. P.R. WANKHEDE
BRANCH COUNSELOR,
IEEE SB, SSGMCE, SHEGAON

D S Sawant
PROF. DATTATRAY SAWANT
CHAIR
STUDENT ACTIVITIES COMMITTEE (SAC)
IEEE BOMBAY SECTION

[illegible]
DR. S. B. SOMANI
PRINCIPAL,
SSGMCE, SHEGAON



IEEE BOMBAY SECTION

Shri Sant Gajanan Maharaj College of Engineering SHEGAON, MAHARASHTRA, INDIA

IEEE BOMBAY SECTION STUDENT ACTIVITIES COMMITTEE

CERTIFICATE

OF PARTICIPATION

THIS CERTIFICATE IS PROUDLY PRESENTED TO

SHUBHAM BONDRE

FOR ACITIVELY PARTICIPATING IN TECHNOVATION 2023 DIVISION 6 (AKOLA, AMRAVATI, BULDHANA, WASHIM, YAVATMAL) HOSTED BY SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGINEERING(SSGMCE) ON 8TH APRIL, 2023

[illegible]	D S Sawant	[illegible]
DR. P.R. WANKHEDE	PROF. DATTATRAY SAWANT	DR. S. B. SOMANI
BRANCH COUNSELOR,	CHAIR	PRINCIPAL,
IEEE SB, SSGMCE, SHEGAON	STUDENT ACTIVITIES COMMITTEE (SAC)	SSGMCE, SHEGAON
	IEEE BOMBAY SECTION	

INFORMATION OF MEMBERS

Name: Raj Thakare

Email: rajthakare237@gmail.com

Mobile: 9834812916

Address: Shahapur, Akola, Maharashtra, 444401.

Name: Ninad Thakare

Email: ninadthakare1305@gmail.com

Mobile: 9067113074

Address: Sushil Layout Moha Fata Yawatmal, Maharashtra, 445001.

Name: Raj Sangtani

Email: rajsangtani1911@gmail.com

Mobile: 8625008218

Address: Shivaji Nagar, Mehkar, Maharashtra, 443301.

Name: Shubham Bondre

Email: sbondre02@gmail.com

Mobile: 9637967174

Address: Ambewadi, Sindkhed Raja, Maharashtra, 443202.