UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Informatics Institute of Technology

Software Development Group Project

Implementation Report

5COSC021C

Module Leader-Mr.Banuka Athuraliya

SE _-41

Name	IIT Mail	IIT ID	UOW ID
Dumindu Gamage(TL)	dumindu.20221168.ac.lk	20221168	w1953846
Rajeen Balasooriya	sachintha.20220175@iit.ac.lk	20220175	w1953138
Sakith Dissanayake	sakthi.20221409@iit.ac.lk	20221409	w1956131
Savin Pathirana	savin.20222009@iit.ac.lk	20221409	w1985684
Kavindu Hasaranga Yapa	kavindu.20210673@iit.ac.lk	20210673	w1953856
Himan Welgama	himan.20221109@iit.ac.lk	20221109	w1998723

Date of Submission-2024.03.18

I. Declaration

We declare that this report in its entirety, together with all of the artifacts to it, is our original work and hasn't been submitted for credit towards any other degree, certificate, or academic qualification to any other institution, college, or organization.

Name	IIT Mail	IIT ID	UOW ID
Dumindu Induwara Gamage(TL)	dumindu.20221168@iit.ac.lk	20221168	w1953846
Rajeen Balasooriya	Sachintha.20220175@iit.ac.lk	20220175	w1953138
Himan Welgama	himan.20221109@iit.ac.lk	20221109	w1998723
Kavindu Hasaranga Yapa	kavindu.20210673@iit.c.lk	20210673	w1953856
Savin Patharana	savin.20222009@iit.ac.lk	20221409	w1985684
Sakith Dissanayake	sakthi.20221409@iit.ac..lk	20221409	w1956131

Date - 18/03/2024

II. Abstract

Inflation is a significant issue affecting many people today, making life tougher financially. Often, individuals overspend due to tempting offers or recommendations, depleting their hard-earned money unnecessarily. This leaves them struggling until their next paycheck arrives. Additionally, when shopping online, they may overlook price variations for the same product across different platforms, resulting in overspending. To address these challenges, this report proposes the development of a budget planning application. This app would offer guidance on smarter money management, provide alerts for important events such as price drops, and even forecast future prices. Moreover, it would feature a user-friendly design to ensure accessibility for all.

III. Acknowledgment

We would like to express our sincere gratitude to all who have contributed to the completion of this report.

First and foremost, we extend our heartfelt thanks to our module leader Mr. Banu Athuraliya and our supervisor Mr. Saadh Jawwadh for their invaluable guidance, continuous encouragement and insightful feedback throughout the course of this project. Their expertise and encouragement have been instrumental in shaping this work

Also, we would like to express our gratitude towards every supervisor who is a part of the SDGP module and who guides us to the right path.

Last but not least, we owe a debt of gratitude to our family and friends for their unwavering encouragement and understanding during this endeavor.

IV. Table of Content

V. List of Figures

VI. List of Tables

VII. Abbreviations Table

Abbreviation	Definition
GIT	Global Information Tracker
NLP	Natural Language Processing
IOS	iPhone Operating System
UI	User Interface
NLTK	Natural Language Toolkit
POS	Part of Speech
NER	Named Entity Recognition
CSV	Comma - Separated Values
ROC	Receiver Operating Characteristics
SVM	Support Vector Machine
JSON	JavaScript Object Notation
API	Application Programming Interface
LSTM	Long Short - Term Memory
NFLX	Netflix
AAPL	Apple Inc. Common Stock
ORM	Object Relational Mapper
SQL	Structured Query Language
CRUD	CREATE, READ, UPDATE and DELETE
JWT	JSON Web Token
URL	Uniform Resource Locator

ID	Identity Document
CLI	Command Line Interface
CSS	Cascading Style Sheet
CI - CD	Continuous Integration and Continuous Delivery/Continuous Deployment
AWS	Amazon Web Services
FR	Functional Requirement
NFR	Non - Functional Requirement
HTML	Hypertext Markup Language
GDPR	General Data Protection Regulation
AI	Artificial Intelligence
CV	Curriculum Vitae
SDGP	Software Development Group Project
SRS	Software Requirement Specification

Table 1 - Abbreviations Table

Chapter 1: Implementation

1.1 Chapter Overview

This chapter will focus on how this project was built. In this chapter it includes why chose the technologies included in this project. Will mention the implementation of data science components, the development of front and backend. Also how the GIT repository helped this project to make everything go smoothly.

1.2 Overview of the prototype

The SaveNest prototype is a user-friendly finance management application. By allowing customers to browse grocery products, it eliminates the need for manual data entry while creating budgets. This finance management application offers real-time price change notifications for budgeted items and integrates with Keells Supermarket (adaptable to others). Purchases and the budget are immediately synchronized when a supermarket receipt is uploaded. In addition to budgeting, SaveNest provides stock price predictions and an NLP-enabled financial chatbot to help users make educated investment selections and receive tailored financial guidance. Standard features like password management and sign-in/up ensure user security. By offering insights into spending patterns, visual reports help customers optimize their budgets and improve their financial well-being. With the goal of revolutionizing personal finance management, SaveNest offers a user-centric platform that integrates investment research, budgeting, and shopping interaction.

1.3 Technology selections

Technology selections for SaveNest are detailed below along with the language selection and libraries/frameworks selection.

1.3.1.Language Selection

After evaluating the programming languages available, Javascript was selected as the main programming language for the implementation of this project. Javascript is gaining significant traction in both commercial and educational environments due to its versatility and power.

- Extensive presence in Open Source Projects: JavaScript Serves as a foundation for a vast array of open-source projects, making it a language of choice for collaborative development.
- Thriving Community Support: The JavaScript community is renowned for its helpfulness, offering extensive assistance to developers encountering challenges during the implementation process.
- Developer Comfort and Learning Curve: Javascript's syntax is known for its readability, allowing developers to feel comfortable using it and fostering a desire to further enhance their knowledge within this domain.
- Machine Learning with Javascript: A rich ecosystem of libraries and frameworks empowers developers to leverage JavaScript for Machine Learning Tasks.

At the Documentation Phase, the team decided to use Flutter for front-end Development but due to the steeper learning curve the team decided to change the decision and migrate to React Native for front-end Development. This choice relied on the many benefits of JavaScript, which is the main Technology used in React Native. Here are some reasons why the team has decided to use react native as the main technology for the Front End.

- Code Reusability: This is the main advantage of React Native. Unlike native app development where Developers need separate codebases for IOS and Android, React

Native allows developers to write 90% of the Code once and Deploy it on both platforms. This significantly reduces development time and cost, especially for projects targeting both major mobile operating systems.

- Faster Development: Building on the code reusability aspect, React Native offers pre-built UI components that act as building blocks for the app. This eliminates the need to develop everything from scratch, streamlining the development process. Additionally, features like live Reloading allow developers to see code changes reflected in real-time, accelerating the Development Cycle.
- Easy Maintenance: Since large portions of the codebase are reusable across platforms, maintaining a React Native app becomes Simpler. Developers only need to change one, and they'll reflect on both iOS and Android versions of the App.

1.3.2. Libraries and Frameworks

1.3.2.1. NLTK

In developing the SaveNest Project chatbot, NLTK played a crucial role in enabling natural and informative interactions. Here's the breakdown of how the functionalities of NLTK's have been utilized.

1. Preprocessing User Input

- Tokenization -NLTK helps us break down user sentences into individual words or phrases(tokens). This allows a chatbot to understand the basic building blocks of the message.
- Normalization - NLTK could have been utilized for tasks such as streaming, which involves reducing words to their base form (e.g.: "running" to "run"), or lemmatization, which involves converting words to their dictionary form. These processes aid in managing variations and typos in user input.

- Stop word Removal - NLTK's pre-built tools likely came in handy to remove common words like "the" or "a" (stop words) that don't hold much meaning, allowing the SaveNest to focus on the core content of the message.

2. Understanding User Input

- Part-of-Speech(POS) Tagging: NLTK might have been used to assign grammatical labels(tags) to each word (e.g., noun, verb, adjective). This helps SaveNest understand the function of each word in the sentence and identify the user's intent.
- Named Entity Recognition (NER): If relevant to SaveNest's functionalities, NLTK could have been used to identify specific entities within the user's message, such as locations, people, or organizations. This allows SaveNest to tailor responses based on the context.

3. Generating Responses

- Classification: NLTK's classification could have been used to categorize user input based on predefined patterns. This helps SaveNest bot to determine the most appropriate response based on the user's intent.
- Pattern Matching: A set of Query patterns and corresponding responses could have been established within NLTK.When the user's input aligns with a pattern, SaveNest might retrieve the pre-defined response, thereby ensuring consistent and informative answers.

1.3.2.2 Pandas

Pandas, renowned for its speed and efficiency, played a crucial role in handling data analysis tasks within the Savenest project. Widely adopted by professionals for Python projects, Pandas facilitated the reading of CSV datasets and provided a convenient means to visualize the data within a data frame. Given the extensive nature of the reviews dataset, leveraging Pandas proved highly effective due to its superior performance compared to alternative CSV reading methods.

1.3.2.3.Pickel

The pickle module in Python serves the purpose of serializing Python objects. Serialization, also known as pickling, involves converting a Python object hierarchy into a byte stream. Conversely,de-serialization,or unpicking, is the reverse process of converting a byte stream back into a Python object hierarchy. Pickel files are employed for persistence, allowing trained data to be stored for future use in the web application. Thanks to the utilization of pickle files, the necessity of a database was obviated.

1.3.2.4.Matplotlib

Matplotlib, a Python plotting library, is employed in this project to visualize price prediction algorithms results through various graphical representations such as bar graphs, ROC curves, and confusion matrices. These visualizations are implemented within the iPython notebook, which serves as the development environment for the project.

1.3.2.5 Scikit-Learn

Scikit-Learn is a powerful and easy-to-use machine-learning library in Python. It provides a wide range of tools for various machine-learning tasks, including classification, regression, clustering, dimensionality reduction, and more. Here's a simplified overview of how sci-kit learn is used:

1. Data Preparation

- Data Collection
- Data Preprocessing(e.g., cleaning, normalization)

2. Model Building:

- Choose an appropriate algorithm(e.g., SVM, Random Forest, K-Means)
- Split data into training and testing sets
- Train the model on the training data

3. Model Evaluation

- Evaluate the model's performance using metrics like accuracy, precision, recall, etc.
- Fine-tune the model parameters its necessary(e.g.: through cross-validation)
- Train the model on the training data.

4. Prediction

- Use the trained model to make predictions on new, unseen data

5. Model Deployments

- Deploy the trained model into production, making it available for real-world use

1.3.2.6. Numpy

Numpy, short for Numerical Python, is a fundamental library for numerical computing in Python. It provides support for powerful and efficient array operations, enabling you to work with large multidimensional arrays and matrices. Numpy is the foundation for many other libraries in the Python scientific ecosystem, making it an essential tool for data scientists, engineers, researchers, and developers alike. This Python library helps to do some calculations on stock price prediction algorithms.

1.3.2.7. Keras

A high-level Python deep learning library called Keras offers an intuitive user interface for creating, honing, and implementing deep learning models. It makes neural network prototyping quick and easy with little code, so both novice and expert researchers can use it. This is a summary of the standard Keras workflow:

1. Model Definition

- Choose a model architecture (e.g.Sequential, Functional ApI)
- Add layers to the model(e.g.Dense, Convolutional, Recurrent)
- Configure Layer Parameters (e.g.Activation Function, number of Units)

2.Compilation

- Compile the Model by specifying loss function, optimizer, and evaluation metrics
- This step prepares the Model for training

3. Training

- Train the model on training data
- Keras automatically performs backpropagation and updates model parameters during training

4. Evaluation

- Evaluated the Trained models' performance on test data
- Computes metrics such as accuracy, loss, etc.

5. Prediction

- Use the trained model to make predictions on new, unseen data

1.3.2.8. Tensorflow

Define the Model: The Keras API of TensorFlow is used to define the neural network model. The model architecture is created using tf.keras.Sequential(). Layers are added to the model using model.add() method. The model in this code is made up of three dense layers:

ReLU activation function is used by the first layer (tf.keras.layers.Dense), which consists of 128 neurons. To avoid overfitting, a dropout layer (tf.keras.layers.Dropout) is placed after the initial dense layer.

ReLU activation function is also used by the 64 neurons in the second layer. The second dense layer is followed by another dropout layer.

The output layer uses the softmax activation function and contains neurons equal to the number of classes.

Compile the Model: The model is compiled using model.compile() after it has been defined. Since this is a multi-class classification problem, categorical cross entropy is chosen as the loss function in this instance. The model's accuracy is chosen as the metric for evaluation, and the Stochastic Gradient Descent (SGD) optimizer is utilized with certain parameters.

Training the Model: The model.fit() method is used to train the model. This method receives training data (trainX and trainY) as well as additional parameters like batch size, verbosity, and the number of epochs. In order to minimize the loss function, the model's weights are updated during the training phase.

Save the Model: The model is saved using model.save() after training. Both the trained weights and the model architecture are saved by this function. The model is saved as 'chatbot_model.h5', and a.h5 file named, by this code.

1.4 Implementation of the data science component

The following project consists of two data science components compared to each other. Both these components were implemented using Natural language processing and supervised Learning Machine Learning.

1.4.1.Chatbot Implementation

Within the SaveNest Finance Management Mobile application, a sophisticated chatbot has been integrated to streamline user interactions and bloster customer support capabilities. Employing cutting-edge natural language processing (NLP) techniques and machine learning algorithms, the chatbot ensures a seamless user experience. The implementation of the chatbot is explained below.

1.4.1.1.Data Preparation and Preprocessing

Training Data: The chatbot’s training data is structured as JSON format (intents.json). It contains intents, each consisting of patterns (user queries) and corresponding responses. For Example

```
{
    "intents": [
      {
        "tag": "greeting",
        "patterns": [
                        "Hello",
                        "Hi",
                        "Hey",
                        "Good day",
                        "Greetings",
                        "What's up?",
                        "Yo",
                        "Hi there",
                        "Good morning",
                        "Good afternoon",
                        "What's up",
                        "Howdy",
                        "Hiya",
                        "Hey there",
                        "Hello there",
                        "Hi, can I get some help?",
                        "Hello, may I ask a question?",
                        "Hi, I need assistance",
                        "Hey, could you help me?",
                        "Hello, I have a query",
                        "Hi, is anyone there?",
                        "Hey, is this support?"
                    ],
        "responses": [
                        "Hello!",
                        "Hi there!",
                        "Hey! How can I assist you?",
                        "Hey",
                        "Hey, what's up?",
                        "Hi! How can I be of service?",
                        "Greetings! How may I assist you today?",
                        "Hello there!","Hi! What can I do for you?",
                        "Hey! Need any help?",
                        "Welcome! How can I help you today?",
                        "Hello, how may I be of assistance?",
                        "Hey there! What can I do to assist you?",
                        "Greetings! How may I help you?",
                        "Hi! How can I support you?",
                        "Hey! What brings you here?",
                        "Hello! How may I assist you?",
                        "Hi there! How can I be of service?",
                        "Hey! How can I assist you today?",
                        "Hello! What can I help you with?"
                        ]
      },
      {
```

Figure 1 - Data Preparation and Preprocessing

Text Preprocessing: In ‘chatbottrainingNLP.py’, the training data is processed:

- Tokenization

  In tokenization, patterns (user queries) are split into individual words. This is done using NLTK’s ‘word_tokenize()’ function.

  Example:

```
def clean_up_sentence(sentence):
    sentence_words = nltk.word_tokenize(sentence)
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
```

Figure 2 - Tokenization

- Lemmatization:

Words are lemmatized, or reduced to their most basic or root form. This is done to make sure that distinct usages of the same word are handled similarly as tokens. For example “running” and “ran” would both be lemmatized to “run”.The WordNet lemmatizer is used for lemmatization in the chatbot.

Here's the sample code fragment where lemmatization is applied:

This is how lemmatization is applied in chatbot.py

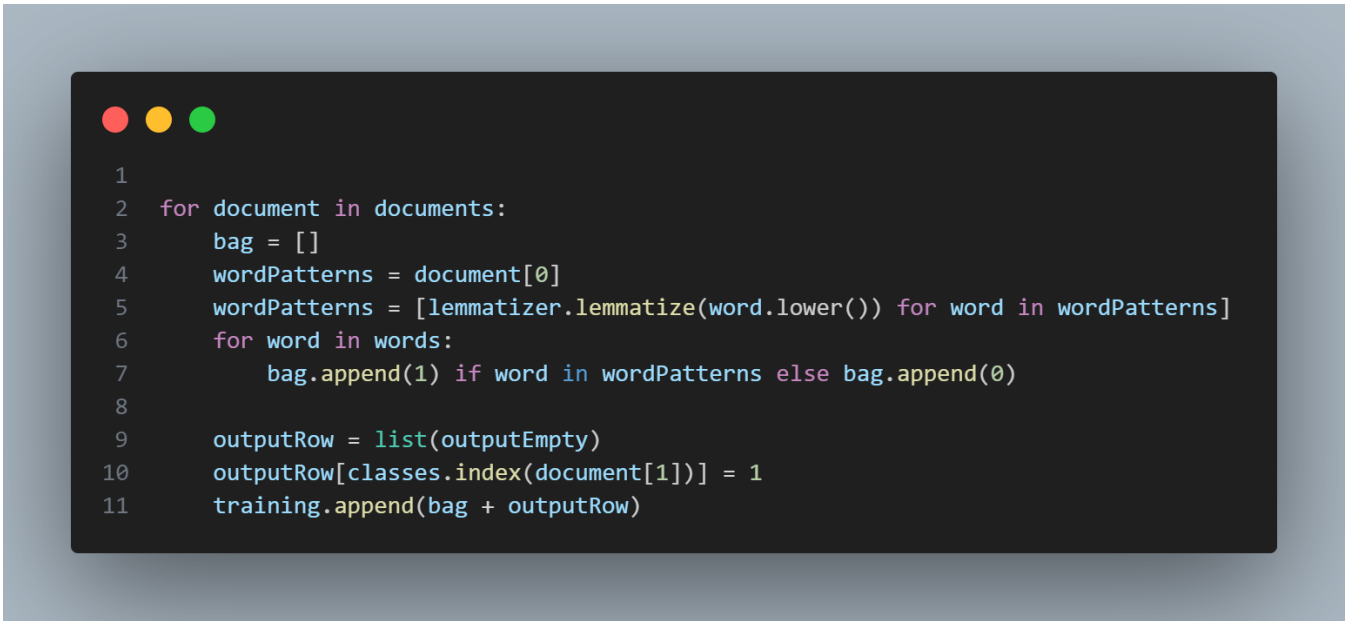
```python
def clean_up_sentence(sentence):
    sentence_words = nltk.word_tokenize(sentence)
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words
```

Figure 3 - lemmatization 1

This is how lemmatization is applied in chatbottrainingNLP.py

```python
#print (documents)

words = [lemmatizer.lemmatize(word) for word in words if word not in ignoreLetters]
words = sorted(set(words))

#print(words)

```

Figure 4 - lemmatization 2



```
for document in documents:
    bag = []
    wordPatterns = document[0]
    wordPatterns = [lemmatizer.lemmatize(word.lower()) for word in wordPatterns]
    for word in words:
        bag.append(1) if word in wordPatterns else bag.append(0)

    outputRow = list(outputEmpty)
    outputRow[classes.index(document[1])] = 1
    training.append(bag + outputRow)
```

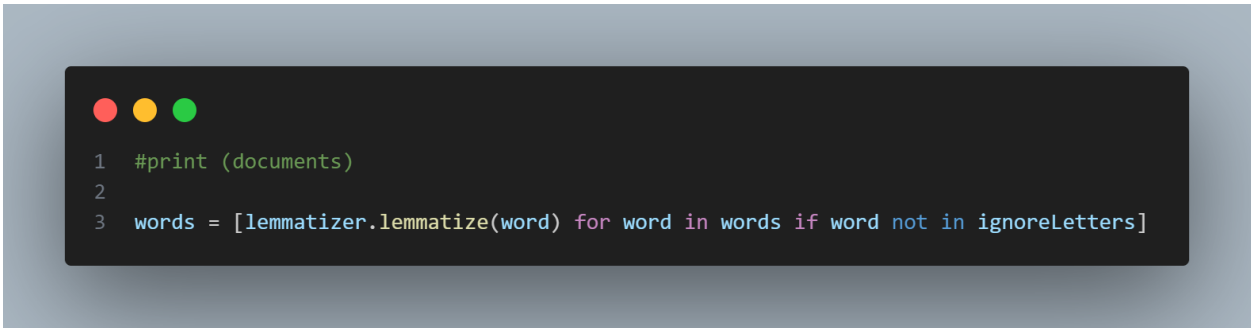
Figure 5 - lemmatization 3

- Building Vocabulary

The words taken out of the training data patterns are lemmatized, and then the distinct lemmatized words are gathered to expand the vocabulary. Each training document's bag-of-words representation is then made using this vocabulary.

Here is where the building of vocabulary is used:

1. Building Vocabulary from Training Data Patterns



```
#print (documents)

words = [lemmatizer.lemmatize(word) for word in words if word not in ignoreLetters]
```

Figure 6 - lemmatization 4

This line creates a list of lemmatized words (words) by lemmatizing every word that was taken from the training data patterns.

2. Creating Bag-of-Words Representation

```

for document in documents:
    bag = []
    wordPatterns = document[0]
    wordPatterns = [lemmatizer.lemmatize(word.lower()) for word in wordPatterns]
    for word in words:
        bag.append(1) if word in wordPatterns else bag.append(0)


```

Figure 7 - bag of words

Here, the script goes through every document in the training set (documents) one by one. Depending on whether words from the vocabulary are present or absent, it creates a bag-of-words representation (bag) for each document. It is determined whether each word appears in the vocabulary using the lemmatized words from the document (wordPatterns).

3. Shuffling and Conversion to Numpy array

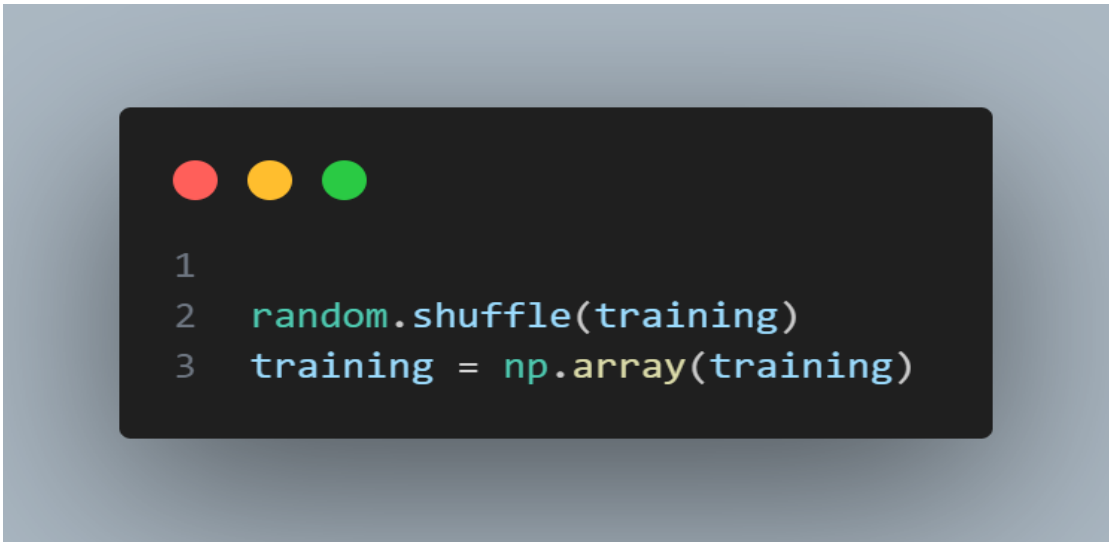


Figure 8 - Shuffling and Conversion to Numpy array

Lastly, the training data is shuffled to guarantee randomness and then transformed into a NumPy array (training) to make it compatible with TensorFlow, following the creation of the bag-of-words representations for each training document.

All things considered, vocabulary building makes sure that the neural network model is trained on a consistent set of input features, which stand in for the distinct lemmatized words that are taken out of the training data patterns. This aids in the development of a reliable and consistent representation of the textual data used to train the chatbot model.

1.4.1.2. Training the Model

Feature Representation: Each Pattern is converted into a bag-of-words representation. For instance, the bag-of-words representation of the pattern "Hi, how are you?" would be [1, 0, 1, 0, 1. if our vocabulary includes the words "hi", "hello", "how", "are", and "you".

Here's how the above-described process is implemented in the training of the neural network Model

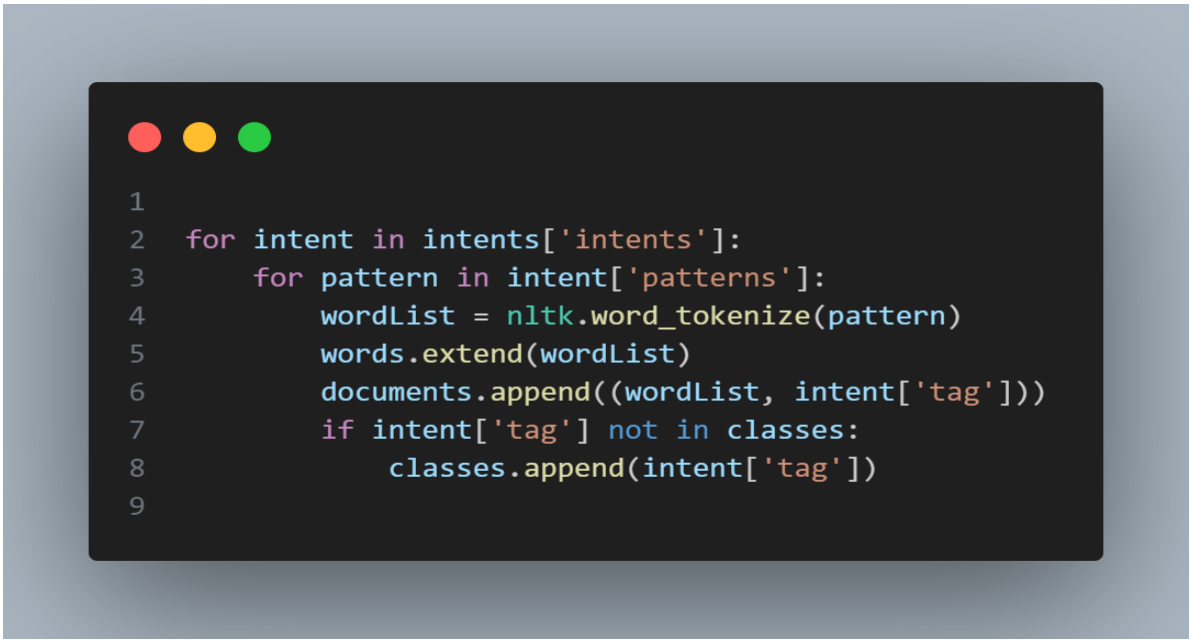
Feature Representation (Bag-of-Words)

```python
def bag_of_words(sentence):
    sentence_words = clean_up_sentence(sentence)
    bag = [0] * len(words)
    for w in sentence_words:
        for i, word in enumerate(words):
            if word == w:
                bag[i] = 1
    return np.array(bag)

# Example usage:
sentence = "Hi, how are you?"
bag_of_words_representation = bag_of_words(sentence)
print(bag_of_words_representation)

#output
[1, 0, 1, 1, 1]  # Assuming the vocabulary contains ["hi", "hello", "how", "are", "you"]
```

Figure 9 - Feature Representation

2. Label Encoding

Label encoding for the chatbot feature in this project is carried out inside the loop that prepares the training data.



```
for intent in intents['intents']:
    for pattern in intent['patterns']:
        wordList = nltk.word_tokenize(pattern)
        words.extend(wordList)
        documents.append((wordList, intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

Figure 10 - Label Encoding 1

In this loop:

- Intent['tag'] represents the intent tag that is connected to every pattern.
- classes: classes is a list that contains the unique intent tags.
- classes. index(document[1]) is used to find the index of the intent tag in the classes list.
- outputRow[classes.index(document[1])] = 1 sets the value at the index corresponding to the intent tag to 1, effectively performing one-hot encoding.

Here is the code explanation in detail.

1. Collecting intent tags

- Every intent in intents['intents'] is iterated through in the loop.
- It gathers the corresponding intent tag (intent['tag']) for every intent.
- It appends the intent tag to the classes list if it isn't there already.

2. Label Encoding Explanation

- classes. index(document[1]) is used to find the intent tag's index in the classes list for each pattern. The intent tag is retrieved (document[1]).
- This index is used to set the corresponding element in the ‘outputRow’ list to 1, indicating the presence of that class.

Here is the relevant part of the code

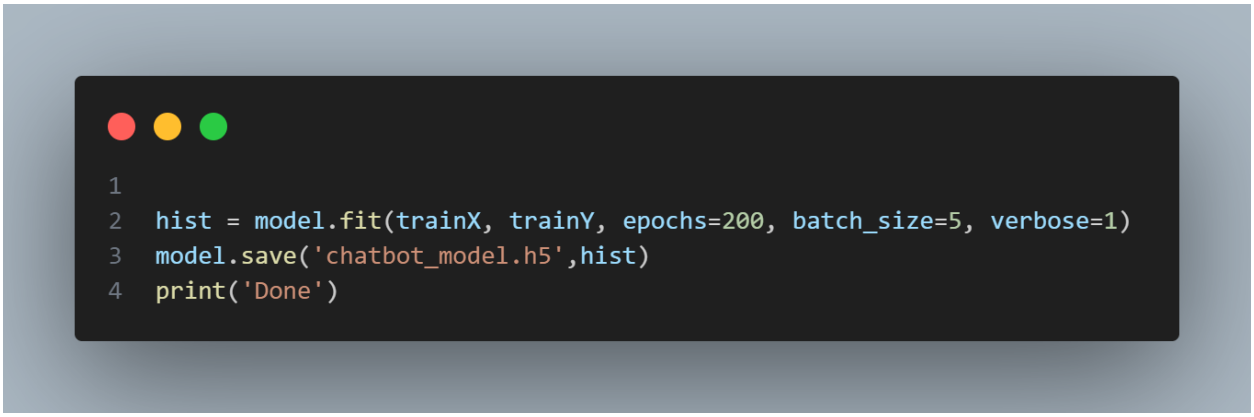
```
# Set the index corresponding to the intent label to 1
    outputRow = list(outputEmpty)
# Append bag-of-words representation and encoded label to training data
    outputRow[classes.index(document[1])] = 1
    training.append(bag + outputRow)
```

Figure 11 - Label Encoding 2

To summarize, this section of the code maps each intent tag to a distinct numerical label that is subsequently used to encode labels in the training set. This guarantees that during training, the neural network will be able to comprehend and learn from the intent classes

3. Model saving

TensorFlow’s model saving functionality is used to store the trained model to a file (chatbot_model.h5) after training is finished. As a result, in subsequent sessions, the model can be loaded and reused without requiring retraining. Below code snippet below shows how the team has done the model-saving part.



```
1
2  hist = model.fit(trainX, trainY, epochs=200, batch_size=5, verbose=1)
3  model.save('chatbot_model.h5',hist)
4  print('Done')
```

Figure 12 - Model Saving

The chatbot can comprehend user queries and provide contextually relevant responses by utilizing supervised machine learning, neural networks modeling, and natural language processing approaches. This all-encompassing strategy makes it possible for users to connect with the SaveNest finance management application more effectively and improves their entire experience.

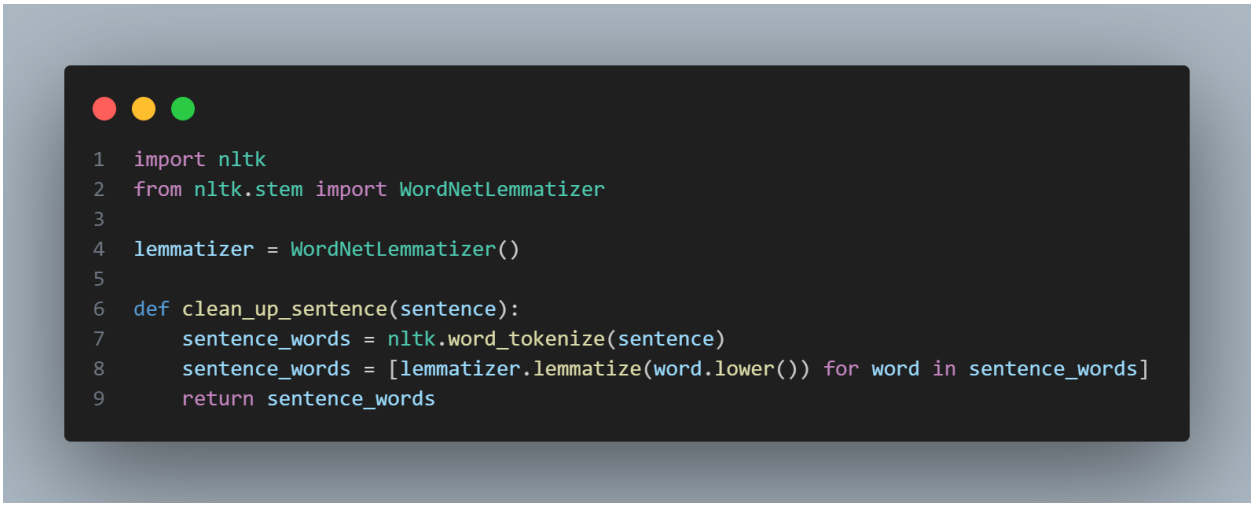
4. Chatbot Implementation

- Model Loading: The saved model is loaded into memory in chatbot.py along with any appropriate libraries, such as NLTK and NumPy.Below code snippets below show how the team has done the Model Loading part. The 1 and 6 lines are responsible for model loading in the below code screenshot below.

```
from keras.models import load_model
from nltk.stem import WordNetLemmatizer
model = load_model('chatbot_model.h5')
```

Figure 13 - Chatbot Implementation

- User Input Processing: A query that a user submits is preprocessed using techniques that are similar to those used in training: Tokenization, lemmatization, and conversion into a bag-of-words representation. Below code snippets below show how the team has done the User Input Processing.



```
import nltk
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

def clean_up_sentence(sentence):
    sentence_words = nltk.word_tokenize(sentence)
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words
```

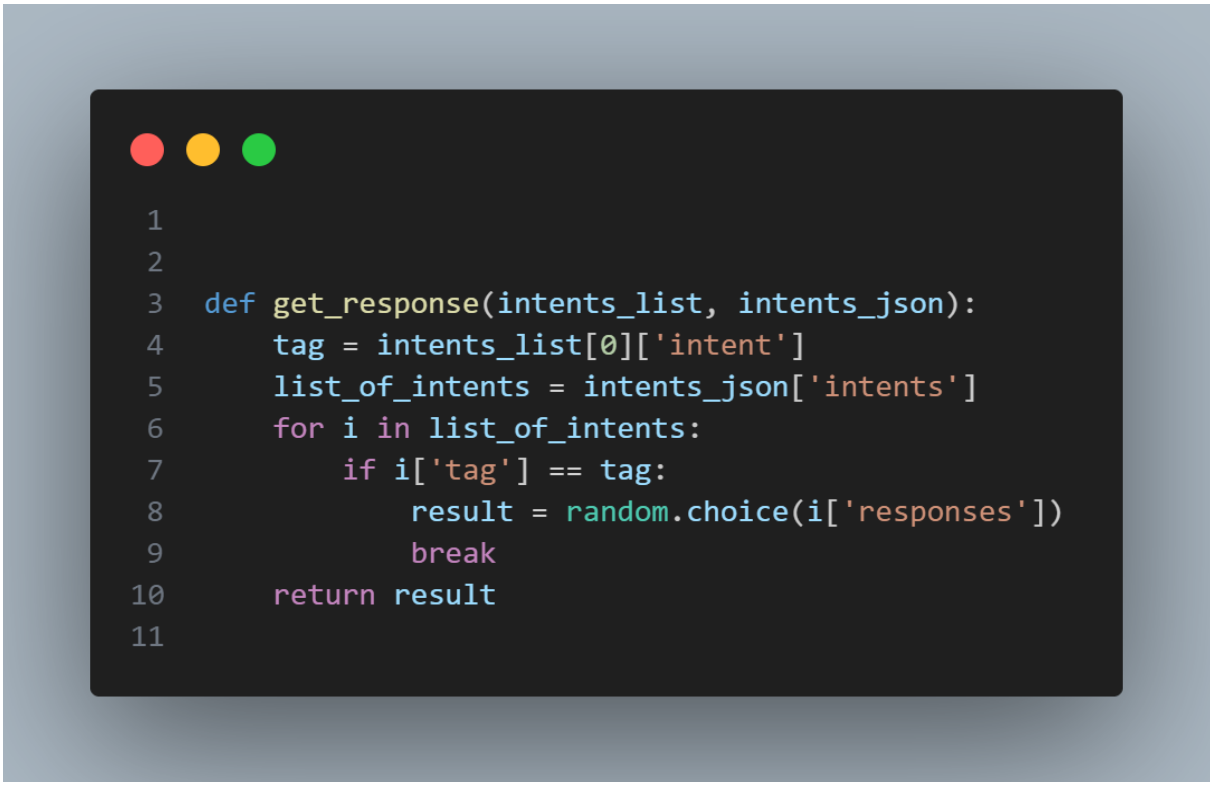
Figure 14 - User Input Processing

- Intent Prediction: To determine the most probable intent, the preprocessed input is passed along with the loaded model. The neural network, that generates a probability distribution over all intents, is fed the bag-of-words representation in order to accomplish this. Below code snippets below show how the team has done the Intent Prediction.

```python
def bag_of_words(sentence):
    sentence_words = clean_up_sentence(sentence)
    bag = [0] * len(words)
    for w in sentence_words:
        for i, word in enumerate(words):
            if word == w:
                bag[i] = 1
    return np.array(bag)

def predict_class(sentence):
    bow = bag_of_words(sentence)
    result = model.predict(np.array([bow]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i, r] for i, r in enumerate(result) if r > ERROR_THRESHOLD]
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({'intent': classes[r[0]], 'probability': str(r[1])})
    return return_list

```

Figure 15 - Intent Prediction

- Response Generation: The generated answer is chosen at random from the response list associated with the related intent and given back to the user based on the expected intent.Below code snippets below show how the team has done the Intent Prediction.



```
1
2
3  def get_response(intents_list, intents_json):
4      tag = intents_list[0]['intent']
5      list_of_intents = intents_json['intents']
6      for i in list_of_intents:
7          if i['tag'] == tag:
8              result = random.choice(i['responses'])
9              break
10     return result
11
```

Figure 16 - Response Generation

The essential features of the chatbot implementation make it easier to process user queries, anticipate intents, and generate relevant responses using the trained model. The SaveNest finance management application's user experience is greatly improved by these features.

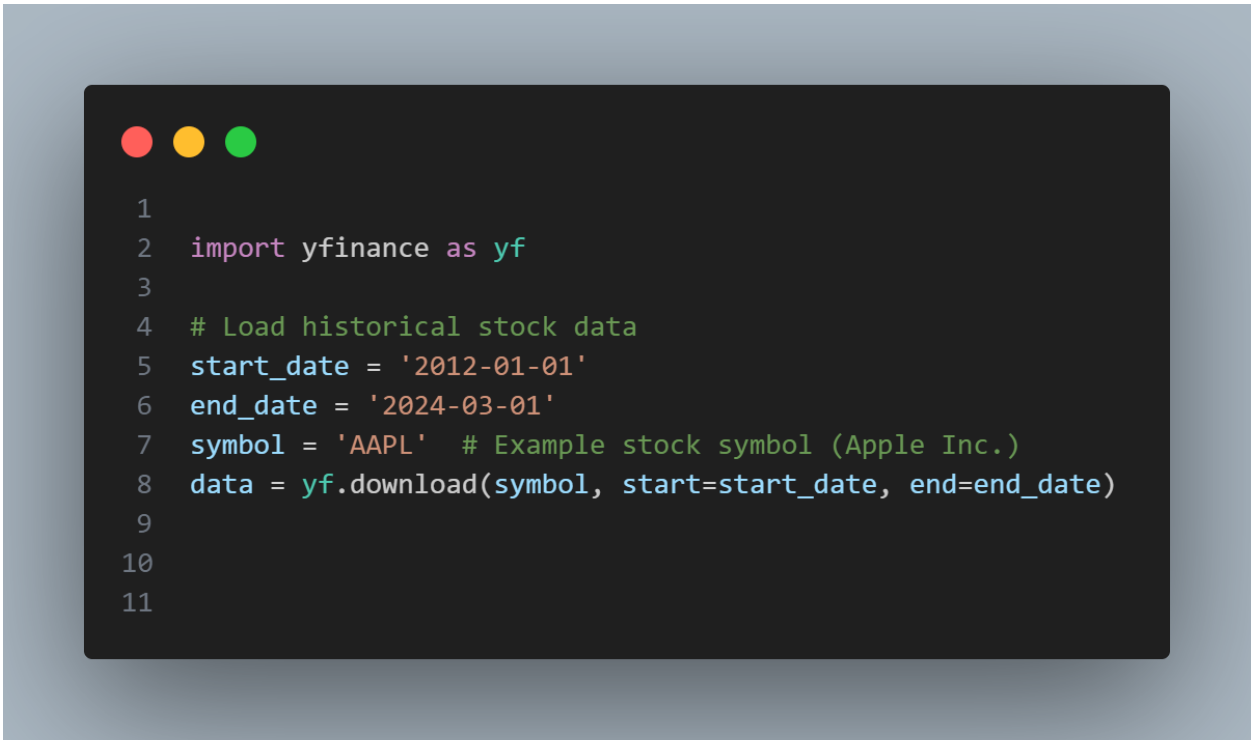
1.4.2. Stock Price Prediction Algorithm Implementation

An advanced stock price prediction algorithm has been carefully incorporated into the SaveNest Finance Management Mobile application to completely transform the way users experience investing. This algorithm uses advanced models of machine learning and predictive analytics to give users the ability to predict stock price movements with unmatched accuracy, enabling them to make well-informed investment decisions. Below is a detailed explanation of how the Stock Price Prediction Algorithm is implemented.

In the realm of finance and data science, predicting stock prices has always been a captivating challenge. Leveraging Python along with frameworks like Flask and libraries such as NumPy, Pandas, and Keras, the team built a simple yet effective stock price prediction system. Delve into the code to understand how it works.

1.4.2.1.Gathering Data

We used the Yahoo Finance API (yfinance) to fetch historical stock price data. This includes information like opening price, closing price, highest price, lowest price, and trading volume. For our demonstration, let's focus on predicting the closing price.



```
1
2  import yfinance as yf
3
4  # Load historical stock data
5  start_date = '2012-01-01'
6  end_date = '2024-03-01'
7  symbol = 'AAPL'  # Example stock symbol (Apple Inc.)
8  data = yf.download(symbol, start=start_date, end=end_date)
9
10
11
```

Figure 17 - Gathering Data

1.4.2.2. Preprocessing Data:

Before feeding the data into the model, the team decided to preprocess it by scaling it to a range between 0 and 1 using Min-Max scaling. This step is crucial for enhancing the convergence and performance of the neural network.Below code snippet shows how the team did the data Preprocessing data part.

```
from sklearn.preprocessing import MinMaxScaler

# Preprocess data
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data['Close'].values.reshape(-1, 1))
```

Figure 18 - Preprocessing Data

1.4.2.3. Building the LSTM Model

The LSTM neural network architecture is at the center of the stock price prediction strategy. Using the user-friendly neural network API Keras, a sequential model is constructed. Several LSTM layers are integrated with dropout layers in this model to counteract overfitting.

```
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout

model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50))
model.add(Dropout(0.2))
model.add(Dense(units=1))

model.compile(optimizer='adam', loss='mean_squared_error')
```

Figure 19 - Building the LSTM Model

1.4.2.4. Training the Model

The LSTM model is trained using historical data on stock prices. In order to reduce the mean squared error between the predicted and actual stock prices, the model's parameters are iteratively adjusted throughout the process.

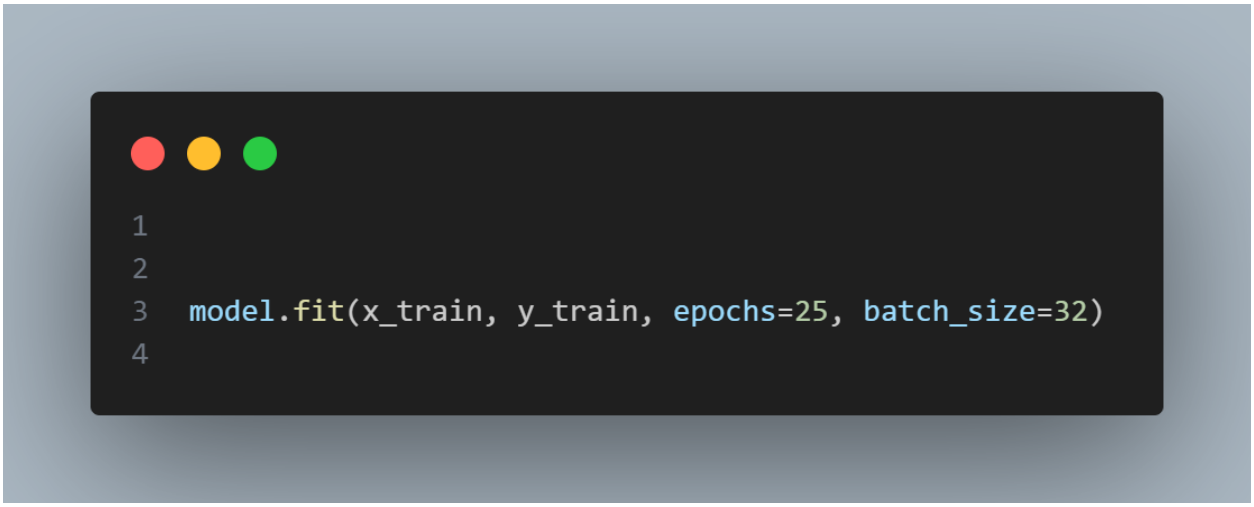


Figure 20 - Training the Model

1.4.2.5. Making Predictions

After that, the team used a trained model to make predictions on unseen data.

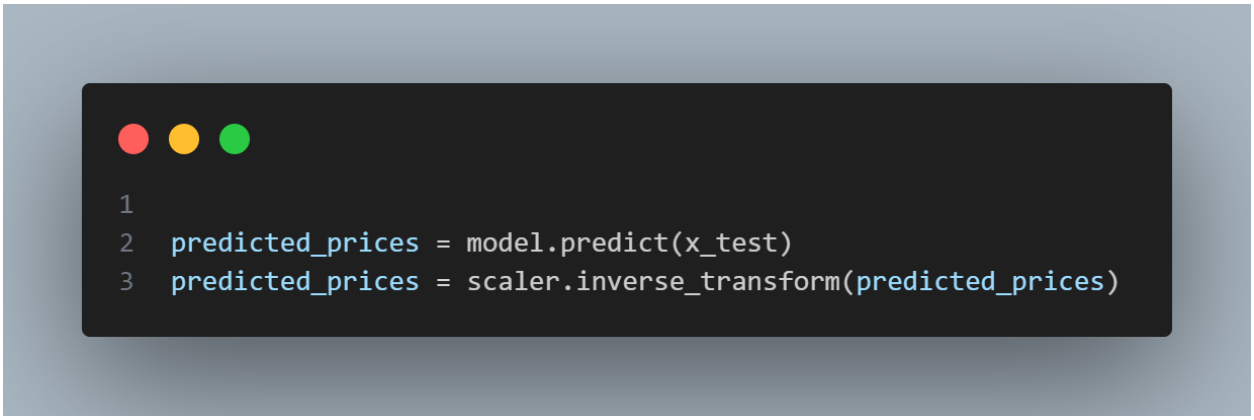
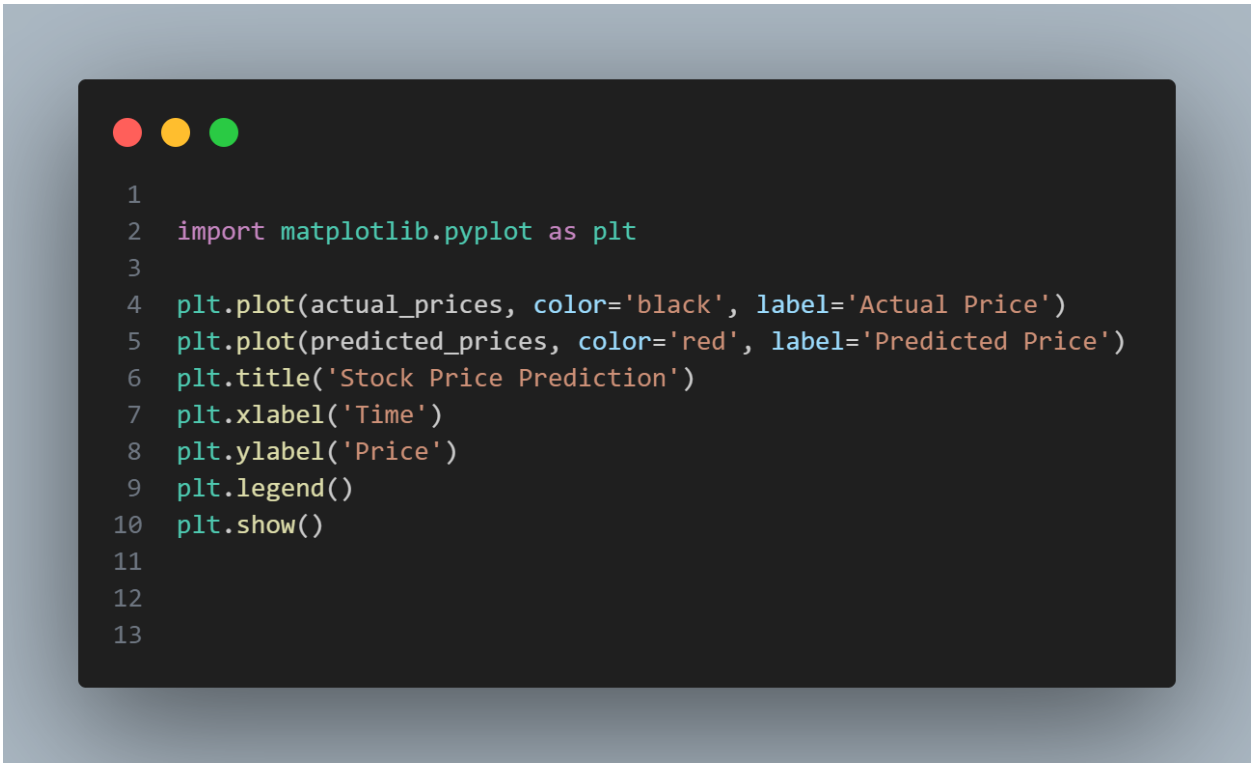


Figure 21 - Making Predictions

1.4.2.6. Visualization and Evaluation: Making Predictions

The team decided to visualize both actual and predicted stock prices using Matplotlib.



```
1
2  import matplotlib.pyplot as plt
3
4  plt.plot(actual_prices, color='black', label='Actual Price')
5  plt.plot(predicted_prices, color='red', label='Predicted Price')
6  plt.title('Stock Price Prediction')
7  plt.xlabel('Time')
8  plt.ylabel('Price')
9  plt.legend()
10 plt.show()
11
12
13
```

Figure 22 - Visualization and Evaluation

Here is how the visualization is done.

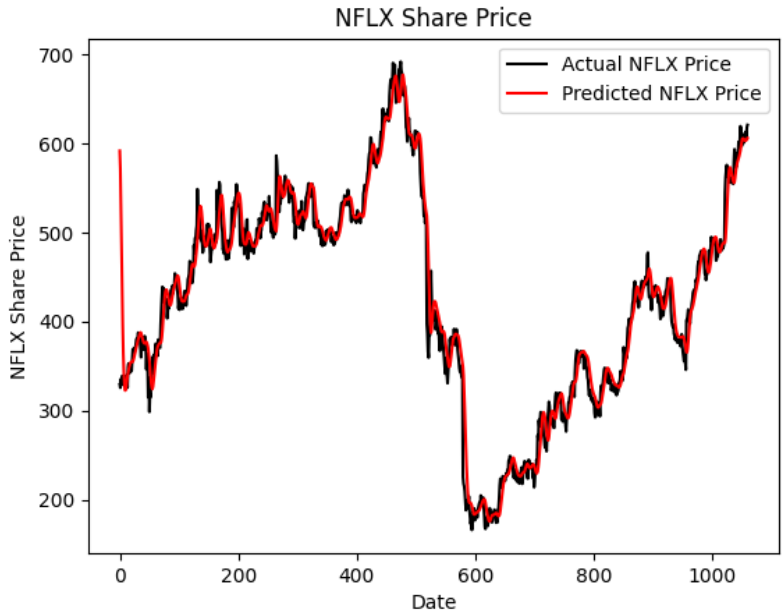


Figure 23 - NFLX Share Price

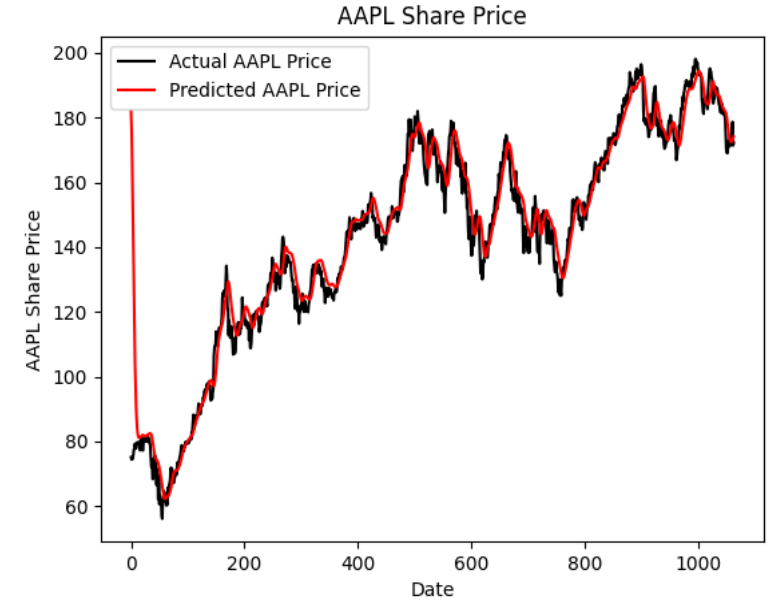
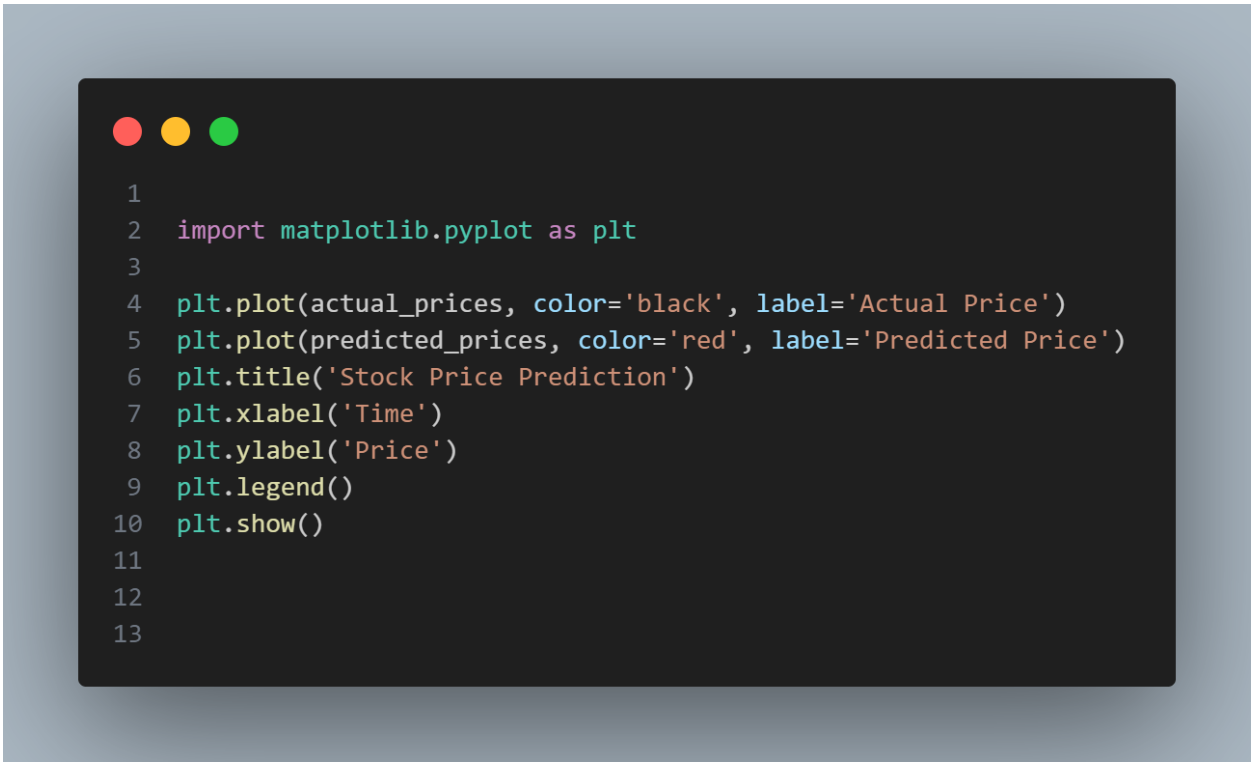


Figure 24 - AAPL Share Price

7. Deployment with Flask

The team decided to deploy a stock price model as web service using flask. Here is how that task is done.



```
import matplotlib.pyplot as plt

plt.plot(actual_prices, color='black', label='Actual Price')
plt.plot(predicted_prices, color='red', label='Predicted Price')
plt.title('Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()
```

Figure 25 - Deployment with Flask

1.5 Implementation of the backend component

1.5.1 Implementation of database

SaveNest's developing process employed Sequelize ORM to streamline database interactions and facilitate seamless integration between Node.js back-end and the underlying SQL database management system. Sequelize helps the project to reduce the use of raw SQL queries which in return helps to reduce the time needed for database implementation and working with databases. Sequelize is capable of creating tables using the models in the code. Before starting the creation tables,"sequelize init" command creates the "models" and "config" folders in the back end.

Model Definition

After setting up the configurations project database is started to be designed by defining the Sequelize models. Following is the model folder in the project.

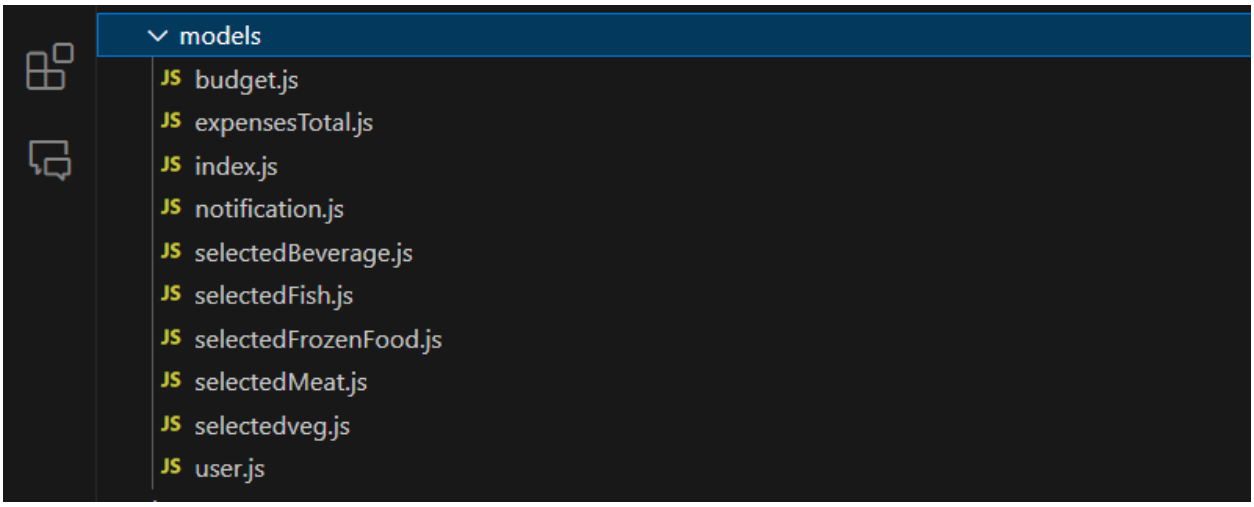


Figure 26 -Model Definition 1

Each model corresponds to a table in the relational database and encapsulates attributes and associations relevant to the respective entity. For instance, It has defined models for users, products, transactions, and other domain-specific entities, specifying attributes such as data types, constraints, and relationships. For instance, project authors defined models for user, budget, notifications, and other domain-specific entities, specifying attributes such as data types, constraints, and relationships. The following is sequelize model for the user.

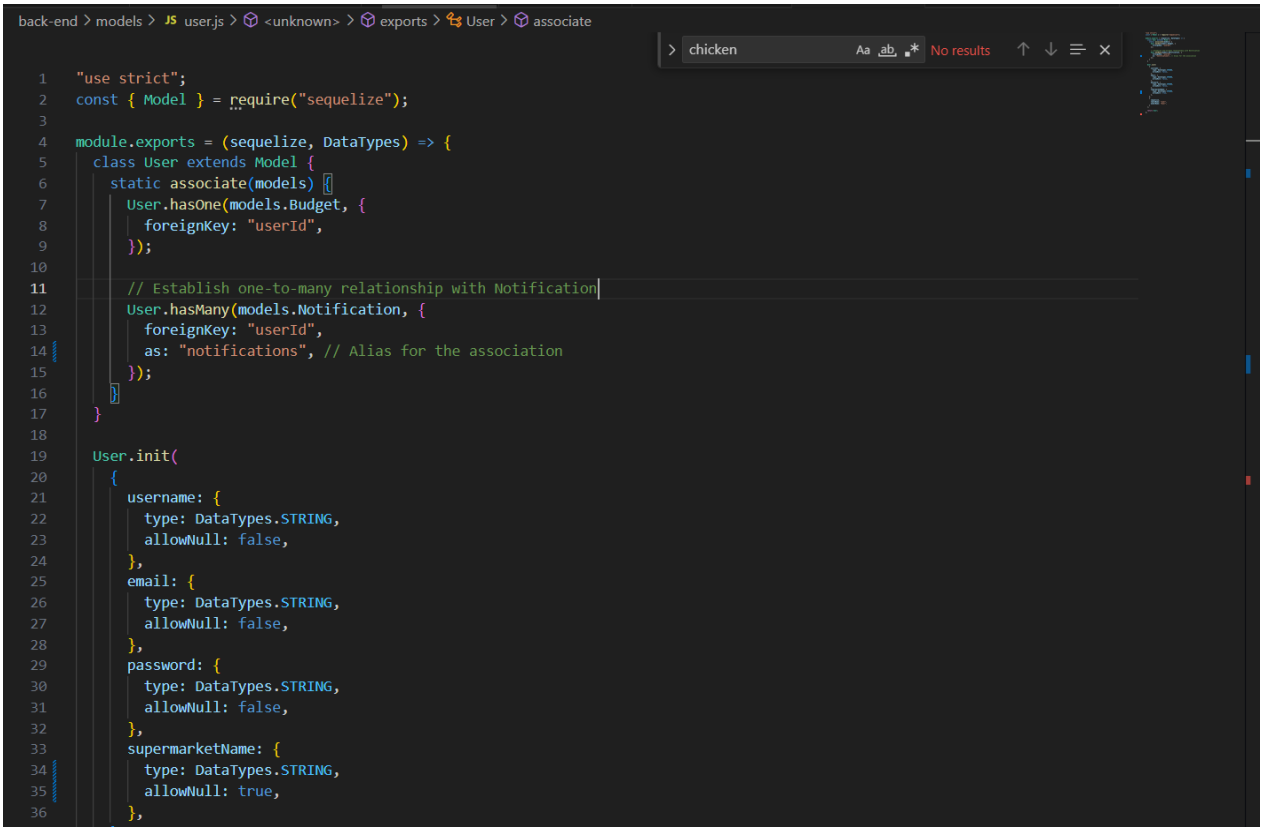


Figure 27 - Model Definition 2

Initialization and Synchronization

Upon initializing the SaveNest application, Sequelize establishes a connection with the configured database instance. Project authors utilized Sequelize's synchronization feature to automatically create database tables based on the defined models. This process ensures that the database schema remains synchronized with the application's data model, eliminating the need for manual intervention.

CRUD Operations

Sequelize simplifies CRUD operations by providing methods for creating, reading, updating, and deleting records in the database. Developers can interact with Sequelize models as JavaScript objects, abstracting away the underlying SQL queries.

For instance, consider the following query to create user in user table,

```
INSERT INTO users (name, age) VALUES ('John', 25);
```

Figure 28 - CRUD Operations 1

Can be written in a more developer-friendly manner with Sequelize like this:

```
User.create({ name: 'John', age: 25 });
```

Figure 29 - CRUD Operations 2

Sequelize ORM played a pivotal role in simplifying database interactions within our project. By leveraging Sequelize's powerful features for model definition, synchronization, and CRUD operations, we were able to efficiently manage our application's data layer and focus on delivering robust and scalable solutions to our end users.

1.5.2 Implementation JWT Authentication

JSON Web Tokens (JWT) are a popular method for implementing authentication and authorization in web applications. JWTs are compact, URL-safe tokens that can securely carry information between parties. They are commonly used for authentication purposes, where a token is issued to a user upon successful login and is then used to authenticate subsequent requests to protected resources.

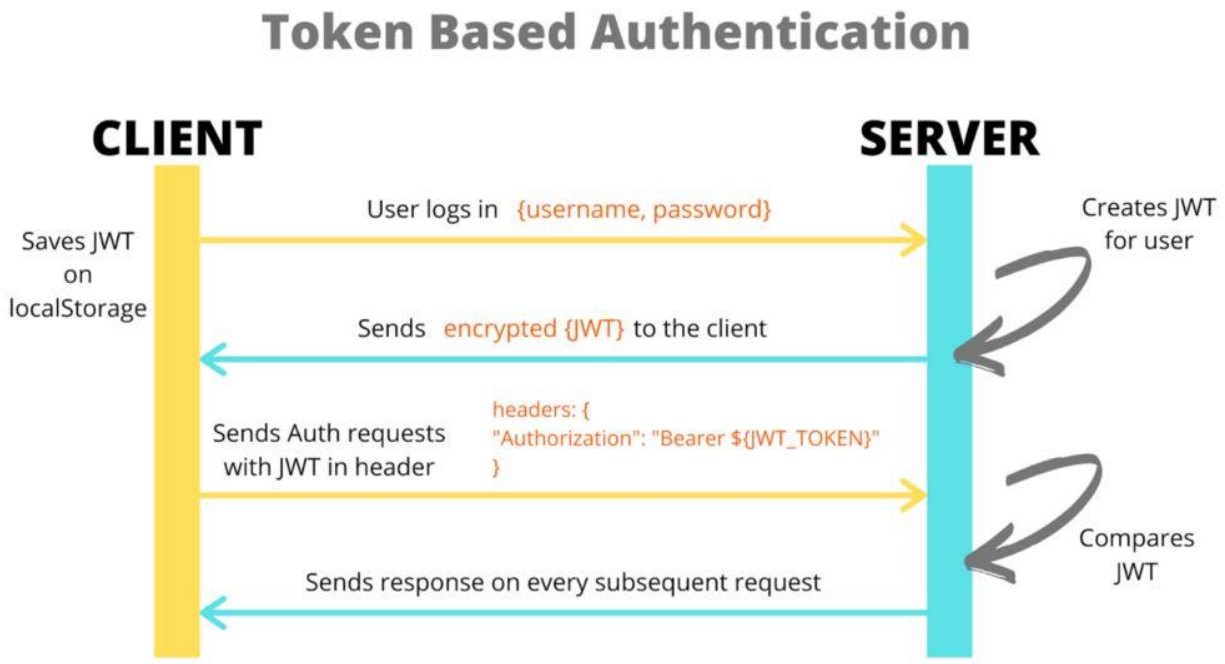


Figure 30 - Token Based Authentication

In the process of implementing JWT authentication for the project, several key steps were undertaken to ensure secure user authentication and access control.

Installation of Dependencies and Configuration of Environment

Installing dependencies like “bcryptjs” and “jsonwebtoken” was the first step. Furthermore, environment variables were set up, most notably the JWT secret key, which improved security and adaptability in various scenarios.

Implementation of User Authentication

The user authentication process was divided into two main functionalities: user sign-up and sign-in. During sign-up, user-provided passwords were securely hashed using “bcryptjs” before being

stored in the database. Conversely, the sign-in functionality verified user credentials against the database and issued JWTs upon successful authentication.

JWT Verification Middleware and Route Protection:

To ensure secure access to protected routes, a JWT verification middleware was developed. This middleware authenticates incoming requests by verifying the JWTs attached to them. By applying this middleware to routes requiring authentication, access to protected resources was restricted to authenticated users only.

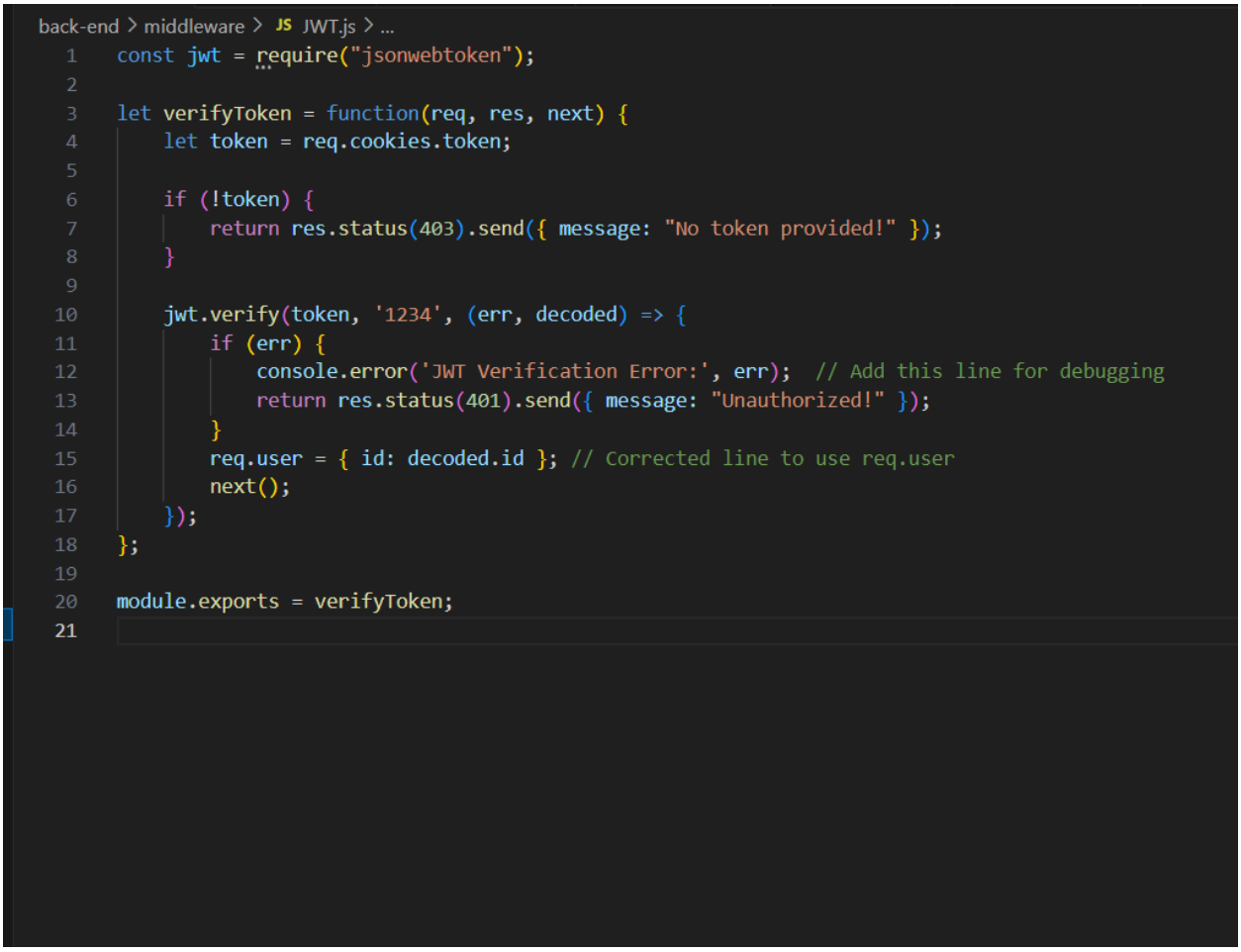


Figure 31 - JWT Verification Middleware and Route Protection

User Profile Endpoint and Access Control:

A user profile endpoint was established to facilitate the retrieval of user information. Leveraging the user's ID extracted from the JWT payload, relevant user details were fetched from the database. Access to the user profile endpoint was tightly controlled, allowing only authenticated users to access their respective profiles.

Through these concerted efforts, the project successfully integrated JWT authentication, bolstering security measures while providing seamless user authentication and access control within the application.

```
//router for the signup endpoint
router.route("/signup").post(controllerUser.signUp);
router.route("/signIn").post(controllerUser.signIn);

//routers for the budget endpoints
router.route("/getBudget").get(verifyToken, controllerBudget.getBudgets);
router.route("/createBudget").post(verifyToken, controllerBudget.createBudget);
router.route("/updateBudget").post(verifyToken, controllerBudget.updateBudget);

//routers for the scraping endpoint
//routes for selected Items(tables crud) end-points
router.route("/addVeg").post(verifyToken, selectedVegController.addVeg);
router.route("/addFish").post(verifyToken, selectedFishController.addFish);
router.route("/addMeat").post(verifyToken, selectedMeatController.addMeat);
```

Figure 32 -User Profile Endpoint and Access Control

1.6 Implementation of the front-end component

React Native was chosen as the preferable framework for constructing the front-end component of the SaveNest project because of its adaptability, robustness, and ability to speed cross-platform mobile application development. Below the specifics of the React Native implementation will be discussed.

1.6.1. Get Started Page

Figure 33 - Get Started Page

Get started page where the user will find first when come to the app.

1.6.2. Register Page

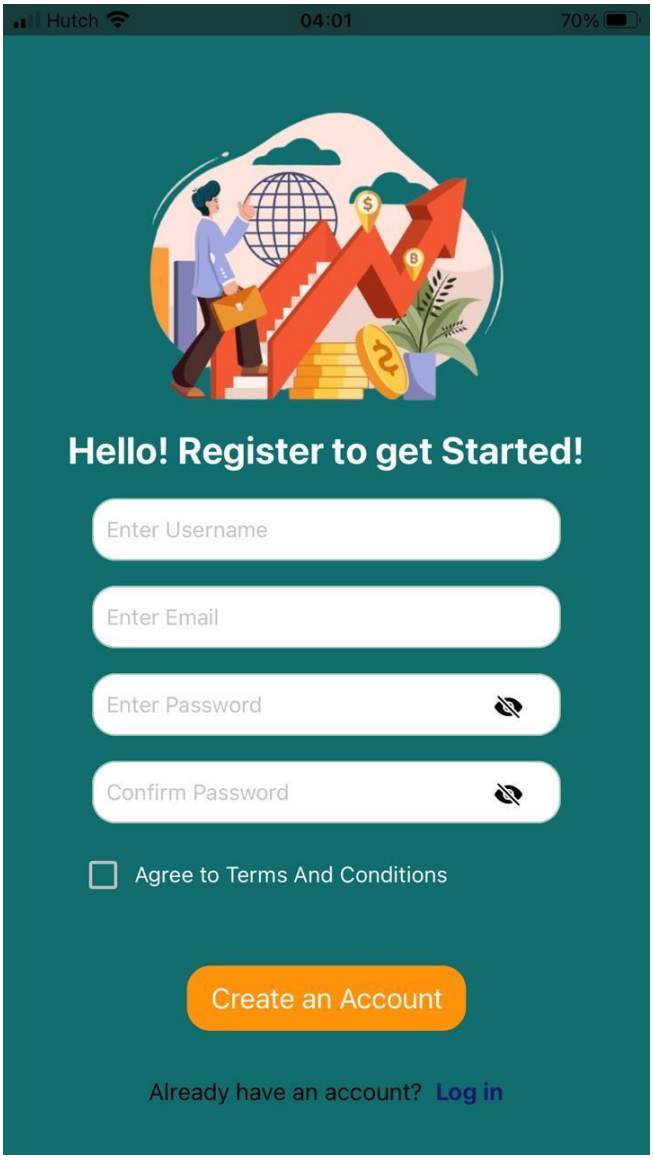


Figure 34 - Register Page

Users can register to the app using this page.

1.6.3. Login Page

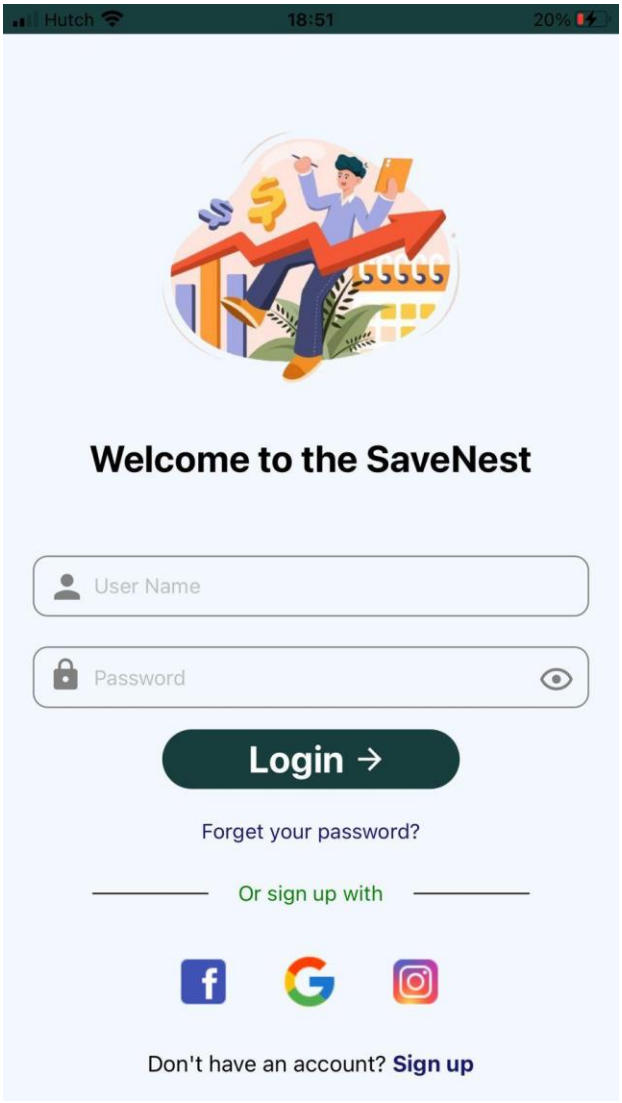


Figure 35 - Login Page

Users can log into the app using this page if they are already registered.

1.6.4. Budgeting Screen

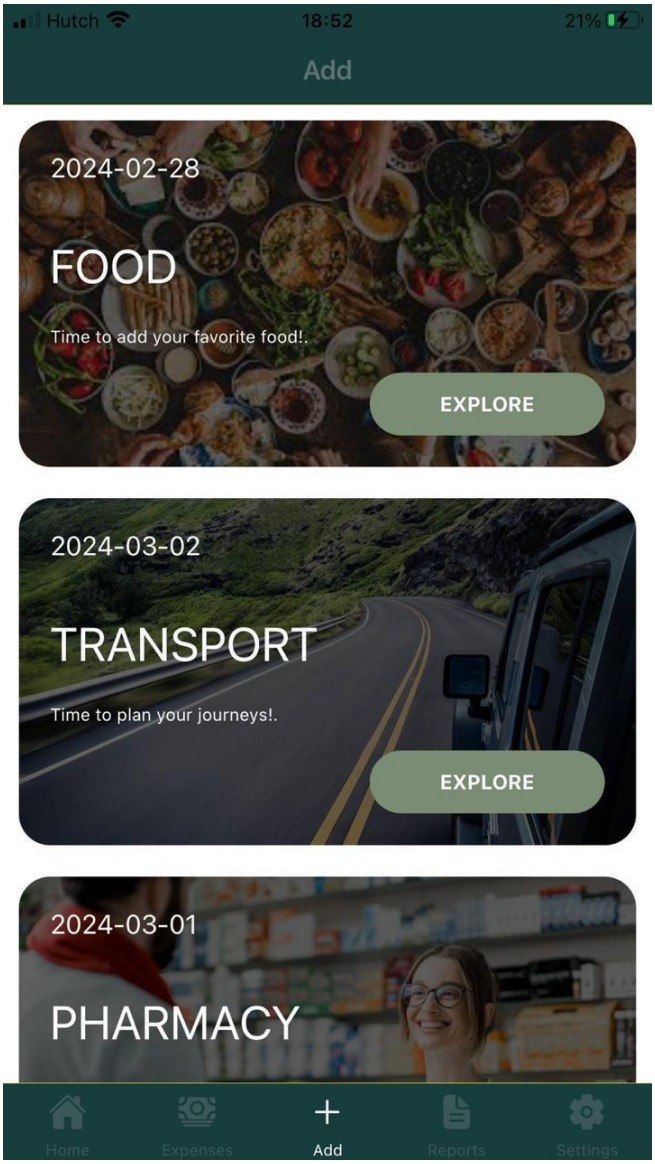


Figure 36 - Budgeting Screen

Users can configure the budget using this page.

1.6.5. Home Page

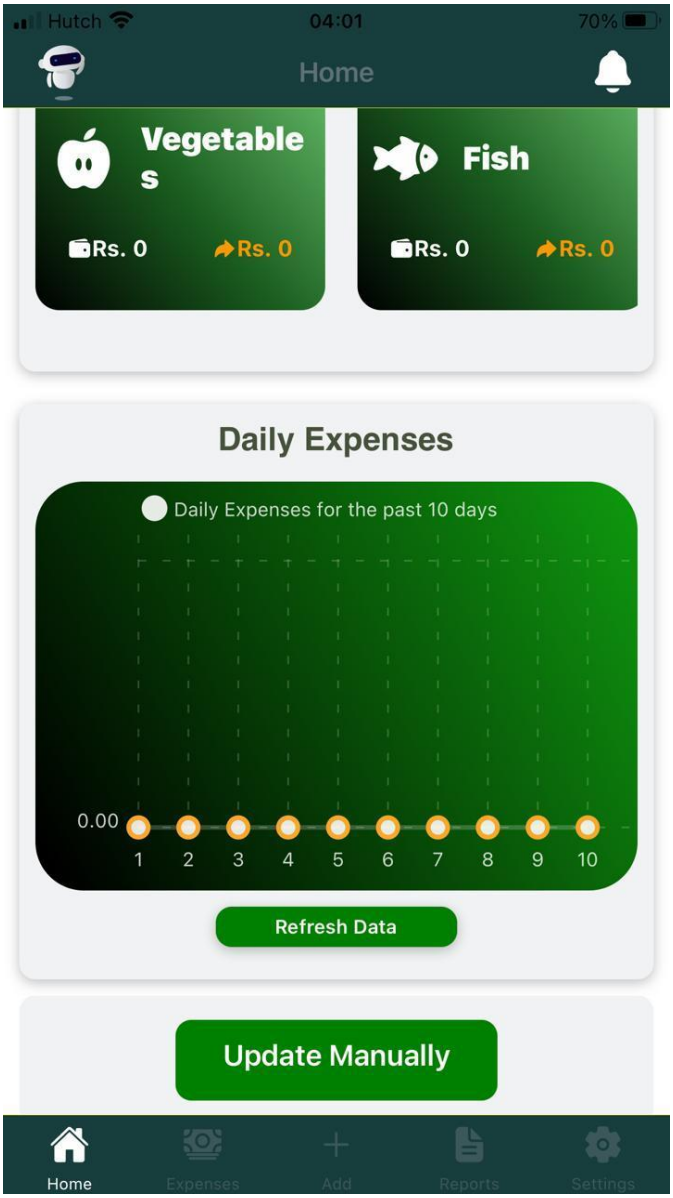


Figure 37 - Home Page

This is the frontend of the front page with options to update the budget and expenses are displayed.

1.6.6. Expenses page

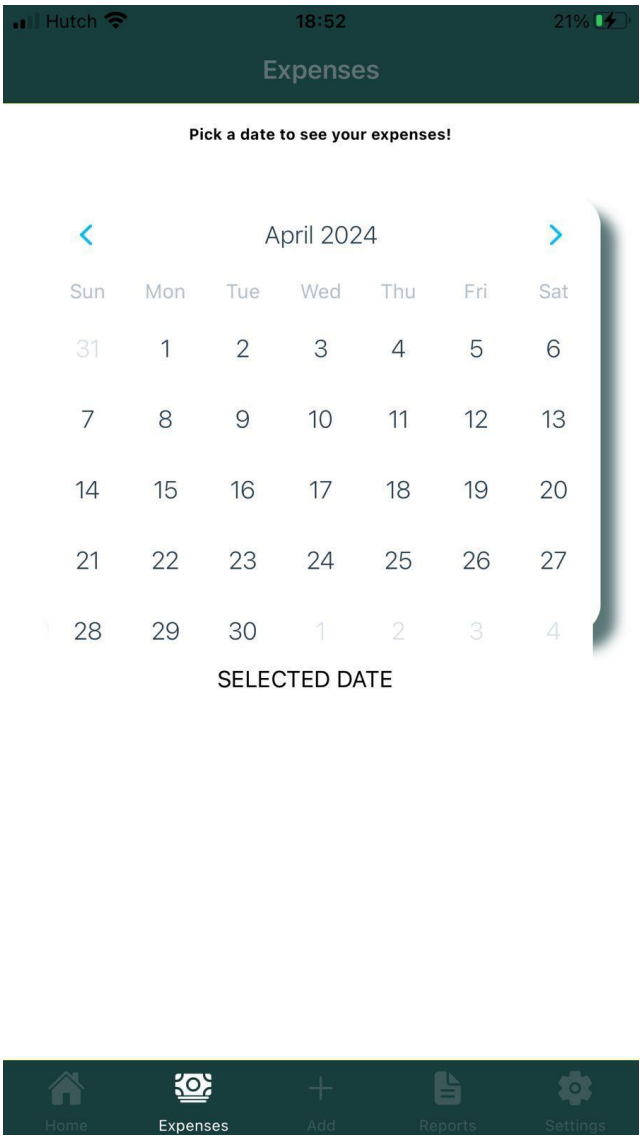


Figure 38 - Expenses page

Can check the past expenses using this frontend expenses page.

1.6.7. Update expenses page

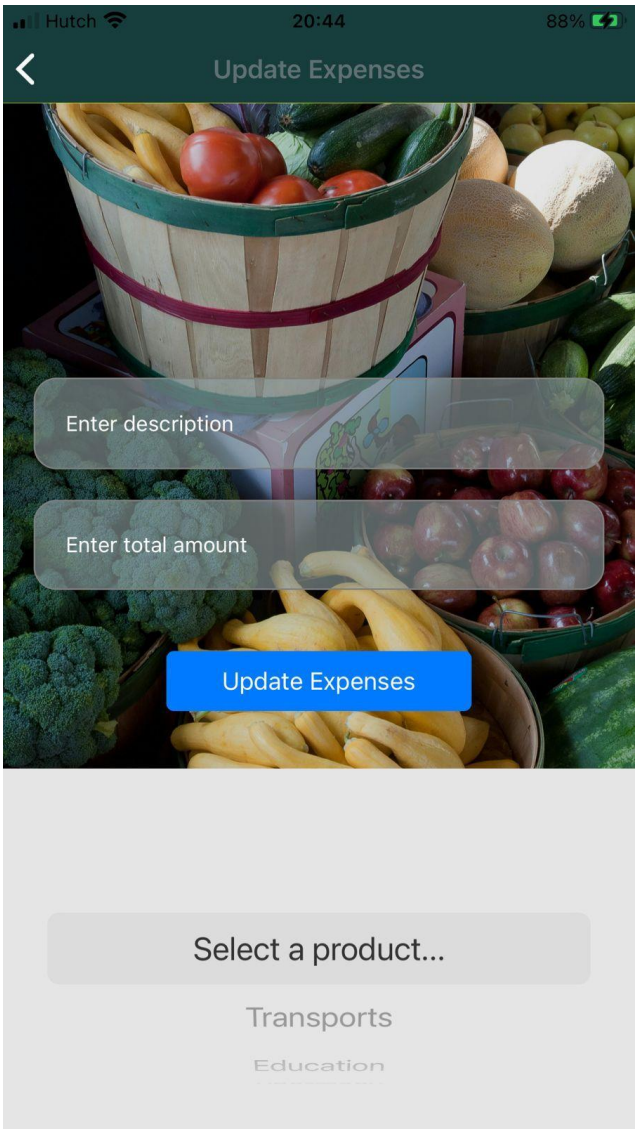


Figure 39 - Update expenses page

Add expenses using manual mode on this page. Should enter a product, description of payment, and total amount.

1.6.8. Chatbot Page

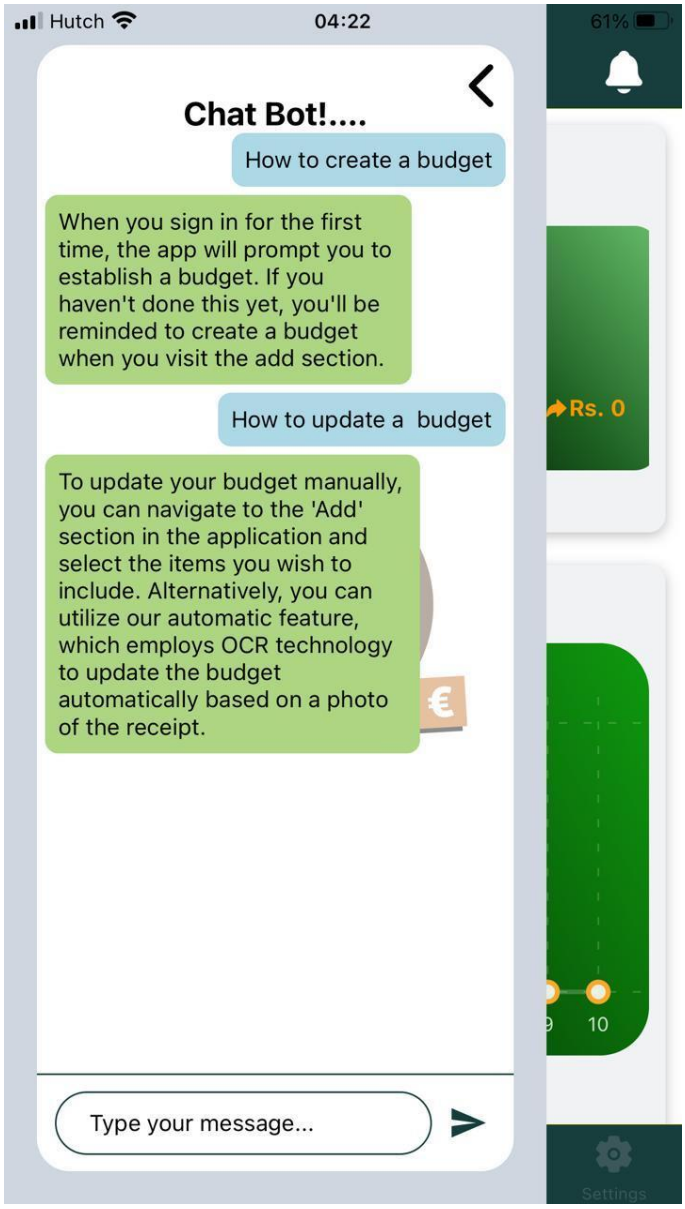


Figure 40 - Chatbot Page

Chatbot page for asking questions about using the app. Users can ask questions about using the app.

1.6.9. Settings Page

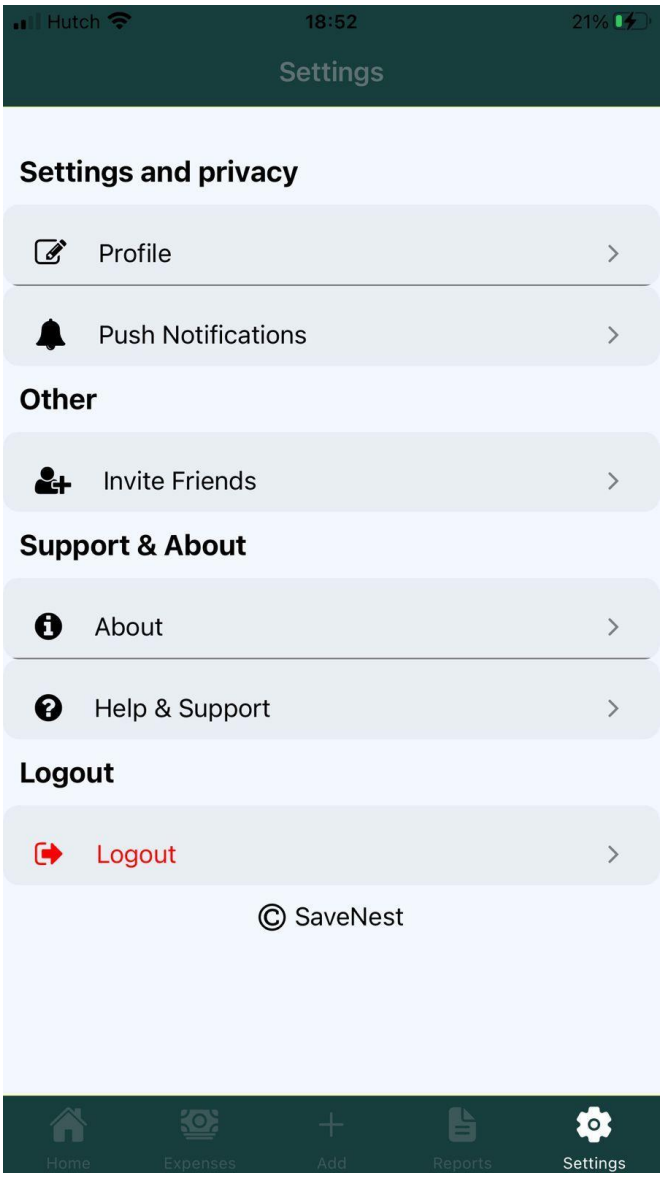


Figure 41 - Settings Page

Can configure the application features and user profile as the user likes using this settings frontend page.

1.6.1 Technology Section and Development Process

Leveraging React Native it was possible to create a single codebase for both ios and android using a codebase for both ios and Android using React Native, which dramatically reduced development overheads, its component-based architecture made modular development easier, increasing code reusability and maintenance. Below is an example.

Our team meticulously designed and implemented individual components of the application, including screens, navigation elements, and UI widgets. React Native's declarative syntax and component lifecycle methods empowered us to create highly responsive and intuitive user interfaces.

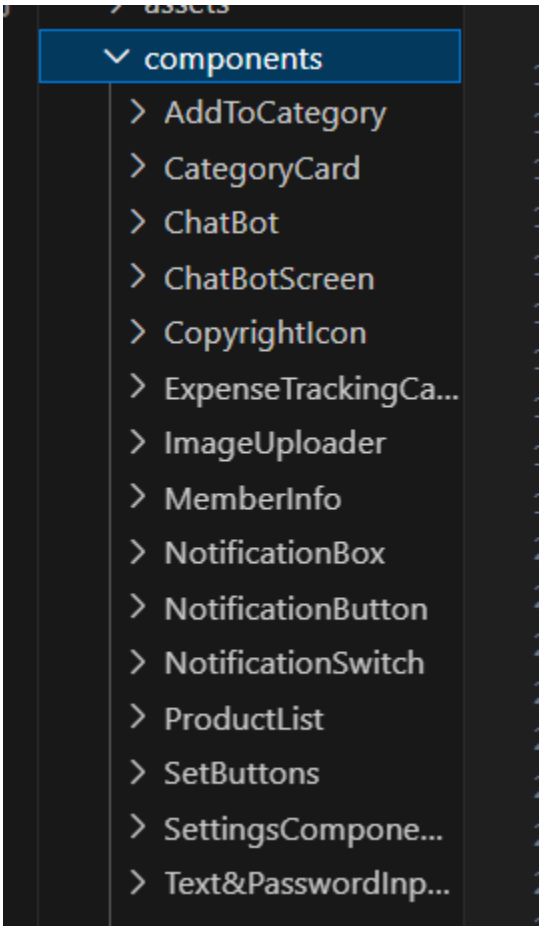


Figure 42 - Technology Section and Development Process

```
const Copyright = ({ text }) => {
  return (
    <View style={styles.container}>
      <Icon name="copyright" size={20} color="black" />
      <Text style={styles.text}>{text}</Text>
    </View>
  );
};
```

```
  <Copyright text="SaveNest" />
```

It initiated the development process by setting up a new React Native project, utilizing the React Native CLI tools. This laid the groundwork for subsequent development endeavors. Styling for the application components was accomplished using CSS- like stylesheets or inline styles, ensuring consistency and visual appeal across the application. React Native's styling capabilities facilitated the creation of pixel-perfect designs adaptable to various screen sizes and resolutions.

Rigorous testing processes were used throughout the development process to ensure the front-end components' functionality, performance, and user experience. This iterative testing method allowed us to quickly discover and resolve issues, maintaining a consistent end-user experience.

1.6.2 Integration with Backend

RESTful APIs were used to provide seamless integration with the backend, allowing for bidirectional communication between the front and backend components. This made it easier to transmit data, authenticate users, and sync application state.

The integration of a JSON Web Token (JWT) authentication system improved the application's security by preserving user data and assuring authorized access to protected sites.

```
const handleChangeEmail = async () => {
    try {
      const response = await fetch(
        `http://${ipAddress}:8080/user/editProfile`,
        {
          method: "POST",
          headers: {
            "Content-Type": "application/json",
          },
          body: JSON.stringify({ email, password }),
        }
      );
      const data = await response.json();

      if (response.ok) {
        navigation.navigate("Email Change Confirmation");
      } else {
        Alert.alert(
          "Error",
          data.message || "Failed to change email. Please try again."
        );
      }
    } catch (error) {
      console.error("Error:", error);
      Alert.alert("Error", "An error occurred. Please try again
later.");
    }
  };
```

1.7 GIT Repository

For the whole implementation phase of the project, the GIT Version control was used along with repositories. A project management tool Trello was used to organize tasks. Trello helps to visualize and organize tasks into stages like “To Do”, “In Progress”, and “Done” using boards and cards. According to the features of the app, branches were separated and commits were managed and

also core features and additional features are shared among team members so that commitment is shared equally among every member of the team. All the Screenshots of the commits, Trello and branches that were divided according to the features of the app are available in Appendix.

1.8 Deployments/CI-CD Pipeline

The deployment pipeline is a critical component of the project, facilitating the seamless deployment of changes to the system while maintaining reliability and efficiency.

AWS CodePipeline and AWS CodeDeploy were used for setting up the CI/CD pipeline.

The process involved configuring CodePipeline to monitor the version control repository for changes, triggering the pipeline upon detection. During the build phase, CodePipeline pulled the latest code from the repository and initiated build scripts to compile the code. Subsequently, AWS CodeDeploy deployed the application, ensuring consistency across all instances.

EC2 instance connected to GitHub using CodeDeploy:

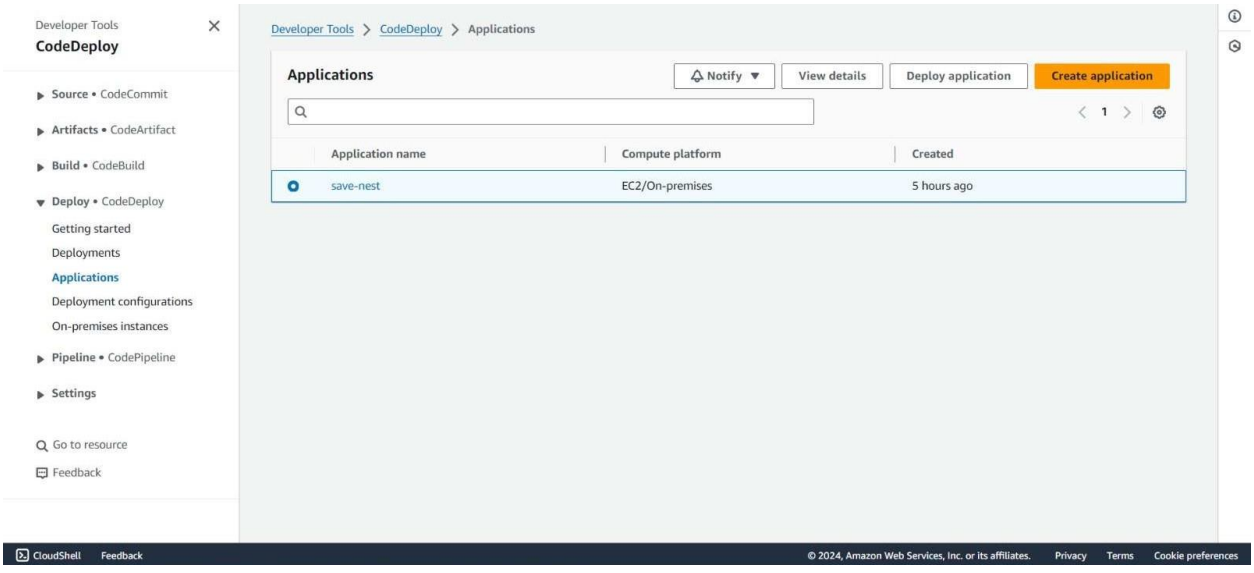


Figure 43 - CI-CD Pipeline 1

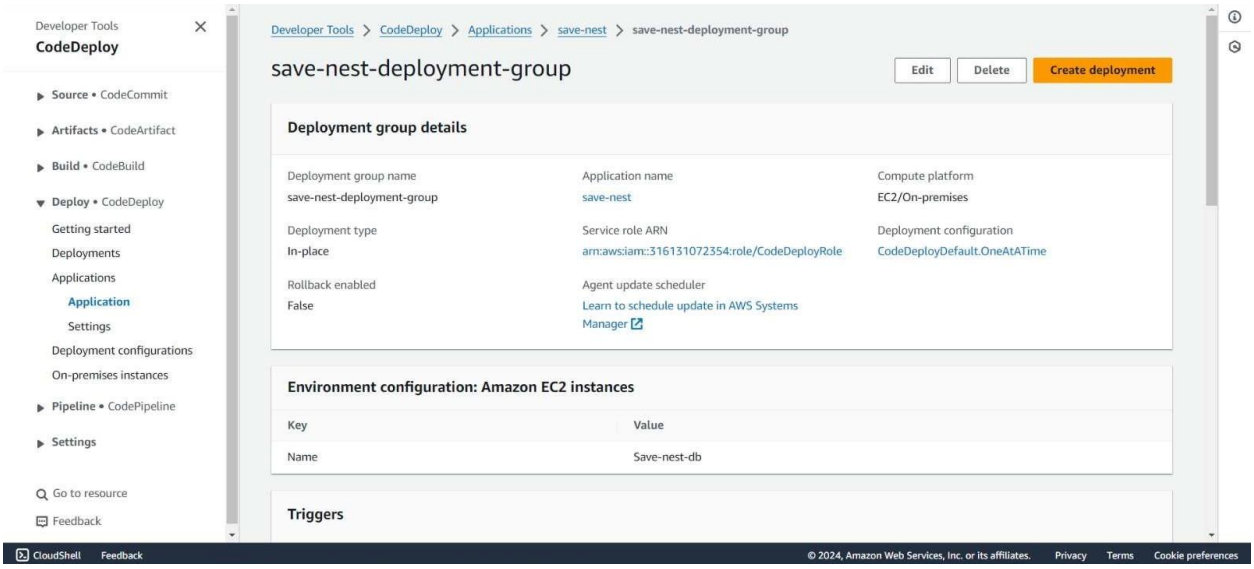


Figure 44 - CI-CD Pipeline 2

1.8.1. Advantages of the CI/CD Pipeline:

- Automation: The CI/CD pipeline automated the deployment process, reducing the need for manual intervention and minimizing the risk of errors.
- Rapid Deployment: With CI/CD, changes could be deployed quickly, enabling timely delivery of new features and updates to users.
- Consistency: Using CodeDeploy ensured consistent deployment across all environments, reducing configuration drift and maintaining reliability.

1.9 CRUD operations

This describes how every member of the team contributed to CRUD operations in the SaveNest Project.

1.9.1.Dumindu Gamage

Contribution:

- Implemented the createExpensesTotal function to create expenses total records.
- Provided input validation for the parameters.
- Ensured error handling and proper error message propagation.

Justification:

```
// Function to create expensesTotal
const createExpensesTotal = async (description, totalExpenses, budgetId) => {
  console.log(description);
  console.log(totalExpenses);
  console.log(budgetId);
  try {
    // Create expensesTotal directly with provided parameters
    const newExpensesTotal = await ExpensesTotal.create({
      description,
      totalExpenses,
      budgetId,
    });
    // Return the created expensesTotal
    return newExpensesTotal;
  } catch (error) {
    // Handle any errors and return an error response
    throw new Error(error.message);
  }
};

```

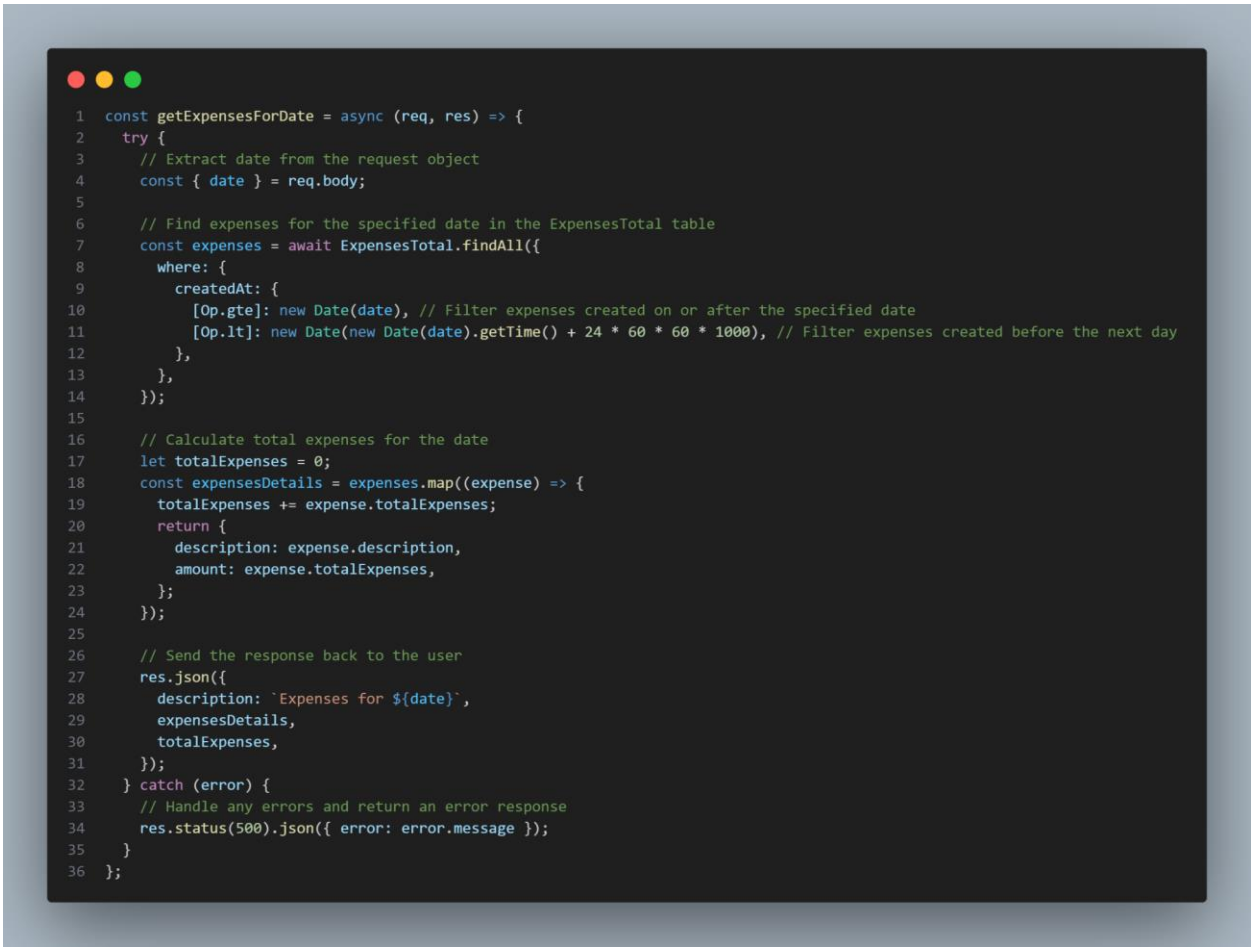
Figure 45 - CRUD operations 1

1.9.2.Rajeen Balasorriya

Contribution:

- Implemented the getExpensesForDate function to retrieve expenses for a specific date
- Implemented date filtering logic to fetch expenses within a specified date range.

Justification:



```
const getExpensesForDate = async (req, res) => {
  try {
    // Extract date from the request object
    const { date } = req.body;

    // Find expenses for the specified date in the ExpensesTotal table
    const expenses = await ExpensesTotal.findAll({
      where: {
        createdAt: {
          [Op.gte]: new Date(date), // Filter expenses created on or after the specified date
          [Op.lt]: new Date(new Date(date).getTime() + 24 * 60 * 60 * 1000), // Filter expenses created before the next day
        },
      },
    });

    // Calculate total expenses for the date
    let totalExpenses = 0;
    const expensesDetails = expenses.map((expense) => {
      totalExpenses += expense.totalExpenses;
      return {
        description: expense.description,
        amount: expense.totalExpenses,
      };
    });

    // Send the response back to the user
    res.json({
      description: `Expenses for ${date}`,
      expensesDetails,
      totalExpenses,
    });
  } catch (error) {
    // Handle any errors and return an error response
    res.status(500).json({ error: error.message });
  }
};
```

Figure 46 - CRUD operations 2

1.9.3.Savin Pathirana

Contribution:

- Implemented the CreateNotification function to create notifications for users.
- Provided error handling for notification creation

Justification:

```
// Function to create notification
async function CreateNotification(userId, notificationContent) {
  try {
    // Create notification in the Notification table
    const notification = await Notification.create({
      userId: userId,
      readStatus: 0, // Assuming false is represented by 0
      notificationContent: notificationContent,
    });

    // Return the created notification
    return notification;
  } catch (error) {
    // Handle errors
    console.error("Error creating notification:", error);
    throw new Error("Failed to create notification");
  }
}
```

Figure 47 - CRUD operations 3

1.9.4.Sakith Dissanayake

Contribution:

- Implemented the getNotification function to retrieve notifications for a specific user.
- Handled error cases for fetching notifications.

Justification:

```
//get notifications
async function getNotification(req, res) {
  const userId = req.user.id;

  try {
    // Find all notifications for the given user
    const notifications = await Notification.findAll({
      where: {
        userId: userId,
      },
    });

    // Respond with the found notifications
    res.status(200).json(notifications);
  } catch (error) {
    // Handle errors
    console.error("Error fetching notifications:", error);
    res.status(500).json({ error: "Failed to fetch notifications" });
  }
}

```

Figure 48 - CRUD operations 4

1.9.5.Himan Welagama

Contribution:

- Implemented the addBeverage function to add new beverages to the system
- Ensured error handling for adding beverages

Justification:

```
const addBeverage = async (req, res) => {
  try {
    const { beverageName, price, imgSrc, quantity, unitPrice } = req.body;

    const userId = req.user.id;

    // Ensure quantity is parsed as a number
    const parsedQuantity = parseInt(quantity);

    // Find the user's budget
    const userBudget = await Budget.findOne({
      where: { userId },
    });

    if (!userBudget) {
      return res.status(404).json({ message: "User budget not found." });
    }

    // Check if a selected beverage with the given beverageName already exists
    let existingSelectedBeverage = await SelectedBeverage.findOne({
      where: {
        beverageName,
        budgetId: userBudget.budgetId,
      },
    });

    if (existingSelectedBeverage) {
      // If it exists, update the quantity and total price
      const newQuantity = existingSelectedBeverage.quantity + parsedQuantity;

      // Update the values
      existingSelectedBeverage = await existingSelectedBeverage.update({
        quantity: newQuantity,
        totalPrice: unitPrice * newQuantity, // Update the total price as well
      });

      return res.status(200).json(existingSelectedBeverage);
    }

    // If it doesn't exist, create a new selected beverage
    const newSelectedBeverage = await SelectedBeverage.create({
      beverageName,
      price,
      budgetId: userBudget.budgetId,
      quantity: parsedQuantity, // Ensure parsed quantity is used
      totalPrice: unitPrice * parsedQuantity,
      unitPrice,
      imgSrc,
    });

    return res.status(201).json(newSelectedBeverage);
  } catch (error) {
    console.error(error);
    return res.status(500).json({ message: "Internal Server Error" });
  }
};
```

Figure 49 - CRUD operations 5

1.9.6.Kavindu Yapa

Contribution:

- Implemented the updateBeverage function to update beverage information in the system.
- Collaborated with team members to ensure the smooth flow of beverage updates.

Justification:

```
const updateBeverage = async (beverages, userBudget) => {
  let totalBeveragePriceAtTime = 0; // Initialize as a number
  try {
    let messages = [];

    // Iterate over each beverage in the request
    for (const beverage of beverages) {
      const { beverageName, totalPrice } = beverage;

      // Find the selected beverage by name and budgetId
      const selectedBeverage = await SelectedBeverage.findOne({
        where: {
          beverageName,
          budgetId: userBudget.budgetId,
        },
      });

      if (selectedBeverage) {
        // If the beverage already exists in the user's budget, update the spent amount
        const newSpentAmount =
          selectedBeverage.spentAmount + parseFloat(totalPrice);

        // Update the selected beverage's spent amount
        if (selectedBeverage.totalPrice - newSpentAmount >= 0) {
          await selectedBeverage.update({ spentAmount: newSpentAmount });

          // If beverage is successfully processed, add a success message
          messages.push(
            `'${beverageName}' spending amount is updated in the budget`
          );

          totalBeveragePriceAtTime += parseFloat(totalPrice);
        } else {
          // If there is not enough money left, add an error message
          messages.push(`There is no money left for '${beverageName}'.`);
        }
      } else {
        // If the beverage doesn't exist, add an error message
        messages.push(`Please add '${beverageName}' to your budget first.`);
      }
    }

    // Create expenses total for beverages
    await expensesTotalController.createExpensesTotal(
      "Beverages",
      totalBeveragePriceAtTime,
      userBudget.budgetId
    );

    // Return the messages array
    return { messages };
  } catch (error) {
    console.error(error);
    // Return an error message
    return { error: "Internal Server Error" };
  }
};
```

Figure 50 - CRUD operations 6

Chapter 2: Testing

1.1 Chapter Introduction

This chapter contains an in-depth overview of the SaveNest application's testing methods and results. To guarantee the reliability, functionality, and efficiency of the program, testing is an essential stage in the software development process. This chapter offers information about the testing procedures, standards, and outcomes that were attained throughout the testing stage.

1.2 Testing Criteria

The testing criteria for SaveNest contain a comprehensive evaluation of both functional and non-functional requirements to ensure the application's reliability and effectiveness. Functional testing focuses on validating specific features and functionalities outlined in the application requirements, ensuring that each component performs as intended and meets user expectations. This includes testing scenarios such as budget creation, transaction recording, data visualization, and price predicting to verify that SaveNest delivers the desired functionalities seamlessly. On the other hand, non-functional testing evaluates critical aspects such as performance, usability, security, and compatibility. Performance testing assesses the application's speed, responsiveness, and stability under various load conditions, ensuring optimal performance for users. Usability testing examines the user interface and overall user experience to ensure intuitiveness and ease of navigation. Security testing validates the application's robustness against potential vulnerabilities and threats, safeguarding users' financial data and privacy. Compatibility testing ensures that SaveNest operates smoothly across different platforms maximizing accessibility for users. By adhering to these strict testing criteria, SaveNest aims to deliver a high-quality and reliable finance management solution to its users.

1.3 Testing functional requirements

Requirements List		Priority Level	Expected Result	Actual Result	Status
FR1	User Authentication	Critical	The user must be able to create accounts and log in securely.	The application lets the user create accounts and log in securely.	Pass
FR2	Expense and Income Tracking through Image Analyzing	Critical	The user must be able to input their transactions from images or screenshots of their transaction recipients.	The user can add their transactions from images or screenshots of their transaction recipients.	Pass
FR3	Real-time budget balance updating	Critical	The application must be able to update the budget balance in real-time with analyzed image data or manually entered data to help the user reduce expenses.	The application updates the budget balance in real time with analyzed image data or manually entered data.	Pass
FR4	Real-time alerts	Critical	The user must be able to receive alerts when they approach or exceed budget limits in any category and also the user should be able to receive price fluctuation alerts on time.	The app sends alerts to the user when they exceed their budget in any category and when the price of a selected product changes in the supermarket.	Pass
FR5	Image Filtering	Critical	The application must be able to choose transaction recipients and filter unnecessary images	The application sends the user a prompt saying “Something went wrong, try again.” to the user if	Pass

			the user uploads.	the uploaded image is not a receipt.	
FR6	Graphical Representation of Transactions	Desirable	The application should be able to provide charts and graphs of the history of transactions and budget statuses to help the user understand their spending patterns.	The application shows the user expenses for the last 10 days in a graph and also shows how much the user has left to spend in each category.	Pass
FR7	User profile customizability	Luxury	The user should be able to customize their profile by adding profile pictures, custom backgrounds, ...etc	Customization features are currently unavailable and will be available in the future.	Fail

Table 2 - Testing functional requirements

1.4 Testing non-functional requirements

Requirements List		**Priority Level**	**Expected Result**	**Actual Result**	**Status**
NFR1	Performance	Critical	The application must be able to perform smoothly without any issues.	The application performs smoothly without any issues. But the machine learning part which predicts the next day's price of a stock takes around 3-5 minutes to make its prediction due to the training process.	Pass

NFR2	Security	Critical	The application must have robust encryption methods to guarantee the security and confidentiality of user information.	The application encrypts user information and user information are secured.	Pass
NFR3	Reliability	Critical	The application must minimize system downtime to provide a reliable service for the user.	The application is working without any system downtime and provides a reliable service.	Pass
NFR4	Usability	Critical	The application must have a user-friendly interface, making it easy for the user to navigate.	The application has a user-friendly easy-to-use interface making it easy for the user to navigate.	Pass
NFR5	Scalability	Critical	The application must be able to handle multiple users without losing any performance.	The application handles multiple users without any issues.	Pass
NFR6	Compliance	Desirable	The application should comply with relevant financial regulations and laws.	The application follows relevant financial regulations and laws.	Pass
NFR7	Documentation	Luxury	The application should provide the user with user guides and tutorials.	The app will guide the user regarding their technical queries using the chatbot. Other guides and tutorials regarding finance are yet to be implemented.	Partially done.

Table 3 - Testing non-functional requirements

1.5 **Unit testing**

Unit testing for a mobile application backend involves testing individual components like APIs, databases, and services, ensuring functionality. Frontend unit testing validates UI components and interactions for a seamless user experience.

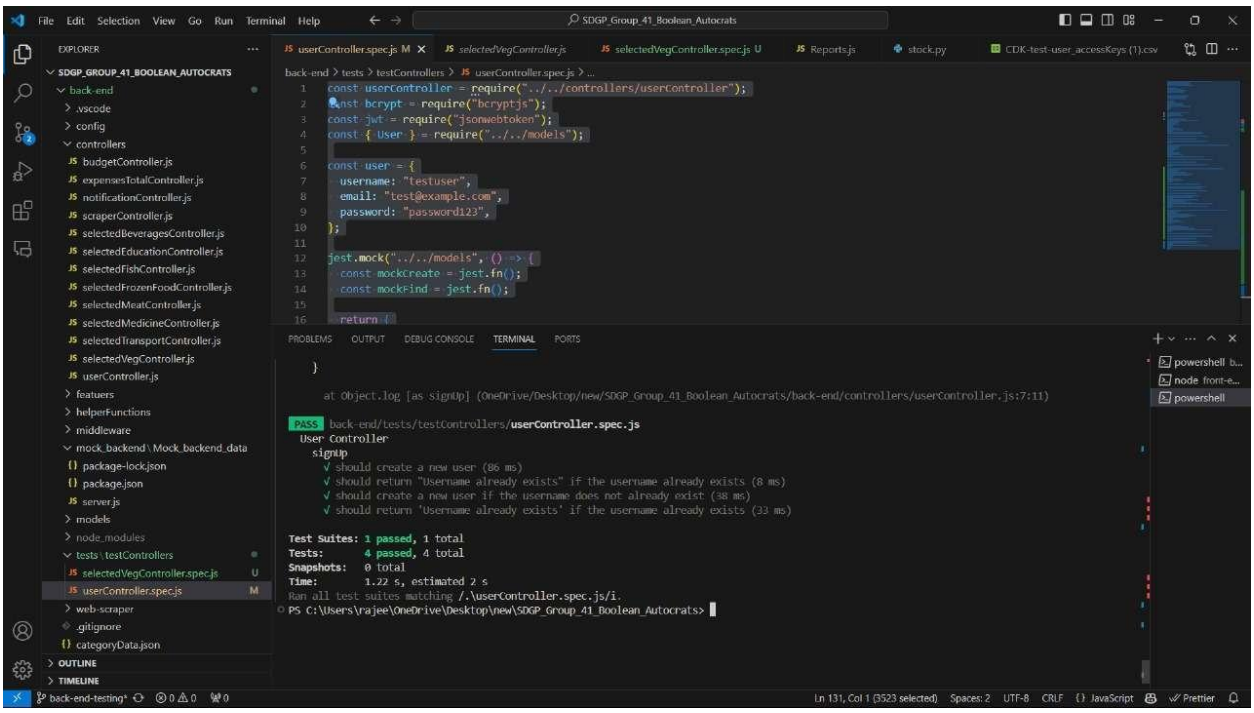


Figure 51 - Unit testing 1

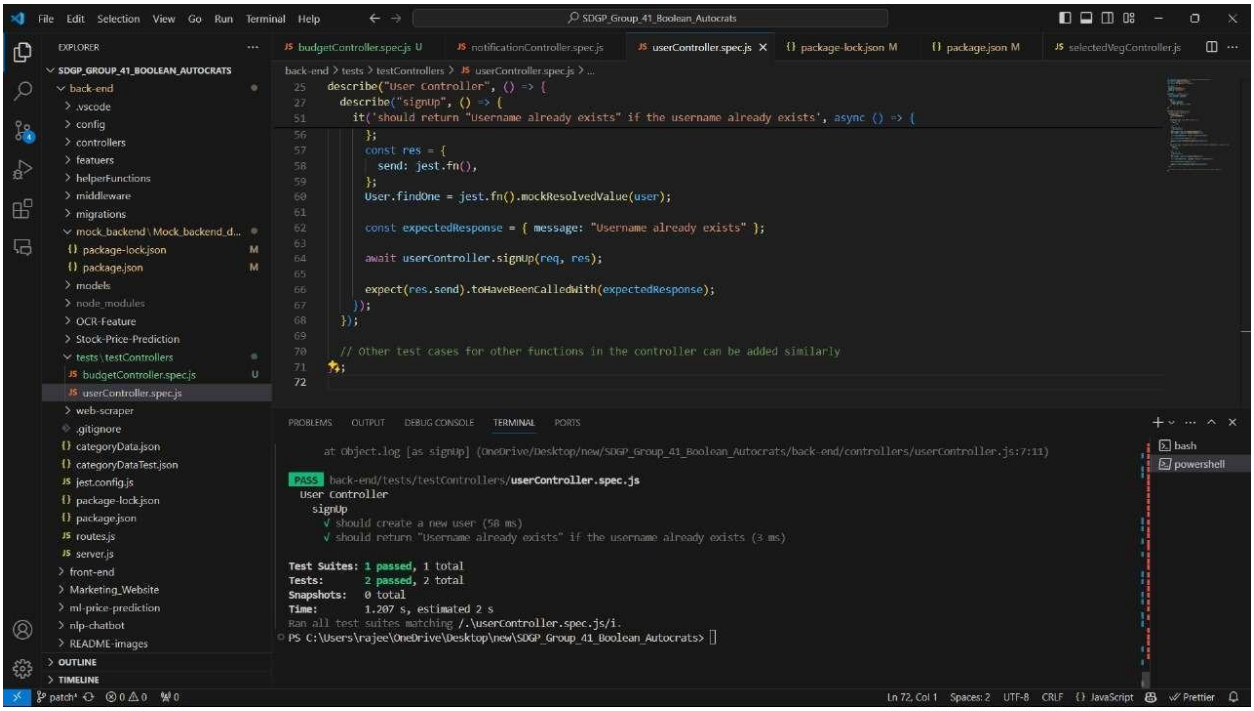


Figure 52 - Unit testing 2

The unit testing conducted on the user detail management backend of SaveNest yielded positive results across various key functionalities. Registration of new users proceeded smoothly, with accurate details stored securely in the database. Users could seamlessly update their profile information without encountering errors, ensuring data integrity. Password management functionalities, including change and reset options, operated securely, with passwords encrypted and stored safely. Email verification during registration worked effectively. The backend demonstrated robust error handling, gracefully managing exceptions and providing informative feedback to users. Security measures such as encryption and access controls were implemented rigorously to safeguard user data. Overall, the thorough unit testing confirms the reliability, security, and seamless functionality of the user detail management backend within the SaveNest application.

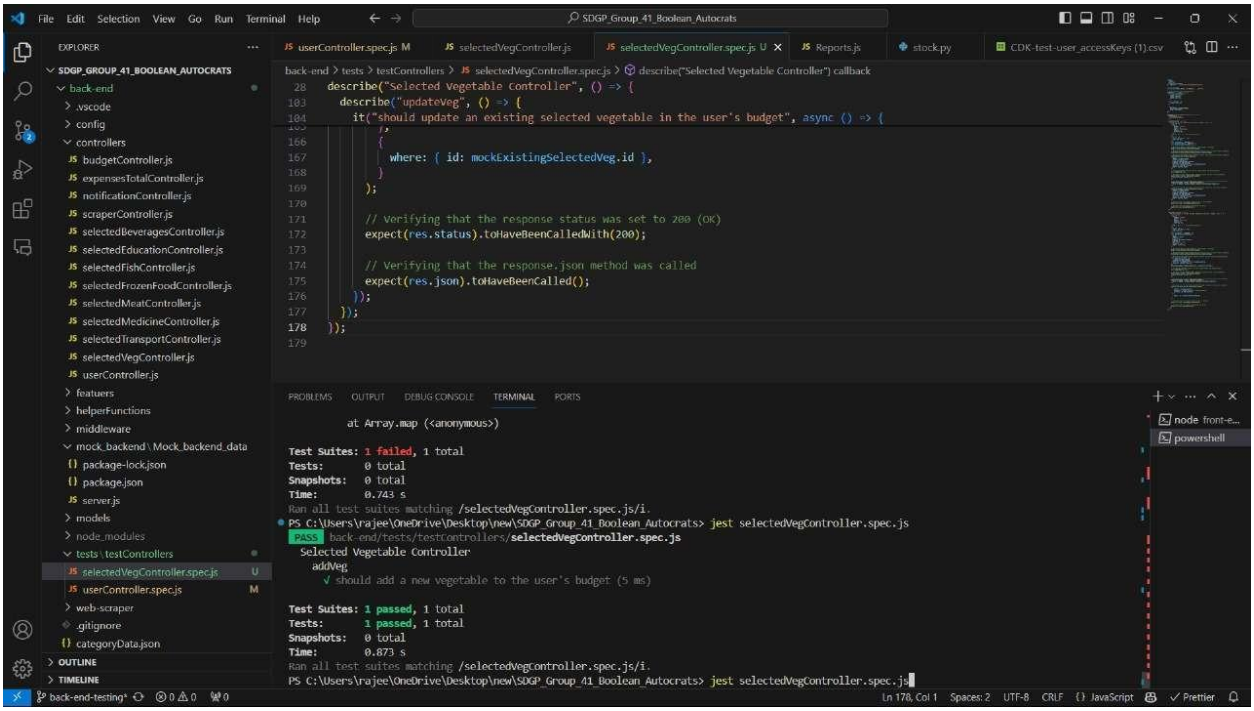


Figure 53 - Unit testing 3

In the unit testing phase of the backend code for the vegetable purchasing budget management in SaveNest, we are pleased to report that all aspects have passed successfully. This comprehensive testing evaluated various functionalities such as budget planning, and expense tracking ensuring that each component operates effectively and accurately. The backend code demonstrates robustness in handling user inputs, processing data, and generating accurate budget recommendations. This successful outcome confirms the reliability and functionality of the vegetable purchasing budget management underscoring SaveNest's commitment to providing users with a seamless and efficient budgeting experience for their vegetable shopping needs.

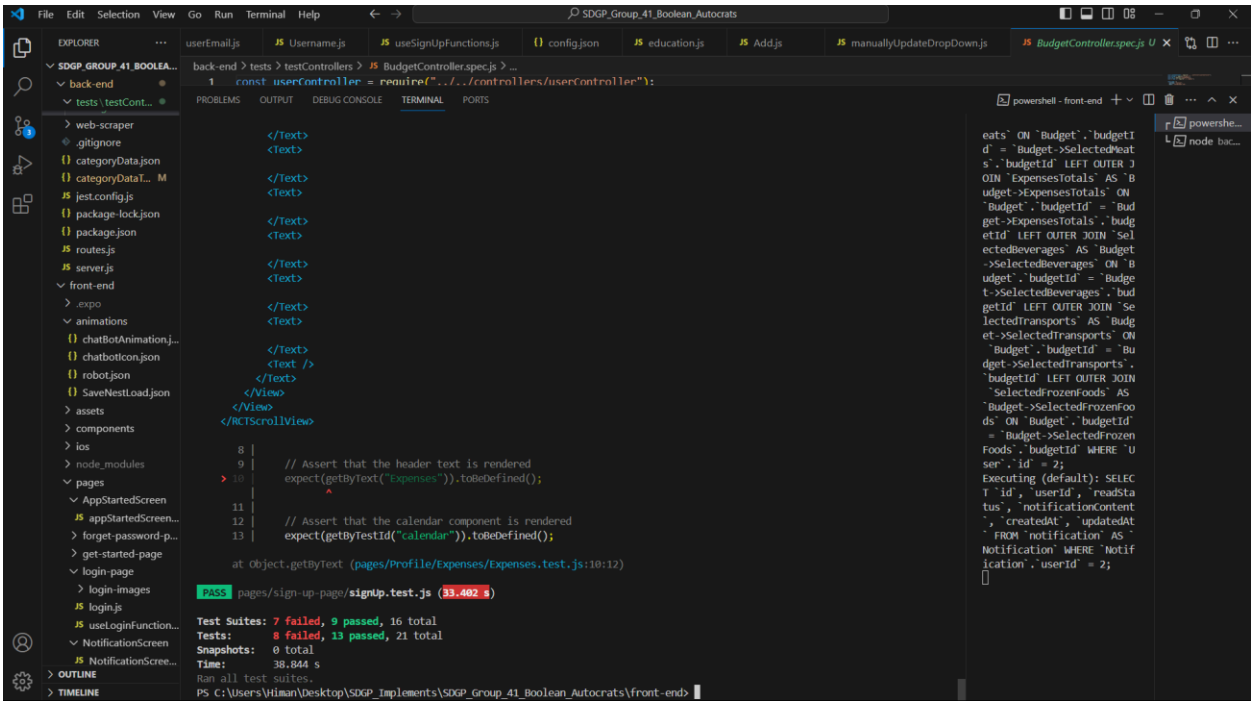


Figure 54 - Unit testing 4

During the unit testing of every frontend component in SaveNest, developers encountered a mix of errors and successful outcomes. Some components passed the tests seamlessly, demonstrating robust functionality and adherence to design specifications. However, developers also identified errors in certain components, indicating potential issues that require further attention and debugging. These errors may range from UI inconsistencies to functional discrepancies, highlighting areas for improvement to ensure a cohesive and error-free user experience across all frontend elements. By addressing these errors and refining the affected components, developers aim to enhance the overall quality and reliability of the SaveNest application frontend.

1.6 Usability testing

Usability testing evaluates the ease of use and user experience of a product by observing real users interact with it. Through tasks and scenarios, testers identify user interface issues, navigation challenges, and overall satisfaction levels, informing design improvements to enhance the product's usability and effectiveness.

2.7.1. Get Started Page

Figure 55 - Get Started Page

Get started page works perfectly when a user opens the SaveNest mobile application for the very first time on their devices. The user can press the get started button to redirect into the register page.

2.7.2. Register Page

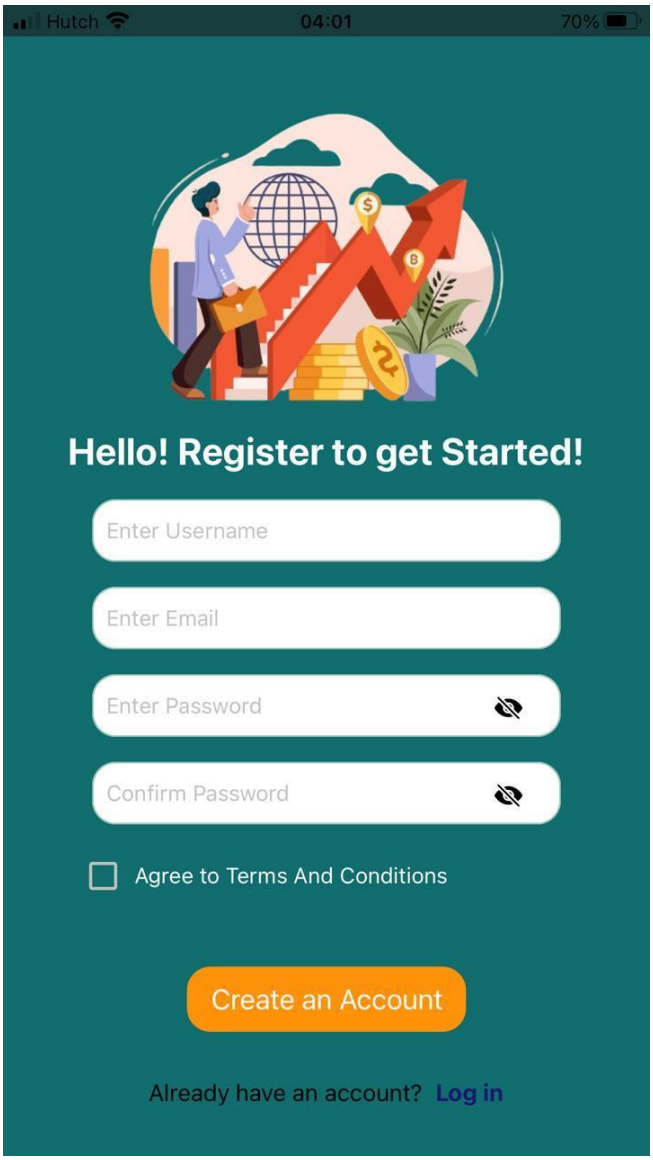


Figure 56 - Register Page

Here the user can experience the register page. Users can enter a user name that he or she prefers and enter their email address and password to register for the application. Now users can agree to terms and conditions to create an account with the press of a button. If he or she has already registered in the application and tries to log in with a different device they can simply use the log-in option at the bottom of the page which will redirect them to the login page.

2.7.3. Login Page

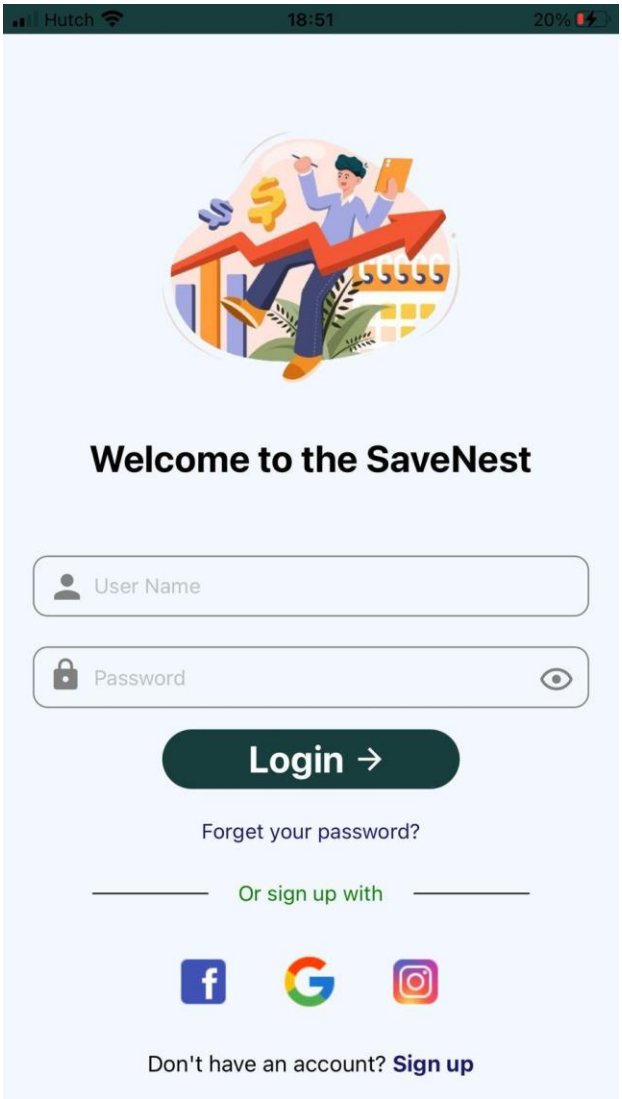


Figure 57 - Login Page

The login page works perfectly. with the user entering their username on username blank and password on password and they can simply login. If users forget their passwords they can recover it through the forget your password button. Also, users can login via their socials or sign up if they are not members using the sign-up button at the bottom of the page.

2.7.4. Budgeting Screen

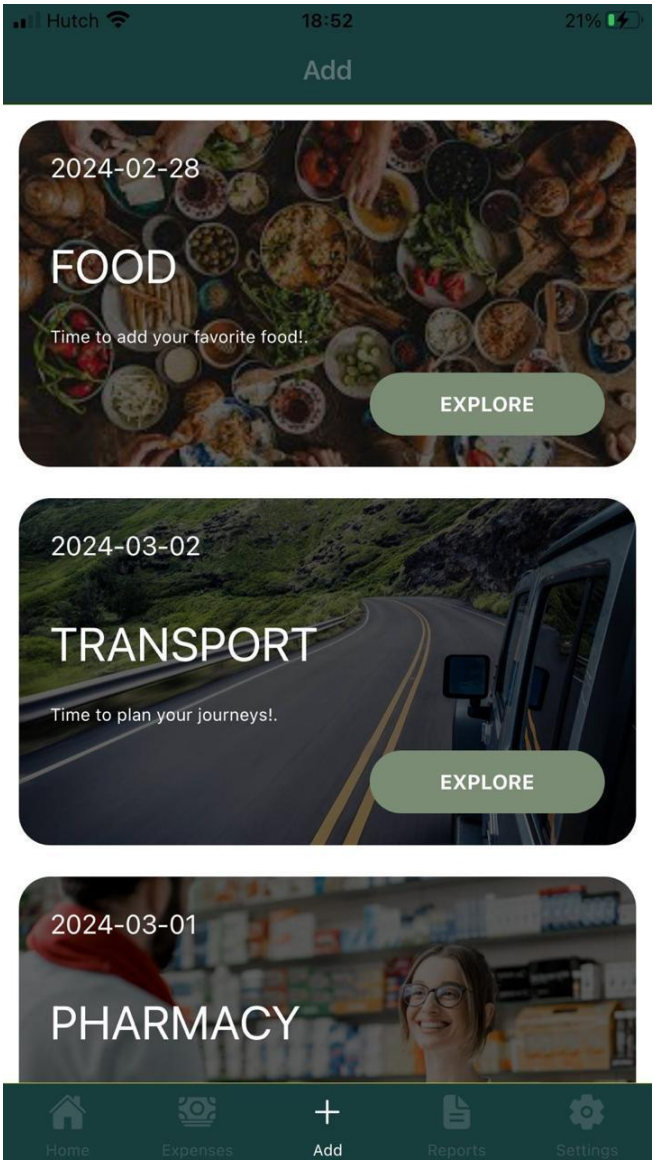


Figure 58 - Budgeting Screen

The budgeting screen can be used to add items to the user's budget and the budgeting screen offers users a variety of budgeting areas. This page works fine and gives easy access to budgeting for the user.

2.7.5. Home Page

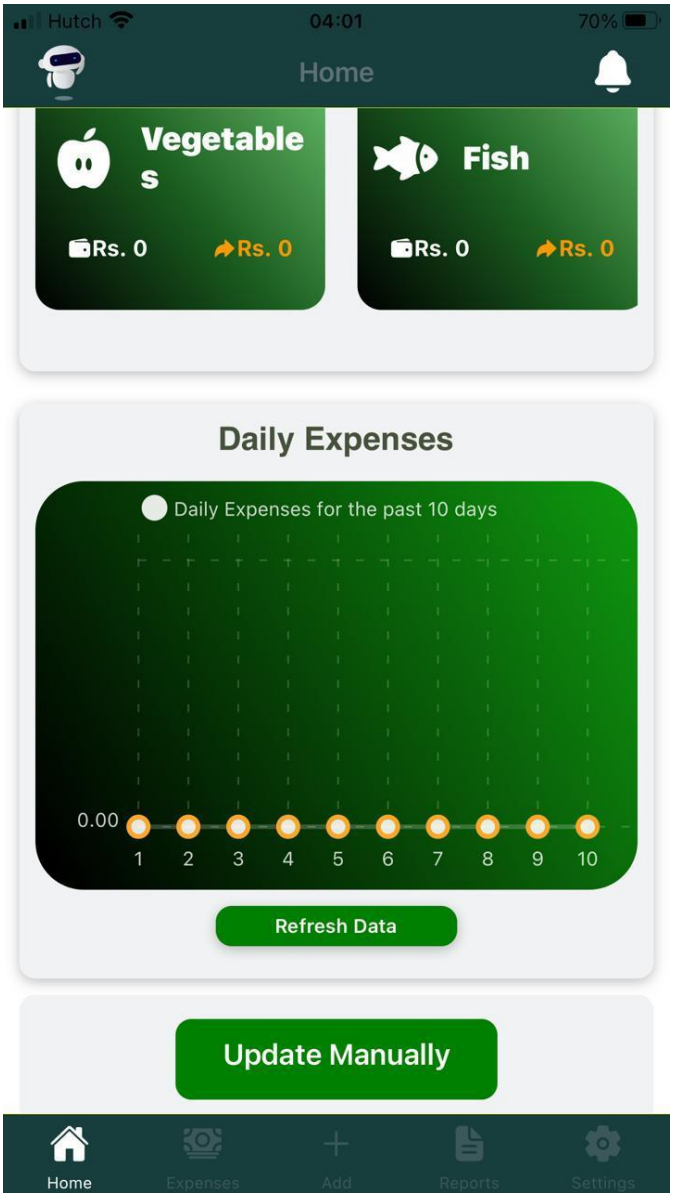


Figure 59 - Home Page

The home page is a simple and easy-to-use interface for the user. Widgets at the top show the user what categories the user spends money on and the stats graph shows how much the user spends on the past few days.

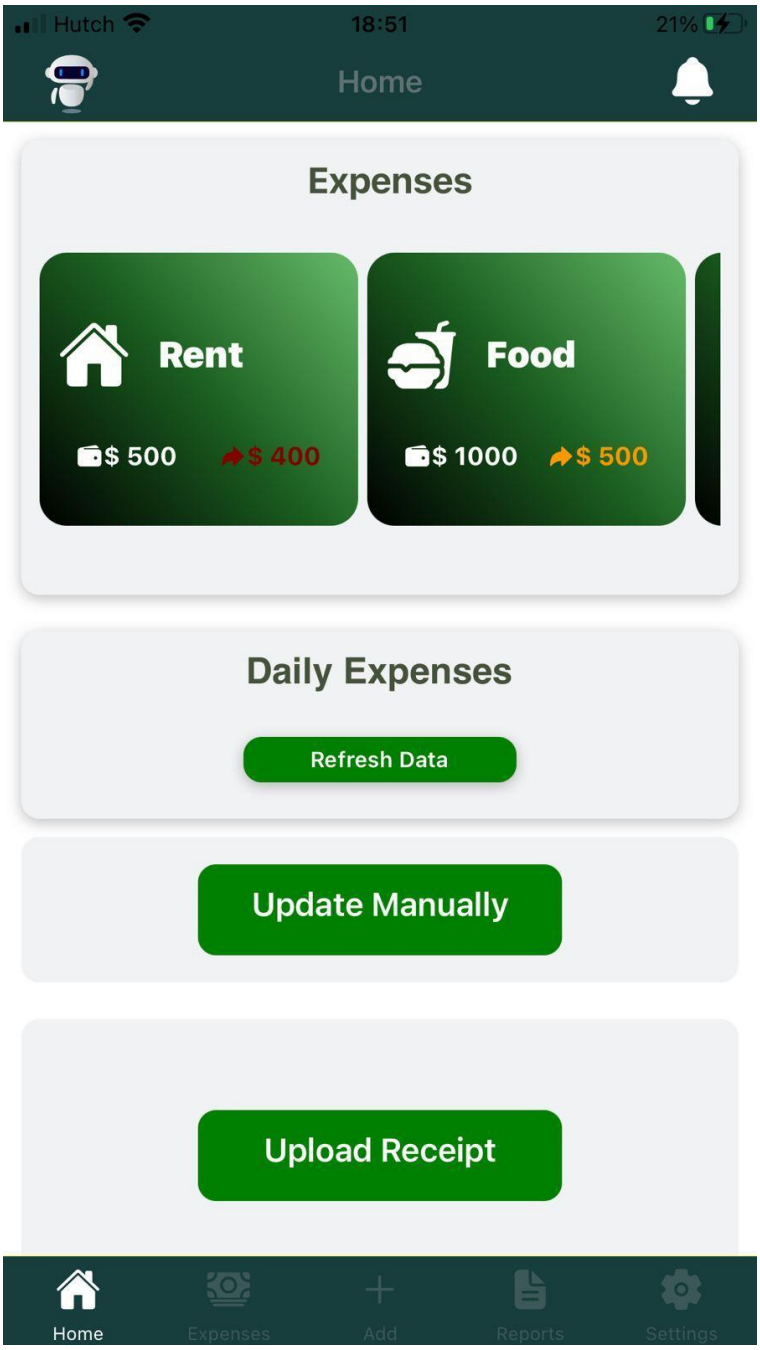


Figure 60 - Home Page 2

In the home tab, users can update their budget manually or use photographs of the bills. The app uses OCR technology to update the budget using photographs of the bills. To do that simply press upload receipt and experience the exclusive feature on the SaveNest financial management page.

2.7.6. Expenses page

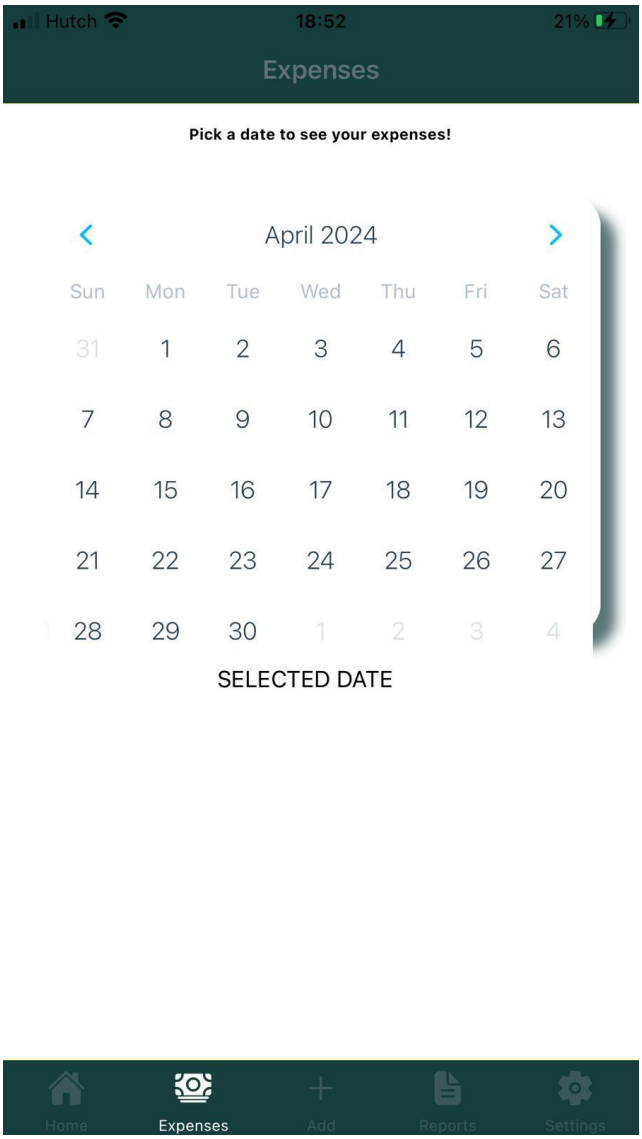


Figure 61 - Expenses page

In the expenses page, users can choose a day to check their expenses, so they can keep track of their payments using this page. Works fine and is easy to use.

2.7.7. Update expenses page

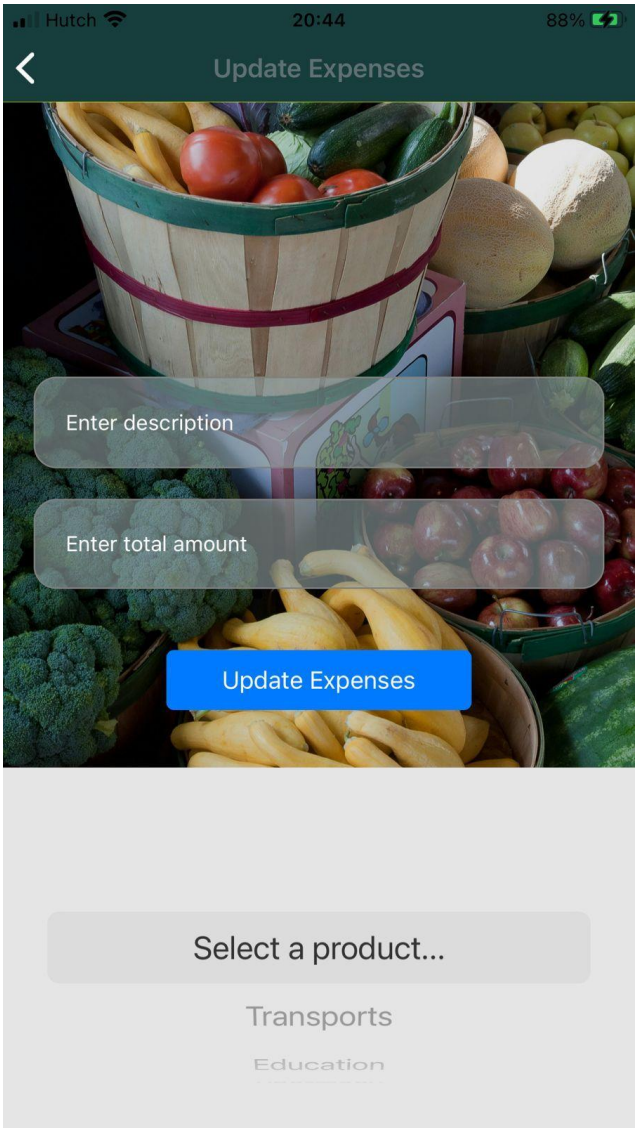


Figure 62 - Update expenses page

Users can edit their expenses, he or she simply need to give a description about their expenses that happened, the category in which they did the payments, and the total amount to update. This is the manual budget updating method.

2.7.8. Chatbot Page

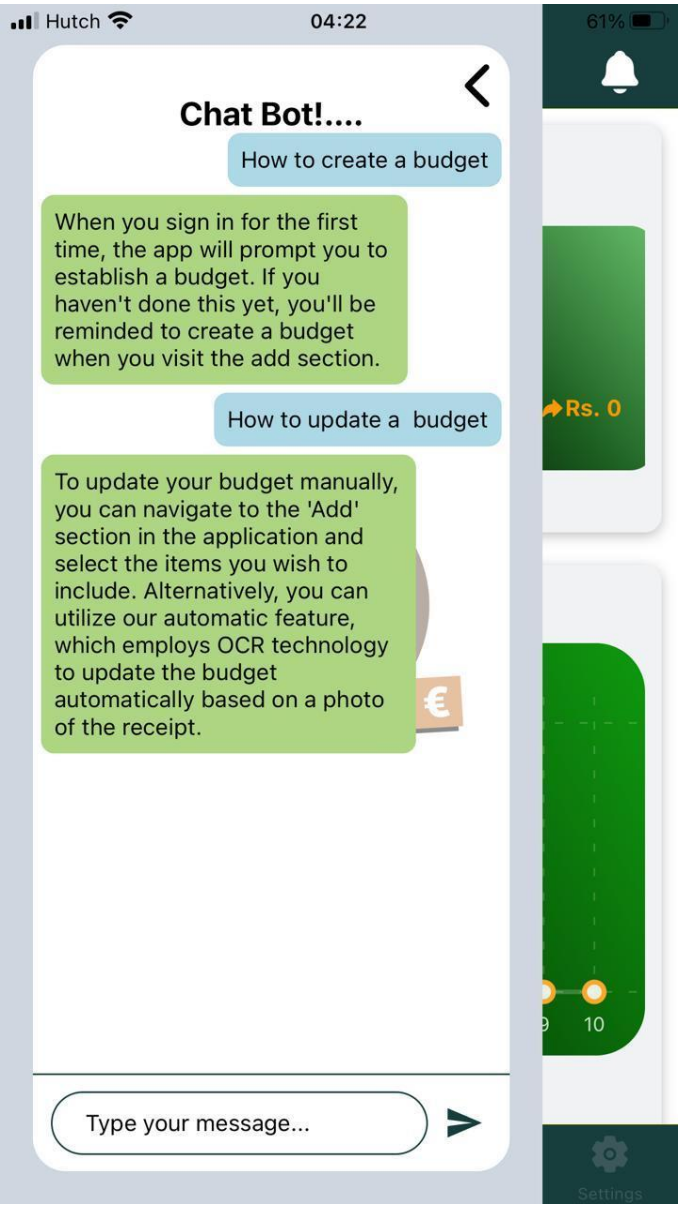


Figure 63 - Chatbot Page

Chat bot is the SaveNest user's ultimate companion to call for simple problems arises during the use of the app. Chat bot is trained for many different inputs from the user and works smoothly as expected.

2.7.9. Settings Page

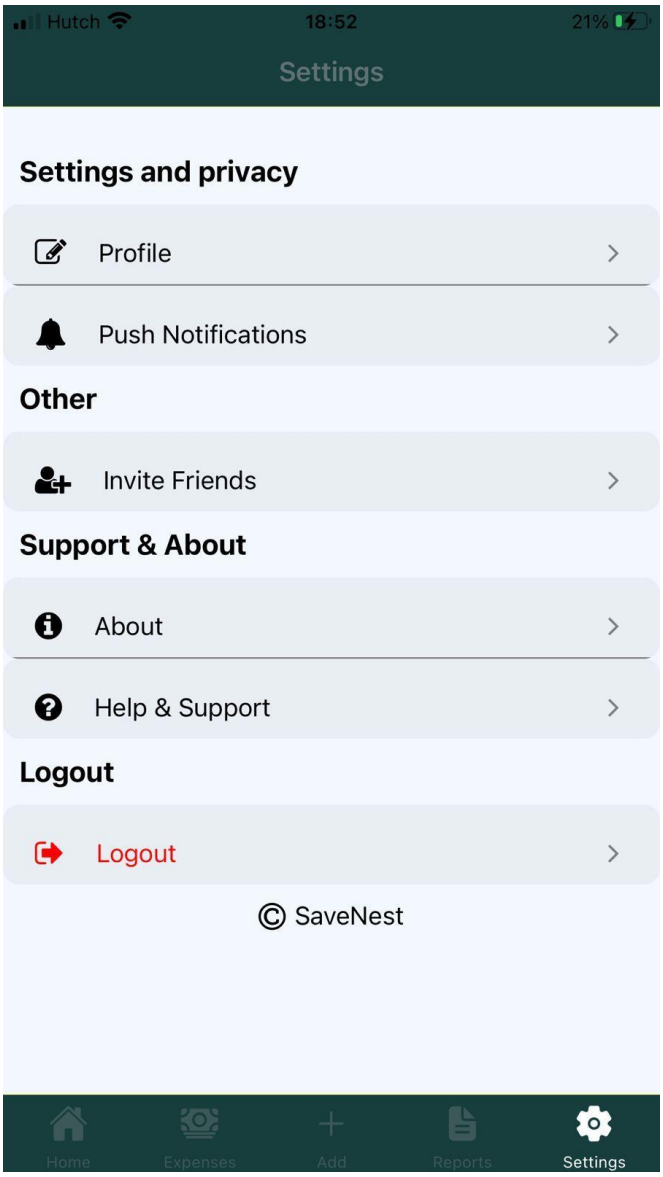


Figure 64 - Settings Page

From the settings page users can customize the app according to their taste as well as request support from customer support and logout from the account using the logout button. This page and its functions works perfectly,

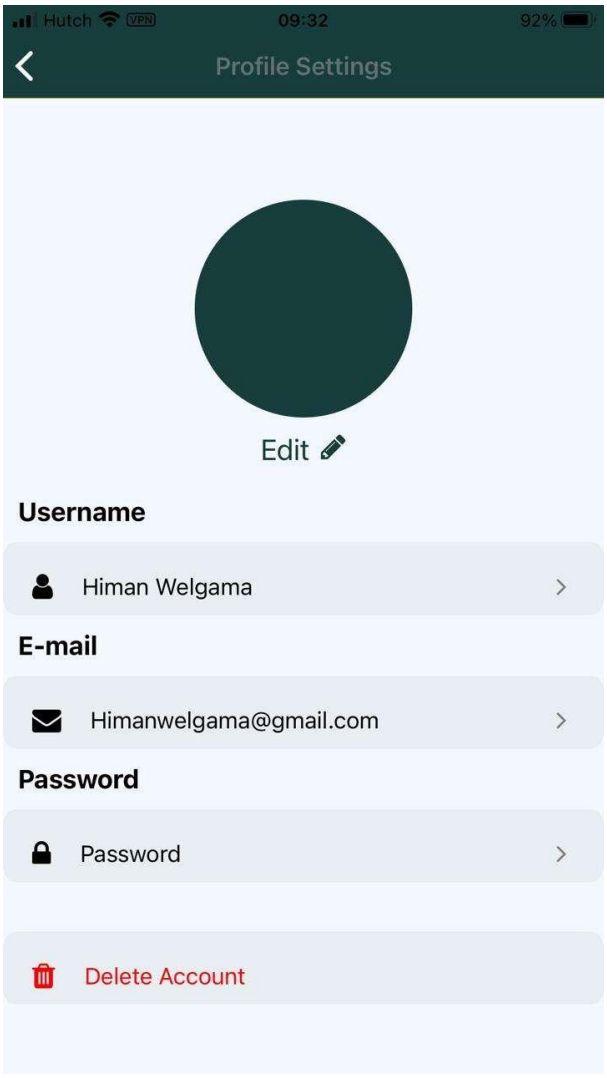


Figure 65 - Profile Settings

From the settings menu user can access to the profile settings where user can edit their profile picture, username, the active email of their account and the password he or she uses. These functions also work here. Also users can delete their SaveNest account if he or she likes to leave the platform at any time.

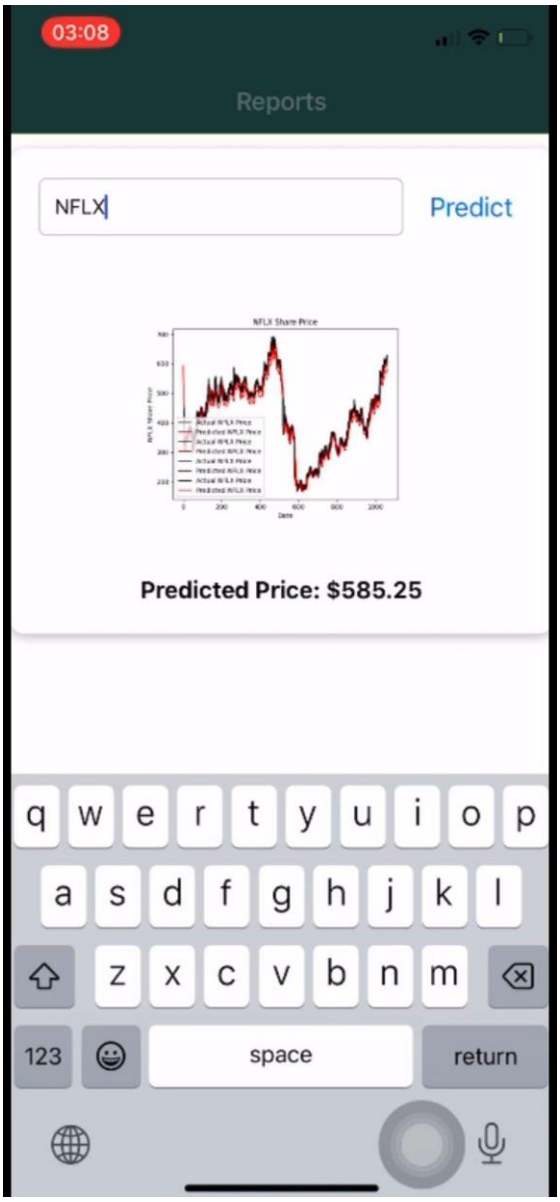


Figure 66 - Reports Page

In the reports page the user can search for stocks' predicted price using the search bar and the app will successfully predict the price of the requested stock by the user. So users can invest in stocks more productively.

1.7 Compatibility testing

The "SaveNest" mobile application has successfully passed all compatibility testing criteria, ensuring seamless functionality and performance across a diverse range of mobile devices. Through rigorous testing efforts, the application has demonstrated its capability to adapt and perform optimally on various smartphones and tablets, accommodating different screen sizes, resolutions, and hardware configurations without compromising user experience. Furthermore, "SaveNest" exhibits compatibility with multiple versions of popular operating systems such as iOS and Android. With its successful completion of compatibility testing, "SaveNest" ensures a consistent and reliable experience for all users, regardless of their device preferences or operating system choices.

1.8 Chapter Summary

This chapter focused on ensuring the quality and reliability of our project through comprehensive testing. We outlined the criteria used for testing and provided detailed test results for both functional and non-functional requirements. Unit testing validated individual components, while performance, usability, and compatibility testing assessed system capabilities and user experience. Screenshots and justifications were provided as evidence of our rigorous testing approach.In summary, this chapter demonstrates our commitment to delivering a high-quality product that meets stakeholder expectations across various dimensions

Chapter 3: Evaluation

3.1 Chapter Overview

This chapter focuses on the critical process of evaluating the "SaveNest" finance management application. It outlines the various evaluation methods used, including both quantitative and qualitative approaches. Additionally, the chapter highlights the importance of self-evaluation by team members. Through this comprehensive evaluation, the chapter aims to provide insights into the performance and effectiveness of the application, ensuring its alignment with user needs and expectations.

3.2 Evaluation methods

A diverse array of evaluation methods are employed to assess the efficacy, usability and impact on users' financial management of the "SaveNest" finance management application. These methods include both quantitative and qualitative approaches, ensuring a comprehensive evaluation process.

Quantitative evaluation methods involve the collection and analysis of numerical data to measure specific metrics such as task completion rates, time taken to accomplish tasks, and error rates. Through techniques like usability testing, evaluation surveys, and analytics tracking, quantitative data provides valuable insights into the efficiency and effectiveness of the application's features and functionalities. Similarly, online evaluation surveys distributed to a representative sample of users can gather quantitative feedback on various aspects of the application, such as user satisfaction levels, frequency of app usage, and perceived ease of use through analyzing results of numerical ratings and statistical data.

Justifications for the selection of quantitative evaluation methods are provided, emphasizing their ability to provide objective and measurable data that can inform

decision-making and iterative improvements. By quantifying user interactions and behaviors, these methods enable researchers to identify patterns, trends, and areas of improvement in a systematic manner.

Conversely, qualitative evaluation methods focus on gathering in-depth feedback and insights from end users, domain experts, and industry professionals. These methods include user testing sessions, focus groups, and interviews, which allow for a deeper understanding of user preferences, pain points, and overall satisfaction with the application. Not only that but also from the online evaluation survey included questions that are open - ended to obtain comprehensive feedback regarding user experiences, difficulties and suggestions for enhancement. Qualitative data can uncover subtleties that quantitative metrics may overlook, providing rich contextual information that helps contextualize user experiences and behaviors.

Justifications for the use of qualitative evaluation methods are elucidated, highlighting their ability to capture detailed feedback, uncover usability issues, and identify areas for enhancement that may not be evident through quantitative metrics alone. By engaging directly with users and stakeholders, qualitative methods foster empathy and understanding, facilitating the development of user-centered solutions that address real-world needs and concerns.

The integration of quantitative and qualitative assessment techniques results in a thorough and diverse evaluation of the "SaveNest" application, facilitating the identification of its advantages and disadvantages as well as areas for improvement to ultimately improve user experience and meet the project's objectives.

3.3 Quantitative evaluation

Quantitative evaluation of unit testing using the Jest framework provides a systematic approach to measuring and analyzing the effectiveness and quality of code testing. By writing automated tests to verify individual units of code, developers can objectively assess their codebase's robustness. Metrics such as test coverage, passing and failing tests, test execution time, and code complexity are quantitatively evaluated to gauge the thoroughness of testing efforts. This data analysis allows for comparisons against predefined benchmarks or standards, informing decision-making processes regarding code quality, release readiness, and resource allocation. Insights derived from quantitative evaluation drive continuous improvement initiatives, guiding teams in refining their testing practices to deliver higher-quality software products. Ultimately, quantitative evaluation with Jest facilitates a data-driven approach to software testing, enhancing overall development efficiency and product reliability.

3.3.1. Online Evaluation Survey

The contribution of surveys to quantitative evaluation is explained in the “Evaluation methods” section. Screenshots of the questions in the evaluation questionnaire along with the obtained responses from the online evaluation survey are included in Appendix C - 1. And the analysis of the results of the evaluation survey is included in Appendix C - 2.

3.4 Qualitative evaluation (Feedback from end users, domain experts and industry experts)

3.4.1 Feedback from End Users

SaveNest caters will have a diverse range of end users seeking efficient financial management solutions. Individual consumers, small business owners, freelancers, financial professionals all benefit from its user-friendly tools. Whether it's budgeting, expense tracking or stock market price prediction SaveNest provides tailored features to help users achieve their financial goals effectively.

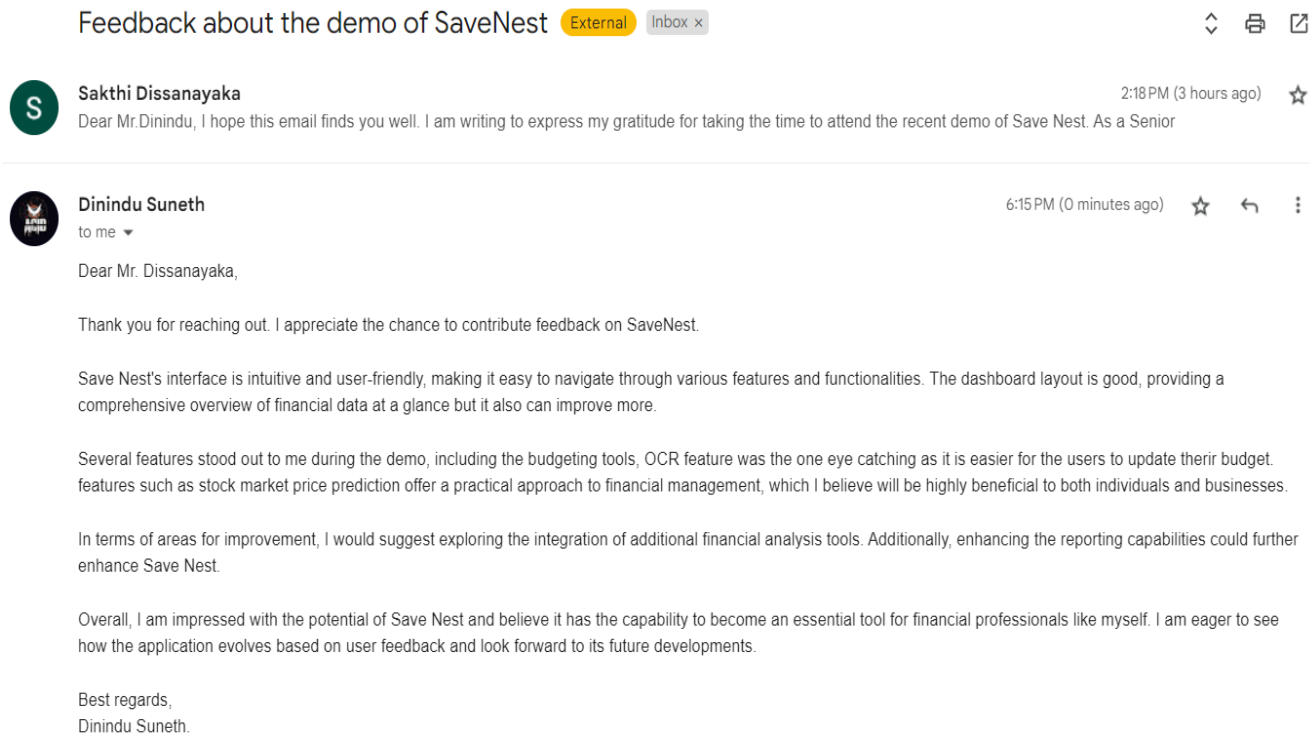
- Kalum Dissanayaka (Gem Businessman, Ratnapura)
  As a busy businessman, managing finances can often be overwhelming.
  Its user-friendly interface and features have streamlined our financial management processes, allowing us to track expenses. One of the standout features for me is the OCR feature, as I am a busy businessman this allows me to quickly update my expenses. The budgeting tools have helped us identify areas where we can optimize spending and allocate resources more efficiently.

- Imaya Chiranthi Herath (HSBC, Havelock)
  As a bank worker, I encounter numerous customers seeking advice on financial management every day. SaveNest might be a recommendation for individuals looking to take control of their finances. One of the standout features of SaveNest is its ability to predict stock market prices so users can efficiently invest money on stocks. Moreover, the budgeting tools offered by SaveNest are incredibly helpful in assisting customers to track their spending and identify areas where they can save more effectively. SaveNest has proven to be an invaluable tool for me. Its user-friendly interface, comprehensive features, made me recommend it to anyone looking to improve their financial well-being.

- Dr. Premajayantha (Doctor, Divisional Hospital Ratnapura)

  As a doctor, my schedule is often hectic, leaving me with little time to manage my finances effectively. SaveNest's intuitive interface and features make it easy for me to track expenses, set savings targets, and monitor my financial progress on the go. Additionally, the budgeting tools offered by SaveNest have helped me identify areas where I can cut back on expenses and allocate more resources towards savings.This proactive approach to financial management has allowed me to achieve my financial goals more efficiently and with less stress.

3.4.2 Industrial Experts



Feedback about the demo of SaveNest External Inbox ×

Sakthi Dissanayaka 2:18 PM (3 hours ago)

Dear Mr.Dinindu, I hope this email finds you well. I am writing to express my gratitude for taking the time to attend the recent demo of Save Nest. As a Senior

Dinindu Suneth 6:15 PM (0 minutes ago)

to me

Dear Mr. Dissanayaka,

Thank you for reaching out. I appreciate the chance to contribute feedback on SaveNest.

Save Nest's interface is intuitive and user-friendly, making it easy to navigate through various features and functionalities. The dashboard layout is good, providing a comprehensive overview of financial data at a glance but it also can improve more.

Several features stood out to me during the demo, including the budgeting tools, OCR feature was the one eye catching as it is easier for the users to update therir budget. features such as stock market price prediction offer a practical approach to financial management, which I believe will be highly beneficial to both individuals and businesses.

In terms of areas for improvement, I would suggest exploring the integration of additional financial analysis tools. Additionally, enhancing the reporting capabilities could further enhance Save Nest.

Overall, I am impressed with the potential of Save Nest and believe it has the capability to become an essential tool for financial professionals like myself. I am eager to see how the application evolves based on user feedback and look forward to its future developments.

Best regards,
Dinindu Suneth.

Figure 67 - Industrial Experts

Mr. Dinindu Suneth is working as a senior credit officer in a leading financial institution in Sri lanka. As SaveNest's team was able to demonstrate the product to Mr. Dinindu, he was kind enough to provide the team with valuable feedback about the SaveNest application.

3.5 Self evaluation

3.5.1. Dumindu Gamage

As the Team leader of our university software development group project, I have played a pivotal role in guiding our team through the implementation phase. In addition to leading our team effectively, I took the initiative to expand my technical skills by learning new technologies crucial to our project's success, including React Native and Node.js. Embracing these technologies, I actively participated in the development process, providing hands-on support to my teammates as we navigated the challenges of incorporating them into our project. I facilitated knowledge-sharing sessions to ensure everyone in the team had a solid grasp of these technologies and their relevance to our project objectives. Moreover, I maintained transparent communication channels to keep both our team and project stakeholders updated on our progress and any hurdles encountered during the adoption of these new technologies. This experience not only bolstered my technical prowess but also emphasized the significance of adaptability and lifelong learning in the dynamic field of software development. Looking ahead, I am committed to further refining my expertise in React Native, Node.js, and other emerging technologies, thereby enhancing my contributions to future projects and fostering an environment of innovation within our university software development community.

3.5.2. Rajeen Balasooriya

SaveNest has been an enriching journey where I delved into React Native, Node.js, and MySQL to develop a cross-platform mobile application. Learning React Native provided me with insights into mobile app development, enabling me to create intuitive user interfaces and navigation structures. Additionally, mastering Node.js and MySQL empowered me to build a robust backend infrastructure, handling data storage, and API integrations efficiently. Throughout the project, I honed my project management skills, breaking down tasks and prioritizing them effectively to

meet deadlines. While I'm proud of the progress made, I recognize areas for improvement, particularly in optimizing performance and implementing advanced features. Nevertheless, SaveNest has been a valuable learning experience that has equipped me with practical skills and a deeper understanding of software development principles. Moving forward, I'm excited to continue refining the project and exploring new technologies to enhance its functionality and user experience.

3.5.3. Himan Welgama

I have made significant progress in implementing the front- end component of the SaveNest project using React Native. I have successfully developed various screens such as the Login page, Budgeting Screens, Settings pages, Forget password pages, Chatbot page and navigated them. This demonstrates a comprehensive understanding of React Native and its components-based architecture.

Moreover, I have also begun integrating some of the front-end components with the backend endpoints, such as handling user authentication and profile management. This shows initiative in bridging the gap between the front-end and back-end components of the application, which is crucial for the overall functionality and user experience.

In terms of project management, My use of Git version and Trello for task organization is commendable practices.

Moving forward, continue focusing on integrating more front-end components with back-end endpoints, ensuring smooth communication and data flow between the two layers of the application. Additionally, prioritize thorough testing to deliver a stable and reliable product to end users.

3.5.4. Savin Pathirana

I'm really satisfied with what our group accomplished during this project. We faced a bunch of challenges, but with everyone's help, we managed to tackle them and finish the project successfully.

I took care of the machine learning part, which was completely new to me. Thanks to the support of our team, I was able to learn a lot about machine learning and APIs. We had a bit of a problem

when we realized there weren't enough changes in product prices to make our prediction model work. So, we decided to switch gears and predict stock prices instead, which turned out to be a good move for helping users with their finances.

Other than that, developing the chatbot for technical queries went smoothly. I got to learn a bunch of new tools like Node.js, Python, TensorFlow, Keras, NLTK, Flask, and matplotlib, and also got some hands-on experience with creating and training machine learning models and hosting them.

Overall, this project was a great learning experience for me. It's given me a bunch of new skills and knowledge that I know will be useful in the future. I'm grateful to have been a part of such a supportive and collaborative team.

3.5.5. Sakith Dissanayaka

Working with my team was great. We shared ideas and helped each other solve problems. This made the project more enjoyable and we were able to finish it together. This project taught me to always be ready to learn new things. I kept an open mind and tried new technologies and ways of working.During my time on the "SaveNest" project, I developed skills in Node.js, React Native, and web development. I learned to build scalable backend solutions using Node.js, create responsive mobile interfaces with React Native, and design user-friendly web interfaces using HTML, CSS, and JavaScript. These experiences have greatly enhanced my professional skills. When faced with challenges, I learned to stay positive and find solutions. This helped me grow as a problem-solver and become more confident in my abilities.

Overall, working on "SaveNest" was a valuable experience. I gained new skills, faced challenges, and became better at what I do. I'm proud of what we accomplished and excited to use what I've learned in future projects.

3.5.6. Kavindu Yapa

Reflecting on my involvement in the development of the “SaveNest” application, it was a truly rewarding and enriching experience. Working together with my team was a pleasure since we built a mutually supportive and idea - sharing environment. The collaborative environment not only increased the project’s enjoyment but also aided in our group’s advancement toward our objectives.

I had a continuous learning mindset throughout the "SaveNest" project, constantly keen to experiment with new technologies and approaches such as Node.js, React Native, and web development etc. Those technologies gave me invaluable insights into creating reliable front-end and back-end systems.

The obstacles faced during the project provided chances for development. Through the process of debugging complex issues and optimizing efficiency, I acquired the ability to confront challenges head-on with resilience and determination. This experience made me more adept at addressing problems and gave me confidence in my capacity to conquer obstacles. I found that seeing the results of our work with “SaveNest” were among the most satisfying parts of the experience. It was quite satisfying to watch the application come to life and enable customers to manage their financial resources more effectively. It really motivated me to use technology to create meaningful solutions and emphasized the importance of our work.

In conclusion, the “SaveNest” project has had a deeply transformational impact on me. Through collaborative software development, I have developed a growth mentality, developed my technical skills, and acquired priceless experience. Moving forward, I am excited to apply these lessons learned and continue making a positive impact through my work in future projects and in my future career path.

3.6 **Chapter Summary**

This chapter explored our project evaluation, combining quantitative and qualitative methods. Quantitative analysis provided metrics for efficiency and performance, while qualitative feedback enriched our understanding of user experiences and preferences. Additionally, individual self-evaluations fostered reflection and teamwork. By integrating diverse evaluation approaches, we gained insights for improvement and laid the groundwork for future success.

Chapter 4: Conclusion

1.1 Chapter Overview

This chapter provides a comprehensive conclusion to the project. It discusses the successful completion of objectives outlined in the previous report, identifies research limitations, proposes future enhancements, lists extra activities undertaken, and offers concluding remarks on the project's outcomes.

1.2 Achievements of Aims and Objectives

- **Project Initiation**

  During the project's initial phase, SaveNest conducted a thorough analysis of the financial market to determine the most common problems that users encountered. This required examining competitor offerings and market trends, as well as performing in-depth market research to comprehend user demands and preferences. Through collecting knowledge about the difficulties people face when handling their money, SaveNest was able to specify the project's goals and parameters. The group also developed a strong foundation for resolving these issues by putting forth creative ideas that would set SaveNest apart from other financial management programs. With careful preparation and study, SaveNest created the foundation for a development path that was successful.

- **Existing Work:**

  In order to guide its development strategy, SaveNest carried out a thorough analysis of the financial applications and solutions currently in use. This required a thorough examination of rival products, taking into account their features, functionality, user interface, and position in the market. SaveNest obtained useful information about potential areas for innovation and uniqueness by comparing itself to market competitors and discovering openings in the industry. Furthermore, SaveNest was able to pinpoint problems and areas where current solutions needed to be improved by looking through customer comments and reviews. SaveNest gained important insights from this in-depth analysis of the competition that shaped its product strategy and development roadmap.

- **Project Management:**

  The successful implementation of the SaveNest project was based on a foundation of effective project management. This involved adopting appropriate methodologies and frameworks to promote collaboration, streamline processes, and manage resources effectively. By using Agile methodology, SaveNest was able to respond to changing requirements with flexibility and through iterative development. Additionally, the team utilized tools and techniques such as Gantt charts, work breakdown structures, and risk management strategies to ensure project objectives were met on time. By implementing robust project management practices, SaveNest was able to maintain focus, transparency, and accountability throughout the development lifecycle.

- **System Requirements Specification:**

  The goal of the system requirements specification(SRS) was to precisely define the SaveNest application's functional and non-functional needs. To make sure that all relevant needs and expectations were documented, this required getting feedback from stakeholders, such as end users, domain experts, and business owners. SaveNest used a selection of methods, including surveys, use case analysis, and stakeholder interviews, to efficiently gather requirements. SaveNest reduced misunderstandings and ensured the effective delivery of the finished product by carefully documenting requirements and providing the development team and stakeholders with a clear roadmap.

- **Legal, Social, Ethical, and Professional Issues:**

  Throughout the development process, SaveNest gave careful attention to legal, social, ethical, and professional aspects. This required getting ethical approval for using datasets and making sure that applicable data protection laws, such as GDPR, were followed. In-depth risk assessments were also carried out by SaveNest to find any possible ethical and legal consequences of data processing, storage, and gathering. SaveNest aimed to establish trust and credibility among users and stakeholders by putting in place solid data management processes and abiding by industry standards and guidelines. In addition, SaveNest took into account the application's social impact, guaranteeing inclusion, accessibility, and user empowerment through well-thought-out functionality and design. Aiming to have a beneficial and long-lasting effect on society, SaveNest prioritized ethical and responsible development techniques.

- **Design:**

  The design phase of SaveNest focused on translating requirements into intuitive and user-friendly interfaces. This involved creating wireframes, mockups, and prototypes to visualize the application's layout, navigation, and functionality. SaveNest employed user-centered design principles to ensure that the application met the needs and preferences of its target audience. Additionally, the team considered factors such as usability, accessibility, and aesthetics to create a compelling user experience. By incorporating feedback from usability tests and design iterations, SaveNest iteratively refined its designs to optimize user engagement and satisfaction. Furthermore, SaveNest developed high-level architectural diagrams and system designs to guide the implementation process effectively. By investing in thoughtful design, SaveNest aimed to differentiate itself in the market and deliver an exceptional user experience.

- **Implementation:**

  The implementation phase of SaveNest involved translating design specifications into functional code. This required selecting appropriate technologies, frameworks, and programming languages to build the application. SaveNest adopted best practices such as modular design, code reusability, and version control to ensure scalability, maintainability, and extensibility of the codebase. Additionally, the development team followed coding standards and conventions to promote consistency and readability. By conducting code reviews, unit testing, and integration testing, SaveNest aimed to deliver a high-quality and reliable product. Furthermore, SaveNest prioritized security and data privacy throughout the development process, implementing encryption, access controls, and other security measures to protect user information. By focusing on robust implementation practices, SaveNest aimed to build a solid foundation for long-term success and scalability.

- **Testing:**

  In order to verify the functionality, dependability, and performance of the SaveNest program, comprehensive testing methods were needed. This required carrying out a number of testing procedures, such as system, user acceptance, integration, and unit tests. SaveNest employed both manual and automated testing techniques to identify defects, inconsistencies, and performance bottlenecks. By systematically identifying and addressing issues, SaveNest aimed to deliver a high-quality product that met user expectations and requirements. Additionally, SaveNest prioritized accessibility testing to ensure that the application was usable by individuals with diverse needs and abilities.SaveNest made a significant investment in thorough testing in order to win over users' and stakeholders' trust and guarantee the application's successful deployment.

- **Evaluation:**

  The evaluation phase of SaveNest involved assessing the effectiveness, usability, and impact of the application on users, domain experts, and industry stakeholders. This included gathering feedback through surveys, interviews, and usability tests to identify areas for improvement and refinement. SaveNest employed both qualitative and quantitative evaluation methods to measure user satisfaction, engagement, and performance. By analyzing evaluation data and insights, SaveNest gained valuable insights into user needs, preferences, and pain points. SaveNest also used evaluation results to continuously improve the features and functionalities of the program. By incorporating user feedback and addressing usability issues, SaveNest aimed to enhance the application's value proposition and ensure its long-term success in the market.

1.3 Limitations of the research

In assessing the progress and outcomes of this project, it is crucial to acknowledge certain limitations encountered during its execution, which have impacted the completeness of this project.

One significant limitation pertains to the proposed implementation of a budget plan recommendation feature utilizing artificial intelligence (AI) immediately upon user login. Although developers recognized the potential value this feature could offer in enhancing user experience and financial management capabilities, the realities of this project timeline and resource availability posed challenges to its thorough development. Given the intricacies involved in training and fine-tuning the AI model for accurate budget recommendations, developers found themselves constrained by both time and resource limitations.

As a result, it was crucial to make the strategic decision to prioritize the development of other core features within the project scope to ensure timely delivery and overall project success.

Furthermore, another area where original intentions faced limitations was in the integration of product future price prediction functionality. Initially, we aimed to incorporate a sophisticated mechanism for predicting future prices of products. However, as our research progressed and we conducted further analysis, it became apparent that the market dynamics in Sri Lanka were

characterized by a notable degree of price stability. Unlike in more volatile markets, the prices of products in Sri Lanka exhibited minimal fluctuations on a day-to-day basis, rendering the proposed price prediction feature less relevant and impactful for target users.

These limitations display the inherent challenges and complexities associated with software development projects, where the balancing act between ambitious objectives and practical constraints often necessitates difficult decisions and trade-offs. By transparently acknowledging these limitations in this report, the aim is to provide a comprehensive understanding of the contextual factors that have influenced the scope and outcomes of our research.

1.4 Future enhancements

- Improvement in Data Visualization: Improving the app's data visualization features would help users comprehend their financial patterns and trends on a more fundamental level. Users can be inspired to reinforce good financial habits by tracking their spending, income, and savings targets in real time with the use of customizable charts, graphs, and dashboards.
- Integration with Third-Party Financial Services: Users will be able to see their financial situation thoroughly thanks to seamless integration with banking, investing, and payment platforms, which will gather all of their financial data and transactions in one place. Users' experience managing their finances will be made easier and the value proposition of the app will be improved.
- Implementation of Advanced Machine Learning Algorithms: Users' spending methods will be optimized with the integration of advanced machine learning algorithms that offer proactive and customized financial information. Based on past spending patterns, predictive analytics can notify users of potential financing problems before they happen, enabling them to make well-informed decisions and effectively accomplish their financial goals.
- Multi-Currency Support: Users will be able to handle their money more effectively if multi-currency functionality is implemented, particularly if they have expenses or revenue in several currencies. This feature will improve accuracy and usefulness by giving users a more complete picture of their financial standing.

- Multi-Language Support: Adding support for multiple languages can improve usability and accessibility for a wider range of users, encouraging variety and expanding the customer base. With this functionality, users from a wide range of language backgrounds will find the app more engaging and user-friendly.
- Management of Investment Portfolios: Users will be able to monitor and control their investments in addition to their budgeting activities if the software includes a tool for managing investment portfolios. Users will have a thorough financial overview and the procedure for managing investments will be simplified as a result.
- Facilitation of Transactions and Investments through the App: Users will have greater flexibility and authority over their financial operations if the app's capabilities are expanded to allow transactions and investments through the application directly. The smooth and seamless financial management experience this feature offers will increase customer happiness and trust.

1.5 Extra work (Competitions, research papers, etc)

As a team, we enthusiastically participated in the ongoing competition, "CodeSprint 8". Our decision to join was driven by several compelling reasons:

Real-world Application: CodeSprint 8 provided us with a platform to apply our academic skills and knowledge to real-world challenges. This opportunity allowed us to leverage our group projects as a strong foundation for the competition, bridging the gap between theory and practice.

Enhanced Project Management and Teamwork: Collaborating on a startup project during CodeSprint 8 not only honed our project management skills but also fostered teamwork and communication. These qualities are essential for success in the industry and were further strengthened through our participation.

Guidance from Industry Experts: Engaging with industry experts and mentors throughout the competition provided us with invaluable guidance and mentorship. This exposure enhanced our understanding of industry trends and best practices, enriching our learning experience.

National Recognition and Cash Prizes: By showcasing our Software Development Group Project (SDGP) on a national stage, we competed for recognition and valuable cash prizes. Winning teams stood to receive financial rewards and gain recognition for their innovative solutions, motivating us to strive for excellence.

Credential Enhancement: Participation in CodeSprint 8 earned us e-certificates, serving as additional credentials to enhance our CVs. This recognition not only demonstrated our commitment to innovation and entrepreneurship but also set us apart from our peers in the competitive job market.

Our participation in CodeSprint 8 not only contributed to our personal and professional development but also reinforced the practical relevance of our academic endeavors.

Link to the CodeSprint document:

https://docs.google.com/document/d/1L8Ptd1aXz3xNOJSsHBmILUw4FTtJMVK9ND0WVhOLgQc/edit?usp=sharing

1.6 Concluding remarks

This project has successfully achieved its goals outlined in the Software Requirements Specification (SRS). Despite encountering limitations, valuable insights were gained, providing a foundation for future research. Looking forward, there is ample room for improvement and innovation through technology integration and user feedback. Additionally, participation in various events beyond the project scope has enriched the team's experience. In conclusion, this project highlights the importance of collaboration, innovation, and perseverance, setting the stage for further advancements and discoveries in the future.

References

- Henning Heitkotter, T. A. M. B. R. a. T. W., 2013. Evaluating Frameworks for Creating Mobile Web Apps, milnster: Department of Information Systems, University of Munster, M ¨ unster, Germany
- Samrgandi, N., 2021. User Interface Design & Evaluation of Mobile Applications , Makkah: International Journal of Computer Science and Network Security.

Appendix A - Implementation

Appendix A -1 Version Control / GIT (Screenshots)

All the screenshots of related components (Commits, Branches, Trello project management tool, Teamwork Breakdown Structure) in GIT are as follows. In the screenshot of GitHub Backlog it can be seen that the cards titled “Have to create Video Demonstration” and “Create Login Interface” are likely to represent tasks associated with the project.

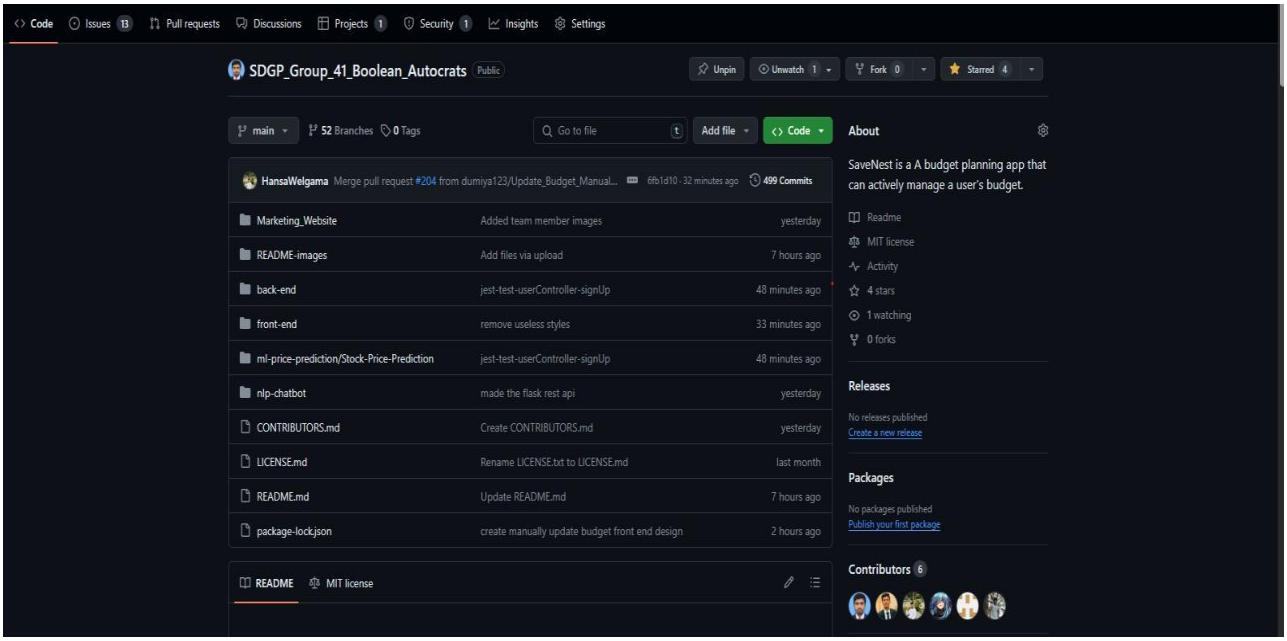


Figure 68 - Appendix A -1 Version Control / GIT 1

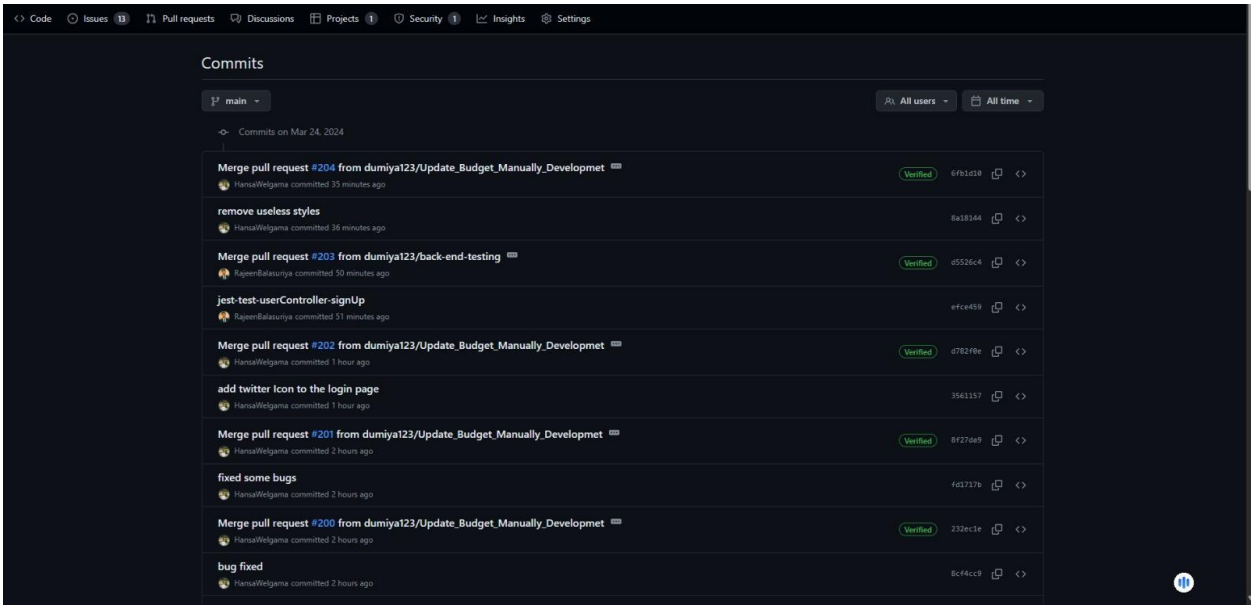


Figure 69 - Appendix A -1 Version Control / GIT 2

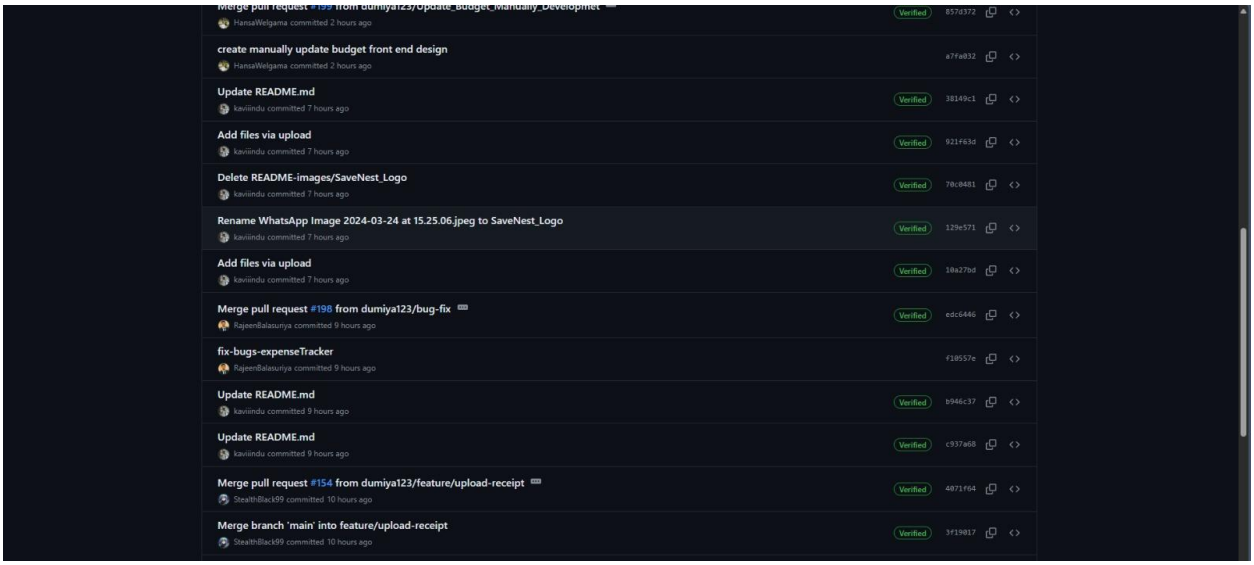


Figure 70 - Appendix A -1 Version Control / GIT 3

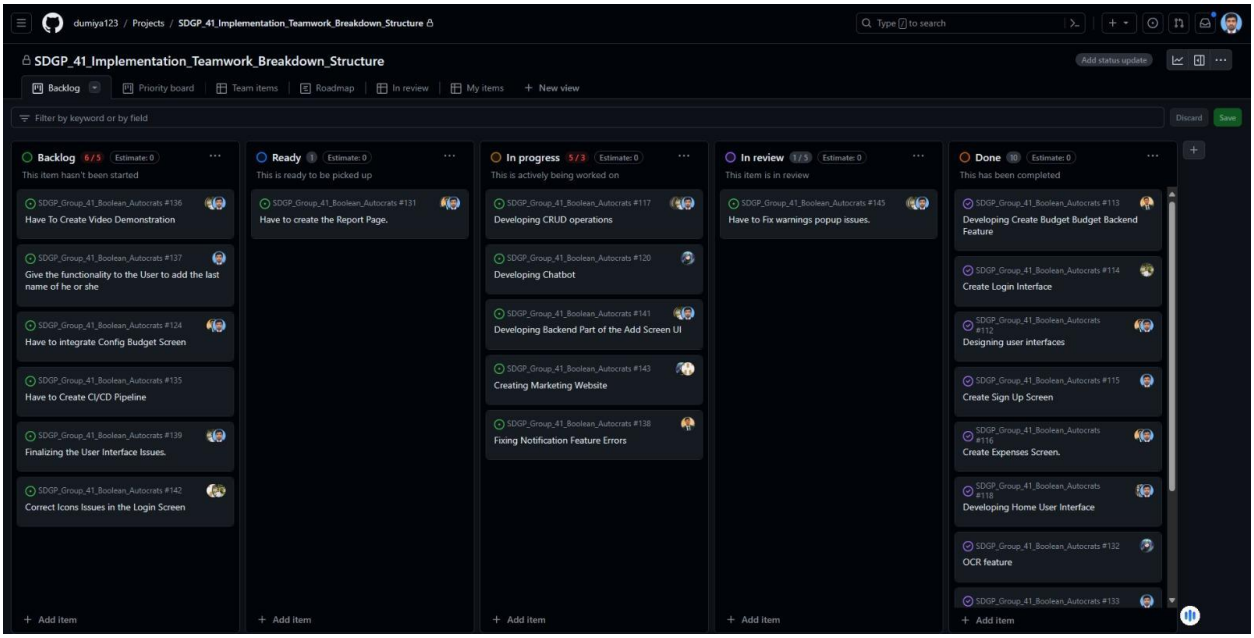


Figure 71 - Appendix A -1 Version Control / GIT 4

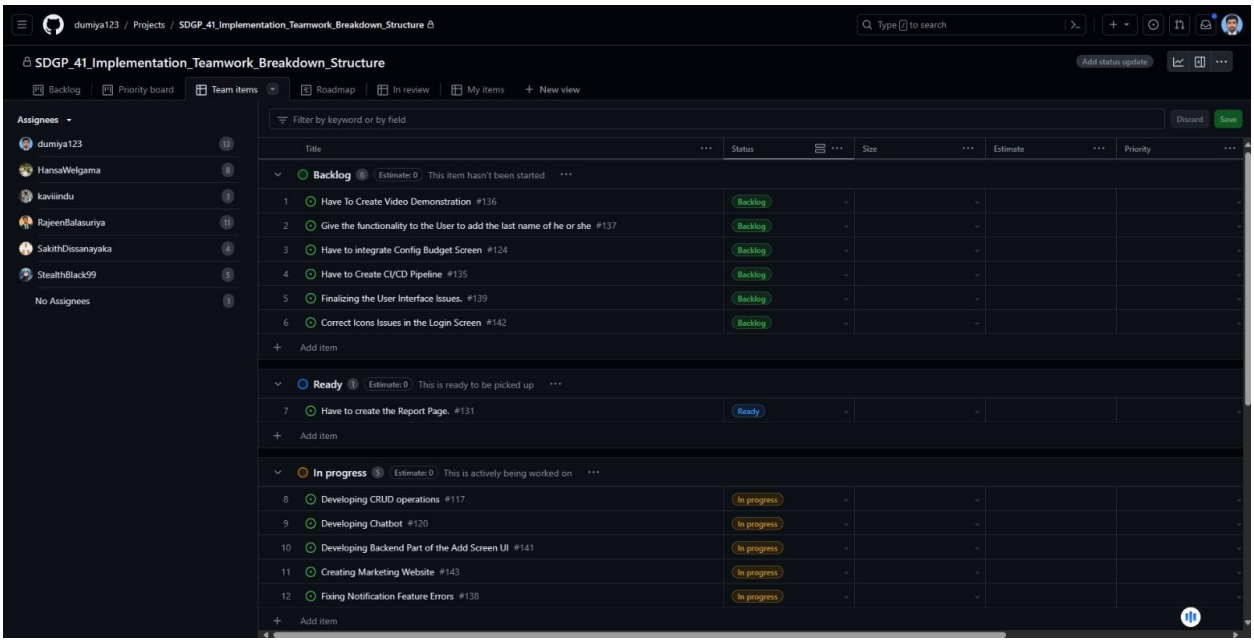


Figure 72 - Appendix A -1 Version Control / GIT 5

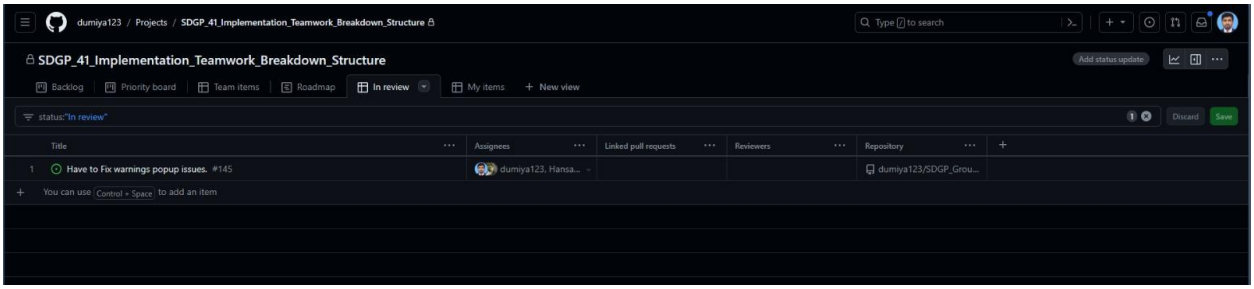


Figure 73 - Appendix A -1 Version Control / GIT 6

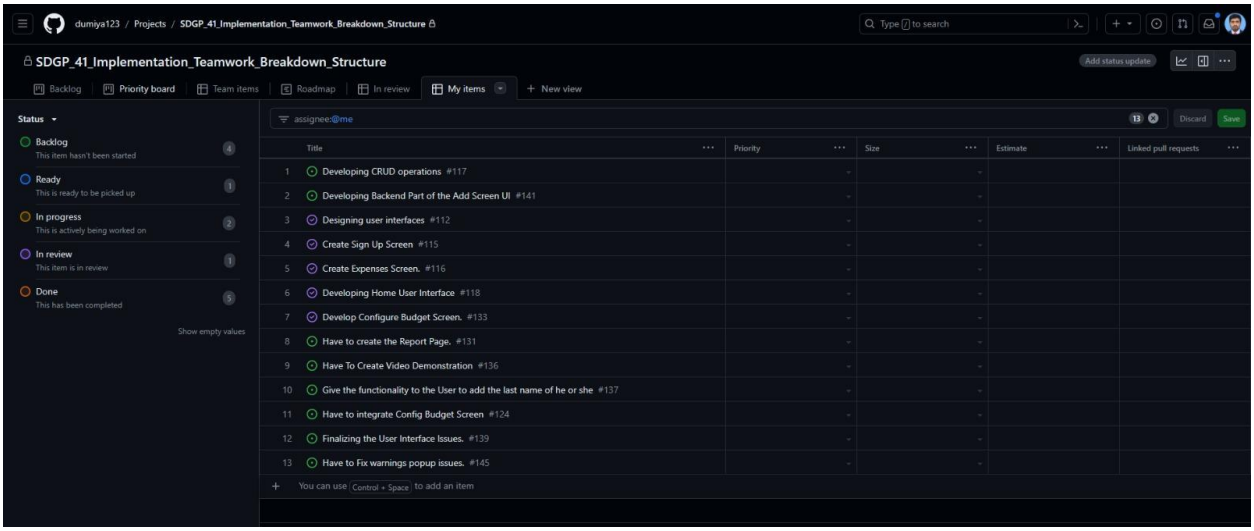


Figure 74 - Appendix A -1 Version Control / GIT 7

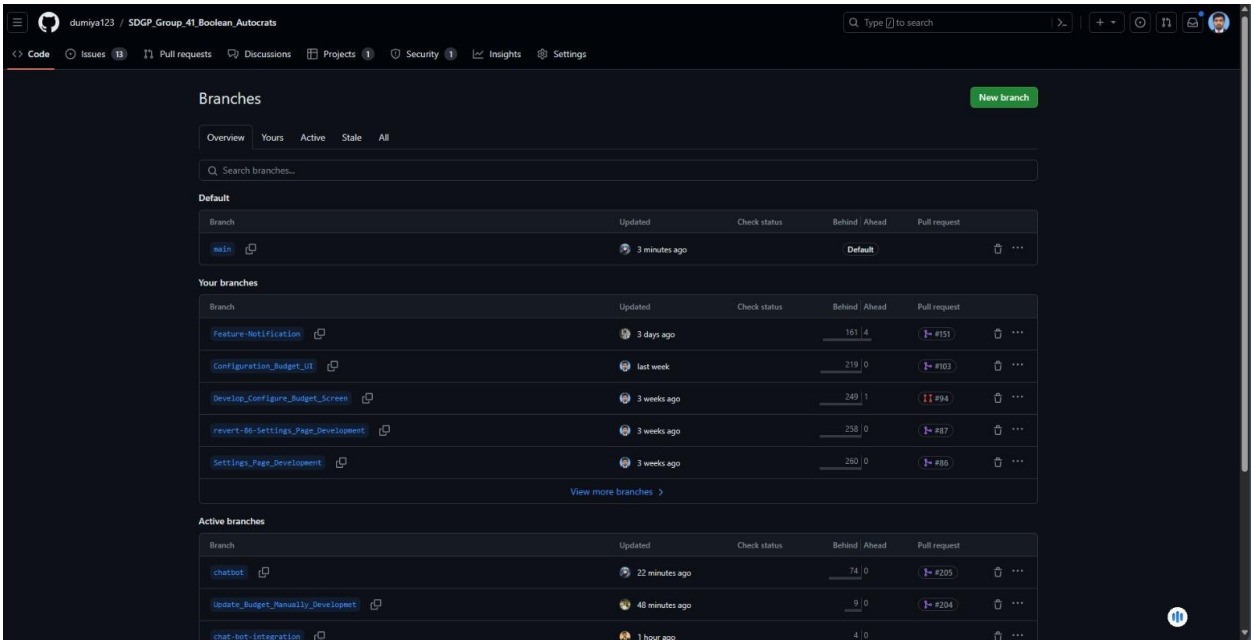


Figure 75 - Appendix A -1 Version Control / GIT 8

Figure 76 - Appendix A -1 Version Control / GIT 9

Figure 77 - Appendix A -1 Version Control / GIT 10

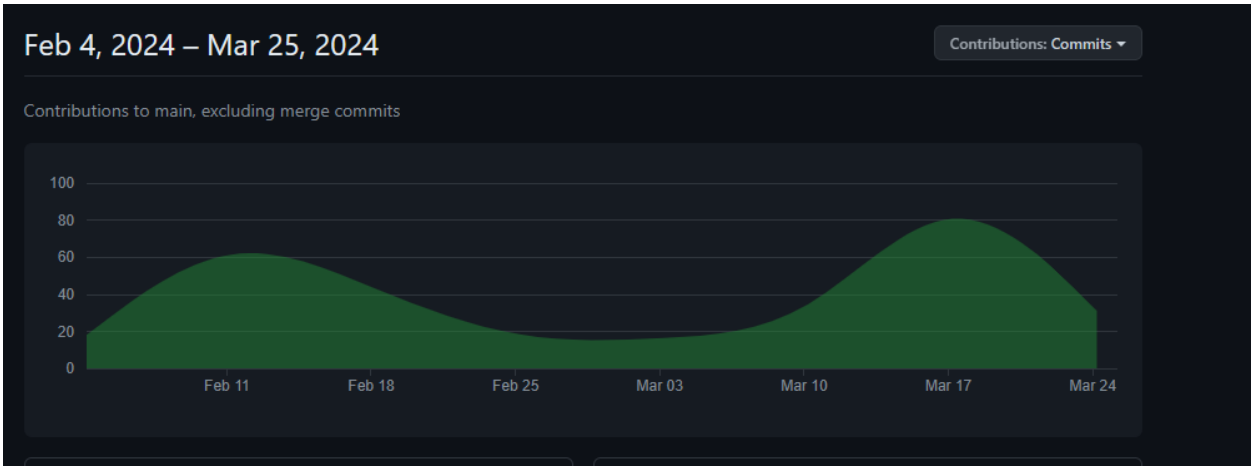


Figure 78 - Appendix A -1 Version Control / GIT 11

Appendix C - Evaluation

Appendix C -1 Evaluation Survey

Screenshots of the online evaluation survey that was sent to collect information.

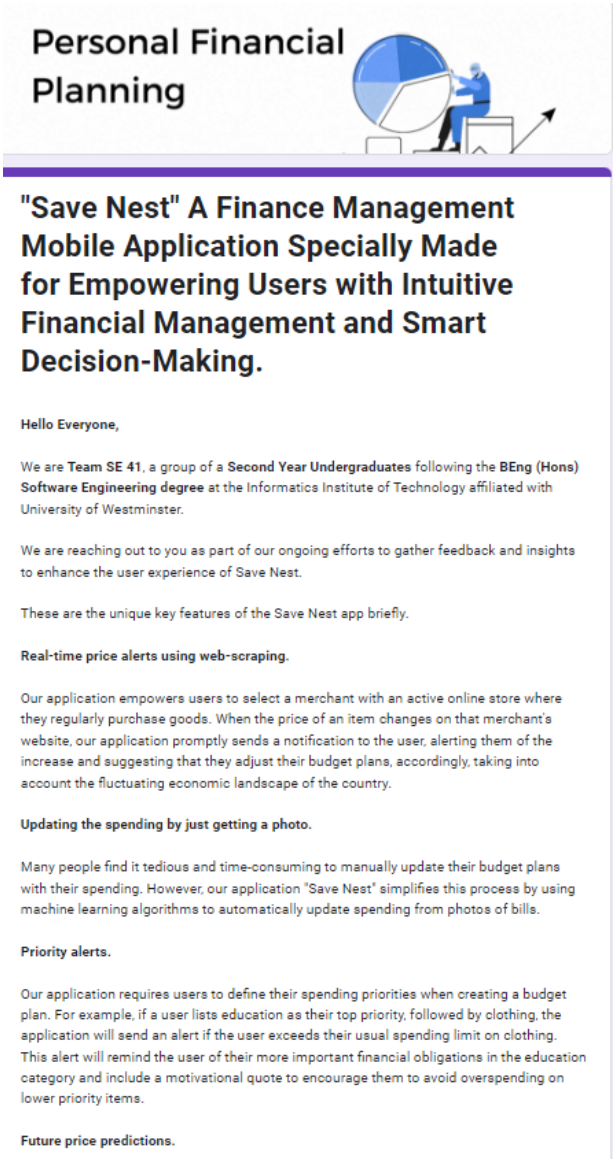


Figure 80 - Appendix C -1 Evaluation Survey 1

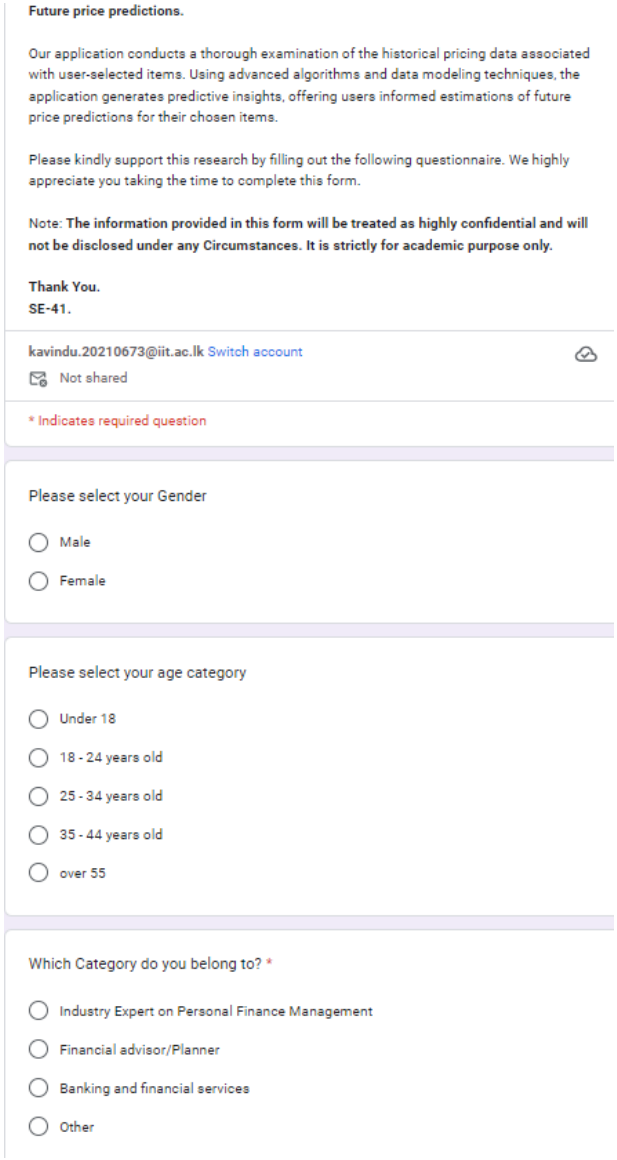


Figure 79 - Appendix C -1 Evaluation Survey 2

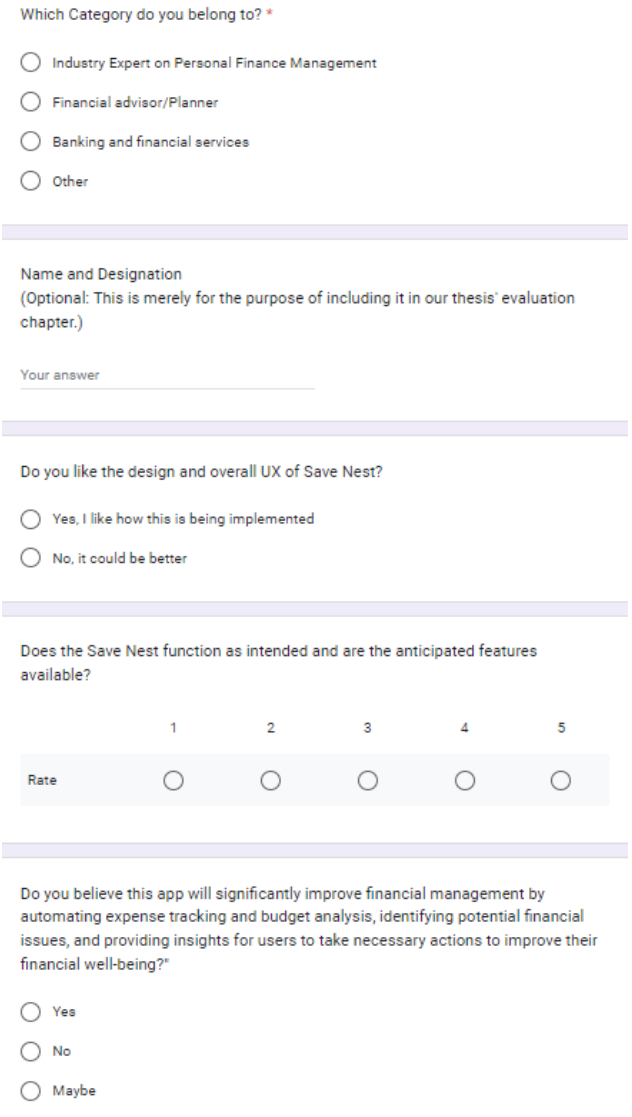


Figure 81 - Appendix C -1 Evaluation Survey 3

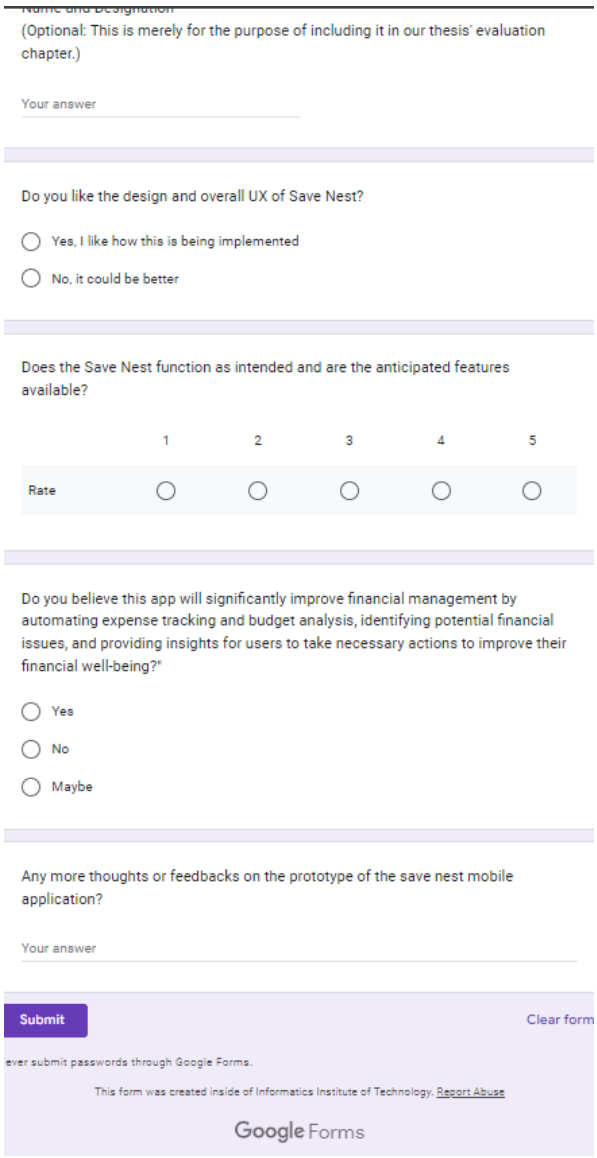
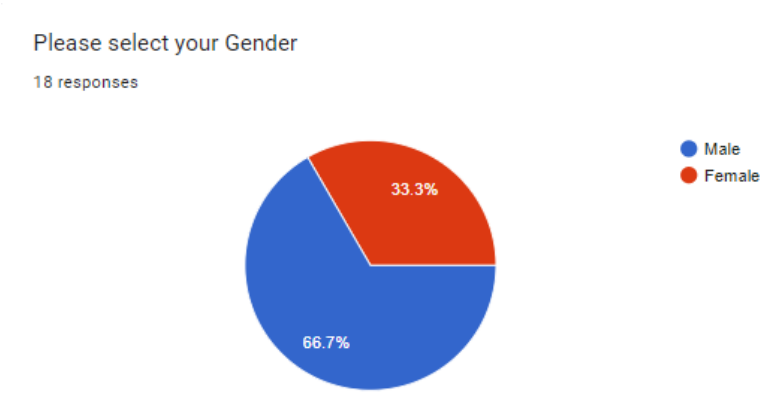
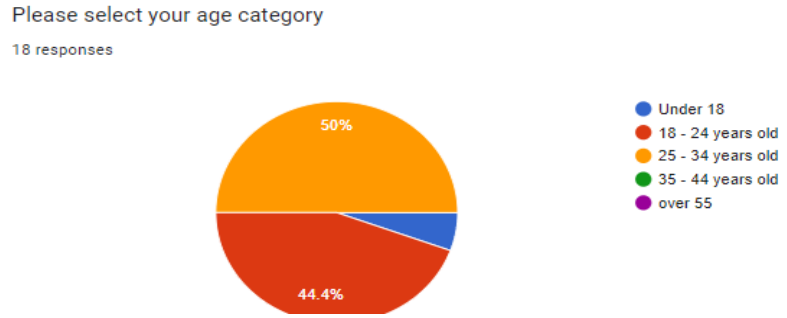
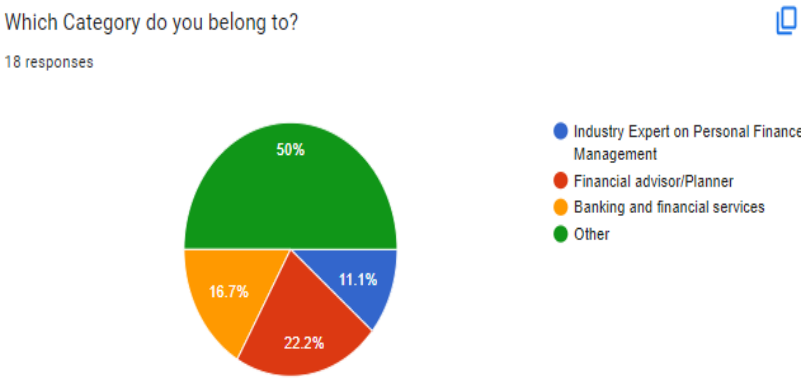
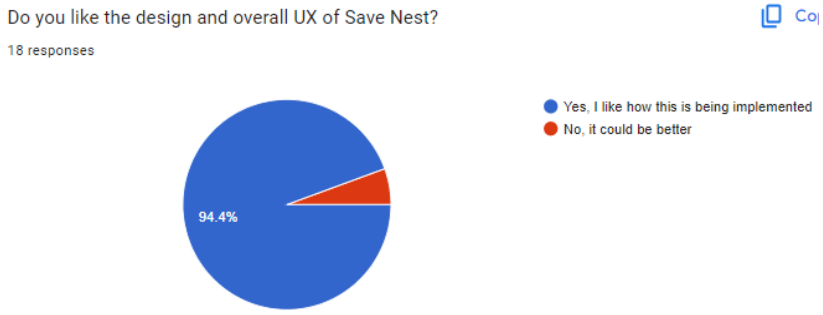
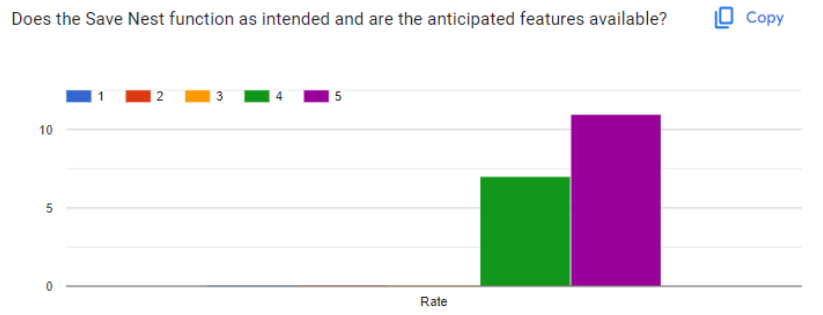


Table 4 - Figure 81 - Appendix C -1 Evaluation Survey 4

Appendix C -2 Analysis of Evaluation survey results.

Questionnaire Result	Analysis
	The majority of respondents, constituting 66.7% identify as male.
	The majority of respondents, comprising 50% fall within the 25-34 age category suggesting that young adults are really involved in this.
	• Financial advisor/Planner: 22.2% • Banking and financial services: 16.7% • Industry Expert on Personal Finance Management:

	11.1% The largest proportion of respondents representing 22.2% identify as financial advisors/planners
	The vast majority of respondents, comprising 94.4% out of the sample expressing their satisfaction with the design and overall user experience(UX) of the SaveNest.
	Based on the responses, the majority of respondents rated SaveNest highly for functioning as intended and providing anticipated features

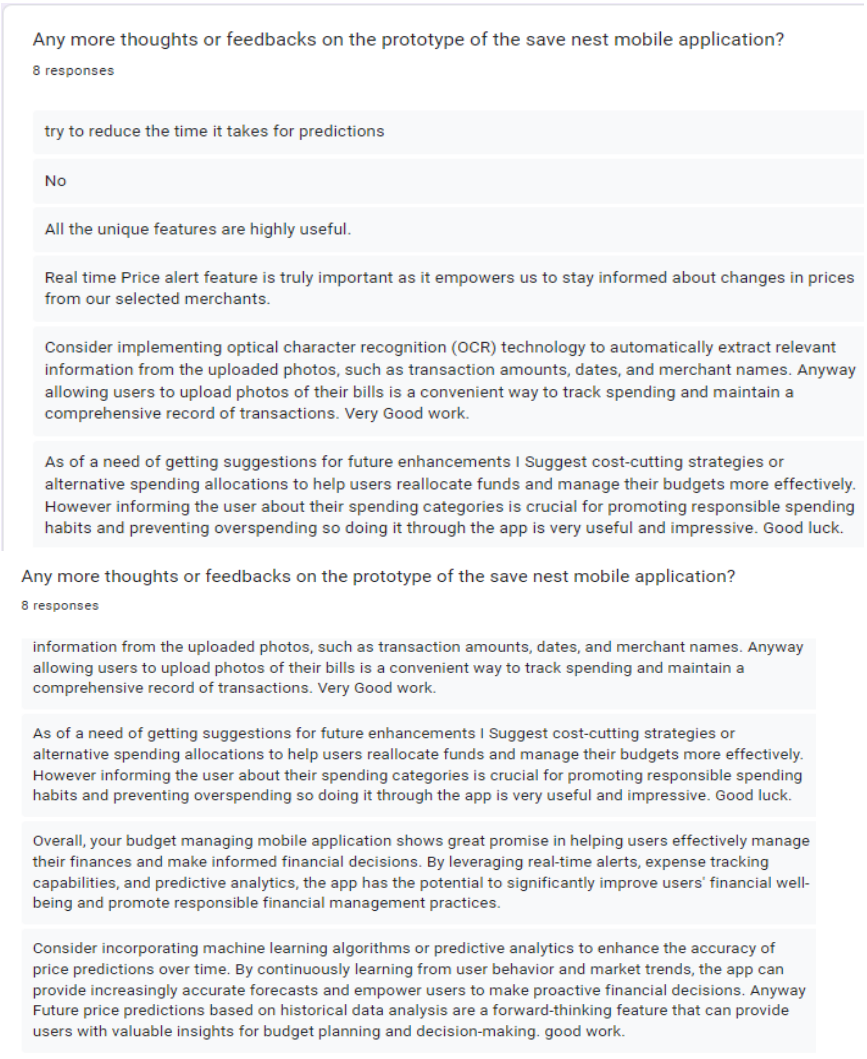
<table>
<tr>
<td> 
</td>
<td>The overwhelming majority of respondents, representing 94.4% out of 18 responses, expressed confidence that the app will significantly improve financial management</td>
</tr>
<tr>
<td> 
</td>
<td>Some of the recommendations made by certain evaluators.</td>
</tr>
</table>

Table 5 - Appendix C -2 Analysis of Evaluation survey results.