Mini Project Report

on

Expense Prediction

Submitted by
Group id - 04

Project Members

KRISHNA HITNALIKAR (1032221979)

RONAK PATIDAR(1032221964)

HARSH KATIYAR(1032221852)

AMAN KANTHALIA(1032221904)

Under the Guidance of

Prof. Pramod Mali

School of Computer Engineering and Technology MIT
World Peace University, Kothrud,
Pune 411 038, Maharashtra - India

2024-2025

Abstract

The design and deployment of the long short-term memory neural networks are explored, which fall in the category of recurrent neural networks (RNNs) for general time-series forecasting. The system will be used to help people manage their budgets by predicting daily and weekly expenses from various categories such as food, transportation, entertainment, and other items that users might need.

In this day and age, with speed dominating society and consumption characterizing an era, it is tough for human beings to exercise financial discipline over their proper spending. Many people spend their savings too early, mostly because of unintended spending or lack of knowledge about where they spend their money.

The system essentially looks at historical financial data and makes future predictions about spending by individuals in real time. It provides users with day-to-day and weekly predictions of money to be spent and personalizes insights toward future expenses so that an individual can adjust his or her financial behavior proactively.

This approach of predicting will be placing individuals better in avoiding over-spending hence remain within their monthly budgeting and, therefore eventually maintain longterm financial stability.

This is even more applicable with the use of an LSTM because it will reflect not only the short-term fluctuations but also the long-term trends in spending behavior. It is, therefore, well suited to such a prediction task. Its recurrent structure enables it to keep track of relevant information from previous days' expenditure and generate predictions that take into account not only the patterns of recent days but also the dependency that exists between different types of spending.

It details all the processes involved in the lifecycle of an expense prediction system, starting with problem-domain analysis, where objectives were formulated. It also provides a comprehensive literature survey of the existing techniques in financial forecasting and enumerates the weaknesses of traditional methods, thereby pointing out the advantage of using machine learning models, especially LSTM networks. Such a system discusses technical requirements and design choices related to a model of data preprocessing, model architecture, training methodology, and evaluation metrics. Besides that, the system's performance will be exhaustively experimented with -- a sequence of experiments made on the model which compares the prediction made by it to a series of actual user expenditure data.

This system shall bridge this gap between these conventional budgeting techniques and modern technological advances, helping the user provide an

intelligent tool through which he can manage his finances more efficiently. The prediction nature of the system helps the users approach their budgeting in a more proactive manner, thus providing a sense of financial discipline that offers great advantages in decisions made about daily expenditures.

With the advancement of machine learning, this path can be considered one of the possible further developments in using advanced neural networks like LSTM for personal finance management. This report illustrates some practical applications of such systems and demonstrates how artificial intelligence can apply to everyday financial challenges faced by people around the world.

LIST OF FIGURES-

LIST OF TABLES-

Contents

Chapter 1:
Introduction

1.1 Challenges in Personal Financial Management

With the hurried lifestyle and increased consumption, proper financial management is a major challenge to one who wishes to hold everything together financially without drowning in debt. People thus struggle, while living expenses continually rise and personal spending grows at unpredictable intervals.

Personal budgeting is, after all a core element of sound financial management. Here, most people have to plan and monitor their spending habits very carefully so as not to go above their limits. However, personal budgeting can sometimes be such a challenge to many people once they are faced with irregular sources of income, unexpected expenses, and the numerous financial decisions individuals have to make daily. Most people end up in financial difficulties by the end of each month due to not planning on how they should manage their incomes and expenses.

1.2 Budgetary Limits and Financial Control

In addition to predicting expenses, the expense prediction system allows users to set individual budgetary limits for various spending categories, such as groceries, transportation, entertainment, and utilities. By setting these limits, users can gain control over their spending in specific areas, ensuring that they do not exceed their planned budget for any category. This feature encourages responsible financial behavior by providing clear boundaries for spending, making it easier for individuals to avoid overspending.

The system will alert users when they approach or exceed their set limits, allowing them to adjust their behavior in real-time. This not only prevents financial imbalances but also helps users identify areas where they may need to cut back. With a clear overview of how their money is being spent across

different categories, users are empowered to make informed decisions and prioritize their expenses based on their financial goals.

By incorporating budgetary limits, the system reinforces financial discipline and promotes a proactive approach to managing money, helping users maintain long-term financial stability.

1.2 Expense Prediction System Using LSTM Networks

A good number of people lack control of their spending habits in food, transport, entertainment, and other utilities, which often results in imbalances in the budget and creates pressure on finances.

Impulse buying, unaccounted for expenditures, as well as a lack of transparency as to the expenditures that will be incurred in the future all make it more complicated to budget, and with ease one will find them over-spending without realizing the implications at the long term. In countering this major challenge, this project will introduce an expense prediction system aimed to enable users to gain insight into their spending patterns in the near future.

By exploiting the LSTM neural networks, the system can predict the daily and weekly expenses based on the historical data of the spending of users, so that the users can have a good sense of spending and thus avoid excessive expenditure.

The actual core technology associated with the prediction system of expenses is the LSTM neural network, which is a special kind of recurrent neural network that has been successfully applied for time-series predictions. The use of LSTM networks for tasks that imply sequential dependency in data is apt since the networks retain both short-term and longterm dependencies in the data. This project trains an LSTM model on a user's past spending data, learning how to recognize patterns in the way a person allocates his or her funds across different categories over time. This cost prediction system is very effective for the fact that it can classify expenses and provide a detailed forecast regarding the amount of money an individual would spend on specific groups, such as

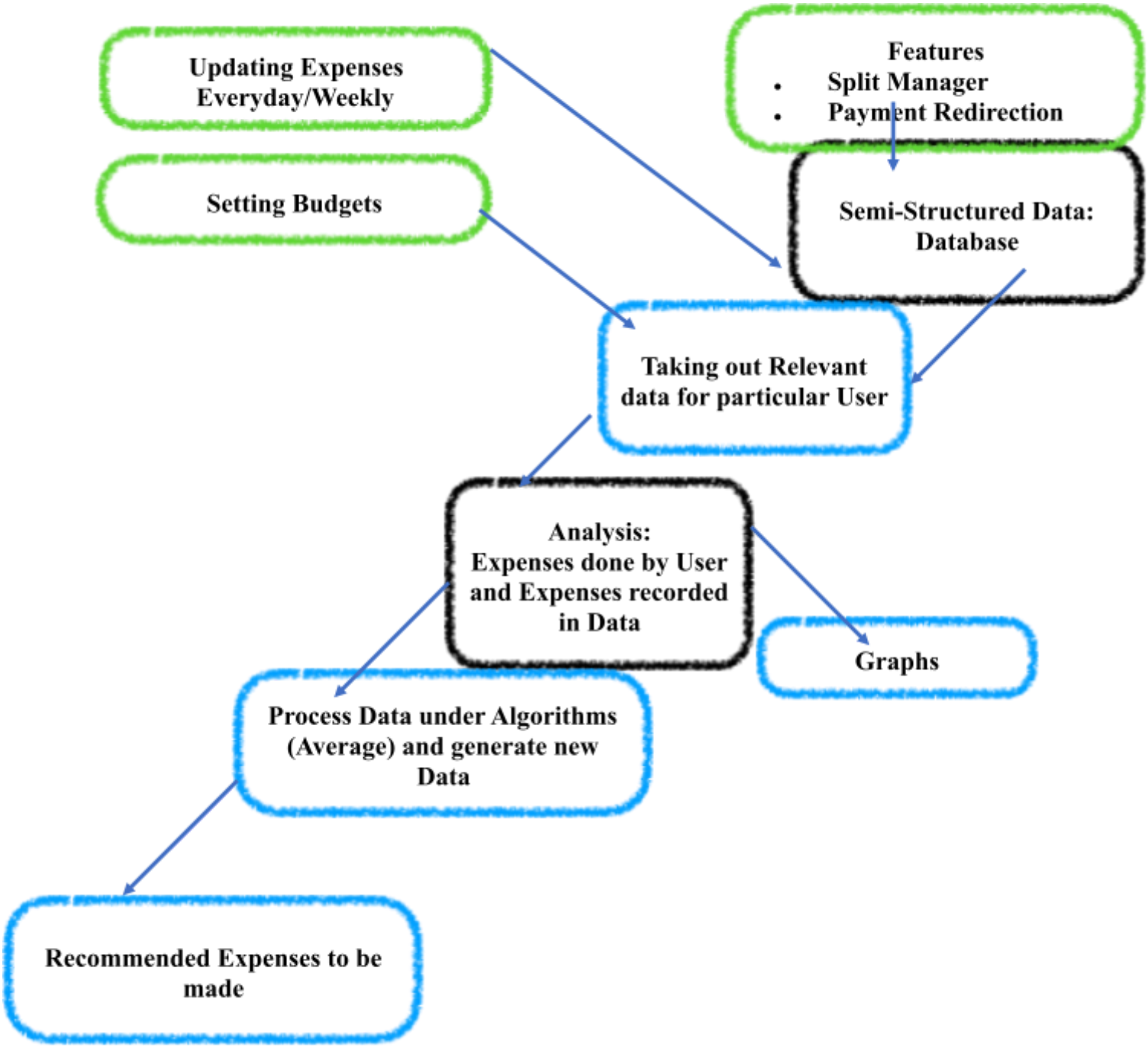
grocery expenditures, transportation, entertainment, and many others, so that they can identify where their money should be used correctly.

Apart from this, the system has a provision where users can set individual budgetary limits for every category so that there will be no overspending in any particular area. Offering users a clear view of their financial future, it empowers them to control their spending habits and make more informed financial decisions.

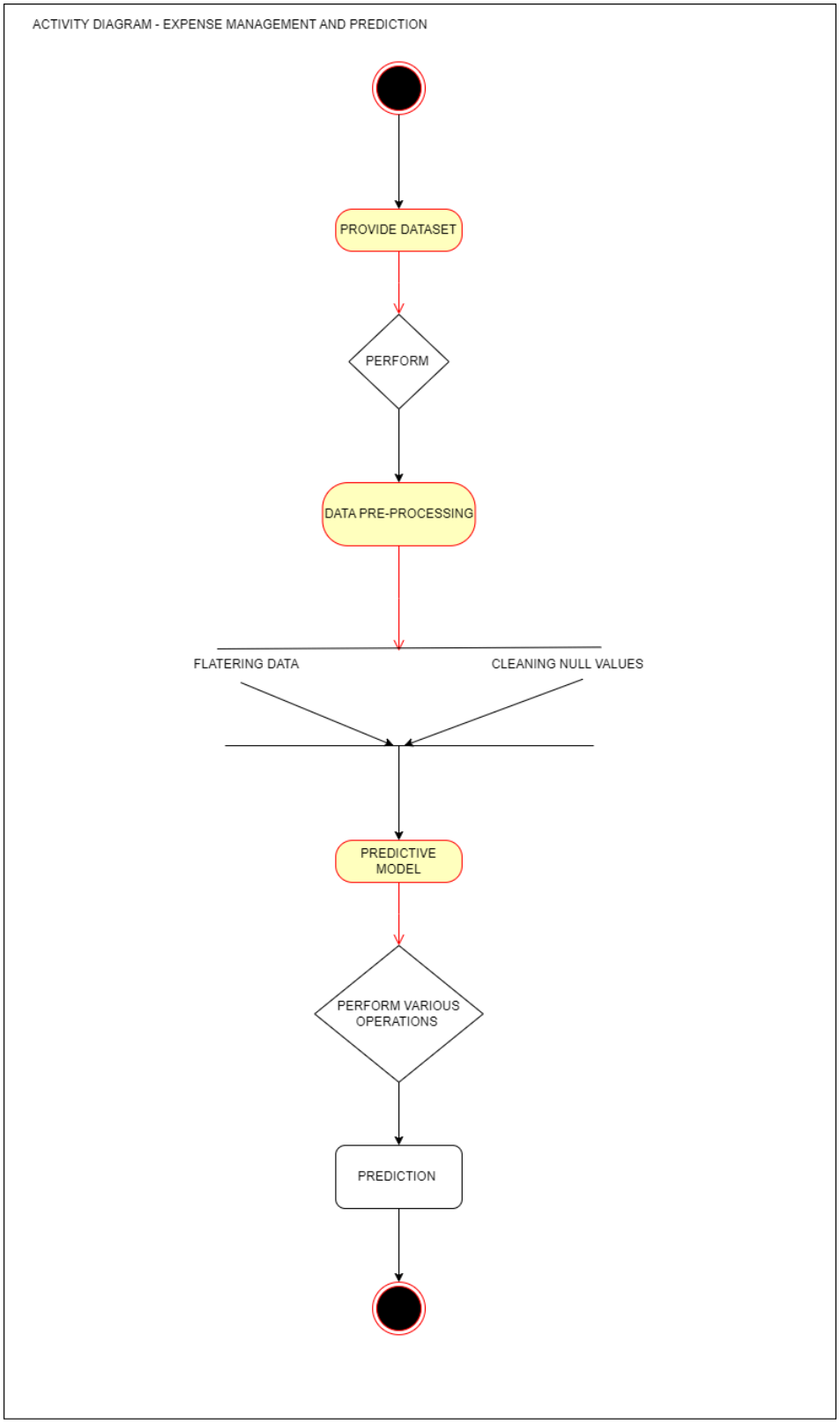
Chapter 2:

BLOCK DIAGRAMS

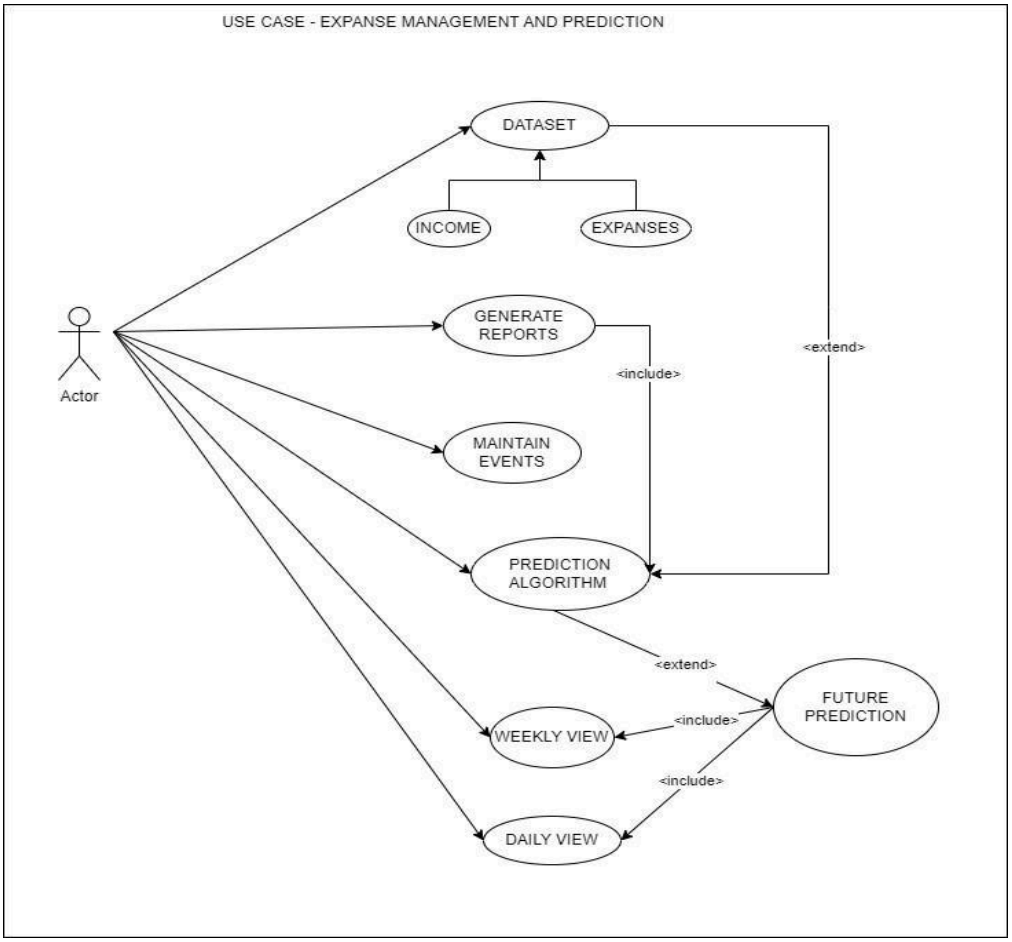
1)Architecture:



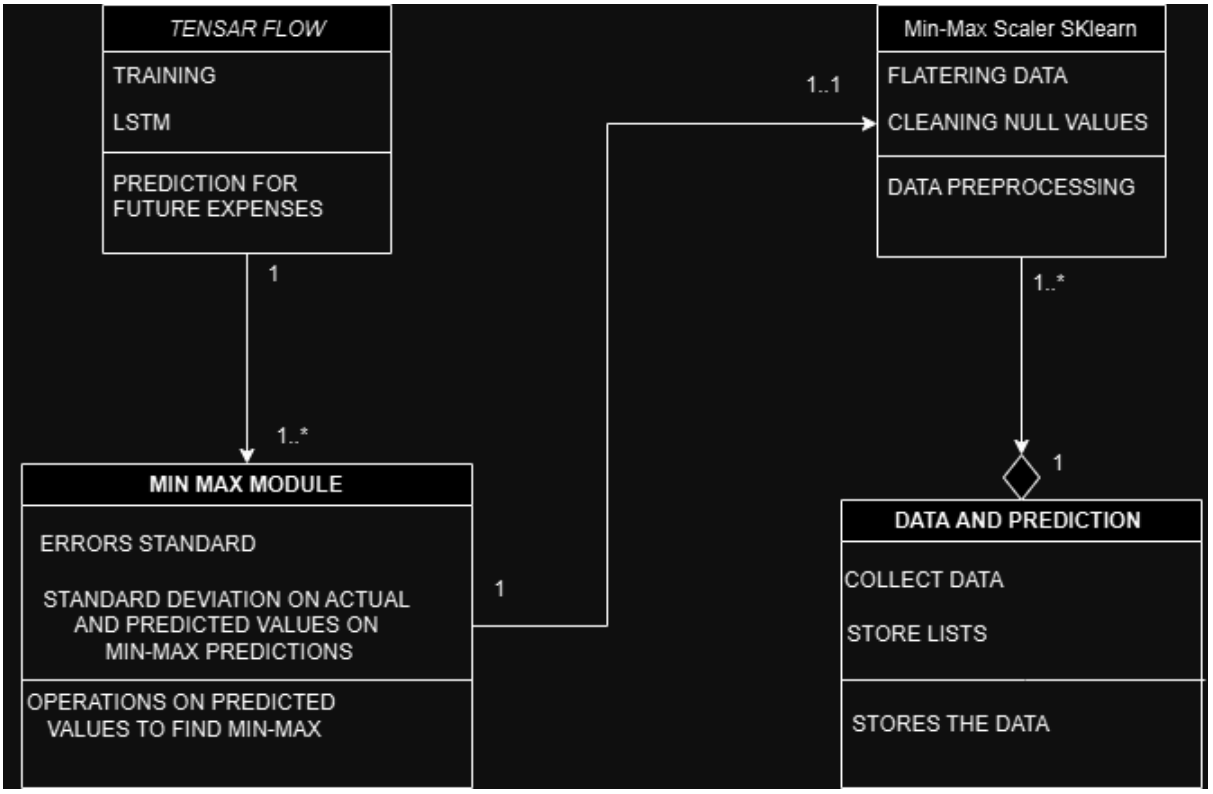
2)ACTIVITY DIAGRAM -



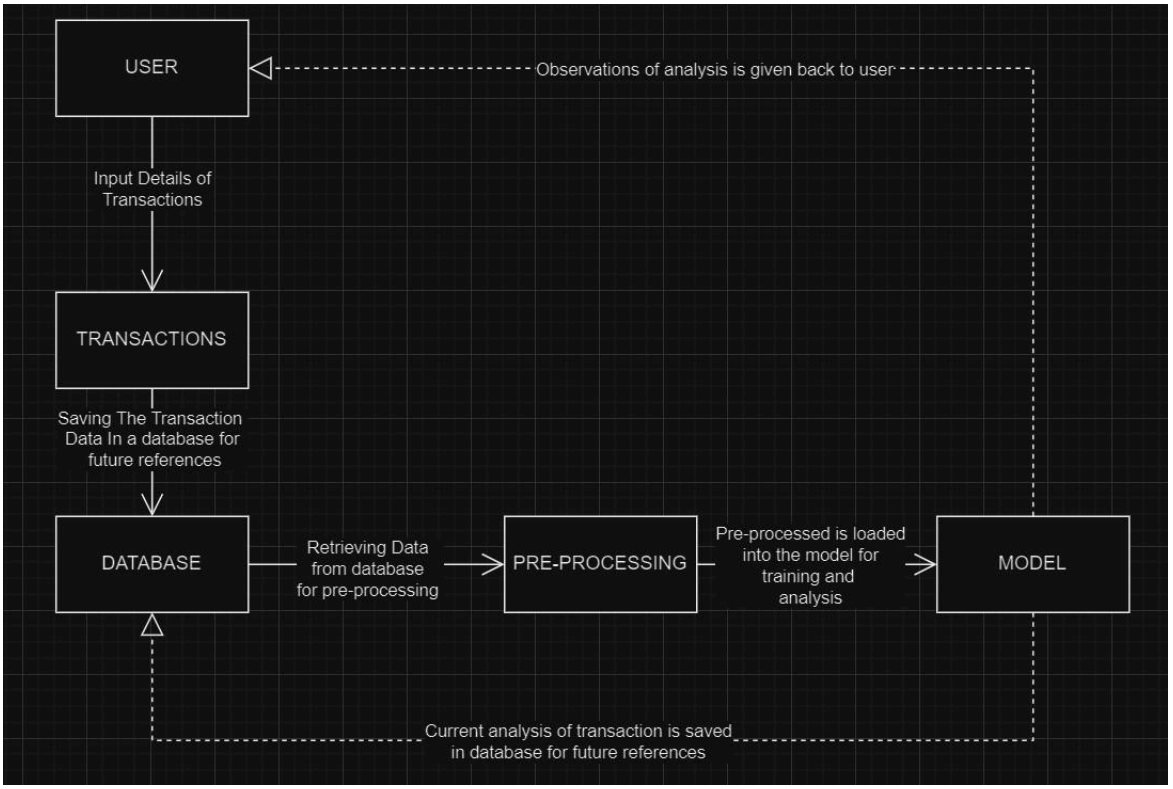
3)USE-CASE DIAGRAM -



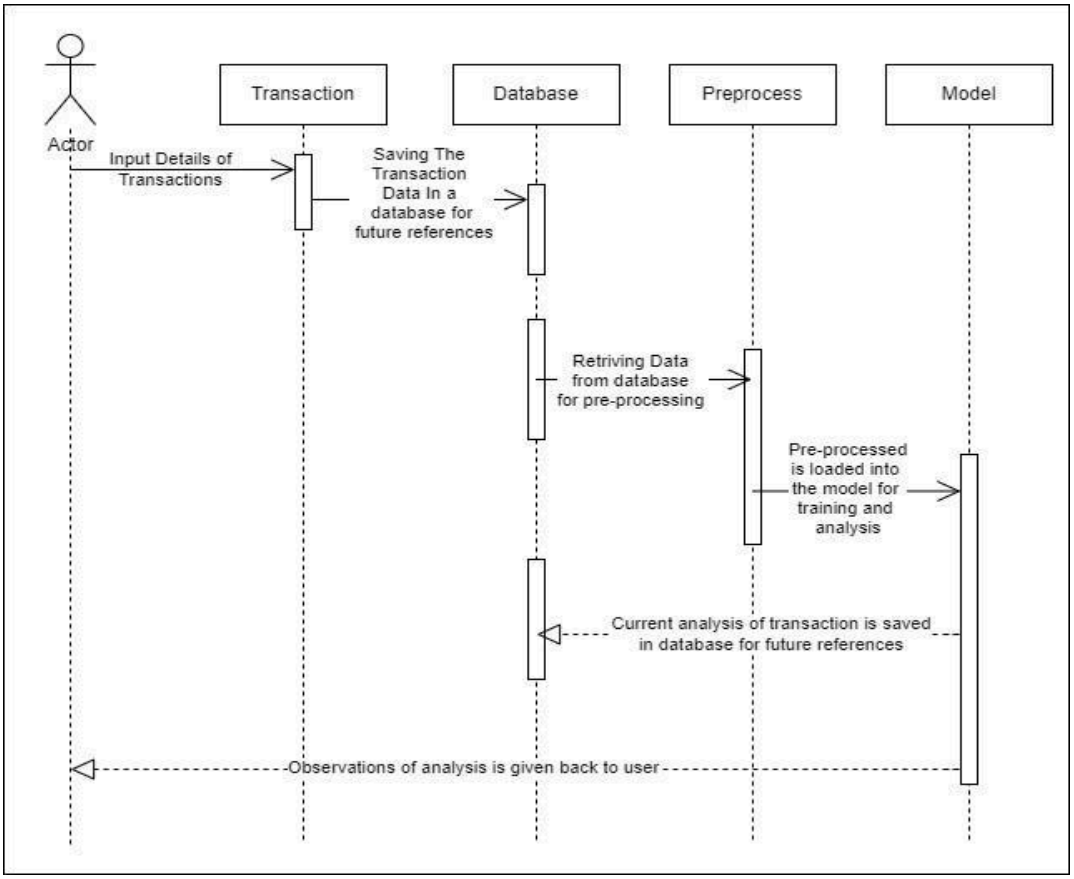
4)CLASS DIAGRAM -



5)COLLABORATION DIAGRAM -



6)SEQUENCE DIAGRAM -



Chapter 3:
Literature Survey

Table 1: Literature Survey

S.No	Title	Year of Publication	Algorithm/Methodology	Findings	Research Gaps
1	**SNAP**	2021	Data Analysis	Effective in tracking expenses	Lacks predictive capabilities
2	**Jupiter**	2020	Statistical Methods	Efficient for expense management	Does not integrate machine learning or predictive analytics
3	**Swindon Simplify**	2019	Basic Expense Tracking	Simplifies expense tracking for users	No inclusion of predictive analytics
4	**Finery**	2022	Data Visualization	Strong visual representation of expenses	No predictive recommendations provided
5	**Rather Go Commands Player**	2023	Command-based tracking	Useful for tracking daily expenses	Not integrated with advanced machine learning methods
6	**Expense Tracker**	2021	Basic Expense Tracking	Effective in categorizing expenses	Lacks predictive analysis features
7	**Expenditure Management System**	2022	Expense Management	Manages daily and monthly expenses	Does not use advanced predictive models

8	**Expense Tracker**	2021	Expense Management	Tracks daily expenses	No predictive analytics or forecasting tools

Literature Survey-

The domain of personal finance management has undergone substantial evolution, primarily driven by advancements in data analytics and machine learning (ML). Early systems were primarily focused on recording and categorizing expenses, offering users insights into their historical spending. However, such systems lacked the capability to predict future expenditures. With the development of machine learning models, particularly Long Short-Term Memory (LSTM) networks, there is now potential to forecast personal expenses based on historical spending data. This innovation opens up new possibilities for financial planning, offering users a more proactive approach to managing their budgets.

TAKINGS FROM RESEARCH PAPERS-

[1]. SNAP (2021) was highly effective in tracking real-time expenses but identified a gap in its predictive capabilities, which would enhance its ability to provide users with foresight into future spending behavior.

[2]. Jupiter (2020), utilizing statistical methods, showed the limitations of relying on such models for predicting future expenses, especially when confronted with non-linear or irregular financial patterns.

[3]**.** Swindon Simplify (2019) underscored the importance of ease of use in expense tracking but highlighted a gap in the integration of machine learning techniques for future financial forecasting.

[4]. Finery (2022) demonstrated the value of strong data visualization in helping users understand their historical expenses. However, the absence of predictive analytics in the system left users without forward-looking insights.

[5]. Rather Go Commands Player (2023), with its command-based interface, showcased efficiency in daily tracking but lacked the sophistication required for predicting future financial behavior through machine learning models.

[6]. Expense Tracker: A Smart Approach to Track Daily Expenses (2021) by Nutheti et al. demonstrated efficiency in daily tracking but did not explore predictive capabilities, leaving users reliant on retrospective tracking rather than proactive financial planning.

[7]. Expenditure Management System (2022) by Gomathy presented a simplified approach to managing finances but did not include advanced predictive modeling, which could have enhanced its utility for users planning future expenditures.

[8]. Expense Tracker (2021) by Garg et al. focused on categorizing expenses without providing the users with predictive insights, thus limiting its application in long-term financial planning.

Chapter 4:
Problem Requirements

4.1 Problem Statement-

Personal financial management requires accurate tracking and forecasting of expenses to ensure that users stay within their budgetary limits. The **Expense Prediction System** proposed in this research is designed to predict daily and weekly expenditures based on historical transaction data. By employing **LSTM neural networks**, the system will forecast spending in key categories such as food, transport, and entertainment. Furthermore, it allows users to set budgetary limits and receive min-max predictions, providing a confidence interval that helps them prepare for fluctuations in spending. The system's goal is to enable users to make informed financial decisions, offering transparency and ease in managing their finances.

4.2 Project Scope-

The scope of this research extends to the complete **design**, **development**, and **deployment** of a sophisticated **Financial Expense Prediction Tool** based on **LSTM neural networks**. The tool's primary function is to predict future expenditures by analyzing users' historical financial data. By focusing on key expense categories—**food**, **transport**, and **entertainment**—the system assists users in monitoring and understanding their spending behavior. To prevent overspending, users can set **budgetary limits** for each category. Additionally, the system offers **min-max predictions**, which provide a confidence interval, accounting for uncertainties in future spending patterns. This forecasting feature equips users with foresight into possible fluctuations in their financial activity.
The system integrates **real-time financial data** collected from users' bank transactions in **CSV format**, ensuring accuracy and currency in financial predictions. The core programming language for this project is **Python**, supported by a robust set of libraries:

- **TensorFlow** for building the LSTM model,
- **pandas** for efficient data preprocessing,
- **Matplotlib** and **Seaborn** for graphical data visualization.

4.3 Project Resources-

Data Requirements:

- **Historical Transaction Data:** A dataset containing transaction records with the following information:
    - Date
    - Amount
    - Category (e.g., food, transportation, entertainment)
    - Optional: Other relevant features (e.g., location, merchant)
- **Data Format:** CSV or other suitable format.
- **Data Quality:** Ensure the data is clean, consistent, and free from errors.

Technical Requirements:

- **Programming Language:** Python
- **Libraries:**
    - TensorFlow or Keras for building and training the LSTM model
    - pandas for data manipulation and analysis
    - NumPy for numerical operations
    - Matplotlib or Seaborn for data visualization
- **Hardware:** A computer with sufficient processing power and memory to handle the LSTM model.

Model Development Requirements:

- **LSTM Architecture:** Design an appropriate LSTM architecture, considering factors such as the number of LSTM units, layers, and activation functions.
- **Hyperparameter Tuning:** Experiment with different hyperparameters to optimize the model's performance, such as learning rate, batch size, and dropout rate.
- **Training and Evaluation:**
    - Split the dataset into training and testing sets.
    - Train the LSTM model on the training set using appropriate loss function and optimization algorithm.

    - Evaluate the model's performance on the testing set using metrics like MSE, MAE, and RMSE.

Evaluation Metrics:

- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual values.
- **Mean Absolute Error (MAE):** Measures the average absolute difference between predicted and actual values.
- **Root Mean Squared Error (RMSE):** The square root of the MSE, providing a more interpretable measure of error.

Additional Considerations:

- **Feature Engineering:** Explore creating additional features from the raw data to improve model performance, such as time-based features (e.g., day of week, month) or categorical features (e.g., income level).
- **Data Normalization:** Normalize the data to a common scale to ensure fair comparison between different features.
- **Overfitting:** Be cautious of overfitting, where the model performs well on the training data but poorly on unseen data. Techniques like regularization can help mitigate overfitting.
- **Interpretability:** Consider techniques to interpret the LSTM model's predictions, such as feature importance analysis or visualization of learned representations.
- **Deployment:** Plan for deployment of the trained model in a production environment, ensuring scalability and accessibility for users.

Chapter 5:
Implementation

This kind of expense forecasting system was set using a structured approach in terms of preprocessing the data, developing the model, training it, and evaluating it. The historical spending data were normalized to the same scale so as not to bias the model towards more significant numerical values. Also, the data were categorized into different types of expenditure: food, transportation, and entertainment. In this way, the model will learn specific patterns in spending for every category.

The LSTM network architecture defines an input layer, which feeds the preprocessed data to it. It then consists of multiple hidden LSTM layers to capture long-term dependencies and consequently generate temporal patterns from the data. Then finally, the output layer produces a prediction for next-day expenditure.

LSTM Model Back propagation through time and gradient descent are used for optimization of the model in training. It minimizes the error between the predicted expenditures and actual ones. Now, after training this model on the historical dataset, it is evaluated with a separate testing set in terms of accuracy.

The forecasted versus actual expenses visualized enable the user to compare his or her real spending behavior against the forecast values. The system should predict daily and weekly expenses so that users can plan for those periods and avoid overshooting their monthly budgets.

5.1) Code-

· Model Creation and Training Works as Follows:

1. Importing Libraries

```
•  import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

pandas: For data manipulation and analysis.

numpy: For numerical operations.

matplotlib: For plotting the results.

MinMaxScaler: To normalize the data between a defined range.

Sequential, LSTM, Dense: For creating and training a neural network using LSTM (Long Short-Term Memory).

2. Loading and Preprocessing Data

```
# Load data
file_path = 'total_expenses.csv'
df = pd.read_csv(file_path)

# Convert the 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'])
```

```
# Filter data for the last 6 months
six_months_ago = df['Date'].max() - pd.DateOffset(months=6)
df_filtered = df[df['Date'] >= six_months_ago]

# Sort by date if necessary
df_filtered = df_filtered.sort_values(by='Date')

# Extract the 'Amount' column or other columns as needed
data = df_filtered[['Amount']].values
```

Reads the expense data from a CSV file.
Converts the Date column to datetime format.
Filters data from the last 6 months.
Extracts the 'Amount' column for modelling.

3. Normalizing the Data

```
# Normalize the data

scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data)
```

Scaling: Since machine learning models perform better when data is on a similar scale, the expense data is scaled to a range of 0 to 1 using MinMaxScaler. This prevents larger values from dominating the learning process and helps the model converge faster.

4. Creating Sequences for LSTM

```
def create_sequences(data, seq_length):
X = []
for i in range(len(data) - seq_length):
X.append(data[i:i + seq_length])
return np.array(X)

seq_length = 7  # History sequence
X = create_sequences(scaled_data, seq_length)
```

Function for Sequences: The LSTM model requires input data in the form of sequences. The function create_sequences() prepares sliding window sequences of a specified length (7 in this case). This means the model will look at the last 7 days of expenses to predict the next day's value.
Creating X: A sequence X is created from the normalized data using the history of the last 7 days (seq_length = 7).

5. Training and Testing Data Split

```
train_size = int(len(X) * 0.8)
```

Data Split: The data is divided into 80% for training the model and 20% for testing its performance. The first 80% of the sequences is used to train the model, and the remaining 20% is used to evaluate how well the model generalizes.

6. Building the LSTM Model

```
model = Sequential([
```

```
LSTM(50, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2])),
Dense(1)
])
model.compile(optimizer='adam', loss='mse')
```

Defining the Model:

- Sequential model: The layers are stacked sequentially.
- LSTM layer: This layer has 50 units (neurons), which will learn patterns in the sequential expense data. The relu activation function introduces non-linearity, making the model more capable of learning complex patterns.
- Dense layer: The final layer has just one neuron since the goal is to predict a single numerical value (the next day's expense).

Compiling: The model is compiled using the Adam optimizer (a widely used optimization algorithm) and the Mean Squared Error (MSE) loss function, which measures how far the model's predictions are from the actual values

7.Training the Model

```
model.fit(X_train, X_train, epochs=50, verbose=1)
```

Training: The model is trained for 50 epochs on the training data. During each epoch, the model learns from the input sequences and attempts to improve its predictions by minimizing the MSE loss.

8.Making Predictions

```
scaled_data_with_input = scaler.transform(np.array(all_data).reshape(-1, 1))
new_seq = scaled_data_with_input[-seq_length:]
new_seq = new_seq.reshape((1, seq_length, 1))

# Make prediction for the next day
new_prediction = model.predict(new_seq).flatten()
```

🍫 **Scaling New Input**: The input data (including recent expenses) is scaled using the same `scaler` used earlier, ensuring consistency between the training and prediction stages.

🍫 **Creating New Sequence**: The most recent sequence (the last 7 days) is prepared as input for the model.

🍫 **Predicting**: The LSTM model predicts the next day's expense based on the last 7 days of spending patterns.

🍫 **Inverse Scaling**: Since the prediction is in the normalized range (0 to 1), it's inversely transformed back to the original scale (currency).

9. Calculating Min and Max Predictions

```
def calculate_min_max_with_error(prediction, error_std,actual ,predicted):
"""Returns a tuple with min and max predictions based on error standard
deviation."""
adjusted_error = error_std * np.where(actual > predicted, 1.5, 0.5)
min_pred = prediction - 1.96 * adjusted_error
max_pred = prediction + 1.96 * adjusted_error
return min_pred, max_pred
```

Min and Max Calculation: Based on the prediction and the standard deviation of errors, this function calculates a confidence range (minimum and maximum) around the predicted value. The range is wider if the model consistently underperforms or the errors are larger.

- 1.96 multiplier: This corresponds to a 95% confidence interval, giving the user a range of likely values for the next day's spending.

10. Plotting the Results

```
ax.clear()

# Plot the actual expenditures
ax.plot(np.arange(len(all_data)),    all_data,    label='Actual    Spending',
color='blue')

# Plot the predicted spending
ax.plot(np.arange(len(all_predictions)),  all_predictions,  label='Suggeseted
Spending', color='red')
#         ax.plot(np.arange(len(all_predictions)),         old_predicted_data,
label='Predicted Spending', color='gree/n')

# Plot the min and max predictions as bands
x.fill_between(np.arange(len(all_predictions)),           all_min_predictions,
all_max_predictions   ,color='orange',   alpha=0.3,   label='Prediction   Range
(Min-Max)')

# ax.text(1, 13, f'Accuracy: {100-mape:.2f}%', fontsize=12, color='black',
#                         bbox=dict(facecolor='none',        edgecolor='black',
boxstyle='round,pad=0.5'))


ax.set_xlabel('Day Of Month')
ax.set_ylabel('Expenditure (₹)')
ax.set_title(f'Daily Expenditure with 1-Day LSTM Prediction and Range    ')
ax.legend()
ax.grid(True)

# Redraw the updated plot
fig.canvas.draw()
fig.canvas.flush_events()
```

Plotting: A live, interactive plot is created using matplotlib to visualize both actual and predicted spending. The actual expenses are shown in blue, while predictions are displayed in red. The plot updates dynamically as new data is provided.

11. Updating the Graph with New Data

```
def update_graph(new_data):
global                all_predictions,                all_min_predictions,
all_max_predictions,new_data_feeded,new_predicted_data,budget,i,avg
# old_predictions=[]

budget=budget-new_data
if i>=30:
present_day_avg=budget
```

```
else:
present_day_avg=budget/(30-i)



new_data_feeded.append(float(new_data))

# Calculate errors between actual and predicted values (after the look-back
period)
errors           =           np.abs(np.array(new_data_feeded[:])           -
np.array(new_predicted_data[:]))

# Compute the standard deviation of the errors (this helps assess the
spread of the predictions)
error_std = np.std(errors)

# Update the actual data with new input
all_data.append(new_data)

# Expand the all_predictions, all_min_predictions, and all_max_predictions
lists
all_predictions.append(None)
all_min_predictions.append(None)
all_max_predictions.append(None)
# old_predictions.append(None)

scaled_data_with_input   =   scaler.transform(np.array(all_data).reshape(-1,
1))

# Update the sequence with new data and predict the next 1 day
new_seq = scaled_data_with_input[-seq_length:]
new_seq = new_seq.reshape((1, seq_length, 1))

# Make prediction for the next day
new_prediction = model.predict(new_seq).flatten()
new_prediction_inv   =   scaler.inverse_transform(new_prediction.reshape(-1,
1)).flatten()

print("presnt day average:",present_day_avg)
# old_predicted_data=new_prediction_inv[0]
print("Original Prediction:",new_prediction_inv[0])

percentage=None



ideal_avg=1/30*i
if                      new_limit>=present_day_avg                      and
(present_day_avg/new_prediction_inv[0]<0.8                               or
new_prediction_inv[0]/present_day_avg<0.8):
actual_avg=budget/3000
if (ideal_avg/actual_avg)>=0.5:
decrement=1-(ideal_avg/actual_avg)
```

```
else:
decrement=0.5-(ideal_avg/actual_avg)
new_prediction_inv[0]=new_prediction_inv[0]*decrement
if new_prediction_inv[0]<=5:
new_prediction_inv[0]=present_day_avg*0.8
elif                    new_limit>=present_day_avg                    and
(present_day_avg/new_prediction_inv[0]>0.8                             or
new_prediction_inv[0]/present_day_avg>0.8):
if present_day_avg>new_prediction_inv[0]:
new_prediction_inv[0]=new_prediction_inv[0]*1.08
else:
new_prediction_inv[0]=new_prediction_inv[0]*0.92
else:
actual_avg=budget/3000
if ideal_avg<actual_avg:
if                 new_prediction_inv[0]>new_limit*0.3                 and
new_prediction_inv[0]<new_limit*1.4:
incremenet=1+(ideal_avg/actual_avg)
else:
if new_limit<present_day_avg and new_limit<new_prediction_inv[0]*0.7 :
print("hii")
if ideal_avg/actual_avg<0.5:
incremenet=1+(ideal_avg/actual_avg)
else :
incremenet=0.5+(ideal_avg/actual_avg)
else:
print("Hello")
incremenet=1-(ideal_avg/actual_avg)
else:
print("Hello2")
incremenet=0.6+(actual_avg/ideal_avg)

new_prediction_inv[0]=new_prediction_inv[0]*incremenet

print("Percentage:",percentage)

# new_prediction_inv[0]=new_prediction_inv[0]*((100+(percentage))/100)
print( "calculated Prediction :",new_prediction_inv[0])

# Calculate min and max predictions

prediction=new_prediction_inv[0]
if new_data>=2.5*(all_predictions[len(all_predictions) - 2]):
min_pred,                              max_pred                              =
calculate_min_max_with_error(present_day_avg,error_std,new_data_feeded[0],n
ew_predicted_data[0])
else:
min_pred,                              max_pred                              =
calculate_min_max_with_error(prediction,error_std,new_data_feeded[0],new_pr
edicted_data[0])

mape, mae = calculate_accuracy(new_data_feeded, new_predicted_data)
```

```
print(f'MAPE: {100-mape:.2f}%, MAE: {mae:.2f}₹')
new_predicted_data.append(new_prediction_inv[0])
# Update predictions
all_predictions[len(all_predictions) - 1] = new_prediction_inv[0]
all_min_predictions[len(all_predictions) - 1] = min_pred
all_max_predictions[len(all_predictions) - 1] = max_pred
all_min_predictions = np.array(all_min_predictions, dtype=np.float64)
all_max_predictions = np.array(all_max_predictions, dtype=np.float64)


print()
# Clear the previous graph
ax.clear()

# Plot the actual expenditures
ax.plot(np.arange(len(all_data)),   all_data,   label='Actual   Spending',
color='blue')

# Plot the predicted spending
ax.plot(np.arange(len(all_predictions)), all_predictions, label='Suggeseted
Spending', color='red')
#       ax.plot(np.arange(len(all_predictions)),       old_predicted_data,
label='Predicted Spending', color='gree/n')

# Plot the min and max predictions as bands
ax.fill_between(np.arange(len(all_predictions)),       all_min_predictions,
all_max_predictions ,
color='orange', alpha=0.3, label='Prediction Range (Min-Max)')

# ax.text(1, 13, f'Accuracy: {100-mape:.2f}%', fontsize=12, color='black',
#                       bbox=dict(facecolor='none',       edgecolor='black',
boxstyle='round,pad=0.5'))


ax.set_xlabel('Day Of Month')
ax.set_ylabel('Expenditure (₹)')
ax.set_title(f'Daily Expenditure with 1-Day LSTM Prediction and Range    ')
ax.legend()
ax.grid(True)

# Redraw the updated plot
fig.canvas.draw()
fig.canvas.flush_events()


all_min_predictions             =             np.array(all_min_predictions,
dtype=np.float64).tolist()
all_max_predictions             =             np.array(all_max_predictions,
dtype=np.float64).tolist()

return prediction
```

🎬 **Appending New Data**: The new actual spending data (`new_data`) is added to `all_data`, which stores all past actual expense values.

🎬 **Scaling New Data**: The `scaler` is used to normalize `all_data` (including the new data). This is necessary because the LSTM model was trained on scaled data, and consistency is key for accurate predictions.

🎬 **Creating a New Sequence**: A sequence of the last 7 days (or another sequence length, `seq_length`) is generated from the scaled data. This sequence serves as input for the LSTM model to predict the next day's expense.

🎬 **Predicting the Next Expense**: The model predicts the next day's expense using the recent sequence of expenses. Since the model output is scaled, it is inverse-transformed back to the original currency range.

🎬 **Storing Predictions**: The new prediction is appended to `all_predictions`, which contains all predicted values for comparison with the actual data.

🎬 **Updating the Plot**: The plot is cleared, and then both the actual expenses (`all_data`) and predictions (`all_predictions`) are plotted again. The blue line represents the actual data, and the red line represents predictions.

🎬 **Rendering the Updated Plot**: The updated plot is rendered using `plt.draw()` and a slight pause is added to allow real-time updates.

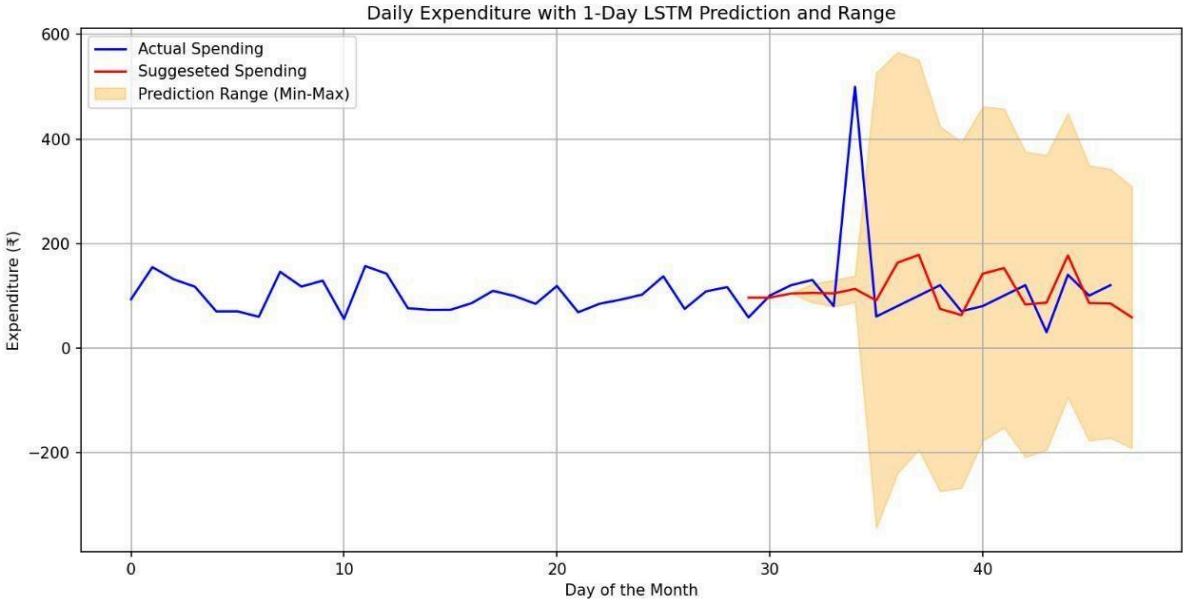
12. Accuracy Function

```
def calculate_accuracy(actual_values, predicted_values):
"""Calculate accuracy of predictions based on actual values."""
actual = np.array(actual_values)
predicted = np.array(predicted_values)

# Calculate Mean Absolute Percentage Error (MAPE)
mape = np.mean(np.abs((actual - predicted) / actual)) * 100

# Calculate Mean Absolute Error (MAE)
mae = np.mean(np.abs(actual - predicted))

return mape, mae
```

The calculate_accuracy() function is useful for evaluating how well your model predicts future expenses compared to actual spending. After running predictions for a period, you can use this function to quantify the model's performance, find weaknesses, and tweak the model for improvement. For example, high MAPE would indicate that the model consistently misses by large percentages, so it may need more tuning or additional features (like different sequences or variables).

Chapter 6:
Result and Discussion

The performance of the LSTM model shows promise in predicting daily and weekly expenditure on multi-category levels, which can perhaps reflect the spending in the future by its users. To evaluate the performance of the model, the author adopted some conventional evaluation metrics such as MSE and RMSE, both of which showed satisfactory performance. Results visualization shows the model closely follows actual patterns of spending, especially where budgeting is more regulated, such as food and transport. However, in categories with less regular expenditure patterns, such as entertainment, the model's predictions show slight deviations, indicating an area for improvement related to managing unpredictable spikes in expenditure.



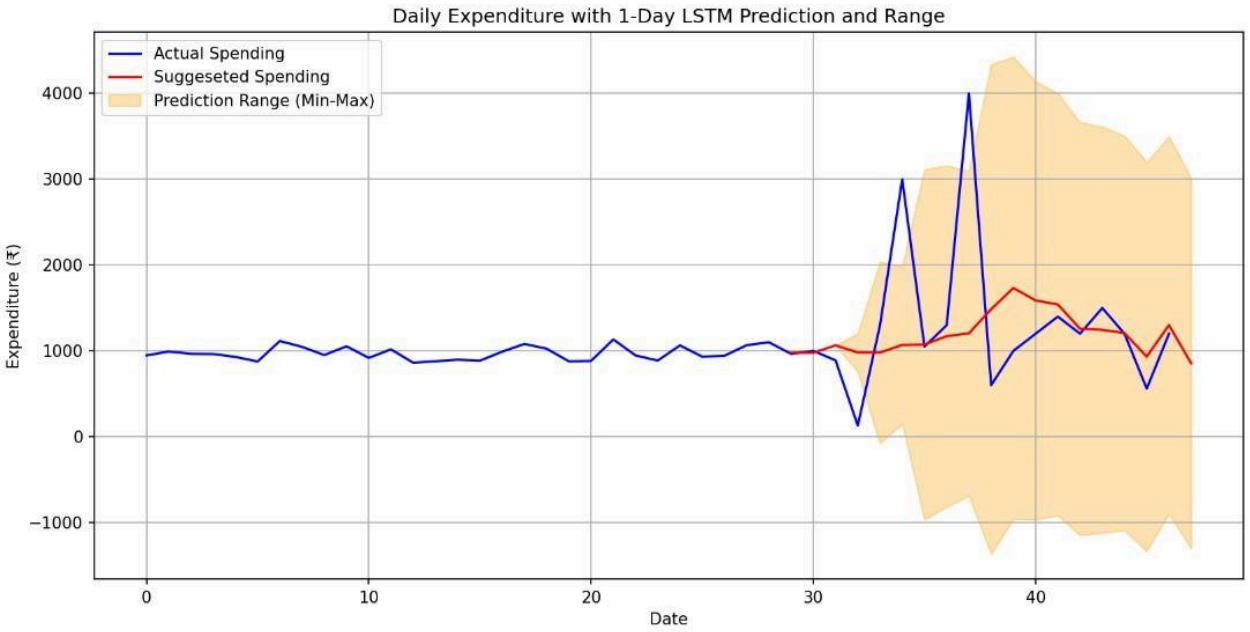
The system can forecast what the near future is going to cost by showing a proactive approach to expense management, thus staying within one's budget of the month without overspending. Once again, the process is user-friendly for even nontechnical people because of how easily one can interpret the results. Further scope for discussion would include the improvement of the model by introducing external influences such as holidays or other special events that may influence the behavior of the people involved in order to enhance the model's predictability.

Despite the effective implementation of the expense prediction system, some drawbacks were detected:

1. Inadequate Training Data: the model fails to make better predictions and calculation as training data are not sufficiently broad and comprehensive.

2. Overprediction and Underprediction: There are instances when the model tends either over to predict or under predict expenses due to unique patterns in previous spending behavior.

3. Boundary Condition-Limitation: Creating mathematical boundaries to avoid extreme outliers at times reduces the flexibility of the model.
4. Category Bias: The model appears to perform better with mundane categories; food items are quite convenient and comfortable, but less frequent or rare expenses such as entertainment do not work well.



5. Uncommon Expenses Sparse Data: Forecasts about infrequent events with high expenditure are not very accurate because the number of instances is far too small.

6. Exclusion of External Factors: The model cares not for how inflationary pressures built on top of previous years, a shift in income, or an emergency may run.

7. Poor Long-Term Accuracy: LSTMs will do very well with short-term predictions but have poor accuracy with respect to long-term expenses.

Here are some accuracies of prediction of some users for month of July

ACCURACY FOR EACH RUN-

Sr. no	Budget	Daily Prediction (Daily)	Food Category prediction (Weekly)
1	15000/-	82.36%	90.12%
2	8000/-	90.85%	85.24%
3	5000/-	91.68%	88.12%
4	10000/-	82.58%	84.36%
5	12000/-	79.27%	77.03%
6	9800/-	88.31%	83.53%
7	20000/-	76.69%	72.36%

Chapter 7:
Conclusion

The project utilizes Long Short-Term Memory (LSTM) artificial neural networks to predict expenditures, demonstrating their effectiveness in the realm of personal finance management. LSTMs are particularly adept at handling time-series data, making them suitable for analyzing spending trends over time.

By examining historical spending behavior, the system can generate accurate daily and weekly expenditure forecasts. This capability enables users to understand their financial habits better and make more informed decisions regarding their spending, thus promoting better financial discipline.

The model's ability to predict future expenditures helps users stay within their monthly budgets. This proactive approach significantly reduces the financial stress often associated with unexpected expenses, allowing users to plan their finances more effectively.

The results indicate that the system performs well in predicting routine expenditures, providing users with reliable insights. However, it also highlights areas for potential improvement, especially concerning irregular spending categories, which may be influenced by unique circumstances or events.

To enhance the model's accuracy and robustness, it is recommended to integrate additional data sources. Incorporating factors such as income levels, economic trends, and seasonal variations could provide a more comprehensive view of spending behavior, leading to more precise predictions.

Ultimately, this project serves as a proof of concept for the application of LSTM and machine learning techniques in addressing real-world financial challenges. It opens up avenues for further research and development in personal finance applications, highlighting the potential of technology to improve individual financial management.

Chapter 7:

References

1) ISSN: 1001-4055 Vol. 45 No. 1 (2024), Expense Tracker: A Smart Approach to Track Daily Expense.

2) EXPENDITURE MANAGEMENT SYSTEM.

3) International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429 Volume 9 Issue IV Apr 2021- Available at www.ijraset.com ©IJRASET: All Rights are Reserved 1067 Expense Tracker