**Documentation for Receipt Prediction Model and Web Application**

**Goal/Purpose:**

To forecast the monthly volume of receipts to be scanned in 2022 by examining the trends observed in a dataset containing daily receipt counts from 2021.

**Prerequisites:**

* Python 3.x(latest version if available)
* Internet connection for downloading Python and necessary libraries.

**Installation and Setup:**

* **Install Python:** Download and install Python from python.org. Ensure to add Python to PATH during installation.
* **Install Libraries:** Open a command line interface and simply load the requirements.txt” by running this command in your terminal:  
  ”pip install -r requirements. txt”
* **Prepare Project Files:** Place the provided “receipts.py”, “app.py”, and “data\_daily.csv” in a single folder. Inside this folder, create a subfolder named templates and place index.html file there.

**Running the Machine Learning Model:**

* Open a command line interface and navigate to the project folder.
* Run the script “receipt.py” with the command: “python receipt.py”
* This trains the machine learning model and prepares it for predictions.

**Starting the Flask Web Application:**

* In the same command line interface, run your Flask application script (e.g., app.py) using: “python app.py”.
* Open a web browser and go to “http://127.0.0.1:5000/” to access the application.

**Using the Web Application:**

* Select a month from the dropdown menu and click "Predict" to view the predicted receipt count.
* This developed application also displays visual data charts for better understanding.

**Detailed Explanation of Code Functionalities:**

* **Receipt.py:**

The “receipt.py” script is responsible for training a machine learning (ML) model using historical data. This model predicts the number of scanned receipts for each month of the year 2022 based on the data from 2021.

1. Import Libraries:

* “pandas” for data manipulation.
* “numpy” for numerical operations.
* “tensorflow” and its components for building and training the neural network model.
* “matplotlib” for visualizing.
* “flask” to establish server connection.

1. Load and Preprocess Data:

* Load data from data\_daily.csv.
* Convert dates to a datetime format and extract year and month.
* Group data by month and calculate the sum of receipts for each month.

1. Data Normalization:

* Normalize months and receipt counts to ensure efficient training.

1. Create LSTM Model:

* LSTM layers are used for their ability to remember long-term patterns, ideal for time series data.
* Dropout layers prevent overfitting by randomly dropping units from the neural network during training.
* The Dense layer is used for outputting the prediction.

1. Compile and Train the Model:

* The model uses Mean Squared Error (MSE) as the loss function, suitable for regression problems.
* Early stopping is used to prevent overtraining.

1. Predictions for 2022:

* Normalize 2022 months and predict receipt counts.
* Denormalize predictions to obtain actual values.

1. Save Model and Parameters:

* The trained model and normalization parameters are saved for later use in the Flask app.
* **App.py:**

This “app.py” script creates a Flask web application that interacts with the trained ML model to provide predictions based on user input.

1. Set Up Flask and Import Libraries:

* Flask is used to create a web server.
* Other libraries are for data processing, model loading, and visualization.

1. Load Trained Model and Data:

* The script loads the trained model and the normalization parameters.

1. Flask Routes:

* ‘/’: The index route displays the home page with a dropdown for month selection.
* ‘/predict’: Handles the prediction request and displays results and visualizations.

1. Prediction and Visualization:

* It uses the model to predict receipt counts.
* Generates plots for 2021 data and 2022 predictions.

1. Render Templates:

* The script uses index.html to render the web pages.
* **Index.html:**

“index.html” provides the structure and layout of the web application's user interface.

1. HTML Structure:

* Contains basic HTML tags (head, body) for webpage structure.

1. Form for Month Selection:

* Includes a dropdown to select a month and a submit button to request predictions.

1. Display Predictions and Visualizations:

* Conditional statements ({% if %}) to show predictions and plots if available.

1. Styling and Layout:

* While this file doesn’t include CSS, it structures the content in a user-friendly manner.