### Uber Trip Prediction – FastAPI Project Report

# 1. Objective

The goal of this project is to build a **web-based prediction system** that estimates the **number of Uber trips** based on key features like:

- Dispatching Base Number
  - **Active Vehicles**
- Day
- Month
- Year

The system uses a **machine learning regression model** trained on Uber trip data, served using **FastAPI**, and integrated with a **SQLite database** for storing predictions.

### 2. Tech Stack

Component	Technology Used
Backend Framework	FastAPI
Frontend	HTML, CSS, Jinja2
Database	SQLite (via SQLAlchemy ORM)
Model	Scikit-learn Regression Model
Language	Python 3.10+
Model Files	uber_trip.pkl, scaler.pkl, label_encoder.pkl
Server	Uvicorn

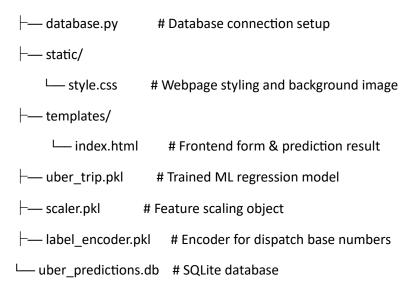
## 🤏 3. System Architecture

### Project Structure

Uber\_Trip\_Prediction/

├— app.py # FastAPI main backend file

├— models.py # Model loading, prediction logic, and ORM table



## 🧩 4. Database Design

### **Table: predictions**

Column	Туре	Description
id	Integer (Primary Key)	Unique ID for each prediction
dispatching_base_number	String	Encoded base number of Uber hub
active_vehicles	Integer	Number of active vehicles
day	Integer	Day of the month
month	Integer	Month number
year	Integer	Year
prediction	Float	Predicted number of trips

Database File: uber\_predictions.db

# 5. Model and Preprocessing

#### **Model Files**

- **uber\_trip.pkl** → Trained regression model (e.g., RandomForest or XGBoost)
- scaler.pkl → StandardScaler or MinMaxScaler used during training
- label\_encoder.pkl → Encodes dispatch base numbers like B02512, B02764, etc.

#### **Model Input Features**

[dispatching\_base\_number, active\_vehicles, day, month, year]

#### **Model Output**

**Predicted Number of Trips** 

### **6.** Backend Logic (app.py)

- 1. FastAPI serves two routes:
  - $\circ$  /  $\rightarrow$  Renders the main form.
  - o /predict → Takes user input, runs the ML model, saves result to DB, and displays the prediction.
- 2. Uses SQLAlchemy for ORM.
- 3. Auto-generates the SQLite database and table.
- 4. Background image + styled frontend using CSS.

## 7. Frontend Design (index.html + style.css)

#### **Features**

- Beautiful form UI with Uber image background
- Dropdown for dispatch base number
- ✓ Displays the prediction dynamically
- Shows recent prediction history at bottom of the page

### 8. Execution Steps

1 Create Virtual Environment

python -m venv venv

venv\Scripts\activate

2 Install Dependencies

pip install fastapi uvicorn sqlalchemy jinja2 joblib scikit-learn

Run the App

uvicorn app:app --reload --port 5000

Open in Browser

http://127.0.0.1:5000

## 9. How to View SQLite Database

### Option 2 — Use **DB Browser for SQLite** (GUI tool):

- 1. Download from <a href="https://sqlitebrowser.org">https://sqlitebrowser.org</a>
- 2. Open uber\_predictions.db
- 3. Go to the **Browse Data** tab to view your predictions

# 🚺 10. Sample Prediction Example

Input	Output
Dispatch Base: B02764	Predicted Trips: 1234.56
Active Vehicles: 180	
Day: 12	
Month: 7	
Year: 2024	

### 11. Future Enhancements

- Deploy the app using Render, Azure, or AWS EC2.
- Add user authentication.
- Visualize historical prediction trends using Plotly.
- Allow CSV upload for batch predictions.
- Connect to PostgreSQL or MySQL instead of SQLite.

## 🔽 12. Key Learnings

- Integration of machine learning models into real-time web apps.
- Use of FastAPI + Jinja2 templates for frontend rendering.
- Proper management of **ORM models and DB sessions**.
- Clean separation of code into **modular files**: app.py, models.py, database.py.