Project Design Phase-II
Technology Stack (Architecture & Stack)

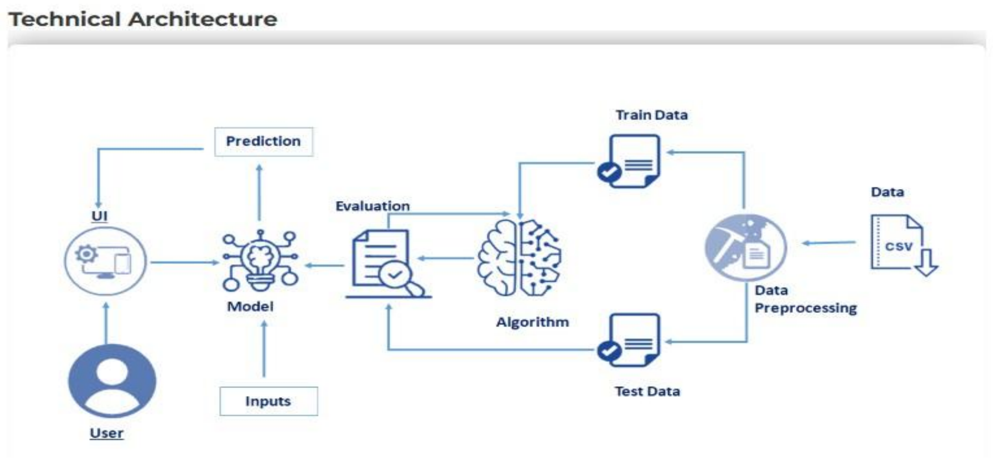
Date	31 January 3035
Team ID	LTVIP2025TMID45044
Project Name	traffictelligence: advanced traffic volume estimation with machine
Maximum Marks	4 Marks

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

Example: Order processing during pandemics for offline mode

Reference: https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/



S.No	Component	Description	Technology
1.	User Interface	Web interface where users input features to get traffic volume prediction	HTML, CSS
2.	Application Logic-1	Backend logic to receive form data, load model, and return predictions	Flask,Python
3.	Application Logic-2	Preprocessing input and encoding categorical features (e.g., weather)	Scikit-learn encoder (joblib/pickle)
4.	Application Logic-3	Model prediction logic using trained ML model	Scikit-learn, Pandas, NumPy
5.	Database	Storing historical traffic data (CSV or tabular format)	(Optional) Hosting traffic data in a cloud
6.	Cloud Database		IBM DB2, IBM Cloudant etc.
		(Optional) Hosting traffic data in a cloud environment	
7.	File Storage	Model and encoder pickle files stored locally	Local File System
8.	External API-1	(Optional) Real-time weather information can be pulled to enhance predictions	IBM Weather API / OpenWeather API

9.	External API-2	Not applicable	Not used
10.	Machine Learning Model	Predict traffic volume based on weather/time features	Random Forest Regressor, Decision Tree, SVM
11.	Infrastructure (Server / Cloud)	Application deployed locally using Flask	local server.

Table-2: Application Characteristics:

S.No	**Characteristics**	**Description**	**Technology**
1.	Open-Source Frameworks	Open-source libraries and frameworks used in model building and web development	Flask, Scikit-learn, Pandas, NumPy, Matplotlib
2.	Security Implementations	Model files and user inputs are locally handled; Flask provides route-based access. Optional: SHA-256 hashing	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	3-Tier Architecture using Flask + REST API
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Nginx, Cloud Load Balancer (optional)
S.No	**Characteristics**	**Description**	**Technology**
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN’s) etc.	Flask multithreading,pickle loading

References:

https://c4model.com/ https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/ https://www.ibm.com/cloud/architecture https://aws.amazon.com/architecture https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d