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

Date	
Team ID	LTVIP2026TMIDS45779
Project Name	Electric Motor Temperature Prediction Using Meachine Learning
Maximum Marks	4 Marks

Technical Architecture:

Electric Motor Temperature Prediction System

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

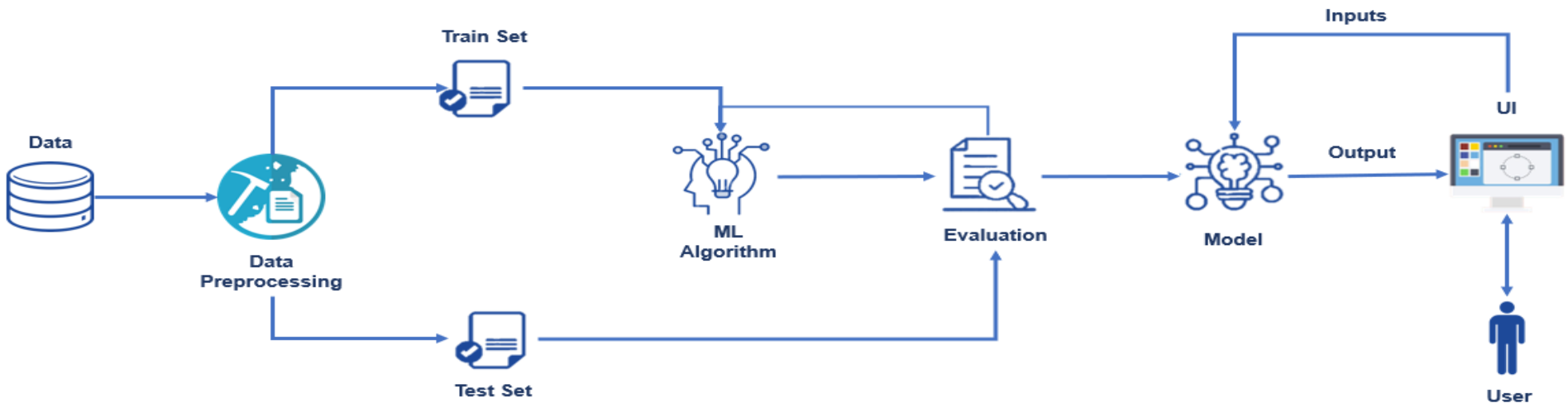

Table-1: Components & Technologies:

S.No	Component	Description	Technology
1	User Interface	Web interface where users enter motor parameters (ambient, coolant, voltage, current) and receive prediction results	HTML, CSS
2	Application Logic-1	Backend processing logic to receive form input and handle requests	Python (Flask Framework)
3	Application Logic-2	Data preprocessing logic (scaling input features before prediction)	Scikit-learn (MinMaxScaler)
4	Application Logic-3	Machine learning prediction logic	Random Forest Regressor (Scikit-learn)
5	Database	Not required for current version (static ML model prediction). Can be extended for storing logs/history	Optional: MySQL / SQLite
6	Cloud Database	Not implemented currently (Future scope for cloud logging)	IBM Cloudant / AWS RDS (Future Enhancement)
7	File Storage	Storage of trained ML model and scaler files	Joblib (.save files) stored in Local File System
8	External API-1	Dataset source for training	Kaggle Electric Motor Temperature Dataset
9	External API-2	Not used currently (Can integrate IoT sensor API in future)	IoT Sensor API (Future)
10	Machine Learning Model	Predict Permanent Magnet (PM) surface temperature based on motor parameters	Random Forest Regression Model
11	Infrastructure (Server / Cloud)	Application deployed locally for demo and academic use	Local Server (Flask Development Server), VS Code

Table-2: Application Characteristics

S.No	Characteristics	Description	Technology
1	Open-Source Frameworks	Frameworks and libraries used for development and ML implementation	Flask, Scikit-learn, Pandas, NumPy
2	Security Implementations	Input validation on form fields, local hosting, debug control, protection against invalid data	Flask validation, Python exception handling
3	Scalable Architecture	Modular design separating UI, preprocessing, ML model, and backend. Can be extended to microservices architecture	Flask (3-tier structure possible), REST API (Future)
4	Availability	Currently single-instance local server. Can be deployed to cloud with high availability setup	AWS EC2 / Azure / IBM Cloud (Future Scope)
5	Performance	Fast prediction (single inference < 1 second). Model preloaded at server start. No retraining required during runtime	Random Forest Model, Preloaded Joblib Model

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