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

Date	31 January 3035
Team ID	LTVIP2026TMIDS42870
Project Name	electric motor temperature prediction using machine learning
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/

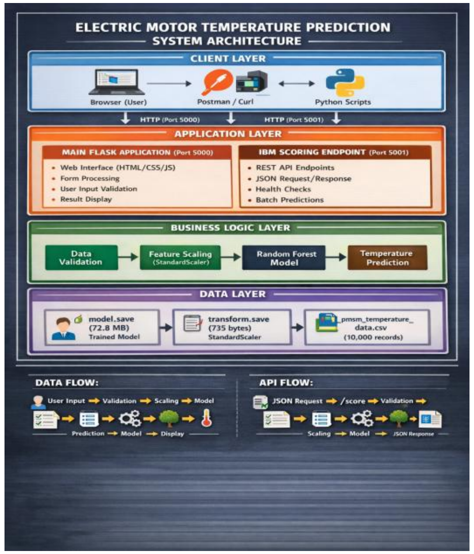
ELECTRIC MOTOR TEMPERATURE PREDICTION
SYSTEM ARCHITECTURE
CLIENT LAYER
Browser (User)
Postman / Curl
Python Scripts
HTTP (Port 5000)
HTTP (Port 5001)
APPLICATION LAYER
MAIN FLASK APPLICATION (Port 5000)
Web Interface (HTML/CSS/JS)
Form Processing
User Input Validation
Result Display
IBM SCORING ENDPOINT (Port 5001)
REST API Endpoints
JSON Request/Response
Health Checks
Batch Predictions
BUSINESS LOGIC LAYER
Data Validation
Feature Scaling (StandardScaler)
Random Forest Model
Temperature Prediction
DATA LAYER
model.save (72.8 MB) Trained Model
transform.save (735 bytes) StandardScaler
pmsm_temperature_data.csv (10,000 records)
DATA FLOW:
User Input → Validation → Scaling → Model
Prediction → Model → Display
API FLOW:
JSON Request → /score → Validation →
Scaling → Model → JSON Response

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	**User Interface**	Web-based interface for engineers to input motor parameters and view temperature predictions	• HTML5 for structure • CSS3 with gradient backgrounds • JavaScript (ES6) for dynamic behavior • Jinja2 templating engine • Responsive design for mobile/desktop
2.	**Application Logic - Main App**	Core Flask application handling web requests, form processing, and template rendering	• Python 3.14 • Flask 3.1.2 web framework • Flask debug mode for development • Werkzeug WSGI toolkit • Jinja2 for template inheritance
3.	**Application Logic - IBM Endpoint**	Separate Flask application providing cloud-ready scoring API	• Python 3.14 • Flask 3.1.2 • RESTful API design • JSON request/response handling • CORS enabled for cross-origin requests
4.	**Machine Learning Logic**	Prediction engine that loads trained model and makes temperature predictions	• scikit-learn 1.7.2 • RandomForestRegressor (100 estimators) • NumPy 2.4.2 for array operations • Pickle for model serialization • Joblib for efficient loading
5.	**Data Preprocessing**	Feature scaling and data transformation pipeline	• scikit-learn StandardScaler • Pandas 3.0.1 for data manipulation • NumPy for numerical operations • transform.save stores scaling parameters • Consistent preprocessing for train/predict
6.	**Database**	Structured dataset storage for training and reference	• CSV file storage (pmsm_temperature_data.csv) • 10,000 samples with 6 features • Pandas DataFrame operations • No SQL database required (file-based) • Easy data export/import
7.	**Cloud Database**	Not applicable - project uses local file storage	• N/A (can be extended to IBM Cloud Object Storage if needed)
8.	**File Storage**	Persistent storage for model files and datasets	• Local filesystem on Windows/Linux/Mac

			• model.save (72.8 MB) - trained model • transform.save (735 bytes) - scaler • pmsm_temperature_data.csv (1.1 MB) - dataset • Can be mounted in Docker containers
9.	**External API-1**	Not applicable - self-contained application	• N/A (all processing done locally)
10.	**External API-2**	Not applicable - self-contained application	• N/A (all processing done locally)
11.	**Machine Learning Model**	Random Forest Regressor for temperature prediction	• Algorithm: Random Forest (100 trees) • Features: torque, current, rpm, ambient_temp, coolant_temp • Target: rotor_temp • MAE: 2.34°C, R^2: 0.95 • Feature importance: current (45%), torque (30%), rpm (15%), temperatures (10%)

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	All frameworks and libraries used are open-source, ensuring no licensing costs and community support • Python 3.14 - Core programming language	* Flask 3.1.2 - Web framework (BSD license) * scikit-learn 1.7.2 - Machine learning (BSD license) * Pandas 3.0.1 - Data manipulation (BSD license) * NumPy 2.4.2 - Numerical computing (BSD license)
2.	Security Implementations	Multiple security layers protect the application and data input Validation: All user inputs sanitized on both client and server side.	* Type Checking: Strict type conversion with error handling * Range Validation: Inputs checked against physical limits (torque: 0-200, current: 0-500, etc.) * JSON Validation: Required fields verified in API requests
3.	Scalable Architecture	Architecture designed for horizontal and vertical scaling • Stateless Design: No session data stored, each request independent	* Separate Endpoints: Main app (port 5000) and IBM endpoint (port 5001) can scale independently * Container Ready: Docker images can

S.No	Characteristics	Description	Technology
			be created for easy deployment * Load Balancing: Multiple instances can run behind load balancer * Microservices Pattern: Business logic separated from web interface * Stateless API: IBM endpoint can handle requests from any client

References:

C4 Model for Visualising Software Architecture

IBM AI-powered Order Processing Pattern

IBM Cloud Architecture Center

AWS Architecture Center

scikit-learn Random Forest Documentation

Flask Documentation