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

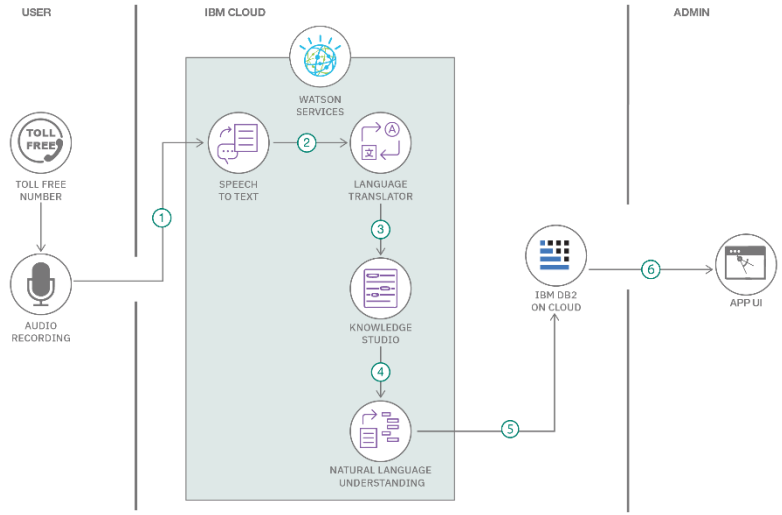
Date	21 December 2025
Team ID	LTVIP2026TMIDS83736
Project Name	**Rising Waters: A Machine Learning Approach to Flood Prediction**
Maximum Marks	4 Marks

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/



Guidelines:

- Include all the processes (As an application logic / Technology Block)
- Provide infrastructural demarcation (Local / Cloud)
- Indicate external interfaces (third party API's etc.)
- Indicate Data Storage components / services
- Indicate interface to machine learning models (if applicable)

Table-1: Components & Technologies

S.No	Component	Description	Technology
1.	User Interface	Web-based UI for user interaction, input parameters, and view predictions	HTML5, CSS3, JavaScript
2.	Application Logic - Web Server	Flask web server to handle HTTP requests, route management, and serve HTML templates	Python Flask Framework
3.	Application Logic - Data Processing	Preprocessing user inputs using saved transformer before prediction	Python, Scikit-learn (transform.save)
4.	Application Logic - ML Inference	Load trained ML model and generate flood predictions based on input parameters	Python, Scikit-learn, Pickle (floods.save)
5.	Machine Learning Model	Pre-trained model for flood probability prediction based on 8 parameters (monsoon, topography, drainage, river management, deforestation, urbanization, climate change, dam)	Scikit-learn (Logistic Regression / Random Forest / Decision Tree)
6.	Model Training	Jupyter notebook for model development and training pipelines	Jupyter Notebook (Floods.ipynb), Pandas, NumPy
7.	Model Storage	Serialized ML model and transformer files	Pickle files (.save format)
8.	Dataset Storage	Historical flood data for model training and retraining	CSV/Excel files (local filesystem)
9.	Model Regeneration Script	Script to retrain and update ML model with new data	Python (regenerate_model.py)
10.	Static File Server	Serve CSS stylesheets for UI styling	Flask static file handling

Table-2: Application Characteristics

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask - Web application framework Scikit-learn - Machine learning library Pandas - Data manipulation NumPy - Numerical computing Jupyter - Model development	Python 3.x, Flask, Scikit-learn, Pandas, NumPy, Jupyter
2.	Security Implementations	Input validation to prevent injection attacks Form data sanitization Secure file handling for model files Protection against XSS and CSRF	Flask built-in security features, HTML escaping, Input validation
3.	Scalable Architecture	3-tier architecture separating UI, logic, and data Modular design allows easy feature additions ML model can be retrained independently Stateless Flask application supports horizontal scaling	Flask (WSGI-compatible), Modular Python code, Serialized models
4.	Availability	Flask development server for local deployment Can be deployed on production WSGI servers (Gunicorn/uWSGI) Model files loaded once at startup for fast inference Minimal external dependencies	Flask, Gunicorn (for production), In-memory model loading
5.	Performance	Pre-loaded ML model in memory (no repeated disk I/O) Lightweight Flask framework Fast prediction response (<3 seconds) Preprocessed transformer for efficient data preparation Static CSS files cached by browser	In-memory model caching, Pickle serialization, Flask optimization
6.	Maintainability	Clean separation of concerns (templates, static files, application logic) Model regeneration script for easy updates Jupyter notebook for model experimentation Virtual environment for dependency management	Modular code structure, Python virtual environment, Documentation
7.	Portability	Python-based cross-platform application Can run on Windows, Linux, macOS Virtual environment ensures consistent dependencies	Python virtual environment (.venv), Platform-independent