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

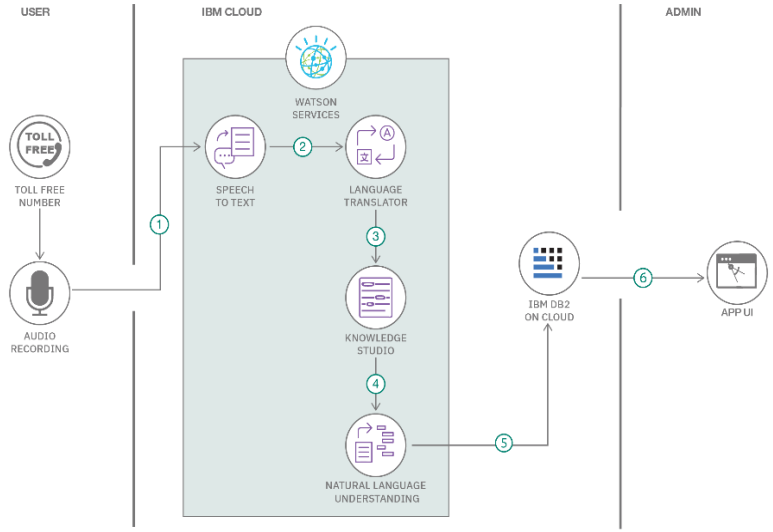
Date	26th June 2025
Team ID	LTVIP2025TMID59290
Project Name	FlightFinder
Maximum Marks	4 Marks

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)

S.No	Component	Description	Technology

1.	User Interface	How user interacts with application Web UI	HTML, CSS,ReactJS, Bootstrap, CSS etc.
2.	Application Logic-1	Logic for a process in the application	JavaScript.
3.	Database	Data Type, Configurations etc.	MongoDB, Mongoose.
4.	File Storage	File storage requirements	MongoDB Cluster storage.
5.	External API-1	Purpose of External API used in the application	
6.	External API-2	Purpose of External API used in the application	

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Frontend uses React (via Vite), Tailwind CSS, Bootstrap for UI components, Axios for HTTP requests. Backend is built using Node.js with Express.	React, Vite, CSS, Bootstrap, Axios, Node.js, Express.js
2.	Security Implementations	Passwords are encrypted using bcrypt. CORS is implemented for secure cross-origin communication. Input validations prevent injection attacks.	bcrypt, CORS, express-validator, Helmet (optional)
3.	Scalable Architecture	Follows a modular architecture separating frontend, backend, and database (3-tier). Can be containerized using Docker for scaling.	Node.js Microservices (optional),
4.	Availability	Application can be deployed on cloud platforms (e.g., Heroku, Render, AWS) with horizontal scaling. Load balancers can be used if demand increases.	Cloud platforms (Render, AWS, etc.), Nginx (optional)
5.	Performance	Efficient API calls with Axios, caching static content using CDN. MongoDB handles high-volume reads/writes efficiently.	Axios, MongoDB, CDN (e.g., Cloudflare), Compression

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