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

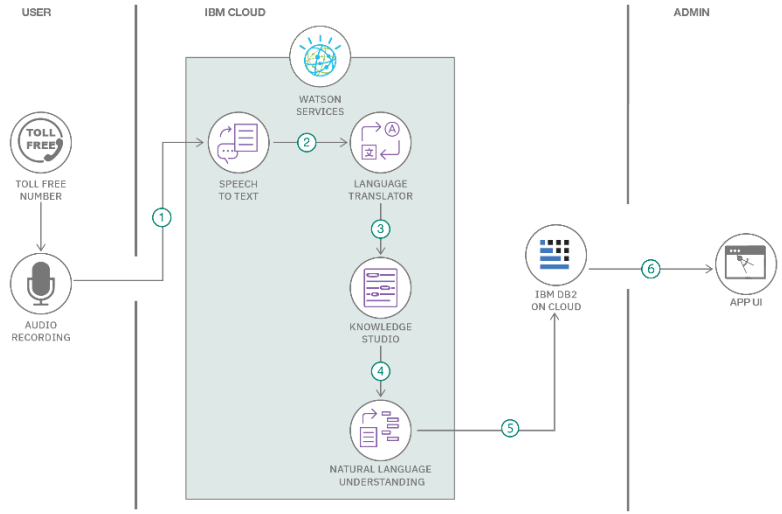
Date	24 June 3035
Team ID	LTVIP2025TMID56176
Project Name	FlightFinder: Navigating Your Air Travel
Maximum Marks	4 Marks

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. 1	**User Interface**	Web UI for users to search, filter, and book flights; responsive design for mobile and desktop.	HTML, CSS, JavaScript, React.js
2. 2	**Application Logic-1**	Backend logic: user registration, authentication, flight search, booking, payment processing.	Node.js, Express.js
3. 3	**Application Logic-2**	(Optional) Customer support chatbot integrated in the frontend.	Bot built with Node.js or third-party services like Dialogflow
4. 4	**Application Logic-3**	(Optional) Voice search or commands for accessibility.	Web Speech API or third-party services
5. 5	**Database**	Store users, flights, bookings, and transaction records.	MongoDB
6. 6	**Cloud Database**	Managed MongoDB instance for scalability & high availability.	MongoDB Atlas
7. 7	**File Storage**	Store e-tickets, invoices, user profile photos, etc.	AWS S3, Google Cloud Storage, or local filesystem
8. 8	**External API-1**	Real-time flight schedules, airline info.	Aviationstack API, Amadeus API
9. 9	**External API-2**	Payment processing.	Stripe API, PayPal API
10. 10	**Machine Learning Model**	(Optional enhancement) Recommend flights based on user history/preferences.	Node.js ML libraries (e.g., Brain.js) or call a Python microservice if needed
11. 11	**Infrastructure (Server / Cloud)**	Deploy Node.js backend and React frontend on cloud or containers.	Docker, Kubernetes, Nginx; deployment on AWS / Azure / GCP / Render / Vercel

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1. 1	**Open-Source Frameworks**	The application uses modern open-source frameworks for frontend and backend development, improving development speed and maintainability.	React.js (frontend), Flask (Python backend), SQLAlchemy (ORM), Bootstrap/Tailwind CSS

S.No	Characteristics	Description	Technology
2. 2	**Security Implementations**	Implements secure authentication (JWT tokens), password hashing, HTTPS encryption, role-based access control for admin and users, protection against OWASP top 10 vulnerabilities.	SHA-256 for password hashing, JWT for tokens, HTTPS with SSL/TLS, Flask-Security, IAM policies
3. 3	**Scalable Architecture**	The system is designed as a modular, loosely coupled architecture with separate frontend, backend API, and database layers — can be extended to microservices if traffic grows.	3-tier architecture: React frontend → Flask REST API → Database (PostgreSQL / MySQL)
4. 4	**Availability**	Ensures high availability using container orchestration, possible deployment on cloud servers with auto-scaling, and load balancing.	Docker, Nginx as load balancer, cloud deployment (e.g., AWS/GCP/Azure)
5. 5	**Performance**	Optimized API responses, caching frequently accessed data, minimized frontend bundle size, and potential use of CDN for static assets to handle large user traffic smoothly.	Redis (caching), CloudFront (CDN), React lazy loading, optimized SQL queries

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