**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

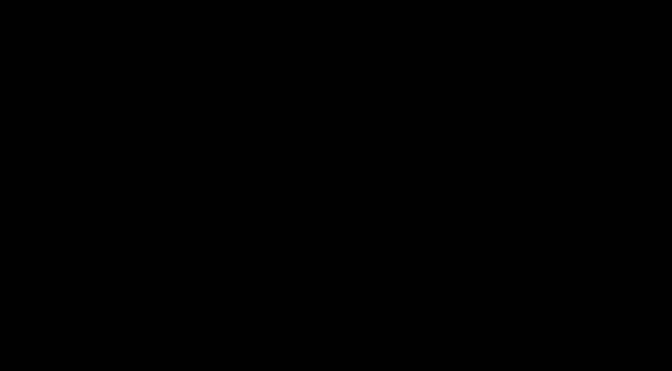
| Date | 31 May 2025 |
| --- | --- |
| Team ID | LTVIP2025TMID56096 |
| Project Name | Flightfider:Navigating your Air Travel Options |
| Maximum Marks | 4 Marks |

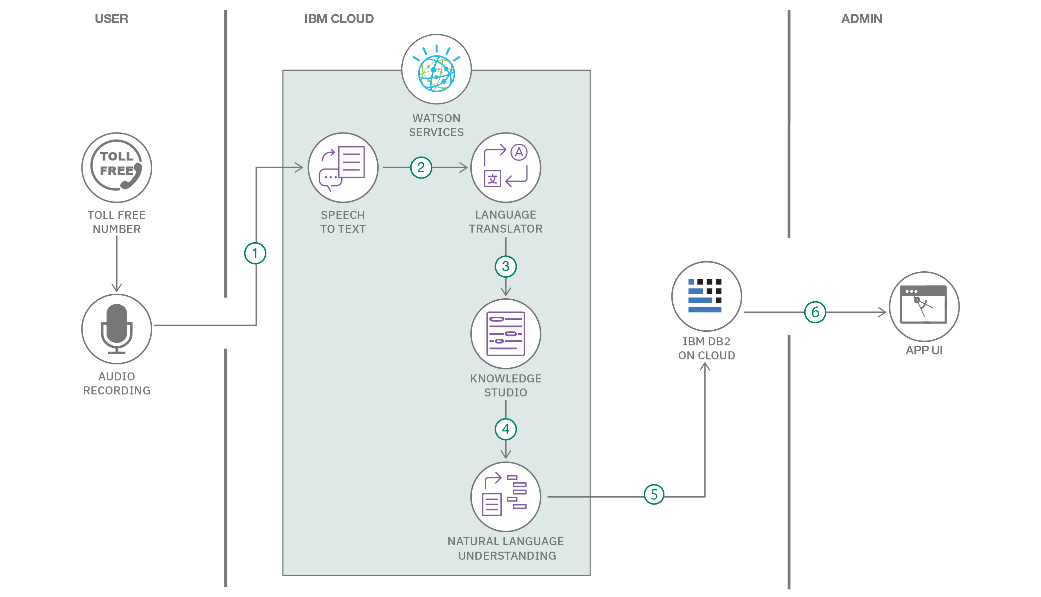
**Technical Architecture:**

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/**](https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/)





**Table-1 : Components & Technologies:**

| **S.No** | **Component** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | User Interface | Frontend pages for users to search, filter, and book flights | HTML, CSS, JavaScript / Angular Js / React Js |
|  | Search & Filter Engine | Enables users to search flights by location, date, class, airline, and price. | React.js state, MongoDB queries, |
|  | Flight Listings | Shows available flights with details, | React.js,CSS |
|  | Booking System | Handles passenger detail forms and booking confirmation | React Forms, Express.js, MongoDB |
|  | Authentication & Access | Provides login/signup for users and admins |  |
|  | Admin Dashboard | Allows admin to add/edit/delete flights and manage bookings | React.js, Express.js, MongoDB, JWT |
|  | Backend API | Handles all logic for bookings, users, flights, and filters | Node.js, Express.js |
|  | Database Layer | Stores flight data, bookings, and user/admin credentials | MongoDB, Mongoose |
|  | Version Control | Manages codebase and team collaboration | Git, GitHub |
|  | Deployment | Hosts the frontend, backend, and database for live access | Render, Netlify, MongoDB Atlas |
|  | Project Management Tools | Organizes tasks, designs, and communication during development | Trello, Notion, Figma, Google Docs/Meet |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Uses open-source tools like React.js, Node.js, Express.js, MongoDB, and CSS to ensure cost efficiency and community-driven support. | React.js, Node.js, Express.js, MongoDB, Tailwind CSS |
|  | Security Implementation | Implements JWT for authentication, bcrypt for password encryption, HTTPS for secure data transfer, and input validation to prevent common attacks. | JWT, bcrypt.js, Helmet.js, HTTPS, Express-validator |
|  | Scalable Architecture | Built using a modular MERN stack (MongoDB, Express, React, Node.js) with RESTful APIs, allowing easy scalability by adding microservices if needed. | MERN stack, RESTful APIs, Microservices |
|  | Availability | Deployed on cloud platforms like Render or Vercel with MongoDB Atlas for high uptime, ensuring 24/7 access and reliable performance | Render, Vercel, MongoDB Atlas |
|  | Performance | Designed with optimized queries, React’s virtual DOM, backend caching (if needed), and load balancing support for handling multiple requests fast. | React Virtual DOM, MongoDB Indexing, Node.js, Caching |

**References:**

[**https://c4model.com/**](https://c4model.com/)

[**https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/**](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)

[**https://www.ibm.com/cloud/architecture**](https://www.ibm.com/cloud/architecture)

[**https://aws.amazon.com/architecture**](https://aws.amazon.com/architecture)

[**https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d**](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)