**Requirement Gathering and Analysis Phase**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 3rd July 2024 |
| Team ID | SWTID1720194751 |
| Project Name | Project – Flight Booking App(Fly High) |
| 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 | How user interacts with the flight booking page | HTML, CSS, JavaScript, React Js |
|  | Search Functionality | Search for flights based on destination, date, price, etc. | JavaScript, React.js, Node.js |
|  | Filter and Sorting Logic | Filter and sort search results | JavaScript, React.js |
|  | Flight Details Display | Display flight options and details | HTML, CSS, JavaScript, React.js |
|  | Database | |  | | --- | | Store user, flight, and booking details |  |  | | --- | |  | | |  | | --- | |  |  |  | | --- | | MongoDB, NoSQL | |
|  | Cloud Database | Not using | Not using |
|  | File Storage | Local File storage | Local Filesystem |
|  | External API-1 | Not Using | Not using |
|  | Server-side Logic | Handle server-side logic for booking and user management | Node.js, Express.js |
|  | Machine Learning Model | Not using | Not using |
|  | Infrastructure (Server / Cloud) | Application Deployment on Local System | Local Server Configuration |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Frameworks utilized to develop the application, which are open-source and community-supported | Node.js, Express.js, React.js, Mongoose |
|  | Security Implementations | Security measures and access controls implemented to protect the application and user data | Password Protected (JavaScript and related technologies) |
|  | Scalable Architecture | Not using | Not using |
|  | Availability | Measures taken to ensure the application remains accessible and operational under various conditions | MongoDB |
|  | Performance | Enhances the performance, Considerations used to ensure the application performs well under load | Express.js |

**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)