**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

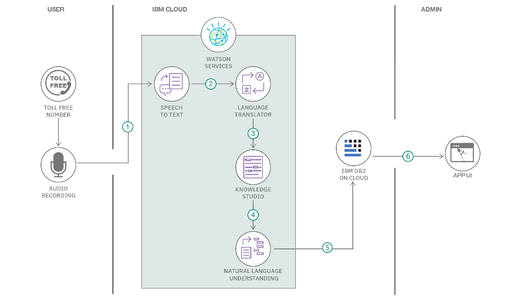
|  |  |
| --- | --- |
| Date | **27 June 2025** |
| Team ID | **LTVIP2025TMID60609** |
| Project Name | **DocSpot: Seamless Appointment Booking for Health.** |
| Maximum Marks | 4 Marks |

**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** |
| 1 | **User Interface** | Interface for users (patients, doctors, admin) to interact with system | HTML, CSS, JavaScript, React JS |
| 2 | **Application Logic-1** | Business logic for registration, login, booking, dashboard, etc. | Python (Flask) |
| 3 | **Application Logic-2** | Logic for teleconsultation through video call | WebRTC / Jitsi |
| 4 | **Application Logic-3** | Chatbot-based support for user queries and appointment help | IBM Watson Assistant / Dialogflow |
| 5 | **Database** | Data storage for doctors, users, appointments, and reviews | MongoDB (NoSQL) |
| 6 | **Cloud Database** | Cloud-hosted database for redundancy and scaling | MongoDB Atlas |
| 7 | **File Storage** | Storage for uploaded reports, prescriptions, profile images | Firebase Storage / AWS S3 / Local FS |
| 8 | **External API-1** | Email/OTP integration for confirmation and alerts | SendGrid / Twilio |
| 9 | **External API-2** | Identity verification and single sign-on | Google OAuth / Aadhar API (if applicable) |
| 10 | **Machine Learning Model** | Suggesting doctors based on user-entered symptoms | Custom ML model (Python, scikit-learn) |
| 11 | **Infrastructure** | Hosting the frontend, backend, DB, media | Cloud Platforms like Render, Railway, Netlify |

**Table-2: Application Characteristics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology** |
| 1 | **Open-Source Frameworks** | Used for both backend and frontend | Flask (Python), React JS, Bootstrap, Express |
| 2 | **Security Implementations** | Secure password hashing, role-based access, token-based auth | SHA-256 / bcrypt, JWT, OAuth2, HTTPS, Helmet.js, OWASP guidelines |
| 3 | **Scalable Architecture** | Supports growth using modular design (separate backend, frontend, DB) | 3-tier architecture (Frontend–Backend–Database) with optional microservices |
| 4 | **Availability** | Cloud-hosted services with support for auto-restart and horizontal scaling | Railway, MongoDB Atlas, Render, Netlify; Load balancing supported |
| 5 | **Performance** | Fast load times using caching, asynchronous APIs, and optimized assets | Redis (optional), CDN (Cloudflare), Lazy loading, Gzip compression |

**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)

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 / servicesIndicate interface to machine learning models (if applicable)