**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 25-05-2025 |
| Team ID | LTVIP2025TMID55946 |
| Project Name | Docspot |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

**Docspot :** This project aims to design and implement a seamless healthcare appointment booking system that enables patients to find healthcare providers, schedule in-person or telehealth appointments, and receive reminders, while also offering administrative tools for doctors and staff.

Designing a **technical architecture for a seamless appointment booking system for healthcare** involves integrating multiple components to ensure user convenience, data privacy (HIPAA/GDPR compliance), system scalability, and real-time availability. Below is a layered technical architecture, from frontend to backend, with optional integrations for scalability and intelligence.

1. **Frontend Layer (User Interface)**

**Purpose:** Interface for patients, doctors, and admins.

* **Web App** (React, Angular, Vue)
* **Mobile App** (Flutter, React Native, Swift/Kotlin)
* **Features:**
  + Appointment booking calendar
  + Doctor search & filters
  + Patient portal (appointments, records)
  + Notifications & reminders (push, SMS)

**2. Backend (Application Logic Layer)**

**Purpose:** Handles business logic and APIs.

* **Frameworks:** Node.js, Django, Spring Boot, Express
* **Core Services:**
  + **Appointment Service**: Schedules, modifies, cancels bookings.
  + **Availability Service**: Checks real-time doctor availability.
  + **User Management**: Patient/doctor/admin auth, profiles.
  + **Notification Service**: SMS, Email, Push (via Twilio, SendGrid, Firebase).
  + **Payment Service** (if applicable): Stripe, Razorpay, etc.
  + **Calendar Integration**: Google Calendar, Outlook sync.
* **API Layer:**
  + RESTful / GraphQL APIs
  + Token-based auth (JWT/OAuth2)

**3. Database Layer**

**Purpose:** Stores persistent data.

* **Relational DB:** PostgreSQL / MySQL for structured data (users, appointments)
* **NoSQL DB:** MongoDB for flexible data (logs, metadata)
* **Search Engine:** Elasticsearch for fast doctor/clinic search
* **Caching:** Redis for session management & availability caching

**4. Infrastructure Layer (Cloud/DevOps)**

**Purpose:** Ensures high availability and scalability.

* **Cloud Provider:** AWS / GCP / Azure
  + Compute: EC2 / App Engine / ECS
  + Load Balancing: ALB / NGINX / Cloud Load Balancer
  + Storage: S3 for documents/images
  + Secrets: AWS Secrets Manager / Vault
* **Containerization:** Docker + Kubernetes (EKS/GKE/AKS)
* **CI/CD Pipeline:** GitHub Actions, Jenkins, or GitLab CI
* **Monitoring:** Prometheus, Grafana, New Relic
* **Logging:** ELK Stack / CloudWatch

**5. Security & Compliance Layer**

**Purpose:** Ensures data protection & compliance.

* **Authentication/Authorization:** OAuth2, Role-Based Access Control
* **Data Encryption:** TLS for transit, AES for storage
* **Audit Logging:** Immutable logs for all sensitive actions
* **Compliance:** HIPAA (U.S.), GDPR (EU)
* **Rate Limiting / Throttling:** To avoid abuse

**6. Integration Layer**

**Purpose:** For connecting with external systems.

* **EHR/EMR Systems**: HL7/FHIR APIs (Epic, Cerner)
* **Insurance APIs**: Eligibility checks, pre-authorization
* **Third-party Calendar APIs**
* **Telemedicine API**: Zoom, Twilio Video, Jitsi

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | **User Interface (UI)** | Patient & Doctor App/Portal | React, Vue.js (Web)  Flutter, React Native, Swift/Kotlin (Mobile) |
|  | **API Gateway** | Entry point for client → backend requests | Kong, AWS API Gateway, NGINX, Apigee |
|  | **Authentication & Identity** | User login, role management, SSO | OAuth2.0, JWT, Auth0, Firebase Auth, Keycloak |
|  | **Appointment Scheduler** | Handles bookings, rescheduling, cancellations | Custom microservice (Node.js, Django, Spring Boot) |
|  | **Availability Management** | Syncs doctors' calendars and time slots | Redis (cache), Google Calendar API, Outlook Calendar API |
|  | **Notification System** | SMS, Email, In-app notifications | Twilio (SMS), SendGrid (Email), Firebase Cloud Messaging |
|  | **Patient & Doctor Profiles** | Stores user info, medical history, preferences | PostgreSQL, MySQL |
|  | **Calendar Sync Module** | Integrates external calendar systems (optional) | Google Calendar API, Microsoft Graph API |
|  | **Telehealth Module** | Virtual appointment capability (optional) | Twilio Video, Zoom SDK, Jitsi |

**Table-2: Application Characteristics:**

|  |  |  |
| --- | --- | --- |
| Category | Characteristic | Description |
| User Interface | Intuitive Navigation | Easy-to-use steps for booking, rescheduling, and cancellation. |
|  | Responsive Design | Works across mobile, tablet, and desktop devices. |
|  | Accessibility Features | Screen reader support, adjustable fonts, and multilingual options. |
| Scheduling | Real-Time Availability | View and book actual available slots instantly. |
|  | Calendar Sync | Syncs with doctors’ calendars to prevent double-booking. |
|  | Waitlist Management | Notifies patients if earlier appointments become available. |
| Access Channels | Multi-Platform Access | Supports web, mobile app, and clinic-based kiosks. |
|  | Telehealth Integration | Offers virtual visit options during booking. |
| Security | Secure Login | Account protection with password or health ID. |
|  | Two-Factor Authentication | Enhances login security. |
|  | Compliance | Meets HIPAA, GDPR, or other relevant regulations. |
| Health Records | EHR/EMR Integration | Syncs appointments with patient medical history and data. |
|  | Pre-Visit Forms | Allows symptom checklists or intake forms before visits. |
|  | Follow-Up Scheduling | Automated booking prompts based on prior visits. |
| Notifications | Appointment Reminders | Sent via SMS, email, or app notifications. |
|  | Instruction Alerts | Pre-visit reminders (e.g., fasting, medication). |