**Project Design Phase-II**

# 💻 Technology Stack — DocSpot

**📅 Date:** 27 June 2025  
**📌 Project Title:** *DocSpot — Seamless Appointment Booking for Health*  
**🆔 Team ID:** LTVIP2025TMID57021  
**🏆 Maximum Marks:** 4 Marks

## 🏗️ Technical Architecture Overview

DocSpot is a web-based platform built to streamline appointment scheduling for patients and doctors. The architecture includes a responsive web UI, a robust backend with REST APIs, secure user authentication, and cloud-based storage. The system is scalable, secure, and integrated with third-party APIs and notification services.

### 🔧 Table-1: Components & Technologies

| S.No | Component | Description | Technology |
| --- | --- | --- | --- |
| 1 | User Interface | Web-based UI for patient & doctor access | HTML5, CSS3, JavaScript, React.js |
| 2 | Application Logic-1 | Booking system logic | Node.js, Express.js |
| 3 | Application Logic-2 | Email/OTP confirmation service | NodeMailer, Twilio API |
| 4 | Application Logic-3 | Doctor schedule availability logic | Custom Python Logic |
| 5 | Database | Booking & user data | MongoDB (NoSQL) |
| 6 | Cloud Database | Cloud-hosted document database | MongoDB Atlas |
| 7 | File Storage | Profile photos, attachments | AWS S3 |
| 8 | External API-1 | OTP & SMS Notifications | Twilio API |
| 9 | External API-2 | Doctor verification using Aadhar API | UIDAI Aadhar API (Future Scope) |
| 10 | Machine Learning Model | Booking pattern prediction (optional scope) | scikit-learn / TensorFlow (TBD) |
| 11 | Infrastructure | Cloud-hosted app with scaling | AWS EC2, Load Balancer, Docker, Kubernetes |

### ⚙️ Table-2: Application Characteristics

| S.No | Characteristics | Description | Technology Used |
| --- | --- | --- | --- |
| 1 | Open-Source Frameworks | All core frameworks used are open source | React.js, Node.js, Express, MongoDB |
| 2 | Security Implementations | Authentication, role-based access, data encryption | JWT, HTTPS, bcrypt, OAuth 2.0 |
| 3 | Scalable Architecture | Microservices with containerized deployment | Docker, Kubernetes, AWS ECS |
| 4 | Availability | Highly available via distributed deployment | AWS Load Balancer, Multi-zone deployment |
| 5 | Performance | Fast response, async handling, caching for frequent calls | Redis, CDN (Cloudflare), Lazy Loading |