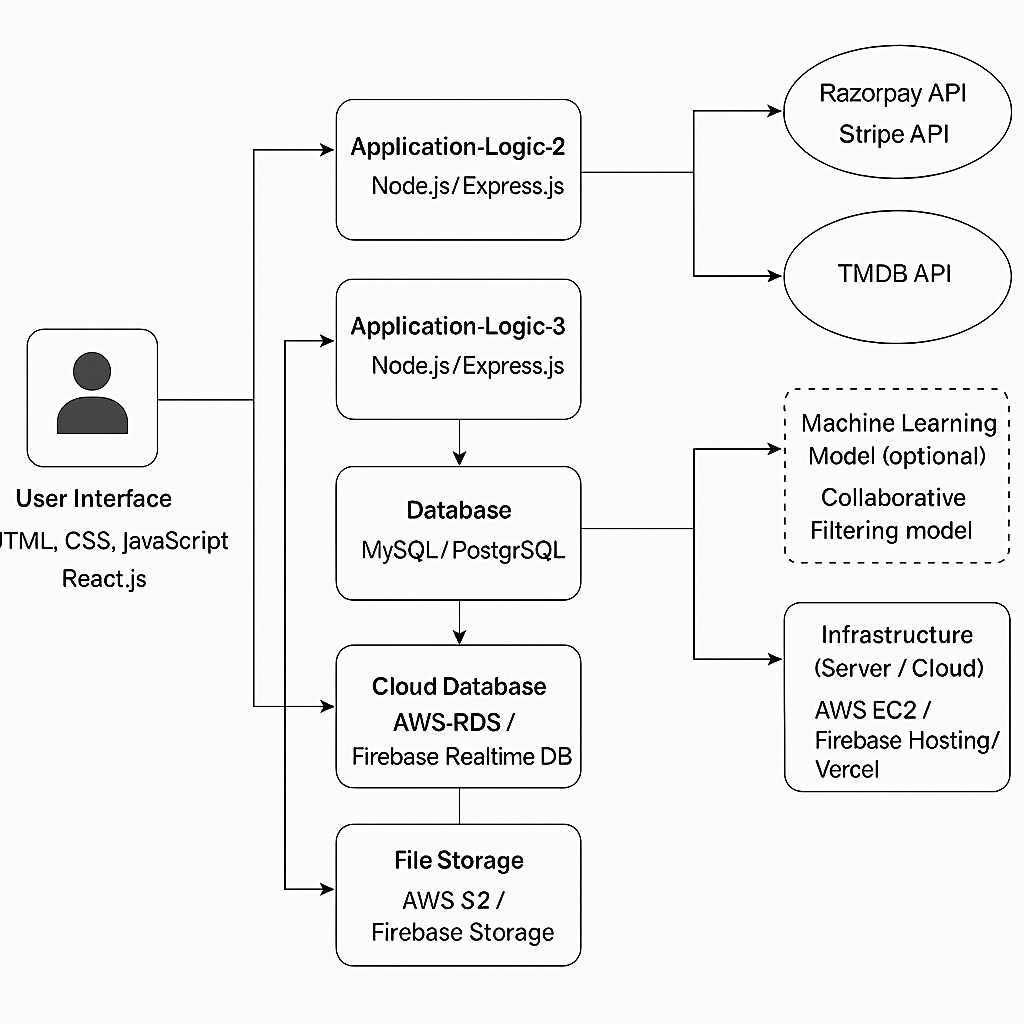
### **Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

| **Date** | **10 April 2025** |
| --- | --- |
| **Team ID** | SWTID1742572631 |
| **Project Name** | Movie Ticket Booking System |

### **Technical Architecture**

This section describes the components and technologies used in building the movie ticket booking system. It outlines the User Interface, Backend Logic, APIs, Database, Infrastructure, and optional ML model if included (like recommendation engine).



#### **Table-1: Components and Technologies**

| **S.No** | **Component** | **Description** | **Technology** |
| --- | --- | --- | --- |
| **1** | **User Interface** | How users interact with the application (Web/App) | **HTML, CSS, JavaScript, React.js** |
| **2** | **Application Logic-1** | Movie listing, filtering by genre, date/time | **Node.js / Express.js** |
| **3** | **Application Logic-2** | Booking process, payment integration, seat selection | **Node.js / Express.js** |
| **4** | **Application Logic-3** | Admin panel for theater management | **React.js, Express.js** |
| **5** | **Database** | Stores movie details, user profiles, bookings | **MongoDB** |
| **6** | **Cloud Database** | Cloud-based relational DBMS (if deployed on cloud) | **MongoDB Atlas** |
| **7** | **File Storage** | Poster images, trailers | **MongoDB cluster** |
| **8** | **External API-1** | Payment Gateway | **Razorpay API / Stripe API** |
| **9** | **External API-2** | Movie info source (optional) | **TMDB API** |
| **10** | **Machine Learning Model (optional)** | Recommend movies based on user history | **Collaborative Filtering model** |
| **11** | **Infrastructure (Server/Cloud)** | Application deployment and scaling infrastructure | **AWS EC2 / Firebase Hosting / Vercel** |

#### **Table-2: Application Characteristics**

| **S.No** | **Characteristics** | **Description** | **Technology Used** |
| --- | --- | --- | --- |
| **1** | **Open-Source Frameworks** | Frameworks used for front-end/back-end | **React.js, Express.js, Node.js** |
| **2** | **Security Implementations** | SSL, Payment security, Authentication (JWT), Role-based access | **JWT, HTTPS, OAuth, OWASP controls** |
| **3** | **Scalable Architecture** | Microservices for booking/payment, scalable DB | **Node.js microservices, AWS RDS** |
| **4** | **Availability** | Hosted on scalable cloud platforms with load balancers | **AWS EC2, Firebase, Load Balancer** |
| **5** | **Performance** | Caching, CDN, minimal API latency | **Redis (caching), Cloudflare (CDN)** |