**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 5 july 2025 |
| Team ID | LTVIP2025TMID49612 |
| Project Name | **BookNest: Where Stories Nestle** |
| Maximum Marks | 4 Marks |

**BookEase - Architectural Overview**

System Description: BookEase is a MERN-based role-based e-commerce web application that enables users to buy books, sellers to list/manage inventory, and admins to monitor system analytics. It supports user authentication, cart, wishlist, order management, review approvals, and analytics. Local deployment is supported with optional cloud integrations (e.g., AWS for storage or backups).

**Table-1: Components & Technologies**

| **S.No** | **Component** | **Description** | **Technology** |
| --- | --- | --- | --- |
| 1. | User Interface | Web interface for Customer, Seller, Admin | HTML, CSS, JavaScript, React.js |
| 2. | Application Logic-1 | Backend APIs for login/signup, orders, books, cart | Node.js, Express.js |
| 3. | Application Logic-2 | Book review sentiment analysis (optional enhancement) | Python (NLTK/TextBlob) (future scope) |
| 4. | Application Logic-3 | Chat-based order assistant (future enhancement) | Dialogflow / IBM Watson Assistant |
| 5. | Database | Stores users, books, orders, wishlist, reviews | MongoDB (NoSQL) |
| 6. | Cloud Database | For cloud deployment (optional) | MongoDB Atlas |
| 7. | File Storage | Book images and user avatars | Local file system / AWS S3 (optional) |
| 8. | External API-1 | Book metadata or payment gateway integration (future scope) | Razorpay API / Google Books API |
| 9. | External API-2 | Aadhar/PAN Verification API for seller KYC (optional enhancement) | UIDAI API / NSDL (optional) |
| 10. | Machine Learning Model | Recommendation engine (future scope) | Collaborative Filtering (Python) |
| 11. | Infrastructure | Application deployed locally with optional cloud migration | Node.js Server, Vite, MongoDB Local |

**Table-2: Application Characteristics**

| **S.No** | **Characteristics** | **Description** | **Technology Used** |
| --- | --- | --- | --- |
| 1. | Open-Source Frameworks | Full stack using open-source tech stack | React.js, Express.js, MongoDB, Node.js |
| 2. | Security Implementations | JWT Authentication, HTTPS, Password Hashing, Role-Based Access Control | bcrypt.js, JWT, Helmet.js, CORS |
| 3. | Scalable Architecture | Modular folder-based, loosely coupled backend APIs with frontend separation | 3-tier MERN Architecture |
| 4. | Availability | Load balancing via Nginx or PM2 cluster (future) | Nginx, PM2 (future scalable setup) |
| 5. | Performance | Cached API responses, paginated listings, compression, static delivery | Node.js compression, Mongoose lean(), Pagination, CDN for assets (optional) |