**Requirement Gathering and Analysis Phase**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 04/07/2024 |
| Team ID | SWTID1719997659 |
| Project Name | Book-Nest |
| Maximum Marks |  |

**Technical Architecture:**

The book store website should follow a standard web application architecture with a front-end, back-end, and database layer. A architecture could be beneficial for better scalability and modularity. The front-end could use React.js along with a CSS framework like Bootstrap. The back-end is built with Node.js (Express.js) using database MongoDB. We are using google books api for getting books info and razorpay api for payment which is sufficient to build the desired website

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | How user interacts with application e.g.  Web UI, Mobile App etc. | HTML, CSS, JavaScript, React Js etc. |
|  | Login/Signup Page | User authentication | Nodejs,Express.js |
|  | Orders and Cart Page | For fetching order and cart details | Nodejs,Express.js |
|  | Home Page | For fetching and displaying books | Nodejs,Express.js |
|  | Database | Data Type, Configurations etc. | MongoDB |
|  | External API-1 | Google books Api is used to fetch the book details | Google Books API |
|  | External API-2 | Razorpay Api is used for Payment | Razorpay Api |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Open-source frameworks used in the application ensure cost-effectiveness, community support, and flexibility in development | React JS, Node.js, Express.js, MongoDB |
|  | Security Implementations | Security and access controls implemented to protect user data and ensure secure transactions. This includes methods for encryption, secure communications, and user authentication | .JWT for authentication, bcrypt for hashing passwords, HTTPS for secure communication |
|  | Scalable Architecture | The application's scalability is supported by a 3-tier architecture, which separates the presentation, logic, and data storage layers. This modular design allows for easy scaling of individual components. | The application's scalability is supported by a 3-tier architecture, which separates the presentation, logic, and data storage layers. This modular design allows for easy scaling of individual components. |
|  | Availability | Availability of the application is ensured through server redundancy and load balancing, although in a basic setup, a single server deployment is used. | N/A for basic setup (Single server deployment) |
|  | Performance | Performance considerations include using non-blocking I/O operations in Node.js for handling multiple requests efficiently and MongoDB for quick data retrieval. | Node.js for non-blocking I/O operations, MongoDB for efficient data retrieval |