**Requirement Gathering and Analysis Phase**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 14July 24 |
| Team ID | SWTID1720068764 |
| Project Name | Book-Store |
| Maximum Marks | 3 Marks |

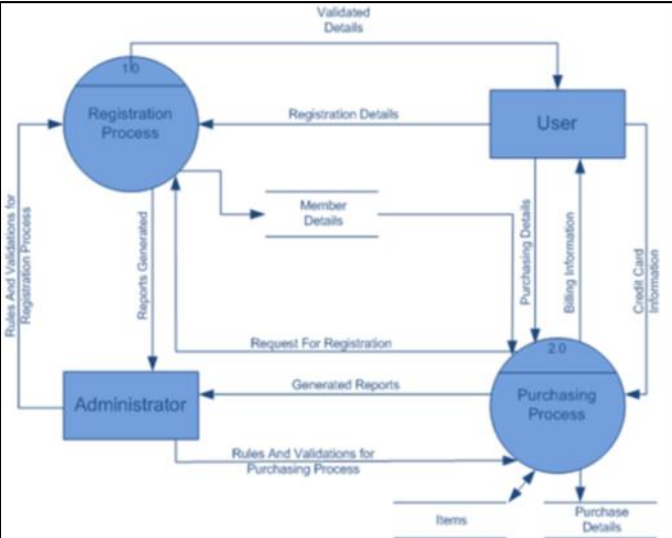
**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

**Example: Order processing during pandemics for offline mode**

**Reference:** [**https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/**](https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/)







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

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | How users interact with the application (Web UI) | HTML, CSS, JavaScript, React.js |
|  | Application Logic-1 | Logic for handling user registration and authentication processes | Node.js, Express.js |
|  | Application Logic-2 | Logic for personalized book recommendations | Node.js, Express.js |
|  | Application Logic-3 | Logic for processing book purchase transactions | Node.js, Express.js |
|  | Database | Data storage and management | MongoDB |
|  | Cloud Database | Database service on the cloud for scalability and reliability | MongoDB Atlas |
|  | File Storage | Storage for book images, user files, etc. | AWS S3 or local filesystem |
|  | External API-1 | API for payment processing | Stripe API, PayPal API |
|  | External API-2 | API for fetching book details from external sources | Google Books API |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
| 1 | Open-Source Frameworks | Utilize open-source frameworks for UI development and backend logic (MERN Stack). | React, Node.js Express.js, MongoDB |
| 2 | Security Implementations | List all security/access controls implemented | JWT for authentication, SHA-256 encryption, IAM controls |
| 3 | Scalable Architecture | Justify the scalability of architecture (e.g., micro-services) | Microservices architecture using Docker and Kubernetes |
| 4 | Availability | Justify the availability of application (e.g., load balancers) | AWS Elastic Load Balancing, distributed servers |
| 5 | Performance | Design considerations for application performance | Use of caching (Redis), CDN (CloudFront), optimized database queries |

**References:**

[**https://c4model.com/**](https://c4model.com/)

[**https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/**](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)

[**https://www.ibm.com/cloud/architecture**](https://www.ibm.com/cloud/architecture)

[**https://aws.amazon.com/architecture**](https://aws.amazon.com/architecture)

[**https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d**](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)