**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

| Date | 31 January 3035 |
| --- | --- |
| Team ID | SWTID1743516636 |
| Project Name | SB Food Ordering App |
| Maximum Marks | 4 Marks |

**TECHNOLOGY STACK -**

**Our food ordering website is developed using a modern MERN stack (MongoDB, Express.js, React, Node.js), providing a full-stack JavaScript solution for building a fast, scalable, and responsive web application. Here's a detailed overview of the technologies used:**

### Frontend:

### React.js – A powerful JavaScript library used for building interactive and component-based user interfaces. React helps us create a seamless and dynamic user experience with efficient state management and routing.

**Styled Components -** Used to style the application, ensure responsiveness, and maintain a clean, modern UI/UX.

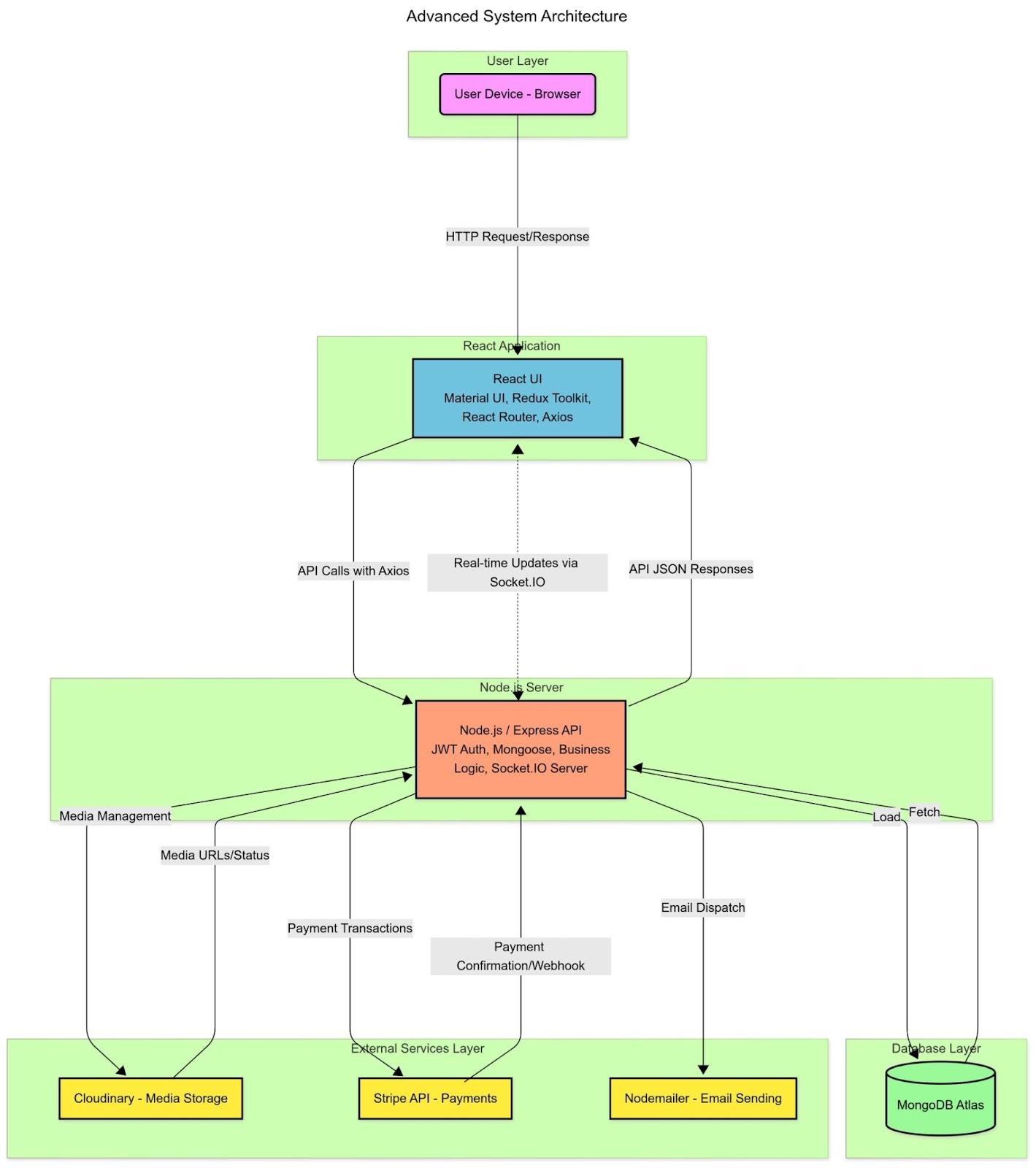
**Fetch API** -Enables smooth communication between the frontend and backend through RESTful API calls.

### Backend:

* **Node.js** – A JavaScript runtime used for building the backend server and handling asynchronous operations.
* **Express.js** – A minimalist and flexible Node.js framework used for creating RESTful APIs, routing, and middleware logic.

### 🔹 Database:

* **MongoDB** – A NoSQL database used to store collections of data such as users, food items, orders, and cart information in JSON-like documents.
* **Technical Architecture:**



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

| **S.No** | **Component** | **Description** | **Technology** |
| --- | --- | --- | --- |
| 1 | **User Interface** | Frontend Interface for Customers, Restaurants, and Admins to Interact with Menus, Orders, and Dashboards | HTML, CSS, JavaScript, React.js, Material UI |
| 2 | **Application Logic-1** | Handles Food Item Creation, Menu Management, and Restaurant Module Operations | React.js, Redux Toolkit |
| 3 | **Application Logic-2** | Manages Customer Authentication, Order Placement, Payment Validation, and JWT-Based Session Handling | Node.js, Express.js, JWT Authentication |
| 4 | **Application Logic-3** | Generates Analytics and Reports for Admins and Restaurants (Sales Charts, Order Stats, Customer Insights) | Chart.js, React-Chartjs-2, Node.js |
| 5 | **Database** | Stores user accounts, course metadata, progress data, and transaction logs. | MongoDB Atlas |
| 6 | **Cloud Database** | Cloud-hosted NoSQL database for scalability and distributed storage. | MongoDB Atlas |
| 7 | **File Storage** | Securely Stores Uploaded Assets like Food Images, Restaurant Logos, and Menu Files | Cloudinary Storage |
| 8 | External API-1 | Handles transactional email notifications (password reset, payment receipts, etc.). | Nodemailer, SMTP |
| 9 | **External API-2** | Integrates with payment services and automation tools (payments, receipts, notifications). | Stripe API, Webhooks |
| 10 | **Machine Learning Model** | Personalized Food Recommendations Based on Customer Behavior and Order History | Python, Scikit-learn, TensorFlow |
| 11 | **Infrastructure** | Cloud-hosted deployment of backend services with containerization and secure scaling. | Docker, Kubernetes, IBM Cloud / DigitalOcean / AWS |

Table -2: Application Characteristics

| **S.No** | **Characteristics** | **Description** | **Technology Used** |
| --- | --- | --- | --- |
| 1 | **Open-Source Frameworks** | The platform uses open-source, community-backed frameworks for creating responsive UIs and robust backend services. | React.js / Node.js / Express.js / Chart.js / MongoDB |
| 2 | **Security Implementations** | Security is ensured through HTTPS, JWT-based authentication, SSL encryption, role-based access, and secure Stripe payment handling. | SSL, HTTPS, JWT Tokens, Stripe API, bcrypt hashing |
| 3 | **Scalable Architecture** | Designed using a microservices-friendly and modular 3-tier architecture (Presentation, Logic, Data Layer) for easy scaling and maintenance. | Node.js (Backend), React.js (Frontend), MongoDB (Database), Docker |
| 4 | **Availability** | Cloud deployment with distributed architecture, auto-scaling, and load balancers ensure 99.9% uptime and fault tolerance. | DigitalOcean / AWS / IBM Cloud, Docker, Kubernetes |
| 5 | **Performance** | Optimized for fast response times using caching, Content Delivery Network (CDN) for static files, and efficient MongoDB indexing. | Cloudflare (CDN), Redis (Caching), MongoDB Indexing |