**Project Design Phase**

**Solution Architecture**

|  |  |
| --- | --- |
| Date | 25 June 2025 |
| Team ID | LTVIP2025TMID59612 |
| Project Name | SB Foods - On-Demand Food Ordering Platform |
| Maximum Marks | 4 Marks |

**Solution Architecture:**

**Solution Architecture** bridges the gap between business needs and technical implementation. For the SB Foods application, the goal is to ensure a **scalable, responsive, and secure full-stack food ordering system** using the **MERN stack** (MongoDB, Express.js, React.js, Node.js), with clear data flow and modular components for users, restaurants, and administrators.

**Objectives of the Architecture:**

* Define how the application components interact across the tech stack.
* Ensure clear separation of concerns: UI, business logic, data storage.
* Enable smooth data flow between client, server, and database.
* Support future scaling (e.g., more users, multi-region deployment).
* Allow role-based access (User, Restaurant, Admin).

**Architecture Layers and Components:**

|  |  |  |
| --- | --- | --- |
| **Layer** | **Components / Tools Used** | **Description** |
| **Frontend** | React.js, HTML/CSS, Axios | User interface for all roles (User, Restaurant, Admin). Components include Login, Cart, Orders, Dashboard, etc. |
| **Backend** | Node.js, Express.js | RESTful API server that handles routing, authentication, product management, order processing. |
| **Database** | MongoDB (with Mongoose ODM) | Stores all structured data: Users, Restaurants, Products, Orders, Carts, Admin data. |
| **Authentication** | JWT (JSON Web Tokens) + bcrypt | Secure login for users, restaurants, and admin roles. |
| **Hosting** | Localhost (Dev) / Future: Vercel/Heroku/Render | Deployment of frontend and backend in cloud or container environments. |
| **Dev Tools** | Git, VS Code, MongoDB Atlas, Postman | Development and testing tools used for building and verifying the application. |

**Data Flow Summary**

**1. User Journey:**

* User registers/login → token stored → fetches food listings → adds to cart → places order → order stored in DB.

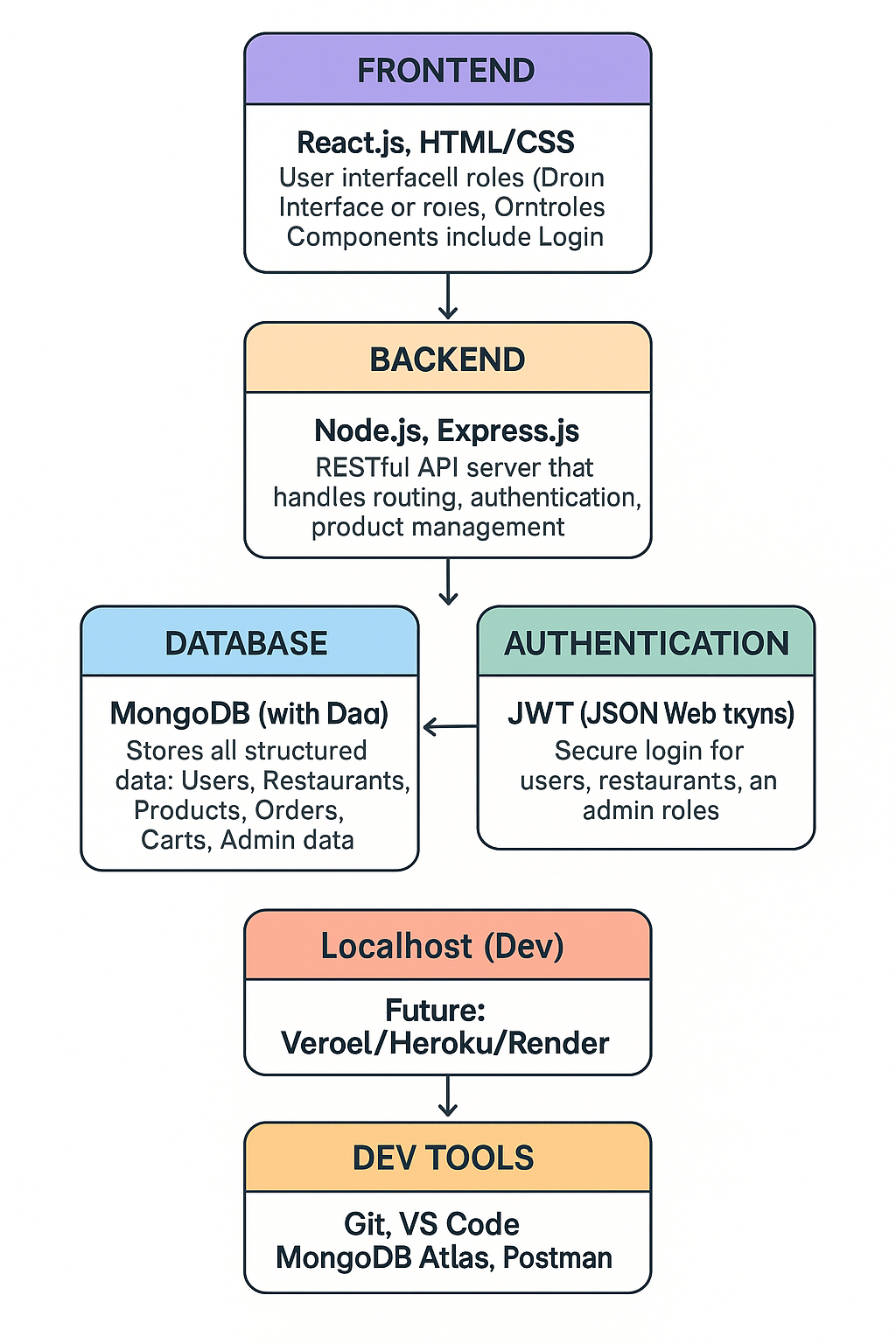
**2. Restaurant Journey:**

* Logs in → lists or edits products → products stored in DB → can view orders made on their items.

**3. Admin Journey:**

* Logs in → views all users/products/orders → approves restaurants/products → manages categories/promotions.

**Example - Solution Architecture Diagram:**

****

*Figure 1: Architecture and data flow of the food ordering sample application*

**Reference:** [**https://aws.amazon.com/blogs/industries/voice-applications-in-clinical-research-powered-by-ai-on-aws-part-1-architecture-and-design-considerations/**](https://aws.amazon.com/blogs/industries/voice-applications-in-clinical-research-powered-by-ai-on-aws-part-1-architecture-and-design-considerations/)

**Features Defined in Architecture**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Handled By** | **Status** |
| User Authentication | JWT, bcrypt, Express.js | ✔ Implemented |
| Product Listings | MongoDB, React, Axios | ✔ Implemented |
| Cart & Order Management | React, Express, MongoDB | ✔ Implemented |
| Role-Based Access Control | JWT Middleware | ✔ Implemented |
| Admin Approval Flow | Admin Panel, DB updates | ✔ Implemented |
| API Layer Security | Auth middleware, CORS | ✔ Implemented |
| Scalability Provision | Modular Code, REST APIs | Scalable |
| Database Optimization | Mongoose ODM | ✔ Optimized |

**Notes**

* The current design supports future enhancements like third-party payment integration, push notifications, and analytics.
* With MongoDB Atlas, the platform can be scaled vertically and horizontally based on usage.
* Role-based segregation ensures minimal coupling and better maintainability.