|  |
| --- |
| **Team ID :** LTVIP2025TMID54810 |
| **Project Name :** ShopSmart: Your Digital Grocery Store Experience |

**Backend Development**

Backend development involves building the server-side components and logic of the online shopping web application. It focuses on handling the business logic, processing requests from the front end, and interacting with the database. The following activities are part of the backend development process:  


**Set Up Backend**

**Set Up Project Structure:**

* Create a new directory for your project and set up a package.json file using npm init command.
* Install necessary dependencies such as Express.js, Mongoose, and other required packages.

**Database Configuration:**

* Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.
* Create a database and define the necessary collections for hotels, users, bookings, and other relevant data.

**Create Express.js Server:**

* Set up an Express.js server to handle HTTP requests and serve API endpoints.
* Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.

**Define API Routes:**

* Create separate route files for different API functionalities such as hotels, users, bookings, and authentication.
* Define the necessary routes for listing hotels, handling user registration and login, managing bookings, etc.
* Implement route handlers using Express.js to handle requests and interact with the database.

**Implement Data Models:**

* Define Mongoose schemas for the different data entities like hotels, users, and bookings.
* Create corresponding Mongoose models to interact with the MongoDB database.
* Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

**API Design and Development:**

* Identify the necessary functionality and data required by the frontend.
* Design a set of RESTful APIs using a framework like Express.js or Django REST Framework.
* Define API endpoints for user management, product catalog, shopping cart, order management, payment gateway integration, shipping integration, etc.
* Implement the API routes, controllers, and data models to handle the corresponding operations.
* Ensure that the APIs follow best practices, are secure, and provide appropriate responses.

**User Management and Authentication:**

* Implement user registration and login functionality.
* Choose an authentication mechanism like session-based authentication or token-based authentication (e.g., JWT).
* Store and hash user credentials securely.
* Implement middleware to authenticate API requests and authorize access to protected routes.

**Product Catalog and Inventory Management:**

* Design the database schema to store product details, pricing, availability, and inventory levels.
* Create APIs to retrieve product information, update inventory quantities, and handle search and filtering.
* Implement validations to ensure data integrity and consistency.

**Shopping Cart and Order Management:**

* Design the database schema to store shopping cart details and order information.
* Create APIs to handle cart operations like adding items, modifying quantities, and placing orders.
* Implement logic to calculate totals, apply discounts, and manage the order lifecycle.

**Payment Gateway Integration:**

* Choose a suitable payment gateway provider (e.g., Stripe, COD).
* Integrate the payment gateway SDK or API to handle secure payment processing.
* Implement APIs or callback endpoints to initiate transactions, handle payment callbacks, and receive payment confirmation.

**Shipping and Logistics Integration:**

* Identify shipping and logistics providers that align with your application's requirements.
* Utilize the APIs provided by these providers to calculate shipping costs, generate shipping labels, and track shipments.
* Implement APIs or services to fetch rates, generate labels, and obtain tracking information.

**Database Integration:**

* Choose a suitable database technology (e.g., MySQL, PostgreSQL, MongoDB) based on your application's requirements.
* Design the database schema to efficiently store and retrieve flower and gift delivery data.
* Establish a connection to the database and handle data persistence and retrieval.

**External Service Integration:**

* Identify third-party services like email service providers, analytics services, or CRM systems that are required for your application.
* Utilize the APIs or SDKs provided by these services to exchange data and perform necessary operations.
* Implement the integration logic to send order confirmations, track user behavior, or manage customer relationships.

**Security and Data Protection:**

* Apply appropriate security measures like encryption techniques for secure data transmission and storage.
* Implement input validation and sanitization to prevent common security vulnerabilities.
* Implement access control to ensure authorized access to sensitive data.

**Error Handling and Logging:**

* Implement error handling mechanisms to handle exceptions and provide meaningful error messages to the frontend.
* Use logging frameworks to record application logs for monitoring and troubleshooting purposes.

 Connect database to backend:  
              