Grocery Delivery App

Technical Design Document

	Prepared By / Last Updated By	Reviewed By	Approved By
Name			
Role			
Signature			
Date			

Table of Contents

1. Project Overview:

The "Grocery Delivery App" project aims to create a convenient mobile application that allows users to order groceries online and have them delivered to their doorstep. This includes building a microservices backend for order processing and a React Native frontend for the mobile app. Trainees will gain practical experience in building e-commerce apps, microservices architecture, and mobile app development.

2. Project Duration and Scope:

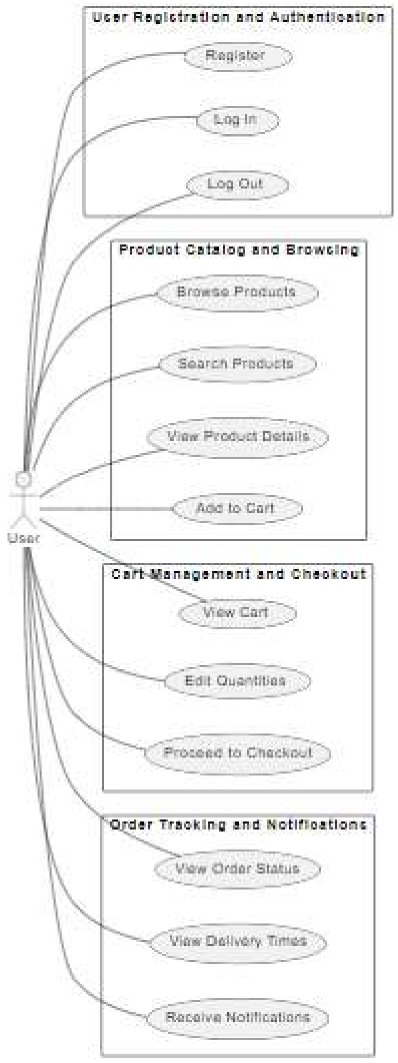
The project spans 80 hours over two weeks. It involves designing and implementing a microservices architecture for product catalog, cart management, and order processing. The backend will be developed using Java/.NET/Node.js and Express, while the mobile app frontend will use the React Native framework. Participants will create an intuitive and user-friendly mobile app for grocery shopping.

3. Business Problem:

In today's fast-paced world, users often seek convenient ways to order groceries. This project addresses the need for a mobile app that simplifies the process of ordering and receiving groceries.

4. Project Requirements:

4.1. Functional Requirements:



User Story 1: User Registration and Authentication

- As a user, I want to register and log in to the app to browse and order groceries.

  Acceptance Criteria:

  - Users should be able to create accounts with email and password.
  - Users should be able to log in and log out securely.

User Story 2: Product Catalog and Browsing

- As a user, I want to browse grocery products, view details, and add them to my cart.

  Acceptance Criteria:

- Users should be able to browse and search grocery products by categories and keywords.
- Users should be able to view product details, including images, descriptions, and prices.
- Users should be able to add products to their shopping cart.

User Story 3: Cart Management and Checkout

- As a user, I want to review my cart, edit quantities, and proceed to checkout.

  Acceptance Criteria:

  - Users should be able to view the contents of their cart and edit quantities.
  - Users should be able to proceed to checkout and provide delivery and payment information.

User Story 4: Order Tracking and Notifications

- As a user, I want to track the status of my orders and receive notifications.

  Acceptance Criteria:

- Users should be able to view the status of their orders and estimated delivery times.
- Users should receive notifications when their orders are confirmed, out for delivery, and delivered.

4.2. Non-functional Requirements:

- **Security:** Implement secure authentication and payment processing.
- **Performance:** Optimize app performance for smooth browsing, cart management, and order tracking.
- **User Experience:** Design an attractive and user-friendly UI for shopping and navigation.

5. Data Model / Entity Description:

5.1. User Entity:

Attributes:

- UserID (Primary Key)
- FirstName
- LastName
- Email
- Password (Hashed)

5.2. ProductEntity Entity:

Attributes:

- ProductID (Primary Key)
- ProductName

- Category
- Description
- Price
- Images (Array of image URLs)

5.3. CartItem Entity:

Attributes:

- CartItemID (Primary Key)
- UserID (Foreign Key)
- ProductID (Foreign Key)
- Quantity

5.4. OrderEntity Entity:

Attributes:

- OrderID (Primary Key)
- UserID (Foreign Key)
- TotalAmount
- OrderStatus
- EstimatedDeliveryTime

6. Architecture Design Guidelines:

- **Microservices:** Design independent, single-responsibility Microservices.
- **Communication:** Implement RESTful APIs for inter-microservice communication.
- **Database:** Utilize separate databases for each microservice's data storage.
- **Deployment:** Deploy microservices individually in containers for scalability.

7. Technology Stack:

Backend (Java)	
Programming Language	Core Java 12
Framework	Spring Boot
Database	MySQL
Authentication	JWT
Backend (.NET)	
Programming Language	C#
Framework	ASP.NET Core Web API
Database	SQL Server
Authentication	JWT
Backend (Node.js)	
Programming Language	Node.js
Framework	Express
Database	MongoDB
Authentication	JWT-based authentication
Frontend	
Framework	Choose React Native for frontend development.

UI Components	Develop mobile app UI components for product browsing, cart management, and order tracking.
Communication	Utilize REST APIs to interact with the backend microservice.

8. Evaluation Criteria:

- Successful implementation of microservices with inter-service communication.
- Smooth grocery product browsing, cart management, and order tracking.
- Effective communication between React Native frontend and microservices.
- Security measures for microservices communication, user authentication, and payment processing.
- User-friendly mobile app UI design, responsiveness, and navigation.
- Real-time order tracking and notifications.
- Code quality, documentation, and error handling.
- Project presentation and demonstration.

9. Deliverables:

- Source code for the microservices and React Native-based mobile app.
- Comprehensive API documentation detailing endpoints, request-response formats, and authentication mechanisms.
- Unit tests with sufficient code coverage for microservices.
- Deployment instructions for both microservices and mobile app components.
- Project summary report discussing challenges and solutions.

10. Timeline:

- Days 1-2: Project setup, technology selection, and architecture design.
- Days 3-5: Backend microservice development and API implementation.
- Days 6-9: Frontend UI development and UI component implementation.
- Days 10: Integration of frontend and backend components.

11. Resources:

Backend(Java)	Core Java	https://www.geeksforgeeks.org/java/
	Spring Boot Microservices	https://www.geeksforgeeks.org/java-spring-boot-microservices-example-step-by-step-guide/
	Data JPA	https://spring.io/guides/gs/accessing-data-jpa/
	Unit Testing	https://www.springboottutorial.com/unit-testing-for-spring-boot-rest-services
Backend(.NET)	C#	https://www.geeksforgeeks.org/csharp-programming-language/

	ASP.NET Core Microservices	https://www.c-sharpcorner.com/article/microservice-using-asp-net-core/ https://learn.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/data-driven-crud-microservice
	Entity Framework Core	https://www.tektutorialshub.com/entity-framework-core-tutorial/
	Unit Testing	https://learn.microsoft.com/en-us/aspnet/core/mvc/controllers/testing?view=aspnetcore-3.1
Backend (Node.js)	Node.js	https://nodejs.dev/en/learn/
	Express	https://www.geeksforgeeks.org/express-js/
	Mongo DB	https://www.mongodb.com/docs/manual/tutorial/
	create a REST API with Node.js and Express	https://blog.postman.com/how-to-create-a-rest-api-with-node-js-and-express/
Frontend (React Native)	React Native	https://reactnative.dev/

12. Support and Communication:

- Regular progress updates through daily stand-up meetings.
- Communication and assistance available through designated communication channels.

13. Change Log

Version Number	**Changes made**			
V<n.n>	*<If the change details are not explicitly documented in the table below, reference should be provided here>*			
	Page no	Changed by	Effective date	Changes effected