**Thato Sebelemetja (ST10067544)**

**Nakeisha Naidoo (ST10054051)**

**Jashil Roopnarain (ST10092261)**

**Malik Fahad Mannan (ST10091422)**

**Table of Contents**

[**1. INTRODUCTION TO PROJECT** 4](#_Toc183097072)

[**1.1 Ethical and Privacy Concerns** 4](#_Toc183097073)

[**1.2 Work Agreement** 5](#_Toc183097074)

[**1.3. Definition of Ready (DoR)** 6](#_Toc183097075)

[**1.4. Definition of Done (DoD)** 6](#_Toc183097076)

[**1.5. Roadmap (High-Level Plan)** 7](#_Toc183097077)

[**2. REQUIREMENTS** 8](#_Toc183097078)

[**2.1 User Roles** 8](#_Toc183097079)

[**2.2 User Stories** 8](#_Toc183097080)

[**2.3 User Experience Journey Map** 9](#_Toc183097081)

[**3. NON-FUNCTIONAL REQUIREMENTS** 11](#_Toc183097082)

[**4. ANALYSIS ARTIFACTS** 14](#_Toc183097083)

[**5. SOFTWARE IMPLEMENTATION** 18](#_Toc183097084)

[**5.1 UML Object Diagram** 18](#_Toc183097085)

[**5.2. UML Sequence Diagram** 20](#_Toc183097086)

[**5.3. UML State Diagram** 20](#_Toc183097087)

[**6. DATA SCHEMA DOCUMENTATION** 22](#_Toc183097088)

[**6.1. Data Storage Technologies and Choices** 22](#_Toc183097089)

[**6.2. Entity Relationship Diagram (ERD)** 22](#_Toc183097090)

[**6.3. JSON Schemas for Document Store (Firestore)** 23](#_Toc183097091)

[**7. ARCHITECTURE ARTIFACTS** 26](#_Toc183097092)

[**7.1. Design patterns** 26](#_Toc183097093)

[**7.2. Architecture patterns** 26](#_Toc183097094)

[**7.3 Cloud** 26](#_Toc183097095)

[**7.3.1. Cloud Architecture Overview** 26](#_Toc183097096)

[**7.3.2. Cloud Architecture Decisions** 27](#_Toc183097097)

[**7.3.3. Cloud Networking** 27](#_Toc183097098)

[**7.3.4. Cloud Security** 28](#_Toc183097099)

[**8. SECURITY** 29](#_Toc183097100)

[**8.1. Potential Threat Actors Documented** 29](#_Toc183097101)

[**8.2. Potential Threat Vectors Documented** 30](#_Toc183097102)

[**8.3. Mitigations for Threats Documented** 30](#_Toc183097103)

[**8.4. Balancing Security with Usability** 31](#_Toc183097104)

[**8.5. Complete Mediation and Security for Data Access** 32](#_Toc183097105)

[**9. DEVOPS** 33](#_Toc183097106)

[**9.1 GitHub Actions Pipeline** 33](#_Toc183097107)

[**10. RUNNING COSTS** 35](#_Toc183097108)

[**10.1. Predicted User Growth Documentation** 35](#_Toc183097109)

[**10.2. Scaling Points of Firebase Services** 35](#_Toc183097110)

[**10.3. Predictive Models for User Growth** 36](#_Toc183097111)

[**10.4. Adopting Alternative Technologies at Scale** 37](#_Toc183097112)

[**11. CHANGE MANAGEMENT** 38](#_Toc183097113)

[**11.1. Organizational Adoption** 38](#_Toc183097114)

[**11.2. User Adoption** 39](#_Toc183097115)

[**11.3. Adoption Strategy** 39](#_Toc183097116)

[**11.4. Maintaining and Supporting the Software** 40](#_Toc183097117)

[**REFERENCE LIST** 42](#_Toc183097118)

# **1. INTRODUCTION TO PROJECT**

The project aims to develop a delivery app for FARRARMERE EXPRESS MART, a local supermarket/convenience store. The company currently operates with a traditional, manual approach to order processing, delivery management, and inventory control. The lack of a comprehensive digital system limits efficiency, customer convenience, and the ability to scale operations in response to growing demand for online shopping and delivery services. Implementing a dedicated delivery app would modernize these operations, streamline workflows, and enhance both customer satisfaction and business performance. The app will provide a user-friendly platform for customers to order groceries, track deliveries, and manage their accounts while ensuring operational efficiency for the store.

# **1.1 Ethical and Privacy Concerns**

1. **Data Privacy:**

* **Customer Information Security:** The app will collect sensitive customer information, including names, addresses, payment details, and shopping preferences. Safeguarding this data from unauthorized access and breaches is crucial to maintaining customer trust and complying with data protection regulations (termly, 2023).
* Transparency in Data Use: Customers should be informed about how their data will be used, stored, and shared. Clear and accessible privacy policies must be established to ensure customers understand their rights regarding their personal information (termly, 2023).

1. **Consent and User Rights:**

* **Informed Consent:** Obtaining explicit consent from users before collecting personal data is essential. Users should have the option to opt in or out of data collection practices, especially regarding tracking and marketing communications (smartlook, 2023).
* **User Control Over Data:** Customers should have the ability to view, edit, and delete their personal information stored in the app. Providing users with control over their data enhances trust and accountability (smartlook, 2023).

1. **Data Sharing and Third-Party Access:**

* **Third-Party Collaborations:** If the app collaborates with third-party services (e.g., payment processors, delivery services), it is vital to ensure that these partners uphold similar ethical standards regarding data privacy and security (medium, 2023).
* Minimizing Data Sharing: The app should only share customer data with third parties when absolutely necessary and should seek customer consent before doing so (medium, 2023).

# **1.2 Work Agreement**

The development team for the FARRARMERE EXPRESS MART delivery app consists of four members, each bringing their unique skills and expertise to the project. This agreement outlines the roles, responsibilities, and collaboration guidelines for the team to ensure effective communication, task management, and successful project delivery.

**1. Team Members and Roles**

* **Project Manager (Thato Sebelemetja):** Responsible for overseeing the overall progress of the project, facilitating communication between team members, and ensuring that deadlines and project milestones are met. The project manager will also coordinate sprint planning and review meetings.
* **Backend Developers (Malik Mannan and Jashil Roopnarain):** In charge of developing the server-side logic, API integration, and database management. These members will ensure that all backend functionalities, such as order processing, user authentication, and inventory synchronization, are implemented and optimized.
* **Frontend Developer (Nakeisha Naidoo):** Responsible for the development of the mobile app's user interface and ensuring a seamless user experience. This member will work on the design and implementation of the app's key features, such as browsing products, placing orders, and managing deliveries.

**2. Collaboration and Communication**

* **Bi-Weekly Meetings:** The team will hold bi-weekly meetings to discuss progress, address any blockers, and plan upcoming tasks. These meetings will be led by the project manager and will follow a structured agenda, including a review of the previous week's sprint and setting goals for the upcoming sprint.
* **Communication Channels:** The primary communication tool will be Microsoft Teams. Urgent matters will be communicated WhatsApp messaging. All team members are expected to respond within a reasonable timeframe, typically within 24 hours for non-urgent matters.

# **1.3. Definition of Ready (DoR)**

**Criteria for Readiness**:

* **Clear Requirements:** User stories and acceptance criteria must be well-defined.
* **Acceptance Criteria:** Each task or user story must have clear and testable acceptance criteria.
* **Dependencies:** Any dependencies or external factors must be identified and addressed.
* **Designs and Prototypes:** Necessary designs, mockups, or prototypes should be provided.
* **Resources:** Required resources, such as APIs or databases, must be accessible.

# **1.4. Definition of Done (DoD)**

**Completion Criteria**:

* **Acceptance Criteria Met:** All acceptance criteria for the user story or task must be met.
* **Code Review:** Code must be reviewed and approved by peers.
* **Testing:** The feature must pass unit tests, integration tests, and user acceptance testing.
* **Documentation:** Relevant documentation must be updated, including user guides and API documentation.
* **Deployment:** The feature must be deployed to the staging environment and verified before production release.
* **Bug-Free:** No critical bugs or issues should be present.

# **1.5. Roadmap (High-Level Plan)**

* **Project Plan:** The project will be divided into 2-week sprints. Key milestones include the completion of the initial design, development of core features, beta testing, and final deployment.
* **Sprint Planning:** Sprints are planned based on the availability of team members. Adjustments will be made as necessary to accommodate changes in availability or scope.

# **2. REQUIREMENTS**

* **Requirement Artifacts**: This includes requirement documents detailing user needs, feature specifications, and technical constraints. Research involved understanding the current supermarket operations, customer expectations, and industry best practices.

# **2.1 User Roles**

* **Customer:** Users who place orders, track deliveries, and manage their accounts.
* **Delivery Personnel:** Individuals responsible for delivering orders to customers.
* **Store Staff:** Employees who manage inventory, process orders, and oversee delivery operations.

# **2.2 User Stories**

**Customer Stories:**

* As a customer who wants to order groceries, I want to browse the product catalog so that I can select and order the items I need.
* As a customer who wants to track my order, I want to receive real-time updates on the delivery status so that I know when to expect my groceries.
* As a customer who wants to manage my account, I want to update my personal information and payment methods so that I can ensure my account details are current.

**Delivery Personnel Stories:**

* As a delivery person who wants to view my delivery schedule, I want to see a list of orders assigned to me so that I can plan my delivery route efficiently.
* As a delivery person who wants to confirm deliverird, I want to mark orders as delivered in the app so that the store and customer are notified of successful delivery.

**Store Staff Stories:**

* As a store staff member who wants to manage inventory, I want to update stock levels in the app so that the availability of products is accurately reflected to customers.
* As a store staff member who wants to process orders I want to view and manage incoming orders so that I can ensure timely delivery.

# **2.3 User Experience Journey Map**

**Customer Journey**:

* 1. **Login/Sign Up:** Customer creates an account or logs in.
  2. **Browse Products:** Customer views product categories and selects items.
  3. **Place Order:** Customer adds items to the cart and proceeds to checkout.
  4. **Payment:** Customer provides payment information and confirms the order.
  5. **Order Tracking:** Customer receives updates on the order status.
  6. **Delivery:** Customer receives the delivery and can provide feedback.

**Delivery Personnel Journey**:

* 1. **Login**: Delivery person logs in to the app.
  2. **View Schedule**: Delivery person views the list of assigned orders.
  3. **Navigate**: Delivery person uses the app to navigate to delivery locations.
  4. **Confirm Delivery**: Delivery person marks orders as delivered.

**Store Staff Journey**:

* 1. **Login**: Store staff logs in to the app.
  2. **Manage Inventory**: Store staff updates stock levels and product information.
  3. **Process Orders**: Store staff views and manages incoming orders.
  4. **Track Deliveries**: Store staff monitors delivery progress and manages delivery assignments.

# **3. NON-FUNCTIONAL REQUIREMENTS**

**1. Performance Requirements**

* **User Story:** As a customer, I want the app to load within 2 seconds, so that I can browse products quickly without frustration.
* **Definition:** The application should respond to user actions, such as searches and transactions, in under 2 seconds to ensure a smooth user experience (geeksforgeeks, 2024).
* **Backlog Priority:** High

**2. Scalability Requirements**

* **User Story:** As a store manager, I want the app to handle up to 1,000 simultaneous users, so that we can accommodate high traffic during peak hours.
* **Definition:** The application must be designed to scale horizontally, allowing additional server resources to be added without downtime, to support increased user load (geeksforgeeks, 2024).
* **Backlog Priority:** High

**3. Reliability Requirements**

* **User Story:** As a customer, I want the app to be available 99.9% of the time, so that I can place orders whenever I need.
* **Definition:** The application should ensure high availability and uptime, with minimal disruptions, to maintain user trust and operational continuity (geeksforgeeks, 2024).
* **Backlog Priority:** High

**4. Maintainability Requirements**

* **User Story:** As a developer, I want the codebase to be modular and well-documented, so that I can easily make updates and bug fixes without affecting other parts of the application.
* **Definition:** The application should follow best practices for code organization and documentation to facilitate ongoing maintenance and future enhancements (geeksforgeeks, 2024).
* **Backlog Priority:** Medium

**5. Security Requirements**

* **User Story:** As a customer, I want my payment information to be securely processed and stored, so that I can shop without worrying about data breaches.
* **Definition:** The application must implement secure payment processing, data encryption, and user authentication to protect sensitive information from unauthorized access (geeksforgeeks, 2024).
* **Backlog Priority:** High

**6. Usability Requirements**

* **User Story:** As a first-time user, I want the app to have an intuitive interface, so that I can easily navigate and complete my purchases without prior experience.
* **Definition:** The application should provide a user-friendly interface that is easy to navigate, with clear instructions and accessible design to enhance user experience for all demographics (geeksforgeeks, 2024).
* **Backlog Priority:** High

**7. Interoperability Requirements**

* **User Story:** As a store manager, I want the app to integrate with existing inventory management systems, so that we can keep stock levels updated in real-time.
* **Definition:** The application must support integration with third-party services and existing systems, such as payment gateways and inventory management solutions, to ensure seamless operation (geeksforgeeks, 2024).
* **Backlog Priority:** Medium

**8. Internationalisation / Localisation Requirements**

* **User Story:** As a customer, I want to use the app in my preferred language, so that I can easily understand and navigate the platform.
* **Definition:** The application should support multiple languages and localize content, including currency formats and regional preferences, to cater to a diverse customer base (geeksforgeeks, 2024).
* **Backlog Priority:** Medium

# **4. ANALYSIS ARTIFACTS**

In the context of the FARRARMERE EXPRESS MART delivery app, the project operates within multiple **bounded contexts** that clearly reflect the functional and business areas of the system. These contexts help in defining responsibilities and interactions within the app, ensuring that the system is modular, scalable, and easy to maintain. Each bounded context corresponds to a specific domain of responsibility within the supermarket's operations and user interactions.

**1. Bounded Contexts Identified**

1. **Customer Ordering Context**
   * **Description:** This context is responsible for all customer-facing functionalities, such as browsing products, managing the shopping cart, placing orders, and tracking deliveries. The context also includes account management (profile, delivery addresses, and order history).
   * **Core Entities:** Customer, Order, Cart, Product, Delivery, Profile.
   * **Responsibilities:**
     + Product browsing and search functionality.
     + Order placement and tracking.
     + Payment processing integration.
     + Order history and account management.
2. **Delivery Management Context**
   * **Description:** Focuses on managing delivery assignments and logistics. Delivery personnel interact with this context to update the status of deliveries and view assigned orders.
   * **Core Entities:** Delivery Personnel, Assigned Orders, Route, Status Update.
   * **Responsibilities:**
     + Assigning deliveries to personnel.
     + Tracking delivery status and updating the system.
     + Delivery route optimization.
3. **Inventory and Product Management Context**
   * **Description:** This context involves managing the product catalog, updating stock levels, and handling promotions. It is primarily used by the store manager to ensure that product availability is accurate and up to date.
   * **Core Entities:** Product, Inventory, Promotion, Stock.
   * **Responsibilities:**
     + Managing product listings and availability.
     + Stock updates and inventory tracking.
     + Setting up product promotions and discounts.
4. **User and Role Management Context**
   * **Description:** This context manages user authentication, authorization, and role-based access control. It is used by administrators to manage system users and assign roles such as customer, delivery personnel, and store manager.
   * **Core Entities:** User, Role, Permissions, Administrator.
   * **Responsibilities:**
     + User account creation and role assignment.
     + Managing user permissions and access levels.
     + System auditing and security management.
5. **Order Fulfillment Context**
   * **Description:** Responsible for processing orders placed by customers, ensuring that the orders are packed, ready for delivery, and passed on to the delivery personnel for completion. This context manages the internal processing of orders and is primarily used by the store managers.
   * **Core Entities:** Order, Fulfillment, Packing, Ready for Delivery.
   * **Responsibilities**:
     + uml between the product and delivery contexts.

**2. UML Documentation of Bounded Contexts**

Each bounded context is represented as a **service** in the UML Package Diagram, showing how the contexts are modularized and how they interact with each other. The goal is to maintain separation of concerns while ensuring smooth communication across different parts of the system.

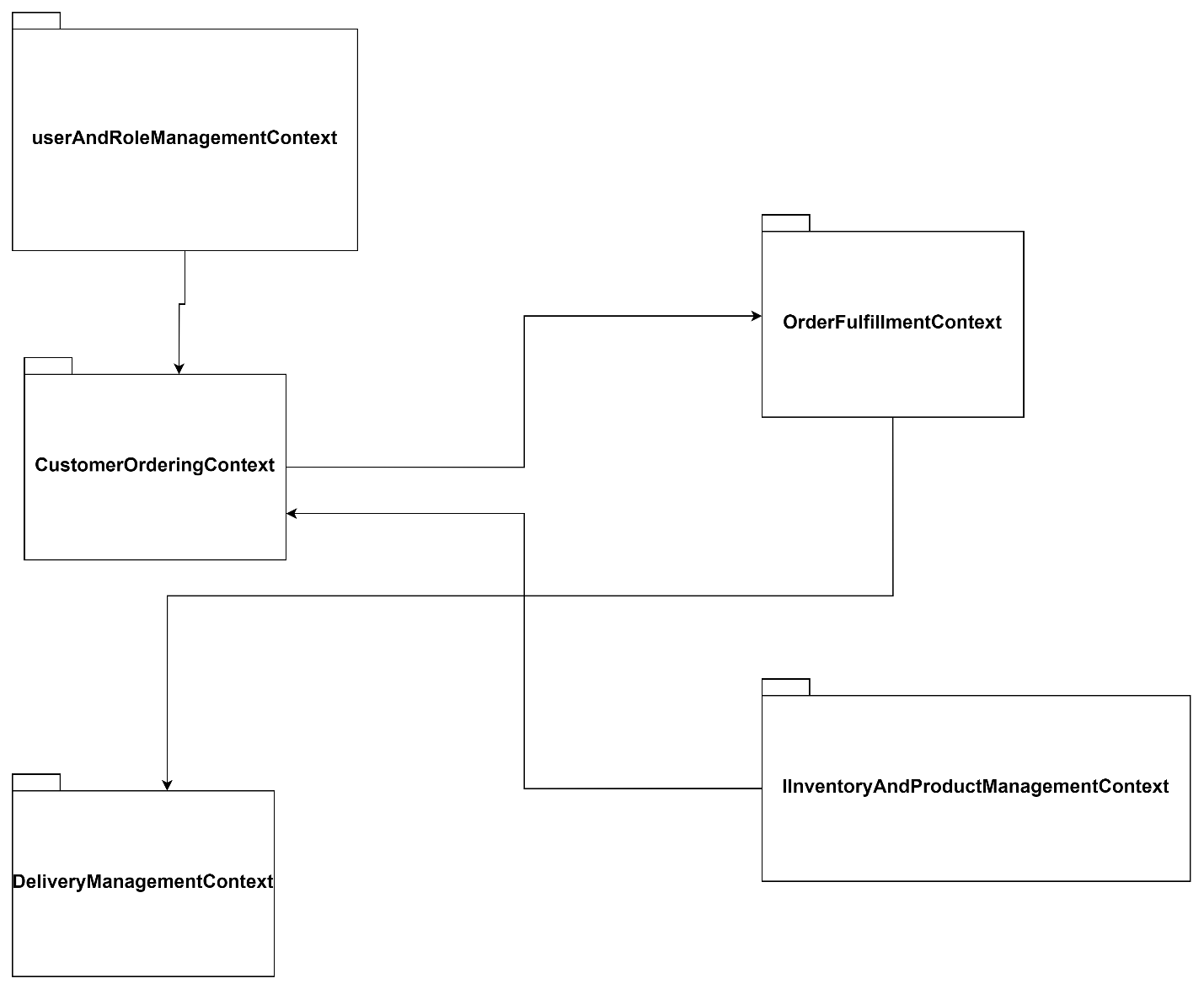


Figure 1: UML Package Diagram of the FARRARMERE EXPRESS MART Delivery App.

**UML Package Diagram Explanation**

* **Customer Ordering Context:**
  + Includes packages for browsing products, managing the cart, and placing orders.
  + Interfaces with the Order Fulfillment Context when an order is placed.
* **Delivery Management Context:**
  + Handles packages for delivery assignments and updates the order status.
  + Communicates with the Order Fulfillment Context to receive delivery-ready orders.
* **Inventory and Product Management Context:**
  + Contains packages for product catalog management, inventory updates, and promotions.
  + Provides product data to the Customer Ordering Context.
* **User and Role Management Context:**
  + Controls user roles, authentication, and permissions.
  + Connects to all other contexts for user access control.
* **Order Fulfillment Context:**
  + Orchestrates order processing and readiness for delivery.
  + Interfaces with the Customer Ordering Context for new orders and the Delivery Management Context for dispatching orders.

By defining these bounded contexts, the project ensures that each domain of the supermarket's delivery app is independently manageable and can evolve without affecting other contexts. This separation of concerns also promotes better scalability, maintainability, and security within the system. The UML Package Diagram provides a high-level overview of how these contexts are organized and interact with one another.

# **5. SOFTWARE IMPLEMENTATION**

This section focuses on demonstrating the understanding of the software implementation process through various UML diagrams that capture class relationships, interactions between components, and important state transitions for the app. The diagrams provide a detailed overview of how different parts of the system are structured and how they interact to fulfill key functionalities.

# **5.1 UML Object Diagram**

The UML Object Diagram captures the relationships between different entities identified in the analysis phase for the **FARRARMERE EXPRESS MART Delivery App**. These entities represent the key classes involved in the app, such as Customer, Order, Product, and DeliveryPersonnel.

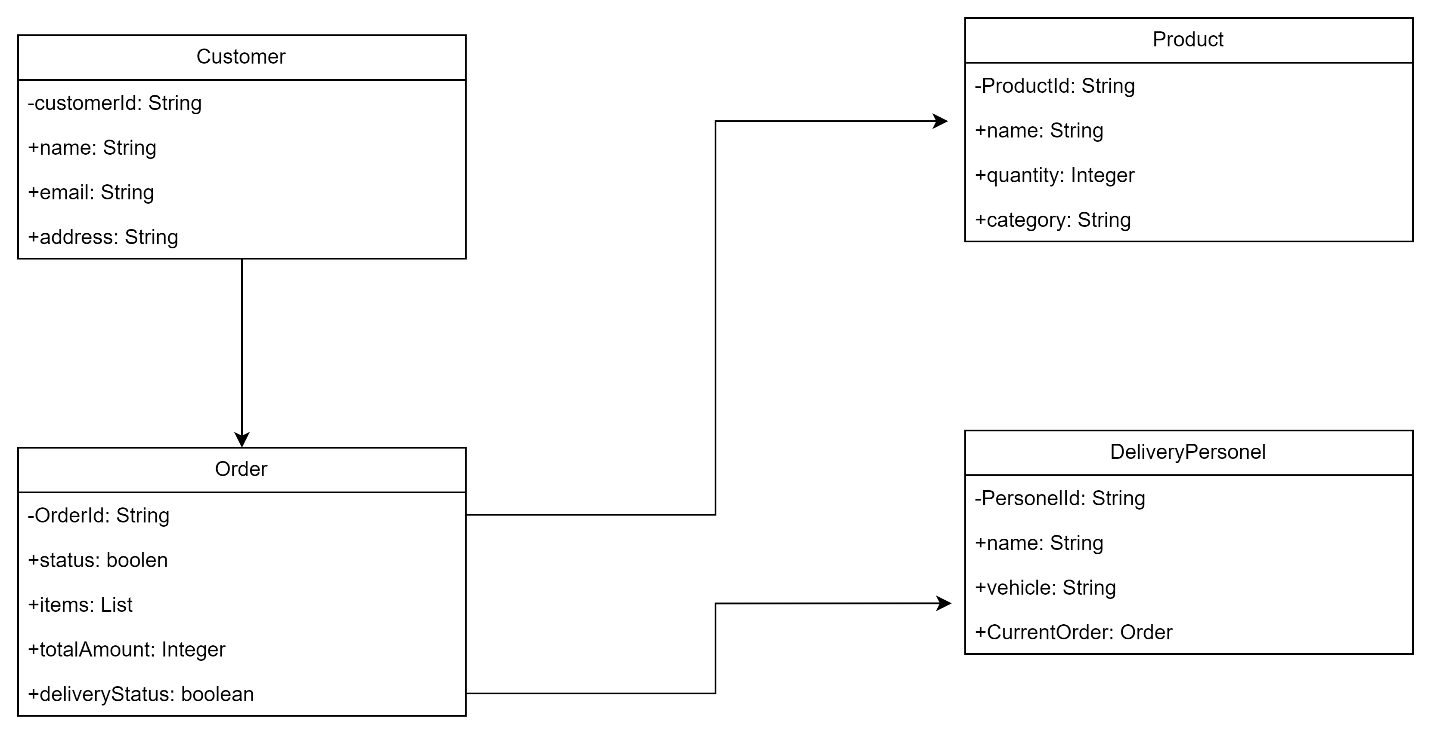


Figure 2: FARRARMERE EXPRESS MART Delivery App UML Object diagram

**Entities and Relationships:**

* **Customer**:
* Attributes: customerId, name, email, address, orderHistory
* Relationships:
  + - 1 to Many relationship with Order (A customer can place multiple orders)
* **Order**:
  + Attributes: orderId, orderStatus, orderItems, totalAmount, deliveryStatus
  + Relationships:
    - Many to 1 relationship with Customer (Orders belong to customers)
    - 1 to Many relationship with Product (An order contains multiple products)
    - 1 to 1 relationship with DeliveryPersonnel (An order is assigned to a delivery person)
* **Product**:
  + Attributes: productId, name, price, quantity, category
  + Relationships:
    - Many to Many relationship with Order (Multiple products in one order, multiple orders can include the same product)
* **DeliveryPersonnel**:
  + Attributes: personnelId, name, vehicleType, currentAssignedOrder
  + Relationships:
    - 1 to 1 relationship with Order (Delivery personnel are assigned to orders)

# **5.2. UML Sequence Diagram**

A UML Sequence Diagram helps to visualize the interactions between objects in specific scenarios, focusing on the ordering process and the delivery status update within the app. The diagram shows how components like the Customer, system and delivery interact during these flows.

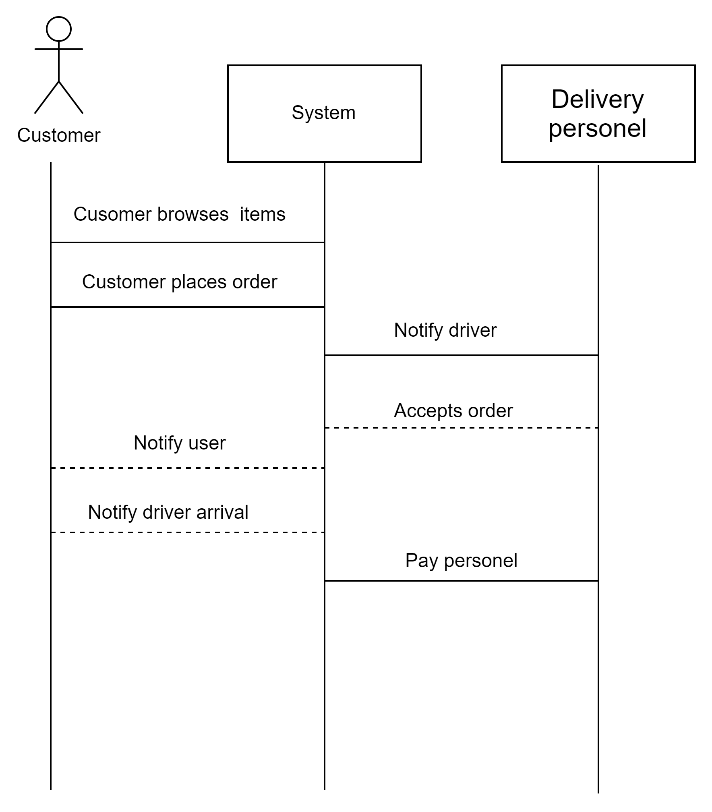


Figure 3: UML Sequence Diagram of the flow an order

# **5.3. UML State Diagram**

UML State Diagrams show the state transitions of key entities during the operation of the app. For this app, the primary entity to model with state transitions is the **Order**, as it moves through different stages of processing and delivery.

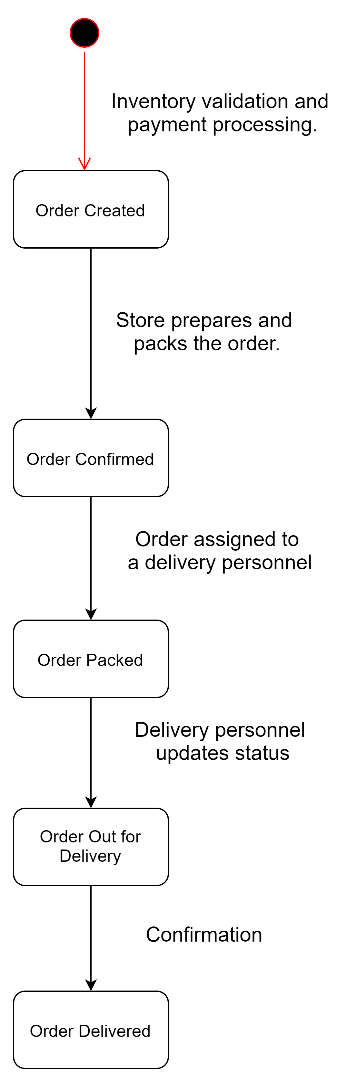


Figure 4: UML State Diagrams showing the operation of the app

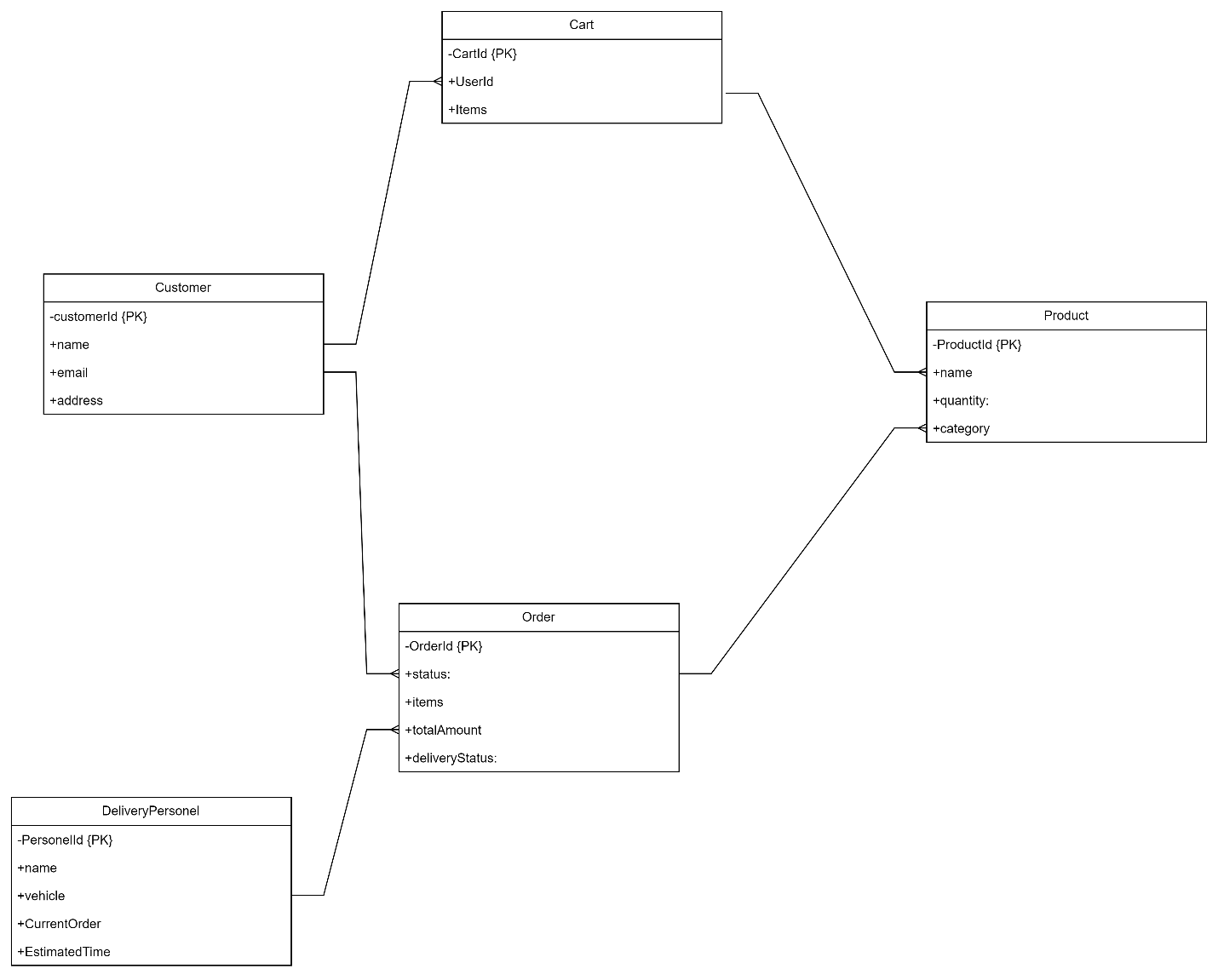
# **6. DATA SCHEMA DOCUMENTATION**

# **6.1. Data Storage Technologies and Choices**

For the delivery app, we use **Firebase Firestore**, a NoSQL document database (geeksforgeeks, 2024), due to its real-time data synchronization, scalability, and ease of integration with other Firebase services. Firestore is well-suited for applications requiring dynamic, real-time data interactions and flexible schema designs.

# **6.2. Entity Relationship Diagram (ERD)**

Since Firestore is a NoSQL database, traditional ERD concepts are adapted into a document-oriented structure. Here’s how the entities and their relationships are represented:



# **6.3. JSON Schemas for Document Store (Firestore)**

Customer:  
{

“CustomerID": "string",

"Name": "string",

"Email": "string",

"Role": "string",

"Address": {

"Street": "string",

"City": "string",

"State": "string",

"ZipCode": "string"

}

}

Product:  
{

"ProductID": "string",

"Name": "string",

"Description": "string",

"Price": "number",

"StockQuantity": "number",

"Category": "string"

}

Order:  
{

"OrderID": "string",

"UserID": "string",

"Products": [

{

"ProductID": "string",

"Quantity": "number"

}

],

"TotalPrice": "number",

"OrderStatus": "string",

"OrderDate": "timestamp"

}

Cart:  
{

"CartID": "string",

"UserID": "string",

"Items": [

{

"ProductID": "string",

"Quantity": "number"

}

]

}

Delivery:  
  
{

"DeliveryID": "string",

"OrderID": "string",

"DeliveryPersonID": "string",

"DeliveryStatus": "string",

"EstimatedDeliveryTime": "timestamp"

}

# **7. ARCHITECTURE ARTIFACTS**

# **7.1. Design patterns**

**Repository Pattern**

* **Description:** This pattern abstracts data access, allowing the application to interact with data sources (such as Firestore) without being tightly coupled to them. It provides a clean separation between data access logic and business logic (medium, 2017).
* **Usage:** will be sed to manage interactions with Firestore. For instance, repositories handle CRUD operations for orders, inventory, and user data, making the codebase easier to manage and test.

# **7.2. Architecture patterns**

**Microservices Architecture**

* **Description:** This pattern divides the application into loosely coupled services that are independently deployable and scalable (Atlassian, *[s.d]*).
* **Usage:** will be implemented using Firebase Cloud Functions to handle discrete functionalities like order processing, payment handling, and delivery management. Each service operates independently, enhancing scalability and maintainability.

# **7.3 Cloud**

## **7.3.1. Cloud Architecture Overview**

The application is built on **Firebase**, utilizing various cloud services for a seamless delivery app experience. The key components are:

* **Firebase Authentication:** Manages user authentication securely (medium, 2018).
* **Firestore:** A NoSQL real-time database to store customer orders, product data, and user information (medium, 2018).
* **Firebase Cloud Functions:** Serverless backend to handle logic such as order processing, payment handling, and notifications (medium, 2018).
* **Firebase Hosting:** For serving static assets like the frontend or web-based components (medium, 2018).
* **Firebase Cloud Messaging (FCM):** Push notifications for delivery updates and order status changes (medium, 2018).

This architecture ensures scalability, security, and real-time interaction without the need for heavy server management.

## **7.3.2. Cloud Architecture Decisions**

* **Serverless infrastructure:** Firebase Cloud Functions were selected to reduce server management complexity and allow automatic scaling. This supports growth as the app’s user base increases without needing manual provisioning of server resources (IBM, 2018).
* **Real-time capabilities:** Firestore was chosen to handle live data updates, particularly important for tracking delivery status, product availability, and real-time customer notifications (IBM, 2018).
* **Integrated services:** Firebase services are tightly integrated, providing secure authentication, scalable databases, and cloud functions with minimal configuration and seamless interaction (medium, 2018).
* **Global presence:** Firebase offers global CDN (Content Delivery Network) support, ensuring fast access to content and services for users no matter where they are (IBM, 2018).

## **7.3.3. Cloud Networking**

* **Protocols**:
  + All communications are secured using HTTPS protocols for data transmission between the client-side (mobile app) and Firebase services.
  + REST APIs are used for database interactions (reading/writing order data, updating inventory) and user authentication (IBM, 2018).
  + WebSockets: Firestore uses WebSocket connections to enable real-time data synchronization for use cases like delivery tracking or inventory updates (IBM, 2018).
* **Network Configuration**:
  + Firewall and IP restrictions can be applied through Firebase rules and functions to ensure access is granted only to trusted sources (IBM, 2018).
  + Routing and Load Balancing: Firebase automatically handles load balancing by distributing traffic across multiple global servers to ensure smooth performance even under heavy loads (IBM, 2018).

## **7.3.4. Cloud Security**

* **Authentication and Access Control**:
  + **Firebase Authentication** ensures secure user login and registration. User roles (admin, customer, delivery personnel) are implemented using Firebase’s role-based security model (medium, 2018).
  + Security rules are implemented in Firestore to restrict access based on user roles and permissions. Customers can access their own orders, but not other customer or administrative data (medium, 2018).
* **Data Encryption**:
  + All data transmitted between the client and Firebase servers is encrypted in transit using **SSL/TLS** encryption.
  + Data at rest in Firestore and other Firebase services is also encrypted by default to ensure protection against unauthorized access.
* **Complete Mediation**:
  + Firebase rules are used to enforce complete mediation, ensuring that every request (read or write) to the Firestore database is authenticated and authorized according to the predefined user roles and permissions.
* **Threat Mitigation**:
  + Common threats like **SQL Injection** or **Cross-Site Scripting (XSS)** are avoided since Firestore is a NoSQL database and all input is validated.
  + **Authentication brute force attacks** are mitigated through rate-limiting and multi-factor authentication (MFA) features provided by Firebase.

# **8. SECURITY**

Security is a crucial aspect of the Application, given the sensitive nature of customer data, payment information, and personal information associated with online orders and deliveries. In this section, we will outline the potential threat actors and threat vectors, as well as the mitigations implemented to protect against them. Security considerations also include balancing protection with usability and ensuring complete mediation when accessing every data object in the system.

# **8.1. Potential Threat Actors Documented**

**Internal Threat Actors:**

* **Malicious Employees:** An internal staff member (e.g., admin or delivery personnel) who has access to sensitive information such as customer data, stock levels, or order statuses and might misuse this information for personal gain or malicious purposes (Ibm, *s.d.*).
* **Negligent Employees:** Employees who may unintentionally expose the system to threats due to improper handling of sensitive data, weak password management, or failure to follow security protocols (Ibm, *s.d.*).

**External Threat Actors:**

* **Hackers/Cybercriminals:** Individuals or groups seeking to exploit vulnerabilities in the system for financial gain, data theft, or disruption of services (Ibm, *s.d.*).
* **Competitors:** Rivals in the market who might attempt to gain unauthorized access to sensitive business data to acquire competitive advantages (Ibm, *s.d.*).
* **Third-Party Vendors:** External services and APIs integrated with the app (such as payment gateways or delivery tracking) may also present risks if not properly secured, as attackers can use them as backdoors into the system (Ibm, *s.d.*).

**End-Users:**

* **Customers:** Although typically benign, customers may introduce security risks through compromised accounts, weak passwords, or phishing attacks that grant unauthorized access to their accounts (Ibm, *s.d.*).

# **8.2. Potential Threat Vectors Documented**

**Common Vulnerabilities:**

* Phishing Attacks: End-users, especially customers, may be susceptible to phishing emails or fake websites designed to steal their credentials, leading to unauthorized access to their accounts.
* Injection Attacks (e.g., SQL/NoSQL Injection): Attackers may exploit the backend API endpoints by inserting malicious code to retrieve or modify data from the database.
* Broken Authentication: If the app fails to properly authenticate users or validate user credentials, attackers may gain unauthorized access to accounts.
* Data Leakage: Sensitive data, such as customer payment information, personal addresses, or order details, may be exposed if not properly encrypted or secured.
* Man-in-the-Middle (MITM) Attacks: Without secure communication protocols, attackers may intercept data being transmitted between the app and the backend.

# **8.3. Mitigations for Threats Documented**

**Authentication & Authorization:**

* Multi-Factor Authentication (MFA): Implemented for administrators and delivery personnel, providing an additional layer of security beyond just passwords. This helps mitigate the risk of compromised accounts (IBM, 2024).
* Strong Password Policies: Enforced for all users, requiring a minimum length, complexity, and expiration policies, reducing the likelihood of weak passwords being exploited (IBM, 2024).

**Encryption:**

* Encryption in Transit: All communications between the mobile app, backend servers, and the database are encrypted using HTTPS/TLS. This mitigates MITM attacks by ensuring data cannot be intercepted in transit (IBM, 2024).
* Encryption at Rest: Sensitive data, such as passwords and payment information, are stored using industry-standard encryption algorithms (e.g., AES-256), protecting the data even if a breach occurs (IBM, 2024).

**API Security:**

* Rate Limiting: Implemented on the API Gateway to prevent brute force and Denial of Service (DoS) attacks by limiting the number of requests from a single source within a specific time frame (IBM, 2024).
* Input Validation: Strict whitelisting of user input (using regular expressions and sanitization libraries) is enforced to mitigate injection attacks such as SQL/NoSQL injection and cross-site scripting (XSS) (IBM, 2024).

**Economy of Mechanism:**

* Simple, modular security measures have been used to avoid unnecessary complexity. This ensures that each mitigation is easy to understand, implement, and maintain without introducing new vulnerabilities through complexity. For example, using a centralized authentication service (like **Firebase Authentication**) reduces the complexity of handling login security (IBM, 2024).

# **8.4. Balancing Security with Usability**

While security is of paramount importance, usability must not be compromised, especially in a customer-facing application like the FARRARMERE EXPRESS MART Delivery App. The following measures are implemented to strike the right balance:

* Simplified MFA for Admins and Employees:

While customers only need secure passwords, administrators and delivery personnel are required to use MFA. However, to avoid friction, this MFA is designed to be simple, with options like email or mobile authenticator apps that do not slow down the user experience significantly (techtarget, *s.d*).

* Session Management:

To avoid unnecessary logouts during usage, session tokens are securely managed via JWTs with well-defined expiration times. Refresh tokens are implemented for customers, allowing longer sessions without requiring frequent logins, making the app more user-friendly without compromising security (techtarget, *s.d*).

# **8.5. Complete Mediation and Security for Data Access**

Security is enforced at every level to ensure complete mediation, which means that every access to sensitive data is authenticated, authorized, and validated.

* **Access Controls:** Role-Based Access Control (RBAC) is implemented, ensuring that customers, administrators, and delivery personnel only have access to the specific features and data they need. For instance, customers can only access their order history, while admins can view overall store performance (techtarget, *s.d*).
* **Field-Level Encryption:** Sensitive fields, such as passwords and payment information, are encrypted even within the database, ensuring that even if unauthorized access occurs, the attacker cannot easily read this data (techtarget, *s.d*).
* **API Endpoint Security:** Every API call is authenticated via JWT, ensuring that only users with valid tokens can access or modify data. In the event of a token expiration or invalid token, access is denied immediately (techtarget, *s.d*).

# **9. DEVOPS**

The rapid deployment of a high-quality application was a key consideration throughout the development of the FARRARMERE EXPRESS MART Delivery App. Implementing a robust DevOps strategy enabled the team to automate testing, building, and deployment processes, ensuring faster and more reliable releases. This section outlines the use of DevOps practices in the project, with a particular focus on the GitHub Actions Pipeline.

# **9.1 GitHub Actions Pipeline**

GitHub Actions is an automation tool that allows developers to define workflows for continuous integration (CI) and continuous deployment (CD) in their projects. In the FARRARMERE EXPRESS MART Delivery App project, GitHub Actions played a pivotal role in ensuring that code was tested and deployed efficiently.

**Pipeline Overview**

The GitHub Actions pipeline for this project is designed with the following key steps:

1. **Code Commit and Push**

* Developers work on individual features or bug fixes in separate branches (github, *s.d*).
* Once changes are ready, they push code to the repository on GitHub. This triggers the pipeline (github, *s.d*).

1. **Automated Tests**

* Once a push is detected, GitHub Actions runs the unit tests automatically (github, *s.d*).
* This step ensures that the code changes do not break any existing functionality. If tests fail, the pipeline stops and feedback is provided to the developer (github, *s.d*).

1. **Code Linting and Formatting**

* After testing, the code is passed through linting tools to ensure it adheres to coding standards (github, *s.d*).
* If the code does not meet the predefined formatting or linting standards, the pipeline provides an alert, and the developer needs to make corrections (github, *s.d*).

1. **Build Process**

* If the tests pass, the application is built. For the FARRARMERE EXPRESS MART Delivery App, this means compiling the code, ensuring that all dependencies are resolved, and packaging the app for deployment.
* The build process is automated to ensure consistency and reduce the likelihood of human error (github, *s.d*).

1. **Security Scans**

* Security checks are performed to identify potential vulnerabilities, such as dependency issues, insecure configurations, or code patterns that could introduce security risks (github, *s.d*).

1. **Deployment**

* Once all tests, linting, and security checks pass, the application is automatically deployed to the Firebase environment (github, *s.d*).
* The deployment step pushes the latest build to staging for final testing. Once validated, it can be promoted to the production environment (github, *s.d*).

1. **Notification**

* Upon successful deployment, GitHub Actions sends notifications to the development team via email or Slack, depending on the setup, informing them that the new version is live or ready for production testing (github, *s.d*).

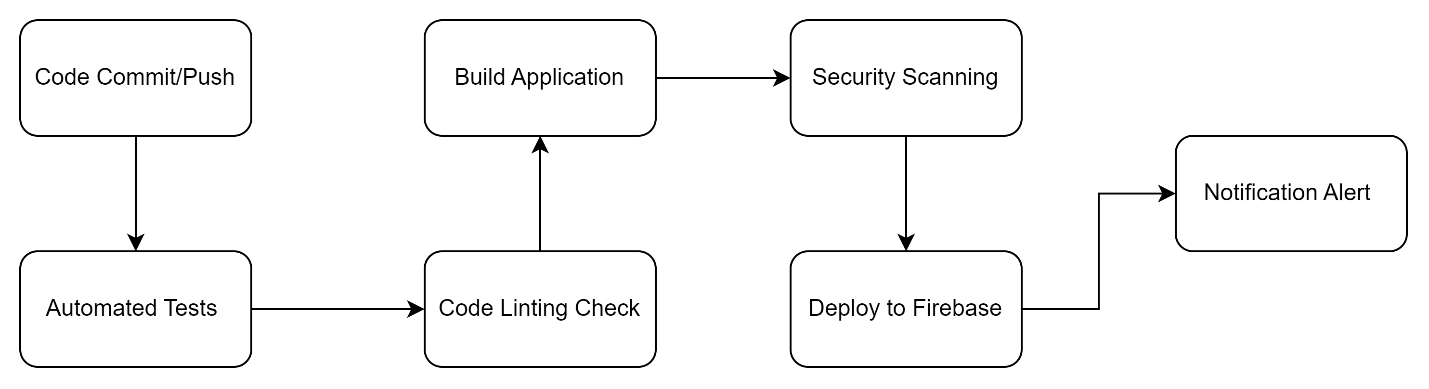


Figure 6: Flow Chart: GitHub Actions Pipeline

# **10. RUNNING COSTS**

To ensure the sustainability of the FARRARMERE EXPRESS MART Delivery App, it is critical to estimate the monthly running costs, taking into account both predicted user growth and the scalability of the technologies used. These estimates will provide a clear understanding of the resources needed over the next two years and identify points where scaling is necessary. In this section, we will outline the projected growth scenarios, predict costs based on various user growth cases, and discuss potential technology replacements when scaling limits are reached.

# **10.1. Predicted User Growth Documentation**

The App is designed to handle both in-store customers who switch to online ordering and new users attracted through delivery services. Based on discussions with the client, the predicted user growth is as follows:

* **Initial User Base:** The app is expected to start with 500 active users in the first month after launch.
* **Monthly Growth Rate:** The client estimates a monthly growth rate of 10% in the best case, 2% in the worst case, and 6% in the average case based on historical data of customer behavior and anticipated marketing campaigns.

# **10.2. Scaling Points of Firebase Services**

Firebase services scale automatically based on the number of users and interactions with the app. However, each Firebase service has its own pricing model based on specific usage metrics, such as the number of reads/writes, the amount of data stored, and the number of invocations for Cloud Functions. Below are the scaling points for each key Firebase service used in the app:

1. **Firebase Authentication:**

* Free tier allows up to 10,000 verifications per month (medium, 2018).
* After that, there's a cost of $0.01 per verification (medium, 2018).

1. **Firebase Firestore (Database):**

* Free tier includes 50,000 reads, 20,000 writes, and 20,000 deletes per day (medium, 2018).
* **Beyond this, pricing is based on operations:**
  + - Reads: $0.06 per 100,000 reads (medium, 2018).
    - Writes: $0.18 per 100,000 writes (medium, 2018).
    - Deletes: $0.02 per 100,000 deletes (medium, 2018).
* **Data storage:** First 1 GB is free, and then $0.18 per GB/month (medium, 2018).

1. **Firebase Cloud Functions:**
   * Free tier includes 2 million invocations per month, with 400,000 GB-seconds and 200,000 CPU-seconds (medium, 2018).
   * Beyond this, pricing is based on invocations, CPU, and memory usage (medium, 2018).
2. **Firebase Hosting:**
   * Free tier includes 10 GB of stored data and 360 MB/day of bandwidth.
   * After that, the cost is $0.026 per GB for storage and $0.15 per GB for bandwidth (medium, 2018).

# **10.3. Predictive Models for User Growth**

The following table outlines projected costs over two years, based on best-case, worst-case, and average user growth scenarios. The main factors considered are the number of reads/writes to the Firestore database, Firebase Authentication verifications, and Cloud Function invocations, with storage and bandwidth scaling based on the number of users.

* Best Case Growth (10% monthly growth)
* Worst Case Growth (2% monthly growth)
* Mean Case Growth (6% monthly growth)

# **10.4. Adopting Alternative Technologies at Scale**

As the app scales, Firebase services can still handle significant user bases, but there are points where adopting alternative technologies might become more cost-effective:

* **Firestore Database**
* **Replacement Technology:** Google Cloud Bigtable may be a better fit for very large datasets, as it provides lower latency for large-scale read and write operations and scales horizontally (medium, 2020).
* **Cost:** Bigtable costs are based on node hours, storage, and network bandwidth, but at high traffic volumes, it may prove more efficient for apps with millions of users (medium, 2020).
* **Cloud Functions**
* **Replacement Technology:** At scale, Firebase Cloud Functions can become expensive due to high invocation costs. Migrating to Google Kubernetes Engine (GKE) or Google Cloud Run may provide more control over costs as it allows for containerized services (geeksforgeeks, 2023).
* **Cost:** While GKE and Cloud Run introduce more management overhead, they could become more cost-efficient for handling millions of function calls compared to Cloud Functions (geeksforgeeks, 2023).

# **11. CHANGE MANAGEMENT**

Effective change management is critical to ensuring the smooth adoption and long-term success of the application. Below is an analysis of how the software will be adopted by both the organization and its users, along with strategies to drive and sustain adoption.

# **11.1. Organizational Adoption**

**How and Why Will the Organization Adopt the Software?**

The FARRARMERE EXPRESS MART has struggled to streamline and modernize its delivery service, relying on outdated manual systems to manage online orders, assign delivery personnel, and track inventory. As the company grows and online ordering becomes more popular, the inefficiencies in the current system become more costly, leading to slower deliveries, poor customer satisfaction, and a limited ability to scale. The delivery app addresses these gaps by:

* **Centralizing Operations:** It provides a platform where the organization can manage all delivery-related tasks, such as processing orders, assigning drivers, and monitoring real-time delivery status, all from one interface (forbes, 2022).
* **Efficiency Gains:** Automating tasks reduces the time spent on manual processing and minimizes human errors (forbes, 2022).
* **Scalability:** The app’s architecture is built to scale with the organization, ensuring that future growth is supported seamlessly (forbes, 2022).

**Argument for Organizational Adoption:**

The adoption of the delivery app by the organization is motivated by the clear benefits of efficiency, improved customer service, and scalability. It allows FARRARMERE EXPRESS MART to remain competitive in an increasingly digital retail market while modernizing their operations for better long-term growth (forbes, 2022).

# **11.2. User Adoption**

**How and Why Will Users Adopt the Software?**

Users of the delivery app are likely to adopt the software because it directly addresses key pain points in their current experience with the supermarket's online ordering and delivery services.

* **User-Centric Features:** The app offers convenient features like real-time delivery tracking, notifications, and order updates, giving customers greater control and transparency over their purchases (forbes, 2022).
* **Enhanced Convenience:** The app streamlines the process of placing orders, paying securely, and communicating with delivery drivers, all from their mobile devices, improving the overall experience (forbes, 2022).
* **Loyalty Programs:** The app integrates loyalty and discount programs that incentivize customers to make repeat purchases (forbes, 2022).

**Argument for User Adoption:**

User adoption is driven by the app’s focus on convenience, transparency, and rewards. By addressing common frustrations (such as delayed deliveries and lack of order updates), the app will appeal to users who prioritize efficiency and reliability.

# **11.3. Adoption Strategy**

* **Strategy for Organizational Adoption:**
* **Training & Support:** A comprehensive training program will be rolled out to ensure that all employees, from management to drivers, are proficient in using the app. This will include live training sessions, recorded tutorials, and accessible documentation (forbes, 2022).
* **Pilot Program:** Initially, the app will be introduced as a pilot in selected regions or branches of FARRARMERE EXPRESS MART. This phased approach allows the organization to monitor and refine operations before full deployment, minimizing disruptions (forbes, 2022).
* **Feedback Loops:** Regular feedback from both employees and customers will be collected and analyzed to ensure that the system continues to meet operational needs and improve over time (forbes, 2022).
* **Strategy for User Adoption:**
* **Marketing Campaigns:** To raise awareness, a marketing campaign will be launched, highlighting the app's features, benefits, and convenience. This will include digital ads, in-store promotions, and targeted email campaigns to existing customers (forbes, 2022).
* **Incentives for Early Adoption:** Early adopters will be incentivized with promotions, such as discounted delivery fees, first-order bonuses, and exclusive loyalty points. This will encourage customers to try the app early and frequently (forbes, 2022).
* **Seamless Onboarding:** The app will have an easy, intuitive sign-up process with clear guidance on how to use key features. Tutorials and FAQs will be built into the app, ensuring that users can quickly become proficient (forbes, 2022).

# **11.4. Maintaining and Supporting the Software**

To ensure the long-term success of the software and continued client satisfaction, a realistic support and maintenance plan will be implemented:

* **Ongoing Maintenance:** Regular updates will be scheduled to ensure that the software remains compatible with the latest technology, including system updates, bug fixes, and security patches (forbes, 2022).
* **User Feedback & Updates:** There will be a continuous feedback mechanism allowing both the organization and customers to report issues or request new features. A dedicated support team will review this feedback and roll out updates accordingly (forbes, 2022).
* **Performance Monitoring:** The app’s performance will be continuously monitored to ensure it scales appropriately with increasing user numbers, and to detect any issues in real-time (forbes, 2022).
* **Customer Support:** A 24/7 support desk will be made available for both users and the organization to troubleshoot any issues that arise. This support can be accessed via chat, phone, or email (forbes, 2022).

**Strategy to Maintain & Support the Software:**

The combination of regular maintenance, performance monitoring, and customer feedback ensures the app remains functional and relevant as it evolves alongside user and organizational needs. As the app gains traction, additional support resources can be scaled to meet growing demand (forbes, 2022).

# **REFERENCE LIST**

Termly. 2023. What Is Data Privacy? Complete Guide for Businesses, 20 October 2023. [online] Available at: <https://termly.io/resources/articles/what-is-data-privacy/> [Accessed 22 September 2024]

Smartlook. 2023. Mobile user consent under data privacy regulations [updated], 19 June 2023. [online] Available at: <https://www.smartlook.com/blog/consent-for-mobile-apps-in-2021/> [Accessed 22 September 2024]

Medium. 2023. Mobile Apps Data Collection: Why Your Personal Information is at Stake, 17 October 2023. [online] Available at: <https://medium.com/@chaintech-network/mobile-apps-data-collection-why-your-personal-information-is-at-stake-4826ee3635c6> [Accessed 22 September 2024]

Geeksforgeeks. 2024. Non-Functional Requirements in Software Engineering, 3 July 2024. [online] Available at: <https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/> [Accessed 24 September 2024]

Geeksforgeeks. 2024. Difference Between Firebase and Firestore, 9 August 2024. [online] Available at: <https://www.geeksforgeeks.org/difference-between-firebase-and-firestore/> [Accessed 25 September 2024]

Medium. 2017. Repository Design Pattern, 20 April 2017. [online] Available at: <https://medium.com/@pererikbergman/repository-design-pattern-e28c0f3e4a30> [Accessed 25 September 2024]

Atlassian.*s.d.* Microservices design patterns for DevOps teams. [online] Available at: <https://www.atlassian.com/microservices/cloud-computing/microservices-design-patterns> [Accessed 25 September 2024]

Medium. 2018. What is Firebase? The complete story, abridged. [online] Available at: [What is Firebase? The complete story, abridged. | by Doug Stevenson | Firebase Developers | Medium](https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0) [Accessed 25 September 2024]

Medium. 2020. Getting Started with Bigtable on GCP, 11 July 2011. [online] Available at: <https://medium.com/google-cloud/getting-started-with-bigtable-on-gcp-adfb896e0b26> [Accessed 25 September 2024]

Geeksforgeeks. 2023. Introduction to Kubernetes (K8S), 28 November 2023. [online] Available at: <https://www.geeksforgeeks.org/introduction-to-kubernetes-k8s/> [Accessed 25 September 2024]

GitHub, *s.d*. GitHub Actions documentation [online] Available at: <https://docs.github.com/en/actions> [Accessed 25 September 2024]

Forbes. 2022. The Four Principles Of Change Management, 7 August 2022. [online] Available at: <https://www.forbes.com/advisor/business/principles-of-change-management/> [Accessed 25 September 2024]

techtarget, *s.d*. application security. [online] Available at: <https://www.techtarget.com/searchsoftwarequality/definition/application-security> [Accessed 26 September 2024]

IBM. 2024. What is application security (AppSec)? 5 June 2024. [online] Available at: <https://www.ibm.com/topics/application-security> [Accessed 26 September 2024]

IBM. *[s.d.]. What is a threat actor?* [online] Available at: <https://www.ibm.com/topics/threat-actor> [Accessed 26 September 2024]