Name: Vaibhavi Balaji Hipparkar

PRN: 23620009

System Design Document

1. Introduction:-

The RM Milk Suppliers system is designed to modernize and streamline the operations of a local dairy store, providing a digital solution to manage inventory, orders, billing, and customer interactions. The system is intended for use by both customers and the dairy provider, ensuring a user-friendly experience while optimizing the business processes involved in dairy product management.

2. Current Software Architecture:-

As this is a new system, there is no existing software architecture in place.

3. Proposed Software Architecture :-

3.1. Overview :

The proposed software architecture for the RM Milk Suppliers system is a mobile-based application developed using Flutter, with Firebase as the backend for database management. The architecture follows a client-server model, where the mobile client interacts with the Firebase backend to perform operations such as order placement, billing, inventory management, and user authentication.

3.2. Subsystem decomposition :

The system is divided into the following subsystems:

User Authentication Subsystem: Manages user registration, login, and authentication.

Product Management Subsystem: Handles the addition, update, and deletion of products, as well as inventory tracking.

Order Management Subsystem: Manages customer orders, including order placement, processing, and status tracking.

Billing Subsystem: Generates invoices automatically upon order confirmation and manages billing-related data.

Notification Subsystem: Sends notifications to users regarding order updates, billing, and other important information.

Admin Subsystem: Allows administrators to manage the system settings, user accounts, and access comprehensive reports.

3.3. Hardware/software mapping:

The system is primarily designed for Android devices. The Flutter framework ensures compatibility across various Android devices. Firebase serves as the backend, hosted on Google Cloud, with no special hardware requirements beyond the standard Android device capabilities.

3.4. Persistent data management:

Firebase Firestore is used for storing persistent data. The data is structured in a NoSQL format, with collections and documents to represent entities such as users, products, orders, and invoices. Each document stores data in key-value pairs, allowing for flexible and scalable data management.

3.5. Access control and security:

User access is controlled through Firebase Authentication, which supports secure login via email and password. Data encryption is applied to both data at rest and in transit. Role-based access control ensures that only authorized users (e.g., administrators, providers) can access sensitive areas of the system.

3.6. Global software control:

The system is designed to handle multiple users concurrently, with Firebase managing real-time data synchronization. Flutter's asynchronous programming capabilities ensure that operations such as data fetching, and order processing are non-blocking, maintaining a responsive user interface.

3.7. Boundary conditions:

Startup: The application initializes by loading the user authentication state. If a user is logged in, the system directs them to the appropriate dashboard (customer or provider).

Shutdown: On shutdown, any unsaved data is stored locally and synchronized with Firebase when the application is restarted.

Error Handling: The system includes global exception handlers to catch and manage errors. Common errors include network failures, authentication errors, and data validation issues. Users are notified of errors through in-app messages, and the system attempts automatic retries where appropriate.

4. Subsystem services:-

4.1 User Authentication Subsystem :

RegisterUser(email, password): Registers a new user and stores their credentials securely.

LoginUser(email, password): Authenticates the user and grants access to the system.

LogoutUser(): Logs out the current user and clears their session data.

4.2 Product Management Subsystem :

AddProduct(productDetails): Adds a new product to the inventory.

UpdateProduct(productID, updatedDetails): Updates the details of an existing product.

DeleteProduct(productID): Removes a product from the inventory.

GetProductList(): Retrieves the list of available products for display.

4.3 Order Management Subsystem :

PlaceOrder(orderDetails): Places a new customer order and generates an order ID.

UpdateOrderStatus(orderID, status): Updates the status of an existing order (e.g., processing, ready for pickup).

GetOrderHistory(userID): Retrieves the order history for a specific user.

4.4 Billing Subsystem :

GenerateInvoice(orderID): Generates an invoice for a confirmed order.

SendInvoice(email, invoiceDetails): Sends the generated invoice to the customer via email.

GetBillingHistory(userID): Retrieves past invoices and billing history for a user.

4.5 Notification Subsystem :

SendNotification(userID, message): Sends a notification to a specific user.

ScheduleNotification(userID, message, time): Schedules a notification to be sent at a specified time.

4.6 Admin Subsystem :

ManageUserAccounts(userID, action): Allows admins to create, update, or disable user accounts.

ViewReports(reportType, parameters): Generates reports on system usage, sales, and other key metrics.

ConfigureSystemSettings(settings): Allows the admin to configure system-wide settings such as product categories, pricing rules, and delivery zones.