3. **CHAPTER THREE - SYSTEM DESIGN**

3.1. **Introduction**

This Part describes the proposed design for the food delivery android application. The system design concerns with the overall view of the application from the functional viewpoint. It is the process of defining the components, modules, interfaces, and data for the application to satisfy specified requirements. This section provides a clear description of the overall design parts of the “Food delivery android application” in organized way.

3.2. **Design Goals**

Design goals describe the qualities of the Application in which, developers should optimize that derived from the nonfunctional requirements. The following major quality perspectives are used to identify the design goals of the application.

**Design Performance:**

The design performance of a food delivery Android app should be focused on creating an intuitive user experience. The app should be easy to navigate and have a clear layout that allows users to quickly find what they are looking for. The app should also be visually appealing and have a modern design that is easy to understand.

**Dependability:**

The dependability of a food delivery Android app should be focused on ensuring that the app is reliable and secure. The app should be tested thoroughly to ensure that it is free from bugs and other issues. The app should also have a secure payment system that is encrypted and protected from hackers.

**Maintainability:**

The maintainability of a food delivery Android app should be focused on ensuring that the app is updated regularly with new features and bug fixes. The app should also be monitored for any potential security issues and have a system in place to quickly address any issues that arise. The app should also have a system in place to quickly respond to customer feedback and requests.

* Create an intuitive and user-friendly interface that allows users to easily navigate the app and find the food they are looking for.
* Ensure that the app is secure and that all user data is protected.
* Allow users to quickly and easily place orders and pay for their food.
* Provide users with accurate information about the restaurants and food items available.
* Allow users to track their orders in real-time.
* Provide users with a variety of payment options.
* Allow users to rate and review restaurants and food items.
* Provide users with helpful customer service and support.

**End User Criteria**

Easy to use interface: The app will have an intuitive and user-friendly interface that is easy to navigate and understand; the app loads quickly and is responsive to user input. The app should provide secure payment options for users to pay for their orders. The app has accurate order tracking and provides accurate order tracking so users can easily keep track of their orders. The app should provide push notifications to alert users of order updates and other important information. The app should allow users to customize their orders with various options such as toppings, sides, and more.

The App will provide an easy checkout process that is quick and efficient. Variety of restaurants: The app should offer a variety of restaurants to choose from so users can find the food they want. The app should provide ratings and reviews from other users to help users make informed decisions. It has Customer support service in which the app should provide customer support to help users with any issues they may have.

3.2 **Proposed Software Architecture**

The proposed software architecture for a food delivery Android application would include the following components:

* User Interface: This component will provide the user with a graphical interface to interact with the application. It will include features such as a search bar, a list of restaurants, and a map view.
* Database: This component will store all the data related to the application, such as user information, restaurant information, and order information for our food delivery app we will use firebase real-time database.
* Backend Server: This component will handle all the requests from the user interface and will communicate with the database to retrieve and store data.
* Payment Gateway: This component will handle all the payments related to the application.
* Delivery Service: This component will handle all the logistics related to the delivery of the food orders.

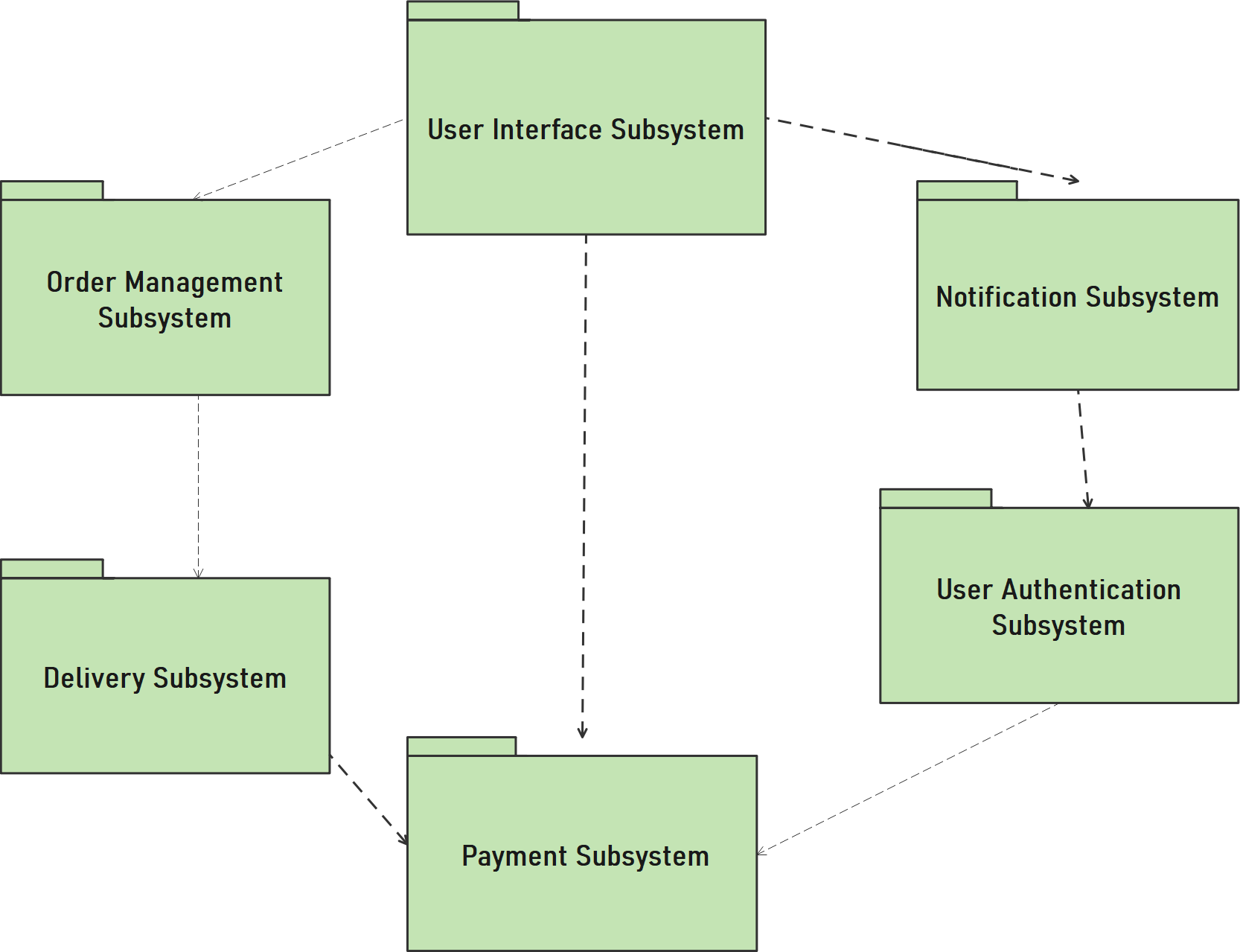
3.3.2 **Subsystem Decomposition**

Subsystem decomposition describes the division of the system into subsystems that are collection of classes, associations, operations, events, and constraints that are closely interrelated with each other and the responsibilities of each subsystem.

The “Food Delivery Android Application” has the following sub systems.

* **User Interface Subsystem:** the user interface subsystem of a food delivery application would be responsible for providing an intuitive and user-friendly experience for customers. This would include designing the user interface, creating navigation menus, and developing features such as search and filter functions be able to view menus, order history, and other information.
* **User Authentication Subsystem:** This subsystem is responsible for user authentication and authorization. It allows users to create accounts, log in, and manage their profiles.
* **Order Management Subsystem:** This subsystem is responsible for managing orders, including creating, editing, and canceling orders.
* **Payment Subsystem:** This subsystem is responsible for processing payments, including credit card, debit card, and other payment methods.
* **Delivery Subsystem:** This subsystem is responsible for managing the delivery of orders, including tracking orders, assigning drivers, and managing delivery routes.
* **Notification Subsystem**: This subsystem is responsible for sending notifications to users, such as order confirmations, delivery updates, and promotional messages.

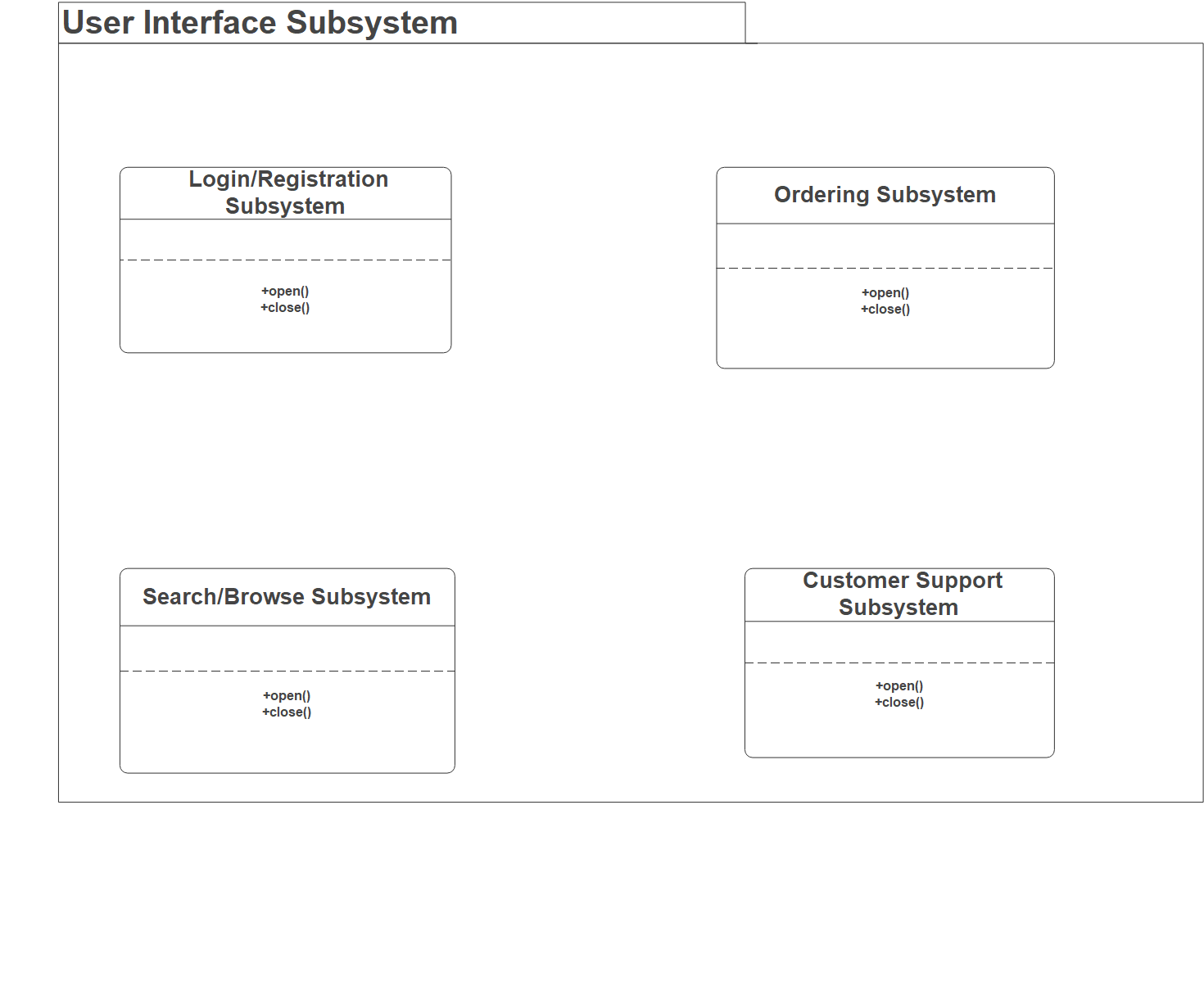
The system decomposition and the interaction among the subsystems are shown in figure 3.3-2



**Figure 3.3-2 System Decomposition**

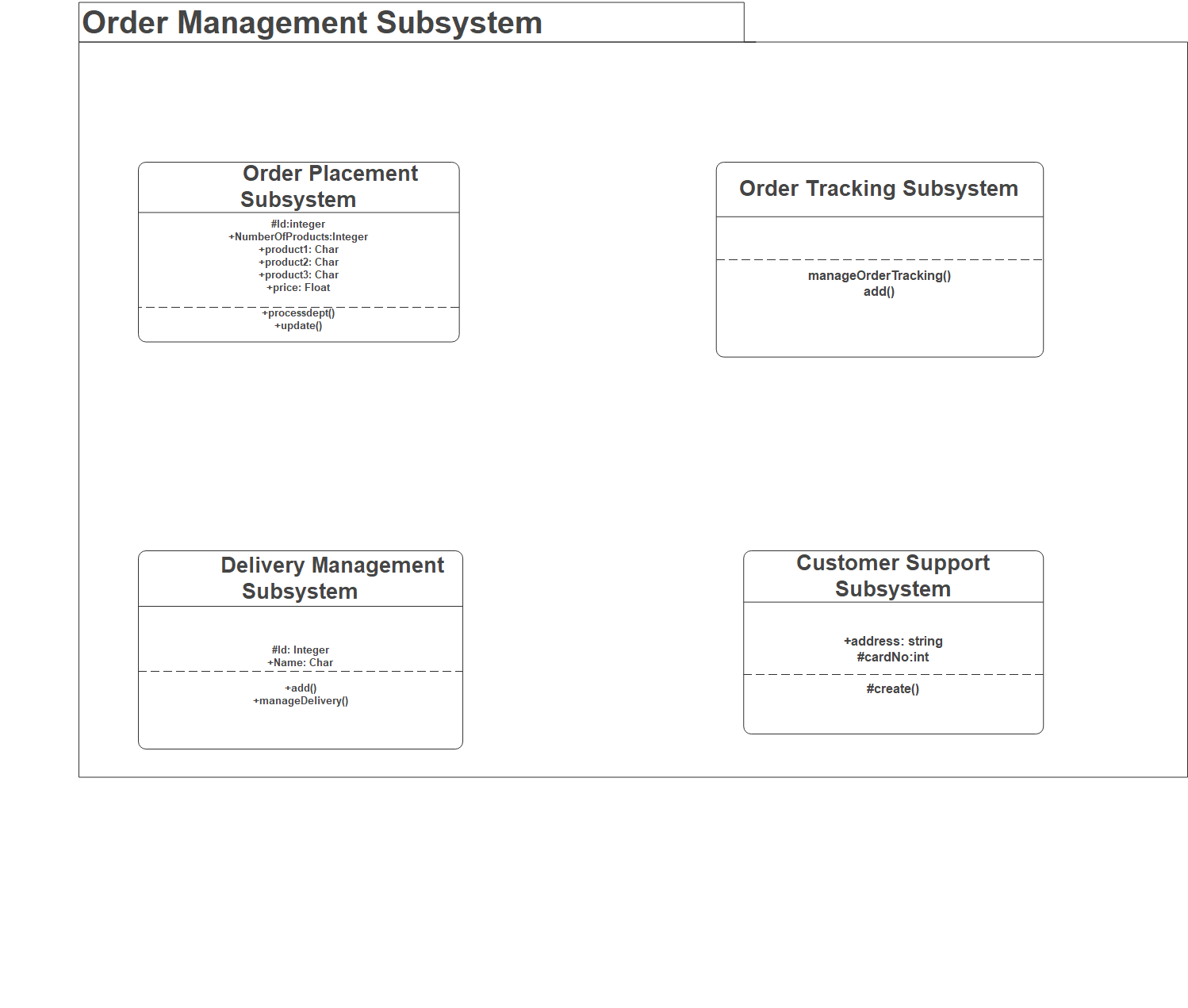
1. **User Interface Subsystem:**

* **Login/Registration Subsystem:** This subsystem allows users to create an account and log in to the application. It also allows users to reset their passwords and manage their account settings.
* **Search/Browse Subsystem**: This subsystem allows users to search for restaurants and browse through menus. It also allows users to filter their search results based on various criteria such as cuisine type, price range, and location.
* **Ordering Subsystem**: This subsystem allows users to place orders for food items from the restaurants they have selected. It also allows users to customize their orders and add special instructions.
* **Customer Support Subsystem**: This subsystem allows users to contact customer support in case of any queries or issues. It also allows users to provide feedback and rate their experience with the application.



1. **Order Management Subsystem:**

* **Order Placement Subsystem**: This subsystem allows customers to place orders for food items from the restaurant. It includes features such as selecting items from the menu, adding items to the cart, and making payment.
* **Order Tracking Subsystem**: This subsystem allows customers to track their orders in real-time. It includes features such as displaying the current status of the order, estimated time of delivery, and estimated cost.
* **Delivery Management Subsystem**: This subsystem allows restaurants to manage their delivery operations. It includes features such as assigning orders to delivery
* **Customer Support Subsystem**: This subsystem allows customers to get help with their orders. It includes features such as providing customer support via phone, email, and chat.



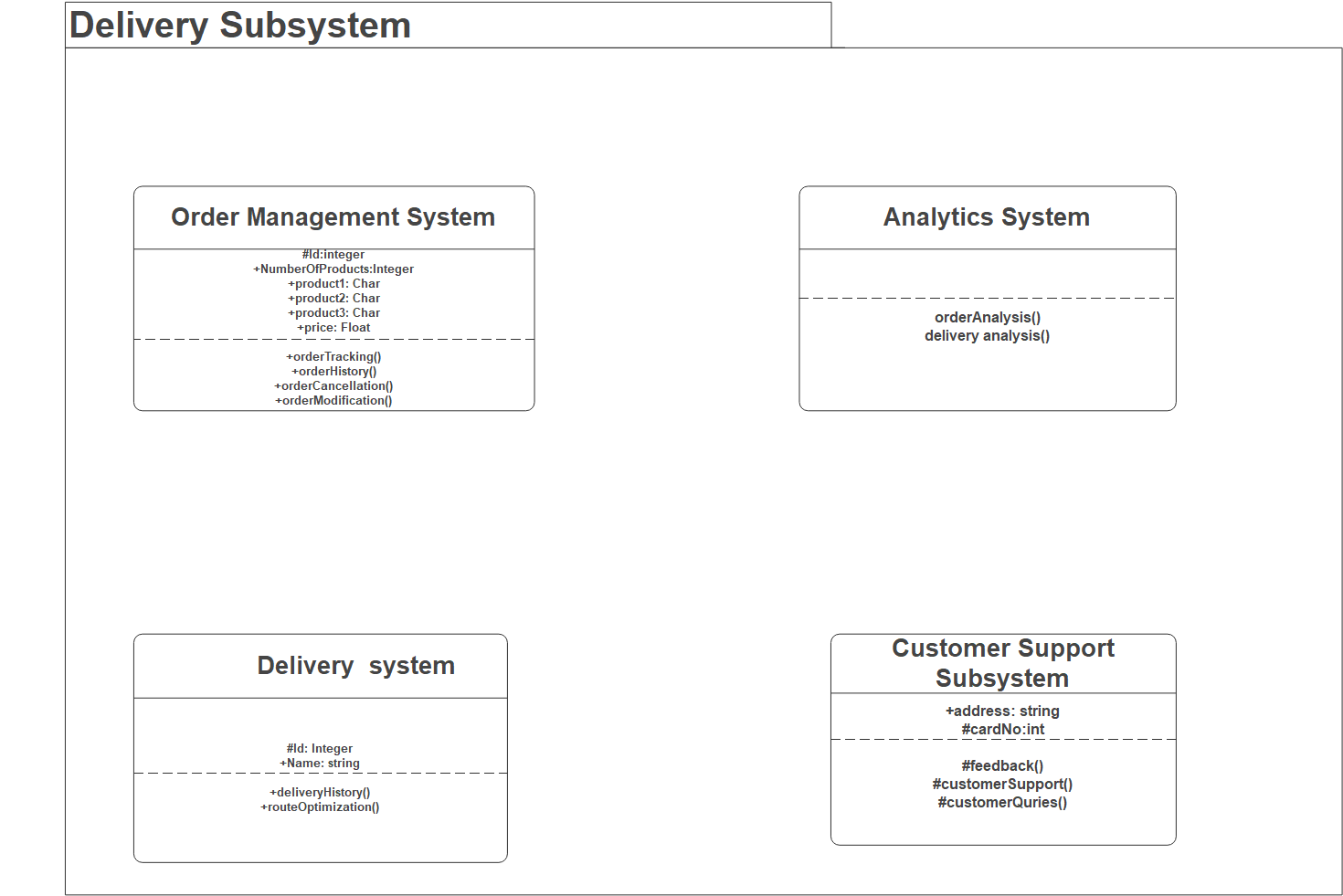
1. **Delivery Subsystem**

1. Order Management System: This system is responsible for managing the orders placed by customers. It includes features such as order tracking, order history, order cancellation, and order modification.

2. Delivery System: This system is responsible for delivering the food to the customers. It includes features such as route optimization, delivery tracking, delivery history, and delivery cancellation.

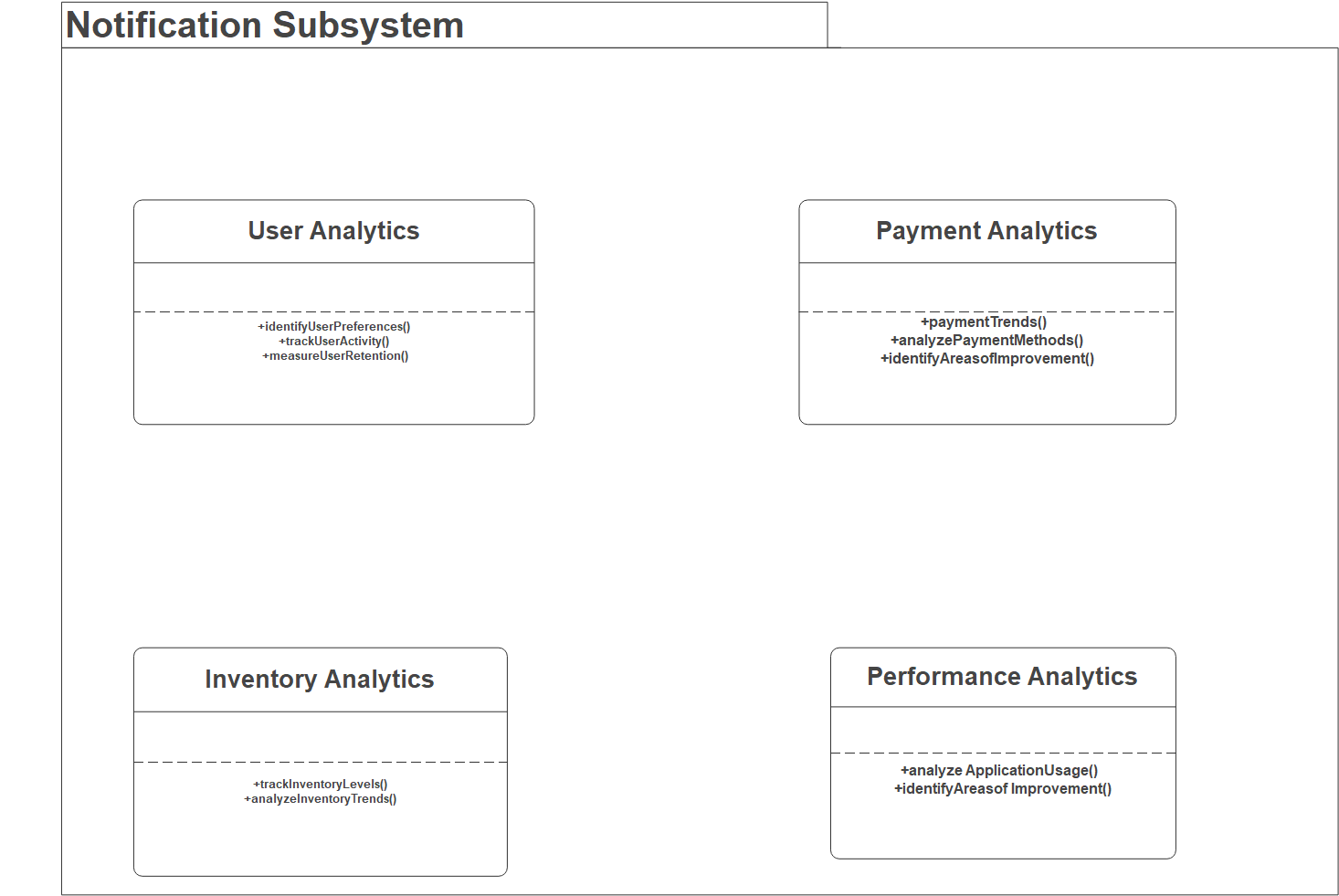
3. Customer Support System: This system is responsible for providing customer support to customers. It includes features such as customer feedback, customer complaints, customer queries, and customer support.

4. Analytics System: This system is responsible for providing insights into the performance of the food delivery application. It includes features such as customer segmentation, order analysis, and delivery analysis.



1. **Notification Subsystem**

* **User Analytics**: This type of analytics helps to track user behavior and engagement with the application. It can be used to identify user preferences, track user activity, and measure user retention.
* **Payment Analytics**: This type of analytics helps to track payment trends, analyze payment methods, and identify areas of improvement.
* **Inventory Analytics**: This type of analytics helps to track inventory levels, analyze inventory trends, and identify areas of improvement.
* **Performance Analytics**: This type of analytics helps to track application performance, analyze application usage, and identify areas of improvement.



1. **Payment Subsystem:**

1. PayPal: PayPal is a popular online payment system that allows customers to pay for their orders using their PayPal accounts.



**3.3.2.1 Hardware/Software Mapping**

Hardware:

* Mobile devices (Smartphone, tablet) with Android operating system.
* GPS enabled device for location tracking.
* Internet connectivity (WiFi, cellular data).

Software:

* Android Studio or other Android app development tools.
* Google Maps API for location-based services.
* Payment gateway API for online transactions.
* Push notifications for order updates.
* Database management system for storing order details and customer information.
* Backend server for data management and processing.



**3.3.3 Persistent data management**

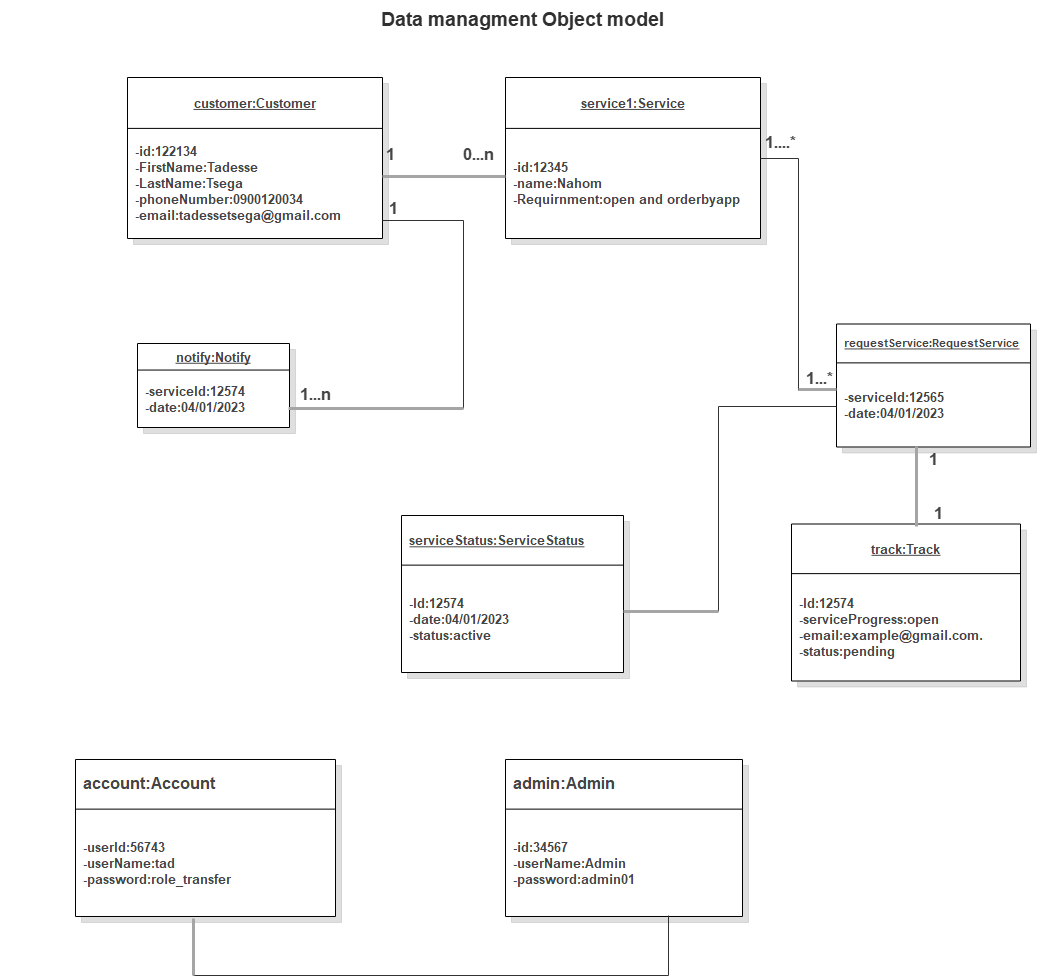
Persistent data management describes the persistent data stored by the Application and the data management infrastructure required for it. The Application uses the Mongo database, a document-oriented database management system classified as a NoSQL database. It uses JSON-like documents. It provides an extensible, highly scalable, cost-effective way to store data. MongoDB offers the advantages of

* Flexibility- ensures that any future changes in customer data requirements can be addressed quickly and efficiently. Also, it does not require defining a schema before you start using the database, making it more flexible than relational databases.
* Scalability - it can easily scale out by adding more nodes to a cluster. And it allows for better performance and increased capacity as the data grows.
* Performance - MongoDB's document-based data model allows faster and more efficient data retrieval.
* Ease of use needed for an effective Android app development
* MongoDB also provides powerful data aggregation and data processing capabilities. Inserting and updating data in MongoDB is faster and more efficient.
* It also provides efficient indexing and querying capabilities, i.e. queries are more straightforward and less complex than those used in relational databases and enable faster data retrieval. All of these features make MongoDB a great choice for creating a powerful and efficient for our application.

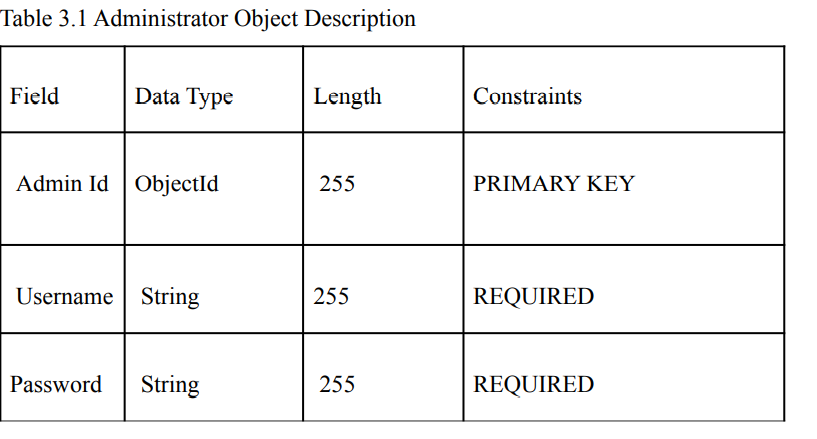
1. Database: A database such as MongoDB can be used to store customer information, order details, payment information, and other data related to the application.
2. Shared Preferences: This is a simple key-value storage system built into Android. It is useful for storing small amounts of data that need to be accessed frequently, such as user preferences or login credentials
3. Content Delivery Network (CDN): A CDN can be used to deliver content quickly and reliably to users around the world.
4. Analytics: Analytics solutions such as Google API can be used to track user behavior and gain insights into how the application is being used.
5. Firebase Real-time Database: This is a cloud-based, NoSQL database provided by Google. It can be used to store and synchronize data for multiple users in real-time
6. Firebase Firestore: This is a cloud-based, NoSQL document database provided by Google. It can be used to store and retrieve data in a flexible, hierarchical structure.

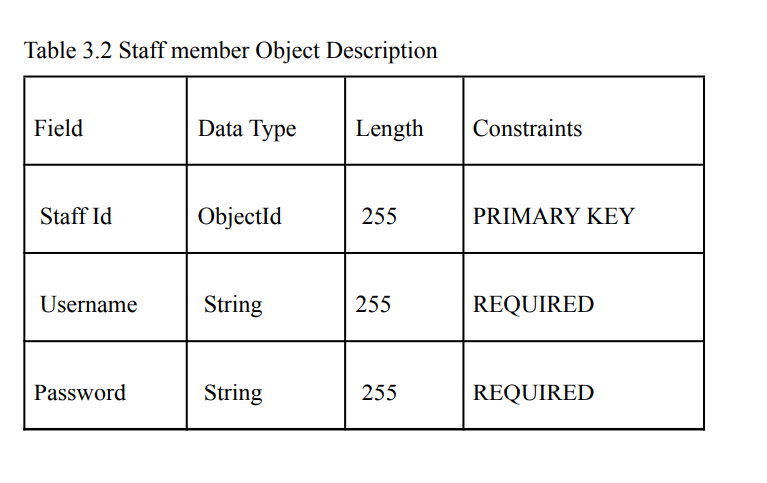
Typical Food delivery android application data schema includes the following information:

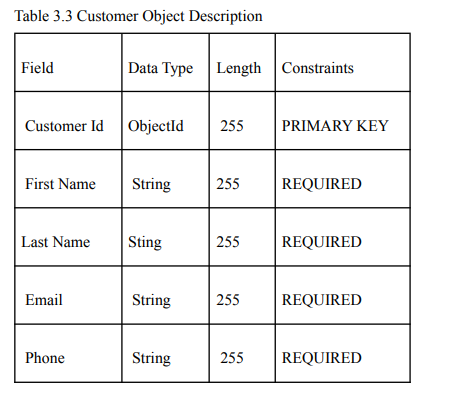
1. **Customer Information** - customer profile data including name (First name and last name), contact information, and other customer-related information like customer interactions (email) and phone.
2. **Administrator**- Administrator data includes Admin Id, username and password.
3. **Track**- this field includes Customer id, Email, status and Service progress
4. **Service**- this field includes Service Id, name and requirement
5. **Request service**- this field includes Request id and date
6. **Service status**- request id, status and date
7. **Notify**- service id and date

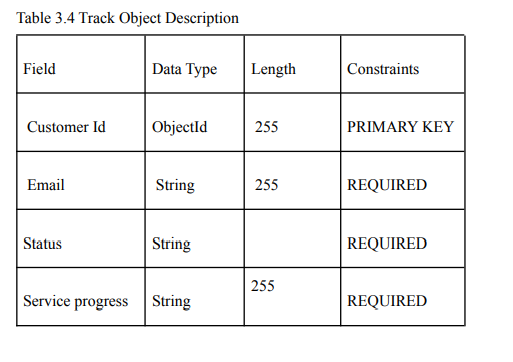


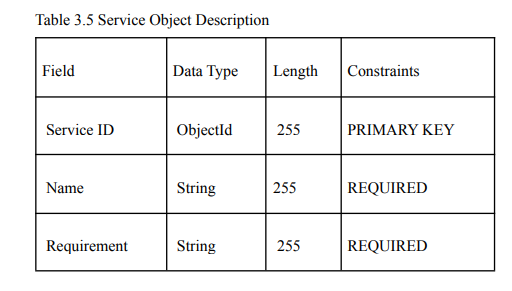
**Object Diagram**

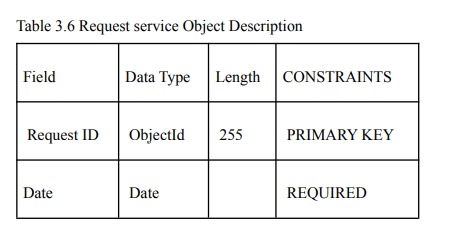


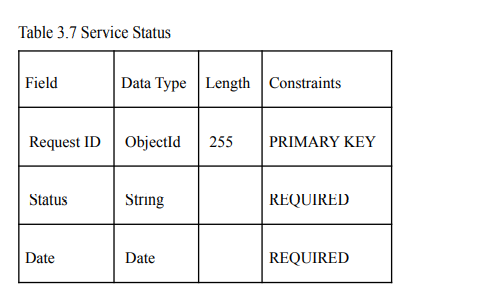


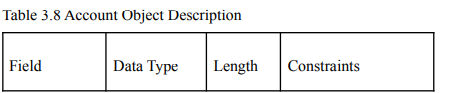


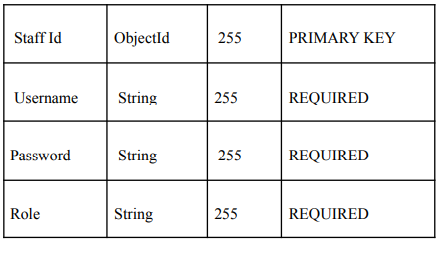


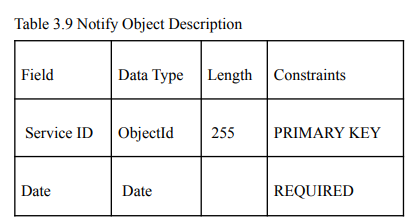


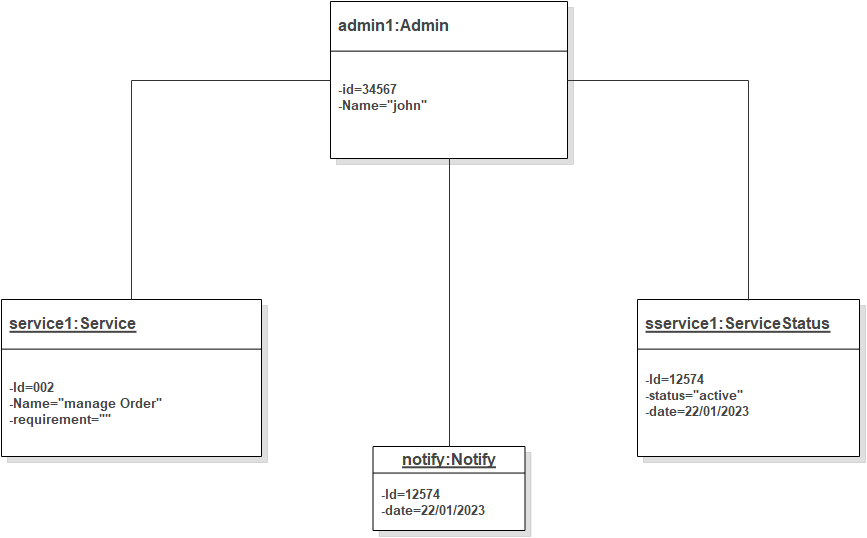


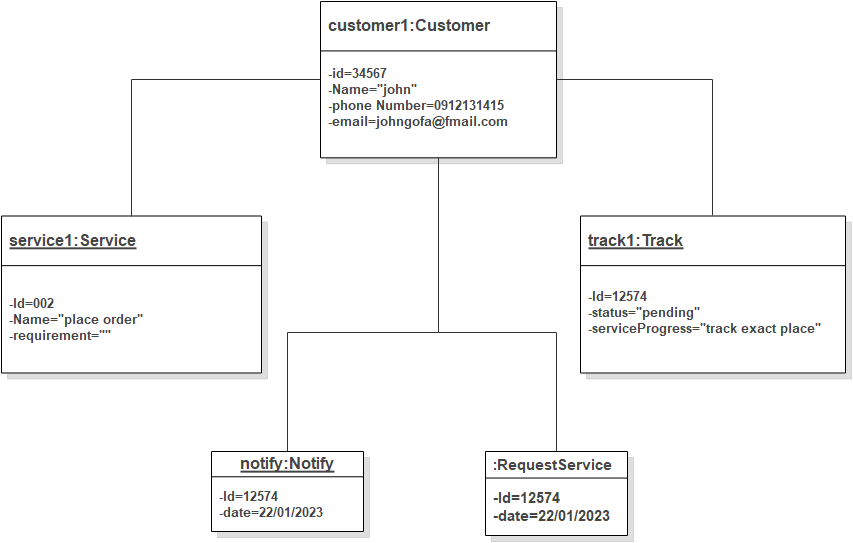


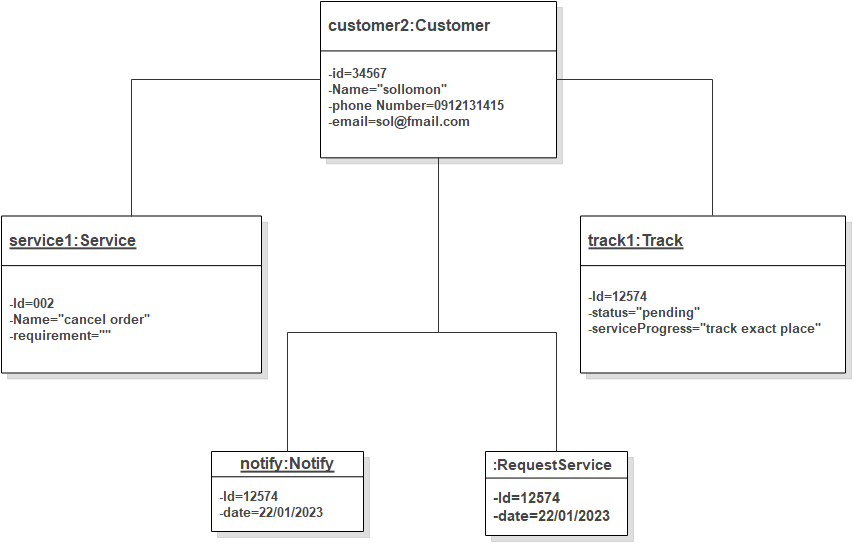


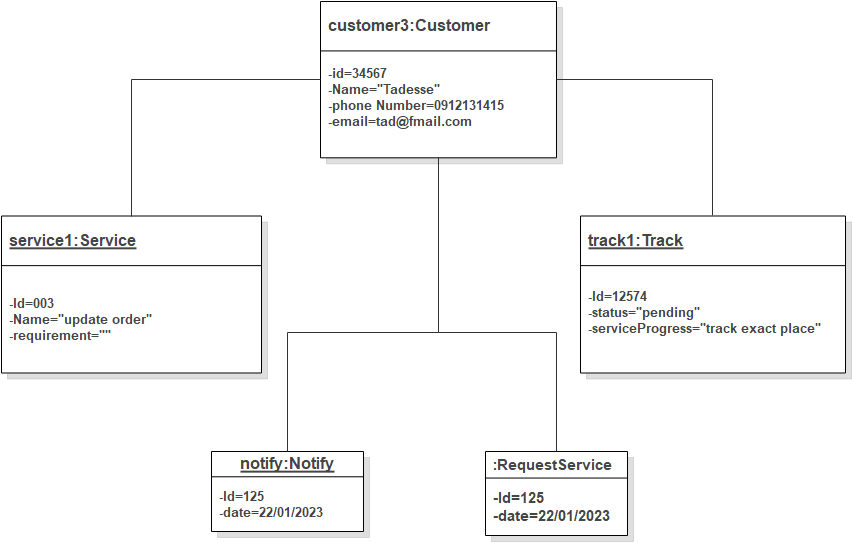


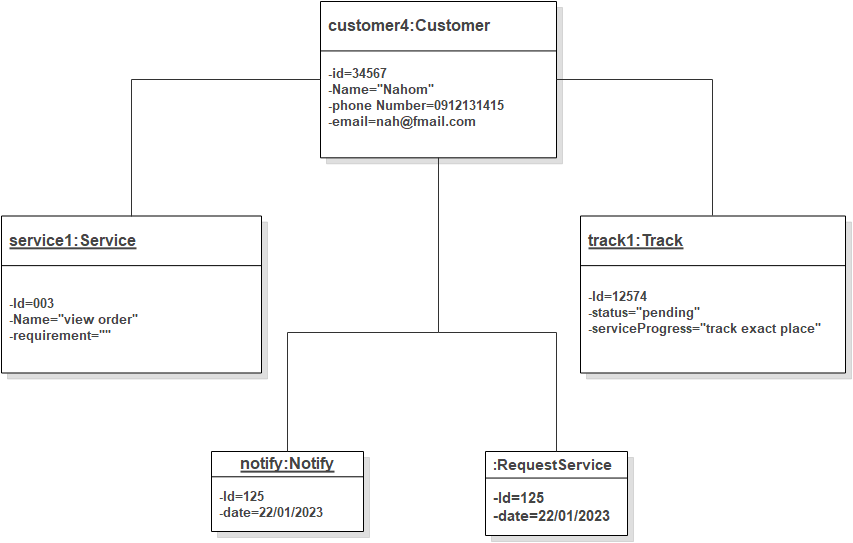












3.4 **ACCESS CONTROL AND SECURITY**

Access control and security are important considerations when developing a food delivery Android application. Some common access control and security measures that can be implemented include

Access control is a security technique that regulates user grants to access the system's functionalities. And make availability of a system to limit what elements can be viewed or modified by a public or private member and who is allowed to access certain resources of the system. and restrict the modification of data or documents to prevent unauthorized changes, malicious modification of data, and accidental introduction of consistency.

Access control is a security technique that governs user permissions to access system functions. And make a system available to limit what elements can be viewed or modified by a public or private member and who is allowed to access specific system resources. And restrict data or document modification to prevent unauthorized changes, malicious data modification, and unintentional consistency introduction.

Some common access control and security measures that can be implemented include:

1. Secure authentication: Food delivery applications should use secure authentication methods such as two-factor authentication, biometric authentication, and password protection to ensure that only authorized users can access the application.
2. Authorization: This is the process of determining whether a user is allowed to access specific resources or perform specific actions within the application. This can be done by assigning roles and permissions to users, or by using access control lists (ACLs) to specify which users are allowed to access specific resources.
3. Data encryption: All data stored in the application should be encrypted to protect it from unauthorized access.
4. Secure payment processing: Food delivery applications should use secure payment processing methods such as tokenization and encryption to protect customer payment information.
5. Secure data storage: All data stored in the application should be stored securely in a cloud-based server to prevent unauthorized access.
6. Secure communication: All communication between the application and its users should be encrypted to protect it from eavesdropping.
7. Secure access control: Food delivery applications should use access control methods such as role-based access control and user authentication to ensure that only authorized users can access the application.
8. Network security: This is the process of protecting the application and its data from unauthorized access over the network. This can be done by using secure protocols such as HTTPS, SSL, and SSH, and by implementing firewalls and intrusion detection/prevention systems.
9. Secure key management: It is important to manage the encryption keys used to encrypt the sensitive data. This can be done by using a Key Management System (KMS) or Hardware Security Module (HSM) to securely store and manage encryption keys.
10. Regular security auditing: It is important to regularly review and audit the application's security measures to ensure that they are functioning as intended and that any vulnerabilities have been identified and addressed.
11. Compliance: It's important to follow the regulations, industry standards, and laws that are relevant to your application, such as the Payment Card Industry Data Security Standards (PCI DSS) and the General Data Protection Regulation (GDPR)
12. Regular software updates: Keeping your application and its dependencies up-to-date can help fix any known vulnerabilities in the software.

These are some of the common security measures that can be implemented in a food delivery android application, but it's important to note that the specific security measures implemented will depend on the requirements of the application and the sensitive data it handles.

3.5. **Detailed Class Diagram**

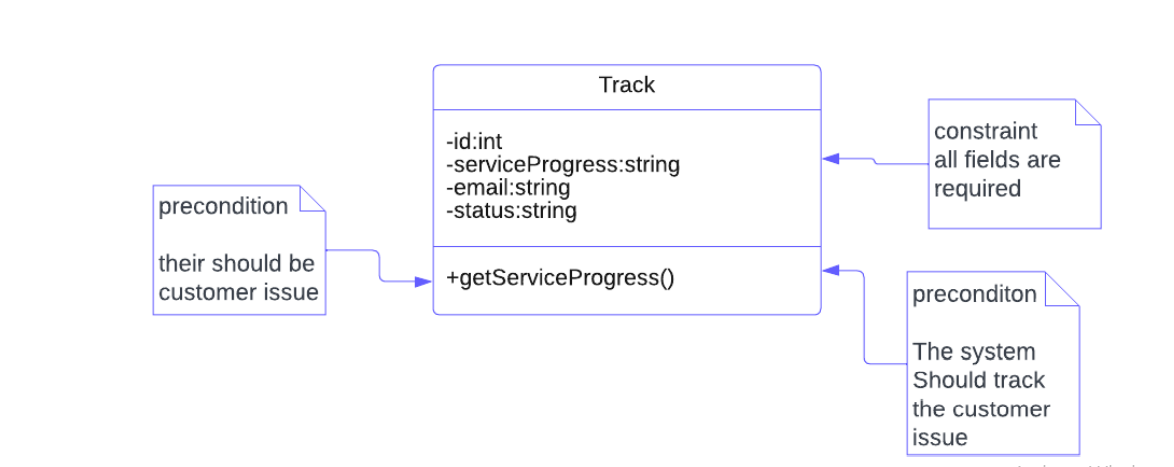


Figure 3.5.1 Track Detailed Class Diagram

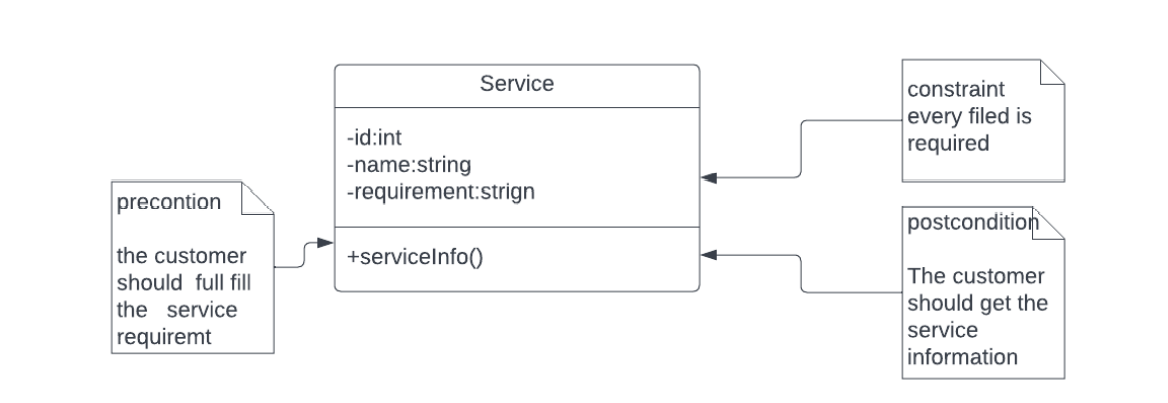


Figure 3.5.2 Service Detail Diagram

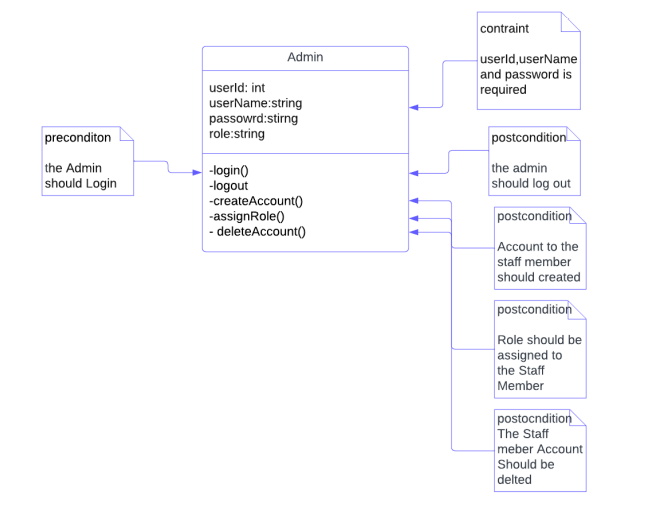


Figure 3.5.3 Admin Detailed Class Diagram

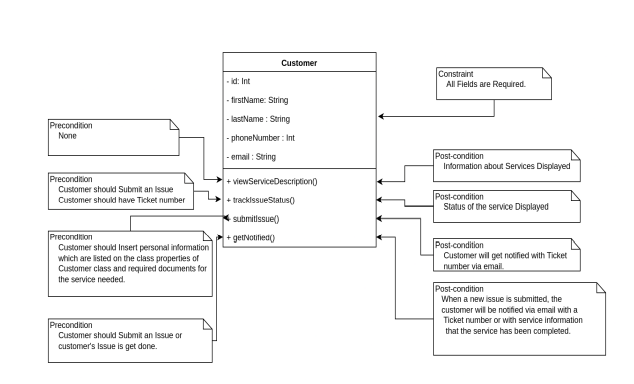


Figure 3.5.4 Customer Detailed Class Diagram

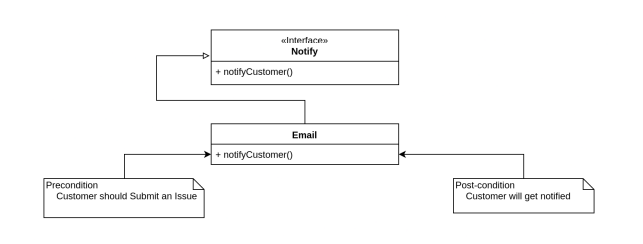


Figure 3.5.5 Notify Detailed Class Diagram

3.6**. Packages**

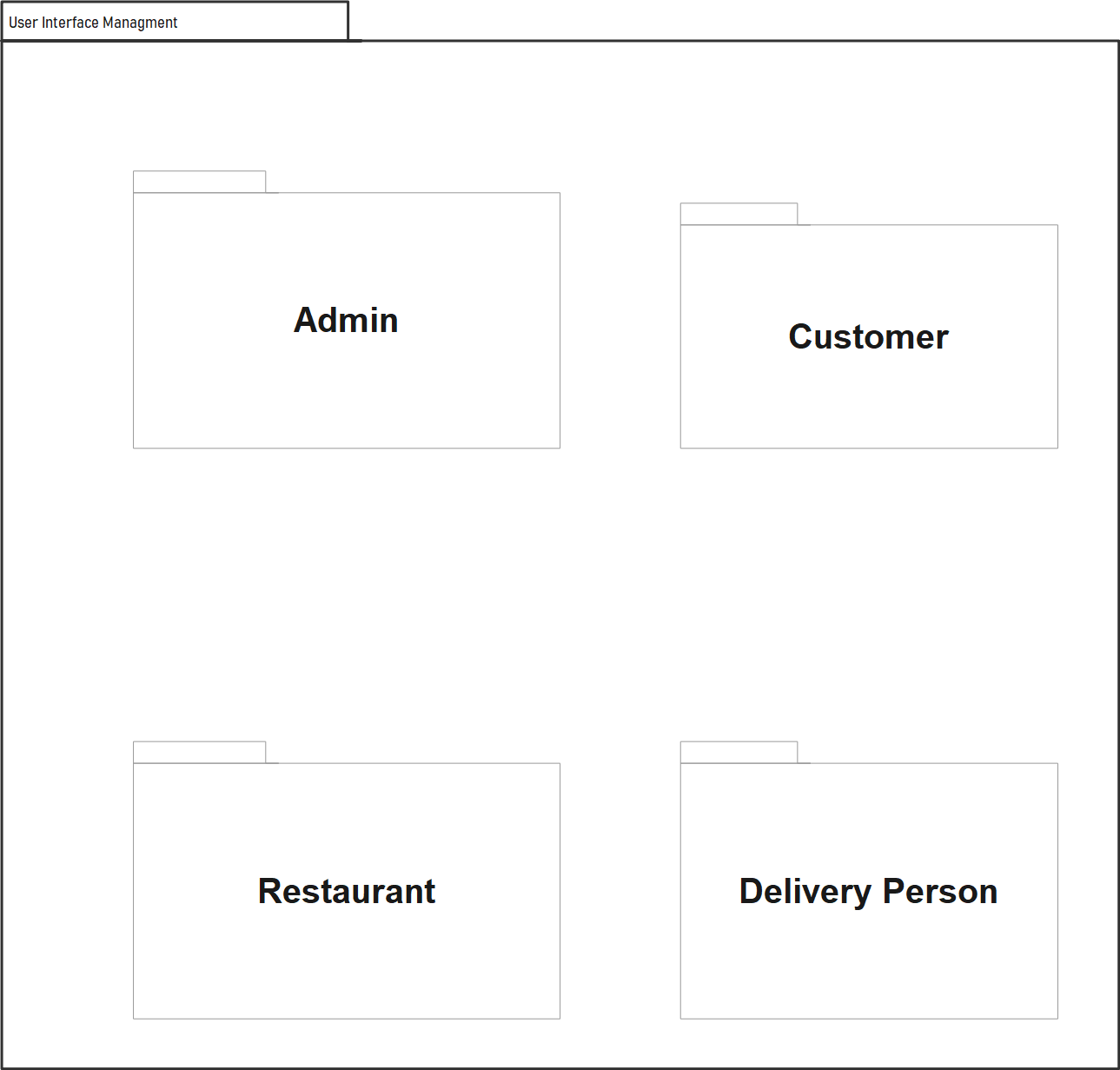
A package diagram in UML (Unified Modeling Language) is a type of structure diagram that shows the organization of packages and their dependencies. A package diagram for a food delivery Android application may include the following packages:

* User Interface Package: This package contains the graphical user interface (UI) elements and layouts that the user interacts with.
* Location Package: This package handles location-based services and maps integration, using APIs like Google Maps.
* Payment Package: This package implements the payment gateway integration and handling of online transactions.
* Order Package: This package manages the ordering process, including menu display, ordering, and order tracking.
* Notification Package: This package implements push notifications for order updates, delivery status, and other relevant information.
* Database Package: This package manages the database, including storing and retrieving customer information, order details, and delivery information.
* Backend Package: This package communicates with the backend server for data processing, management, and storage.

These packages may have dependencies on each other, for example, the Order Package may depend on the User Interface Package for displaying the menu and ordering process, and on the Database Package for storing and retrieving order information.

3.6.1 User Management Package

This package is responsible for administrator, customer, restaurant and delivery person interaction on user interface to login and It includes account creation and edit/update/delete.

  
Figure 3.6.1 User Management Package Diagram

3.6.2 Location management Package

Location Package: This package handles location-based services and maps integration, using APIs like Google Maps.

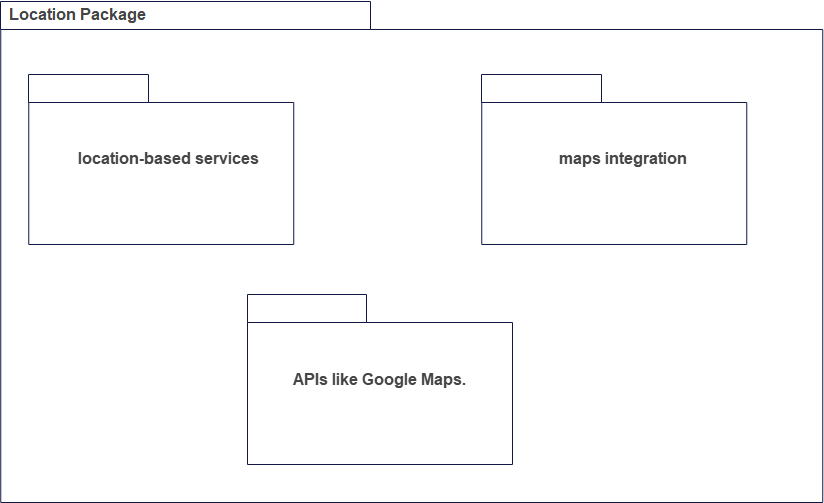


Figure 3.6.2 Location management Package

3.6.3 Data Management Package

The data management package contains classes responsible for data storage and information retrieval triggered by the subsystems. The following diagram is data management package of the system

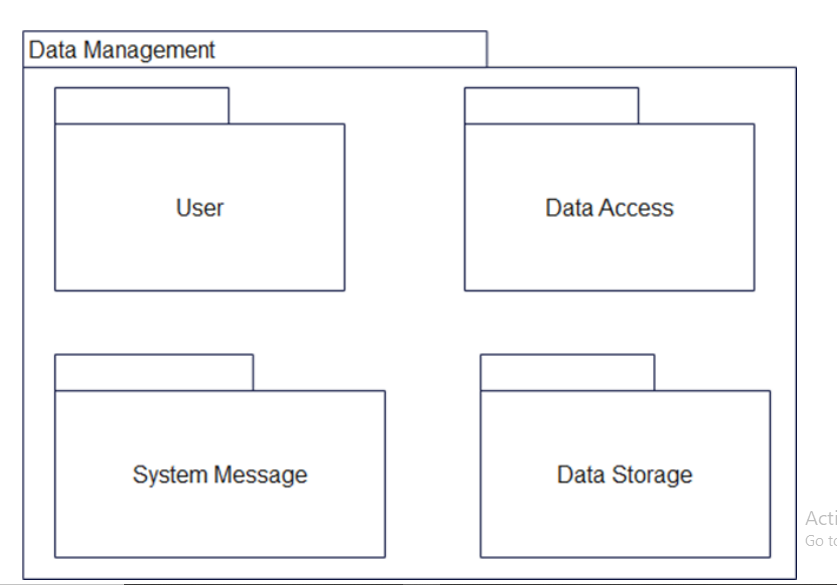


Figure 3.6.3 Data Management Package

3.6.4 Payment Package

Payment Package: This package implements the payment gateway integration and handling of online transactions

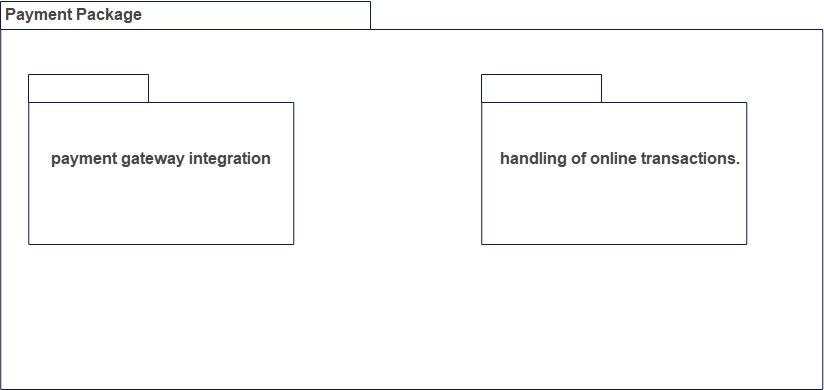


Figure 3.6.4 Payment Package

3.6.5 Order Package

Order Package: This package manages the ordering process, including menu display, ordering, and order tracking.

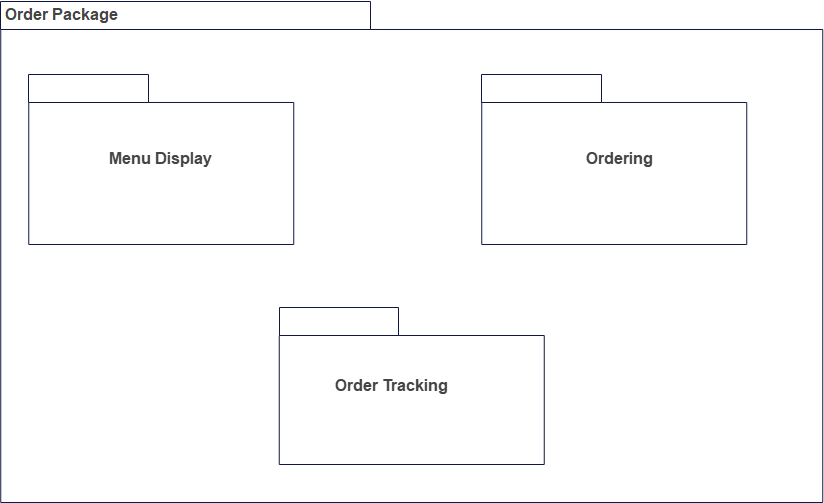


Figure 3.6.5 Order Package

3.6.6 Notification Package

Notification Package: This package implements push notifications for order updates, delivery status, and other relevant information.

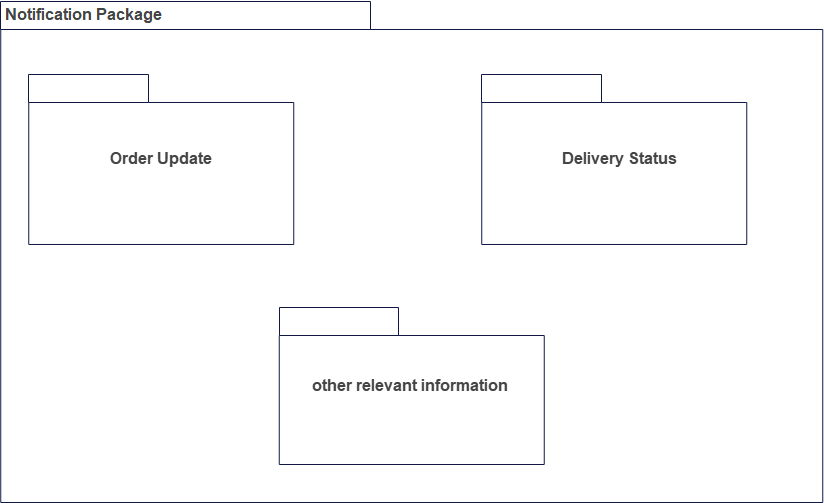


Figure 3.6.6 Notification Package

3.6.7 Backend Package

This package communicates with the backend server for data processing, management, and storage

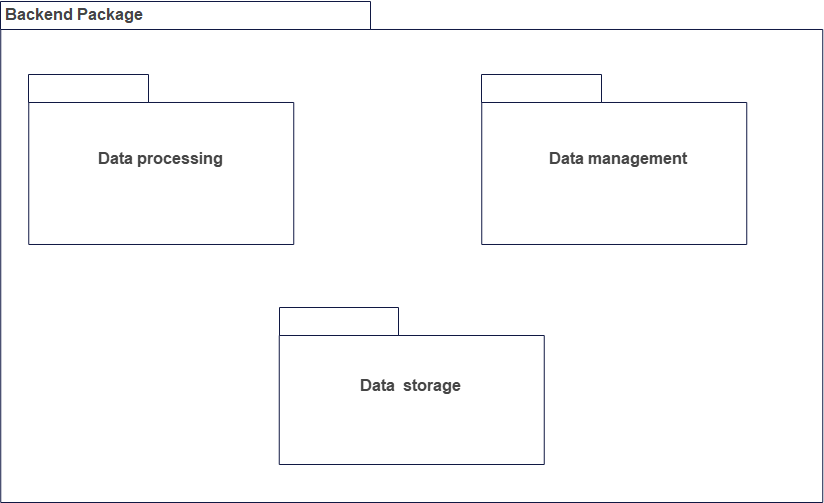


Figure 3.6.7 Backend Package

**Dependency among them**

A package diagram for a food delivery Android application may include the following dependencies:

* User Interface Package depends on the Order Package, Notification Package, and Location Package for displaying relevant information and updates to the user.
* Location Package depends on the Google Maps API or other location-based services for mapping and location tracking functionality.
* Payment Package depends on the Payment Gateway API for handling online transactions.
* Order Package depends on the User Interface Package for displaying menu and order information, the Payment Package for processing payments, and the Notification Package for sending updates.
* Notification Package depends on the Order Package and Backend Package for getting information on order status and delivery updates.
* Database Package depends on the Backend Package for data management and processing, and the Order Package for storing and retrieving order information.
* Backend Package depends on the Database Package for data storage and management, and the Order Package for processing order information.

These dependencies show the relationships between packages and how they interact and depend on each other to provide a seamless user experience for the food delivery application.

