

COMP 3059 – Capstone Project I

Software Requirements Analysis and Design Assignment

This assignment is an overview to gather the software needs with requirements analysis and help to proceed with the design.

The requirements analysis helps to break down functional and non-functional requirements to a basic design view to provide a clear system development process framework. It involves various entities, including business, stakeholders and technology requirements.

The design is the activity following requirements specification and before programming. Software design usually involves problem solving and planning a software solution.

To work on this assignment you could use the references and a sample template given below. The sample template can be customised to suit the nature of your project.

Reference Readings/Example:

http://www.uacg.bg/filebank/acadstaff/userfiles/publ_bg_397_SDP_activities_and_steps.pdf

www.cse.msu.edu/~chengb/RE-491/Papers/SRSEExample-webapp.doc

Source for this template:

www.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.doc

Rose

Requirements Analysis and Design template

1.0

November 1, 2024

Miguel Angel Gutierrez Serrano

Mehmet Ali KABA

Parisa Mohammadkarimi

Pornpajee Sunkkadithee

1.0 Revision History

Date	Description	Author	Comments
Nov 1, 2024	Version 1.0	Pornpajee	First Revision

1. Introduction

The Rose - Flower Delivery App aims to provide a comprehensive platform that facilitates the delivery of flowers and botanical products from local stores to customers. This document outlines the software requirements and design considerations necessary for developing the application, ensuring a user-friendly experience for customers, delivery personnel, and store owners. By conducting a thorough analysis of the software requirements, this document establishes a clear framework for the development process and the functionalities of the system, ultimately contributing to the successful implementation of the project.

1.1 Purpose

The purpose of this document is to present the high-level software requirements for the Rose - Flower Delivery App. It outlines the key functionalities and capabilities that the system will provide without delving into the technical details of how these functionalities will be implemented. The intended audiences for this document include stakeholders such as project team members, potential users, and investors, who require an understanding of the app's objectives and features to assess its viability and usability.

2.2 Scope

Rose - Flower Delivery App will enable users to easily order flowers and other botanical products from local stores with convenient delivery options. The key functionalities of the system include:

- **User Authentication:** Secure sign-up and login processes for customers, store owners, and delivery personnel.
- **Product Browsing and Search:** Users can view and search for available products from various stores.
- **Real-time Order Tracking:** Customers can track their orders in real-time, enhancing transparency and user experience.
- **Payment Integration:** The system will facilitate secure payment processing for orders.
- **Feedback and Ratings System:** Users can provide feedback on products and services, which will help improve quality and user satisfaction.

What the System Will Do:

- Provide a user-friendly interface for customers to order flowers and related products.
- Allow store owners to manage their inventory and receive orders.
- Enable delivery personnel to receive and manage delivery tasks efficiently.

What the System Will Not Do:

- The system will not handle the physical logistics of flower delivery; this will be managed by partnered delivery services.
- It will not provide features for ordering non-botanical products, focusing solely on flowers and related items.

Benefits, Objectives, and Goals:

The primary objective of the Rose - Flower Delivery App is to streamline the flower ordering and delivery process, making it accessible and efficient for users. Key benefits include:

- Increased convenience for customers wishing to send flowers.
- Support for local businesses by providing them with an online platform to reach a wider audience.
- Improved customer satisfaction through real-time tracking and feedback mechanisms.

2. System Overview

The System Overview section provides an introduction to the Rose - Flower Delivery App, outlining its context, design, constraints, and dependencies.

2.1 Project Perspective

Rose - Flower Delivery App is a **new self-contained system** designed to address the growing demand for online flower delivery services. It is not a replacement for existing systems but rather aims to fill a gap in the market by providing a seamless user experience for customers, store owners, and delivery personnel. The app will integrate various features tailored to the needs of each user group, enhancing the overall efficiency of the flower delivery process.

2.2 System Context

Rose - Flower Delivery App operates within the context of the floral retail and delivery industry, specifically targeting local stores and their customers. The strategic issues addressed by the system include:

- **Market Accessibility:** The app facilitates local florists' entry into the online market, allowing them to reach a broader customer base.
- **Customer Experience:** By providing a user-friendly interface, real-time tracking, and secure payment options, the app enhances the overall shopping experience for customers.
- **Operational Efficiency:** The app optimizes delivery logistics for delivery personnel, ensuring timely deliveries and efficient management of orders.

The app aligns with the business objectives of supporting local businesses, improving customer satisfaction, and increasing operational efficiencies within the flower delivery industry.

2.3 General Constraints

The following general constraints may impact the software development lifecycle for the Rose - Flower Delivery App:

- **Technical Constraints:** The app must be developed for both Android and iOS platforms, requiring adherence to specific design and development standards for each.
- **Budget Constraints:** The project must remain within the allocated budget, which will affect feature prioritization and resource allocation.
- **Compliance Requirements:** The app must comply with local regulations concerning data protection (e.g., GDPR) and online payment processing.
- **Time Constraints:** The project timeline is set to complete development by March 28, 2025, which will influence the project schedule and feature implementation.

2.4 Assumptions and Dependencies

The following assumptions have been made during the initiation of the project:

- It is assumed that there will be sufficient interest and demand for an online flower delivery service in the target market.
- The availability of technology and resources (e.g., developers, tools) to build and maintain the app is assumed to be adequate.
- It is assumed that local florists will be willing to partner with the app to offer their products.

The project also has dependencies that may impact its success:

- **Third-Party Services:** The app's functionality, such as payment processing and real-time tracking, depends on reliable third-party services that must be integrated effectively.
- **User Adoption:** The success of the app relies on user adoption rates among customers, store owners, and delivery personnel.
- **Market Competition:** The project's success may be influenced by the actions of existing competitors in the online flower delivery market.

3.0 Functional Requirements

This section outlines the specific features and functionalities of the Rose - Flower Delivery App, detailing how the system will operate to meet the needs of its users.

3.1 Functional Requirement

3.1.1 User Authentication (Sign Up, Login)

- Introduction

The user authentication feature enables users to securely create accounts and log in to the app. This functionality ensures that user data is protected and personalized experiences can be delivered.

- Inputs

For Sign Up:

- User's name
- Email address
- Password
- Phone number (optional)

For Login:

- Email address
- Password

- Processing

For Sign Up:

- Validate the uniqueness of the email address against existing accounts.
- Hash the password for security.
- Store user details in the database.
- Send a confirmation email to the user.

For Login:

- Retrieve user details from the database using the provided email address.
- Compare the entered password with the stored hashed password.
- If successful, generate a session token for the user.

- **Outputs**

For Sign Up:

- Success message or error message (e.g., "Email already in use").
- Confirmation email to the user.

For Login:

- Success message with user session details or error message (e.g., "Invalid email or password").
- Access to the user's dashboard upon successful login.

3.1.2 Customer App Interface Design

- **Introduction**

This feature encompasses the design and layout of the customer-facing app interface, focusing on usability and aesthetics to enhance the shopping experience.

- **Inputs**

- User navigation inputs (taps, swipes)
- Search queries for flowers or arrangements
- Preferences for delivery dates and times

- **Processing**

- Render the app's main interface based on user navigation.
- Confirmation of items added to the cart.
- Notifications for special offers or promotions.

- **Outputs**

- Dynamic interface displaying available flowers, prices, and details.
- Display search results filtered by user input.

3.1.3 Delivery Personnel App Interface Design

- **Introduction**

The delivery personnel app interface design focuses on optimizing the delivery process, providing essential tools for tracking orders and managing deliveries.

- **Inputs**

- Delivery personnel login credentials

- Delivery assignments received from the system
- User feedback inputs for completed deliveries

- **Processing**

- Display assigned deliveries with customer details and delivery locations.
- Provide navigation assistance to delivery personnel using integrated maps.
- Allow delivery personnel to update delivery status (e.g., "Out for Delivery," "Delivered").

- **Outputs**

- List of active deliveries and their statuses.
- Confirmation messages for updated delivery statuses.
- Notifications of new delivery assignments or changes.

3.2 Use Cases

This section outlines specific use cases that illustrate how different users will interact with the Rose - Flower Delivery App. Each use case describes a sequence of actions and the expected outcomes.

3.2.1 User Sign Up

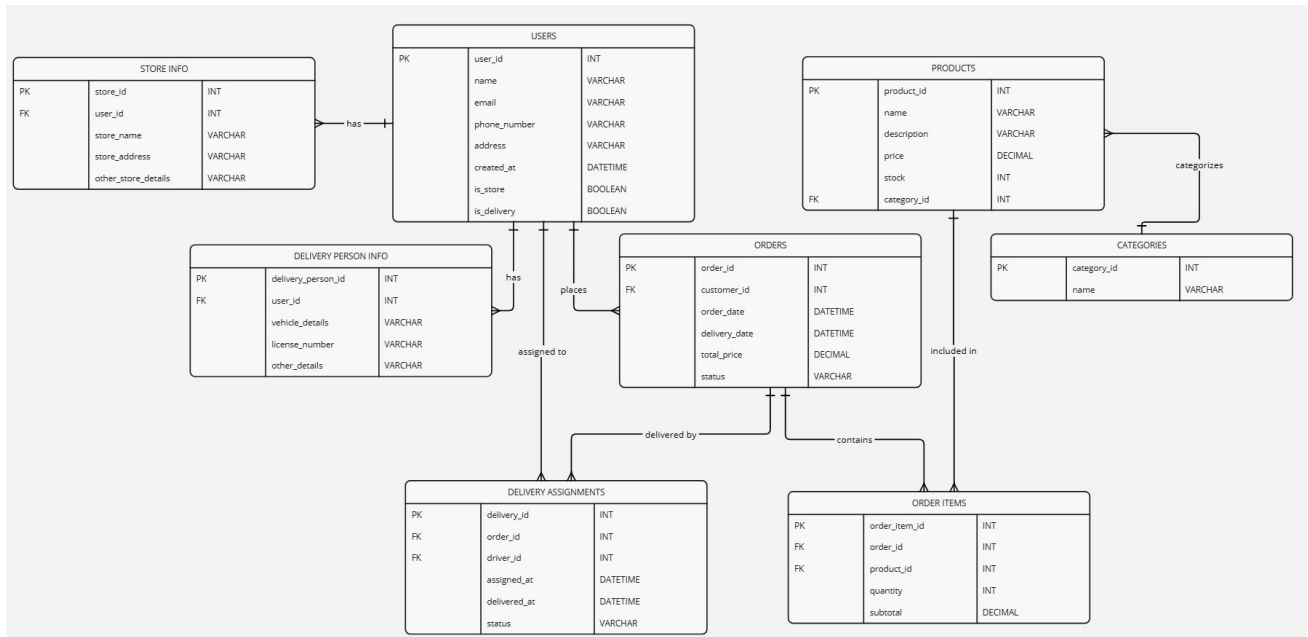
- **Actors:** Customer
- **Description:** This use case allows a new customer to create an account in the app.
- **Preconditions:**
The user must have access to the internet and the app.
- **Postconditions:**
A new user account is created, and the user receives a confirmation email.
- **Basic Flow:**
 1. The user opens the app and selects "Sign Up."
 2. The user enters their name, email address, password, and phone number (optional).
 3. The system validates the email address for uniqueness.
 4. The system hashes the password for security.
 5. The system stores the user details in the database.
 6. The system sends a confirmation email to the user.
 7. The user receives a success message and is redirected to the login screen.
- **Alternative Flows:**
 - **Email Already in Use:**
 1. If the email is already associated with an existing account, the system displays an error message.
 2. The user can choose to log in instead.

3.2.2 User Login

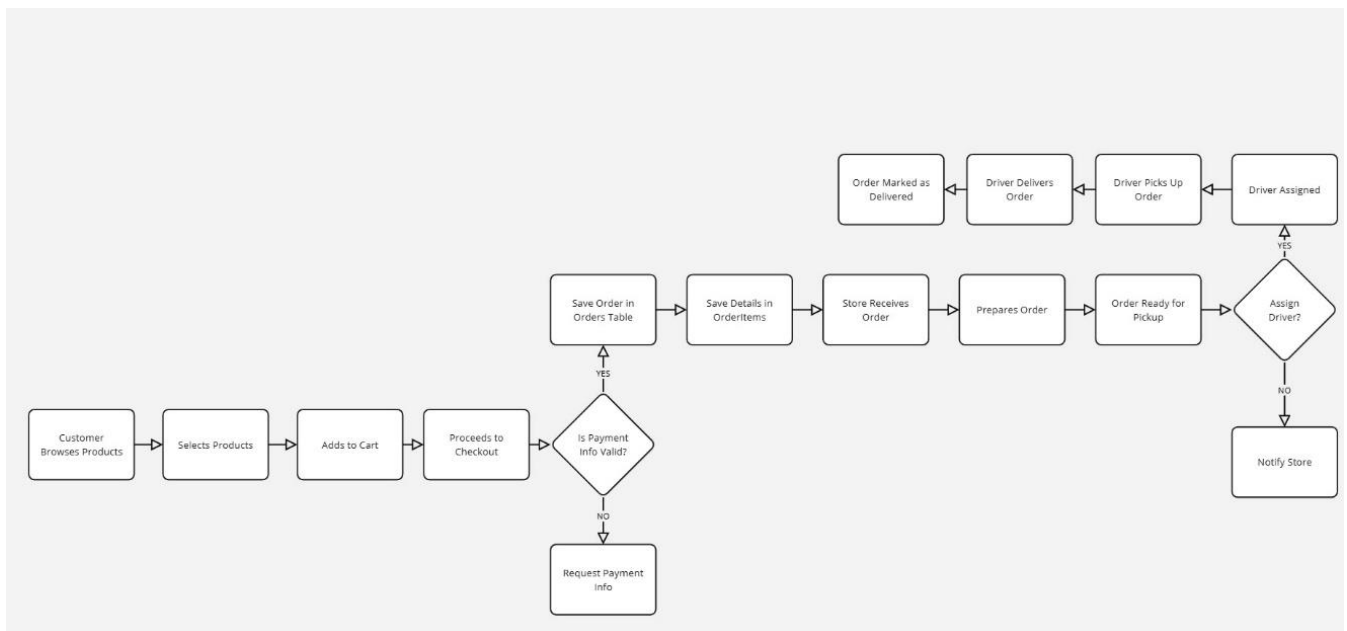
- **Actors:** Customer
- **Description:** This use case allows a registered customer to log in to the app.
- **Preconditions:**
The user must have an existing account.
- **Postconditions:**
The user is authenticated and granted access to their dashboard.
- **Basic Flow:**
 1. The user opens the app and selects "Login."
 2. The user enters their email address and password.
 3. The system retrieves the user details from the database.
 4. The system verifies the entered password against the stored hashed password.
 5. If successful, the system generates a session token and grants access to the user's dashboard.
 6. The user sees their dashboard with options to browse flowers, view orders, and manage account settings.
- **Alternative Flows:**
 - **Invalid Credentials:**
 1. If the entered email or password is incorrect, the system displays an error message.
 2. The user can choose to reset their password or try logging in again.

3.3 Data Modelling and Analysis

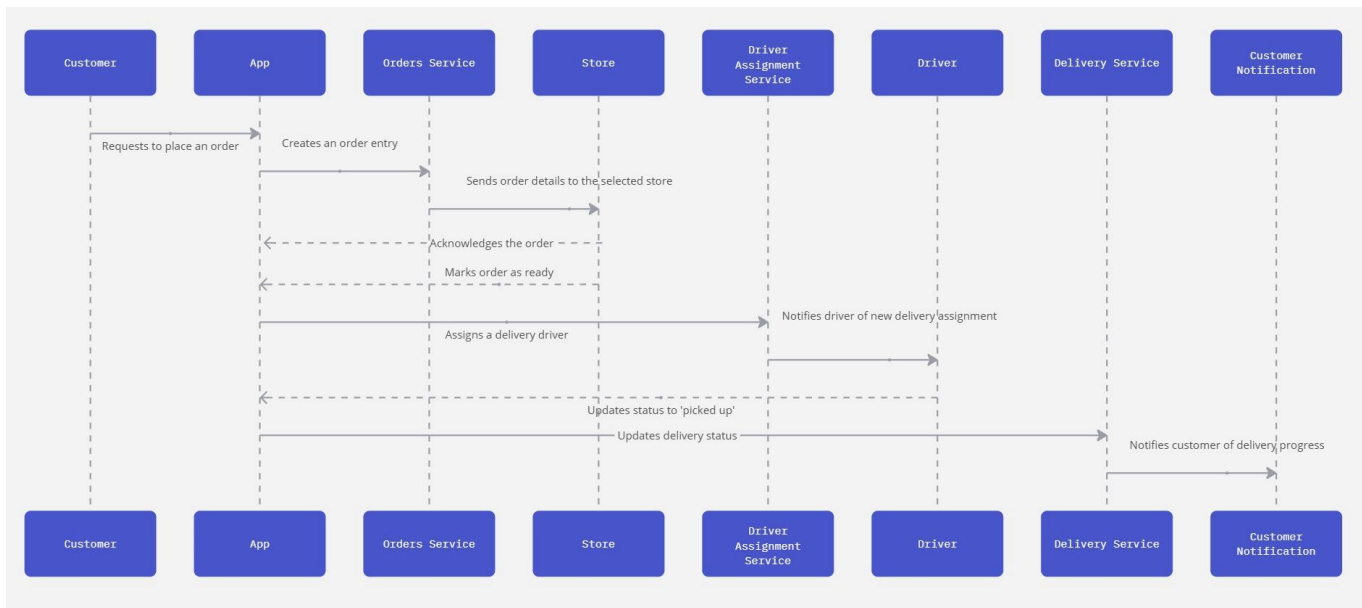
- Normalized Data Model Diagram



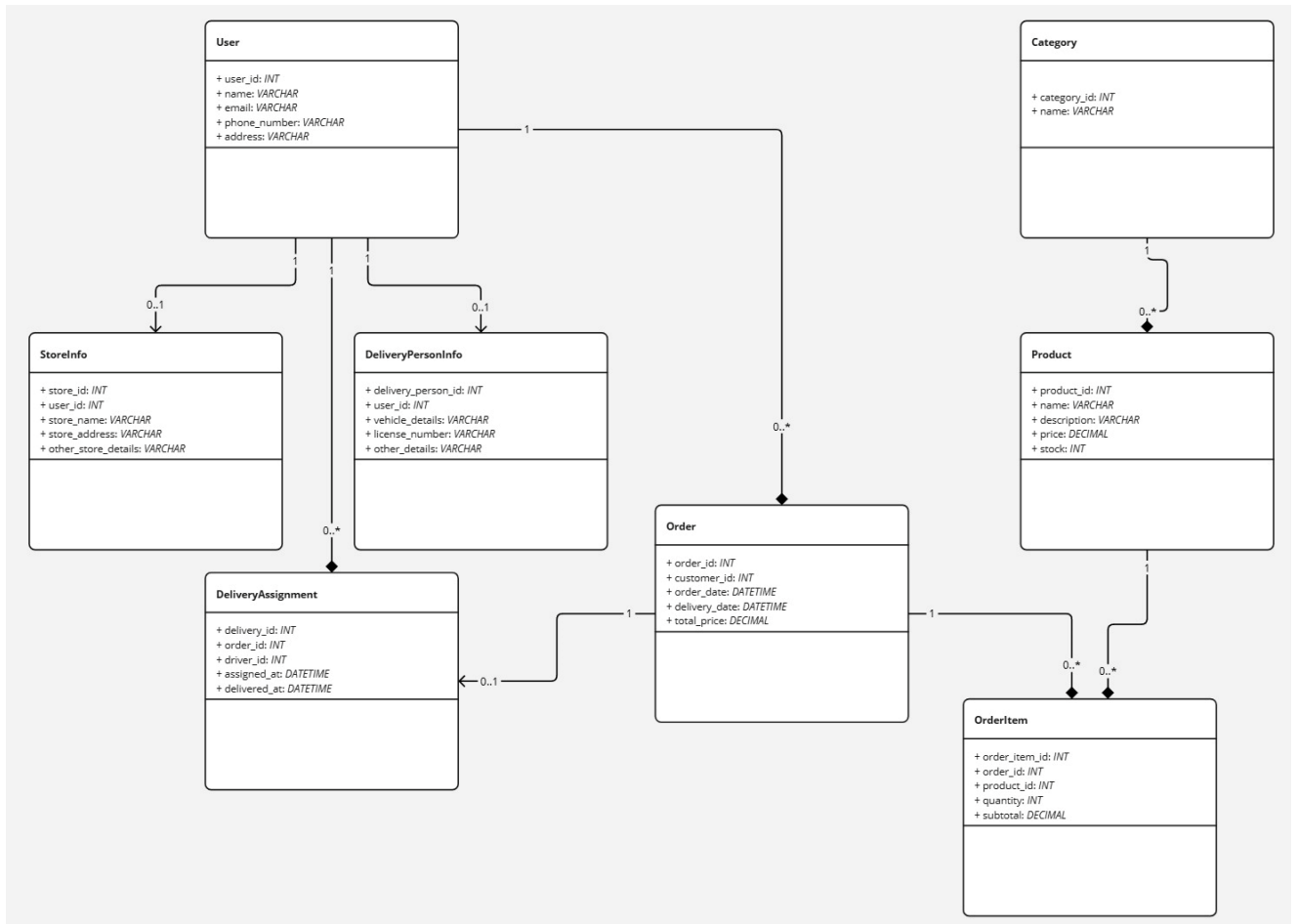
- Activity Diagrams



- Sequence Diagrams

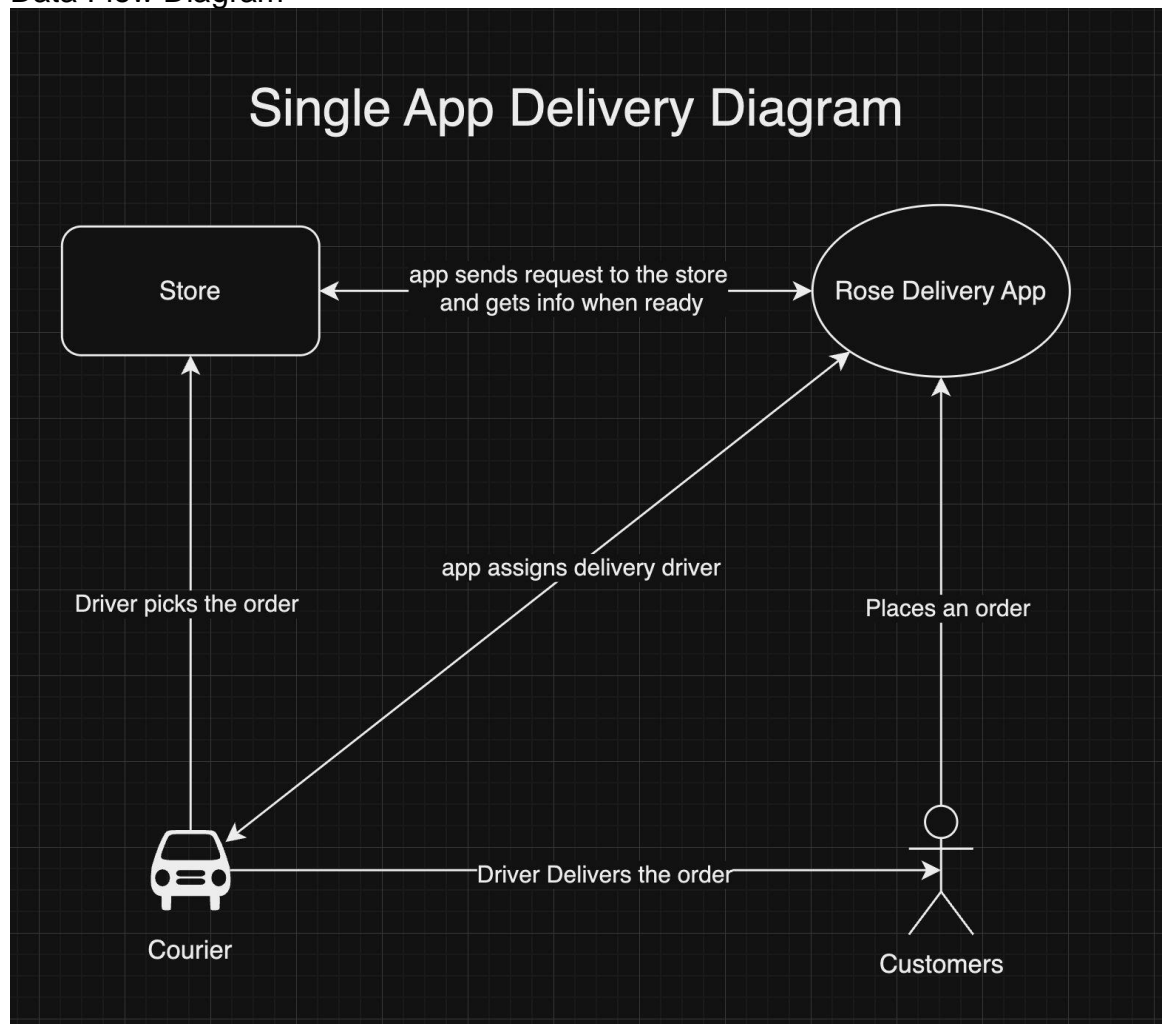


- UML Class Diagram



3.4 Process Modelling

- Data Flow Diagram



4.0 Non-Functional Requirements

4.0 Non-Functional Requirements

Non-functional requirements define how the system should operate in terms of performance, security, availability, and other quality attributes. These are detailed below with measurable terms:

4.1 Performance

- **Response Time:** The system should respond to user requests (such as product searches or catalog navigation) in less than 2 seconds in 95% of cases.
- **Order Processing Time:** Payment processing and order confirmation should complete within 5 seconds after the request.
- **Load Capacity:** The application must support at least 500 concurrent users without noticeable performance degradation.

4.2 Reliability

- **System Uptime:** The app should be available at least 99.9% of the time each month, with unscheduled downtime not exceeding 1 hour per month.
- **Fault Tolerance:** The system should handle individual component failures (such as payment services) without affecting the overall functionality of the app.
- **Data Backup and Recovery:** Data should be backed up daily, and the system must be able to restore data in the event of a failure within 15 minutes.

4.3 Availability

- **Accessibility:** The app should be accessible from any device with an internet connection and compatible with both iOS and Android systems.
- **Scheduled Maintenance:** System updates and maintenance should be scheduled outside peak hours (e.g., between midnight and 4:00 a.m.) to minimize user impact.

4.4 Security

- **Data Protection:** User and transaction information must be encrypted both in transit and at rest, using standards such as SSL/TLS for communications and AES for storage.
- **Authentication and Authorization:** The app should implement multi-factor authentication (MFA) for store accounts and delivery personnel, with strong password requirements for users.
- **Compliance:** The system must comply with local and international data protection regulations, such as GDPR, especially for the collection and storage of user data.

4.5 Maintainability

- **Updates:** The system must allow for software updates without requiring downtime.
- **Documentation:** All code should be documented, and industry best practices should be followed to facilitate future maintenance and improvements.
- **Modularity:** The app's architecture should be modular, enabling changes in specific modules (e.g., payments or authentication) without affecting other components.

4.6 Portability

- **Cross-Platform Compatibility:** The app should run smoothly on both iOS and Android devices, using an adaptive design optimized for various resolutions and screen sizes.
- **Cloud Scalability:** The system should be able to scale horizontally to support an increase in users using cloud infrastructure.

5.0 Logical Database Requirements

• Data Formats:

- MySQL supports structured data formats, which is ideal for relational data, such as customer profiles, order details, and product information.
- MySQL's JSON data type can handle semi-structured data, allowing for flexible storage of any dynamic information (such as user preferences or feedback).

• Storage Capabilities:

- MySQL is highly scalable, providing the capacity to handle large datasets, including an expanding inventory of products, a growing user base, and numerous transaction records.
- It supports indexing, which will be applied to frequently queried fields, such as `customer_id`, `order_id`, and `product_id`, to optimize query performance and ensure quick response times.

• Data Retention:

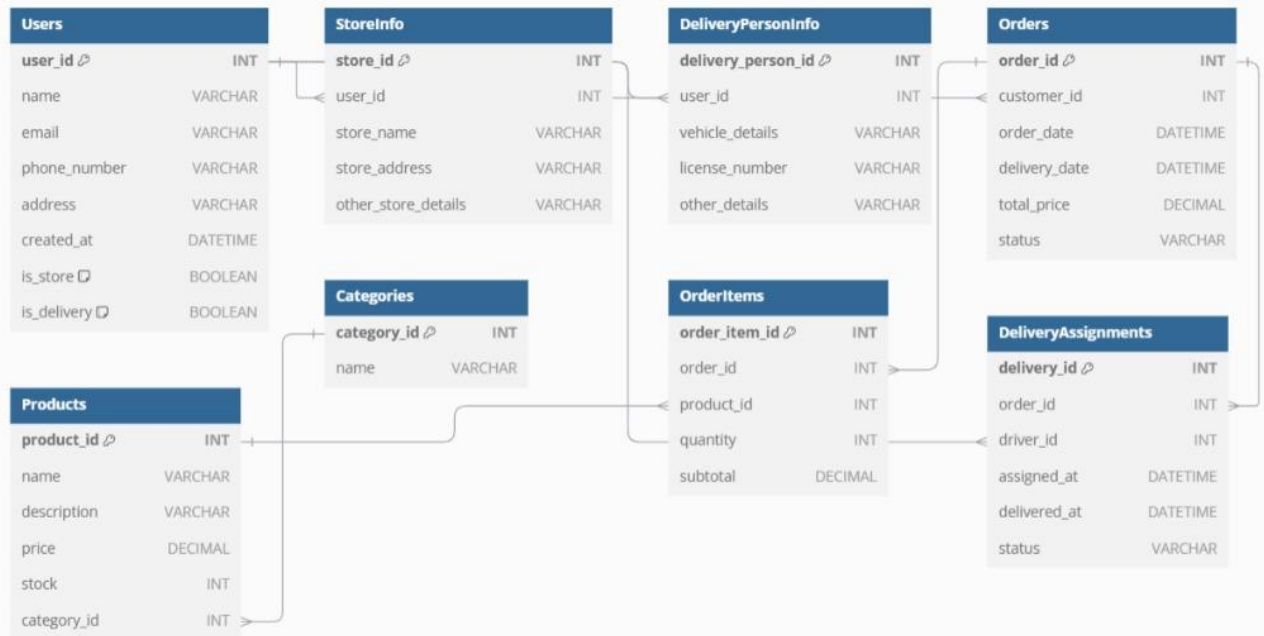
- Retention policies will be implemented to manage the lifecycle of transactional records. For instance, completed order records may be archived after a set period to keep the database manageable while ensuring historical data is available for analytics or compliance purposes.
- MySQL supports partitioning, which can be used to manage large datasets over time and enhance performance.

• Data Integrity

- MySQL's robust transaction management and integrity constraints ensure reliability for critical data operations, such as order placement and payment processing. This setup helps maintain consistency and accuracy, ensuring that all transactions are processed completely and accurately, even in cases of system interruptions.
- To maintain data accuracy, foreign key constraints and relational integrity will be implemented to link customers, orders, and delivery information appropriately.





• Concurrency and Transaction Management

- Using MySQL's InnoDB engine, the app will manage high concurrency with support for multiple simultaneous interactions, such as browsing, ordering, and delivery updates, without data conflicts.



6.0 Approval

The signatures below indicate their approval of the contents of this document.

Project Role	Name	Signature	Date
Lead Developer	Mehmet Ali KABA		Nov 1, 2024
Project Manager	Miguel Angel Gutierrez		Nov 1, 2024
Business Analyst	Pornpajee Sunkkadithee		Nov 1, 2024
QA Specialist	Parisa Mohammadkarimi		Nov 1, 2024