SOEN 343
Software Architecture and Design

Delivery Service Application
Sprint 1

Team Name: Omnivox 2

Hrag Bankian (40245363) - Users (CRUD)
Gregory Demirdjian (40249882) - Shipping and tracking
Elian Gadbois-Roy (40208293) - Database
Alec Kirakossian (40244852) - Chatbot
Sevag Merdkhanian (40247912) - Orders (CRUD)
Leon Robert-Hosein (40092184) - User interface and design

September 26, 2024

Table of Contents

1. Project Definition

1.1 Objectives

The main purpose of the Omnivox 2: Delivery Service Web Application is to facilitate the process of managing orders and their delivery by customers and service providers. Particularly, the system will provide an interface where customers can create requests for delivery, track packages, and manage payments.

- Eliminate lost-order inefficiencies with real-time tracking and updates.
- Smoothen communication between customers and service providers by using integrated messaging and notifications.
- Offer real-time customer support with a chatbot, answering any inquiries customers might have.

1.2 Defined method of approach

The objectives set previously serve as a guideline on the approach the team will take towards the Delivery Service Application, as described below.

- **Agile Methodology:** The project will use the principles of Agile with iterative sprints to allow continuous feedback and adaptation.
- **Development Responsibilities:** Each team member has been assigned roles according to their area of expertise. The roles are separated by features that need to be implemented. All work will be peer reviewed before being pushed to production to ensure a product of utmost quality.

Table 1: **Developer Tasks and Responsibilities**

Name	Responsability
Hrag Bankian	Users (CRUD)
Gregory Demirdjian	Shipping and tracking
Elian Gadbois-Roy	Database
Alec Kirakossian	Chatbot
Sevag Merdkhanian	Orders (CRUD)
Leon Robert-Hosein	User interface and design

Table 2: **Project Timeline and Deliverables**

Sprint #	Estimated Completion Date	Deliverables
1	September 26, 2024	Problem definition, context diagram and domain model
2	October 19, 2024	System architecture with sequence diagram
3	November 15, 2024	Implementation of core features
4	November 30, 2024	Product finalization and final report

1.3 Project scope

The scope of this project involves designing the architecture for a delivery service application that addresses the gap between existing delivery services and the desired system-to-be. The focus will be on implementing the core features, including delivery requests, service quotes, communication, order tracking, payment handling, and customer support through a chatbot interface. Standard design principles and coding conventions will be reinforced.

2. Problem Definition

2.1 What is the problem?

The problem that is being addressed revolves around order management inefficiencies. The lack of a centralized platform for shipping makes keeping track of one or more orders challenging. This causes orders to potentially be lost, duplicated, or mismanaged.

2.2 How did the problem emerge?

This problem emerged as buyers and sellers have trouble keeping track of multiple orders, across multiple platforms. Moreover, manual processes make it easy to lose track of orders. This may be the case when a company is scaling up.

2.3 What is our solution?

The solution we are providing is a centralized web application that simplifies order management and delivery. Customers can place orders, track shipments in real time, and manage orders all on one platform. The system will integrate payment processing, ensuring payments are synced with orders, eliminating discrepancies. This centralized approach reduces the need for manual tracking and minimizes errors.

2.4 What are the advantages of our solution?

The application’s main advantages are its accessibility, ease of use, and efficiency. Being responsive, users can access it from any device. Its simple interface makes it user-friendly, and the centralization of order tracking and payments prevents confusion and mistakes. Real-time updates provide transparency, while automated processes further reduce errors, resulting in a more efficient and reliable system.

3. Technology Used

3.1 Team Collaboration

Discord will be the primary platform for team discussions, communication, and standups. It allows for text, voice, and video communication, ensuring seamless coordination between team members.

3.2 Monitoring and Verification

The team will use version control tools provided by Github to track changes, monitor progress, and verify code accuracy. Regular code reviews and testing will ensure that each component meets the project's standards and specifications. All work will be peer reviewed before being pushed to production to ensure a product of utmost quality. The functionality of the application will be regularly tested with unit tests and integration testing. Optimizing SQL queries will be done regularly in database operations to ensure the performance of the system under load.

3.3 Design and Modeling Work

Draw.io will be utilized to create diagrams and models for the system, which includes a context diagram and a domain model. This tool is ideal for visualizing system architecture and relationships between different components.

3.4 Interface

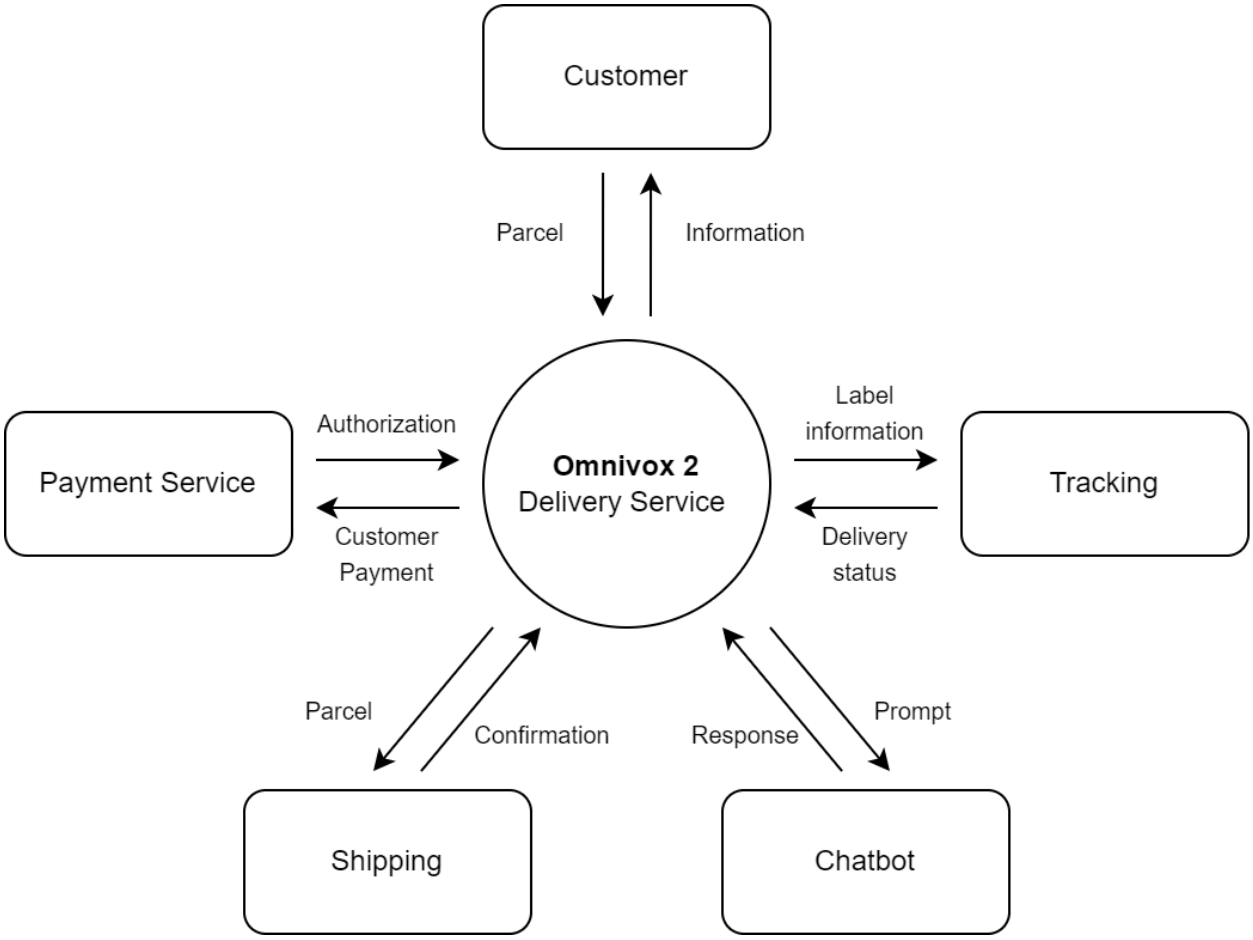
The frontend of the application will be developed using React, a popular JavaScript library for building user interfaces. React’s component-based architecture allows for efficient development, reusable components, and smooth user interactions. The interface will be responsive, ensuring usability across devices.

Additionally, Next.js, a React framework, will be used to enhance the development process by providing server-side rendering (SSR) and static site generation (SSG) capabilities. This will improve performance, SEO, and the overall user experience. Next.js simplifies routing and API integration, making the application more scalable and efficient.

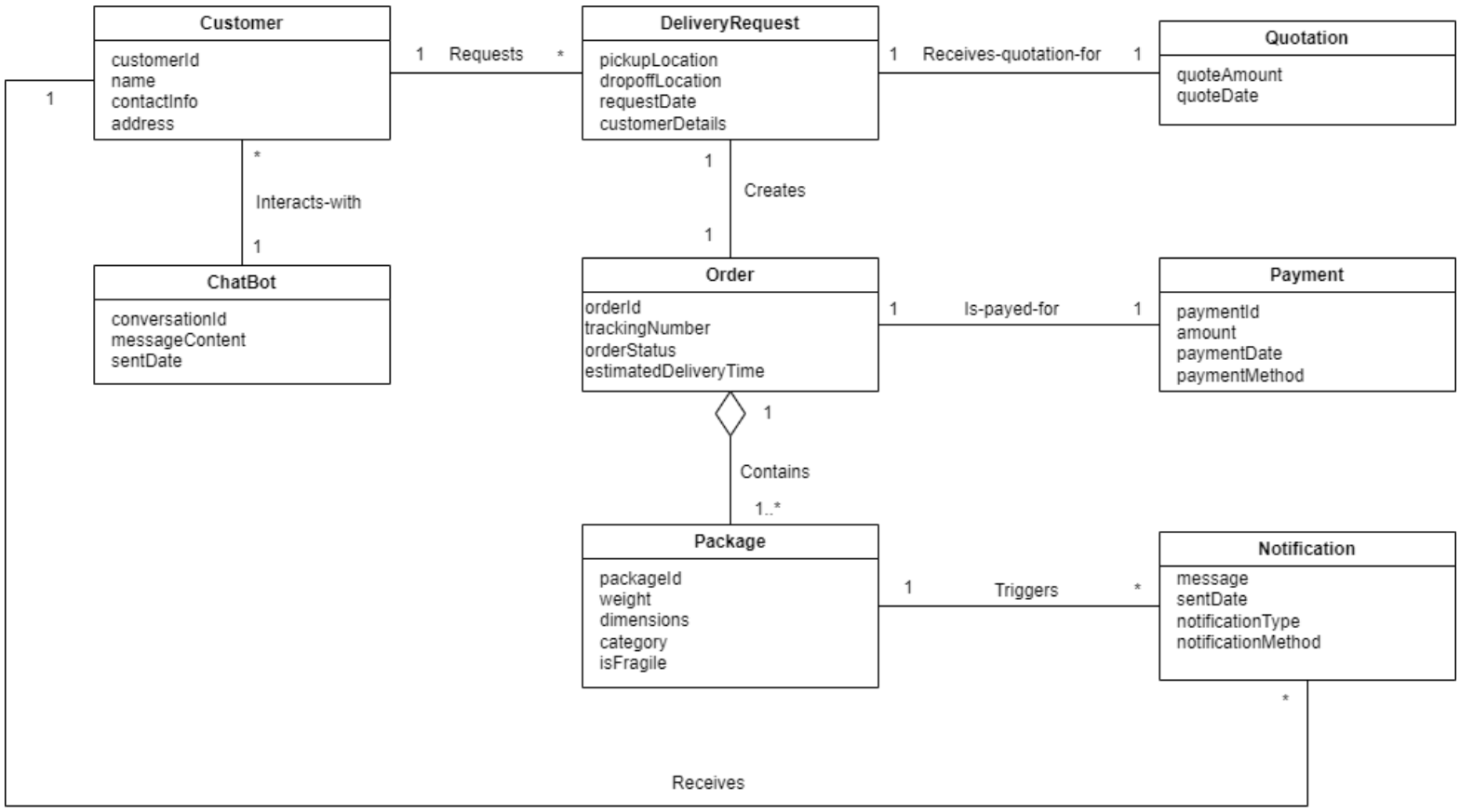
3.5 Coding

For the backend, C# with Entity Framework Core (.net6) will be used, which is a widely adopted framework for building robust applications. Entity Framework Core simplifies database interactions, allowing for object-relational mapping (ORM) with SQL, which will be used to manage data storage and queries efficiently. This tech stack ensures a solid foundation for implementing the core features of the delivery service, including request handling, tracking, and payments.

4. Context Diagram



5. Domain Model



6. References

Fedex - Ship, Manage, Track, Deliver. Fedex. (n.d.). https://www.fedex.com/en-ca/home.html

DHL - global logistics and international shipping. DHL. (n.d.). https://www.dhl.com/ca-en/home.html