MARMARA UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE 3044 – Software Engineering Term Project

QuickRoute

Software Design Specification

Group 17

150119667 Ege Eren Ellez
150119675 Hakan Kenar
150119695 Mustafa Kerem Ekinci
150119703 Mert Akbal
150517059 Özge Saltan
150120531 Hasan Şenyurt
150119906 Zahit Erdem Güzel
150119798 Eşref Emre Koca

Prepared for

CSE3044 Software Engineering Term Project

Table of Contents

1. Introduction

1.1.Purpose

The purpose of this document is to explain our project, which is an application of cargo delivery for customers. It is called QuickRoute. This document describes the software requirements of the system, general description, user interfaces and UML diagrams. The goal of this project is to create a mobile application that makes it easier for clients and couriers to receive their packages. Customers are demanding faster and more convenient delivery options in today's fast-paced world, and courier services need to improve their delivery routes to stay competitive. The suggested mobile application provides a response to these two issues.

Customers can quickly order cargo delivery through the mobile application, selecting their chosen delivery location, and paying for the service within the app. They can also receive reminders regarding the status and anticipated arrival time of their delivery, allowing them to follow its progress in real-time. Overall, this smartphone app strives to increase the convenience and effectiveness of cargo delivery for both clients and couriers, offering a useful resource for both individuals and enterprises.

1.2.Statement of Scope

The scope of a mobile application for cargo delivery which is name QuickRoute with the following characteristics is included in the project's scope:

- Users must register an account in the application by providing personal information such as their name, phone number, and email address.
- The software will pair users with the closest available couriers depending on how close their locations are to each other. When an order is accepted, customers can choose a delivery location, which will be displayed to the courier. Users of the application will be able to submit orders for delivery. The order will then be delivered to nearby couriers who are available. Depending on their schedule and workload, the courier may accept or refuse the request.
- Clients can follow the progress of their orders in real time, getting updates on the courier's whereabouts and anticipated delivery time. The app will provide a safe and practical payment processing system that will let users pay for deliveries right inside the app. Customers will be able to rate the courier and offer feedback on their experience receiving deliveries.

- The software will be created for both the iOS and Android operating systems, and it will work with a variety of mobile devices. Modern software development processes, tools, and technologies will be used throughout the project to provide a high-quality and user-friendly application.
- The project's scope also includes creating an admin panel for managing couriers and customers in addition to the mobile application for cargo delivery. The following functionalities will be available in the admin panel: The admin panel will enable control of the accounts of the couriers, including their availability, personal information, and workload. Additionally, it will enable the evaluation of courier performance, including ratings and delivery timeframes.
- You may manage customer accounts, including their personal data, order history, and payment information, using the admin panel. A consolidated place for managing orders, including the ability to view and update orders, track deliveries, and handle payments, will be made available through the admin panel. A secure login will be required to access the web-based admin panel. Its interface will be simple and contemporary, and it will be made to be user-friendly and intuitive. To provide a top-notch and useful admin panel, the project will make use of contemporary software development processes, tools, and technologies.

1.3.Software Context

Operating system: The QuickRoute program can be used with browsers that are compatible with Android, iOS, Windows, and macOS as well as mobile devices and web-based applications.

- **Web server:** The program needs a web server to manage and respond to incoming HTTP requests from users.
- **Database server:** The application needs a database server to maintain information about users, couriers, and past orders.
- **Payment gateway:** To handle user payments, the program interfaces with a payment gateway.
- **Services for mapping and tracking:** The program connects with mapping and tracking services to track the whereabouts of couriers and give users real-time updates.
- **Notification services:** The application requires notification services to alert users about the status of their orders and deliveries.

- **Development tools:** The application is developed using various software tools such as programming languages, development frameworks, and libraries.
- **Internet connectivity:** The application requires internet connectivity to process orders, payments, and track couriers in real-time.

1.4.Major Constraints

- **Technical constraints:** During the development of the application, restrictions that determine the technical requirements of the application should be taken into consideration. Specific programming languages, development platforms, database management systems, and other technological specifications and limits may be among these restrictions.
- **Performance constraints:** Constraints on performance: User happiness may be directly impacted by the application's performance. As a result, performance limitations for quick processing, precise tracking, dependable database operations, and quick payment processing should be found.
- **Budget constraints:** During the application development process, budgetary restrictions should also be taken into account. Budget restrictions for the application's creation, release, promotion, upkeep, and updating should be noted.
- **Time constraints:** Time constraints should be identified to ensure the application is delivered on time and completed within the defined budget and performance constraints.
- **Usability constraints:** Usability constraints should be identified for the usability, speed, performance, efficiency, and suitability of the application.

1.5.Definitions

- UI: User Interface
- Customer: A person that can give an order to courier.
- Courier: A person that will carry the package of customer to given location.

1.6. Acronyms and Abbreviations

- HTML: Stands for HyperText Markup Language
- CSS: a stylesheet language used to describe the presentation of a document written in HTML or XML.
- TypeScript: A programming language used on the client-side.

- Angular: A JavaScript/Typescript framework.
- MySQL: Open-source relational database management system used to store and manage data for various applications.
- Java Spring Boot: Open-source framework used to quickly and easily create standalone, production-grade Spring-based applications.
- DBMS: Database Management System

1.7. References

[1] IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998, 1998. Available: https://ieeexplore.ieee.org/document/720574. Accessed on: Apr. 9, 2023.

2. Design Consideration

2.1.Design Assumptions and Dependencies

Assumptions:

- To use the program, users must have a smartphone or tablet with an Android 5.0 operating system or above and internet connectivity.
- In order to register for an account and receive updates regarding their courier requests, users must have a working email address and phone number.
- Users will correctly enter the courier requests' pick-up and delivery addresses.
- Users will accurately state the package's dimensions and weight to guarantee the courier is qualified to deliver the goods.
- The application's payment options will be accepted by users.

Dependencies:

- In the application, Java Spring Boot will be used for the backend and MySQL will be used to hold the databases.
- The application will depend on the GPS location of the users' smartphones to determine the pick-up and delivery locations accurately.
- The application will depend on Google Maps API to display the route of the courier and the estimated time of arrival to the destination.

- The application will depend on the Stripe payment system for secure payment processing.
- The application will require access to the camera and photo library of the users' smartphones to upload images of the package.
- The application will depend on push notifications to update users about their courier requests.

Possible Changes in Functionality:

- To satisfy user preferences and needs, the program might need to incorporate new payment options.
- To adjust to modifications in operating systems or hardware requirements, the application can need updates.
- To improve the user experience and satisfy shifting market expectations, the program might need to include new features.
- The creation, deployment, and effective use of the QuickRoute courier application relies on certain presumptions and dependencies.

2.2. General Constraints

Hardware and Software Environment:

The application will be designed to operate on smartphones and tablets with Android 5.0 operating systems and above.

The application will require internet access to function properly. In the application, Java Spring Boot will be used for the backend and MySQL will be used to hold the databases.

End-User Environment:

The application's end users will need access to a smartphone or tablet with one of the aforementioned operating systems as well as the internet. The pick-up and delivery addresses for courier requests, as well as the size and weight of the parcel, should be accurately supplied by the end users.

Availability and Volatility of Resources:

For the application to operate correctly, internet and GPS services must be available. Due to system upkeep or updates, the program could encounter brief interruptions.

Standards Compliance:

The application will comply with relevant industry standards and regulations, including data protection and privacy laws.

Performance Requirements:

With a response time of no more than two seconds, the program must react quickly to user input. The maximum number of concurrent users that the application may support at any given moment is 500.

Security Requirements:

To safeguard user privacy and restrict access, the application should have security features like encryption and password protection. The application should abide by all applicable security laws.

Interface/Protocol Requirements:

Users should be able to explore and use the application's functionalities with ease thanks to its user-friendly and intuitive design. To handle financial transactions, the app has to have a secure payment system.

Verification and Validation Requirements:

To ensure the functioning and performance of the application, extensive testing and validation should be performed. The design of the QuickRoute courier program is significantly impacted by these broad limits. To guarantee its functioning, performance, and adherence to pertinent standards and laws, the application's design must adhere to these restrictions.

2.3.System Environment

With a mobile application running on the client side and a cloud-based server handling backend processing, the courier application will function in a client-server architecture. The system environment requirements for the application are as follows:

Hardware:

Mobile devices running Android 5.0 and cloud server with a minimum of 2 CPU cores, 4GB RAM, and 50GB storage.

Software:

- Developed using the Angular framework for mobile applications
- Built with the Java Spring, server-side applications
- MySQL is a database management system.
- Google Maps API provides functions related to location and routing
- Gateway for payments: Stripe

Network:

For internet connectivity, mobile devices need to have access to cellular data or Wi-Fi. A dependable internet connection with a minimum bandwidth of 5 Mbps is required for the server.

Security:

- Using JSON Web Tokens (JWT), users can authenticate and authorize themselves.
- Using HTTPS for data encryption in transit
- Using the built-in encryption tools of MySQL, encrypt data at rest
- For the system and its data to be confidential, intact, and available, the system environment must adhere to the most recent security best practices and industry standards.

2.4. Development Methods

Agile technique, notably Scrum, will be used as the development approach for the courier application. Planning, development, testing, and review phases will all be included in each of the project's two-week sprints. The Scrum framework will enable ongoing evaluation and adaptation in response to adjustments in requirements or user input.

A central repository will be maintained on GitHub, and Git will be used to manage code versioning. Pull requests will be used to combine modifications made to the main branch when each team member has finished working on their own branch. CircleCI will be used to manage continuous integration and deployment, with automated testing and deployment to a staging environment for additional testing before release to production.

At the conclusion of each sprint, regular user acceptance testing (UAT) will be performed to make sure the application satisfies the users' needs and requirements. The backlog will be adjusted and future development efforts will be given priority based on feedback from UAT.

We will be able to create high-quality software that satisfies user requirements and adapts to shifting needs and priorities thanks to the Agile Scrum approach.

3. Architectural and component-level design

A description of the program architecture is presented.

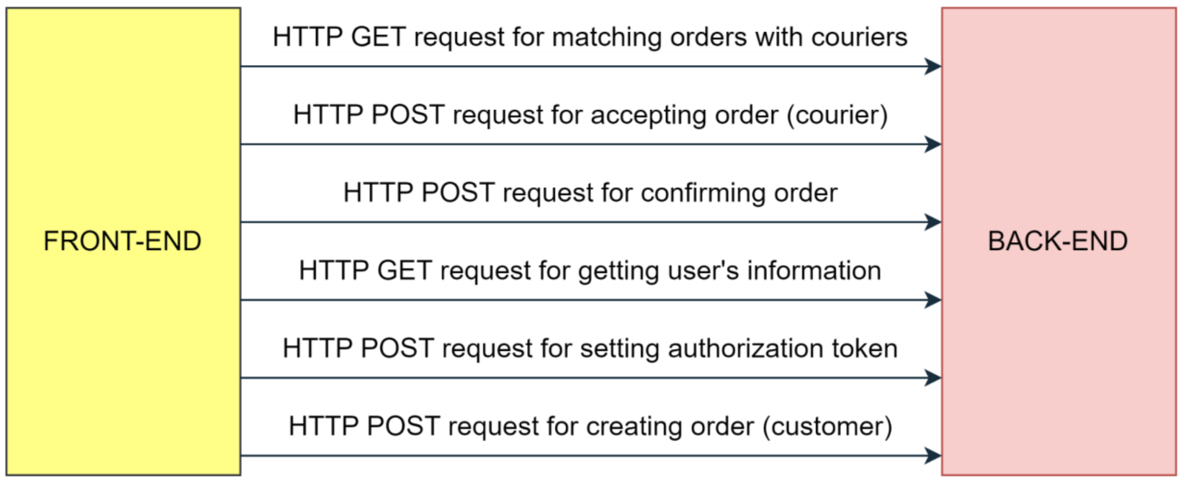
FRONT-END
HTTP GET request for matching orders with couriers
HTTP POST request for accepting order (courier)
HTTP POST request for confirming order
HTTP GET request for getting user's information
HTTP POST request for setting authorization token
HTTP POST request for creating order (customer)
BACK-END

Figure 1 - The Flow between Front-end and Back-end part of the QuickRoute Application

3.1. System Structure

The application follows a client-server architecture, with a mobile frontend for users and a backend server that processes requests and manages data. The backend server communicates with external services, such as mapping and tracking services, payment gateways, and notification services.

3.1.1. Architecture diagram

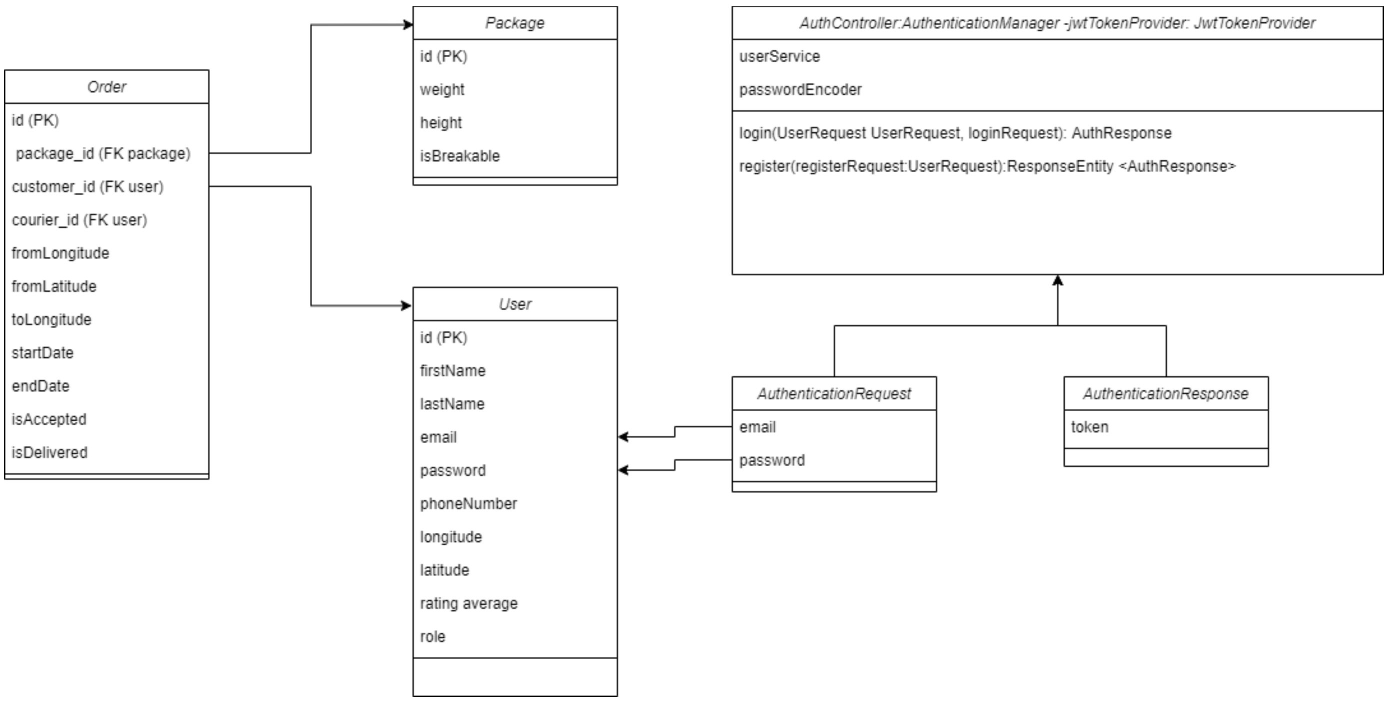

Figure 2 - Architecture Diagram of QuickRoute Application

3.2.Description for Component n

3.2.1. User Component

The User component manages user-related data, such as personal information, authentication, and roles (admin, customer, courier).

1. Processing Narrative (PSPEC)

The User component is responsible for managing user-related data and functions. This component performs the following tasks:

1. Registering new users by collecting their personal information, such as first name, last name, email, password, phone number, and role.
2. Authenticating existing users by verifying their email and password.
3. Updating user information, such as address (longitude and latitude), rating average (for couriers), and role (admin, customer, courier).
4. Retrieving user data for display in user profiles or administrative purposes.

2. Interface Description

Input Interfaces:

1. Registration form: Collects user's personal information, such as first name, last name, email, password, phone number, and role.
2. Login form: Collects user's email and password for authentication.

Output Interfaces:

1. User profile: Displays user's personal information, such as name, email, phone number, and role.
2. Authentication status: Indicates whether the user's login attempt was successful or not.

3. Processing Detail

3.1 Design Class Hierarchy for User Component

- BaseEntity
    - User

3.2 Restrictions/Limitations for User Component

- Users must provide a unique email address during registration.
- Passwords must meet minimum complexity requirements (e.g., length, mix of characters).
- User roles are limited to admin, customer, and courier.

3.3 Performance Issues for User Component

- Ensuring quick response times during user registration and authentication processes.
- Managing potential bottlenecks during high traffic periods.

3.4 Design Constraints for User Component

- Comply with data protection regulations (e.g., GDPR, CCPA) for handling user data.
- Ensure secure storage of sensitive user information, such as passwords.

3. 5 Processing Detail for Each Operation of User Component

- Register User:
    1. Validate user input.
    2. Check if the email is unique.
    3. Hash the user's password.

    4. Save the user to the database.
    5. Return the created user object.
- Authenticate User:
    1. Validate user input.
    2. Retrieve the user from the database by email.
    3. Verify the provided password against the stored hash.
    4. Return an authentication token if successful, or an error message if not.
- Update User Information:
    1. Validate user input.
    2. Retrieve the user from the database.
    3. Update user information.
    4. Save the updated user to the database.
    5. Return the updated user object.
- Retrieve User Data:
    1. Retrieve the user from the database by user ID.
    2. Return the user object or an error message if not found.

3.2.2. Package Component

The Package component manages package-related data, such as weight, height, and whether the package is breakable.

1. Processing Narrative (PSPEC)

The Package component is responsible for managing package-related data and functions. This component performs the following tasks:

1. Creating new packages by collecting package information, such as weight, height, and whether the package is breakable.
2. Updating package information, such as weight, height, and breakability status.
3. Retrieving package data for display in order details or administrative purposes.

2. Interface Description

Input Interfaces:

1. Package creation form: Collects package information, such as weight, height, and whether the package is breakable.
2. Package modification form: Allows users to update package information, such as weight, height, and breakability status.

Output Interfaces:

1. Package details: Displays package information, such as weight, height, and breakability status, in order details or administrative interfaces.

3. Processing Detail

3.1 Design Class Hierarchy for Package Component

- BaseEntity
    - Package

3.2 Restrictions/Limitations for Package Component

- Package weight, height, and breakability status must be provided during package creation.
- Package weight and height must be within acceptable limits (e.g., maximum weight, maximum dimensions).

3.3 Performance Issues for Package Component

- Managing potential bottlenecks during high traffic periods when creating or updating packages.
- Ensuring quick response times during package data retrieval.

3.4 Design Constraints for Package Component

- Design the database schema to efficiently store and retrieve package information.
- Ensure compatibility with various delivery services and their specific package requirements.

3.5 Processing Detail for Each Operation of Package Component

- Create Package:
    1. Validate package input.
    2. Save the package to the database.
    3. Return the created package object.

- Update Package Information:
    1. Validate package input.
    2. Retrieve the package from the database.
    3. Update package information.
    4. Save the updated package to the database.
    5. Return the updated package object.
- Retrieve Package Data:
    1. Retrieve the package from the database by package ID.
    2. Return the package object or an error message if not found.

3.2.3. Order Component

The Order component manages order-related data, such as package and user information, delivery locations, and order status (accepted, delivered).

1. Processing Narrative (PSPEC)

The Order component is responsible for managing order-related data and functions. This component performs the following tasks:

1. Creating new orders by collecting order information, such as package details, customer information, courier information, delivery locations (from/to), and start and end dates.
2. Updating order status, such as isAccepted (whether a courier has accepted the order) and isDelivered (whether the order has been delivered).
3. Retrieving order data for display in user interfaces, tracking purposes, or administrative purposes.

2. Interface Description

Input Interfaces:

1. Order placement form: Collects order information, such as package details, customer information, courier information, delivery locations (from/to), and start and end dates.
2. Order status update form: Allows users or couriers to update the order status, such as isAccepted and isDelivered.

Output Interfaces:

1. Order details: Displays order information, such as package and user information, delivery locations, and order status (accepted, delivered), in user interfaces or administrative interfaces.

3. Processing Detail

3.1 Design Class Hierarchy for Order Component

- BaseEntity
    - Order

3.2 Restrictions/Limitations for Order Component

- Orders must include valid package, customer, and courier information.
- Delivery locations must be within supported areas.
- Orders can only be updated by authorized users (e.g., customers, couriers, or admins).

3.3 Performance Issues for Order Component

- Managing potential bottlenecks during high traffic periods when creating or updating orders.
- Ensuring quick response times during order data retrieval and status updates.

3.4 Design Constraints for Order Component

- Design the database schema to efficiently store and retrieve order information.
- Ensure compatibility with various mapping and tracking services for accurate delivery location information and real-time tracking.

3.5 Processing Detail for Each Operation of Order Component

- Create Order:
    1. Validate order input, including package, customer, and courier information.
    2. Save the order to the database.
    3. Return the created order object.

- Update Order Status:
    1. Validate user authorization.
    2. Retrieve the order from the database.
    3. Update the order status (isAccepted, isDelivered).
    4. Save the updated order to the database.
    5. Return the updated order object.
- Retrieve Order Data:
    1. Retrieve the order from the database by order ID.
    2. Return the order object or an error message if not found.

3.2.4. Authentication Component

The Authentication component handles user authentication, generating tokens for secure communication between the client and the server.

1. Processing Narrative (PSPEC)

The Authentication component is responsible for handling user authentication and securing communication between the client and the server. This component performs the following tasks:

1. Authenticating users by verifying their email and password.
2. Generating secure tokens for authenticated users, which are used for subsequent communication between the client and the server.
3. Validating tokens for each request to ensure secure and authorized access to the application's resources.

2. Interface Description

Input Interfaces:

1. User login request: Processes the user's email and password for authentication.
2. Token validation request: Validates the secure tokens for each request to ensure secure and authorized access to the application's resources.

Output Interfaces:

1. Authentication status: Indicates whether the user's login attempt was successful or not.
2. Generated tokens: Provides secure tokens for authenticated users, which are used for subsequent communication between the client and the server.

3. Processing Detail

3.1 Design Class Hierarchy for Authentication Component

- BaseEntity
    - AuthenticationRequest
    - AuthenticationResponse

3.2 Restrictions/Limitations for Authentication Component

- Authentication tokens must have a limited lifespan to minimize security risks.
- Users must be authenticated for each request to access secured resources.

3.3 Performance Issues for Authentication Component

- Ensuring quick response times during user authentication and token generation/validation processes.
- Managing potential bottlenecks during high traffic periods when authenticating users and validating tokens.

3.4 Design Constraints for Authentication Component

- Use secure algorithms and industry-standard practices for password hashing and token generation.
- Implement token validation to protect against unauthorized access to secured resources.

3.5 Processing Detail for Each Operation of Authentication Component

- Authenticate User:
    1. Validate user input.
    2. Retrieve the user from the database by email.
    3. Verify the provided password against the stored hash.
    4. Generate an authentication token if successful, or return an error message if not.
    5. Return the authentication token or error message.

- Validate Token:
    1. Retrieve the token from the request header.
    2. Verify the token's validity and expiration.
    3. Return an error message and reject the request if the token is invalid or expired, or allow access to secured resources if valid.

3.3. Dynamic Behaviour for Components

User, Package, and Order components interact to manage the process of placing and tracking orders. The Authentication component ensures secure communication between clients and the server.

3.3.1. User Component

The User component interacts with other components such as the Order component and the Authentication component. When a user registers, their information is stored in the User entity. Users can log in using the Authentication component, and their authentication status determines their access to the application's resources.
Customers can create orders using the Order component, which associates the order with the customer's user account. Couriers can accept orders and update the delivery status, which in turn updates the related order in the Order component.

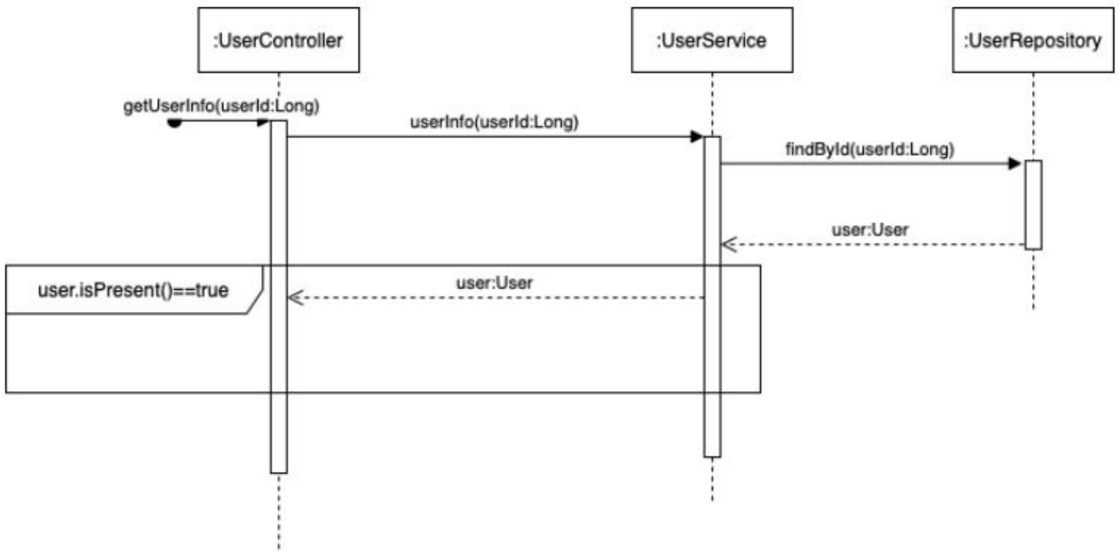

Figure 3 - Dynamic Behaviour of User Component - HTTP GET Request

3.3.2. Package Component

The Package component is primarily associated with the Order component. When a customer creates an order, they must provide package details such as weight, height, and breakability status. These details are stored in the Package entity, and the Order component retrieves this information to display it within the order details.

3.3.3. Order Component

The Order component interacts with the User and Package components. When a customer creates an order, their information is retrieved from the User component, and the package details are retrieved from the Package component. The Order component also interacts with external services such as mapping and tracking services to provide real-time updates on the delivery status.

3.3.4. Authentication Component

The Authentication component interacts with the User component to verify user credentials during login attempts. When a user logs in, their email and password are checked against the stored user information. If the credentials match, the Authentication component generates a secure token, which is used for subsequent requests to secured resources. The Authentication component also validates these tokens to ensure that only authorized users can access the application's resources.

3.3.1. Interaction Diagrams

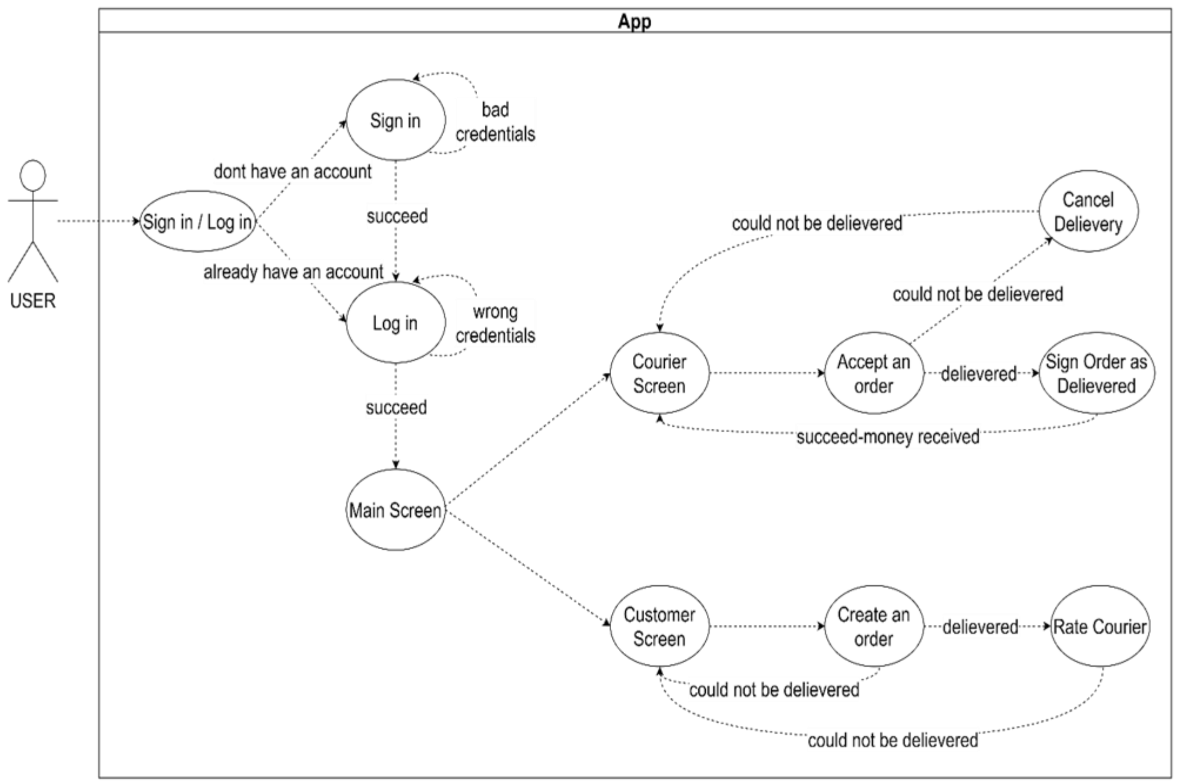

Figure 4 - Interaction Diagram of QuickRoute Application

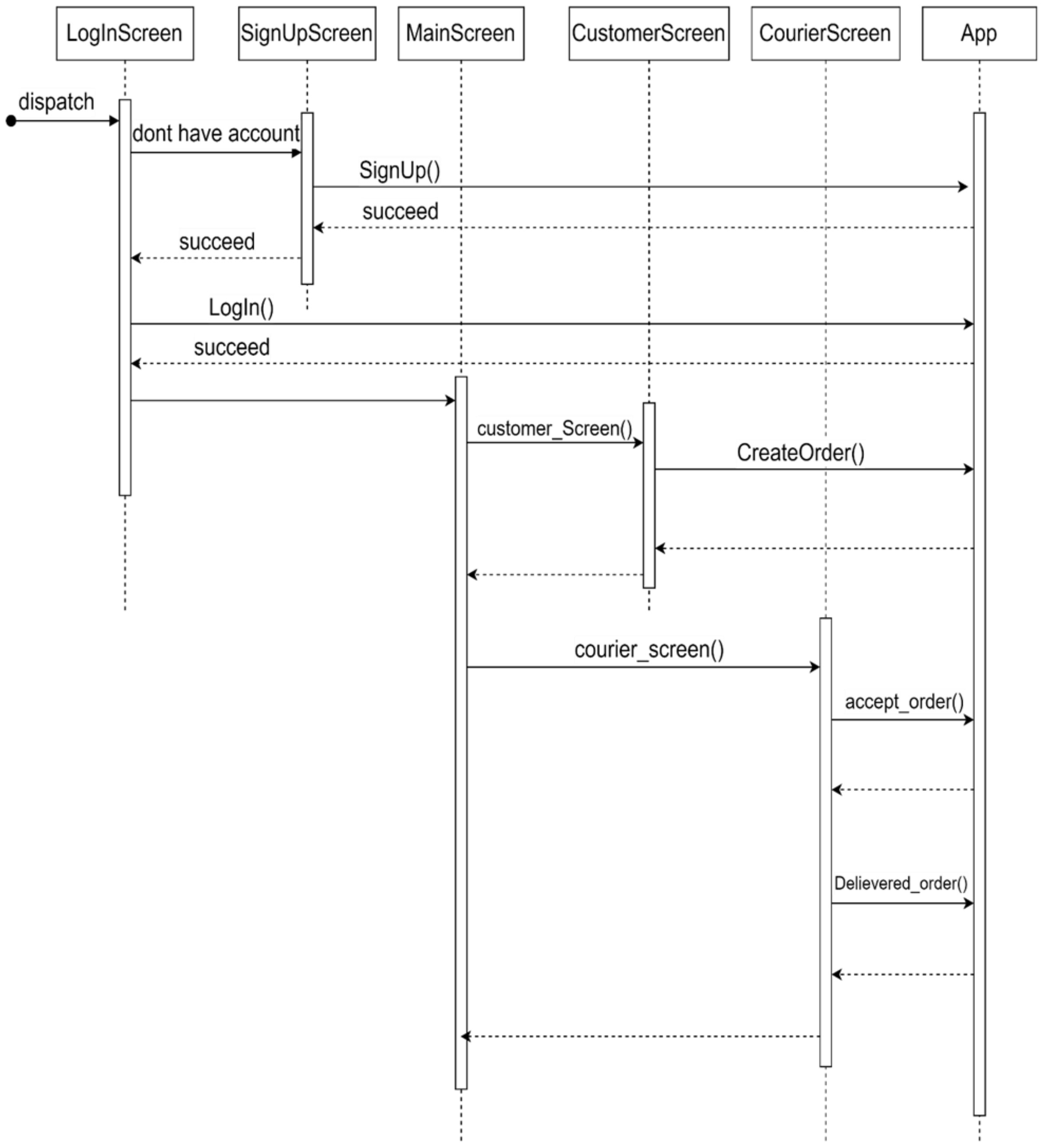

Figure 5 - UML Sequence Diagram of QuickRoute Application

4. Restrictions, limitations, and constraints

- **Operating Systems:** The courier app will be made for mobile devices, specifically those running the iOS and Android operating systems.
- **Network Connectivity:** For the courier program to work correctly, there must be a reliable internet connection. Users will need to have access to Wi-Fi or a mobile data plan.
- **Geographical restrictions:** Only specific cities or regions where the service is provided will have access to the courier application. Prior to spreading, the initial launch will be restricted to a few significant cities.
- **Language Support:** The courier application's initial release will only support English. Future editions might add support for more languages.
- **Security:** In order to safeguard user data and stop illegal access, the courier application must follow industry-standard security procedures. To secure user information, this will necessitate the implementation of encryption and other security measures.
- **Services from Other Parties:** The courier application may rely on other parties to provide certain features, such as processing payments. The overall functionality of the program may be impacted by the dependability and availability of these services.
- **Device Compatibility:** To ensure widespread user acceptance, the courier application must work with a variety of mobile devices. Older or less popular devices may encounter compatibility problems, which could negatively affect the user experience as a whole.
- **Regulations:** The courier application must abide by all applicable local laws and ordinances, including tax and licensing rules, that apply to courier services. Legal repercussions and reputational harm to the business could occur from failure to adhere to these regulations.

In order to ensure that the courier application satisfies user needs while functioning within the confines of the existing technology and regulatory environment, these restrictions, limitations, and constraints will need to be taken into account throughout the development process.

5. Conclusion

We are planning to make QuickRoute, a website and mobile application created by a student group. It will be a platform where people who want to work as a courier can work without a license. In the project, we will try to plan the customer - courier connection in the best way and as soon as possible. For our project, we tried to create it with the main headings "Consideration, Architectural and Component-level design and Restrictions, limitations, and constraints". In addition, a lot of the information we learned in the course while doing this project was useful for us. In addition, thanks to the SRS and DSD documents we learned in the lesson, we will do the project, it will be much easier for us to get our project ready. As a result, we tried to make our QuickRoute application what it experienced during this time.

6. Contribution of the Group Members to Software Design Specification Document

	Group Member	**Contribution**
1	Özge Saltan 150517059	Dynamic Behaviour for Component n (3.3), Conclusion (5)
2	Eşref Emre Koca 150119798	System Structure (3.1), Description for Component n (3.2)
3	Hasan Şenyurt 150120531	Introduction (1), System Structure (3.1)
4	Hakan Kenar 150119675	Introduction (1), Design Consideration (2)
5	Ege Eren Ellez 150119667	Design Consideration (2), System Structure (3.1)
6	Mert Akbal 150119703	Dynamic Behaviour for Component n (3.3), Restrictions, limitations, and constraints (4)
7	Zahit Erdem Güzel 150119906	System Structure (3.1), Description for Component n (3.2), Dynamic Behaviour for Component n (3.3)
8	Mustafa Kerem Ekinci 150119695	Design Consideration (2), Dynamic Behavior for Component n (3.3)