Bilkent University

Department of Computer Engineering

Senior Design Project

Project name: Wheelancer

High-Level Design Report

Onat Korkmaz
Muharrem Berk Yıldız
Muhammed Maruf Şatır
Ümit Çivi
Erdem Ege Eroğlu

Supervisor: Fazlı Can
Jury Members: Shervin Rahimzadeh Arashloo, Hamdi Dibeklioğlu
Innovation Expert: Murat Ergun

December 24, 2021

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

1. Introduction

The shipping sector has developed significantly in the last decades due to the advancements in transportation and other technologies such as e-commercing. All of these advancements led to a dramatic increase in the amount of cargo that has been shipped by individuals and companies. In order to tackle this huge demand, many companies that focus on the delivery of cargo have been founded throughout the last decades. Nevertheless, the abundance of cargo companies caused some problems. For example, people are struggling to choose a trustworthy cargo company that will not harm their goods. Moreover, the disappearance of their cargo makes people hesitate to utilize cargo services. There are also well-known companies such as UPS that have accomplished billions of package deliveries in 2018[1]. Although these companies can deliver goods without a problem, the cost can be expensive even if you want to send a small package. Last but not least, in the pandemic, demand for cargo shipment has increased due to the fact that people started to use e-shopping more frequently. Moreover, the pandemic caused many people to lose their jobs. Some of these people who own a vehicle started to use apps like Uber, BlaBlaCar to gain the money they need to sustain a healthy life.

Therefore, in this project, we aim to design an application that will retain the good qualities of cargo companies while reducing the price for transportation and help people by enabling them to have another source of income in the pandemic.

In this design report, we will provide a high-level design of the system. Firstly, we will explain the purpose of the system and design goals. Then, we will discuss the system architecture of our project. Subsystem decomposition, hardware-software mapping, and further will be discussed to explain system architecture. Then, we will examine the consideration of various factors in engineering design. Finally, the last part of our project shows the teamwork details such as communication between the group and the roles in our team.

1.1 Purpose of the system

Wheelancer turns the current cargo system into a peer-to-peer system. People will be able to satisfy their need of sending cargo without companies. There are four main goals in the Wheelancer application.

The people who have a car or motorcycle can make additional income since they can work as a freelancer.

The people who have a truck for logistic services will have a chance to reduce the expense of turning back from delivering. For instance, the truck driver who had delivered goods from one city to another can get another delivery from the destination to his/her own city.

The people that have sent the cargo will be able to see the cargo location live. Our application provides cargo owners to keep track of their cargo regards to location.

1.2 Design Goals

1.2.1 Accessibility

● Since the new and useful features such as “turn by turn navigation”, Android Marshmallow 6.1 or newer version is required in order for users to run our system [2]. Node.js will be used for the REST API [3].

1.2.2 Accuracy

● The application will ask for shipment specifications. Particularly, if there are fragile items in the cargo, drivers can be informed about the cargo box in advance.

1.2.3 Availability

● The application uses the GPS signals of drivers, so even if he/she does not keep the internet connection open, the client can be informed about the location of the cargo.

● Our initial audience will be Turkey, thus, bug fixes and updates should occur at midnight. So that it is aimed to protect our users from inaccessibility to the system.

1.2.4 Backup and Recovery

● Our database will be MySQL. Therefore, MySQL undertakes backup and recovery of the data [4].

● Personal data and cargo preferences should be stored in local and global storage so that any data will not be lost.

1.2.5 Extensibility

● It must be open to developments on the basis of new features and new functionalities. (Payment methods, security precautions)

1.2.6 Performance

● The list of drivers should display to users in 4 seconds.
● Data updating should not take more than 0.5 seconds.
● Query responses take under 1 second.

1.2.7 Reliability

● Users need to sign the user agreement in order to provide application reliability. Just in case, their information may be shared with the court or police.
● Any crush on account of software, should not occur in the system.

1.2.8 Security

● The users must sign up with their private credentials.
● The application provides the integrity of the customer's private information.

1.2.9 Testing

● The web server, mobile application, and database will be tested regularly to eliminate errors or any mistakes. Some sort of test:
    ○ Load tests to measure performance according to actual user behaviors.
    ○ Reliability tests will be made to control the webserver and function properly.

1.2.10 Usability

● The users will be able to give their suggestions as feedback to developers.
● The user will not spend time learning the functions of the application. The GUI will be intuitive and user-friendly so user interactions would be easy. For that purpose Flutter will be used[5].

1.2.11 Exception Handling

● During execution, if any error/exception occurs, the user will be shown and explained to the user. In this way, the user will be explained what to do when he/she encounters an error.
● If the user encounters an error that has not been explained by the developers, he/she can contact the developers about this exception.

1.2.12 Legal and Regulatory Requirements

● The application would not allow drivers to know about cargo content.
● The contract signed by the users will prevent crimes significantly.

1.2.13 Maintainability

● Subsystems will be loosely coupled so that maintenance will be done easily. Moreover, the integration of any new module during updates will be easy.

1.3 Definitions, acronyms, and abbreviations

Route: Routes consist of nodes.
Node: Singular location indicating start, end or midpoint for a route.
Unready Transportation:When the packet is not matched by courier it is called Unready Transportation.
Ready Transportation: When the packet is matched by a courier, it is called Ready Transportation.
Joined Route: A route that consists of more than one courier for one package.
Singular Route: A route that consists of 1 courier delivery for one package

1.4 Overview

Wheelancer will be a mobile application that will match people who want their packages to be carried and people who want to carry these packages for additional income. The customers will upload their package information to the system and the Wheelancer algorithm will find potential couriers who can carry a package based on their routes. In order to save resources such as time and fuel, Wheelancer will optimize the route of the courier while delivering a cargo. After delivering a cargo, the

courier can be rated by the owner of the cargo for future matchings. The rating will be an indicator of the success of the courier. The system will have detailed user profiles which will be helpful for customers to pick the best courier for their package. Moreover, after each delivery, a report will be created by the system, which will include information about the delivery such as names, locations, and other things. If a problem occurs for a specific delivery, the report for that specific delivery can be investigated to solve the problem.

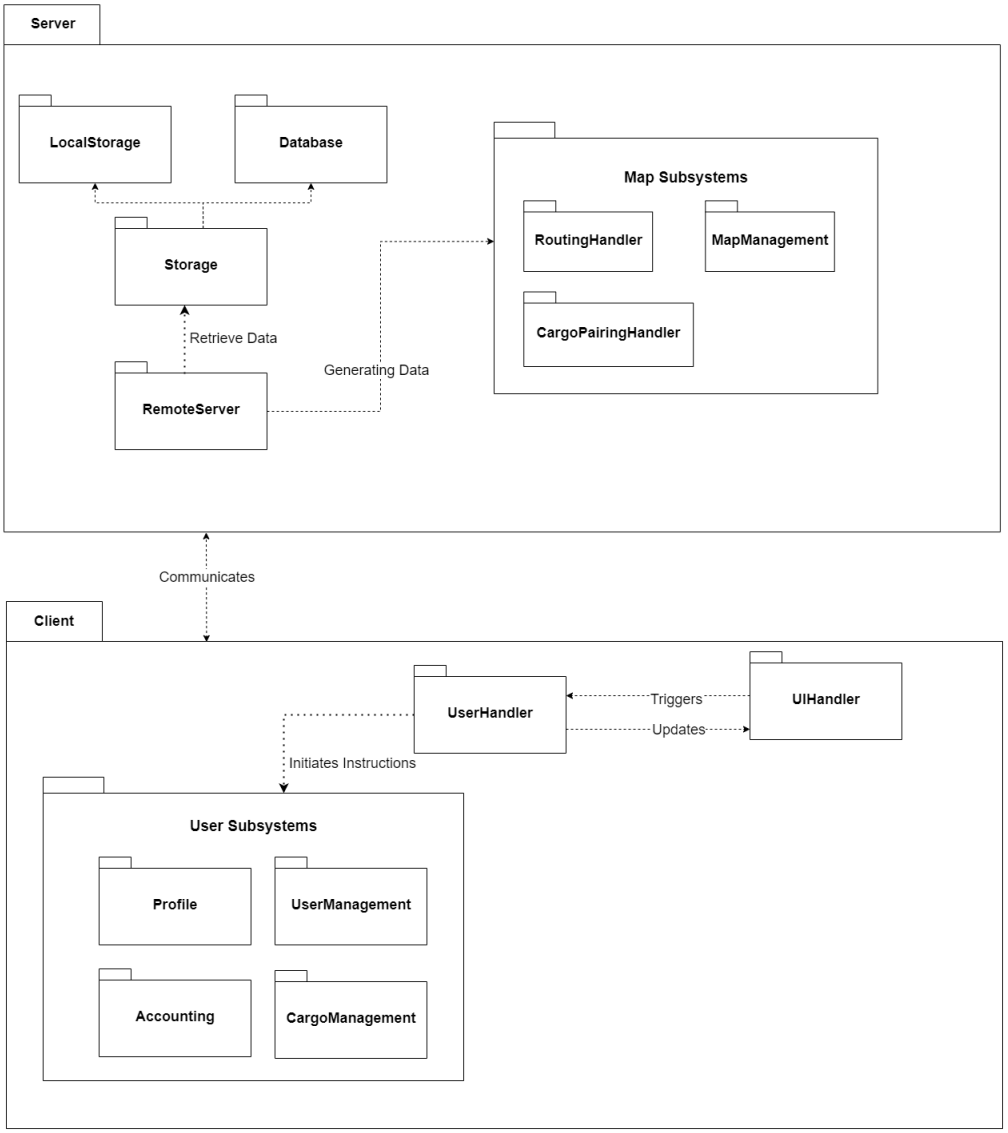
Wheelancer will be done in 3 parts. A database to store persistent data, a front-end to interact with the user, and a back-end that will establish communication between the front-end and the database.

2. Proposed software architecture

2.1 Overview

This section will have information about the subsystem decomposition, HW/SW mapping, persistent data management, access control, SW control and boundary conditions. Each topic will be explained individually in order to make the application structure and logic clear to understand.

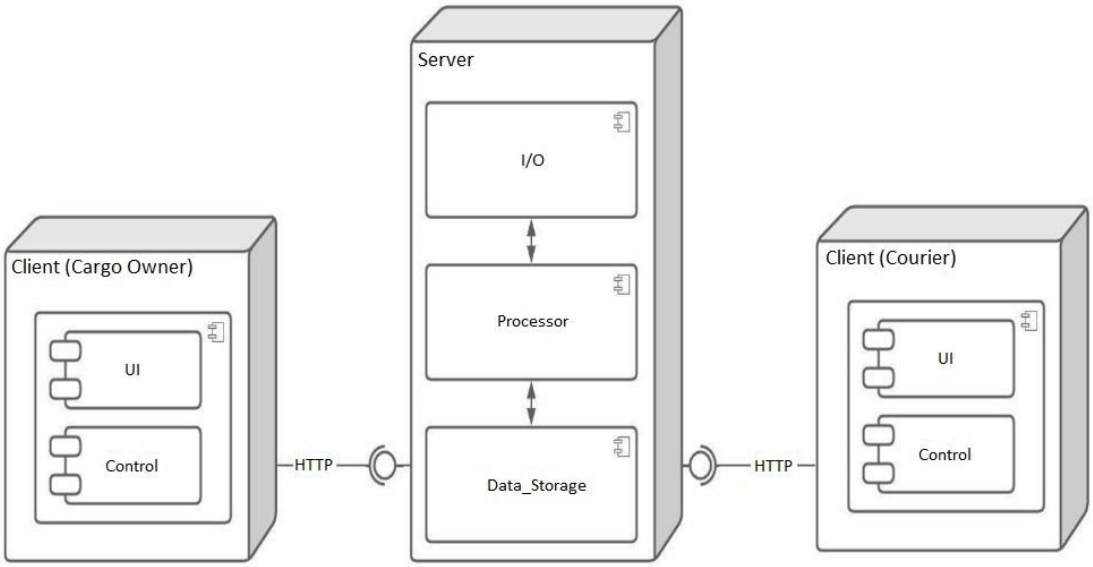
2.2 Subsystem decomposition



2.3 Hardware/software mapping

As usual, ‘Wheelancer’ application have two sides of mapping, those are Client-side and Server-side. Initially, Client-side also has two parts. The first Client-side will be for people (cargo owner) who want to send a cargo and operate on Android smartphones. The second Client-side will be for the people (courier) who want to carry customers' cargos and operate on Android smartphones with GPS for location tracking and camera for taking a picture of the cargo after arriving at the destination point. Couriers who are second side can create a job advert as freelancer. Cargo owners who are first side can choose a courier from the list of adverts. Subsequently, server side will be developed with Node.js and MySQL will be

used to manage the database [6]. Clients will send requests to the server-side by means of HTTP and interactions will be managed by REST API.



2.4 Persistent data management

The personal information of each user is persistent data. Cargo reports, courier locations are also persistent data. Their information will be held in our database until they delete their account from our system.

In the Wheelancer application, persistent data will be kept with the MySQL database. Personal information that is private to each user will be utilized for authentication and some services of the out system such as helping and calling activities. Reports might be shared with other persons in order for other customers to inform about couriers and some personal information like couriers' name can be displayed on customers' pages in order to recognize couriers. However, private information like the photo of a driver's license will not be shared with third parties.

Database interactions and queries are both prime structures for our system. The system will be RESTful, thus, REST API will be used for caching in endpoints.

2.5 Access control and security

In Wheelancer, there are two types of users which are customers and couriers. Customers sign up using their own email addresses as well as their phone numbers so that they are identified clearly. For the couriers, the sign-up process is more complicated as they are the ones who should be inspected better. To enter the Wheelancer system as a courier, these users should verify their driving licenses with photos and they should also clarify their vehicles during this process. To avoid any security issues in terms of accounts, all passwords are hashed inside the server.

After the courier arrangement process, only the customer and the next courier (if any) are able to see the current courier's location from the map interface.

To keep track of any possible lost packages through the delivery process, couriers are obligated to turn off their location services until the delivery is completed.

2.6 Global software control

Wheelancer utilizes event-driven control mechanism with client-server design. Couriers and customers are considered as clients providing user actions such as stating destination point, choosing and contacting couriers, filling evaluation forms, accepting the offers from customers as couriers, stating courier routes from map interface etc. The server side is linked to the actions such as optimizing the requests from clients, and storing submitted information in the database while securing the system using authentication..

2.7 Boundary conditions

2.7.1 Initialization

Users can start Wheelancer by logging in. If they are new, a sign-up step is initially required. After logging in to the system, a customer can continue with selecting a departure and a destination point in order to start a delivery request. A courier can also start the application by logging in to the system and stating a route to go in the near future, waiting for any delivery requests. If they are logged in but away from the application, they can click on a request notification and get directed to the request page.

2.7.2 Termination

Users can terminate Wheelancer by simply logging out of the system or closing the app on their mobile devices.

2.7.3 Failure

If any bugs occur while the user is using the application, for example while a customer creates an order, the user will be notified by the appropriate message indicating the error that happened and the possible solution to the error. Any bug that is missed by the developer can be reported via the email in the help screen.

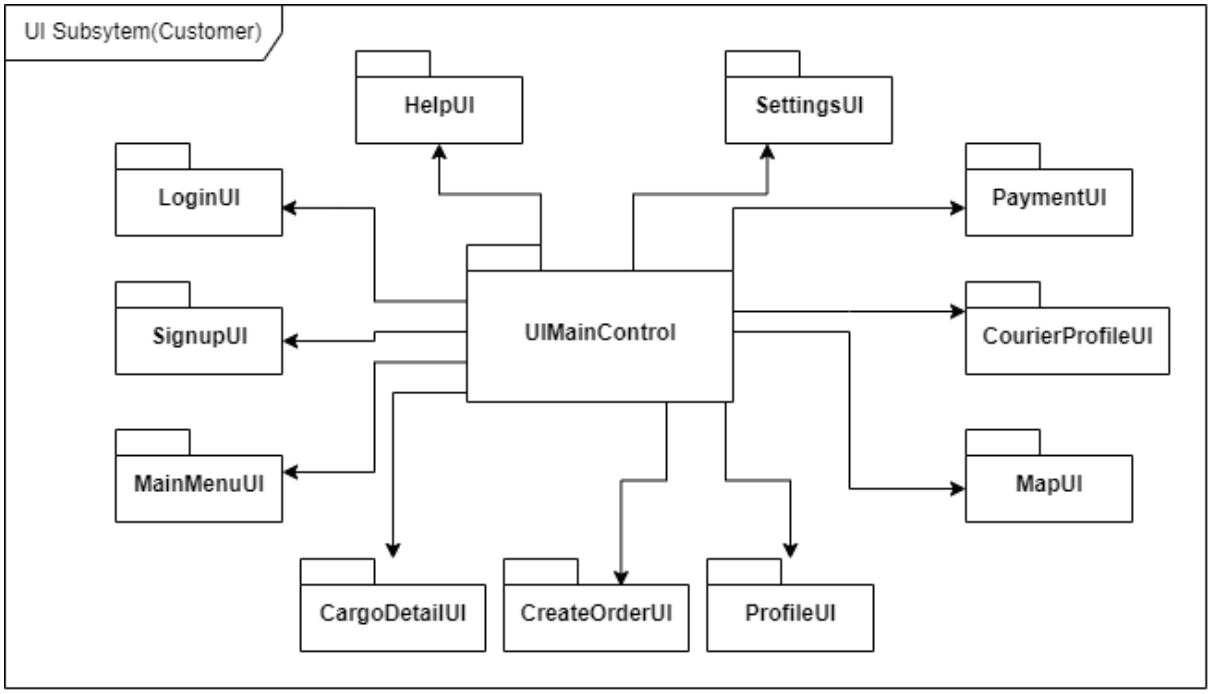
When the GPS or internet connection of the customer fails while using Wheelancer, the user will be notified by the application and the user will not be able to perform any operations during the failure period because this can lead to data loss. The user will be notified when the failure ends and the user can continue to use Wheelancer without a problem.

Another type of failure is server failure. When a server failure occurs due to bugs or other reasons the users must be notified so that they do not perform any operations because their data can be lost. When the server becomes functional again, the user will be notified by removing the error status, hence, the user can continue to utilize Wheelancer services.

3. Subsystem services

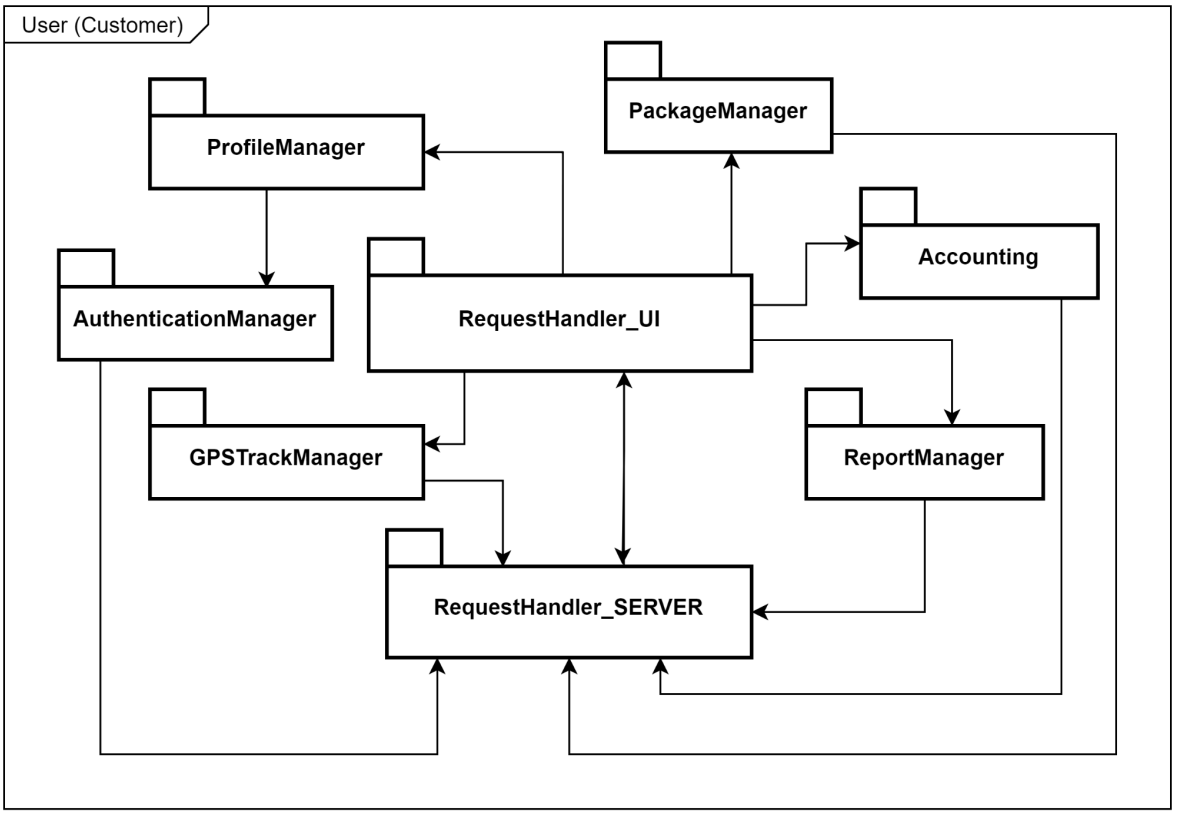
3.1 Client Subsystem (Customer)

3.1.1 UI Subsystem



UIMainControl: The controller of the customer user interface. It manages and changes the user interfaces based on user input and client response.
LoginUI: User interface for the login.
SignupUI: User interface for the signup.
MainMenuUI: User interface for the main menu which the user can access many features such as cargo details, cargo creation, profile etc.
CargoDetailUI: User interface for cargo detail which is chosen by the user.
CreaterOrderUI: User interface for creating a new order
ProfileUI: User interface for the customer profile
SettingsUI: User interface for the application settings for the customer
HelpUI: User interface for the application help screen for the customer
PaymentUI: User interface for payment view after delivery is complete
CourierProfileUI: User interface for customer to inspect a couriers so that the customer can accept or decline the customer
MapUI: User interface to track a cargo that is being delivered by a courier.

3.1.2 User Subsystem



RequestHandler_UI: Manages the responses which originate from the UI and send information to the server or any control part.

RequestHandler_SERVER: The last but not least response manipulation which came from the server or control any part of the system to UI with serviceable responses.

AuthenticationManager: Authentication is for safe connection with the server of the system.

ProfileManager: Manages profile information and changes them when it is desired.

PackageManager: Manages connections of the customer with couriers about their packages.

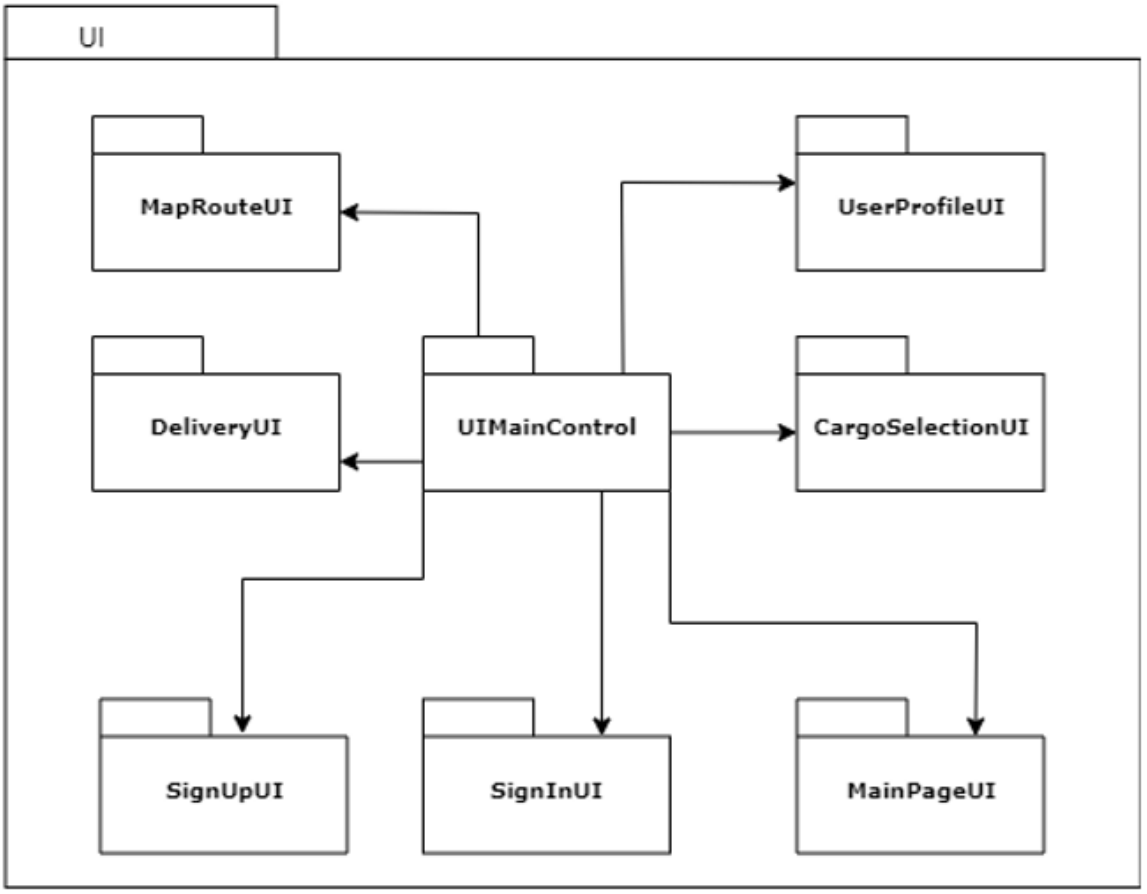
Accounting: Managing the in-app financial situations of the customer. Calculate their total debt and manage it.

ReportManager: Manages evaluations of the deliveries made by customers.

GPSTrackManager: Shares the live location of the courier with the system so that the customer may be informed.

3.2 Client Subsystem (Courier)

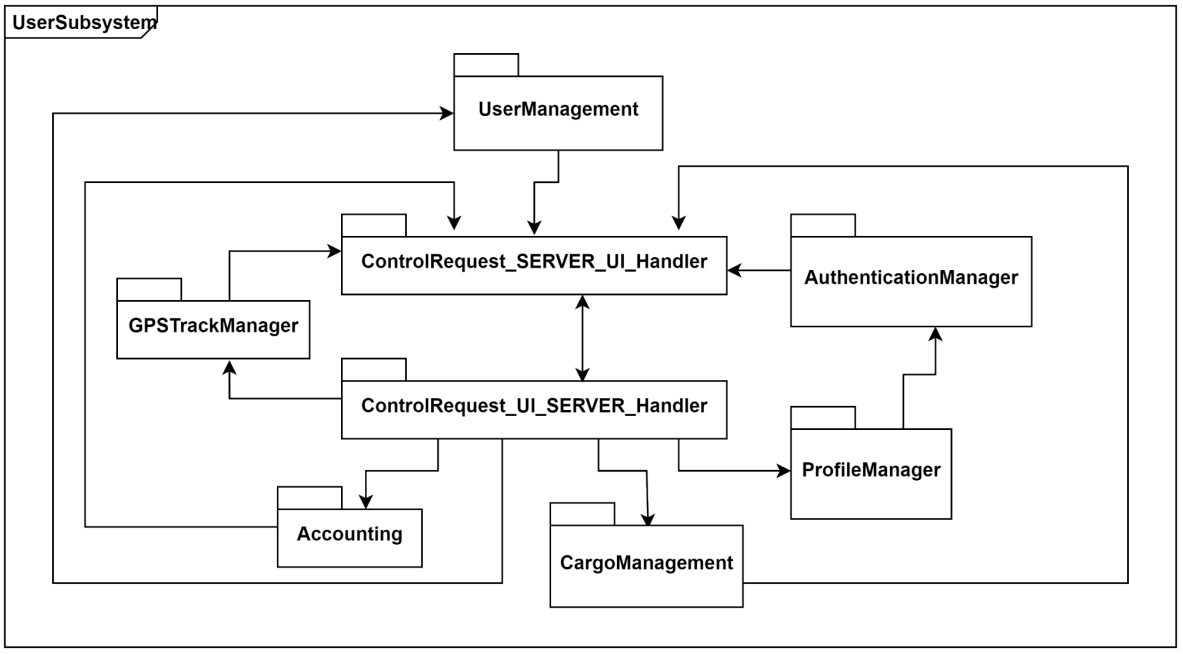
3.2.1 UI Subsystem



***UIMainControl*:** The controller of the courier user interface. It manages and changes the user interfaces based on user input and client response.
***SignInUI*:** User interface for the login.
***SignupUI*:** User interface for the signup.
***MainPageUI*:** Main interface of the courier. Couriers can access functionalities of cargo selection, profile, selecting route for cargo selection etc.
***DeliveryUI*:** Delivery page will be used by courier to upload a photo of the cargo delivering.
***MapRouteUI*:** Route page will be used to see the best route from current location to delivery location. Delivery location can be changed if there is more than one courier on the route.

***UserProfileUI*:** In user profile, courier can see his/her budget and scores given by customers. S/he can upload a credit card to get payment from the customer.
***CargoSelectionUI*:** In cargo selection UI, the courier can see the cargo information and the routes of the cargo. On this page, the courier can contact the customer.

3.2.2 User Subsystem



ControlRequest_UI_SERVER_Handler: Manages the responses which originate from the UI and send information to the server or any control part.
ControlRequest_SERVER_UI_Handler: The last but not least response manipulation which came from the server or control any part of the system to UI with serviceable responses.
UserManagement: Manages user information and activities.
AuthenticationManager: Authentication is for safe connection with the server of the system.
ProfileManager: Manage profile information and change them when it is desired.
CargoManagement: Managing the cargo distribution and according to destination addresses.
Accounting: Managing the in app financial situations of the courier. Calculate their total income and manage it.
GPSTrackManager: Shares the live location of the courier with the system so that the customer may be informed.

3.3 Server Subsystem

The server subsystem consists of three main components. Remote server layer, mapping layer, and data storage layer.

3.3.1 Remote Server Layer

This layer is responsible for handling incoming HTTP requests using REST API while ensuring security. In order to manipulate and proceed incoming HTTP requests, we used Express.js which allows us to bring functionality to the lifecycle of a request easily [7]. In addition to that, any computation for route optimization and merging functionality is redirected to a mapping layer with fast access for users.

3.3.2 Map Subsystem

This subsystem consists of three components. Map management, cargo pairing, and routing handler. Customers register two nodes indicating pickup and delivery point for their goods. Then, the created route (unready transportation) is redirected to data storage in order to list as a potential match for the couriers and released back as ready transportation once match is found. Couriers add their usual route, matching algorithm runs in this subsystem after a new courier route or customer request. Thus, computation is heavily focused on server side so users do not need to run the matching algorithms on their devices while checking found matches. Specifically, map management provides information about route ownership, price calculation. Ready transportation can be completed by one courier using a singular route or multiple courier using a joint route. Cargo pairing helps to match incoming routes with existing potential customers to create pseudo routes. Routes that are classified as a potential match are further optimized in the routing handler with the help of mapping APIs such as Google Route and Mapbox [8]-[9].

3.3.3 Data Storage Subsystem

Data generated from the rest of the map subsystem are stored and provided for later use.

- MySQL is used for storing users, accounting, and routes.
- Local storage is used for storing related documents of users like profile photos, eligibility proofs like driving licenses.

4. Consideration of Various Factors in Engineering Design

Wheelancer is an application that matches the senders and couriers and aims to decrease the cost of sending cargo, assist financial support to any person with a vehicle by creating a network of senders and couriers. The first main consideration of this project is to create an efficient algorithm to match the couriers and senders. The aim of creating an efficient matching system is to minimize the errors of matched senders and couriers so that in the path that is chosen by the sender must be chosen correct couriers even provides more than one courier for the path. Users of this application must not encounter matching problems.

The second consideration of the Wheelancer is to make the user insightful to the application functionalities. Hence, the process of sending a cargo must be explicitly simple as sending cargo must not affect the sender's life.

The third consideration is security. For the security of the cargo, budget systems will be implemented, the courier's security number will be taken by application and the courier photo of the cargo must be sent by the time the cargo is taken and delivered to its destination.

	Effect Level	Effect
Public Health	8	Due to COVID, people started to worry about their health. Our application will alleviate their problems by helping them to carry their cargo without the need of going to the cargo branch.
Public Safety	7	Since we are using personal data, we need to ensure that personal information is stored and secured properly.
Public Welfare	0	This factor has no effect on the application.
Global Factors	2	We need to develop the application in different languages if necessary.
Cultural Factors	0	This factor has no effect on the application.
Social Factors	8	The effects of COVID related to customer preferences affects the application.

Table 1: Factors that can affect analysis and design

5. Teamwork Details

5.1 Contributing and functioning effectively on the team

WP#	Work Package Title	Leader	Members involved
WP1	Project Specification	Ege	All Members
WP2	Analysis Report	Maruf	All Members
WP3	High-Level Design	Onat	All Members
WP4	First Prototype	Berk	All Members
WP5	Low-Level Design	Ümit	All Members
WP6	Database Implementation	Onat	Onat, Berk
WP7	User Interface Implementation	Maruf	Maruf, Ümit
WP8	Second Prototype	Ege	All Members
WP9	Final Implementation	Berk	All Members
WP10	Final Report	Ümit	All Members

Table 2: List of work packages

We have divided Wheelancer into work packages in the above table. This way we ensure that every team member has an equal work balance. Since every member is dealing with one part of the project, team members can be more effective in terms of deciding what to implement and how to implement their corresponding parts. Nevertheless, this does not mean that we stopped communication between groups. This way a group can give support to the other ones when there is a need. Consequently, we will achieve a healthy work environment which will boost our production skills and will enable us to develop a useful application. For the functioning part, we will hold meetings in regular intervals to ensure that every team member is progressing in their respective parts and discuss new ideas or the currently existing ones to enhance our application further.

5.2 Helping creating a collaborative and inclusive environment

In order to create a healthy working environment, we hold meetings regularly. This ensures that we can express our ideas, concerns, and other things regarding the project effectively. For example, if someone has an assignment and cannot do his task for that meeting, another group member who is available can do the job for the team member whose workload is high. This way, we alleviate each other's problems when it is necessary and help each other.

5.3 Taking lead role and sharing leadership on the team

Throughout the project, we did not have a single leader. As described above, each work package has its own respective leader which will be responsible for the development, implementation, and coordination of that work package. This way we are aiming to make every team member dedicate themselves to the project thoroughly. Consequently, every member will be able to enhance their leadership skills and work under leaders with different personalities which will contribute to their adaptation skills to different conditions.

6. Glossary

Customer, Cargo Owner: The people who want to have her/his goods delivered.

Transporter, Courier: The people who have vehicles and deliver the cargos for money.

Good: Cargo that is being transported by courier.

REST API: Representational State Transfer is a stateless architecture using client-server design pattern.

HTTP Request: HTTP is an application layer protocol for transmitting hypermedia documents.

7. References

[1] C. Smith, “Interesting ups statistics and facts,” DMR, 27-May-2021. [Online]. Available: https://expandedramblings.com/index.php/ups-statistics-and-facts/. [Accessed: 06-Oct-2021].

[2] “Android versions comparison: Comparison tables,” SocialCompare. [Online]. Available: https://socialcompare.com/en/comparison/android-versions-comparison. [Accessed: 06-Oct-2021].

[3] “What is a rest api?,” Red Hat - We make open source technologies for the enterprise. [Online]. Available: https://www.redhat.com/en/topics/api/what-is-a-rest-api. [Accessed: 03-Nov-2021].

[4] MySQL tutorial. [Online]. Available: https://www.w3schools.com/mySQl/default.asp. [Accessed: 03-Nov-2021].

[5] “Flutter documentation,” Flutter. [Online]. Available: https://flutter.dev/docs. [Accessed: 03-Nov-2021].

[6] Node.js, “About,” *Node.js*. [Online]. Available: https://nodejs.org/en/about/. [Accessed: 22-Dec-2021].

[7] “Basic routing,” Express basic routing. [Online]. Available: https://expressjs.com/en/starter/basic-routing.html. [Accessed: 03-Nov-2021].

[8] “Directions Service,” *Google*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/directions. [Accessed: 22-Dec-2021].

[9] “Maps SDK: Android,” *Mapbox*. [Online]. Available: https://docs.mapbox.com/android/maps/guides/. [Accessed: 22-Dec-2021].