Bilkent University

Department of Computer Engineering

Senior Design Project

Project name: Wheelancer

Analysis 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

November 15, 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.

With the project specification report, our aim was to explain the description of our shipment application, the constraints we will enforce, functional and non-functional requirements, and the professional and ethical issues that will arise during the development of our project Wheelancer.

With this analysis report, we aim to elaborate our proposed application using UML diagrams, mock-ups of the pages in the user interface and in addition to the explanation, we will mention the ethical and professional constraints as well as the teamwork and planning through the development process of Wheelancer.

1.1 Description

Wheelancer is a mobile application that runs on the Android operating system. It provides a matching service for individuals who want their cargo to be carried and

individuals who want to carry cargo to earn money. The matching system will work as the shipper will select a destination and the senders will specify the destination address and the size of the load and can add note information about the load for the cargo. When possible matches between senders and shippers occur, the senders and shippers can communicate with each other. Before they agree upon shipment, the possible price with respect to the specified vehicle and distance will be calculated by the app. Although the potential price is declared by application, the sender and shipper can agree upon another price if they are not pleased by the recommended price. The payment method includes cash and online payment. If the online payment is chosen by the users, then the application will take the money from the user and will keep the money. If the transportation is completed without a problem, the money will be sent to the shipper. If some problem occurs in the shipment process, the money will be sent back to the sender. The shipper must open their GPS so that the sender can see the live location of the cargo.

In case some companies want to ship their goods, they can choose an option for massive delivery. They also must choose who will be responsible for the loading of goods into the vehicle and unloading of the goods from the vehicle. The shippers with the large load size and with the matching option of the responsibility of the loading will be possible matches.

Furthermore, shippers will be able to view the path between the current location and the destination.

There will be a user star review system so that the sender can understand whether a particular shipper is trustful or not. There is also a badge system that will be given to carriers that have completed successful deliveries and earned the trust of customers, which will be an incentive for carriers to work harder for more income. The punctuality of the shippers will be rewarded with badges. Nevertheless, the shippers with a low rate of punctuality will take the badges indicating their punctuation.

Moreover, there will be an insurance option to protect the load in case some problem occurs. The insurance type or price changes depending on the product value and it is optional.

2. Proposed System

2.1 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 regarding 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 a communication between front-end and the database.

2.2 Functional Requirements

Wheelancer will be capable of the following functionalities:

2.2.1 Logistics Optimization and Matching

In order to start the process of delivery, suitable users and carriers should be matched. Carriers specify their travel route with a specific departure and a calculated arrival time according to the route. Users specify their cargo with departure and arrival location.

- Carriers can specify max load and delivery vehicle type.
- Multiple cargoes can be delivered by one carrier.
- Reward and penalty system for punctuality.
- Courier service for delivery companies.
- Cargo can be delivered using multiple carriers.

2.2.2 User Reviews

Our application heavily relies on the reliability of the carriers so providing them a place to portrait their performance of previous deliveries is crucial. For that reason, Wheelancer will allow users to rate carriers after a completed delivery. Users will be able to rate the performance of the carrier in terms of speed and quality. Encouraging carriers to put extra care and provide users with even more qualified services.

2.2.3 Detailed User Profiles

Apart from user reviews, carriers can showcase trustworthiness with their badge and trophies. Every delivery will be logged in our system and with each reached milestone they will earn badges that will indicate experience and loyalty.

2.2.4 Comprehensive Reporting

Each trip has a delivery report that can be examined by the user. The report includes:

- For carriers, the revenue of the delivery can be examined which is calculated by subtracting expenses gathering vehicle fuel consumption and fuel price.
- For users, live tracking of the vehicle's position and status of delivery.

2.3 Nonfunctional Requirements

2.3.1 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.

2.3.2 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.

2.3.3 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.

2.3.4 Backup and Recovery

- Our database will be MySQL. Therefore, MySQL undertakes backup and recovery of the data.
- Personal data and cargo preferences should be stored in local and global storage so that any data will not be lost.

2.3.5 Extensibility

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

2.3.6 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.

2.3.7 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.

2.3.8 Security

- The users must sign up with their private credentials.

- The application provides the integrity of the customer's private information.

2.3.9 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.

2.3.10 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.

2.3.11 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.

2.3.12 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.

2.3.13 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.

2.4 Pseudo Requirements

- Version Control

    - Git will be used for version checking and tracking.
- Implementation Language
    - Wheelancer will be a mobile application using Flutter software development kit for frontend in Dart language.
    - REST service will be implemented using Node.js environment in order to provide endpoints for frontend to communicate.
    - MySQL database will be utilized by the backend to store user data.

- Frameworks, Libraries and External API's
    - ExpressJS for REST API middleware environment and routing.
    - Google Maps API for map components and location info manipulation.
- Target Platform
    - Wheelancer will have a server. The app will be available for Android and iOS.

2.5 System Models

2.5.1 Scenarios

Scenario 1 Sign Up

Actors: Cargo Owner, Transporter

Entry Conditions: The user opens the app and clicks the "Sign Up" button.

Exit Conditions: The user closes the app "Sign Up" button to Log in.

The Flow of Events:

1. Clicks the "Sign Up" button on the Log In page
2. The registration page is opened.
3. The user gives the required information.
4. User selects one of the checkboxes to be a transporter or Cargo Owner.
5. The user clicks the "Sign Up" button.
6. The system creates an account with the user's information.

Scenario 2 Log In

Actors: Cargo Owner, Transporter

Entry Conditions: The user opens the app.

Exit Conditions: The user closes the app.

The Flow of Events:

1. The user fills the gaps with the required information.
2. The user clicks the “Login” button.
3. The system checks the information of the user.
4. The user is navigated to the main menu according to user type.

Scenario 3 Add a Cargo

Actors: Cargo Owner,

Entry Conditions: Clicks the “+” icon in the menu.

Exit Conditions: The user closes the app OR clicks the “Upload Cargo” button.

The Flow of Events:

1. The user fills the gaps with the required information.
2. Upload the photograph of the cargo.
3. The user clicks the “Upload Cargo” button at the bottom of the page.

Scenario 4 Check Cargo Status

Actors: Cargo Owner

Entry Conditions: Clicks one of the cargoes belonging to himself/herself in the menu.

Exit Conditions: The user closes the app OR clicks the “Back” button OR “Live location” OR “Delete Cargo” button.

The Flow of Events:

1. The system displays information about the specified cargo of the cargo owner.
2. By clicking the “Live location” button, the user can see the map that includes the route of the transporter and the live location.
3. By clicking the “Delete Cargo” button, users can remove the cargo posting.

Scenario 5 Assigning a cargo to a courier

Actors: Cargo Owner

Entry Conditions: Notification in the main menu of the cargo owner.

Exit Conditions: The user closes the app OR clicks the “Back” button OR “X” sign to cancel the courier.

The Flow of Events:

1. By clicking the check sign the user is navigated to the courier’s profile.
2. According to his/her profile, a user can accept the courier with the check sign, and cancel the courier with the “X” sign.
3. By clicking the check sign, the user is navigated to the payment section.
4. User clicks one of the “check” or “X” signs, check sign to accept the payment and card information, X sign to cancel the payment.

Scenario 6 Give Feedback

Actors: Cargo Owner

Entry Conditions: When the cargo arrives, a notification pops up in the main menu of the user, and the user selects the check icon to give feedback.

Exit Conditions: The user closes the app OR selects “X” not to give feedback.

The Flow of Events:

1. The system displays the information of the courier and gives a rating out of ten.
2. The “X” sign cancels the feedback session.
3. The “Check” sign accepts the rate.

Scenario 7 Confirm the Account

Actors: Transporter

Entry Conditions: From his/her profile page.

Exit Conditions: The user closes the app OR log out button to change the account.

The Flow of Events:

1. User fills the gaps with required information about his/her vehicle.
2. User uploads the driver license photo
3. User uploads the vehicle license.
4. User clicks the “Save” button.
5. User is navigated back to his/her profile page.

Scenario 8 Set a Route and Cargo

Actors: Transporter

Entry Conditions: From the main menu of the specified user type.

Exit Conditions: The user closes the app OR log out button to change account.

The Flow of Events:

1. The transporter selects a starting point.
2. The transporter selects an ending point.
3. System displays the cargo options to deliver.
4. User selects one of them.
5. System displays information about the cargo (such as destination, size).
6. If the user accepts the wage for the cargo, can click the “check” icon.
7. Else if the user does not accept the wage, can click the “X” icon.
8. User navigated to the main menu.

Scenario 9 Start the Route

Actors: Transporter

Entry Conditions: “Start the Route” button from the main menu of the specified user type.

Exit Conditions: When the cargo is arrived.

The Flow of Events:

1. Transporter clicks the “Start button”.
2. System provides the user a road map from the start point to the destination point.
3. System keeps track of the GPS signals of the transporter.
4. When the cargo is delivered, the transporter uploads a picture of the cargo.
5. User navigated back to the main menu.

Scenario 10 Withdraw Money

Actors: Transporter

Entry Conditions: With the “Withdraw” button from the main menu of the specified user type.

Exit Conditions: The user closes the app OR log out button to change account OR “Back” button.

The Flow of Events:

1. User writes down the amount of the money that he/she wants to withdraw.
2. If there is enough money, the system pays this amount to his bank account.
3. Else if There is not enough money, user urge to enter another amount.
4. When withdrawal is complicated, the user is navigated to the main menu.

2.5.2 Use Case Model

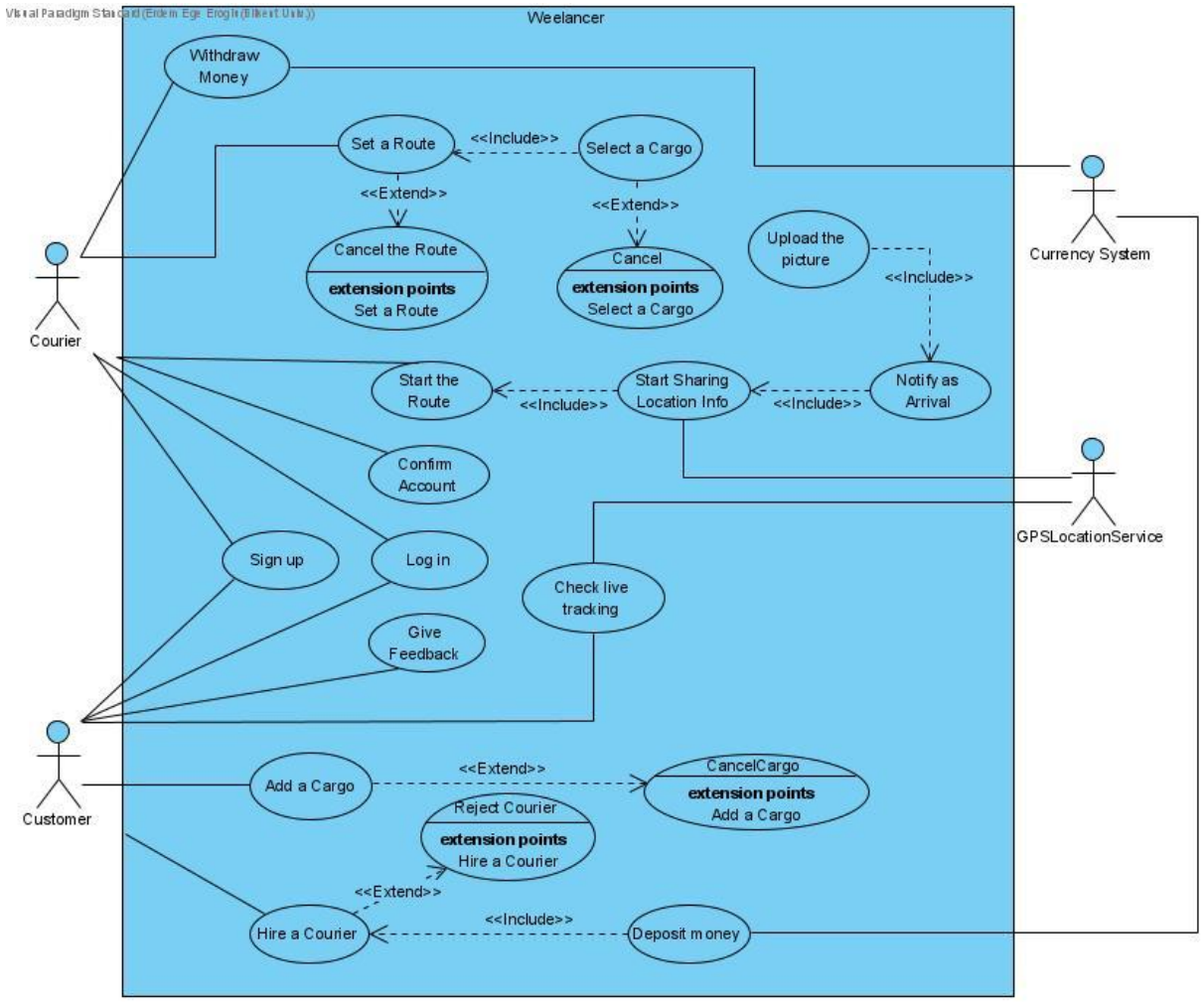

Figure 1: Use case diagram for Wheelancer

2.5.3 Object and Class Model

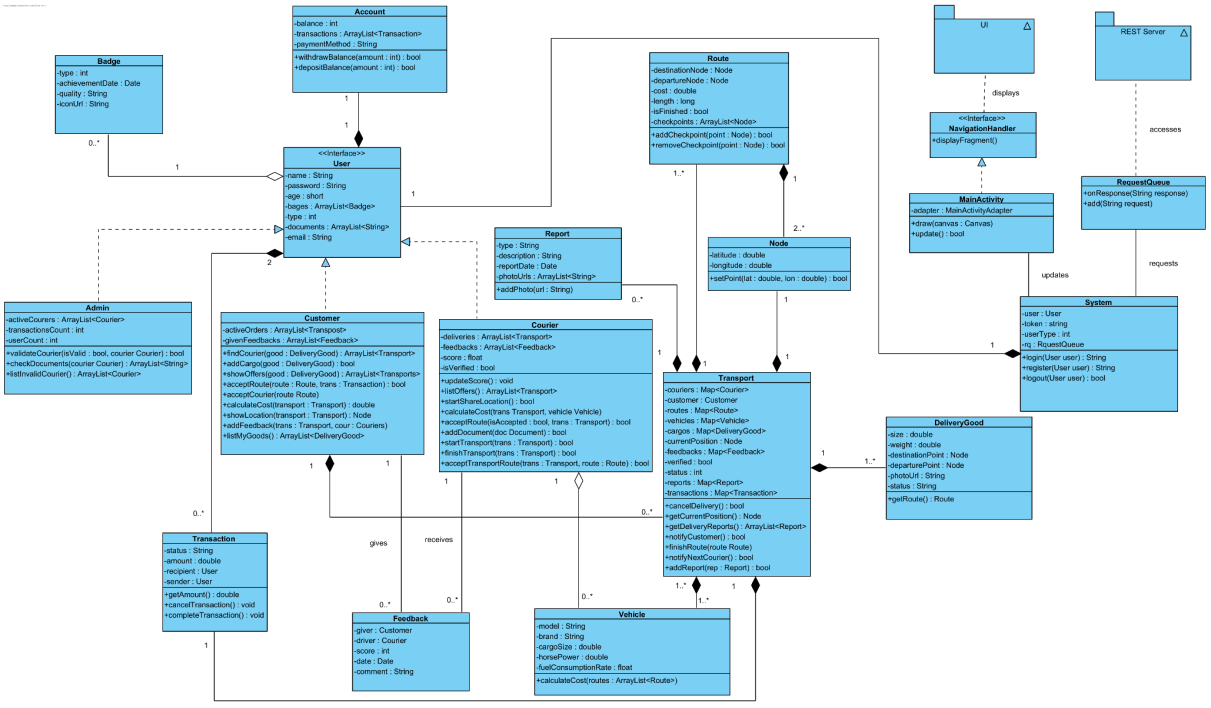

Figure 2: Object and Class Model for Wheelancer

2.5.4 Dynamic Models

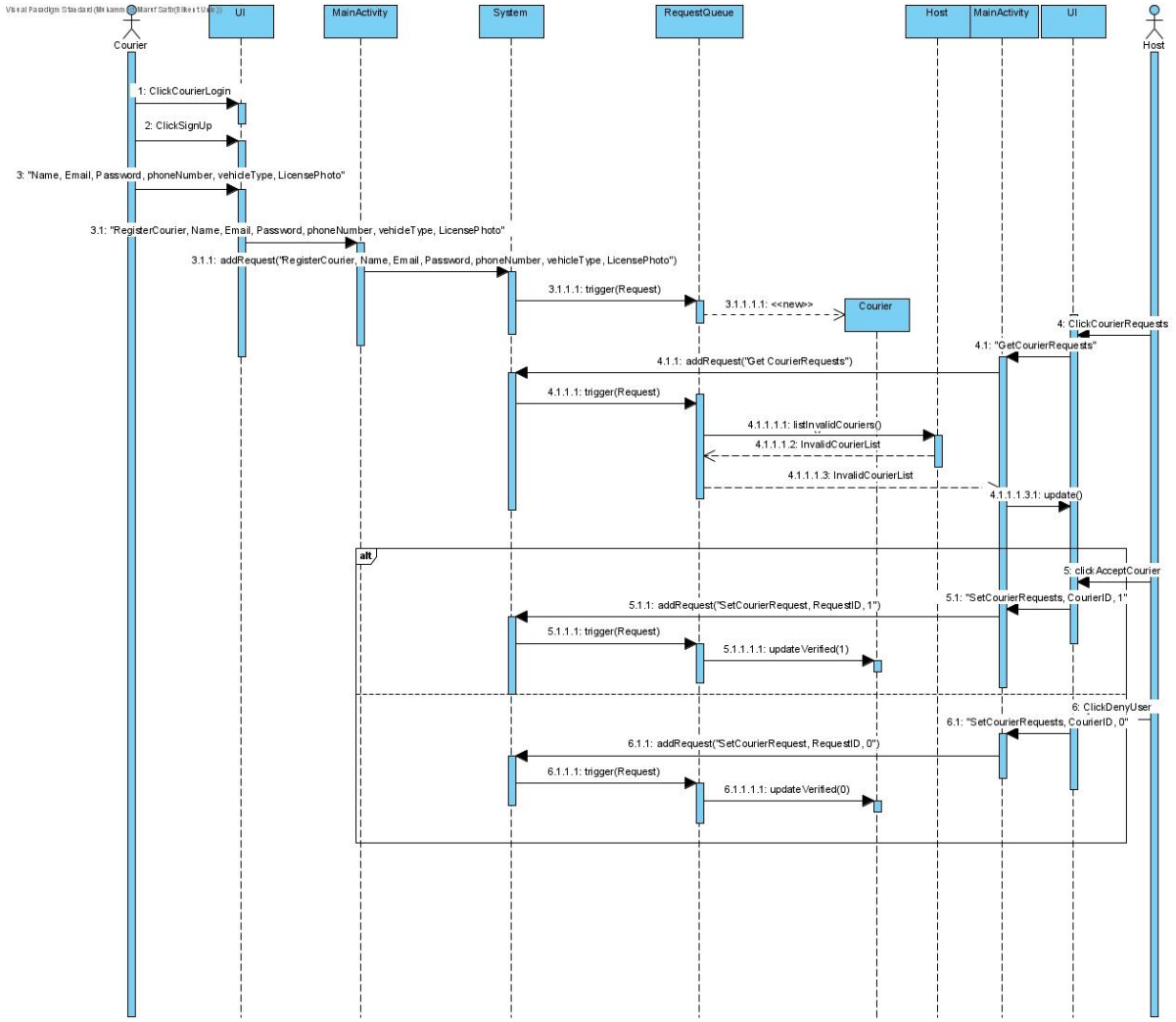

Figure 3: Sequence Diagram for Courier Registration

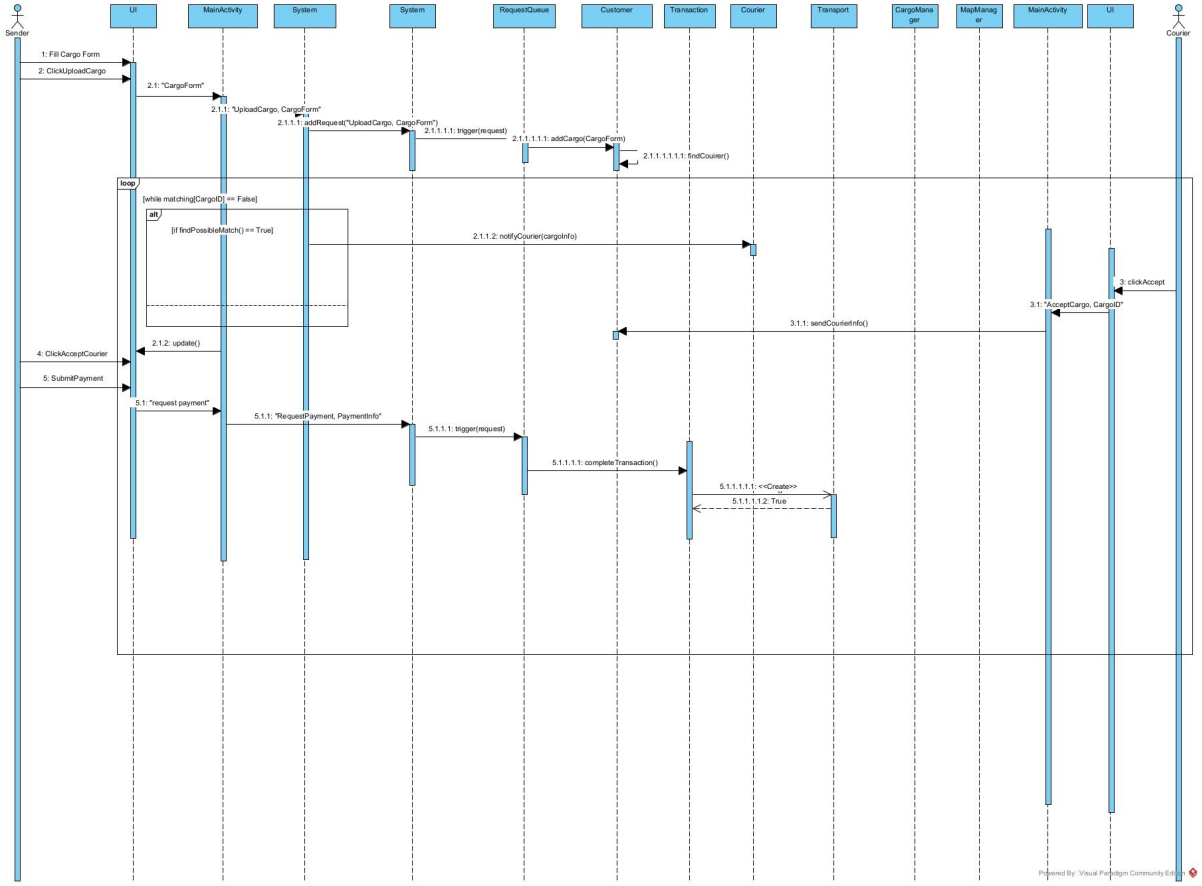

Figure 4: Sequence diagram for matching courier and sender

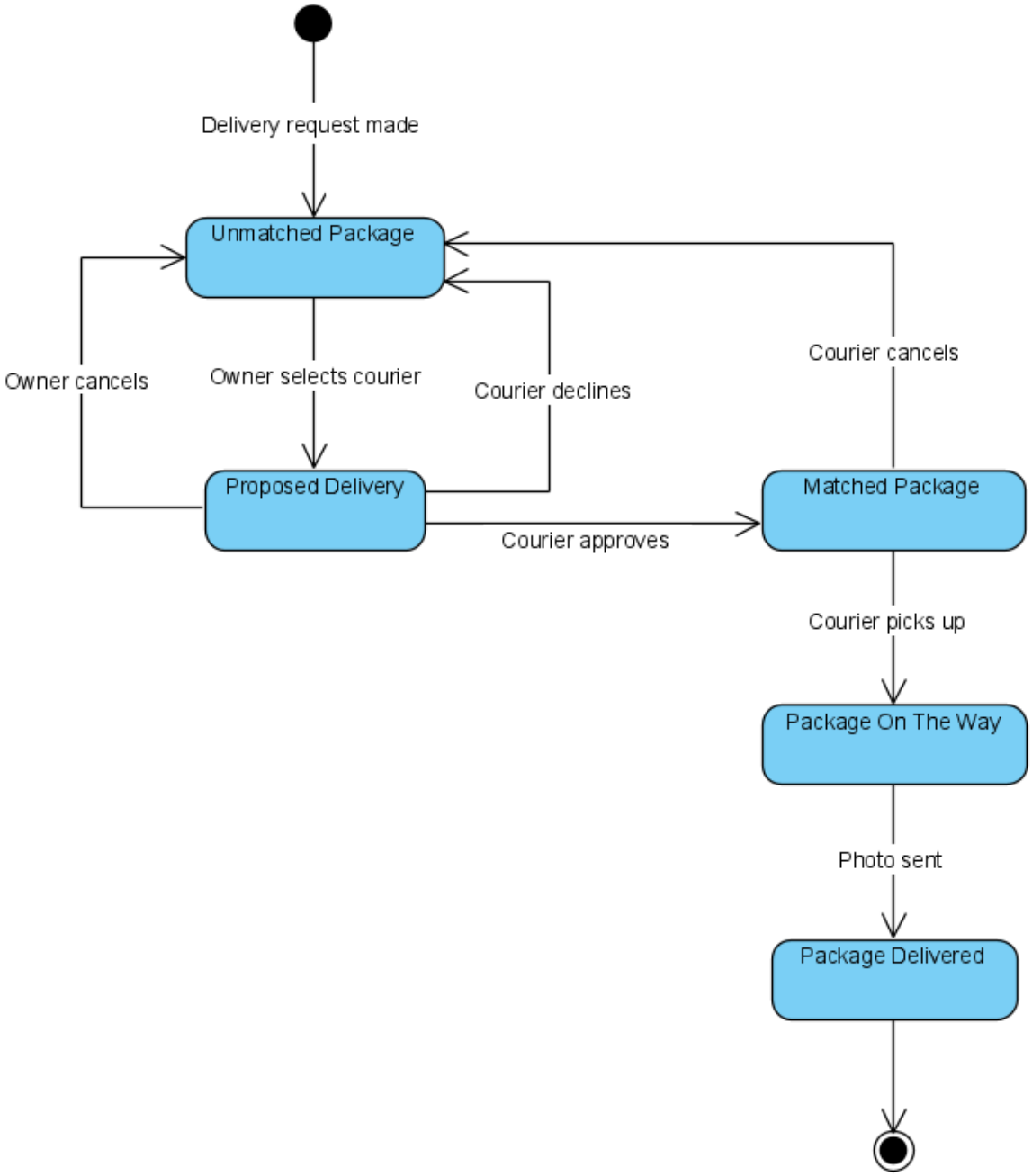

Figure 5: State diagram for the package during delivery

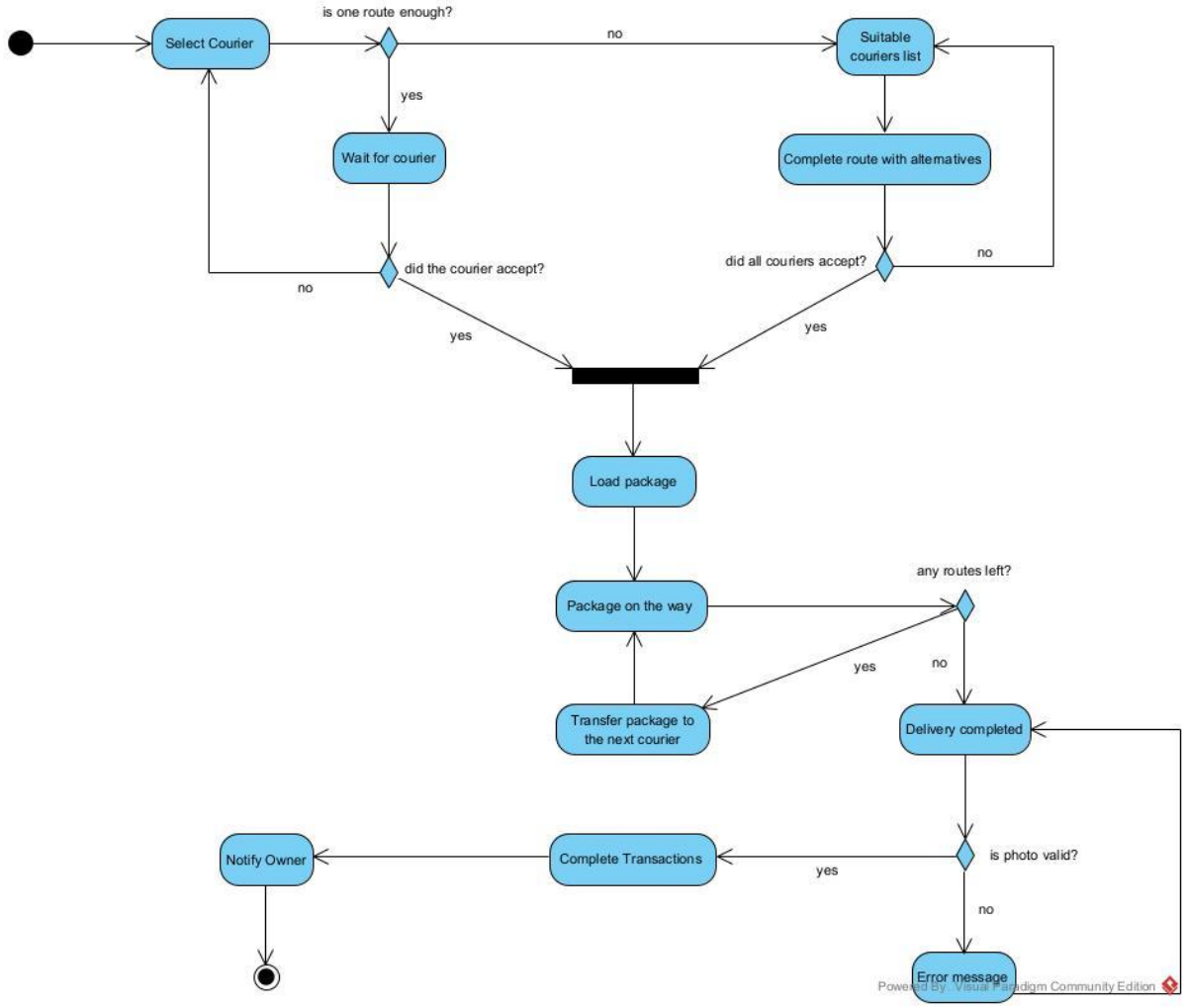

Figure 6: Activity diagram for the shipping process

2.5.5 Navigational Path and User Interface Mock-ups

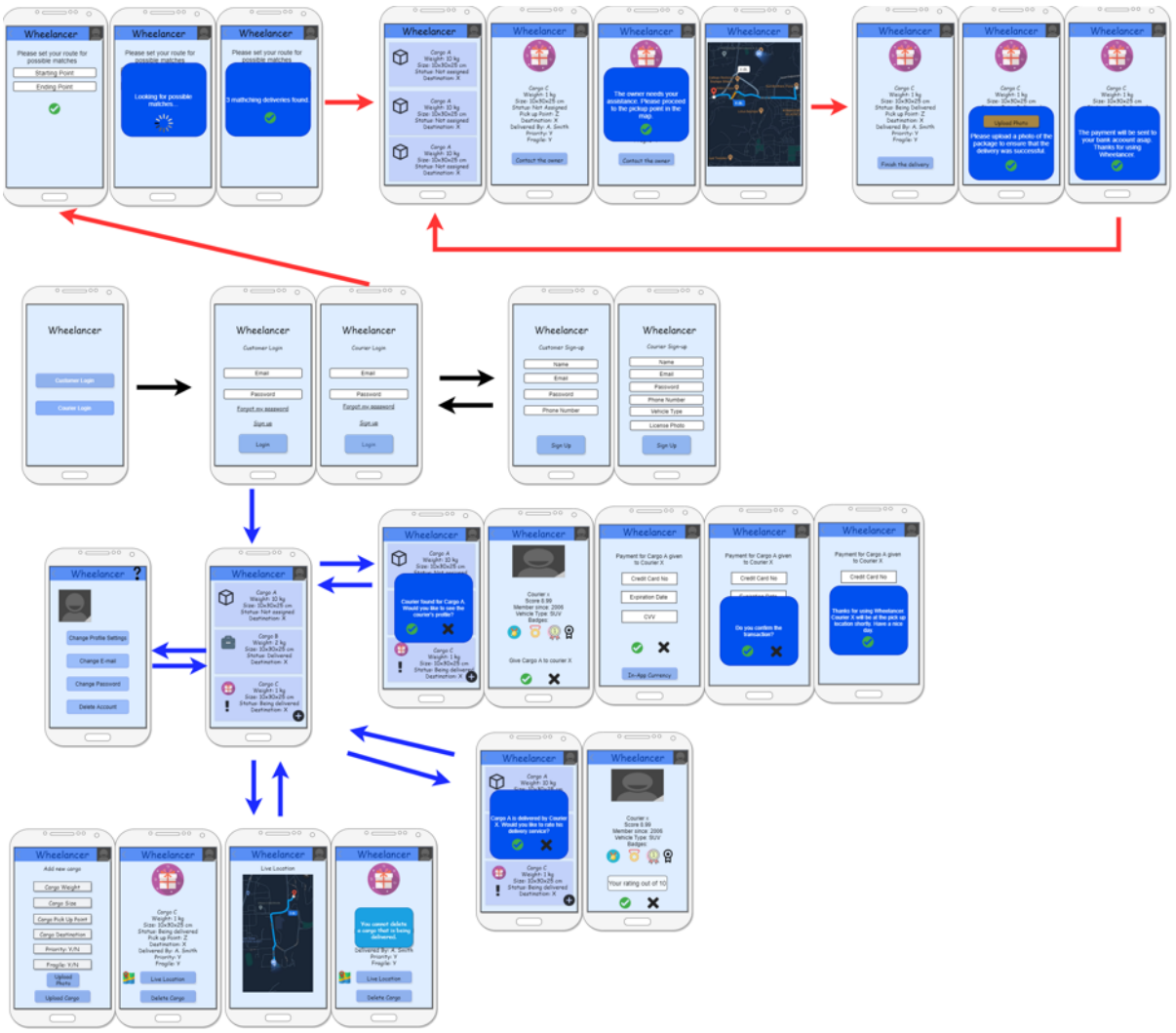

Figure 7: Navigational path for the mock-ups

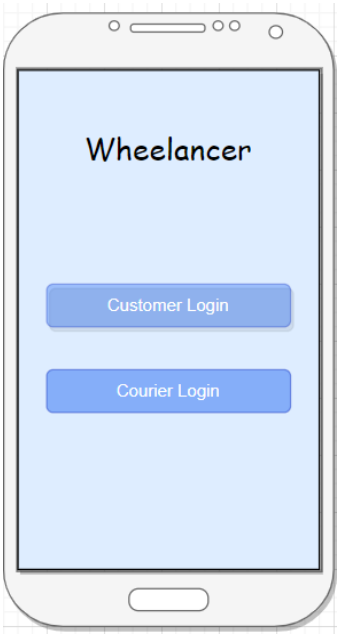

Figure 8: Opening Screen

After a user launches Wheelancer, he/she will be prompted to choose a role.

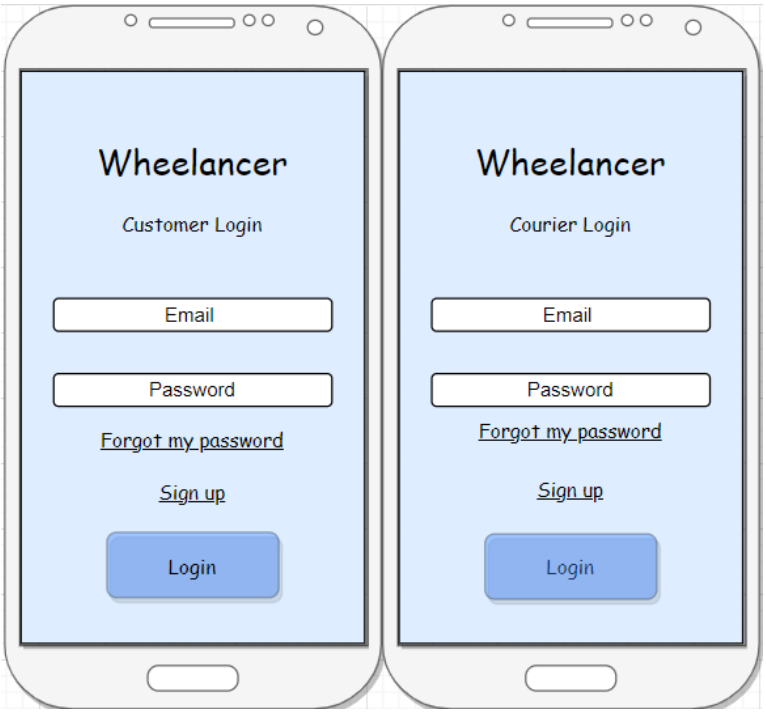

Figure 9: Login screens for different roles

After choosing their role, users will be expected to enter their credentials in order to login to Wheelancer and use delivery services. If the user does not have an account, he/she can press the sign-up button to create a new account.

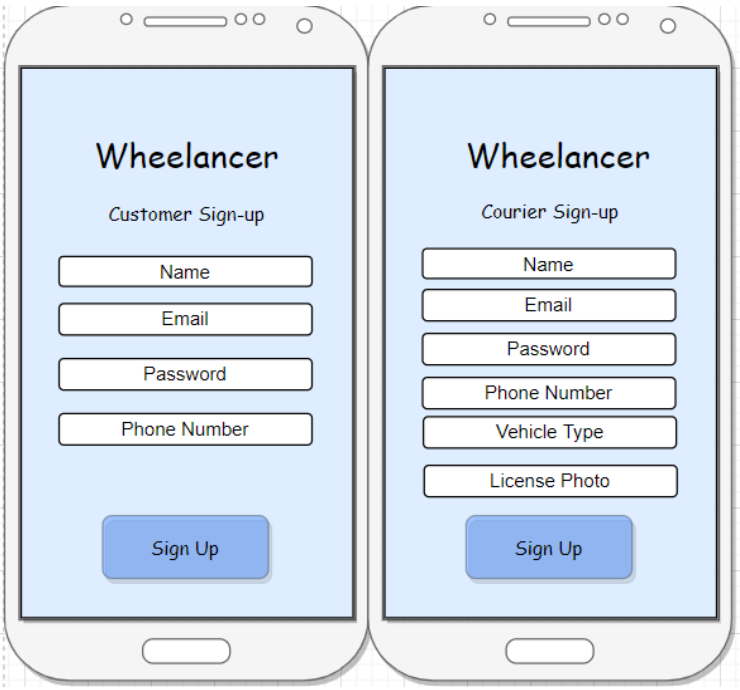

Figure 10: Sign-up screens for different roles

Since each entity will be stored with different attributes in the database, we will create 2 different sign-up screens so that we can get the differentiating attributes of each role.

Figure 11: Main menu for the customer

The customer main menu shows the list of the packages submitted to our system if there is any. The user can add new packages by using the “add” button at the bottom right corner. If the user wants to do something about their profile, they

can use the “profile” button at the top right corner. If they want to inspect a particular package, they can click on that package to see the remaining details. The exclamation mark means that the package is fragile and must be handled with care.

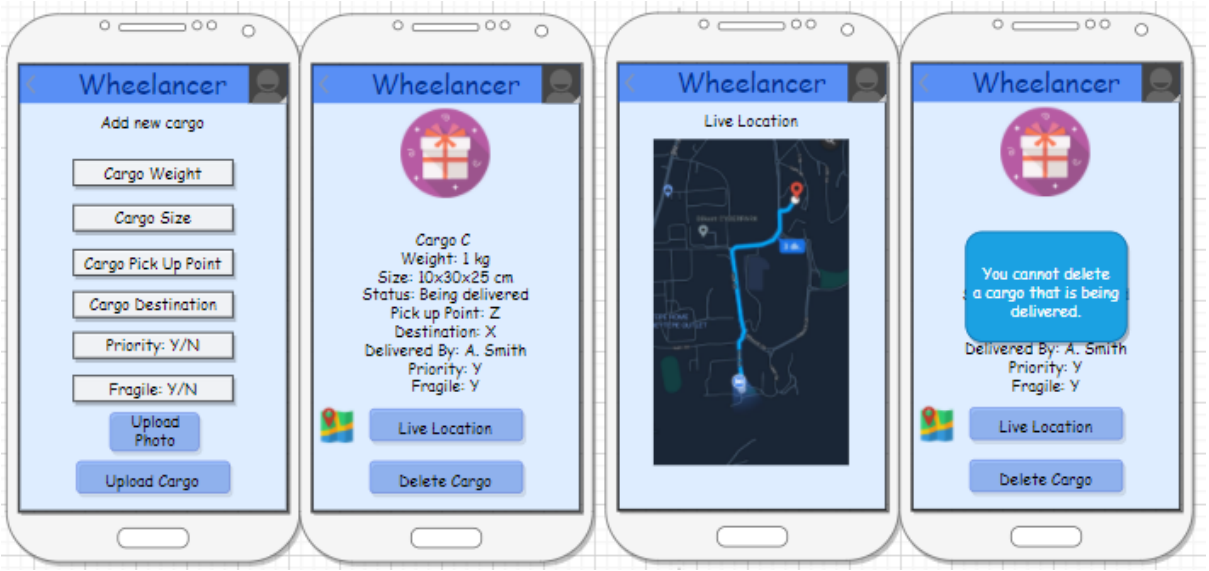

Figure 12: Some cargo mock-ups for the user

When the user wants to add a new cargo, we will take some information about the cargo to match with possible vehicle types. We will take the pick-up and destination points to match them with possible couriers. The customer can mark the cargo as a priority so that it will be delivered faster than normal deliveries with an increased cost. Then the customer will specify whether the cargo contains something fragile. Lastly, the user will upload the photo of the cargo and press “Upload Cargo” to register the package to our system.

The cargo information screen shows detailed information about a specific cargo. If it is assigned to a courier, they can press the live location button to track their package. When they press the button they will get a map screen that shows the location of the courier. If the user wants the cargo deleted, he/she should do this before their cargo is assigned to the courier. Otherwise, the app will show an error similar to the last mock-up.

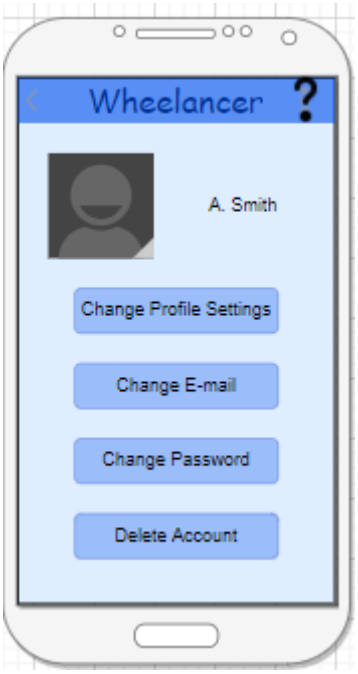

Figure 13: Profile Screen

In the profile page, the users can change their profile settings such as dark mode, name depending on their choices. They can also change their email and password if they want to do so. If they want to delete their account, they should not have a package assigned to a courier. After all assigned deliveries are complete, the user can delete their account. This is implemented in case the package contains illegal things to carry.

Figure 14: Assigning a cargo to a courier

When the algorithm finds a courier that can carry a particular cargo, the user will be notified about the find. The user will be able to see brief profile information about the courier and accept or decline as he/she wants. After accepting the courier, the user will be prompted to make a transaction either by credit card or in-app currency.

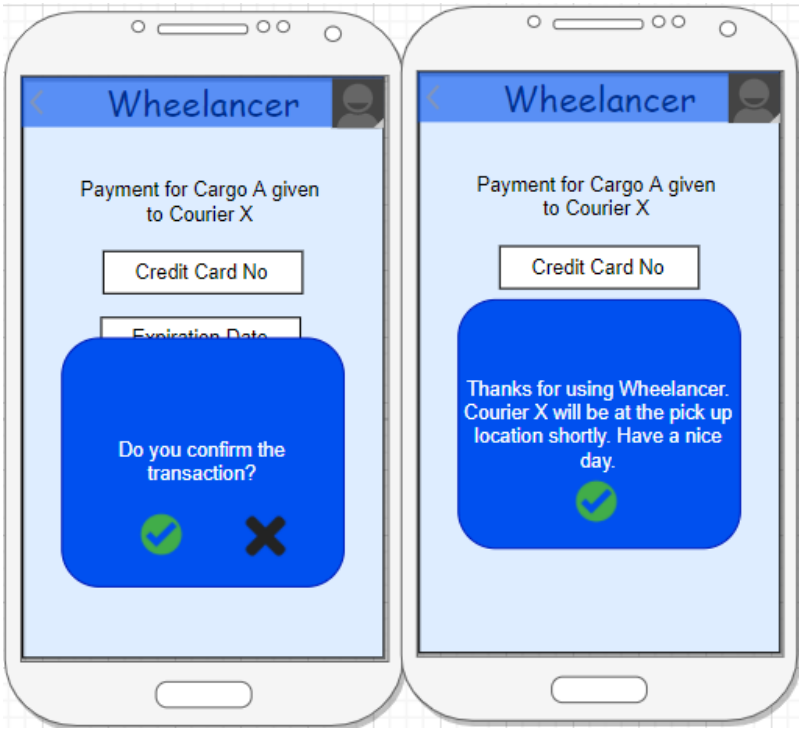

Figure 15: Assigning a cargo to a courier

After choosing the payment method, the user will confirm the transaction. Then, the courier will come to the pickup location as soon as possible. After this, the user will be redirected to the main menu screen.

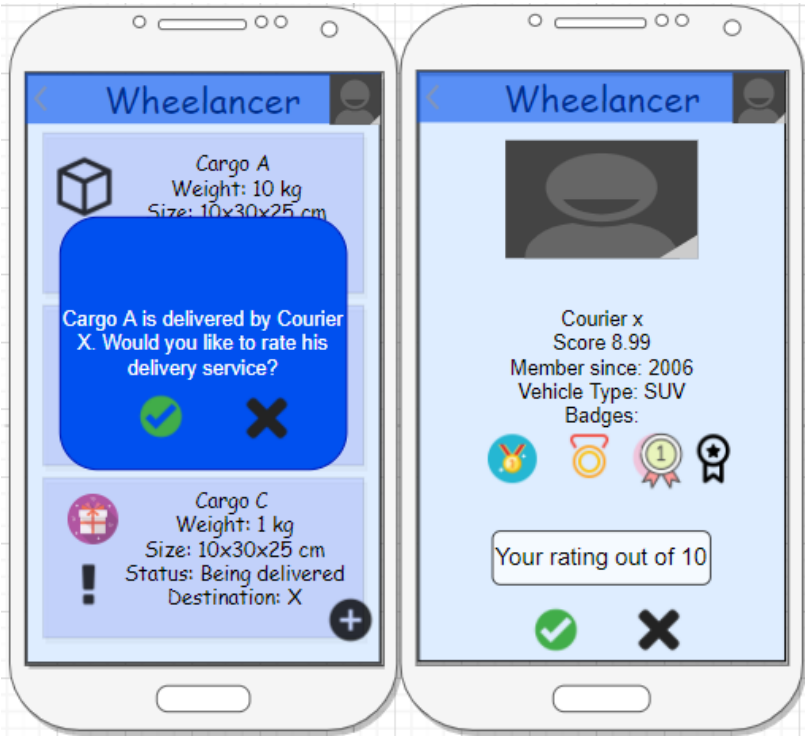

Figure 16: The rating screen after a delivery

After a delivery is completed, the owner of the package can rate the courier if he/she wants. The rating will be added to the other ratings of that courier and a new average score will be computed.

Figure 17: The main menu for the courier

After the courier logins to Wheelancer, he/she will be required to enter the starting and ending points of his/her route. According to the input, the app will try to find possible matches for that courier and return a result whether it has found a match or not.

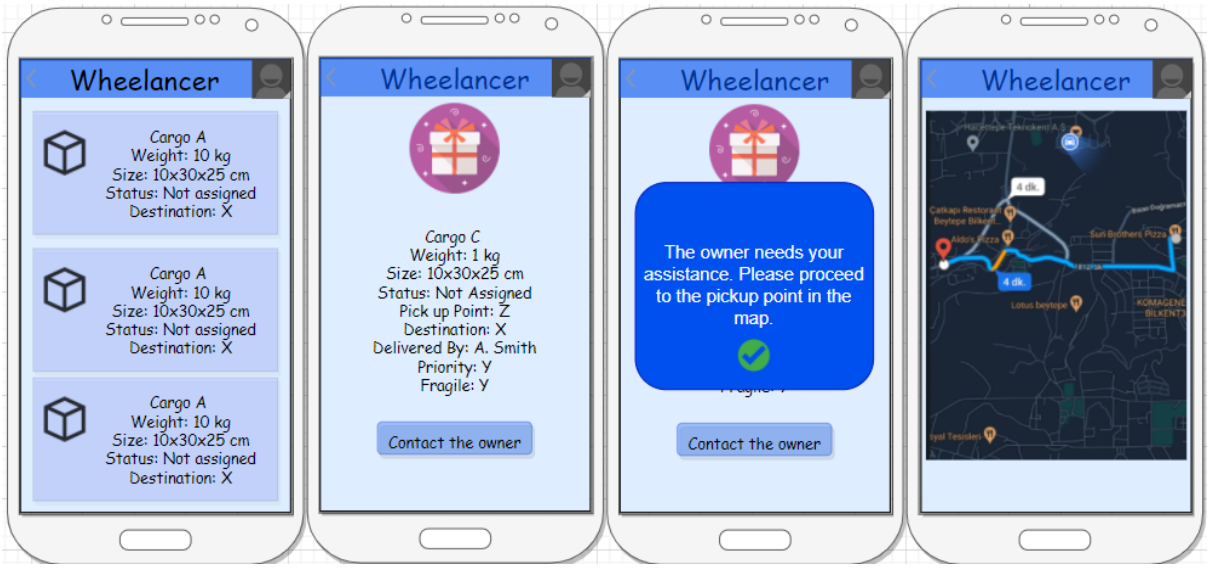

Figure 18: The cargo selection screens for courier

After the algorithm finds any possible matches, they will be displayed at the screen of the courier and the courier will pick a package to deliver. After seeing the delivery details the courier can contact the owner or not. If the courier contacts the owner and the owner accepts the delivery offer of the courier, the courier will be assigned to deliver the package to the destination point. In order to help the couriers, the current location of the courier, starting and ending points for the delivery will be displayed by the Wheelancer app.

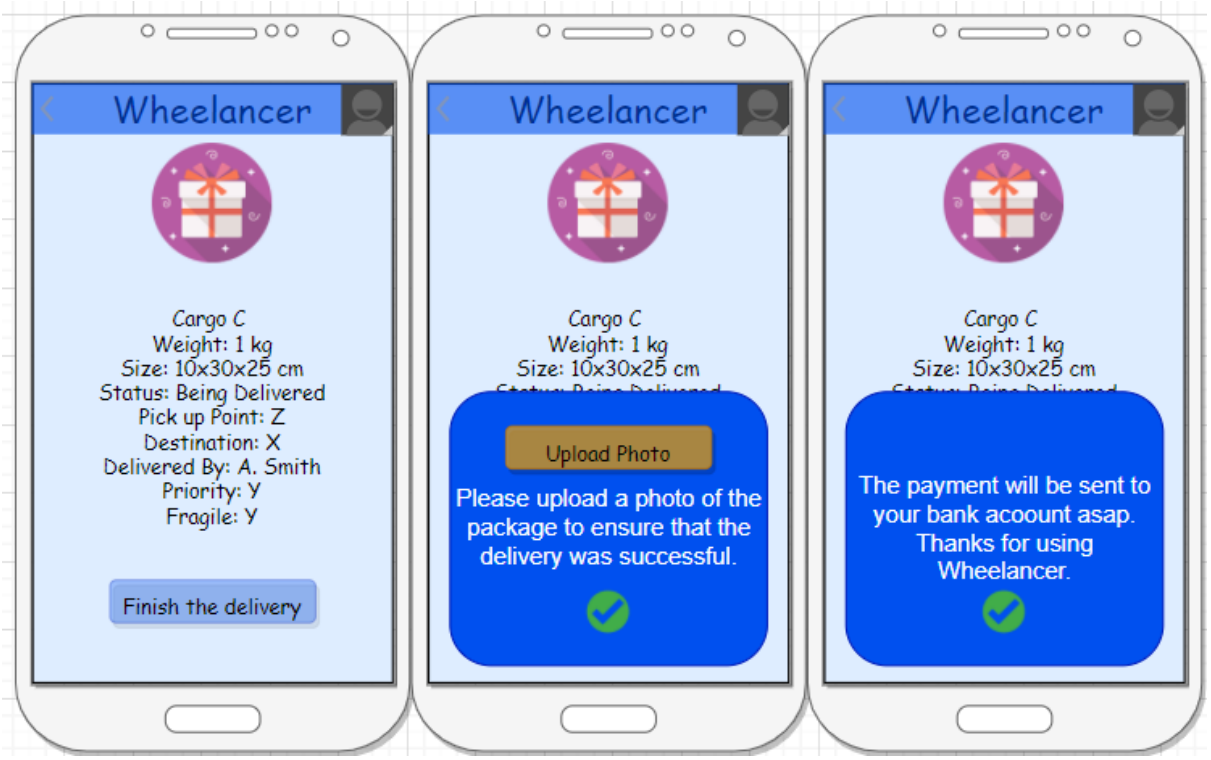

Figure 19: Completing the delivery screens for courier

After the courier successfully delivers the cargo, he/she will press the finish delivery button. Then, the courier will be prompted to take a photo of the package he/she is giving to the receiver to ensure that there will not be any issues regarding the delivery. After the courier submits the photo and gives the package, he/she will receive his/her payment as soon as possible. If there is any problem with the delivery, the payment will wait until the problem is solved.

Apart from these mock-ups, there are pages like credits, how to use, etc. These are not included in the mock-up because we wanted to explain the main functionality and how it will look as much as possible. The names of the pages mentioned above clearly explain what those pages are responsible for.

3. Other Analysis Elements

3.1 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 sender must be chosen correct couriers even provides more than one courier for the path. Users of this application must not encounter matching problems.

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.

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.

3.2 Risks and Alternatives

Wheelancer is a platform that provides couriers to cargo owners, and get couriers earn money. To do this, the application has to know the cargo owner’s address and the route of the courier. During the transportation, the cargo owner entrusts his/her cargo package to the courier. However this practice comes with risks. Courier could potentially steal or damage the goods in order to discourage this behaviour. We will collect all the personal information including national identity card, driver license. In addition to that, we provide tools such as a review system, badge collection and detailed profile to help customers to choose the best courier for their delivery. We check each step of the delivery from matching to the arrival of the destination. At the end each delivery courier needs to take a picture to inform the final state of the goods.

3.3 Project Plan

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 1: List of work packages

WP1: Project Specification	
Start Date: 26.09.2021	**End Date: 11.10.2021**
Leader: Ege	**Members Involved: All Members**
Objectives: Describe the project briefly, determine the functional,non-functional requirements and constraints. Discuss some professional and ethical issues.	
Tasks: ● Identify functional requirements ● Identify constraints ● Discuss professional and ethical issues	
Deliverables: Project Specifications Report	

WP2: Analysis Report	
Start Date: 26.09.2021	**End Date: 08.11.2021**
Leader: Maruf	**Members Involved: All Members**
Objectives: Define the systems in terms of diagrams, consider some ethical professional issues, define work packages, draw initial mock-ups	
Tasks: ● Draw class, state, activity, sequence diagrams to define the proposed systems ● Draw initial mock-up for user interface ● Consider some ethical and professional aspects of the project ● Define work packages and elaborate them	
Deliverables: Analysis Report	

WP3: High-level Design	
Start Date: 26.09.2021	**End Date: 24.12.2021**
Leader: Onat	**Members Involved: All Members**
Objectives: Propose a software architecture for the system, determine the subsystem and boundary conditions, consider various social factors	
Tasks: ● Propose a software architecture by determining the subsystem decomposition, HW/SW mapping, persistent data management etc. ● Determine the boundary conditions ● Consider various social factors for Wheelancer ● Ensure teamwork	
Deliverables: High-level Design Report	

WP4: First Prototype	
Start Date: 26.09.2021	**End Date: 31.12.2021**
Leader: Berk	**Members Involved: All Members**
Objectives: Build the first prototype of the Wheelancer by implementing some basic functionality	
Tasks: Backend support for those functions: ● User profile, login and registration. ● Courier vehicle specification. ● Customer cargo enlistment and basic matching. ● Currency management	
Deliverables: First Prototype of Wheelancer	

WP5: Low-Level Design Report	
Start Date: 12.10.2021	**End Date: TBD**
Leader: Ümit	**Members Involved: All Members**
Objectives: Design and elaborate each module in the project so that the coding process can progress without an issue	
Tasks: ● Writing the modules in detail ● Writing the architecture in detail	
Deliverables: Low-Level Design Report	

WP6: Database Implementation	
Start Date: 26.09.2021	**End Date: TBD**
Leader: Onat	**Members Involved: Onat, Berk**
Objectives: Database Implementation	
Tasks: ● Design the database using ER diagrams ● Convert ER diagram entities into database schemas and normalize them ● Write the corresponding MySQL queries	
Deliverables: Database code in MySQL	

WP7: User Interface Implementation	
Start Date: 29.09.2021	**End Date: TBD**
Leader: Maruf	**Members Involved: Maruf, Ümit**
Objectives: User Interface Implementation	
Tasks: ● Implementation of the user interface with flutter ● Making the implementation fits with android and ios	
Deliverables: User Interface code in Flutter	

WP8: Second Prototype	
Start Date: 01.01.2022	**End Date: TBD**
Leader: Ege	**Members Involved: All Members**
Objectives: Build the second prototype of the Wheelancer by implementing some complex functionality.	
Tasks: ● Implementing some complex functions ● Fixing old functions if necessary	
Deliverables: Second Prototype of Wheelancer	

WP9: Final Implementation	
Start Date:TBD	**End Date: TBD**
Leader: Berk	**Members Involved: All Members**
Objectives: Finalize the implementation according to the feedbacks from previous prototypes	
Tasks: ● Debugging frontend and testing edge cases. ● Database query optimization. ● Backend swagger documentation.	
Deliverables: Swagger documentation, final implementation, documentation for environment deployment.	

WP10: Final Report	
Start Date: 1.02.2022	**End Date: 15.05.2022**
Leader: Ümit	**Members Involved: All Members**
Objectives: Summarize the whole project, the requirements implemented, the improvements made. Give a user guide for the project, the details about the implementation and a maintenance plan	
Tasks: ● Summarize the whole implementation process ● Write functional requirements implemented or could not be implemented ● Write the details of the current architecture ● Write a maintenance plan ● Write a user guide for potential users	
Deliverables: Final Report	

3.4 Ensuring Proper Teamwork

The senior design project is one of the biggest projects that we have undertaken in Bilkent. Therefore we need to keep proper track of everything about the project. Therefore, as a team, we will hold weekly meetings to ensure that we are following our deadlines for each task since all of the team members know that a successful project is a result of healthy communication. In these meetings, new tasks will be assigned, the status of the ongoing task will be analyzed, the finished tasks will be finalized and we will brainstorm frequently in order to find new ideas to improve our mobile application. Moreover, we will divide tasks in a way such that one person does not undertake a huge task and burden himself/herself and perform poorly.We will divide these huge tasks into smaller tasks to ensure work balance. We believe that healthy communication combined with this approach will ensure that we will build a successful mobile application at the end of the year.

A Google Drive folder will be used to keep track of deliverable reports such as this one. This way we will be able to work simultaneously and have access to the diagrams drawn by other team members so that there is no misunderstanding while developing the project modules.

We will use Git as a version control system and GitHub to store our progress in the coding part. This way we will ensure that we will make proper progress while developing our mobile application.

3.5 Ethics and Professional Responsibilities

One of the issues regarding our application is that as the developers we cannot see the contents of the packages of our customer due to privacy. Nevertheless, this also means that Wheelancer can be used to transfer illegal substances. Another issue is that the courier can steal the package, which will cause problems for the customers and decrease the reputation of our app. In order to prevent these types of incidents, we will use a User Agreement to ensure any misconduct will be penalized by the authorities.

In order to show the current location of the package, we will require the couriers to keep their GPS on while delivering the package. This can lead to problems that can endanger the safety of the courier because of unsafe roads or routes. Therefore, we will give the courier to choose whether he/she wants to deliver a particular package or not.

We will provide an agreement about the access permissions when a user downloads Wheelancer. This way every user will know how we process their data and decide whether or not they want to use Wheelancer or not.

In order to ensure that the couriers have licenses for their car type, we will require the photo of their license which will be stored in our database. Therefore, any data leak will be a violation of privacy rights. We will ensure that we will not share this data and no one can reach this information using our app apart from the people who need this information.

Last but not least, we should be careful about the APIs, libraries and other things because of copyright issues. Therefore, we will use free alternatives to avoid any legal trouble. We will also reference the sources if we utilize their technologies in any way.

3.6 Planning for New Knowledge and Learning Strategies

For UI design, we will use Flutter, nevertheless, none of us has used Flutter before. We will read the Flutter documentation page [3], which is very detailed and

contains various information about the widgets used in Flutter. There are also sample videos in the documentation pages that show how a particular widget behaves in a mobile app. Briefly, we will use the documentation and provided example videos to enhance our knowledge about Flutter and design our application.

For the backend service, we will use REST API using Express.js framework for Node.js which will communicate with our database and frontend to provide the main functionalities of our application [4]. In order to do so, we need to understand the concept of routing and middleware with help of the official documentation page[5]. In addition to It is important to learn Javascript's mapping functions and JSON parsing is essential to manipulate and formulate the data [6].

For the database, we will use MySQL. We already know the fundamentals of MySQL and ER diagrams thanks to the CS353 course. Furthermore, we will recall advanced queries with help of W3Schools MySQL tutorials [7]. Moreover, for location, routing and map rendering functions we will use Google's Maps Javascript API which has official documentation and example codes for us to play around and learn regarding our needs[8].

4. 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.

Currency System: The pool that holds money temporarily for safety purposes.

Good: Cargo that is being transported by courier.

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

[4] “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].

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

[6] “JavaScript,” *MDN*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Accessed: 03-Nov-2021].

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

[8] *Google*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/overview. [Accessed: 03-Nov-2021].