**GROUP B6 SDI REPORT**

Jared Awiti N0992216

Toby Allwood N0921157

Daniel Mellors N0995387

Tutor: Peter Blanchfield

**Abstract**

This project is an e-transport marketplace. This project aims to allow for a more leisurely experience for cargo owners, transportation companies and drivers. This will allow for easy interaction between the three, as well as remove the time taken for cargo owners to find their ideal drivers. To achieve this my group will be outlining the project aims, creating diagrams and creating the application. To create the application, we will be using Qt Creator for the GUI and backend, and Sqlite for the database. We will also be using Doxygen for automatic documentation generation. Finally, we will be using git for the remote code repository. For this project to be deemed a success, we will have created an application that allows cargo owners, transportation companies and drivers to communicate and execute an order with ease.

**Plagiarism declaration**

This report and the software it documents is the result of my own work. Any contributions to the work by third parties, other than tutors, are stated clearly below this declaration. Should this statement prove to be untrue I recognise the right and duty of the Board of Examiners to take appropriate action in line with the university’s regulations on assessment.

Name……………………………Jared Awiti………………………………………………………………

ID NO…….N0992216……………………………………………………………………………………...

**Contents**

Introduction…………………………………………………………2….Jared

Project Plan…………………………………………………………2….Jared

Requirements……………………………………………….2….Jared

Risk and Mitigations………………………………………..13….Jared

Gantt Chart………………………………………………….16….Jared

Monitoring Tool……………………………………………..17….Jared

Architecture Design………………………………………………..17….Jared

Use Case Diagram…………………………………………17….Jared

Activity Diagram……………………………………………19….Jared

Class Diagram……………………………………………..23….Jared

Sequence Diagram………………………………………..25….Jared

Components Diagram…………………………………….27….Jared

FSM Diagram………………………………………………28….Jared

Communication Diagram………………………………….29….Jared

Deployment Diagram………………………………………30….Jared

Implementation……………………………………………………30….Jared

Testing………………………………………………………………32….Jared

Video Link…………………………………………………41….Jared

Conclusion…………………………………………………………41….Jared

Reflection…………………………………………………………..42….Jared

(All done by Jared)

**Introduction**

Traditionally, drivers have to wait in long queues to get into a cargo hall. This will allow cargo owners to find drivers and drivers to find cargo to deliver. This is time-consuming for both parties. In these halls, both drivers and cargo owners had to talk to many people to find a suitable cargo or driver to move the goods. As a result, it costs a lot of time and money, this was not easy for both parties.

The aim of this project will be to make this an easier experience. With technology, we can create a better more productive environment for all parties. An e-marketplace is a virtual online market where organisations register and then can conduct business over the Internet. We will be using this concept to create a marketplace for cargo. Cargo owners, drivers and transport companies will be able to register and easily deliver cargo to the customer.

This will allow cargo owners and drivers to save time as well as money. It will make all parties more productive. Furthermore, the ability to track the cargo will be on the application. This allows cargo owners to have information on whether the cargo has arrived at the customer or not.

We will achieve this using QT to create a GUI and backend. This will allow for a user-friendly GUI. SQlite will be managing the database for signing in and orders. Git will be used as a remote repository to make sure no work is lost during the project.

**Project Plan**

Requirements

| **Requirement** | **Description** | **Implications** | **Tasks** |
| --- | --- | --- | --- |
| Cargo owner must be able to create an account | Must be able to click the signup button and create their account. This includes entering their details, username and email. Home and business address and creating a password into a form. Furthermore, a verification message should be sent to their mobile to make sure they are a person | In order for a user to create an account the system must be capable of receiving input and storing that input in the appropriate structure. In this case, a database would be required to store individual account information. | -Create sign up button that leads to the form  - Create sign up form with relevant information to input and save in the database  -Error messages if input boxes blank or incorrect format  -Once successfully signed up send a verification message to the mobile number |
| Cargo owner must be able to view their profile | Must be able to click on a profile icon/button that will enable them to view their details | Information stored by the system should be able to be recalled by the user easily in the form of a profile screen. | -Profile button top corner  -Icon/Picture  -Once clicked can view their details |
| Cargo owner must be able to edit details | Must have the ability to change and save their details if any is currently incorrect | Information stored by the system should be able to be edited by account holders, the system must be able to keep the database updated easily. | -Click an edit button  -Change details  -Save details in a database |
| Cargo owner details must be encrypted when stored | When passwords are saved must be encrypted using the hash method to ensure security | The system will need to be able encrypt/decrypt sensitive information when storing it. Ideally the system should encrypt data before sending it to the database to be stored to improve security. | -Encrypt details when saved in the database  - Hash? |
| Cargo owner must be able to sign in | When the Cargo owner is logged out must be able to type in the correct username and password, this should allow them to get into their account | The system will need to track the sign/in status of users to allow them to be logged in/out. Additionally the system should be able to match entered credentials to stored account credentials in order to confer access to specific account features and information. | -Create a login button that sends the user to a form.  - If correct details are entered accept  - Else push an error message and reload the page. |
| Cargo owner must be able to log out | When Cargo owner is logged into their account, should be able to click a button to logout | As above the system should be able to track if a user is currently logged out. The system will need to manage this log in status and revoke access to any privileged content if the status is not correct. The system should also be deleting or preventing access to any data stored at the client level when a user logs out. | - When logged in a logout button should appear.  - When clicked will log the user out and return to the login form |
| Program must be able to calculate shipping rate | Calculation of shipping rate should be shown clearly and in detail. Variables include source, destination and lorry type | The system should be able to store data regarding locations, rates and vehicle types. The system will be able to perform calculations with these values based on prespecified operations such as distance calculations and fuel expenditure.  *E.g. A function that receives two locations and calculates the distance between them based on existing variables.* | -Use Variables e.g. locations, rates, vehicle types etc. To calculate shipping and outline this clearly to the Cargo owner |
| Cargo owners must be able to place a cargo order | They Must be able to view available cargo and press a button to place an order | In order to fulfill this the system should be able to track all available cargo and hold information regarding type, weight, distance etc. This information should be presented to cargo owners and must be able to receive a flag that assigns it to an individual account, removing it from the list of available cargo. | - Create Form showing all available cargo  -When clicking on cargo create an additional form showing more details about cargo with order button  -When order button clicked display success message e.g. “Order Placed” |
| Cargo owner must be able to view order status | Once the order is placed should be able to view order status. E.g. dispatched, delivered etc | Cargo that is assigned to an individual account should be able to take additional data regarding it’s distance from destination and delivery status. Additionally it would be useful to allow notes to be attached to this data. | - on form showing active orders should be able to view the status of the order. |
| Program must be able to send a notification to cargo owner when order accepted | When the cargo order is accepted a notification should be sent to the owner | The system should have a preset template that allows for notifications to be structured based on existing variables such as deliverer, recipient and order status. A flag will be needed to indicate that assigned cargo has been accepted by a driver. The system should be able to assign messages to individual accounts to inform of order status. When logging in or while logged in the system should be able to display notifications based on the above mentioned templates. | - When a driver accepts order a notification should be sent to cargo owners account |
| Driver must be able to create an account | Must be able to click a signup button and create an account. For the driver, the details must include the driver's personal details, NI number, Driving licence ID, picture, and email. home /business address, username, password, type of lorry and mobile number. The verification message should be a mobile number to confirm identity. | The accounts database must have a separate table for Drivers. The system should be able to generate unique verification codes and destroy them after use. The system should be able to send SMS messages with verification codes and track if these have been entered.  Additionally it may be useful for the system to be able to differentiate between account types such as Driver or Cargo Owner in order to field specific information. | -Create sign up button that leads to the form  - Create sign up form with relevant information to input and save in the database  -Error messages if input boxes blank or incorrect format  -Once successfully signed up send a verification message to the mobile number |
| Driver must be able to modify driver’s details | Must have the ability to change and save their details if any is currently incorrect | Information stored by the system should be able to be edited by account holders, the system must be able to keep the database updated easily. | -Click an edit button  -Change details  -Save details in a database |
| Driver must be able to sign in | When Driver is logged out must be able to type in the correct username and password, this should allow them to get into their account | The system will need to track the sign/in status of users to allow them to be logged in/out. Additionally the system should be able to match entered credentials to stored account credentials in order to confer access to specific account features and information. | -Create a login button that sends the user to a form.  - If correct details are entered accept  - Else push an error message and reload the page. |
| Driver must be able to log out | When Driver is logged into their account, should be able to click a button to logout | The system should be able to track if a user is currently logged out. The system will need to manage this log in status and revoke access to any privileged content if the status is not correct. The system should also be deleting or preventing access to any data stored at the client level when a user logs out. | - When logged in a logout button should appear.  - When clicked will log the user out and return to the login form |
| Program must be able to send order notification to drivers | Should receive a notification when cargo owners have a job. The driver must have the ability to either accept or reject the cargo. Details of the order should be explicitly shown to the driver before a decision is made | The system should have a preset template that allows for notifications to be structured based on existing variables such as deliverer, recipient and order status. A flag will be needed to indicate that assigned cargo has been accepted by a driver. The system should be able to assign messages to individual accounts to inform of order status. When logging in or while logged in the system should be able to display notifications based on the above mentioned templates. | -Create a messaging segment for cargo owners to message drivers. Using Sockets? |
| Drivers must be able to accept/reject cargos | Driver should be able to accept or reject the cargo with a click of a button | The system should be able to present a dialog to drivers allow them to accept/reject an order. If a driver accepts a cargo it should be removed from the pool of available cargos. If a driver rejects a cargo it should be returned to the pool of available cargos. It may be useful for the driver to be able to specify a reason for accepting/rejecting in the form of a small text field. | -Segments for cargo requests  - Two buttons accept or reject cargo  - If accepted add to current Cargo  - If rejected Ignore |
| Driver must be able to notify cargo owner | Driver must be able to send cargo owner messages/updates when loading, on road and delivered | The system should have functionality that allows the driver to easily update various pre-designated statuses for the cargo. The variables storing the current delivery status of the cargo should be updated based on the driver’s current status. The system should be able to present a text field to drivers and then store information entered within it to be passed to the cargo holder. | - Send messages/notifications to cargo owners inbox  - Could integrate in messages feature? |
| Transportation company must be able to create account | Transportation company must be able to click a signup button which will come u with a form. The company details must be placed into this form. | The accounts database must have a separate table for Transportation Company. The system should be able to generate unique verification codes and destroy them after use. The system should be able to send SMS messages with verification codes and track if these have been entered.  Additionally it may be useful for the system to be able to differentiate between account types such as Driver or Cargo Owner in order to field specific information. | -Create sign up button that leads to the form  - Create sign up form with relevant information to input and save in the database  -Error messages if input boxes blank or incorrect format  -Once successfully signed up send a verification message to the mobile number |
| Transportation company must be able to login | Must be able to enter a correct username and password in order to log into their account | The system will need to track the sign/in status of users to allow them to be logged in/out. Additionally the system should be able to match entered credentials to stored account credentials in order to confer access to specific account features and information. | -Create a login button that sends the user to a form.  - If correct details are entered accept  - Else push an error message and reload the page. |
| Transportation company must be able to log out | Must be able to click a button to log out of their account | The system should be able to track if a user is currently logged out. The system will need to manage this log in status and revoke access to any privileged content if the status is not correct. The system should also be deleting or preventing access to any data stored at the client level when a user logs out. | - When logged in a logout button should appear.  - When clicked will log the user out and return to the login form |
| Transportation company must be able to receive orders from customers | Must get a notification when a customer receives the order | The system should have a preset template that allows for notifications to be structured based on existing variables such as deliverer, recipient and order status. A flag will be needed to indicate that assigned cargo has been accepted by a driver. The system should be able to assign messages to individual accounts to inform of order status. When logging in or while logged in the system should be able to display notifications based on the above mentioned templates. | -When a driver confirms the customer received the order notification sent to the transportation company |
| Transportation company must be able to forward orders to free drivers near the source | When a driver rejects the order, the program must redirect the order to other drivers | The system needs to be able to keep track of driver locations and compare these locations to order source destinations. The system should be able to sort drivers based on distance from source. The system should be able to use a ‘rejected’ or ‘free’ flag in order to perform this comparison and suggest cargo to drivers based on proximity. | - Find solution to finding drivers closest to the source location  - If a driver rejects then passes to the next free driver which is close  - Dont recommend to the same driver multiple times. |
| Transportation company must be able to issue invoices to customers | The program must issue an invoice to the customer when the order has been accepted by a driver | The system should have a template for invoices. The system should be able to present a form to users that can take various information relating to invoices. The system should be able to automatically create invoices based on existing variables such as order completion, time taken, delivery rate, address, business name etc. | - Create invoice template  -Automatic invoice to cutomers |
| Program must be able to calculate companies’ commission for each order | Program must be able to calculate the company’s commission for each completed order. This should be easily viewable by the transportation company. | The system should store commission rates for companies in the database. The system needs to be able to perform calculations on order totals to determine commission. | - create commission rates  - apply to the revenue and show to the transportation company |
| The program could Track the driver’s location | Could be able to track the driver's location from when the order is received till delivered | The program should have some way of tracking the driver. In the time allocated, may not be able to be graphical however can have driver input their location/status. | - Create Gui or text base tracking system?  - Could be a stop base |
| Transportation company must be able to view feedback for each order and driver | When the order is delivered customer must be able to submit feedback through a form. The transportation company must be able to view this submitted feedback for each order | When a order is completed. If a transportation company decides to submit feedback the driver should be able to view it. The feedback should be saved in a database | - Create a form for feedback for customers  - Sends this form to the Transportation company after submission |
| Transportation company must be able to view the order history | The transportation company must be able to view the past order history. With dates and other details | Past orders from the transportation company should be able to be viewed by the transportation company | - Page that allows transportation to view past orders  - use database for past orders |

Risks & Mitigations

Technical Risk Assessment

| **Risk** | **Desc** | **Probability** | **Impact** | **Mitigation Plan** |
| --- | --- | --- | --- | --- |
| Inadequate access to proper technical resources. | Should the team require access to resources it does not have, such as specific software, libraries, licenses or machines that are necessary for the completion of the project it may lead to reduced scope and inability to deliver. | Low | High | This can be mitigated by comprehensively understanding the resources available before planning begins so as not to include features and development techniques that would not be possible. Additionally, regular reviews can be taken to assess what assets are available to the project in order to tune expectations accordingly. |
| Poorly defined features and requirements. | If the features of the system are not properly defined it can lead to misunderstanding as to their purpose and function, this can cause instability and lack of functionality in the software as well as result in a product that is not fit for purpose. | Moderate | Moderate | Care should be taken when planning to make sure all members of the team agree on the purpose and function of each feature, this understanding should be (where possible) compared with the tutor’s expectations to ensure accuracy to the intended function. Starting with core features and making sure each one is well understood as it is implemented will allow us to consistently build to the proper specification. |
| Poor code quality. | Code of a low standard will be harder to iterate on, harder to understand and result in more bugs and prolonged testing. | Low-Moderate | Moderate | Ensuring each member is competent with the intended language and adheres to proper standards when programming should help to mitigate this. Additionally, it may be useful to assign in order of complexity to members by their skill level. Finally, regularly auditing code will help to highlight ongoing issues with both practices and the quality of the codebase so that any issues can be resolved while they are still minor. |

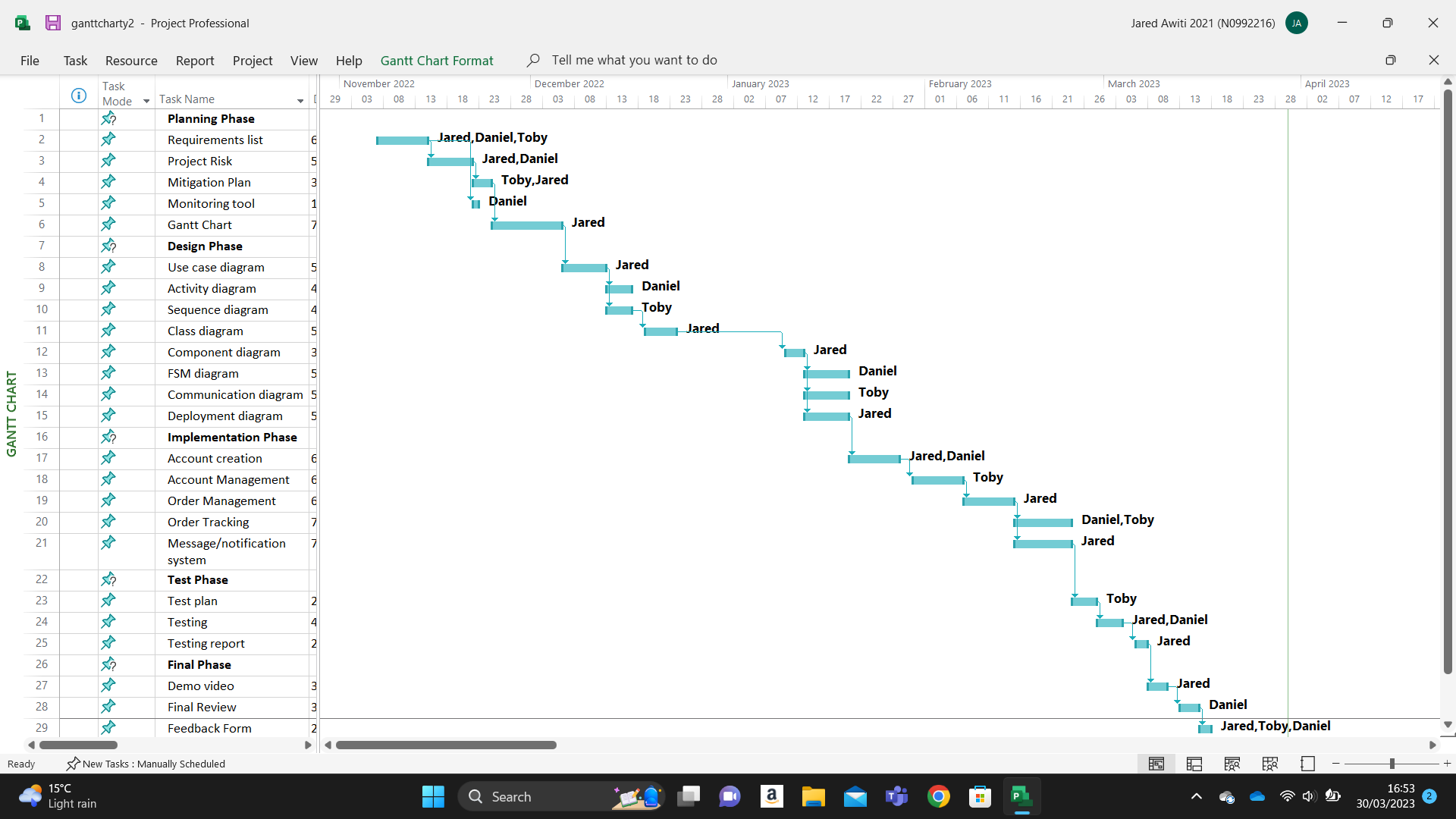
Economic Risk Assessment

| **Risk** | **Desc** | **Probability** | **Impact** | **Mitigation Plan** |
| --- | --- | --- | --- | --- |
| High maintenance cost | If the project is created at a low standard, many errors may occur. Resulting in higher maintenance costs. | low | low | Create the project to a high standard in order to lower the maintenance costs. Furthermore, thoroughly test the code to ensure all errors are dealt with |
|  |  |  |  |  |
|  |  |  |  |  |

Organisational Risk Assessment

| **Risk** | **Desc** | **Probability** | **Impact** | **Mitigation Plan** |
| --- | --- | --- | --- | --- |
| Strict deadlines | Setting strict deadlines may lead to individuals having to work overtime. This increase in work may cause stress and lower quality work. Which may lead to a less-than-satisfactory product. | Low | High | We can mitigate this by starting early and planning workflow. Starting with planning early will allow for more time for the project. As well as designing the workflow allows for the work to be evenly split. |
| Poor documentation | Poor or lack of documentation can make it difficult for third parties to understand the program. Especially if for example, a new member was added to the group. | Low | High | We plan to mitigate this by documenting thoroughly each part of the process. This includes the planning, design, implementation and testing phases. |
| Poor Communication | Poor or a lack of communication can drastically decrease productivity and increase the time taken to complete the project. | Low | High | We plan to mitigate this by having meetings every week to discuss progress on the products and the next steps. |
| Other Commitments | Other commitments may mean we have less time to focus on the project at hand. These other commitments could be class tests, coursework assignments coming up and more personal matters. | High | High | We will mitigate this by trying to plan when to focus on the coursework. However, if it is a personal matter not much can be done. |

Gantt chart



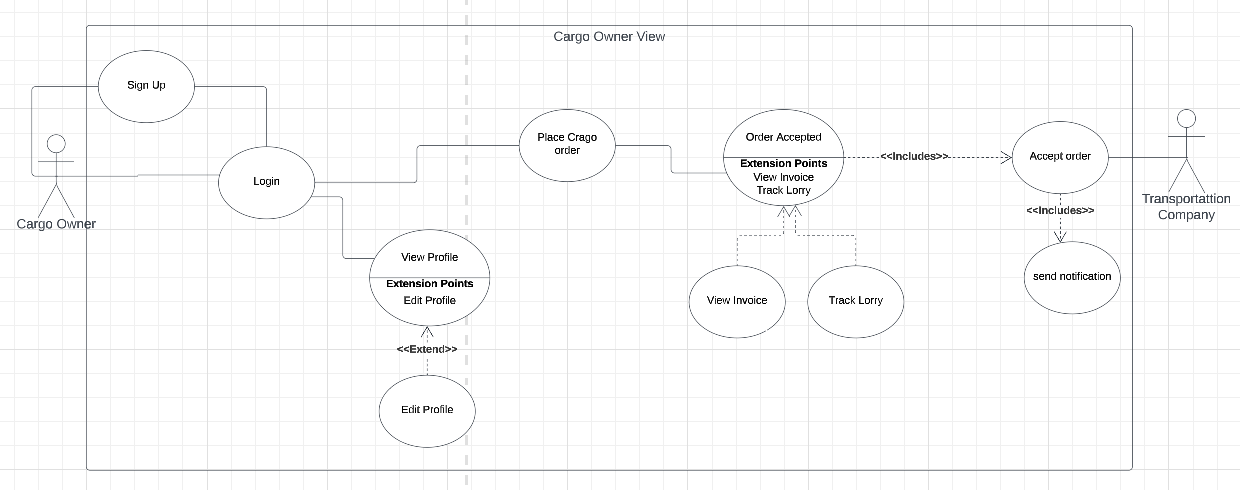
Monitoring tool

The monitoring tool we will be using is teams to track our progress and as the main form of communication. This will allow for more transparency and collaboration within the team. Furthermore, we will use it to assign tasks to each member. This will allow us to track which members of the group are contributing the most.

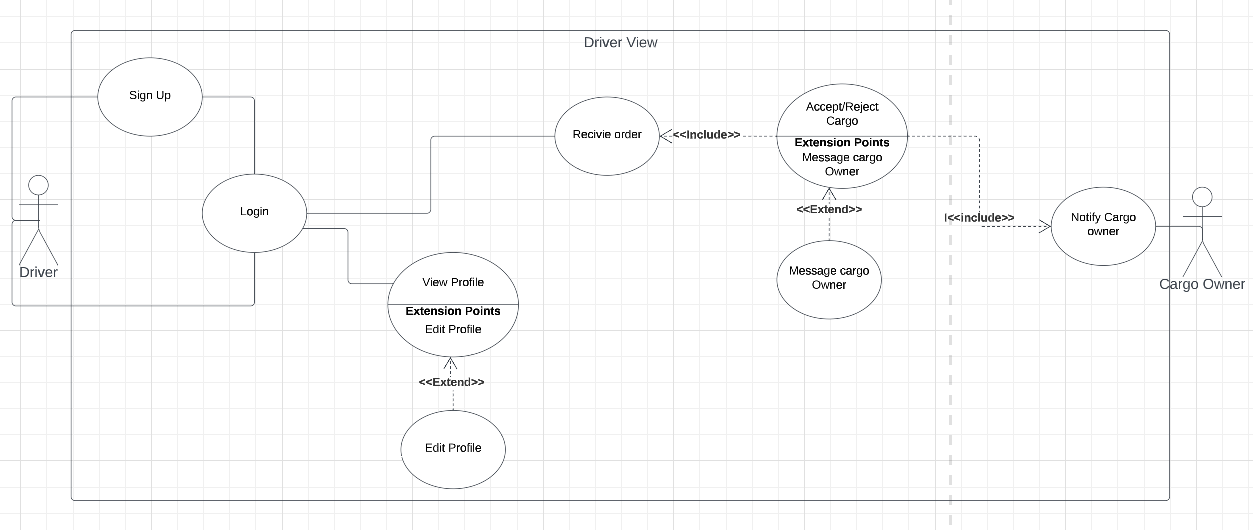
**Architecture Design**

Use case diagram

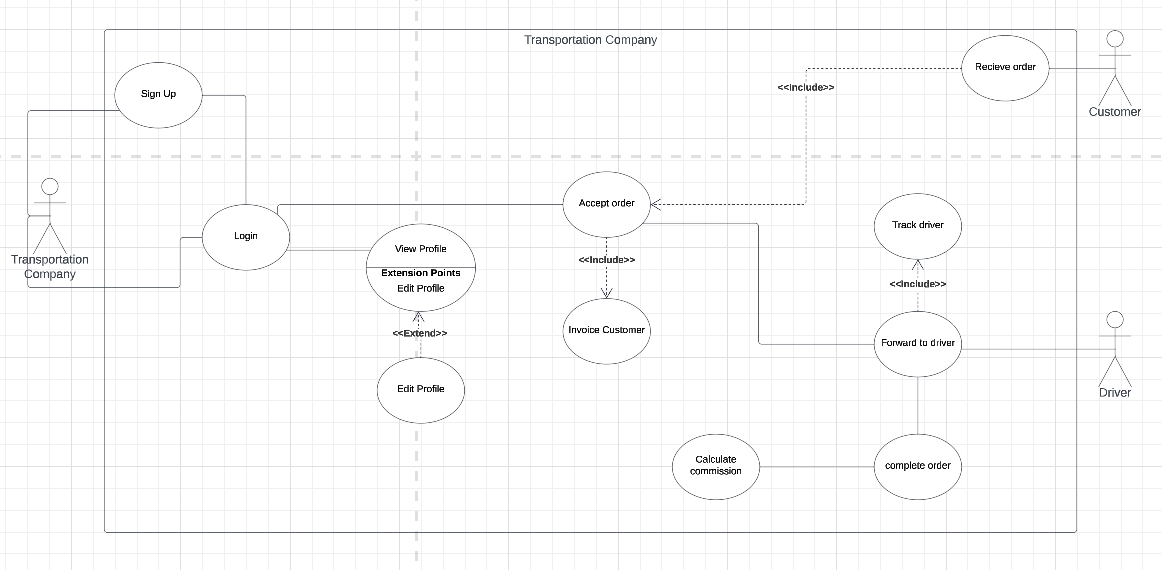
For the use cases, We have decided to create 3 different graphs. This is to show the three different perspectives, driver, transportation company and cargo owner. Each will show the relationship between use cases, actors and systems. These graphs will show the interaction between the actors.



This use case shows a cargo owner's point of view. When the application is opened the user will be prompted to either Sign up or log in. Once they have logged in they can either view their profile which will have the option to edit the profile. Or to place a cargo owner The transport company, which is another actor, needs to accept the order, this will include sending a notification to the cargo owner. Once the order has been accepted, the cargo owner can view the invoice and track the lorry.



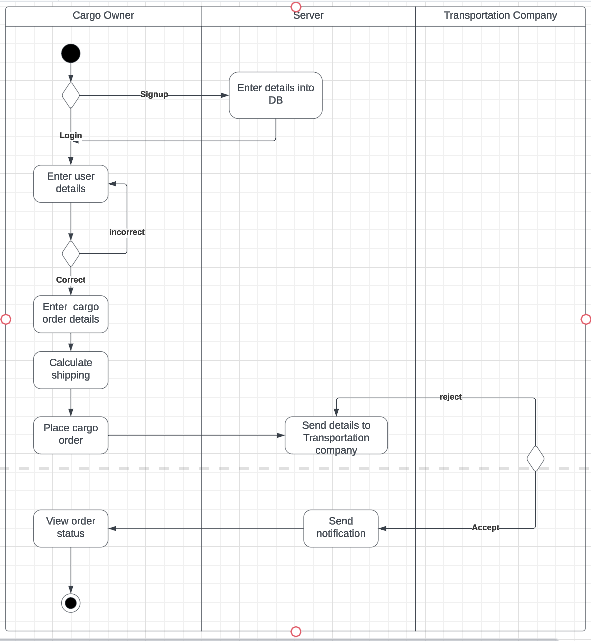
This use case shows a driver's point of view. When the application is opened the user will be prompted to either Sign up or log in. Once they have logged in they can either view their profile which will have the option to edit the profile. Once an order has been received the driver can either accept or reject the cargo. Once it is accepted the cargo will then be notified and the driver can then message the cargo owner if need be.



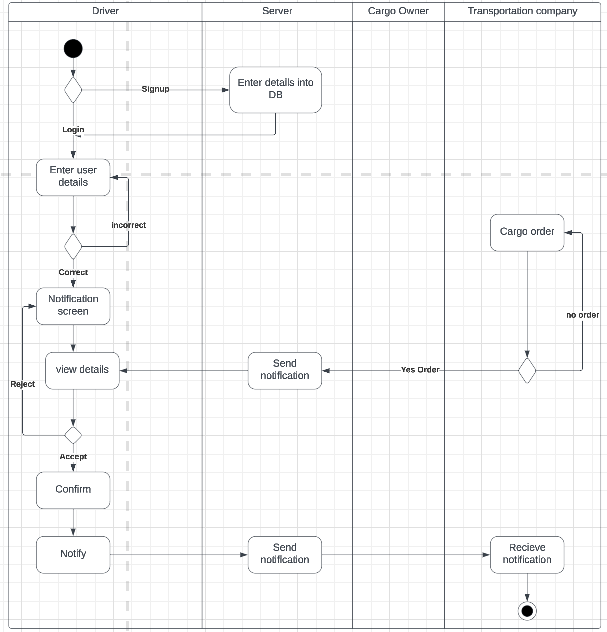
This use case shows a transportation company's point of view. When the application is opened the user will be prompted to either Sign up or log in. Once they have logged in they can either view their profile which will have the option to edit the profile. The customer will first place an order and the transportation company will accept this, an invoice will then be sent to the customer. The order will then be forwarded to the driver. The company will then be able to track the driver. Once the order is completed, the company can calculate the commission.

Activity diagram

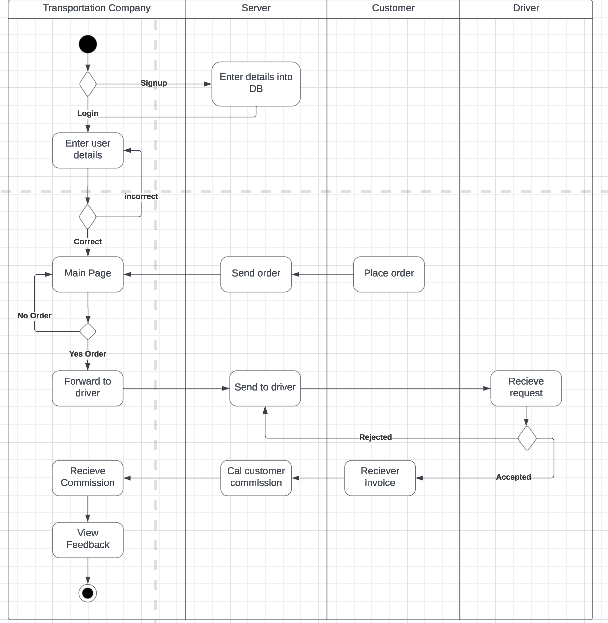
Activity diagrams describe how activities are coordinated. We have created 3 diagrams to show the activity of each user type.



The above activity diagram is from the view of a cargo owner. This shows the cargo owner will sign up. The details will be stored in the database. The user will then be prompted to log in. Once logged in the cargo owner will then enter the cargo details, calculate the shipping and place the cargo owner. This will then be sent to a transportation company. When the order is accepted a notification will then be sent to the cargo owner. Finally, the cargo owner will be able to view the order status.



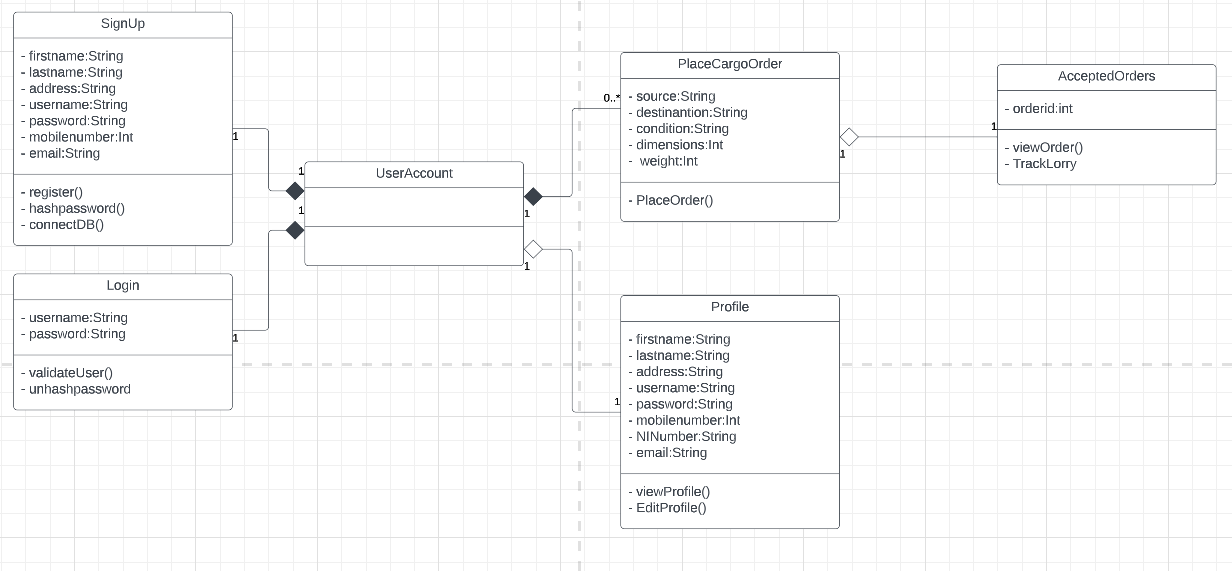
The above activity diagram is from the view of a driver. This shows the driver will sign up. The details will be stored in the database. The user will then be prompted to log in. The cargo owner will first place an order. The transportation company will forward the order to the driver. The driver will then be able to accept or reject the cargo. If accepted, a notification will be sent to the transportation company.



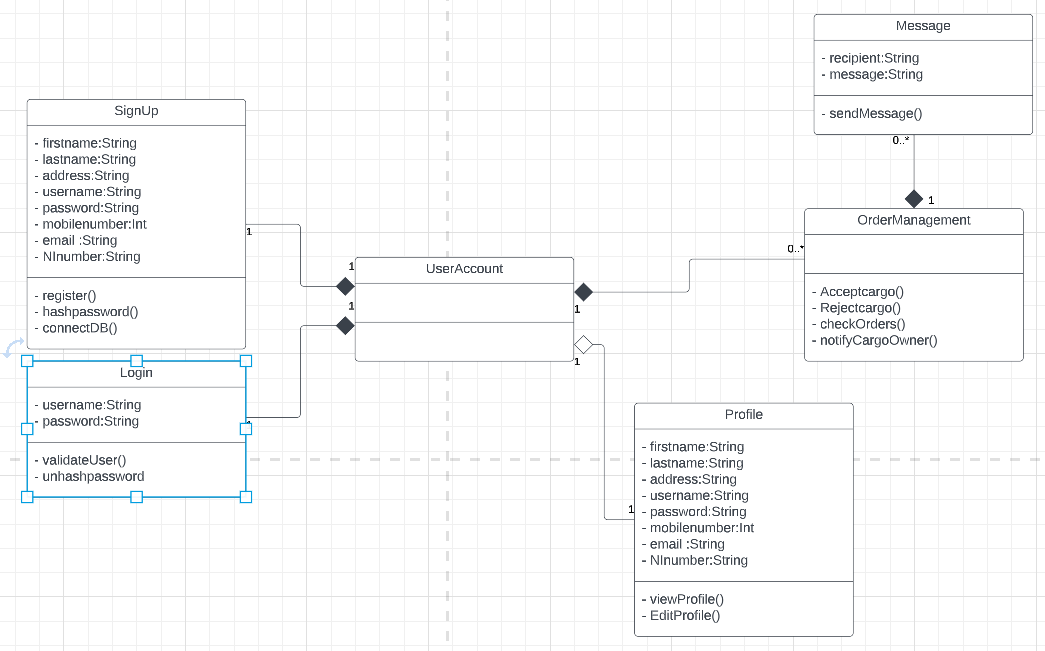
The above activity diagram is from the view of a transportation company. This shows the transportation company will sign up. The details will be stored in the database. The user will then be prompted to log in. The customer will have to first place an order. The transportation company will then forward this to multiple drivers. Once a driver accepts the customer will be invoiced and a commission will be received from the order.

Class Diagram

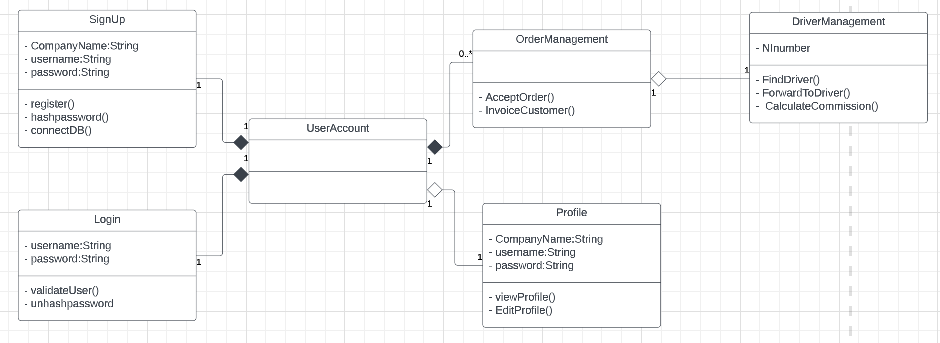
Cargo owner



Driver

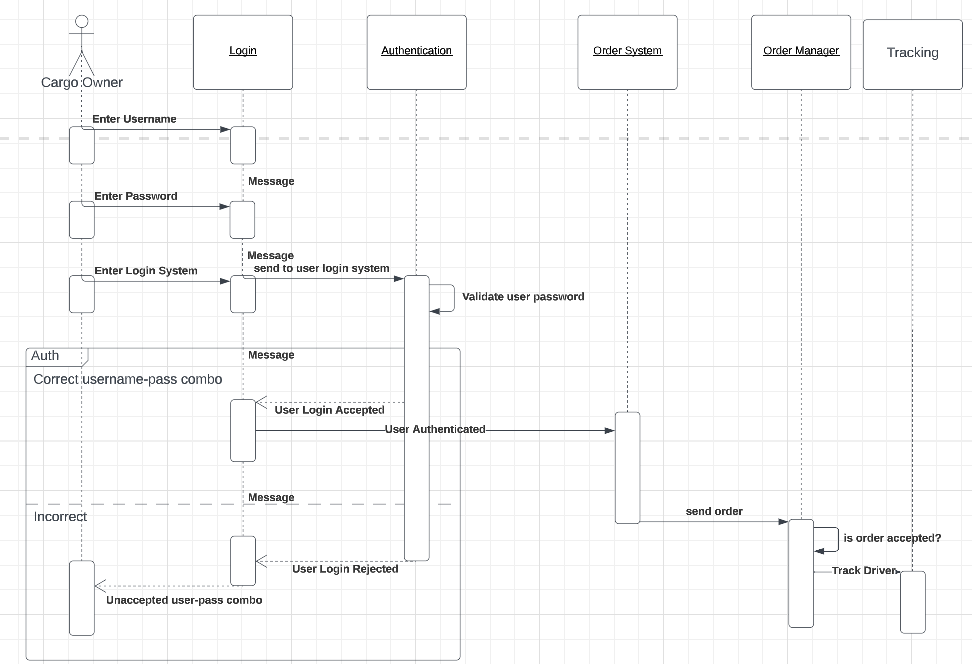


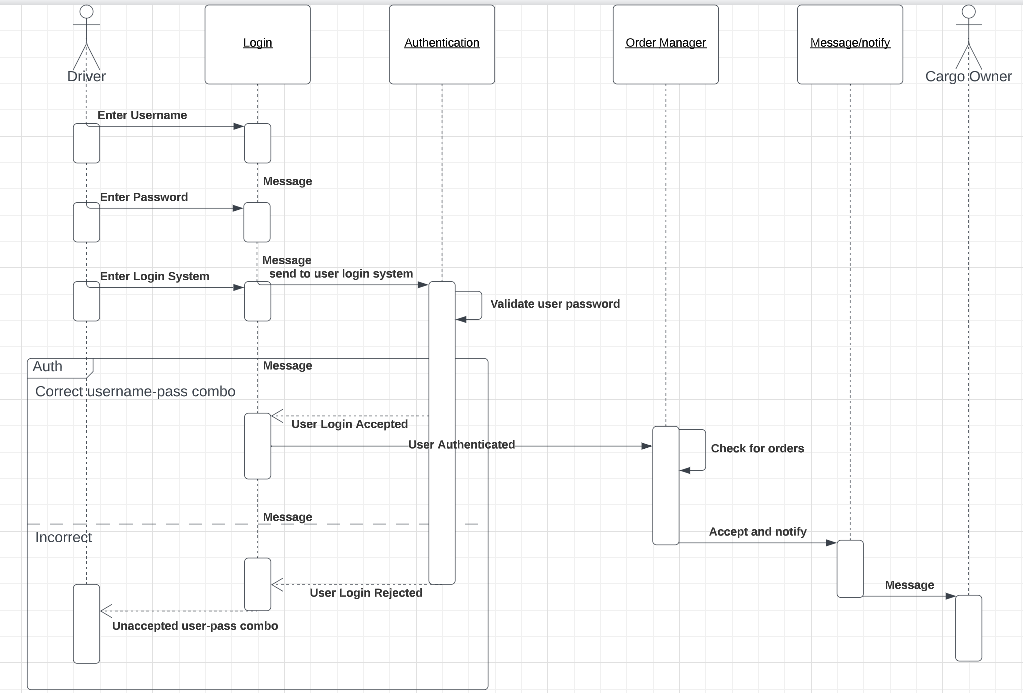
Transportation company

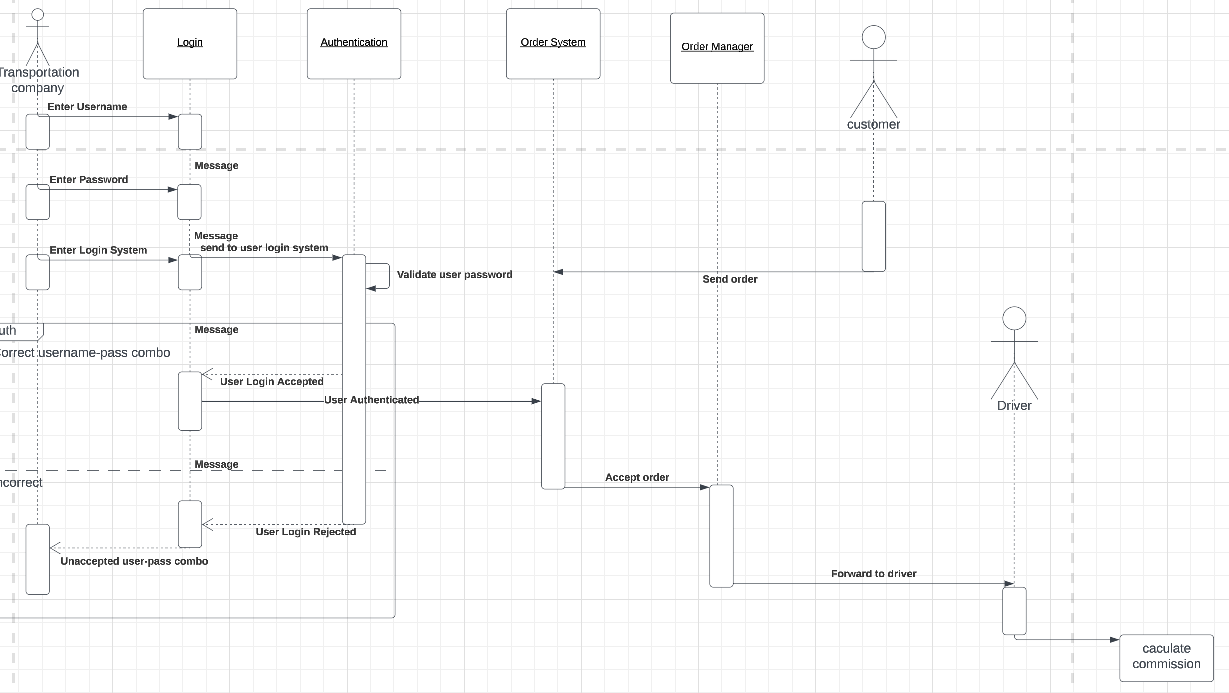


The above diagrams are the class diagrams for each user type. The initial stages of each class diagram are the same. They start with either signing up or login classes. These are in the attributes part. The functions in the sign-up class are to hash the password for security reasons, connect to the database and add this to the database. The login class will unhash the password and validate the user. The user will then have access to their account. They will then use the order management or place order classes to either accept, reject or place an order. Or they could use the profile class to view and edit their details and add this to the database. The cargo owner can then view the order status and track the lorry in the accepted order class. The driver can message the cargo owner in the message class. The transportation company use the driver management class to find and forward to drivers and calculate commission.

Sequence Diagram

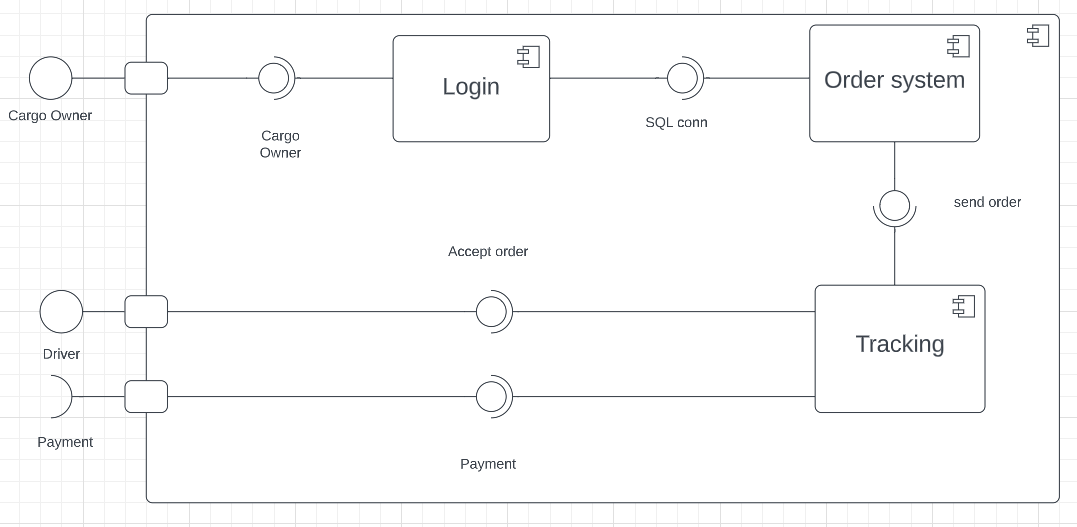


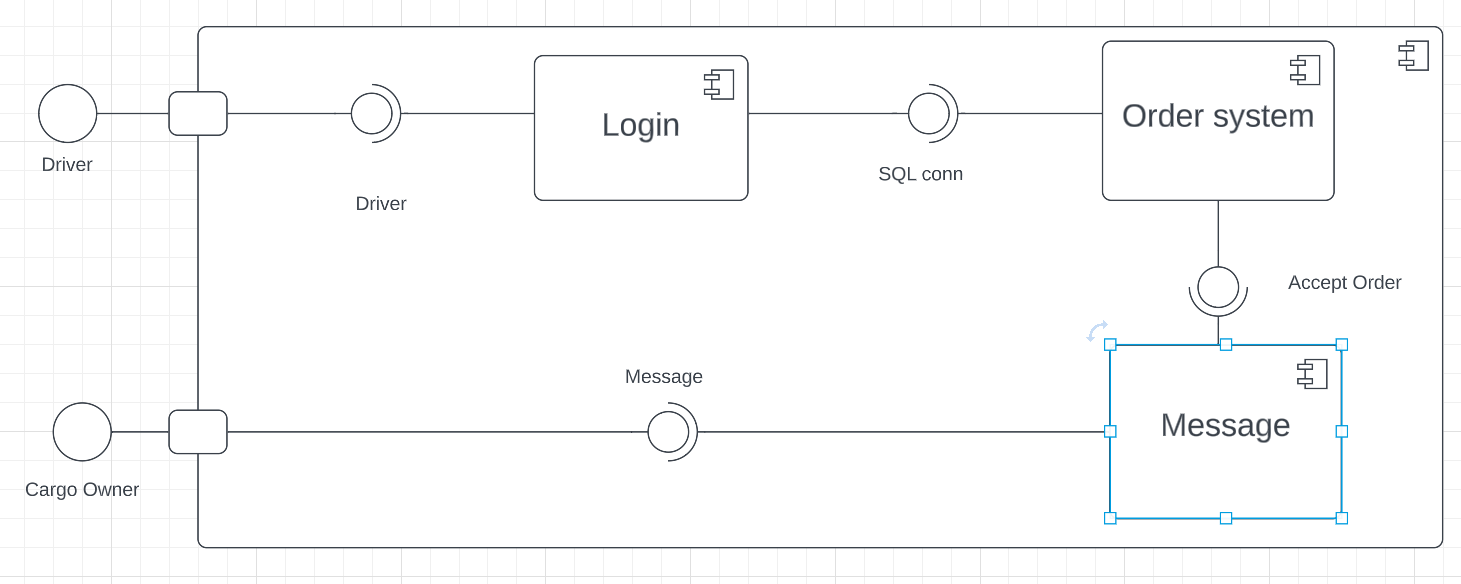


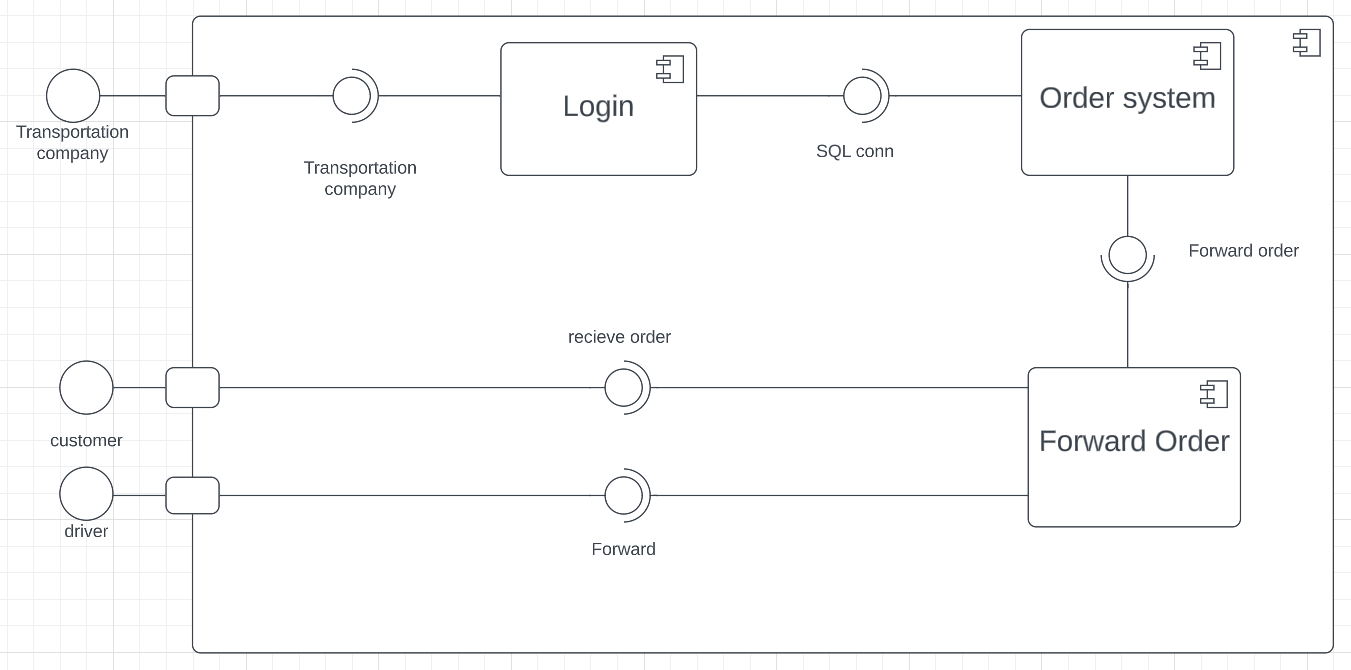


Sequence diagram is an diagram taht details how operations are carried out. It is based off the use cases. Shows the specific details on how the system carries out each task. There a sequence diagram for each user type and their different interactions.

Component Diagram







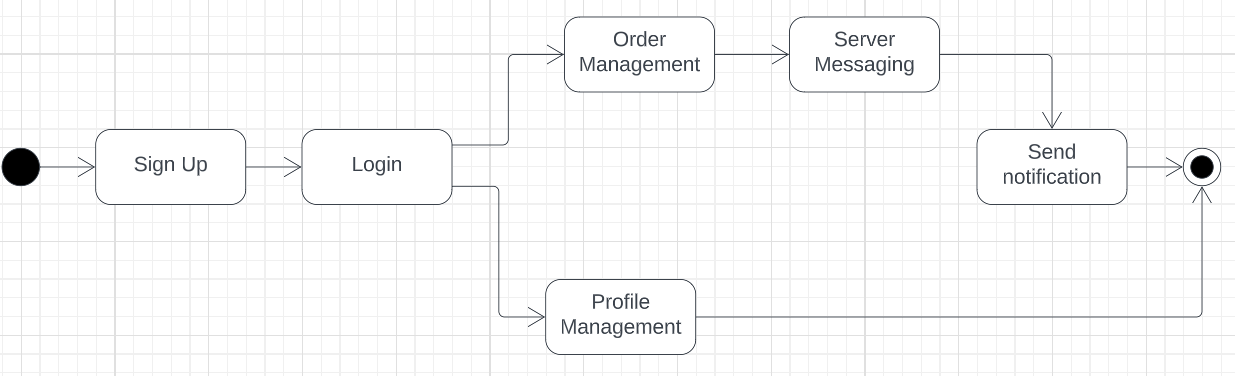
Component diagram model the physical aspects of object oriented system. Above we can see the systems all start at the login phase. This has a sql connection and once verified goes to the order system. Then the diagrams are different following the type of user.

For cargo owner, the order is accepted then gets passed to the driver. The cargo owner then receives payment.

For transportation company, the order is then forwarded to driver once the customer sends there order.

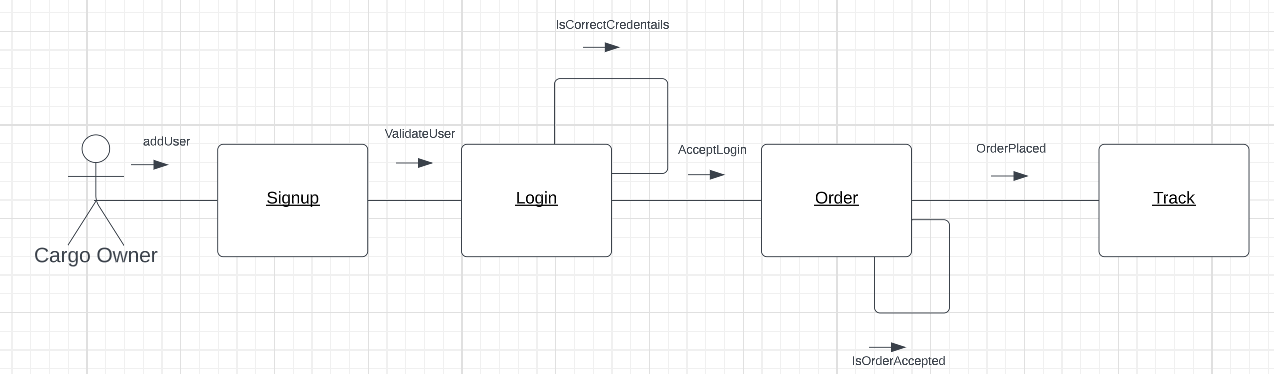
For drivers, accepts the order then is able to message the cargo owner.

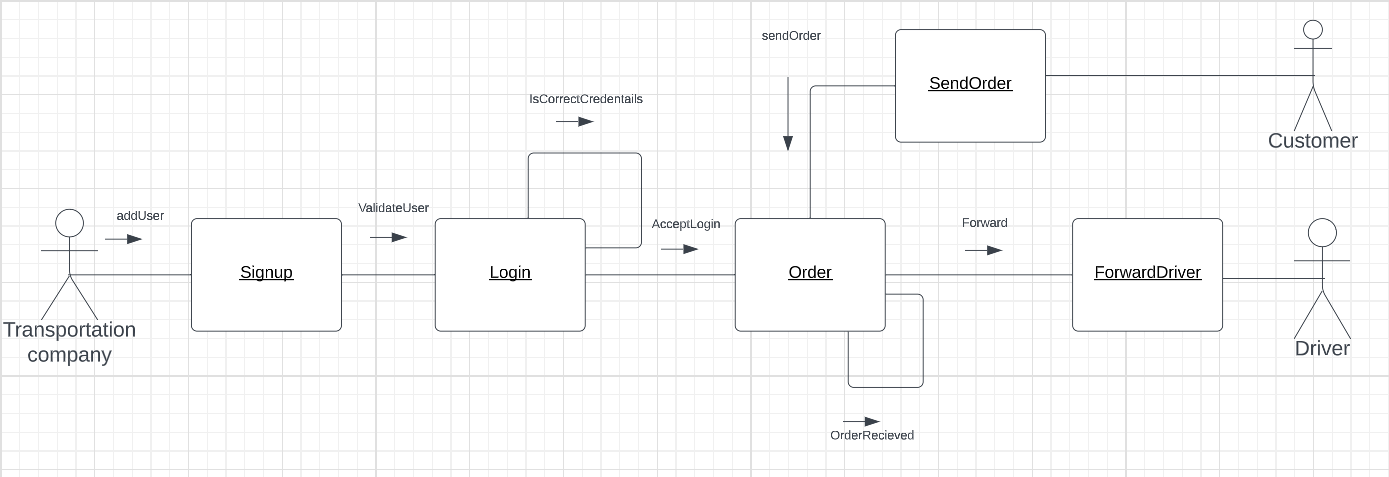
FSM Diagram

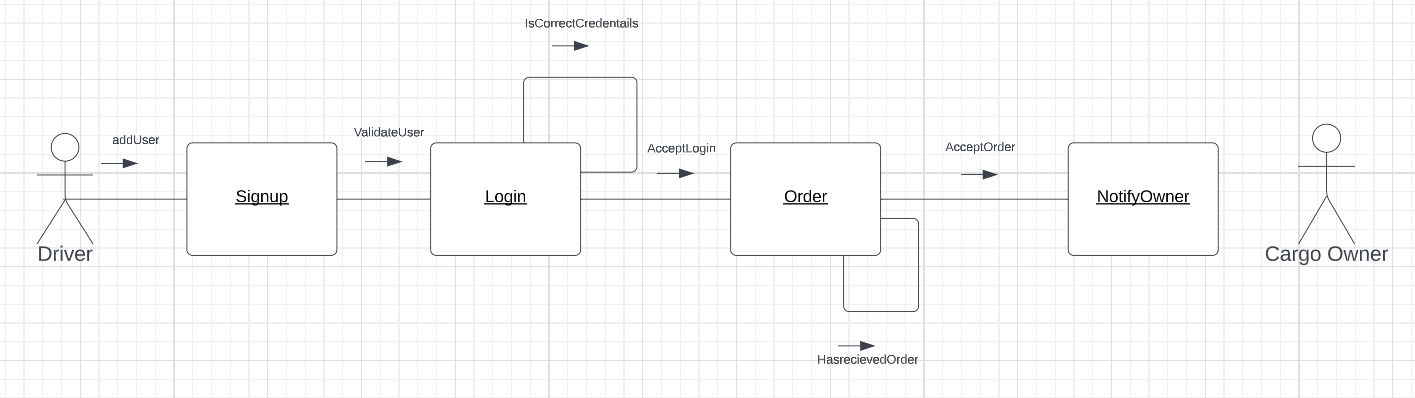


The finite state machine above shows a general view of all types of users. It is used to describe the state-dependent behaviour of an object. We can see the six general states a user will use. Firstly, the user will have to sign up and log in. They can then manage their profile or manage their orders. Afterwards, the user will be able to message other users and notify other users.

Communication UML Diagram

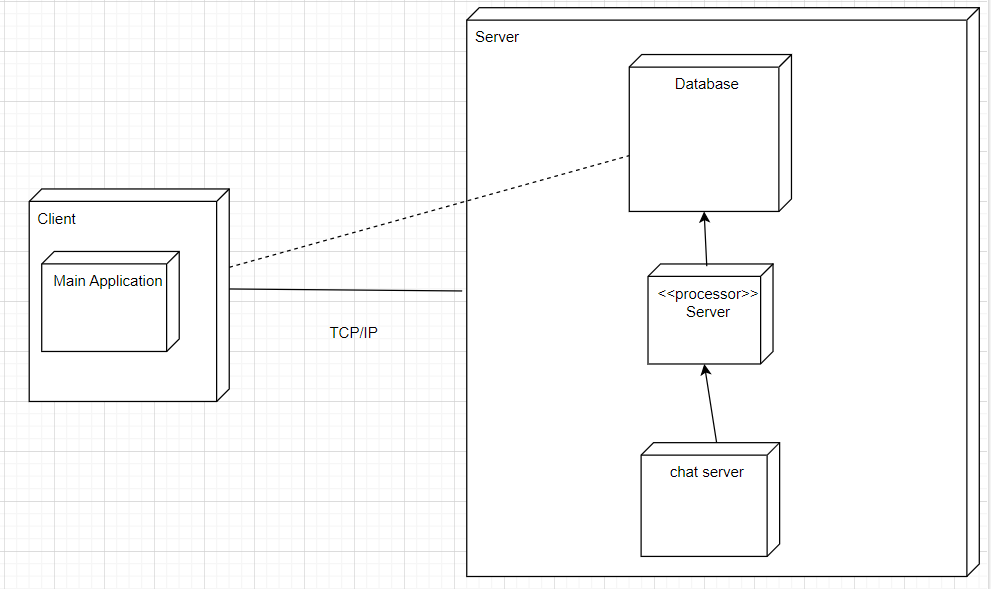






The communication diagrams above reveal the messages with the objects for each type of user. Each diagram starts the same with sign-up and logs in. The message to validate the user enables the user to have access to their account. Then it's the order phase which enables the user to manage their order. Each user has a different order message as they are trying to achieve different goals.

Deployment UML Diagram

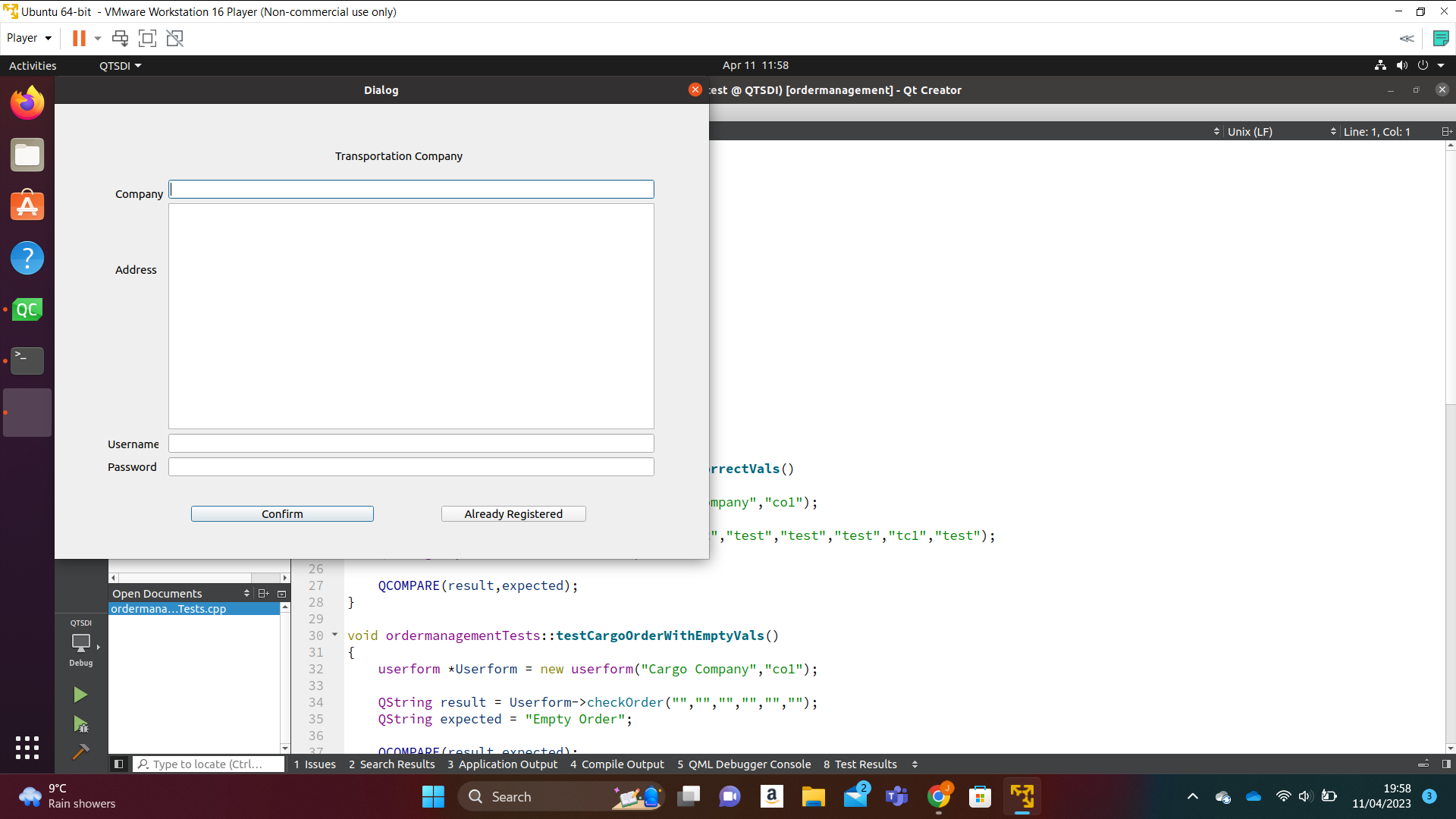


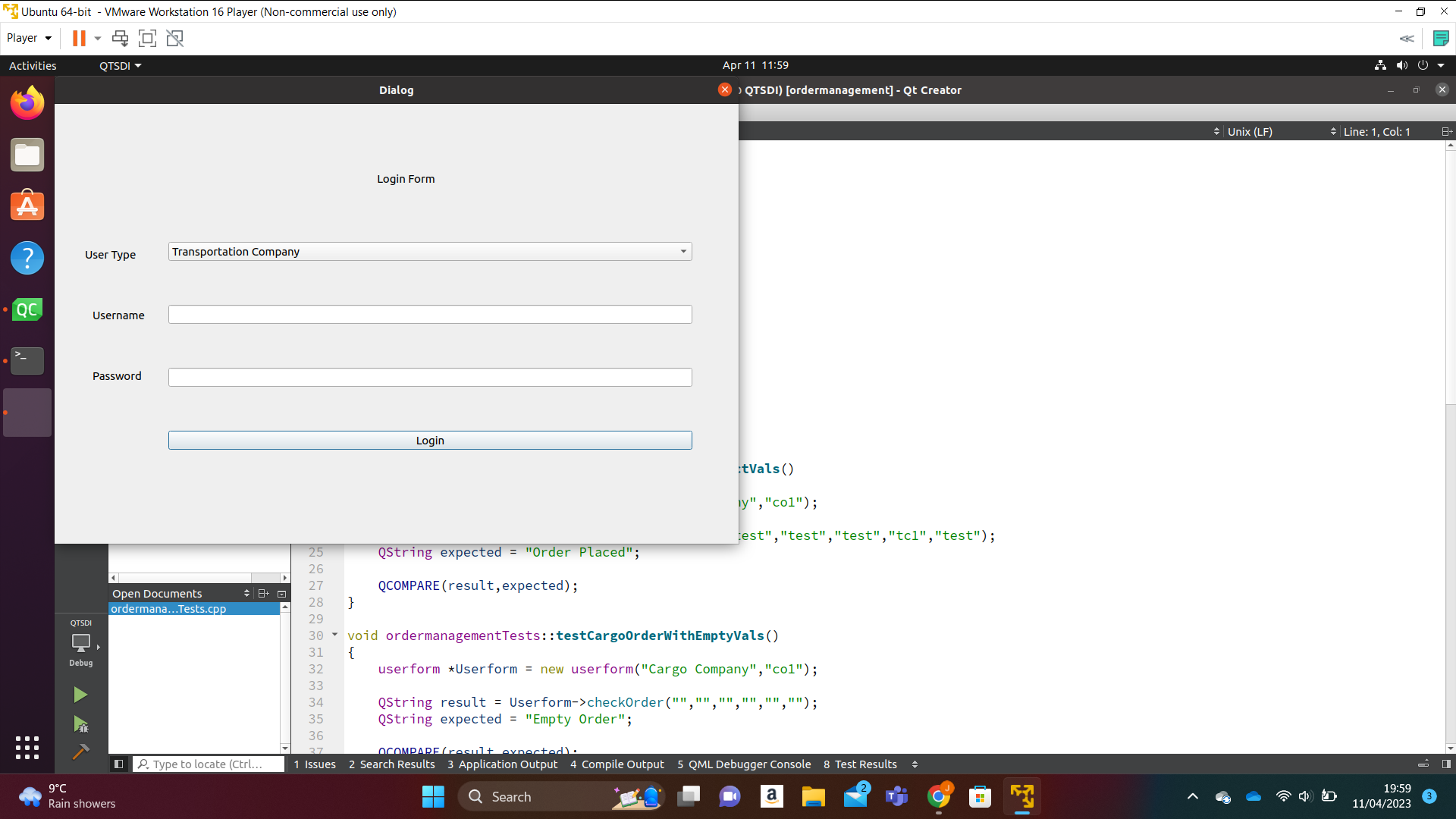
The deployment diagram above shows the structure of a run-time system. From the deployment diagram above we can see the client using the main application to communicate with the server. The server will manage the database and the chat server. This chat server will be where the users notify and communicate with one another. The database will save information such as login and order info.

**Implementation**

To create the program, I will use QT Creator, SQLite and C++. I will be using QT Creator specifically for the front-end design. This will ensure the user interface is user-friendly and accessible to all users. C++ will be used for the back end. SQLite will be used for the database, to secure users' details.

For the front end, we will be using a simple design. Here are examples of this.





We chose to use colours, that are easy to see for the text. Make the design plain and simple so users can use it easily.

I will be following the coding standards. Making it safe, secure, reliable, testable and maintainable. This means it can be used without causing harm. It stores users' information securely. It functions as expected. It can be tested. It can be maintained as more users start to use it.

We will be using threads, the Qthread library, to do multiple tasks simultaneously. This will allow for a smoother user experience. If we get to it we will use client-server networks for the messaging feature.

This will be the basis of the implementation.

**Testing**

We will be testing the main functionality of the program. For this, We will use a mixture of unit and Manual testing. For the unit testing, We will be using the QTest library. This will help create test suites to test the program thoroughly.

Certain Unit tests failed during testing due to QTest library not being able to access the database. However on manual testing the pass. As a result there result is pass.

| ID | 1 |
| --- | --- |
| Test Type | Qualitative |
| Description | Cargo Owner Register entering all correct details |
| Success Criteria | Adds details to database and sends user to login page. |
| Mitigation Procedure | Check database connection and debug inputs |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 2 |
| --- | --- |
| Test Type | Qualitative |
| Description | Cargo Owner Register entering an already existing username |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check database creation function. Check database if username already exists. |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 3 |
| --- | --- |
| Test Type | Qualitative |
| Description | Cargo Owner Register not entering any details |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check database and check error message in code |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 4 |
| --- | --- |
| Test Type | Qualitative |
| Description | Transportation Company Register entering all correct details |
| Success Criteria | Adds details to database and sends user to login page. |
| Mitigation Procedure | Check database connection and debug inputs |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 5 |
| --- | --- |
| Test Type | Qualitative |
| Description | Transportation Company Register entering an already existing username |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check database creation function. Check database if username already exists. |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 6 |
| --- | --- |
| Test Type | Qualitative |
| Description | Transportation Company Register not entering any details |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check database and check error message in code |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 7 |
| --- | --- |
| Test Type | Qualitative |
| Description | Driver Register entering all correct details |
| Success Criteria | Adds details to database and sends user to login page. |
| Mitigation Procedure | Check database connection and debug inputs |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 8 |
| --- | --- |
| Test Type | Qualitative |
| Description | Driver Register entering an already existing username |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check database creation function. Check database if username already exists. |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 9 |
| --- | --- |
| Test Type | Qualitative |
| Description | Driver Register not entering any details |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check database and check error message in code |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 10 |
| --- | --- |
| Test Type | Qualitative |
| Description | Cargo Owner Login with correct details |
| Success Criteria | Sends Cargo Owner to cargo owner main page |
| Mitigation Procedure | Check database connection and debug inputs. Also check whether user is being redirected correctly |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 11 |
| --- | --- |
| Test Type | Qualitative |
| Description | Cargo Owner enter incorrect details to login |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check database and code |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 12 |
| --- | --- |
| Test Type | Qualitative |
| Description | Transportation company Login with correct details |
| Success Criteria | Sends transportation company to transportation main page |
| Mitigation Procedure | Debug code and check database aswell as database connection |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 13 |
| --- | --- |
| Test Type | Qualitative |
| Description | Transportation Company enter incorrect details to login |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check database and check error message in code |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 14 |
| --- | --- |
| Test Type | Qualitative |
| Description | Driver Login with correct details |
| Success Criteria | Sends driver to driver view |
| Mitigation Procedure | Debug code and check database aswell as database connection |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 15 |
| --- | --- |
| Test Type | Qualitative |
| Description | Driver Login enter incorrect details to login |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check database and check error message in code |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 16 |
| --- | --- |
| Test Type | Qualitative |
| Description | Cargo Owner places correct cargo order |
| Success Criteria | Updates database and sends to transportation company |
| Mitigation Procedure | Check database connection and debug inputs |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 17 |
| --- | --- |
| Test Type | Qualitative |
| Description | Cargo Owner leaves empty slots for cargo order |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check database for not null values and debug code |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 18 |
| --- | --- |
| Test Type | Qualitative |
| Description | Transport Company Accepts order |
| Success Criteria | Updates Database and send to drivers |
| Mitigation Procedure | Check database connection and database columns |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 19 |
| --- | --- |
| Test Type | Qualitative |
| Description | Transport Company accepts order with wrong ID |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check for display error positioning in code |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 20 |
| --- | --- |
| Test Type | Qualitative |
| Description | Driver accepts order with correct ID |
| Success Criteria | Updates database status and Displays order |
| Mitigation Procedure | Check instance and database connection |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 21 |
| --- | --- |
| Test Type | Qualitative |
| Description | Driver rejects order with correct ID |
| Success Criteria | Removes order from view |
| Mitigation Procedure | Check instance and database connection |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 22 |
| --- | --- |
| Test Type | Qualitative |
| Description | Driver accepts order with incorrect ID |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check for display error positioning in code |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 23 |
| --- | --- |
| Test Type | Qualitative |
| Description | Driver rejects order with incorrect ID |
| Success Criteria | Displays Error message |
| Mitigation Procedure | Check for display error positioning in code |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | pass |

| ID | 24 |
| --- | --- |
| Test Type | Qualitative |
| Description | Cargo owner status change |
| Success Criteria | When driver accepts order cargo owner status should change |
| Mitigation Procedure | Try using signals and slots to implement |
| Engineer | jared |
| Test Date | 10/04/23 |
| Result | fail |

The order status in the program did not change after another user changed the status. We attempted to use Qt Signals and slots as well as a variety of different methods however it didn’t work. From our debugging, we saw the status was changing in the class but it was not being displayed in real-time.

Video link

https://myntuac-my.sharepoint.com/:v:/r/personal/n0992216\_my\_ntu\_ac\_uk/Documents/SDI.mkv?csf=1&web=1&e=JEchEF

**Conclusion**

To conclude, I believe this project was done well but there are some caveats. Firstly, the base of the project has been completed helping cargo owners and drivers to find each other easily. However some functionality was left out due to time constraints, for example, the messaging system and notification system. This would have drastically improved the project. As well as this some features could have been improved upon. For example, rejecting orders being added to the database. Furthermore, the user interface could be designed more and be less simple.

Even with these caveats, I still believe the project was successful. With more time this project could be much better

The project at the moment is secure, safe, maintainable and portable. The testing could have been more thorough but it was difficult to do this with Qt. This resulted in it being less testable than I wanted it to be.

In the future, I would like to build upon this base even more.

**Reflection**

This project was a success. As I was able to use many of the coding techniques and new things I learned in the project. This enabled me to create a good application that may be used for a cargo e-marketplace. However, the main point of this project was to work as a team like when working in an actual software company. I was not able to do this because my team member did not participate in the project. This would have been a good chance to experience coding as a team.