SDI e-Transport Marketplace

Report

Group SDI_GROUP_F2

Project Manager: Ahnaf Azad (N0958463)

Software Architect: Jaymin Jani (N0955112)

Software Developer: Adarsh Kumar (N0945235)

Software Developer: Berke Kanlikilic (T0298523)

Software Tester: Chubby Egbujie (N0953632)

Lab Tutor: David Adama

SOFT20091 Software Design & Implementation

Nottingham Trent University

Department of Computer Science

United Kingdom

April 2022

Table of Contents

Version Control 4

Individual Contributions 4

Abstract 4

Declaration of Plagiarism 5

Git Repository 5

Introduction 5

Background Research 5

- IDE 5
- UML Creator 5
- Database 6
- Design Patterns Used 6

Design 6

- Requirement List 6
- Risk Analysis 10

Use of monitoring tools 11

- Microsoft Teams 11
- GitHub 11
- WhatsApp 11
- Slack 12

Gantt Chart 12

UML Diagrams 13

- Use case diagram 13
- Activity Diagram 15
- Sequence Diagram 17
- Class diagram 18
- Associations 18
- Cohesion and Coupling 19
- Component Diagram 20
- Deployment Diagram 20
- FSM Diagram 21
- Communication Diagram 22
- Model View Controller 22
- System Architecture 23

User Interface and User Manual 23

Video Demo 26

C++ Libraries Used 27
Search and Sort Algorithms 27
Tests 28
Discussion on Result Analysis 42
Conclusions and Future Work 43
Requirements Implemented 43
Summary of Group Experience 44
Appendix 44
User Manual 44
Code Contributors Guide 45
Coding Standards Guide 45
Reference Documentation for Functions and Classes 46
References 47

Version Control

Version	Issue Date	Stage	Changes	Author
0.1	01/01/22	Report creation	Document created	Ahnaf Azad
1.0	16/01/22	Deliverable 1 added	Deliverable 1 created	Ahnaf Azad
1.1	06/02/22	Deliverable 2 added	Deliverable 2 created	Jaymin Jani
1.2	27/02/22	Deliverable 3 added	Deliverable 3 created	Adarsh Kumar, Berke Kanlikilic
1.3	27/03/22	Deliverable 4 added	Deliverable 4 created	Chubby Egbujie
1.4	23/04/22	Project report finalised	Project report finished	All members
1.5	23/04/22	Project code complete	Project code is finalised	Adarsh Kumar, Berke Kanlikilic
1.6	24/04/22	Finalise the project	Final version of program complete	Adarsh Kumar, Berke Kanlikilic

Individual Contributions

Project Manager: Organization, documentation of groupwork and report, project planning and management, general support

Software Architect: Creating and documenting UML diagrams, researching architecture for software

Software Developer: Implementing the code for the solution, research, developing the software, coding documentation

Software Tester: Researching testing methods, Testing the software, documenting the tests

Abstract

Background: The e-transport marketplace is an already massive yet growing economy in this ever-changing world of ours. In the year of 2019, 1.3 billion parcels were sent only in the UK alone. After the sweeping permanent changes of covid-19, the demand for courier services have skyrocketed, where according to Statista, 4.2 billion parcels were sent in the 2020/2021 fiscal year. That is an almost three times increase over a year. As of now, there are a magnanimous number of delivery services, and according to ibisworld, there are 256,633 couriers and local delivery services businesses in the US, an increase of 5.1% from 2021. This report presents one of our solutions to these e-transport marketplace provider, built with C, QTMake and working on a LINUX virtual machine.

Results: The results produced in this report represent that we have collaboratively created a working, feasible software, that allows customers to place orders to send courier packages. The companies will be able to receive those orders and send those orders to designated drivers. All of these can be done on the user interface created on QTMake and operatable on Ubuntu Linux virtual machine.

Conclusion: In conclusion, a working and effective system has been created and developed by our team that can be used through the process of sending a courier from and to a destination. We have been able to implement all important features and have had a meaningful teamwork experience on our skillset.

Declaration of Plagiarism

This report and the software it documents are the result of our 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 we recognise the right and duty of the Board of Examiners to take appropriate action in line with the university's regulations on assessment.

- Ahnaf Azad (N0958463)
- Jaymin Jani (N0955112)
- Adarsh Kumar (N0945235)
- Berke Kanlikilic (T0298523)
- Chubby Egbujie (N0953632)

Git Repository

https://olympuss.ntu.ac.uk/t0298523/SDI_Assessment

Introduction

As the world of e-Transport marketplace develops in this covid accepting world, we have worked on creating an e-transport marketplace for our customers who work on LINUX. The load of transferring couriers has more than doubled in the last two years and seem to have no stops to its growth. We have created a software that works seamlessly allowing customers to order a courier to be delivered to a destination and easily notified about its whereabouts.

We have thoroughly researched the internet and have set ourselves on the highest standards for creating this software. We believe we have created a solution that can be used throughout the courier system and has potential to be used among regular customer and be scaled in a mass production and distribution environment.

Background Research

IDE

We had worked on the eclipse IDE mainly for out C++ code as it had been recommended to us by our tutors and lecturers. For Java development, it has been the second most used IDE and contains many extensions and plug ins that can be used to customize the environment and make it more convenient for us. We also used QT Creator to create our user interface as it is an easy-to-use tool and have been implemented with our project ideas smoothly.

UML Creator

To create our UML diagrams, we opted for the software StarUML instead of the more popular Papyrus as our software architect was more familiar with StarUML than other software and we were confident it would help us create UMLs efficiently without our architect having to spend time learning anything new.

Database

We have chosen SQL to be used as our database for this project. We have been taught SQL in out Information and Database Engineering module and we believe it would be a database we would be able to work around effectively as a group where everyone understands it. SQL also boasts a ton of features which we can use to search and retrieve information by only using SQL queries.

Design Patterns Used

We have chosen a design pattern for our program where each screen will have all of the information you are looking for. For customers, drivers and companies, all the information is in one screen, including order history, placing and receiving orders, leaving feedback, updating the status of the order and other features. This design was chosen in mind to have every feature available at hand without having to navigate throughout a lot of pages.

Design

Requirement List

Requirement	Description	Implication	Task
Research and Planning	Research on how to begin the project and plan for agile methodology	Research will be done on what tools and IDE will be used to create the software. The libraries that will be used and the most efficient and collaborative tools that can be used will be identified at this stage. Planning in an agile methodology will also be done at this stage where meetings will be scheduled, and a Gantt chart will be created. The tasks will be delegated to the team members and the module labs that will help us build and document the project will also be recognised	T1.1: First meeting T1.2: Research ways to work on project T1.3: delegate tasks and assign roles T1.4: create Gantt chart
Create GUI	Create GUI for the customers, drivers, and company's sign up, notification and other required pages	GUI pages have to be created for all the pages to make this a working software that can be scaled and used among regular customers and users. Creating an easy to navigate GUI is	T2.1: Research what software to use for GUI T2.2: Create GUI for the sign-up and login pages T2.: Create GUI for the notification pages

		essential for scalability.	
Database	Create a database for the user accounts and orders, etc	Database needs to be able to store and access user details We may use PostgreSQL and not MongoDB as we know what specific items we have on our database, and it is structured. To interface with C++ use libqxx OR software engineers' preference **For Cargo owners**: email, address, username, password, mobile number **For Drivers**: email, username, password, Drivers license photo, phone number, address, NI number, type of lorry, lorry reg number, dimension, weight, CPC number (use link on course spec to figure out different dimensions and weights) **For company**: username, password, email **For Orders:** order number, source address, destination address, dimension, weight, conditions	T3.1: Create database T3.2: Add tables for different users T3.3: Add tables for order details T3.4: Further detail the tables for primary keys and normalisation
Sign up, sign in/ logout	Create pages for users to sign up, login and logout	Different pages are to be made for owners, drivers, and companies to sign up A page is to be made to sign in and an option is to be provided to log out	T4.1: Create a login page for all the users T4.2: Create different sign-up pages for customers, drivers, and company T4.3: Create notification pages

		When signing up, the details are to be saved to the database	
Validate CPC and reg	When signing up, Drivers CPC number and lorry reg should be validated	Use an API to validate the CPC number and lorry registration number of the driver	T5.1: Validate CPC number for the drivers T5.2: Validate the registration
Place orders	Cargo owners should be able to place orders	Cargo owners should be able to place an order where an order number will be assigned randomly and linked to that order. They will be asked for source address and destination address, dimensions, weight, and conditions, which will then be added to the database	T6.1: Create place order page for the customer T6.2: Assign orders into the database
Calculate shipping rates	Shipping rates are to be presented to the customer when placing order	Use the cargo source, destination, and weight to calculate a shipping rate To make things easy, only use weight classification ranges to offer shipping rates	T7.1: Calculate shipping rates
Receive order notifications for company	Company must receive an order notification when a cargo owner places an order	Send out a notification to the company which will opt them to forward the order to a nearby driver It should also show the company's commission on the shipping, which we will assume to be 35% on the shipping cost	T8.1: Create company order notification T8.2: Add function to forward order to driver T8.3: Calculate commission for the company T8.4: Present commission on page
Receive order notification for driver	Company should be able to forward the order to a nearby driver should get a notification about it	When the order is forwarded, a notification is sent to a driver located nearby to the source of the cargo. The notification should include details of the source, destination, cargo	T9.1: Create driver order notification T9.2: Present details to the driver about order T9.3: Present fee for order T9.4: Create buttons to accept or reject order

		details and fee (65% of shipping rate) The driver should then be able to accept or reject the order If rejected, it will be forwarded to the next driver nearby if no driver is found, where a message will say “no free drivers found” to the company When an order is accepted by a driver, an order invoice listing the shipping rate and cargo details should be sent to the cargo owner to their email	T9.5: Create invoice and send to customer and company when order accepted
Cargo owner view order status	Cargo owner must be able to view the order status	Drivers should have a current order page where they are provided options to notify the cargo owners of “loading”, “on road” and “delivered” option The customer will have an order status page where the status of the order will be updated according to the driver	T10.1: Create order status page T10.2: Add buttons to update the order status T10.3: Send notification of order status to customer
Feedback (SHOULD)	Cargo owners should be able to send feedback about their orders	A feedback page should be added where they will be prompted to rate the drivers out of 5 and add a comment This feedback should be sent to the company	T11.1: Create feedback page T11.2: Send feedback to company
Encrypt user details (SHOULD)	For security, all user details should be encrypted	Encryption should be done for customer and driver details	T12.1: Encrypt details
View/ modify driver details (SHOULD)	Drivers should be able to modify and view their details	Drivers will be presented with their data, and will be able	T13.1: Add data modification functionality

		to modify it accordingly	
Testing	Tests to make sure product is usable	Test using BOOST methods and match with success criteria to see if software has no bugs	T14.1: Test

Risk Analysis

Probability: 1 – Unlikely, 5 – Highly likely
Impact: 1 – Minimum, 5 - Severe

Risk	Probability	Impact	Description	Mitigation Plan
Software loss	1	5	Losses of the software code or memory	Go to backed up files and try to evaluate and recover files
Conflicts between members	2	4	Conflicts can cause loss of morale and confidence in the team	Have a one on one session with the conflict victims and come to a solution
Member loss	1	5	Member losses can disrupt the cycle of the software plan	Divide the work between remaining members and urgently find a new member
Illness	4	4	In times of covid, illness is very likely these days	Divide the work between healthy members for the time being and wait for illness to subside
Lack of engagement	4	5	A member deciding not to engage as much in the work	Arrange a one-on-one meeting with member and re-engage them with tasks
Motivation	3	4	Lack of motivation to work on project	Hold in person meetings to bond as a team and build morale
Work balance	2	1	Lack of work balance between members	Hold a group meeting and rearrange work

				between members
Computational resources	1	5	Lacking computational resources to work on the project	Hold a one-on-one meeting and find a solution to provide resources by the help of the university
Lack of communication	3	4	Lack of communications may lead to misunderstanding and work being incomplete or wrong	Hold group meetings and resolve the issue

Use of monitoring tools

On unprecedented times as such, the importance of monitoring and collaborating tools have been more crucial to a group project than ever. Coordinating each task requirement and the effort put in by each teammate requires thorough planning and organisations. Thus, as a team, we all have agreed upon using these monitoring and collaborating tools to help us alleviate the stress of tracking and keeping record of what tasks are being done and how much effort every team member is putting in.

Microsoft Teams

Microsoft teams have unstoppably grown in popularity over the pandemic as a standard video collaboration tool to use instead of meetings and we have been using it as such as well. Every week, on a Wednesday, the whole group has a weekly meeting. We update on the tasks that need to be done and tasks that can be better improved and overall, it acts as a motivation booster for the group as the light-hearted conversation energizes and builds trust in our team.

GitHub

We are using GitHub as a version control for our software and is also a method of keeping our code on the cloud in case of any undesirable circumstances. With it, we are able to track updates to the code and share the files among other teammates so we can view, edit and add our own contribution to the project.

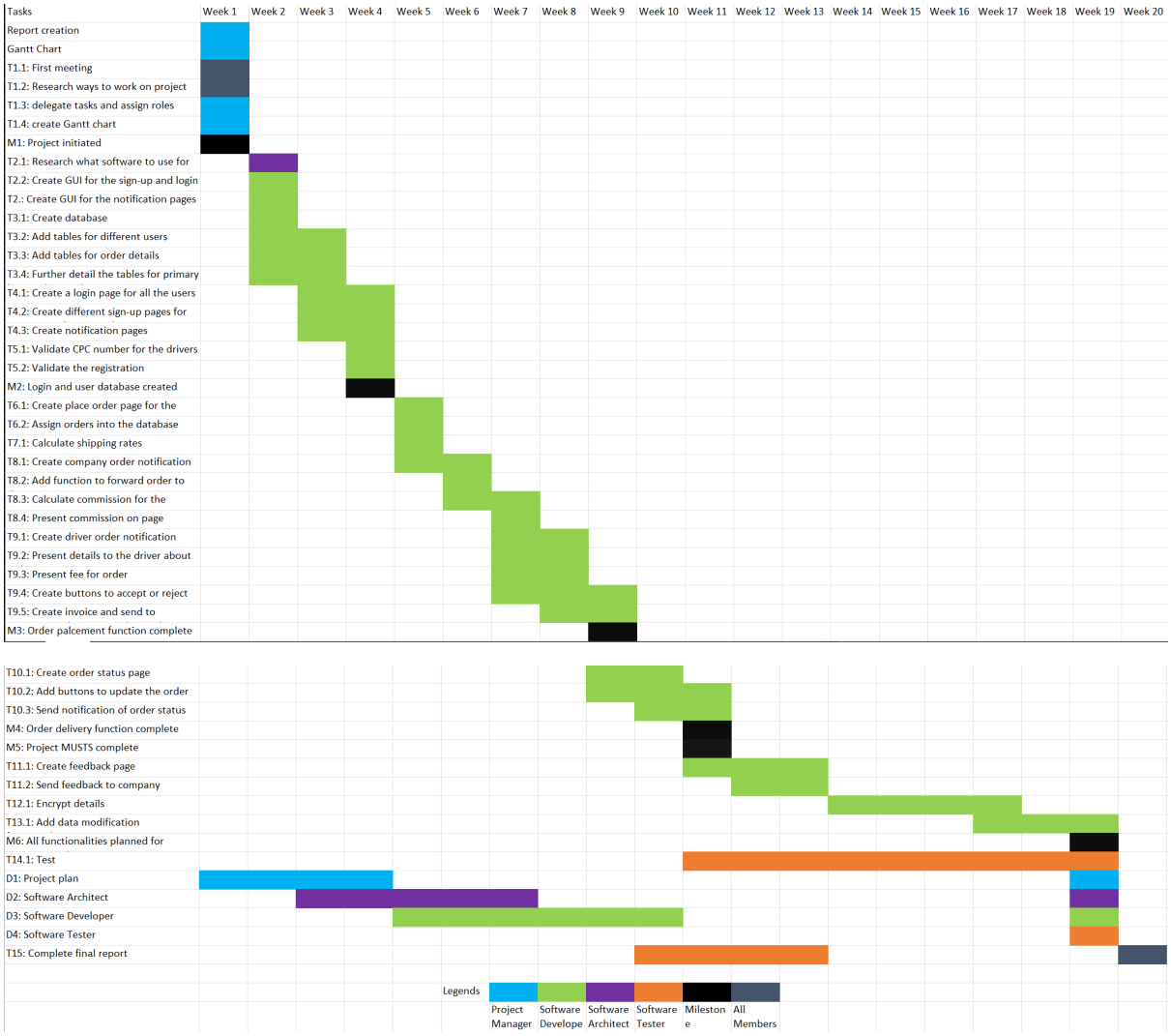
WhatsApp

WhatsApp has been our go to platform for communication from the beginning. Our WhatsApp group was created from day one and we all agreed mutually that this was the best way of communicating as we all had the platform on our phones and was easily accessible. In addition to that, it allows us to send documents and can be accessed on our computers, allowing the sharing and viewing of files much more convenient.

Slack

We have chosen to use Slack to monitor our workflow as it will allow us to assign and view tasks with ease. It also gives us the ability to give feedback to each other and submit approval requests and comment on other teammates workflow.

Gantt Chart



UML Diagrams

Use case diagram

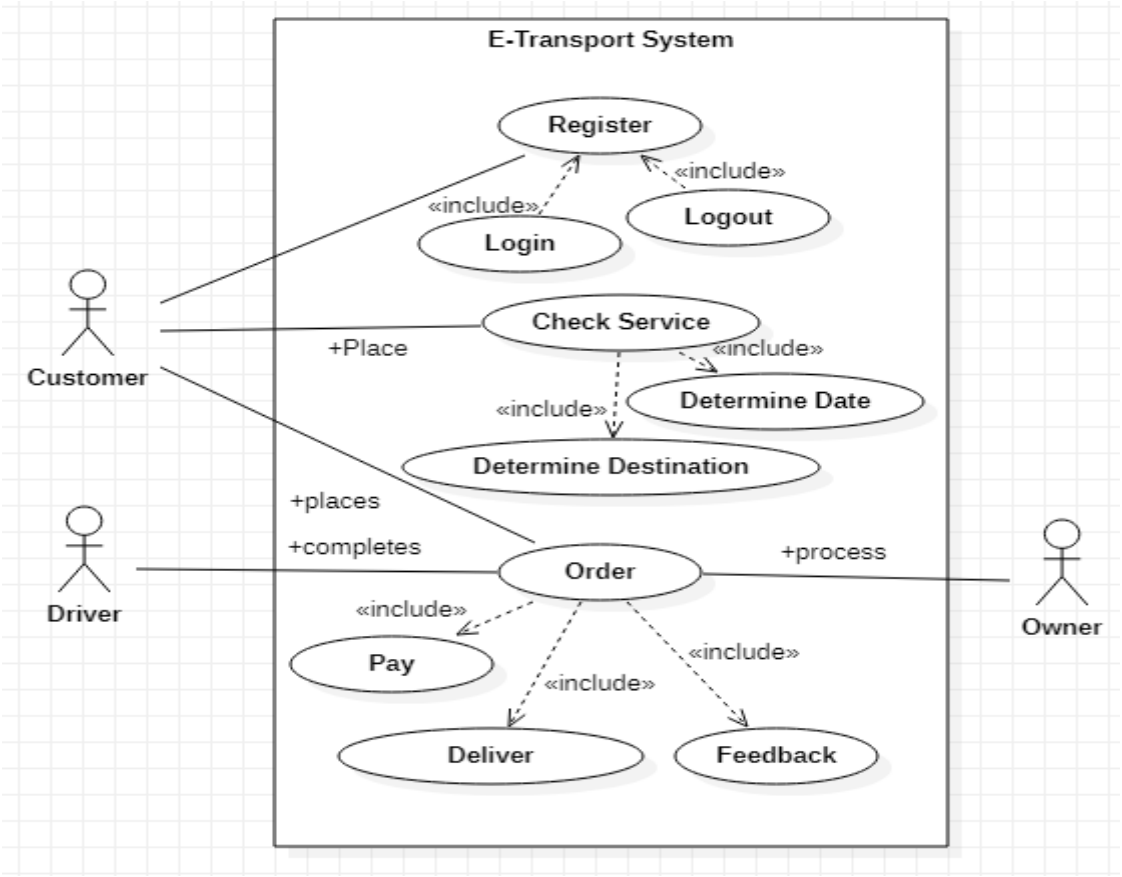


Fig 1: Use Case diagram of E-Transport System

Use case 1: Register

Actors: Customer
Description: Customer will register with the system and then will be able to login and logout.
Precondition: Web application must be up and running
Post condition: User will be able to login into the system
Basic course of action: 1.Enter details 2. Click on Register

Use case 2: Check Service

Actors: Customer

Description: Customer can check service availability
Precondition: Customer must login into the system
Post condition: Order page will be visible
Basic course of action: 1.Enter destination 2. Enter date of delivery 3. click ok

Use case 3: Place Order

Actors: Customer, Owner, Driver
Description: Customer can place order for cargo service
Precondition: Service for checked destination must be available
Post condition: Order will be placed, visible to driver and owner
Basic course of action: 1.Enter CPC, NI 2. Enter other details 3. Check shipping charges 4. Customer pays 5. Click on Place order button (visible to owner) 6. Owner click on send button (visible to driver)

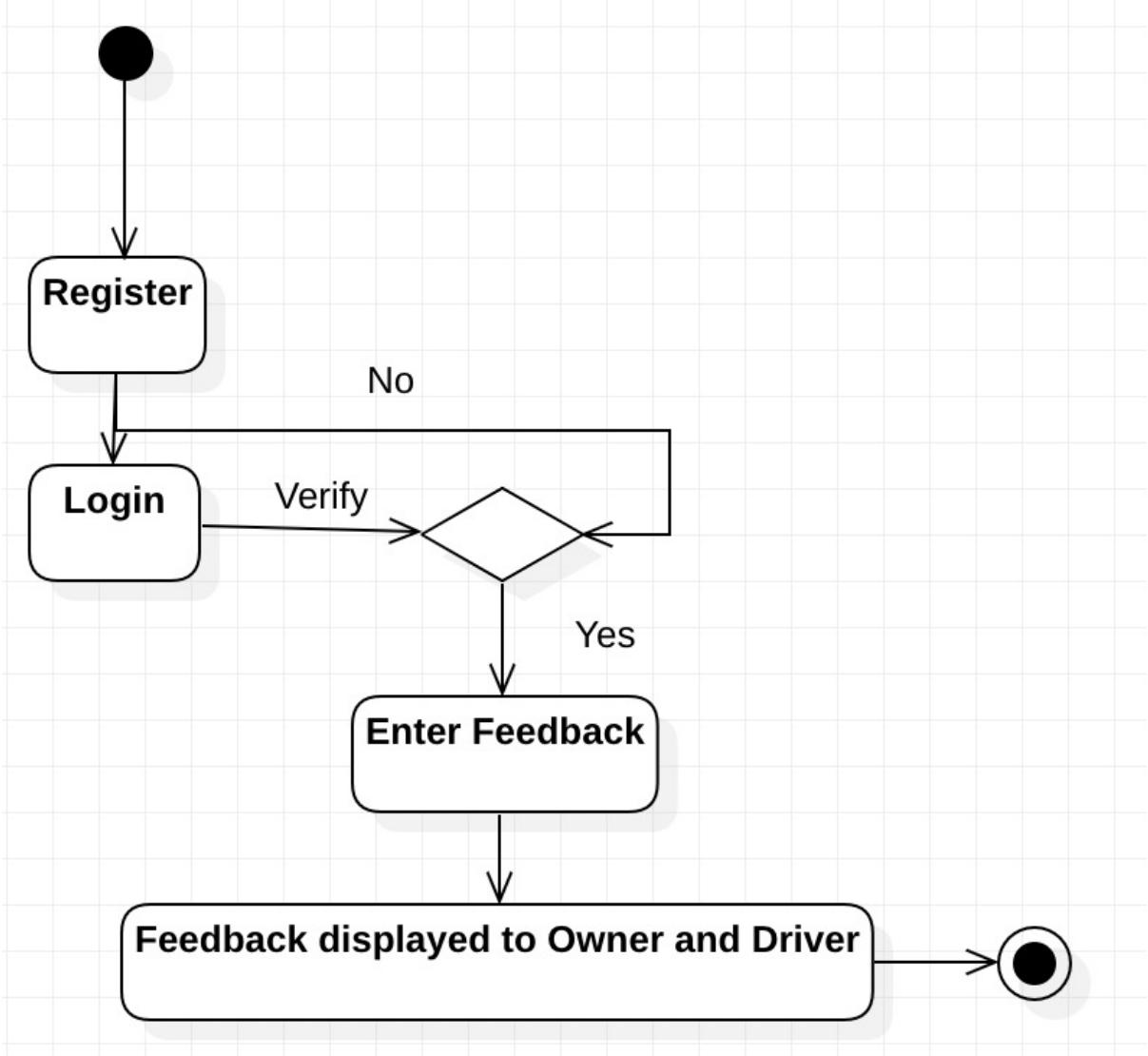
Use case 4: Feedback

Actors: Customer, Owner, Driver
Description: Customer can give feedback which would be visible to owner and driver
Precondition: Feedback can be given for only completed orders
Post condition: Feedback will be visible to driver and owner

Basic course of action:
1.Enter feedback
2. Enter Ratings
3. Click on send (visible to owner & driver)

Activity Diagram

Following diagram shows activity involved in displaying output. Customer must first successfully login to the system to get access of system features. After that customer is verified to submit feedback. Feedback will be visible to Owner and Driver



Following diagram shown in figure 2, shows activities involved in placing order by customer. Here customer will first register with the system. Only after successful registration, he will be able to login. The system will verify each customer if login is successful then check service page will be displayed. If service available, then place order page will be displayed. Once customer pays the order is successfully place and will be visible to the owner.

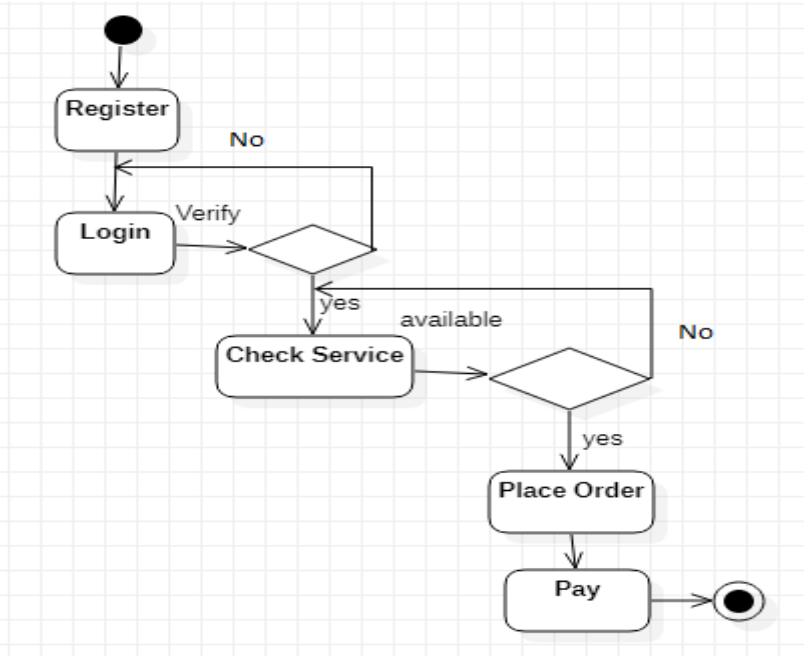


Fig 2: Activity diagram of E-Transport System for Placing Order

Following diagram shown in figure 3, shows activity involved in processing order by the owner. Owner must register and successful login to the system to get access to the system features. Owner can see all placed orders and clicks on send order button to send the order details to the driver. Once the driver delivers the order the order is processed and completed.

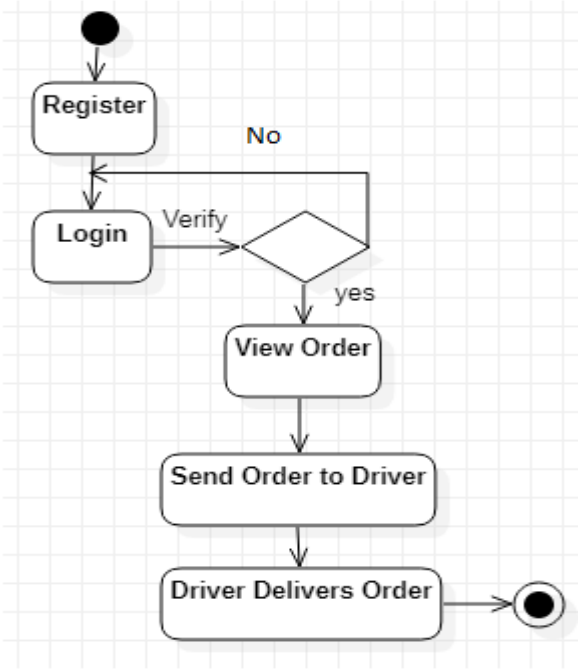


Fig 3: Activity diagram of E-Transport System for Processing Order

Sequence Diagram

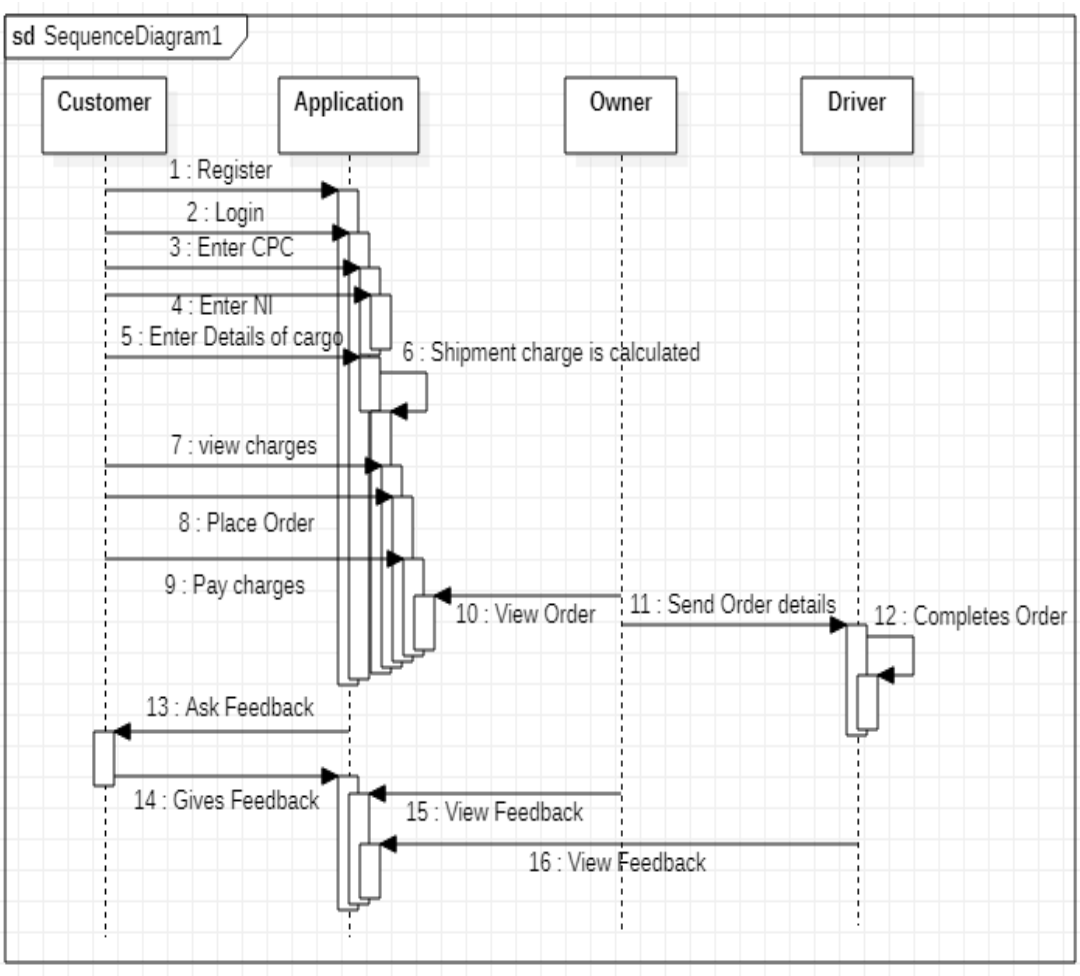


Fig 4: Sequence diagram of E-Transport System

Figure 4 shows the sequence diagram for the system explaining the sequence of action taken by customer, owner, driver in order to place and process an order using E Transport system. Customer will register and login with the system. Once logged in successfully, customer needs to enter his CPC and NI details, then he needs to enter the cargo details. The shipment charge is calculated based on cargo details, after which the customer can place the order and ay the charge. At owner side, all the placed orders will be visible which he can assign to the driver by sending order details to the driver. Driver will deliver the cargo and complete the order. Once an order is completed customer can give the feedback for the completed order. The given feedback can be seen by owner and driver.

Class diagram

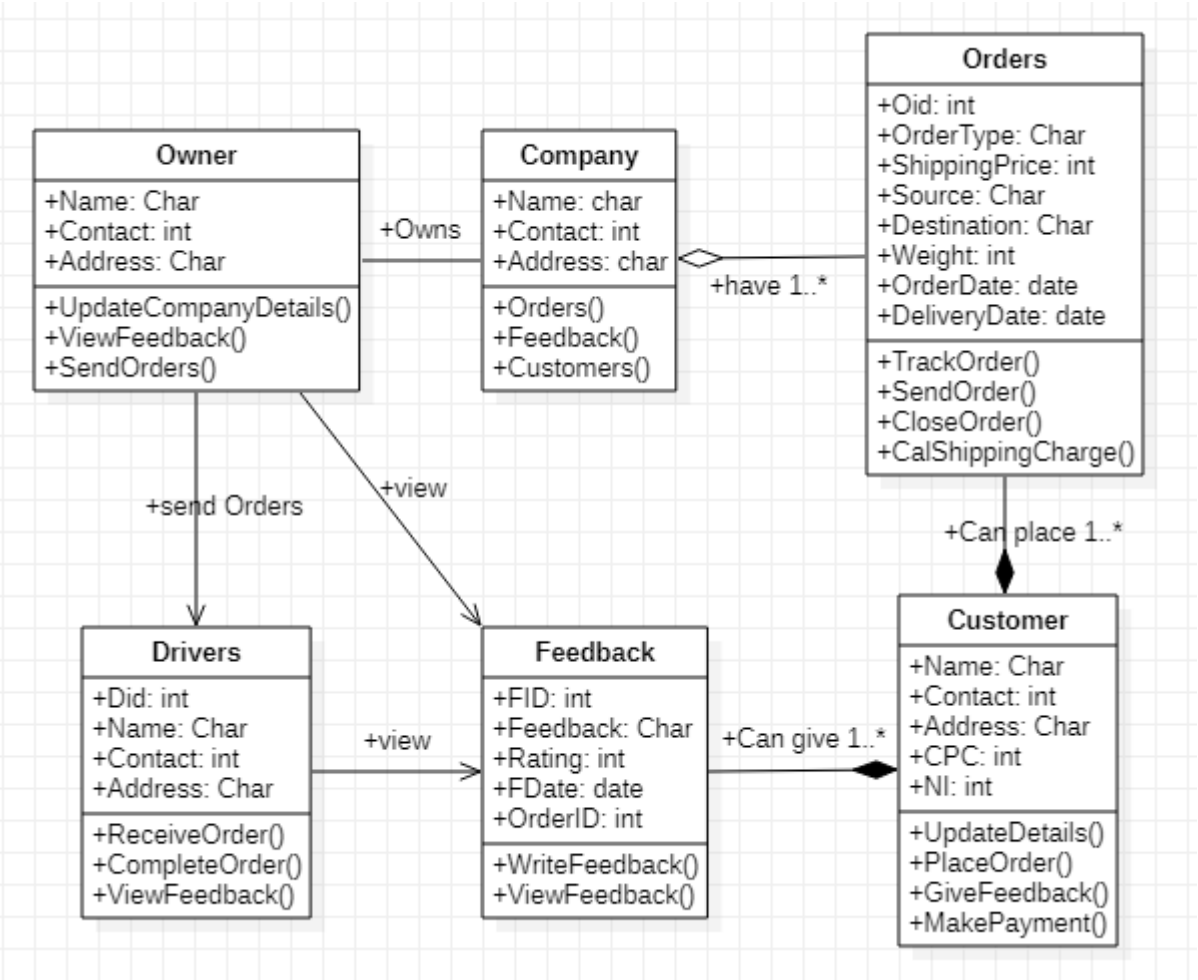


Fig 5: Class diagram of E-Transport system

Associations

Based on class diagram following associations are Identified:

1. **Normal Association:**
   Owner and Company has normally associated with each other.
2. **Directed Association:**
   Owner is directly associated to Driver for sending orders to be delivered.
   Owner is directly associated with feedback to view it.
   Driver is directly associated with feedback to view it.
3. **Aggregation:**
   Company can have one or many orders. Even if we delete order class still company class exists. So they are associated with aggregation association.
4. **Composition:**
   Customer can place one or many orders. If customer class is deleted, then there is no existence of orders class. So they are associated with composition association.
   Customer can give one or many feedbacks. If customer class is deleted, then there is no existence of feedback class. So they are associated with composition association.

1. **Operations and Attributes Identified:**
   1. **Class Company:**

Attributes: Name, contact and address
Operations: Maintain details of Orders, Customers, Feedback

2. **Class Owner:**
   Attributes: Name, contact and address
   Operations: Update company details, view feedbacks, send orders
3. **Class Orders:**
   Attributes: Order Id, Order Type, Shipping Price, Source, Destination, weight, Order date, Delivery Date
   Operations: Calculate shipping charge, update status of order
4. **Class Customers:**
   Attributes: Name, contact, address, CPC, NI
   Operations: Update their details, Place order, Give Feedback, Make payment
5. **Class Feedback:**
   Attributes: Feedback ID, Feedback, Rating, Date, Order ID
   Operations: Write & view Feedback
6. **Class Driver:**
   Attributes: DID, Name, contact and address
   Operations: Receive Order, Complete Order and view Feedback

Important operations are identified:

1. **User verification:** Only if user is registered user is allowed to login and check the services.
2. **Place Order:** User can place order to cargo items.
3. **Pay:** User is allowed to make the payment.
4. **Give Feedback:** Customers can give feedback
5. **View Feedback:** Owner and Driver can view Feedback

Cohesion and Coupling

There are 3 modules of the system user verification, orders and delivery. In the below diagram blue line represents cohesion and red line indicates coupling. As per the concept of cohesion and coupling, there should be more cohesion and less coupling. In our system too in each module each entity is linked internal to module more, and its link with other module is less.

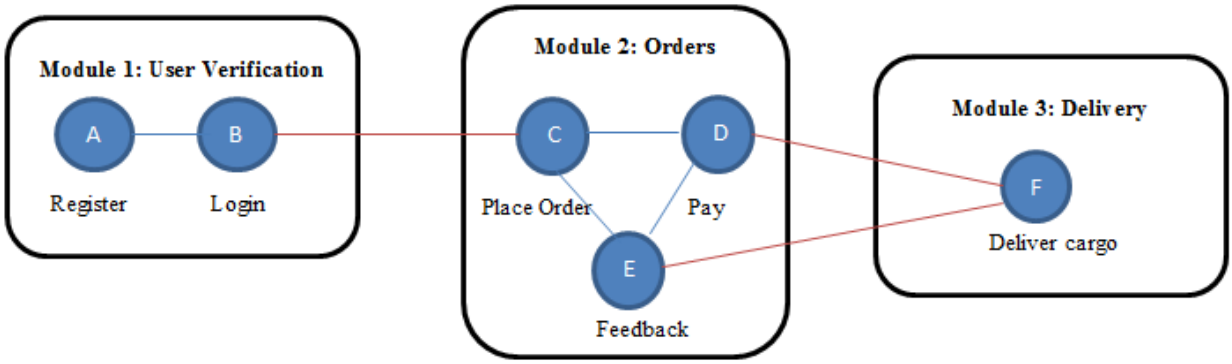


Fig 6: Cohesion and Coupling of E-Transport system

Component Diagram

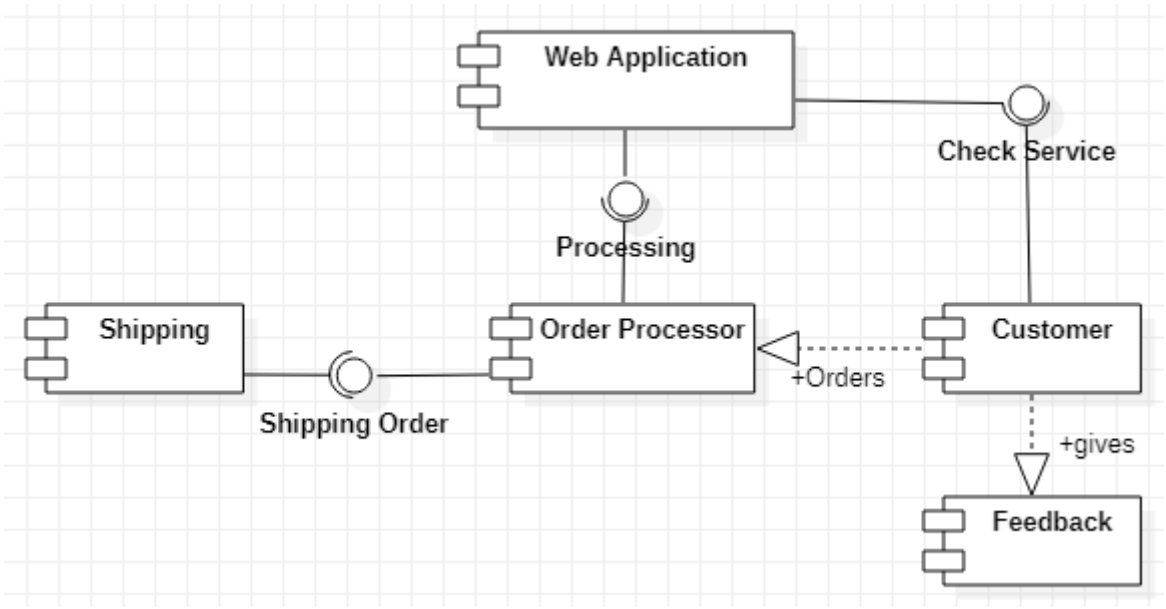


Fig 7: Component Diagram of E-Transport system

Figure 7 shows the component diagram having major component as application, customer, order process, shipping and feedback. All these components are interrelated to each other. Here the web application provides interface to the customer to check service. Also web application provides processing interface to order processor component. All customer using order processor component can place order and give feedback. The order processor component also provide interface for shipping component using which drivers can complete their orders.

Deployment Diagram

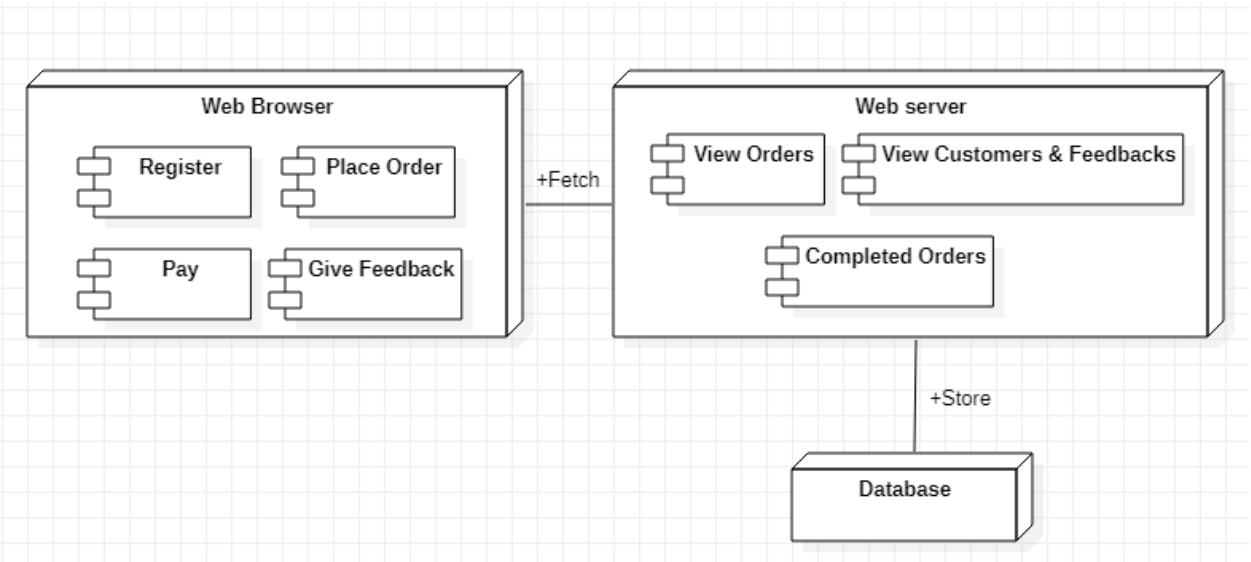


Fig 8: Deployment Diagram of E-Transport system

Deployment diagram shows the hardware and software required by the system. We have 3 important nodes in our system web browser, web server and database. Using web browser all the customer can access the application data which is processed over web server and stored in database.

FSM Diagram

Following diagram shows the different states in which the system moves. Here customer will first enter the register state. Only after successful registration, he will be able to go to login state. The system will verify each customer if login is successful then he will be proceeded to enter order details page else will be go back to login state. After entering cargo details, shipment charge is calculated and displayed if accepted then go to place order state else go to enter cargo details state. After place order customer is taken to make payment state. Once the cargo is delivered the state is changed to shipment delivered. After shipment is delivered, customer can give feedback.

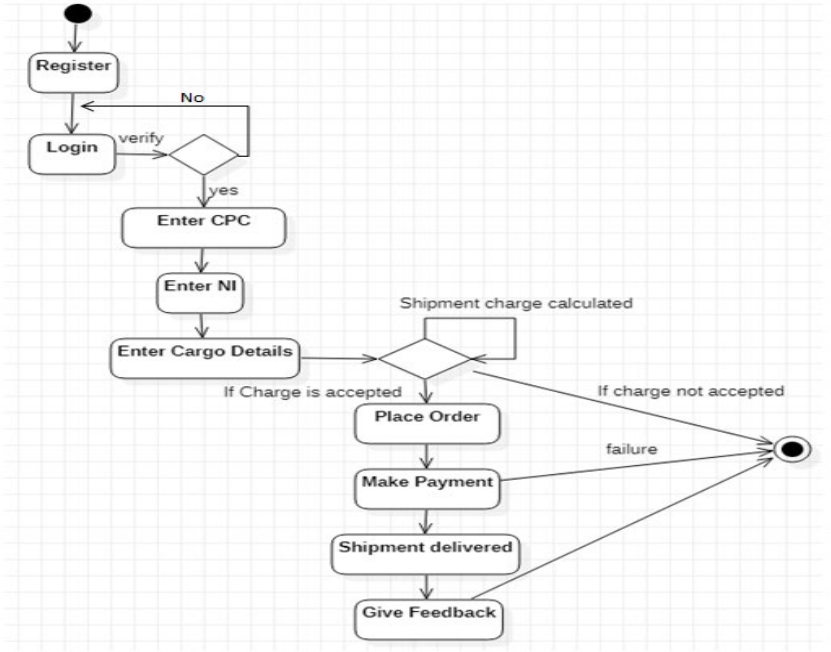


Fig 9: FSM Diagram of E-Transport system

Communication Diagram

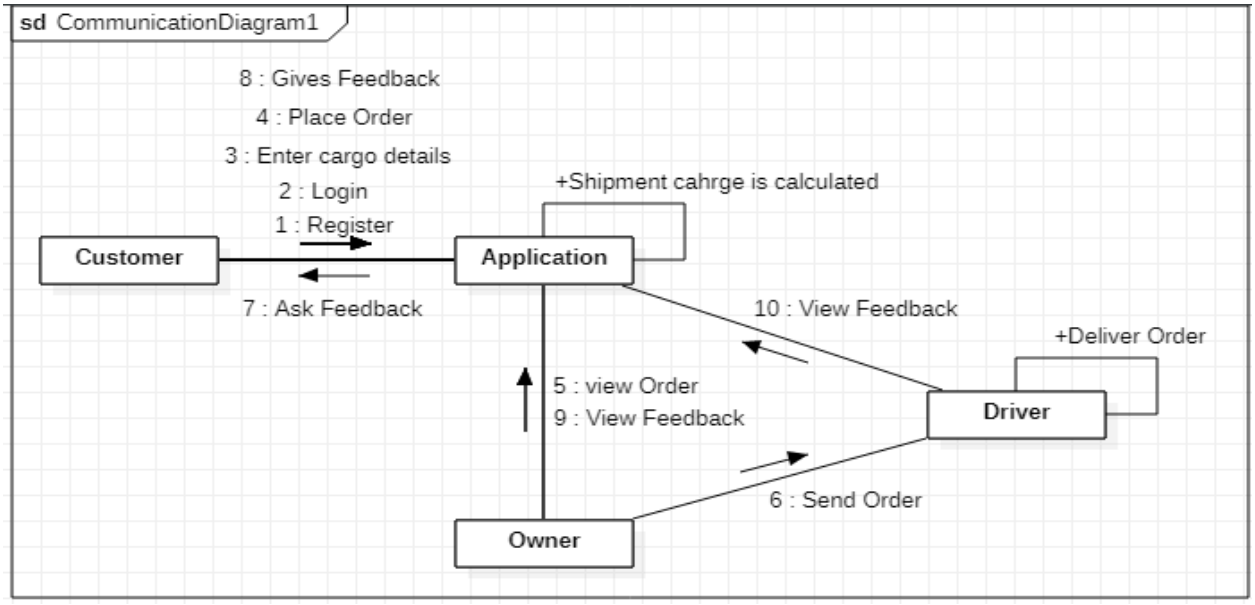


Fig 10: Communication Diagram of E-Transport system

Figure 10 shows the component diagram for the system explaining the interaction of customer, owner, driver in order to place and process an order using E Transport system application. Customer will register and login with the system. Once logged in successfully, customer needs to enter his CPC and NI details, then he needs to enter the cargo details. The shipment charge is calculated based on cargo details, after which the customer can place the order and ay the charge. At owner side, all the placed orders will be visible which he can assign to the driver by sending order details to the driver. Driver will deliver the cargo and complete the order. Once an order is completed customer can give the feedback for the completed order. The given feedback can be seen by owner and driver.

Model View Controller

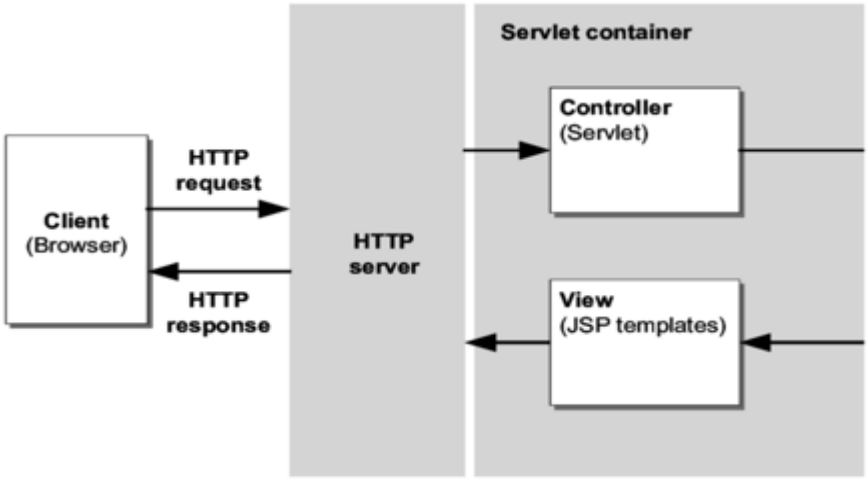


Fig 11: MVC of E-Transport system

Here our web application can use by customer, owner and driver as client using any browser. The HTTP request will be send to HTTP server. The HTTP server will send this request to servlet where the java servlet will be running as an individual process. The processes data will be made visible to user in form of JSP templates send to client side using HTTP response.

System Architecture

Structure, behavior, and other aspects of a system are all part of a system architecture. Formal representations and descriptions of systems, such as architectural descriptions, are used to facilitate reasoning about the structure and behavior of systems. Middleware, user interfaces, and databases are just a few examples of the many components that form a Web application's architecture.

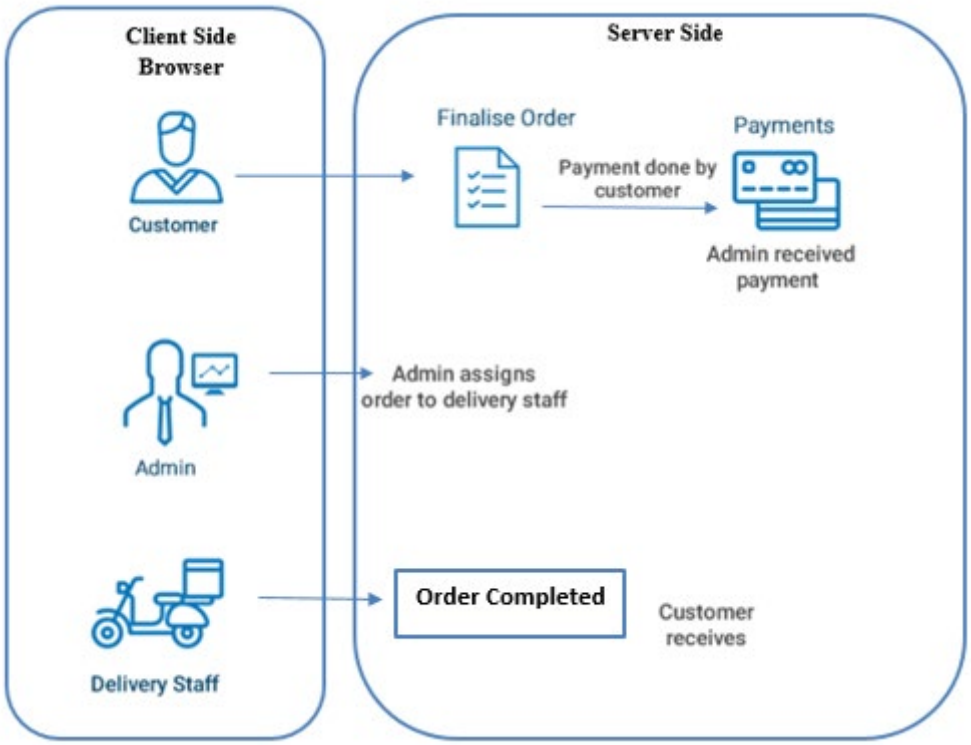
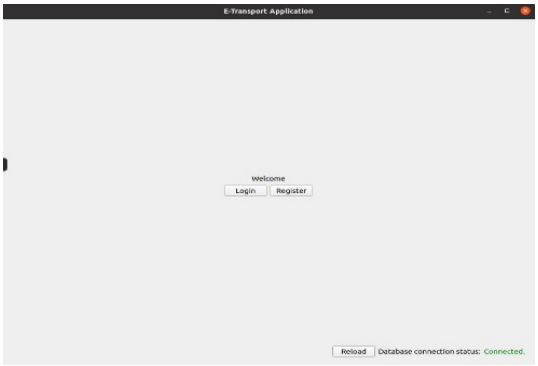
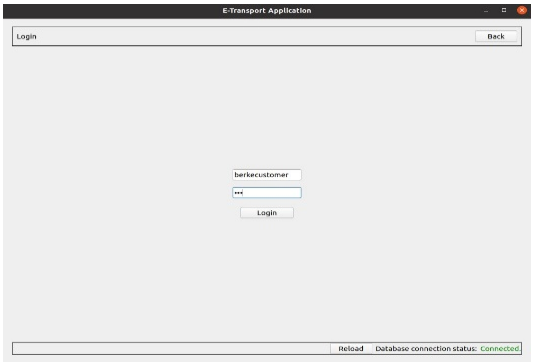


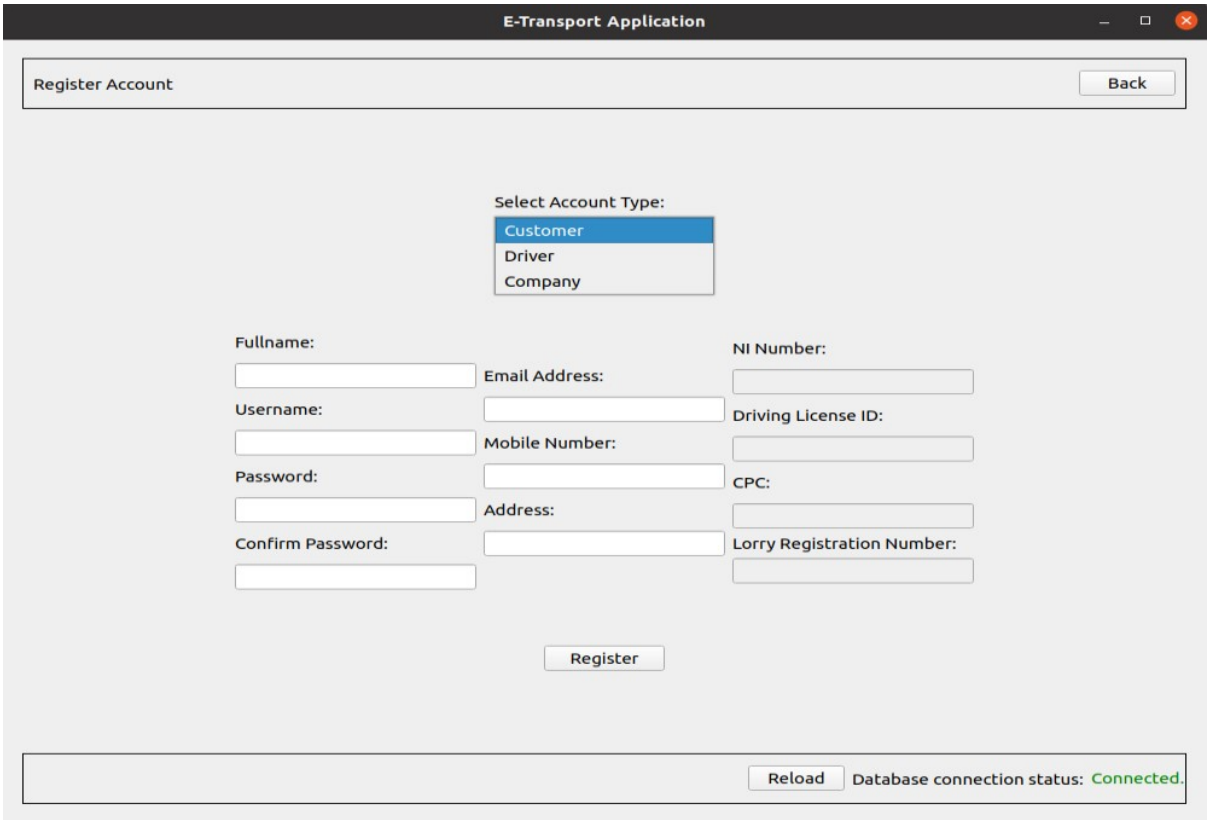
Fig 12: System Architecture of E-Transport system

User Interface and User Manual

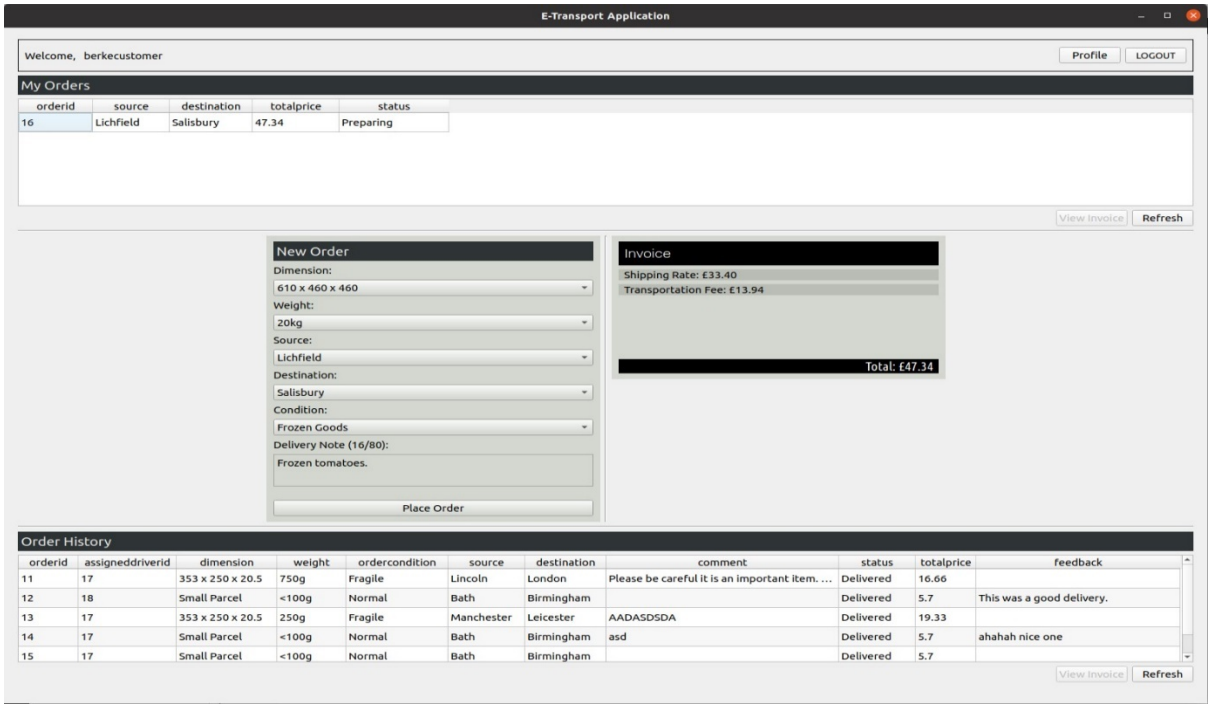




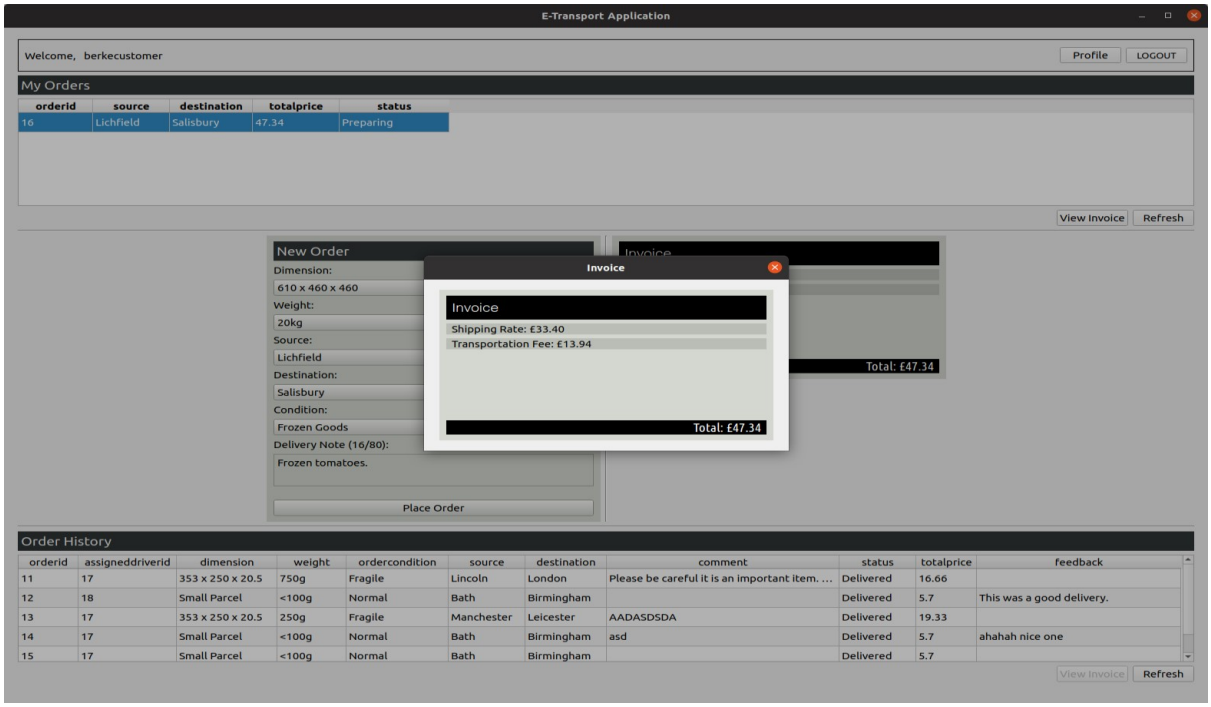
These are the login pages for the users where thay can either click to login or register their accounts into our database. Once you click login, it will take you to the login page where you will be prompted to add your personal username and password. The image below will show the register page



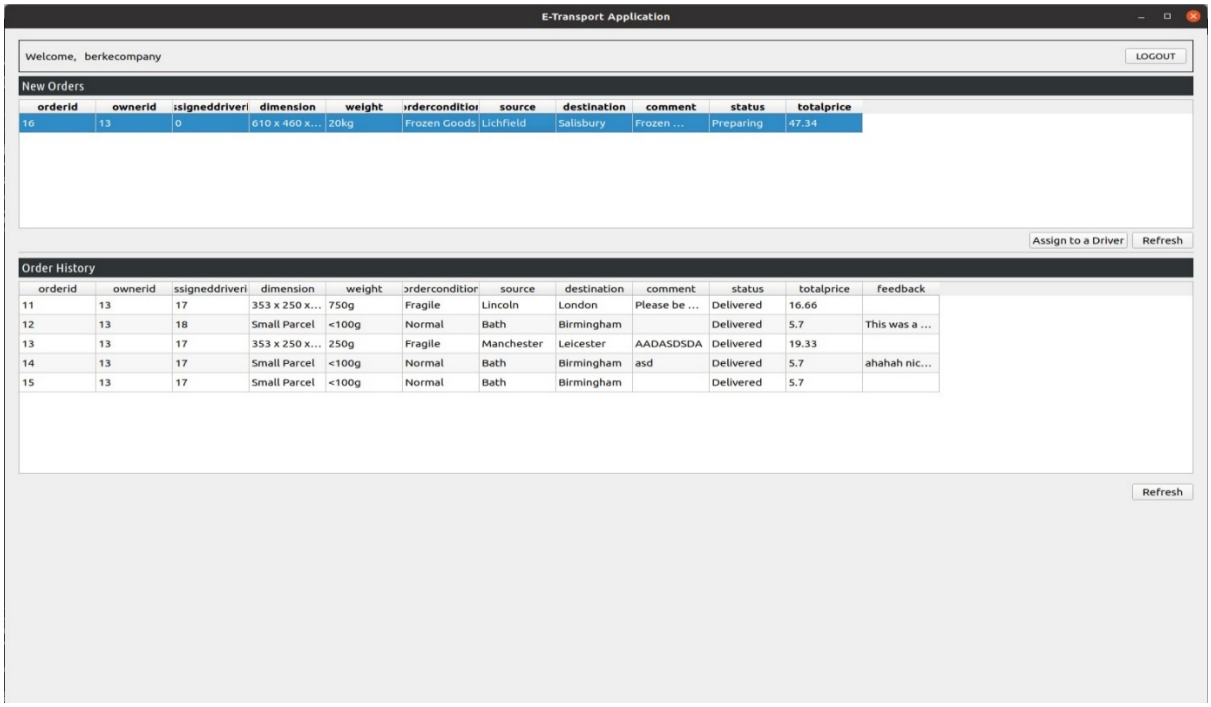
On the register page, there are a lot more options to be put in as the user is new and will require more information to be stores for late use when placing or creating orders. As you can see, at the top, you have the options to select if you are a customer, driver or a company. If you are a customer, you will be asked for your name, username, password, password confirmation, email address, mobile number and address. The same information is asked for the company users as well. If you select yourself as being a driver, in addition to the information asked for above, you will also have to provide your NI number, driving license ID, CPC number and lorry registration number.



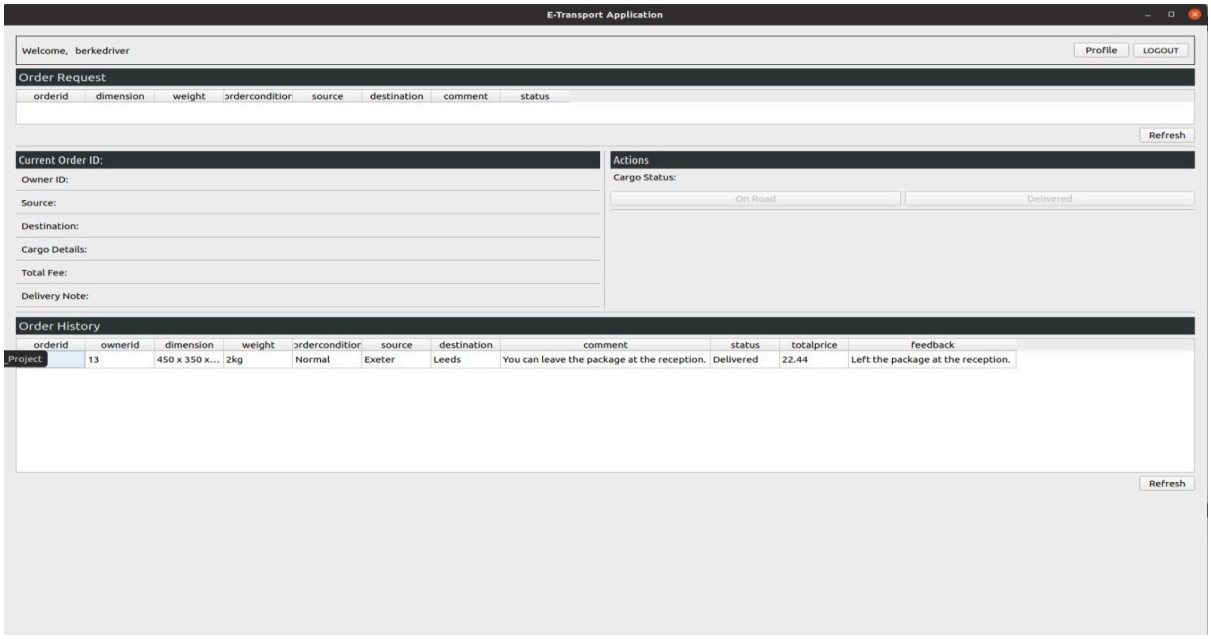
This is the page to create and place orders for the customers. The customer can select the dimension of the parcel, which will determine the cost of the shipping fees. The weight is selected after that, which also plays a big part in the shipping fees. After that, the source location and destination for the parcel is picked, followed by the condition of the parcel. As our e-transport marketplace is based in the UK, we have put in UK cities as the options for source and destination. There is also a space for any delivery notes. On the right, the shipping fees are shown to the customer. At the bottom of the screen, the order history can be seen where the details of the order and past feedback and comments are presented.



Once an order is accepted by the driver, the customer receives an invoice about their order regarding the shipping rate and transportation fee



This is the company page for company users. New orders arrive to the company page, detailing the attributes of each order. The orders can be then assigned to a driver, or the page can be refreshed to check if new orders have come through. At the bottom of the scree, all the orders from all customers can be seen



This is the page for the driver. Once the company assigns the driver an order, the page presents the driver with all the details of the order. The driver can then input information on the Actions box and can update the customer about the order status by either pressing “On Road” or “Delivered”.

Video Demo

YouTube Link: https://youtu.be/MAucFD9qnkY

We have uploaded the video to YouTube foe easier visibility and convention. In the demo we have represented how our software works as an application, showcasing all the usable features it has and the different ways it can be accessed by different users.

C++ Libraries Used

- #include <QMainWindow>

  QMainWindow is a type of convenience type class that can be used as the mainwindow of an application. It comes in with multiple built in features like status bar, toolbars and a menu bar.

- #include <QtSql>

  This library provides support for SQL databases. It contains classes for supporting the databases.

- #include <QSqlDatabase>

  This library represents a connection to a database. It provides an interface for accessing a database through a connection.

- #include <QMessageBox>

  This helps provide a modal dialog which is useful in notifying the user, asking user questions and receiving the answers.

- #include <QtCore>

  This is what all the QT modules rely on. It contains the core classes, including the event loop and Qt's signal and slot mechanism.

- #include <fstream>

  This data type has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

- #include <QDebug>

  The QDebug class provides an output stream for debugging information.

Search and Sort Algorithms

Not a lot of algorithms were used in out project as we used SQL for our database. We have chosen SQL for its vast features and abilities to work out the algorithms itself. Most of the searching and sorting were done using SQL queries.

Tests

We have created and implemented a vigorous testing system for our program. The tests we have chosen will contribute to us realizing if the program has been a success or it needs to be worked on more. The tests are to be held on all the features we have chosen to implement, including the musts, coulds and shoulds. As most of our testing must be done on our own and our system does not have any automatic system on board as we have to place orders, we are not going to run any automatic tests. The tests will be run as many times as required till the success criteria is met.

ID	1	Description:	This test is to check that the Cargo Owner can sign up and create a new account
Test type	Acceptance	Success criteria:	The Cargo Owner can add all their necessary information without any errors
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program Enter the required details click register		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS
ID	2	Description:	This test is to check that the Cargo Owner can view their profiles and edit their details
Test type	Functional	Success criteria:	The cargo Owner can go on their profile page and change their details and the program is able to update and save

			the latest information
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program Login navigate to profile page edit details and save		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	3	Description:	This test is to ensure that the Cargo Owners' details are encrypted in the database
Test type	Functional	Success criteria:	Check the database and ensure that the password set cannot be seen
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program make sure the customers details can't be seen in the database		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		

Test Date	24/04/22	Result	PASS

ID	4	Description:	This test is to check that the Cargo Owner can sign in and out
Test type	Acceptance	Success criteria:	The Cargo Owner can log into their account and log back out again.
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	start the program Login Then log back out		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	5	Description:	This test is to check that the Cargo Owner can place cargo orders
Test type	Acceptance	Success criteria:	Cargo Owners can place orders which is then received by the Transportation Company
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	start the program login as a customer Fill out the details for and order and click place order		

Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	6	Description:	This test is to check that the Cargo Owner can view order status
Test type	Acceptance	Success criteria:	The Cargo Owner can go on their profile and view their order status
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program Login as a customer View the status of their orders on their account		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	7	Description:	This test is to check that the Cargo Owner can view their invoice

Test type	Functional	Success criteria:	After making an order, the Cargo Owner can view their invoice
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	start the program login as a customer View their invoice on their account		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	8	Description:	This test is to check that the Cargo Owner can receive delivery notifications
Test type	Functional	Success criteria:	The Cargo Owner can go on their profile page and see delivery notifications for their order
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program Login as a customer view the notifcation recieved on the page		
Failure correction procedure	Debug the program and fix the bug		

Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	9	Description:	This test is to check that the Cargo Owner can view their history and feedback
Test type	Functional	Success criteria:	The cargo owner can go on their profile and view their history and feedback
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program view the history and feedback on the page		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	10	Description:	This test is to check that the Cargo Owner can add comments
Test type	Acceptance	Success criteria:	The cargo Owner can add comments
Number of attempts:	1	Comments:	

List of equipment / requirements			
Setup instructions	Start the program login as a customer start placing an order include a comment before placing order Place order		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	11	Description:	This test is to check that the transportation company can sign up and create an account
Test type	Acceptance	Success criteria:	The Transportation Company can sign in and create an account.
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	start the program enter the required details for the company Create the account		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	12	Description:	Transportation Company can forward orders to the Driver
Test type	Functional	Success criteria:	The Driver can see in their profile orders forwarded from the Transportation Company
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program login as the company assign an order to the driver		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	13	Description:	This test is to check that the Transportation Company can receive orders from customers
Test type	Functional	Success criteria:	The Transportation Company can go into their account and see orders customers have made
Number of attempts:	1	Comments:	

List of equipment / requirements			
Setup instructions	Start the program login as a customer place an order log out login as the company view the order placed by the customer		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	14	Description:	This test is to check that the Transportation Company can issue invoices to customers
Test type	Functional	Success criteria:	The customer can go on their account and see invoices sent by the Transportation Company
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program login as customer Place an order View the invoice issued by the company		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		

Test Date	24/04/22	Result	PASS

ID	15	Description:	This test is to check that the Transportation Company can calculate commissions for orders
Test type	Functional	Success criteria:	The program can calculate commissions based on the order and display the correct amount.
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program Login as a customer place an order view the commission calculated by the company		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	16	Description:	This test is to check that the Transportation Company can view the order history
Test type	Functional	Success criteria:	The transportation

			Company can go on their account and view the order history
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	start the program Login as the company view the order history on the page		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	17	Description:	This test is to check that Drivers can sign up and create a new account
Test type	Acceptance	Success criteria:	The Driver can sign up and create their account including all the necessary details
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program Enter all the details to create an account for a driver then create the driver account		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	19	Description:	This test is to check that the Drivers details are encrypted when stored in the database
Test type	Functional	Success criteria:	Check the database and ensure none of the drivers personal details can be seen.
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program check the database and make sure the drivers personal details cant be seen		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	20	Description:	This test is to check that the Driver can sign in and log out
Test type	Functional	Success criteria:	The Driver can sign in and out of their account
Number of attempts:	1	Comments:	

List of equipment / requirements			
Setup instructions	Start the program login as the driver then log back out		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	21	Description:	This test is to check that the Driver can receive order notifications
Test type	Functional	Success criteria:	The Driver can receive order notifications from the Transportation Company
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program login as the driver View the order notification on the page		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	22	Description:	This test is to check that the Driver can receive notifications about the order status
Test type	Functional	Success criteria:	The driver can go on their account and see notifications about their order status
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program login as the driver view the notifications of orders on the drive page		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	23	Description:	This test is to check that the Drivers can view the shipment history
Test type	Functional	Success criteria:	The driver can go on their account and view the shipment history
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program login as the driver view the shipment history on the page		

Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

ID	24	Description:	This test is to check that the Driver can send messages to the Cargo Owner
Test type	Acceptance	Success criteria:	The Driver can send messages that can be viewed on the Cargo Owners account.
Number of attempts:	1	Comments:	
List of equipment / requirements			
Setup instructions	Start the program login as the driver On the page		
Failure correction procedure	Debug the program and fix the bug		
Engineer(s)/Technician(s)	Berke Kanlikilic and Adarsh Kumar		
Individual results:	Attempt 1: SUCCESS		
Test Date	24/04/22	Result	PASS

Discussion on Result Analysis

As the deadline comes to a close, we have been able to finish this project from start till the end, being able to facilitate all the intended features, create all necessary UML diagrams and descriptions, run the system through rigorous tests and collaborate to create a final report on it.

As seen above, our software has been able to pass all the tests implemented on it, which we are considering a success. The tests we have run on the software are based on what a client, driver or

the company would tend to use in the program. The tests ensure that all the aspects of this software are run without any major bugs or does not have any running or compilation error.

The software is very reliable as it works flawlessly. It can be maintained well as users increase. To increase the scale of the software, more efficient searching and sorting algorithms may have to be implemented and the design may need to be updated to a commercial standard as ours is very bare bones.

Conclusions and Future Work

To conclude this project, we can say that the program has successfully been deployed as per our research and requirements. We believe we had met all the requirements from the client and have gained a lot of knowledge regarding carrying out a project with a team.

This project has further developed our programming skills and has emphasised on our design and analysis skills which have been taught in year 1, further building the strength of the assets we already have. We have also learned to collaborate as a team and were able to efficiently work in an agile way and maintain professionalism between us. We believe that all these skills will have further developed our professional and personal development and increase the value of our degree.

Looking back at our testing reports, we believe they were extensive and will include most of the use cases in the real world and avoid glitches. The program had run very efficiently, which has been seen in the video that it does not take long to load pages, orders, order histories or even sending orders through users such as drivers or companies. All passwords are encrypted, along with the user information, which keeps the software secure. The only standout concern may be the portability of the software as it can only run on LINUX, which may be popular for itself but is not as popular among regular audience as not everyone tends to use LINUX OS.

Requirements Implemented

Requirements:	**Status:**
Cargo Owners:	
Sign up and create new account	DONE
View profile, edit details	DONE
Encrypt details in database	DONE
Be able to sign in/ log out	DONE
Place cargo orders	DONE
View order status	DONE
View invoice	DONE
View delivery notification	DONE
View history/ feedback	DONE
Add comments	DONE
Transportation Company:	DONE
Sign up and create account	DONE
Receive order from customers	DONE
Froward order to driver	DONE
Issue invoice to customer	DONE
Calculate commission for order	DONE

View order history	DONE
Drivers:	DONE
Sign up and create new account	DONE
Include lorry number, CPC number, NI number, etc	DONE
Encrypt details	DONE
Sign in/ log out	DONE
Receive order notifications	DONE
Notify about order status	DONE
View shipment history	DONE
Send message to cargo owner	DONE

Summary of Group Experience

The group experience of working for this coursework was very pleasant for all of us. Throughout the whole duration of the coursework, from the first meeting till the point of writing this section in the report, the experience was filled with respect, comradery, productivity, and a splendid show of teamwork.

The standards had already been set on the first time we all met as it was exciting experience after the pandemic to meet everyone. The roles were assigned quite fast, and the environment was energetic when discussing about how to proceed with the project. From start to finish, everyone pulled their work and put in every bit of effort they could do make this project the best it can be.

Appendix

User Manual

The user manual has been explained along with the user interface as both sections could be explained with the same pictures

Here is an extended guide to the user manual

1. When a user logs into the system they are prompted with the option to either login or register.
2. If you are a new customer, select register, if you are already registered, please login to your account to continue.
3. All users must have an account to continue further.
4. When a user tries to register, they have to confirm if they are the Driver, Customer or Company.
5. The information fields will change based on what they select as some extra information is required by drivers.
6. It is essential for the user to cover all the fields to make an account.
7. All users are able to update their profile information

Customer

1. After the customer has logged in they will be able to see their default UI screen.
2. They are able to place orders, give feedback after an order has been completed and they are constantly notified with the status of their order.

3. Log out button is available at any time.
4. Shipping rate is calculated when the destination is selected and the customer is required to input information about the cargo, for example, dimensions etc.
5. Invoice is available after a delivery has been done.
6. Notifications are received at every moment, for example when a company confirms an order, a cargo is on the road or delivered.

Company

1. Company receives an order when it is placed by the customer.
2. Log out button is available.
3. Companies can accept the order by forwarding it to the drivers, the order stays up until one of the drivers has accepted the order.
4. Company's commission for each order is calculated.
5. They can get a little feedback from customers.
6. Notifications on driver and cargo status are provided.

Driver

1. Drivers receive notification once the company has forwarded the order.
2. The driver can accept or deny the cargo order. If denied, the order will return back to the company
3. When an order is confirmed by the driver, they are given the option to constantly update their status, so the customer and company can stay notified and up to date with the status of the order.
4. Order details are viewable by them. For example, fees etc.
5. Can select the status, loading, on road or delivered.
6. Logout option is always available.

Code Contributors Guide

1. Before putting in a new repository or working on code, notify the group (especially developers) about your proposal
2. To test code, ask the developers to notify of features implicated
3. Document anything new added to the code to maintain version control
4. Submit code to developers to have the code checked and approved
5. Comment as much as you can to make the code understandable
6. Add code only when approved by developers
7. Notify manager of addition to update the final report

Coding Standards Guide

1. Variables must be understandable and meaningful so a new contributor can use it.
2. Local variables should have camel case lettering
3. Use indentation properly, putting spaces after commas or symbols where appropriate for better understanding
4. All blocks of programming should be separated and clear
5. Braces should start in a new line
6. Avoid coding styles that are too illegible for a new user
7. Avoid using an identifier (eg: name for a variable) for different reasons

8. Code should be documented clearly and frequently
9. Function length should not be larger than necessary to avoid misunderstanding
10. Structure the code well, avoiding GOTO statements that make the code harder to debug

Reference Documentation for Functions and Classes

- ClearAllInputFields
  This function clears every text that has been written in any QLineEdit widget in the project.
- ConnectToDatabase
  This function uses the database information to connect to the database. Then checks if the database is open and successfully connected. If so, then it turns the connection label on the bottom right to Connected. If it fails to connect, then it changes it to Connection Failed.
- encryptString
  This function takes a QString as an input, and then encrypts it with a special key that is embedded in the code. After that, it outputs the new encrypted string. It basically adds the value of the key to every char in the string, so this changes the whole string into an unreadable string.
- decryptString
  This function takes a QString as an input, and then it subtracts the special from every character in the string. After that, it outputs the new decrypted string.
- showMyOrders
  This function populates the active orders table for the current logged in customer account by using a query with logged account id and the status of the order.
- showAllOrders
  This function populates all orders table for the current logged in company account by using a query.
- showCompanyOrderHistory
  This function populates order history table for the current logged in company account by using a query with only showing all of the delivered orders.
- ShowCustomerOrderHistory
  This function shows all the delivered orders of that logged in customer account.
- showDriverOrderHistory
  This function shows all the orders that were delivered by the logged in driver account.
- showDriverOrderRequests
  This function shows the current assigned order for the current driver account with a query using the logged in driver account id and the order status being equal to “Confirming”. Because firstly, the driver needs to accept or reject the order.
- calculateShippingRate
  This function calculates the shipping rate depending on which dimension and weight the customer has chosen. It uses switch cases to quickly set the shipping rate.
- calculateCommissionFee
  This function calculates the commission fee by taking 15 percent of the addition of shipping rate and transport fee.
- calculateTotalPrice
  This function just adds together the shipping rate, transport fee and commission fee and then writes it into the invoice.
- saveOrderToDatabase

This function takes every data from the fields that the customer filled in and then adds them to the database with a query.

- fillCurrentData
  This function fills the information about the current assigned and confirmed order for the logged in driver account.
- setOrderID
  This function is a setter for the current selected order id to view the invoice.
- viewInvoice
  This function shows the invoice by converting blob data to pixmap and shows it to the user.
- cetCurrentOrderID
  This function is a setter for the current selected order id to write feedback to.
- on_driverSubmitButton_clicked
  This function uses a query to update the feedback column of the current selected order.
- setAccountInformation
  This function takes in the special key for encryption and decryption and takes the current logged in account id, then populates the text fields with the users account information.
- on_dProfileSaveChanges_clicked
  This function uses a query to update the current logged in accounts information.

References

- GeeksforGeeks. 2022. *Coding Standards and Guidelines - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/coding-standards-and-guidelines/> [Accessed 24 April 2022].
- En.cppreference.com. 2022. *A list of open source C++ libraries - cppreference.com*. [online] Available at: <https://en.cppreference.com/w/cpp/links/libs> [Accessed 24 April 2022].
- 2022. *SDI Example report*. SDI NTU NOW Page. [Accessed 24 April 2022].
- Ibisworld.com. 2022. *IBISWorld - Industry Market Research, Reports, and Statistics*. [online] Available at: <https://www.ibisworld.com/industry-statistics/number-of-businesses/couriers-local-delivery-services-united-states/> [Accessed 24 April 2022].
- Statista. 2022. *UK parcel shipping: annual volume 2013-2021 | Statista*. [online] Available at: <https://www.statista.com/statistics/1010391/parcel-shipping-annual-volume-uk/> [Accessed 24 April 2022].
- 2022. [online] Available at: <https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/setting-guidelines-for-repository-contributors> [Accessed 24 April 2022].