University Of Engineering and Technology,
Lahore
Faculty of Electrical Engineering
Department of Computer Science

CS-261L Term Project

Data Structures and Algorithms

Distribution Management System

Group 01

Shahzaib Irfan *2021-CS-7*

Afraz Butt *2021-CS-12*

Muhammad Hamza *2021-CS-41*

Supervisor: Mr. Samyan Qayyum Wahla

Submission date: 22/12/2022

Table of Contents

List of Figures

List of Tables

Dedication

Dedicated to our parents, teachers and friends, whose constant support has enabled us to advance to this stage, despite all hurdles faced.

Dedicated to certain people in our lives, who made us work even more vigilantly and effeciently, reminding us of what mattered the most.

Acknowledgment

First of all we thank **Allah Almighty** for blessing us and supporting us throughout this endeavour.
We take this opportunity to express our profound and sincere gratitude to our supervisor **Mr. SAMYAN QAYYUM WAHLA** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project.

Lastly, we thank all our friends without whose support this project would not be possible.

Abstract

Often times in software development, built in data structures are sufficient enough to do the job. But sometimes, we do need to develop our own customisable data structures. The reason for this is that in the context of such applications, a custom data structure is fast, efficient and simple to implement. In the working for a full fledged distribution management system,the above concept takes a unique paradigm. This involves a quantitative methodology in the case of custom data structure implementations.The rider must be delivered orders in a *queue* for a FIRST IN - FIRST OUT order. Furthermore, fuel logs for riders (Managed by Administrator side) must be efficient as to their costs. This problem is taken care by a *Binary Search Tree*.Finally, the customer must be able to view Orders and its history.This report highlights our work in implementing such custom data structures, and implementing our methods on them. Additionally, various path-finding problems such as route management by geocoding were taken care of using Google API. Additionally, this report highlights the usage of these data structures.

How to Read This Document

This document is a comprehensive report on our final semester project. It contains details of all our work in the project, the challenges faced by us during development, and finally the rendering of the app.

- Initially the background of the project is given.
- Then details of use cases, technology stacks and interfaces are given. They are the same we made in our proposal.
- Afterwords comes the
    1. Data Structures Analysis against each entity
    2. Storage and retrieval Methods
    3. Exceptions in the project
    4. Challenges
- User Interfaces have been appended separately at the end.

Chapter 1

Project Background

1.1 Project Statement

Distribution Management, also known as Supply Chain Management System, are essential for the working of any business, let it be small or large. The difference principally comes in the scale, but micro details remain same. Order management is normally tedious, when done *physically*. What if there was a software / project that could automate all that?
This project is intended to assist all persons involved in a supply chain operation (Customers, Administrators and Delivery Riders / Truckers) in their operation.

1.2 Description

Managing the flow of goods from a supplier or manufacturer to a point of sale is referred to as distribution management. It is a general phrase that covers a wide range of operations and procedures, including supply chain management, inventory management, warehousing, and packaging.
For distributors and wholesalers, distribution management is an essential component of the business cycle. The ability of an organization to quickly turn over its products affects its profit margins. The future of the company will be brighter as they sell more and make more money. Businesses need a good distribution management system to maintain customer satisfaction and competitiveness.
Distribution Management System (DMS) is also important in recruiting and satisfying clients. It ensures the organization's long-term viability and competitive advantage. It aids in the management of profitable enterprises. Amazon is an excellent example of how to use DMS successfully and efficiently.
The most recent distribution management systems collect and disseminate pertinent information to assess industry potential for growth and competitiveness. Distribution is

divided into two categories:

1. Sales Distribution
2. Logistics

Distribution Management has a direct impact on the organization's earnings. To comprehend the significance of DMS, consider the obstacles that sales channels encounter. The organization develops numerous sales and marketing tactics to reach a larger audience. The channels are a way for these things to be sold, but many distributors are tiny and inefficient. They lack both finance and technology. To enter rural areas, more levels of the distribution network must be added, incurring additional expenditures. There is no real-time data on orders, inventories, or claims, and returns result in understocking or overstocking.
This system will also assist businesses to maximise labour and space utilisation, as well as equipment investments, by coordinating and optimising resource usage and material movements. It is intended to support the needs of a worldwide supply chain, including distribution, manufacturing, asset-intensive, and service businesses. Connected consumers want to buy everywhere, fulfil anywhere, and return anywhere in today's dynamic, omni channel fulfilment market. To meet this demand, customers must be able to respond fast. This is done using a distribution system.
This system will also get rid of the old pen-and-paper managing technique, which is time taking and super error prone. It will automate and greatly simplify the whole process and streamline fast working to satisfy client.
This system takes into account three characters of its working and revolves around them. Note that these actors are an abstract level representation of real-world entities. So, one entity might refer to several of its counterparts. For example, an admin might refer to either the owner of the distribution management system or the workers physically responsible for dispatching the goods to the customer. Both come under the heading of admin staff, but their functions and operational scopes are vastly different.

1.2.1 Key Takeaways

1. Distribution management oversees a company's whole supply chain.
2. Distribution management maintains order and client satisfaction. *(In this context, client satisfaction means order fulfillment on time and quick actions on any complaints, if any.)*

1.3 Advantages of the system

1. It keeps things organised. Without a proper management system in place, retailers would be forced to keep stock in their own locations, which is a bad idea, especially if the seller lacks adequate storage space.
2. A distribution management system simplifies things for the consumer as well. It enables them to launch a single application for a wide range of products. Consumers would have to visit multiple locations to get what they need if the system did not exist.

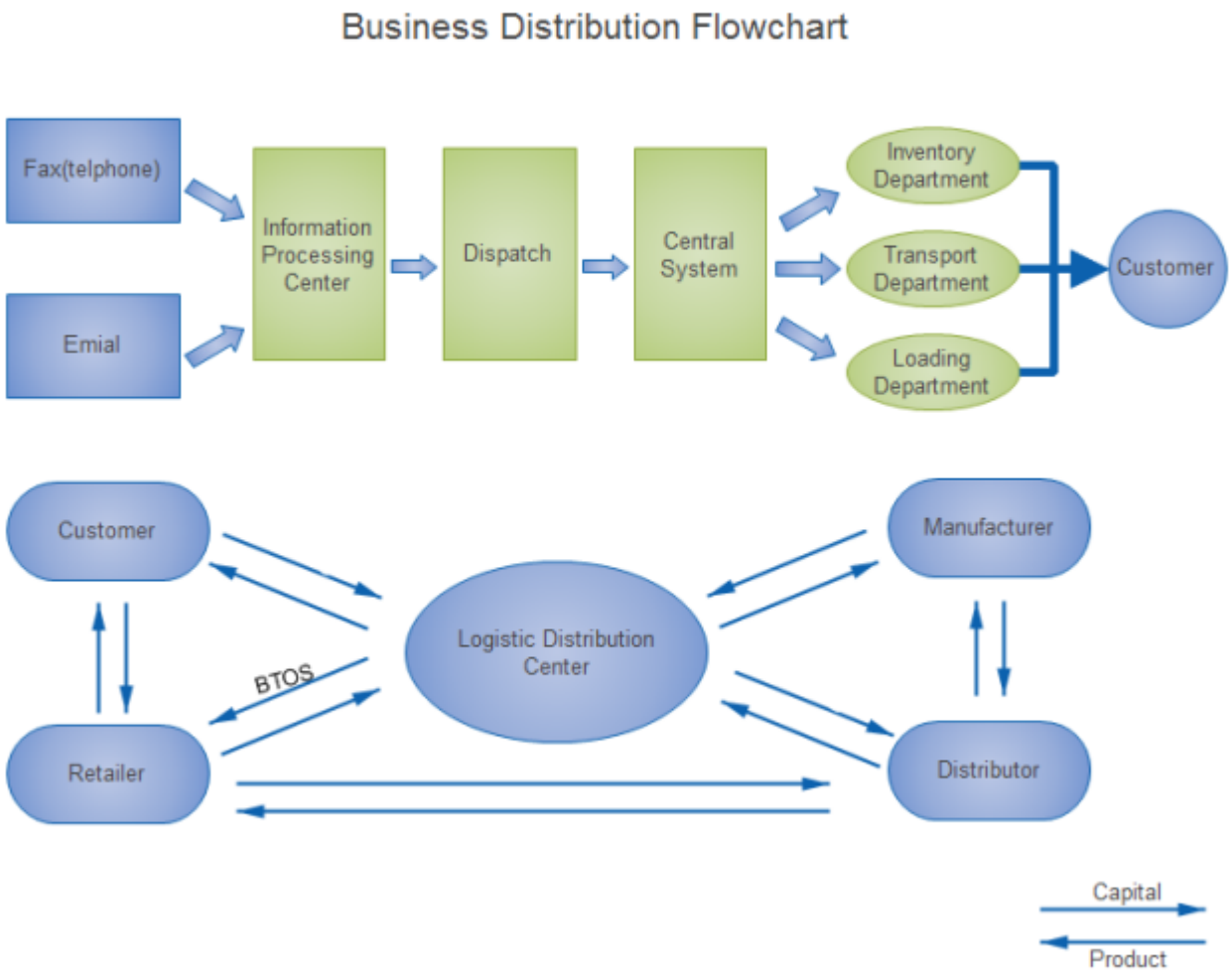

Figure 1.1: A sample business flowchart denoting key elements of a distribution management system.

Chapter 2

Proposal Information and Details

This chapter contains information made in the proposal about the project. The actual implementation of the project is the same as outlined in this chapter.

2.1 Project Features

Following are the features offered by this project to its end user.

- Administrator is able to implement inventory control
- Customers and Administrators are able to synchronize and perfectly able to execute and deliver Orders
- Customers are able to implement online payments
- Rider is able to navigate to his destination
- All actors are able to analyze their actions

What is meant by inventory control is that in stock management, an end user has control of three distinct arenas. That is, update a Product, Delete a product, and Add a product. By inventory control, specifically, we mean stock management. If stock is less than the limit, it should show up as defecient, so that admin can see the column and take action about it.
By navigation, we mean that an order should come up, and route till the order destination should automatically show up on the pane. This avoids us the pain of opening up Maps in the browser and then navigating to the external laypoint.
By analysis of actions, we mean that the rider against actions can be seen visually and saved in form of charts (*this principle per se also applies to other situations too*).

2.2 System Requirements

Table 2.1:

Language	• Python • C sharp • c++
Libraries Included/Python Modules	• Matplotlib (v.3.5.2) • Numpy(v. 1.23.5) • .NET framework (v 4.8 driver, 4.7 macro) • NewtonSoft JSON parser • GMaps Controls • SQL lite server

2.3 Project Actors

2.3.1 Primary Actors

- Admin will be the first actor. They will directly supervise the warehouse , the distribution network and its entities. They will be responsible for the overall upkeep in all aspects. This contains managers, production line workers etc.
- Riders are the second actor. They will be responsible for delivering the finished products from warehouse to end user.

2.3.2 Stake Holders

- Government agencies, which have an interest in collecting tax information.

2.4 Status of the project

This table denotes the status of the project we have implemented. All Data Structures excluding Hash Tables have been implemented. Link List, BST, queue and graph are integrated into project at various paradigm.

Table 2.2:

Use Case Id	**Use Case Name**	**Assigned To**	**Estimate Date**	**Status**	**Known Issues**
U07,08,26,27	Product CRUD	2021-CS-7	14 December	Completed	None
U01,19,20,21	Rider CRUD	2021-CS-7,12	13 December	Completed	View Rider is duplicated
U02,24,25,29	Shop CRUD	2021-CS-7	14 December	Completed	None
U07,08,26,27	Order,Cart CRUD	2021-CS-12,41	Completed	Completed	Bill N/A in Grid, Email server errors
U14	Authenticate User	2021-CS-12	12 December	Completed	Bug in Null User Check
U11	Logging Reservations	2021-CS-12	15 December	Partially Implemented	Not Tested as of yet
U17	Rider Fuel Logs	2021-CS-12,41	16 December	Partially Implemented	BST not being saved
U09	Billing Payments	2021-CS-12,41	17 December	Not Implemented	Not Tested as of yet
U22,23	Send Emails, Save Data	All	Misc.	Completed	None
U16	Sales Dashboard	2021-CS-41	17 December	Not Implemented	None
U28	Analytical Reports	All	18 December	Partially Implemented	Logic written but not integrated in UI
U15	Navigate Rider	All	20 December	Partially Implemented	Maps working fine, but route from DFS is troublesome
U05	Checking Order Status	2021-CS-41	20 December	Not Implemented	Not Tested as of yet

Continued on next page

Table 2.2: (Continued)

Use Case Id	**Use Case Name**	**Assigned To**	**Estimate Date**	**Status**	**Known Issues**
U06	Servicing Order Status	2021-CS-12	20 December	Not Implemented	Logic written but fetch is troublesome

2.5 Use Cases

Use case 01 - Add rider

Use Case ID	U01
Name	Add Rider
Actor	Admin
Description	The system's administrator can register new riders. The Admin will enter all of the riders' details, including their names, contact information, the regions they will be delivering to, and the orders that have been given to them. The rider will be added to the system and be able to log in once all the necessary information has been entered.
Flow	**Main Success Scenario *(Base Flow)*:** 1. Admin logs in to the system and presses the Add Rider prompt in the menu. 2. There are certain informations that the admin has to add about the rider (*trainee*) in the system. These are: • CNIC • Name • Phone • Address • username • password 3. After the rider is added in the system, his details are stored. **Extensions (*Alternate Flow*):**If at any time, the system fails, then: Admin requests the recovery of system to previous stable state, which includes, *but is not limited to*, • filling of all information which was filled before failure(*if there was any*) • If Add option was toggled, then the respective data must be saved into the system as usual. This goes without saying that the admin need not re-enter the information again. 1. If the manager leaves some information empty and presses the add button: • Control will not proceed ahead. A informative icon / popup is displayed, showing to the admin where they missed the information. Then upon entering the correct state, the control proceeds ahead. 1-2. CNIC is less than 13 digits or contains a dash, or contains a letter: • Appropriate use of validators will be applied in this case. The control will not proceed ahead if the CNIC is not according to format. 1-3. Phone number contains digits other than 11 in length, or any digit. • Same extension as in case 1-2. , except that in this case, validation digit • length would be 11 instead of 13.

Use case 02 - Add Shop

Use Case ID	U02
Name	Add Shop
Actor	Rider
Description	During the journey, rider encounters different shops. These shops need to be added into the system so that they can be taken orders from.
Flow	**Main Scenario:** 1. Rider presses the add shop prompt. 2. Rider fills in the Shop identifiers. 3. Shop is added into the system. . **Extensions:** 3. (a) Data Entered in the fields is incorrect or in the wrong format. i. An error message in the form of Message Box will appear if the error is detected. ii. Actor corrects the data. (b) At any point, if the system fails, it is restored to previous failsafe position. (c) If, during location selection, the external API from google maps fails, the whole page is reloaded for the actor to try again. .

Use case 03 - Checking Order Status

Use Case ID	U03
Name	Checking order Status
Actor	Admin
Description	Admin must be able to view dispatched orders status, and any recent developments.This can either be done using plain text messages, or using a sophisticated map tracker.
Flow	**Main Scenario:** 1. Admin opens up the orders pane in the menu. 2. Admin views order details. . **Extensions:** 1. . (a) If the order in question is already *serviced*, it will be removed.

Use case 04 - Add Order

Use Case ID	U04
Name	Add Order
Actor	Admin
Description	Rider needs to take order from rshopkeeper according to their requirements. This order is then placed and then the service part comes up.
Flow	**Main Scenario:** 1. Rider presses the Add Order prompt. 2. Rider chooses the quanitity of the selected product and presses the add to cart button. 3. Rider navigates to the cart and chooses the order details. He presses the Place Order button. 4. Order is placed. . **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. Which includes adding the product in to the cart again, right before the step of actually placing the order itself. The rider will then manually place the order himself. 2. If Quanitity of product is selected greater than the stock, the error message is rectified. 3. If wrong order is placed, the rider has option to delete it. .

Use case 05 - Service Order

Use Case ID	U05
Name	Service Order
Actor	Admin
Description	After an order is paced on behalf of the shopkeeper, the order *vis a vis* all its details comes over to the admin side for servicing. The service part ends by the acknowledgement of the customer.
Flow	**Main Scenario:** 1. The admin recieves order's list from the rider and will idspatch the order according to First in First out principle. 2. After the rider delivers the order, the status is updated to served. . **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. Which includes adding the product in to the cart again, right before the step of actually placing the order itself. The rider will then manually place the order himself. 2. If at any time, the wrong itme is dispatched by mistake, shopkeeper will be able to return the product to the rider. 3. If the product is dispatcehd and stock is finished, Admin will restock the product by delivery time. 4. By *force majeure*, if product is not delivered by the rider, full refund option is provided. .

Use case 06 - Updating Order Status

Use Case ID	U06
Name	Updating Order Status
Actor	Rider
Description	While delaing with products, riders need to update the order status at each step. This reflects the state of order at each step of the journey.
Flow	**Main Scenario:** 1. Rider opens up the orders pane. 2. Rider updates each order status from the value in the combo-box. . **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. Which includes adding the product in to the cart again, right before the step of actually placing the order itself. The rider will then manually place the order himself. 2. If orders pane contains contains orders already delivered, just toggle that order as forgone. 3. By *force majeure*, if product is not delivered by the rider, appropriate reflections are provided. .

Use case 07 - Add product

Use Case ID	U07
Name	Add Product
Actor	Admin
Description	Any distribution system needs one particular category of items to function, that is Products. This use case takes care of that fundamental paradigm.
Flow	**Main Scenario:** 1. Admin opens up the add product tab. 2. Admin fills all the identifiers which are: (a) Name (b) Price (c) ID (d) Category (e) Is Perishable 3. At toggling of the Add Product pane, the product is Added into the system. . **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. Which includes adding the product in to the cart again, right before the step of actually placing the order itself. The rider will then manually place the order himself. .

Use case 06 - Update Product

Use Case ID	U08
Name	Update Product
Actor	Admin
Description	Any distribution system needs one particular category of items to function, that is *Products*. During operation, various aspects need to be changed, that is stock *etc.* This use case takes care of that fundamental paradigm.
Flow	**Main Scenario:** 1. Admin opens up the Update Product tab. 2. Admin update Product Identifiers. 3. At toggling of update Product button, product is updated. . **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. Which includes adding the product in to the cart again, right before the step of actually placing the order itself. The rider will then manually place the order himself. 2. If at any time, the wrong itme is updated by mistake, admin will be able to return the product to the previous state. 3. If incorrect data is entered, validators will be applied. Error message will be shown too. 4. By *force majeure*, if product is not delivered by the rider, full refund option is provided. .

Use case 09 - Billing Payments

Use Case ID	U09
Name	Billing Payments
Actor	Rider
Description	Any order is incomplete without the payment part. This use case takes care of that fundamental aspect.
Flow	**Main Scenario:** 1. At Billing Payments scenario, the window in the browser opens up for the payment to occur. 2. Rider can choose any other payment method if they want to occur. This includes (a) Advance (b) Jazz Cash . (c) Pre Order payment **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. Which includes adding the product in to the cart again, right before the step of actually placing the order itself. The rider will then manually place the order himself. 2. Invalid term ID causes the rider to restart. 3. In case of incorrect payment, refunding facility is provided. .

Use case 10 - Delete Product

Use Case ID	U10
Name	Delete Product
Actor	Admin
Description	Any distribution system needs one particular category of items to function, that is *Products*. This use case takes care of that fundamental paradigm
Flow	**Main Scenario:** 1. Admin opens up the Delete Product tab. 2. Admin update Product Identifiers. 3. At toggling of Delete Product button, product is updated. . **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. Which includes adding the product in to the cart again, right before the step of actually placing the order itself. The rider will then manually place the order himself. 2. If at any time, the wrong product is deleted by mistake, Restore facility must be provided. .

Use case 11 - Log Reservations

Use Case ID	U11
Name	Log Reservations
Actor	Rider
Description	During mode of operation, rider may encounter complaints about the product. These need to be forwarded to admin about action.
Flow	**Main Scenario:** 1. Rider opens up the Complaints tab. 2. Rider records the complaints and forwards them to admin. . **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. Which includes adding the product in to the cart again, right before the step of actually placing the order itself. The rider will then manually place the order himself. .

Use case 12 - Add Route

Use Case ID	U12
Name	Add Route
Actor	Rider
Description	During shop addition, its route needs to be correctly added into the system. This use case takes care of that.
Flow	**Main Scenario:** 1. When a new Shop is added *See use Case U02*, new shop longitude latitudes should be added seamlessly. . **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. Which includes adding the product in to the cart again, right before the step of actually placing the order itself. The rider will then manually place the order himself. .

Table 2.26: Delete Order, Use Case 13

Use Case ID	U13
Name	Delete Order
Actor	Rider
Description	Sometimes a shopkeeper is not content with an order they wish to delete it, in this case this use case is handy.
Flow	Main Success Scenario (Base Flow): 1- Rider opens the orders pane. 2- Rider selects the target order. 3- Rider presses the Delete order button. 4- Order is deleted from the system. Extensions(Alternate Flow): At any point, if the system fails, the rider requests to reload the previous state. In this case, all previous information is restored. 2-a) If wrong order is deleted, an 'escape' option in form of a button is provided to the user to restore previous order states. 2-b) If order is deleted from the system, but later on a modification is required in the 'deleted order', new order pane is opened. Except the target attribute, all details are filled as they were previously.

Use case 14 - Authenticate User

Use Case ID	U14
Name	Authenticate User
Actor	Admin,Rider
Description	The entry point to any application is the login pane. If the user is entered correctly, then and only then a user can be added into the system.
Flow	**Main Scenario:** 1. Main Login screen opens up. 2. User enters correct credentials. 3. User is searched in the back end. 4. User enters the system. . **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. 2. Wrong parameters are notified to the user by informative prompts. 3. Users not found are also notified in the same way as Step 2. .

Use case 15 - Navigate Route

Use Case ID	U15
Name	Navigate Route
Actor	Admin
Description	After collecting all the orders from the inventory, the Rider is ready to deliver the orders to the location. To deliver the orders to their right location, the rider needs to a roadmap. The rider follows the roadmap and delivers the orders before the specified time.
Flow	**Main Scenario:** 1. The rider logs in the system, the checks in to the orders collection center and the orders are packed completely. 2. The Rider will start Delivering the order. 3. At toggling of interactive button, the complete route form rider location to the destination. 4. The rider will navigate to the location. 5. After finishing the Order, Rider clicks the end order button. 6. After End Order, if there is next order remaining, the route ti that address will be shown. . **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. 2. If at any time, Rider delivers order to wrong place, refund facility will be provided. 3. If any irrelevant path is provided to the rider, then the shortest path will be uploaded according to the Rider's current information. .

Use case 16 - View Sales Dashboard

Use Case ID	U16
Name	View Sales Dashboard
Actor	Admin
Description	Any business owner needs to realistically view the statistics of sales trends in a given period of time. This needs to be done in form of figures.
Flow	**Main Scenario:** 1. Main Login Screen is accessed by the user. 2. User opens up sales dashboard. . **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. 2. If dashboard is not opened correctly, reload the data. 3. If the dashboard does not contain any data: (a) Reload the page. (b) If it still does not do anything, the data wasn't present in the file. Show this fact by an underlying text. .

Use case 17 - Check Fuel Logs

Use Case ID	U17
Name	Check Fuel Logs
Actor	Rider
Description	Rider records his fuel expenses in this part of the application. This will be tracked in the weekly allownaces.
Flow	**Main Scenario:** 1. Rider opens up the Insert Fuel Logs tab. 2. Rider enters unique expenses in forms of 100. . **Extensions:** 1. If, at any time, the system fails, then the Rider requests the recovery system to the previous failsafe state. 2. Appropriate warnings would be taken if a non-unique entry is entered. .

Table 2.35: View Order History, Use Case 18

Use Case ID	U18
Name	View Order History
Actor	Admin
Description	Previous order trends need to be monitored time by time in order to guage what sales pattern is currently in flow with the shopkeepers. For this purpose, records about Product and pays need to be periodically maintained.
Flow	**Main Success Scenario (Base Flow)**: 1- Admin opens up the previous orders history pane from the menu. 2- Admin has a view of 4 products ordered (rest of them will be in a csv file) 3- Admin can view and download the csv in any directory of their choice. **Extensions(Alternate Flow)**: At any point, if the system fails, the rider requests to reload the previous state. In this case, all previous information is restored. 2-a): If csv file contains seemingly incorrect data: 2-a)-1: If it contains negative data (indicating returns/damages),an extra column in csv denoting status is filled with the status of order. 2-b) If the csv does not contain any data: Fill the C2 column of csv with the message 'Data not available due to new operations'.

Table 2.36: View Rider, Use Case 19

Use Case ID	U19
Name	View Rider
Actor	Administrator
Description	The Administrator as a lead is able to control and analyze the distribution system. The Administrator will be able to add riders. Hence, he\she will be able to view riders data. As a result, Admin has the control to manage riders.
Flow	**Main Success Scenario(Base Flow)**: The Admin as a system manager has the control to manage rides. Therefore, has the command to do the following tasks: 1. The Admin logs in to system. Then Admin Dashboard will appear to Admin. To view the Riders Admin will select Manage Riders Button from the menu. 2. After clicking the button the Admin will be directed to Manage Riders page. 3. Here Riders will be displayed in a list view. Every Rider in the list will have 2 buttons namely View Single Rider and Delete Rider. 4. To view single rider click on the view single rider button and the Admin will be directed to Rider Data Page. **Extensions(Alternate Flow)**:If at any time the system fails, the Admin requests the recovery of the system to the previous state, which includes, 1(a) The Admin logs in to system. In case the device shuts down or application crashes, the system will store its current state and Admin will be directed to the same page where the system was standing before the shutdown.

Table 2.37: Use Case 20 Deleting Rider

Use Case ID	U20
Name	Delete Rider
Actor	Admin
Description	The riders which are already present in the management team are sometimes need to be shifted to other positions or to be fired from the management team. Thus, by using the given prompt it is possible to delete a rider from the team to upgrade the system according to the requirements. If there is any need to update the riders then this is also be used by the admin to perform the tasks.
Flow	**Main Success Scenario (Base Flow)**: 1. The Admin first logs in to the system using his credentials and after confirmation admin clicks on the delete rider prompt to enter to the system to delete a specific rider according to the inquiry against the rider if any. 2. There will be a list of riders which are registered already into the system in which' the information of a rider is given as below: a. Rider's name b. Rider's ID c. Rider's Present Position 3. After the rider has been selected the admin at last has to pressed the delete prompt for the confirmation to delete the selected rider and an email will be generatred which will further sent to the rider for his acknowledgment. **Extensions (Alternate Flow)**: 1. If there is any problem in the admin's credentials that admin can not logs in to the system then the admin will again first register then again logs in to the system to perform his duty. 2. If admin mistakenly delete any other rider then admin first logs in to the add rider to registered the deleted rider again. 3. If there is no rider according to the info which has to be deleted then there will be two tasks performed by the admin which are given below: a. Admin refresh the system if there is any glitch. b. Admin will take a completer procedure to registered the rider first then delete it from the system. 4. If the email is not generated and sent to the rider Admin will be interact with the developing team to re-consider the system again.

Table 2.38: Update Rider, Use Case 21

Use Case ID	U21
Name	Update Rider
Actor	Rider
Description	Rider needs to update their details. This use case comes in handy in that aspect.
Flow	**Main Success Scenario (Base Flow)**: 1- Rider opens up the 'Update your details' pane from the menu. 2- A page in this regards opens up. 3- The page has textboxes containing the details. Only the text box containing Rider CNIC will be blocked from modification. 4- Rider can modify their details as they wish. **Extensions(Alternate Flow)**: At any point, if the system fails, the rider requests to reload the previous state. In this case, all previous information is restored. This goes without saying that the rider need not fill the information again. 3-a): If the rider enters incorrect phone number: 3-a-s): Appropriate use of validators will be applied in this case *(length of phone num ber must be 11 digits and starts with 03 and must not contain any letter).* 2-b) If password is edited, its length must be 8 digits. 4-a): If, a rider detail isn't modified after the process, an 'escape' facility in form of 'Previous Changes' is applied. It will show changes *(not completely, just in required places)* which the user can then synchronize. 4-b): If a field is left empty, then an informative icon/popup is displayed so that the end user can directly go to that field and do that.

Table 2.39: Send Emails, Use Case 22

Use Case ID	U22
Name	Send Emails
Actor	Admin
Description	After a task is completed, the administrator is responsible for contacting the customer and sending a confirmation email, such as "Your order has been placed successfully!" or "Your order has been delivered!".
Flow	**Main Success Scenario (Base Flow)**: 1- When an order is successfully placed in the system (U04), an email is delivered to the customer having text "Your order has been placed successfully". 2- When the rider succesfully delivers orders, an email to this effect will be sent. *See Emails Section for more details.* **Extensions(Alternate Flow)**: 1- If an order is not placed due to payment defeciency or stock reasons, an email will be sent to this effect. 2- If an order is dispatched, but the shopkeeper was not present, an email will be sent to this regard. This will denote the shopkeeper must now recieve the order personally from the warehouse.

Table 2.40: Save Data, Use Case 23

Use Case ID	U23
Name	Save Data
Actor	Admin, Rider
Description	Data Storage is an essential part of any application flow. This case deals with this use case.
Flow	**Main Success Scenario (Base Flow)**: 1- At every step of application flow, data is stored. 2- Data Structure and file used for the format depends on use case involved. *See Data Storage Section for more details.*

Table 2.41: Update Shop, Use Case 24

Use case ID	U24
Name	Update Shop
Actor	Rider
Description	At any time of operation, the shop details need to be updated. This may include, but is not limited to, shopkeeper contact number is changed, shop mode is changed etc. This use case deals with this paradigm.
Flow	Main Success Scenario(Base Flow): 1- Rider opens up update shop pane. 2- A page opens up in this regard. 3- Rider updates required shop details in this regard. Extensions (Alternate Flow): At any point, if the system fails, the rider requests to reload the previous state. In this case, all previous information is restored. 3-a) If any field is left empty, an informative pane is displayed in this regard. Appropriate measures such as scrolling to the vacant component are also employed. 3-b) If any invalid data is entered, appropriate use of validators is employed. For purposes of data security, updating critical features such as CNIC of shopkeeper are prohibited. 3-c) If any wrong shop data is modified, reverting back to previous state is to be made possible.

Table 2.42: View Shop, Use Case 25

Use case ID	U25
Name	View Shop
Actor	Admin
Description	At any time of operation, the shop details need to be viewed. This may include, but is not limited to, shopkeeper contact number is changed, shop mode etc. This use case deals with this paradigm.
Flow	Main Success Scenario(Base Flow): 1- Admin opens up view shop pane. 2- A page opens up in this regard. 3- Shop details appear in a list format. 4- Admin views required shop details in this regard. Extensions (Alternate Flow): At any point, if the system fails, the rider requests to reload the previous state. In this case, all previous information is restored.

Table 2.43: View Product, Use Case 26

Use case ID	U26
Name	View Product
Actor	Admin , Rider
Description	At any time of operation, the product details need to be viewed. This use case deals with this paradigm.
Flow	Main Success Scenario(Base Flow): 1- User navigates to and opens up view product pane. 2- Shop details appear in a list format. 3- User views required shop details in this regard. Extensions (Alternate Flow): At any point, if the system fails, the rider requests to reload the previous state. In this case, all previous information is restored.

Table 2.44: Use Case 27, Delete Product

Use Case ID	U27
Name	Delete Product
Actor	Administrator
Description	The Administrator as a lead is able to control and analyze the distribution system. The Administrator will be able to add products. Hence, he\she will be able to view products data. As a result, Admin has the control to manage inventory.
Flow	**Main Success Scenario(Base Flow):** The Admin as a system manager has the control to manage products. Therefore, has the command to do the following tasks: 1. The Admin logs in to system. Then Admin Dashboard will appear to Admin. To delete the products Admin will select Inventory Button from the menu. 2. After clicking the button the Admin will be directed to Inventory page. 3. Here products will be displayed in a list view. Every product in the list will have 3 buttons namely View Single Product, Edit Product and Delete Product. 4. To delete a single product click on the delete button and the product will be deleted from the inventory. **Extensions(Alternate Flow)**:If at any time the system fails, the Admin requests the recovery of the system to the previous state, which includes, 1 (a) The Admin logs in to system. In case the device shuts down or application crashes, the system will store its current state and Admin will be directed to the same page where the system was standing before the shutdown. 1 (b). In case the Admin deletes a specific product from the inventory but it still displays at the Riders Side Interface. If the Rider tries to add that product to the cart, a Warning message will display saying ”This Product is no more Available”.

Table 2.45: Use Case 28 Analyze Reports

Use Case ID	U28
Name	Analyze Reports
Actor	Admin
Description	Reports are basically the records which convey the business activities and the financial reports on the monthly basis. Admins on monthly basis considered the reports for knowing what and which percent their systems and workers gave them output and analyze reports and records completely for the betterment of the their companies. Admins will analyze the reports and considered the profits and the total revenue on the daily and the monthly basis.
Flow	**Base Flow**: 1. If any admin has to analyze the report then firstly admin has to logs in to the system and presses the Reports section to analyze it. 2. There will be the graphs and the boxes which shows the following information: 2.1 Total Monthly Profit percentage 2.2 Total Monthly Customers rate 2.3 Total Revenue Generated on Monthly Basis 3. There will be a total Traffic graph also shown in the dashboard so the admin can easily take an estimate about the total social network of people with their Management system. **Alternate Flow** : If there is any deficiency in profit percentage and monthly revenue then there will be a query generated by the admin which will be responded and the percentage will re-calculated.

Table 2.46: Delete Shop, Use Case 29

Use Case ID	U29
Name	Delete Shop
Actor	Rider
Description	The Rider as a co-lead is able to control the distribution system. The Rider will be able to add shops. Hence, he\she will be able to delete shops. As a result, Rider has the control to manage shops.
Flow	**Main Success Scenario(Base Flow)**: The Rider as a system manager has the control to manage shops. Therefore, has the command to do the following tasks: 1. The Rider logs in to system. Then Rider Dashboard will appear to Rider. To delete the shops Rider will select View Shops Button from the menu. 2. After clicking the button the Rider will be directed to Shops page. 3. Here Shops will be displayed in a list view. Every shop in the list will have 3 buttons namely View Single Shop, Edit Shop and Delete Shop. 4. To delete a single shop click on the delete button and the shop will be deleted from the shops list. **Extensions(Alternate Flow)**:If at any time the system fails, the Admin requests the recovery of the system to the previous state, which includes, 1 (a) The Rider logs in to system. In case the device shuts down or application crashes, the system will store its current state and Admin will be directed to the same page where the system was standing before the shutdown. 1 (b). In case the Rider deletes a specific shop from the shops-list but it still displays at the Admin Side Interface. If the Admin will try to do anything with that particular shop a Warning message will display saying ”This Shop is no more On Our Customers List”. 2. After Deleting a shop the Shortest Path Property will hold i.e. every route will have minimum distance as compared to other routes following the same destination.

2.6 Object Oriented Features

2.7 Classes

Following is the summary of classes in our project proposal.

Table 2.47: Class Summary

Class Name	Software/Domain	Is Abstract (Yes/No)	Is Singleton (Yes/No)	Is the class will has parametrized Constructor (yes/No)
User	Real World	NO	No	Yes
Payment	Software		Yes	
Rider	Real World			
Admin	Real World			
Product	Real World			
LineItem	Business			
Shopkeeper	Real World			
Order	Business			
Directions	Software			
Shop	Business			

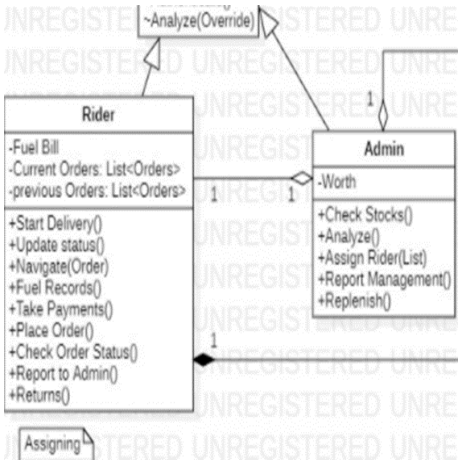

Figure 2.1: Polymorphism

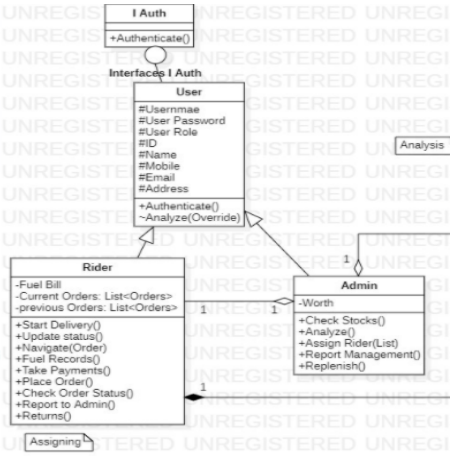

Figure 2.2: Inheritance in the project

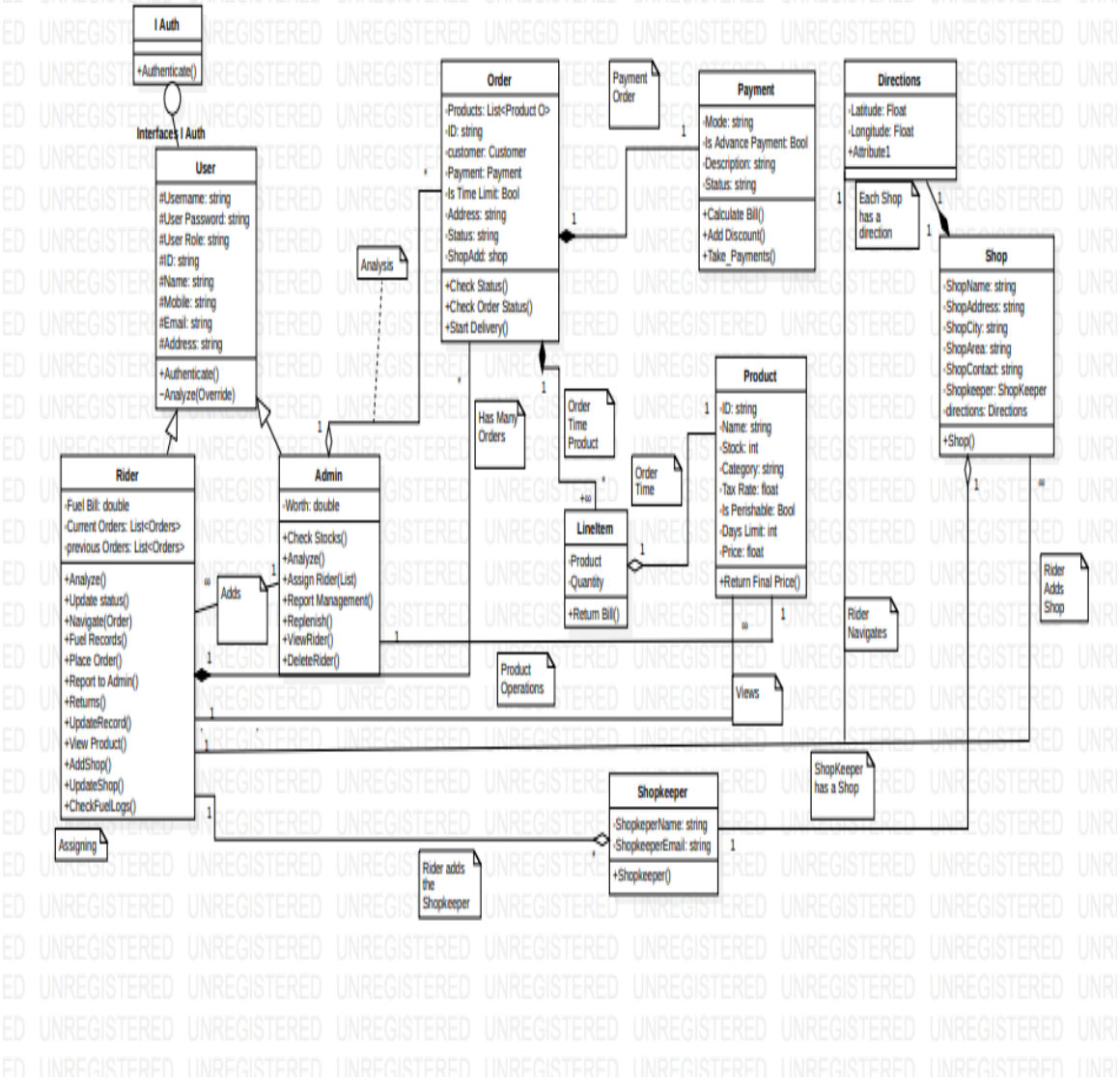

Figure 2.3: Class Diagram Of the project

Chapter 3

Technical Phases

3.1 Data Strucutres Mapped to Each Entity

3.1.1 Rider

Linked Use Cases

U01,14,19,20,21

Data Structure Used

Link Lists

Space Complexity of Chosen Data Structure

O(n)

Alternate Options

Queue,Binary Search Tree(BST),Graphs,Stacks

Justification

Link List is a linear Data Structure, that involves a Node(value), also having to store the address of another node, next or previous. A Link List having the address of both previous and next nodes is called a doubly link list while a list having the address of only one node is called a singly link list.
The Principal Reason behind using Link List for all use cases involving rider is that multiple keys can hold the same value, even though they are distinct objects.
For example, Rider (i) may have the name John Doe, while Rider (ii) may also have the name John Doe. Both of these 'John Does' have distinct CNIC numbers. This

distinctness of CNIC numbers could have prompted the usage of Binary Search Trees in storing their information. This would have made the search operation more efficient by a factor of *lg* n.
But the issue with Binary Seach Tree(BST) is that it does not allow for duplication of values. While CNIC numbers are different in the above example and indeed it does help in BST, but if a search operation on BST, targeting name attribute, is run it may yield no result. Because two distinct nodes are having 'John Doe's, which is impossible according to the very condition of BSTs. This hampers the search operation and thus rules out BST as a data structure. As far as graph is concerned, no particular 'relationship' between each rider is needed that would necessitate the usage of edges.
As far as Queues and Stacks are concerned, they are indeed linear data structures. But they follow a principle of First in, First out and Last in, first out respectively. No such order is needed in management of riders. So the usage of Queues and Stacks is ruled out.

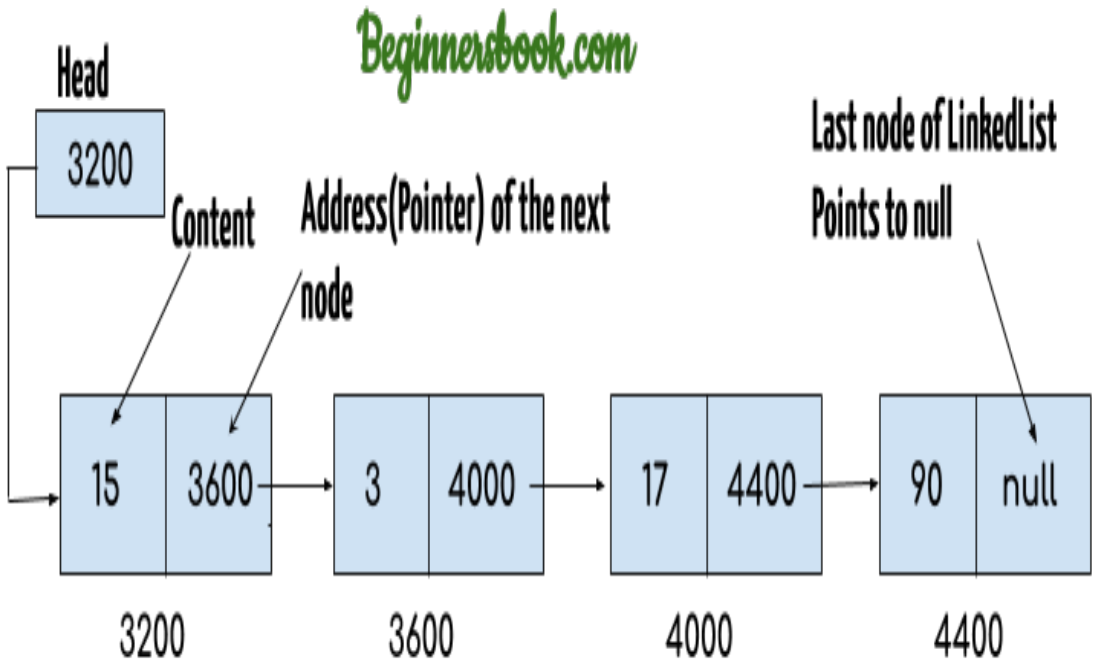

Figure 3.1: A representation of linked list. Each box is a node, containing the address of next node.Source:Beginnersbook.com

3.1.2 Shop

Linked Use Cases

U02,U24,U25,U29

Data Structure Used

Graphs(*Adjacency List Format*)

Space Complexity of Chosen Data Structure

O(V+E) *in average*, O(n^2) *in worst case complete graphs*

Alternate Options

Queue,BST,Link Lists, Stacks

Justification

Graphs are basically networks of edges and vertices. A vertex is a value and edges denote a relationship between vertices.
The reason behind using graphs in adjacency list format is that each vertex 'shop' has relationship in form of edges 'paths' to other shops. Consequently, minimum spanning trees can be calculated using the graph to estimate all paths from one vertex to another. Additionally, path from one vertex to another is also possible from graph. No other data structure offers such versatility in path finding as graphs.
As far as linear data structures are concerned, in large values, they will exponentially grow. So they are ruled out.

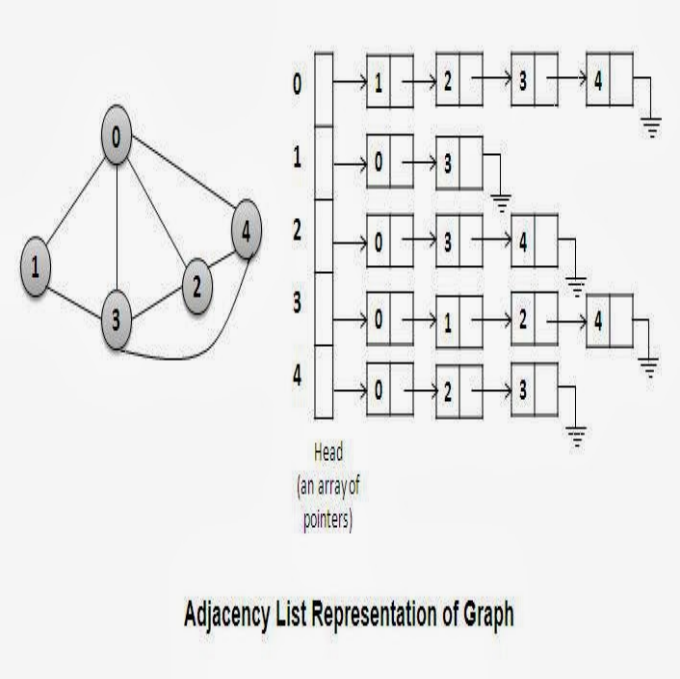

Figure 3.2: A representation of graphs in adjacency list format.Source:Open Source

3.1.3 Order

Linked Use Cases

U03,04,05,06,13,18

Data Structure Used

Queue

Space Complexity of Chosen Data Structure

O(n)

Alternate Options

Link Lists, BST, Graphs,Stacks

Justification

A queue is a linear data structure that involves a particular order of modification, First in First out (FIFO). That means each incoming node will be placed at the back of the data structure and each outgoing element will be at the front of the data structure.
The justification behind using queue for all orders is that orders have a particular tendency, that is each order that is placed first needs to be serviced first. Orders do have distinguishing keys such as order ID, tracking ID etc. This makes BST in form of a priority queue, a suitable contender for data structure. But linearity of queue coupled with, the defining order without any extra bit for storing priority, trumps it over BST for usage in order management.
Other Data structures have no such features so that they can be considered in it.

3.1.4 Product

Linked Use Cases

U07,08,10,26,27

Data Structure Used

Link Lists

Space Complexity of Chosen Data Structure

O(n)

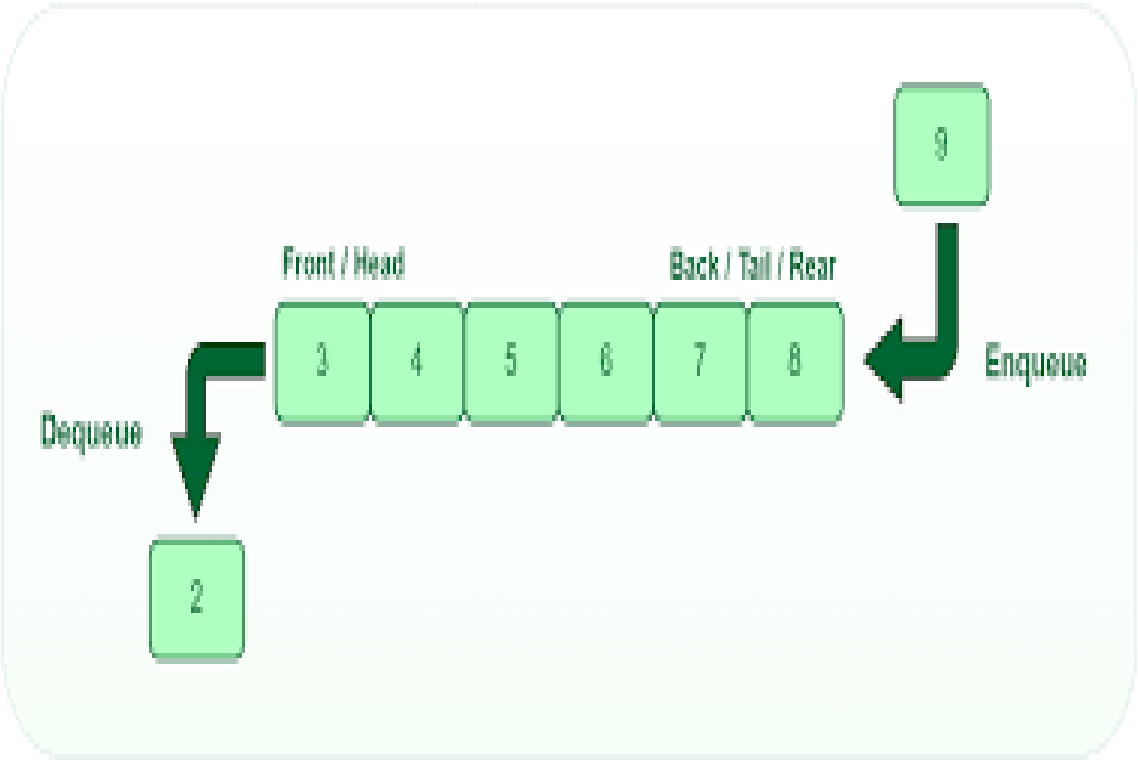

Figure 3.3: A representation of queue. Notice the First in, first out FIFO order- Source:Open Source

Alternate Options

Queue,Binary Search tree, Graphs, Stacks

Justification

Link List is a linear Data Structure, that involves a Node(value), also having to store the address of another node, next or previous. A Link List having the address of both previous and next nodes is called a doubly link list while a list having the address of only one node is called a singly link list.
Product IDs are definitely unique. This helps in storing them in BSTs. But consider a case scenario. Multiple products of the same ID are stored in a BST. The multiplicity of products in same ID can be best described as stocks. But, if categories are different, which one of the same ID products the BST should yield?
Additionally, at very large numbers of products, the height of BST becomes very large and it is a costly operation to balance it.
Because of this reason, BST is ruled out and Link List is preferred.
As far as Queues and Stacks are concerned, they are indeed linear data structures. But they follow a principle of First in, First out and Last in, first out respectively. No such order is needed in management of products. So the usage of Queues and Stacks is ruled out.

3.1.5 Fuel Logs

Linked Use Cases

U17

Data Structure Used

BST

Space Complexity of Chosen Data Structure

O(lg n) *average* , O(n) *worst case*

Alternate Options

Queue,Link Lists,Graphs,Stacks

Justification

The data structure known as the "Binary Search Tree" based on nodes includes the following characteristics:

- A node's left subtree has key values less than the node itself.
- Only nodes with keys higher than the value of node's key are found in the right subtree of node.
- At every node in the tree, the left and right subtrees must be also Binary search Trees.

The reason behind using binary search trees in rider fuel logs management, is that in each fuel log, there is a distinct bill that is unique. So it can be employed using a BST. Constraints are applied on values such that they are unique and distinct.
Additionally, the In order traversal of a binary search tree always yields a sorted output. This can be used to sort the bill-day pairs of fuel.
For these reasons, a BST is preferred for storing Rider Fuel Bills.
As far as linear data structures are concerned, sorting them involves a trade off between space and time. For example, selection sort preserves space of O(n), but trades off equivalent time of O(n^2). As far as merge sort is concerned, stable sort yields a space of O(n) in average case too. But it is time efficient inducing a factor overload of O(n lg n).

3.2 Exceptions

An exception in computer programming is a special condition encountered during program execution that is unexpected or anomalous. An exception occurs when a pro-

gram attempts to open a file that does not exist or encounters a read error. This may also occur if any unexpected behaviour is encountered on runtime.
To avoid a fatal error, the programmer must anticipate exceptions and properly handle them in the program code, branching program execution as needed. Exception handling is a feature of computer programming.

Table 3.1: Possible Exceptions in system

Type of Exception	Why this exception will occur	Use Case Id in which exception could be occurred	How you will handle the exception
Runtime	Incomplete/Empty Data	All	Reload operation
No such Field	Product is out of stock / incomplete entries	U7,U8,U10,U26	Reimbursement, logging, notices
Runtime	Two orders of same composition are made at the exact same time	U4	Rearranging the orders
Interrupted	Maps/ Jazzcash API doesn't work correctly	U9,U15	Reload operation
Runtime	Discrepancy in rider fuel logs	U17	Cancel, log to admin
Format Error	Wrong Data, doesn't fit according to validators	All where validation is required	Inform the end user about the mistake
Location	Shop location is either occupied or doesn't exist	U2, U24	For maximum security layer, shop will not be allowed to add in the system
Chart Plotting Error	Chart is asked to plot between invalid pair of values	U28	Chart plotting operation is closed on account of incompatibility

3.3 Data Storage

File handling will be used by us as a mechanism for data storage in this project. The format for this storage will vary as per requirements, we will use both csv files and txt files.

Table 3.2: parser.txt Text File

File Name	parser.txt
File Type	Text File
Data Format and values	Variable data entries, contains two lists of any data type in two lines in comma separated forms
Special Purpose *(if any)*	Transfer data from front end to grapher.py Python file to display charts *(see Analytical Reports section)*

Table 3.3: CurrentOrders.csv

File Name	CurrentOrders.csv
File Type	csv file
Data Format and values	Column A : List<Line Item> Column B: Shop Name Column C: Shop Address Column D: Shopkeeper Name Column E: Shopkeeper Contact Column F: Rider Name Column G: Rider Contact Column H: Bill
Special Purpose (if any)	None

Table 3.4: Products.csv

File Name	Products.csv
File Type	csv file
Data Format and values	Column A: Name Column B: Price Column C: Tax Ratio Column D: ID Column E: Category Column F: Is Perishable
Special Purpose (if any)	None

Table 3.5: Users.csv

File Name	Users.csv
File Type	csv file
Data Format and values	Column A: Username Column B: Password Column C: Role Column D: CNIC Column E: Address Column F: Phone Number
Special Purpose (if any)	None

Table 3.6: Shops.csv

File Name	Shops.csv
File Type	csv file
Data Format and values	Column A: Shop Name Column B: Shop Address Column C: Shop City Column D: Shop Area Column E: Shop Contact Column F: Shopkeeper Name Column G: Shopkeeper Email Column H: Latitudes Column I: Longitudes
Special Purpose (if any)	None

3.4 Email Sending

1. An email will be sent at order placement. It will be a short email having content, " Greetings customer **XYZ**, an order has been placed under order id *ABC11*. Thanks for using our distribution network".

2. An email will be sent at order payment. It will be a short email having content, " Greetings customer **XYZ**, an order id **ABC** has been paid under payment id **XYZ** using *method of payment*. Thanks for using our distribution network".

3.5 Analytical Reports

In Analytical reports, we will generate 5 types of reports.

- An order report for the admin dashboard to verify sales.The format of this report will be a pdf file, availabale for download. It will contain a traffic bar chart.

- Profit Loss Trends. This will be a pdf file, available for downloads. This will be a graph chart.
- Sales Number Trends. Format and description same as above.
- Rider Trends.This will be an executive summary of all orders worth against each rider, having a bar chart format.
- Rider Fuel Logs. This will be a pie chart, not available for download, It will contain a weekly report of all rider fuel expenses.

3.6 Project Plan

Use Case Id	**Use Case Name**	**Member Name**	**Estimate Date**	**Complete Date**
U01,19,20,21	Add Rider, View Rider ,Update Rider,Delete Rider	Afraz and Shahzaib	13 December	17/12/22
U02,24,25,29	Add Shop, Update Shop, View Shop,Delete Shop	Shahzaib	14 December till 18:00	17/12/22
U07,U08,U26,27	Add Product, Update Product,View Product, Delete Product	Afraz and Hamza	14 December till 23:00	17/12/22
U04,18,13	Add Order, Update Order, View Order, Delete Order	Afraz	15 December till 18:00	19/12/22
U14	Authenticate User	Afraz	12 December	12/12/22
U11	Logging Reservations	Afraz	15 December till 21:00	17/12/22
U17	Rider Fuel Logs	Afraz and Shahzaib	16 December	17/12/22
U09	Billing Payments	Hamza and Afraz	17 December	N/A
U22,23	Send Emails, Save Data	All	Miscellaneuos	Misc
U16	Sales Dashboard	Hamza	17 December	N/A

U28	Analytical Reports	All	till 18 Decemeber	Partial. work
U15	Navigate Rider	All	till 18 December 23:59	21/12/22

3.7 Challenges

Time was most probably the biggest challenge in the implementation of this project. The lack of time, coupled with the enormity of the subroutines involved, made it incredibly difficult for us to fulfill the project. Still, we tried our level best to accomplish the project. Some other challenges we faced were:

1. The data structure implementation of BST was getting hampered by type errors. For unknown reasons, The generic type object was not getting compared with other objects, even when lauded against IComparable interface. We solved that problem by applying a constraint on the very data structure, such that each entity in the data structure MUST be a class.

2. Pointer management in C Sharp is not allowed under normal circumstances. One prospective solution was to use the unsafe context. But unsafe context was getting hampered by the lackluster non-applicability of grouped statements. That is, normally a developer can mark a bunch of statements as *unsafe*. But in this scenario, they were not getting marked unsafe.

3. Implementation of graph data structure was another big problem. Basically, Graphs in adjacency list format were difficult to adjust in Bellman Ford Path finding algorithm implementation.

4. Calling Python backend in the application was another hassle. We managed it by using the Process class and its child class, ProcessStartInfo in the System library of the c sharp, but that had its own drawbacks. For instance, Process relies heavily on manual configuration for its action. This has an inherent drawback, as the path to Python Interpreter is different on each system and has to be provided in *absolute* mode.

5. Integrating GMaps. Control fetching Geocoding locations was not being typecasted correctly. Once that was done, Minimum spanning tree using Prims Algorithm was made. But that too, once made, was not being fetched results correctly. A depth first Algorithm had to be run on the graph to fetch strings. But the code used a dictionary of tuples (*It was provided in the documentation of the code.*)

6. Cross referencing in the project. IF we had to single out just one problem in the

whole project, it would be this one. Basically, whatever work one member of ours did, it was not being pushed over git repository to the other ones, resulting in wastage of time. This fact is evident due to the number of delete-restore commits we have made in the project.

7. Generic Data Structures are very effecient in their working. They help to skip type errors encountered during runtime. Plus, they work with all sorts of data. But implementing Generic Data Structures is a difficult job. And extending this '*genericity*' to hash table seemed impossible for us, let alone a single hash table. Consequently, that remains unimplemented.

8. Extending the functionality of a queue in form of own lists. To Delete Orders, the basic principle of a queue has to be broken. That was possible only if the queue was out of box, i.e. not of the System.Collection.Generics, but own Data Strucuture.

More, complete tracking of project implementation will be found on
https://gitlab.com/buttafraz22/cs261f22finalpid01.git