Software Engineering Class Project

Automating Workshop Software

1. Project title:

Automating Workshop Software

2. Course name and course code:

Software Engineering, Comp220

3. Students' names and IDs:

241546646 Ammar Tariq

241575763 Abdul Rafay Ahmed

231471220 Fahad Siddique

4. Date of submission:

6th July 2022

5. Submitted to:

Dr Saba Toor

6. System Features

6.1 Adding a client

This feature adds a client and their details to the system. It has a high priority in the system.

6.2 Printing clients list

This feature prints the clients list in the system, telling us about the clients. It has a medium priority in the system.

6.3 Deleting a client

This feature deletes a client and their details from the system. It has a low priority in the system.

6.4 Adding a mechanic

This feature adds a mechanic and their details to the system. It has a high priority in the system.

6.5 Printing a mechanic list

This feature prints the list of mechanic in the database. It has a medium priority in the system.

6.6 Deleting a mechanic

This feature deletes a mechanic and their details from the system. It has a low priority in the system.

6.7 Repairing a vehicle

This feature will discuss the repairing to be done on the client's vehicle. It has a high priority in the system.

6.8 Making an invoice

This feature deals with the invoice part, basically of the jobs done by the client. It has a high priority in the system.

6.9 Payment method

This feature will deal with how the client would like to pay for the invoice. It has a high priority in the system.

Table of Contents

7. Introduction

7.1 Purpose

The product we are developing is a workshop system, to be used in a regular workshop. The product has not yet been released. The product has intended usage in a regular workshop and will help them with their tasks.

7.2 Document Conventions

There are no standard or typographical conventions used in this document.

7.3 Intended Audience and Reading Suggestions

The document can be read by everyone managers, developers, testers etc. It contains basic details and also extensive details about the software. The document can help readers know more about the software and how it works and intended use. The document starts with basic details then goes into further details of the software.

7.4 Product Scope

The software basically helps the workshop run smoothly by implementing and managing their daily tasks in a software. The software will be easy to use, as their work will become more efficient and easier. The software would be used to complete and manage daily, and regular tasks and it will also keep track of the work done for each client. Completing and maintaining tasks, jobs and clients will be the main goal of the software.

8. About the Project

8.1 Product Perspective

I have come up with a self-contained idea to develop a highly beneficial software for the workshops with mechanics for a much quicker repair of vehicles. This product is completely self-contained and will be implemented for the very first time.

8.2 Problem Domain

Problem domain for the software will be the multiple mechanics and their outlets doing various jobs. A menu, through which the workshop member will locate and figure out the required items and do the required jobs for the client's vehicle.

8.3 Problem

The system will help workshops run more smoothly as of right now workshops are not automated. It is a total nuisance to travel in rush hours and getting late due to workshops incompetence (human errors, time management). Hence, this is a user-friendly software in order to make these issues less complicated.

8.4 Solution Domain

The workshop system is preferably going to be achieved through python language. An input message will be displayed to the user of the software where after the car and client details will be inserted. A list of related items will be shown to choose from. Variety of payment methods. The jobs and services will be displayed through a database and a friendly interface through the help of python.

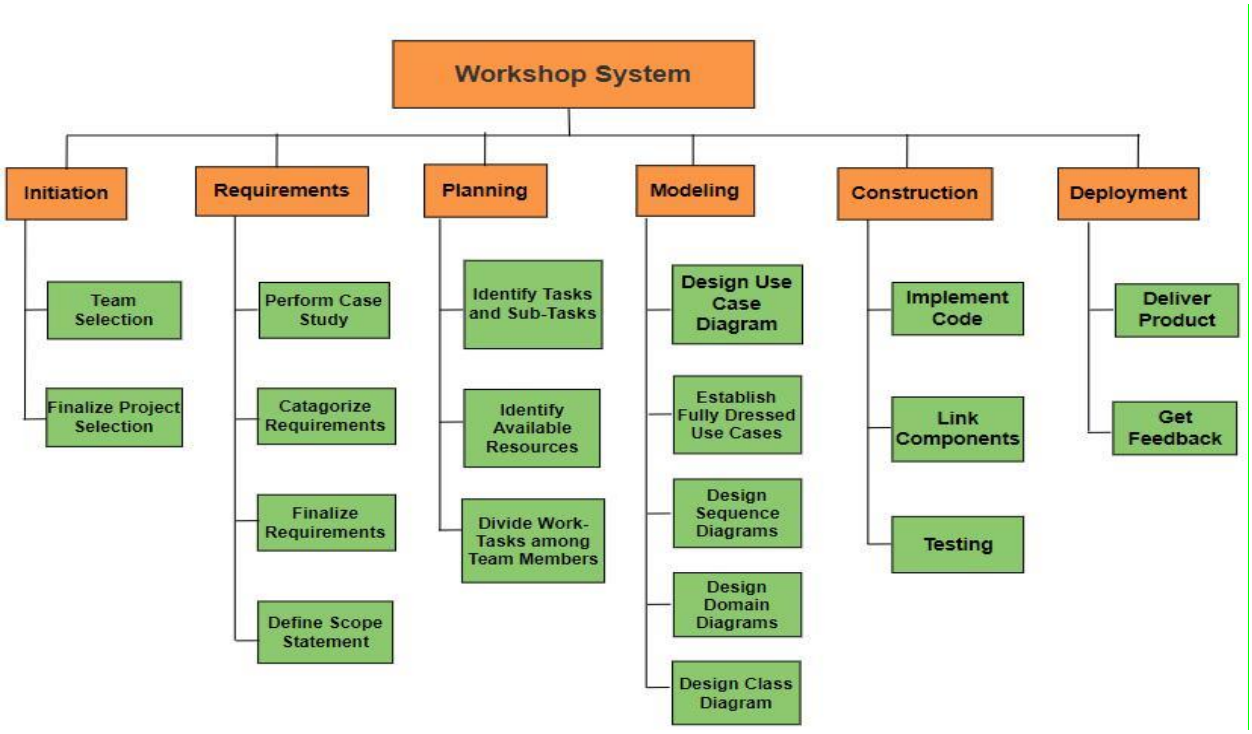
9. SDLC used for project

We use waterfall method for our project because our requirements were clear and understandable, and we have to make a short project therefore waterfall is used because it is good for such projects.

10. Software Architecture for project

Because our project is tiny, we apply the layered architecture technique, and three levels are created. Because it makes it simple to describe project features like performance, quality, scalability, maintainability, manageability, and usability, we employ layered architecture. The first layer includes adding and removing clients, mechanics, and car repairs; the second layer includes invoices and payment methods; and the third layer includes all of the system's data.

11. Work breakdown structure:



Automating Workshop Software

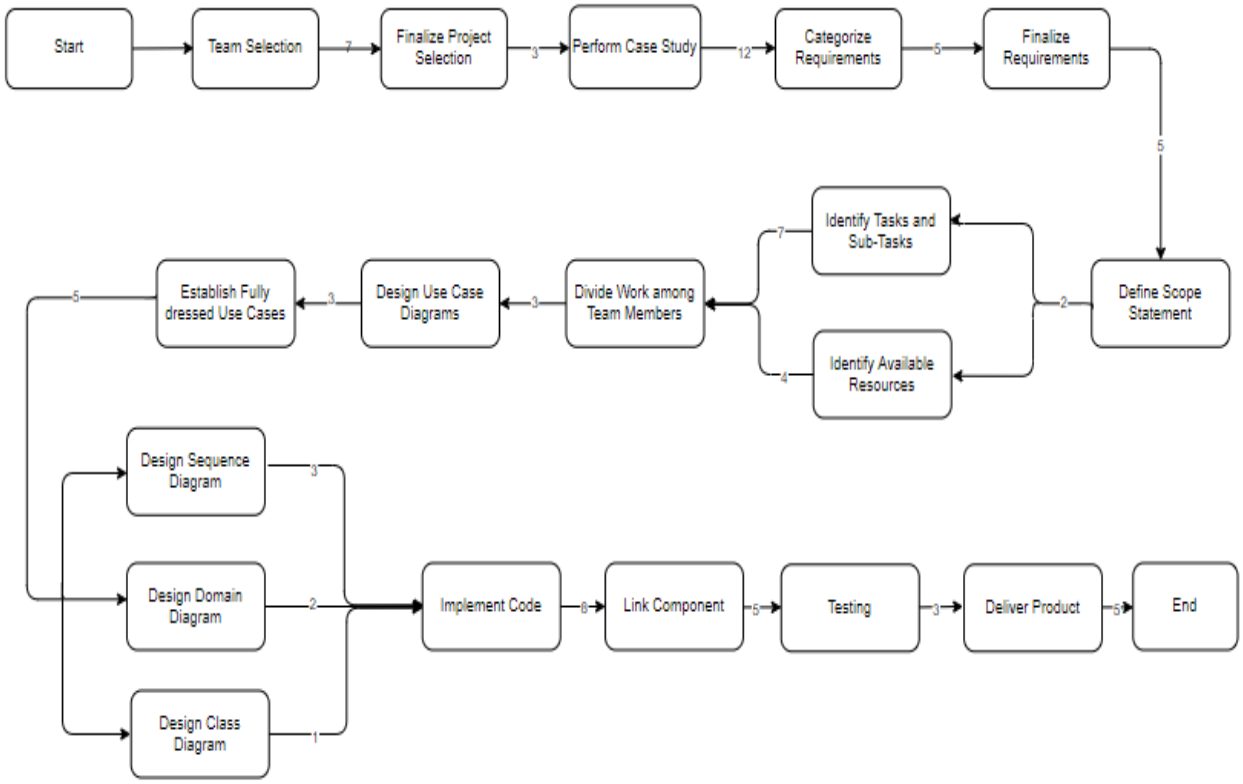
12. AON Diagram

12.1 AON Table

1. Activity	Durations (Days)	Predecessor
Start	0	Null
Team Selection	7	Start
Finalize Project Selection	3	Team Selection
Perform Case Study	12	Finalize Project Selection
Categorize Requirements	5	Perform Case Study
Finalize Requirements	5	Categorize Requirements
Define Scope Requirements	2	Finalize Requirements
Identify Tasks and Sub-Tasks	7	Define Scope Requirements
Identify Available Resources	4	Define Scope Requirements
Divide Work Tasks among Team Members	3	Identify Available Resources
Design Use Case Diagram	3	Divide Work Tasks among Team Members
Establish Fully dressed Use Cases	5	Design Use Case Diagram
Design Sequence Diagrams	3	Establish Fully dressed Use Cases

Design Domain Diagrams	2	Establish Fully dressed Use Cases
Design Class Diagrams	1	Design Domain Diagrams
Implement Code	8	Design Sequence Diagrams, Design Domain Diagrams, Design Class Diagrams
Link Components	5	Implement Code
Testing	3	Link Components
Deliver Product	5	Testing

12.2 AON Diagram



13. Requirements

13.1 User requirements:

- Only the receptionist has the authority to make changes to the database's information.
- The system should collect all a new client's and mechanic information and register him.
- The system should delete any client's and mechanic.
- Do not recreate the client if one already exists.
- Print all clients and mechanics.
- Print list of services we provide and can print invoice.

13.2 System requirements:

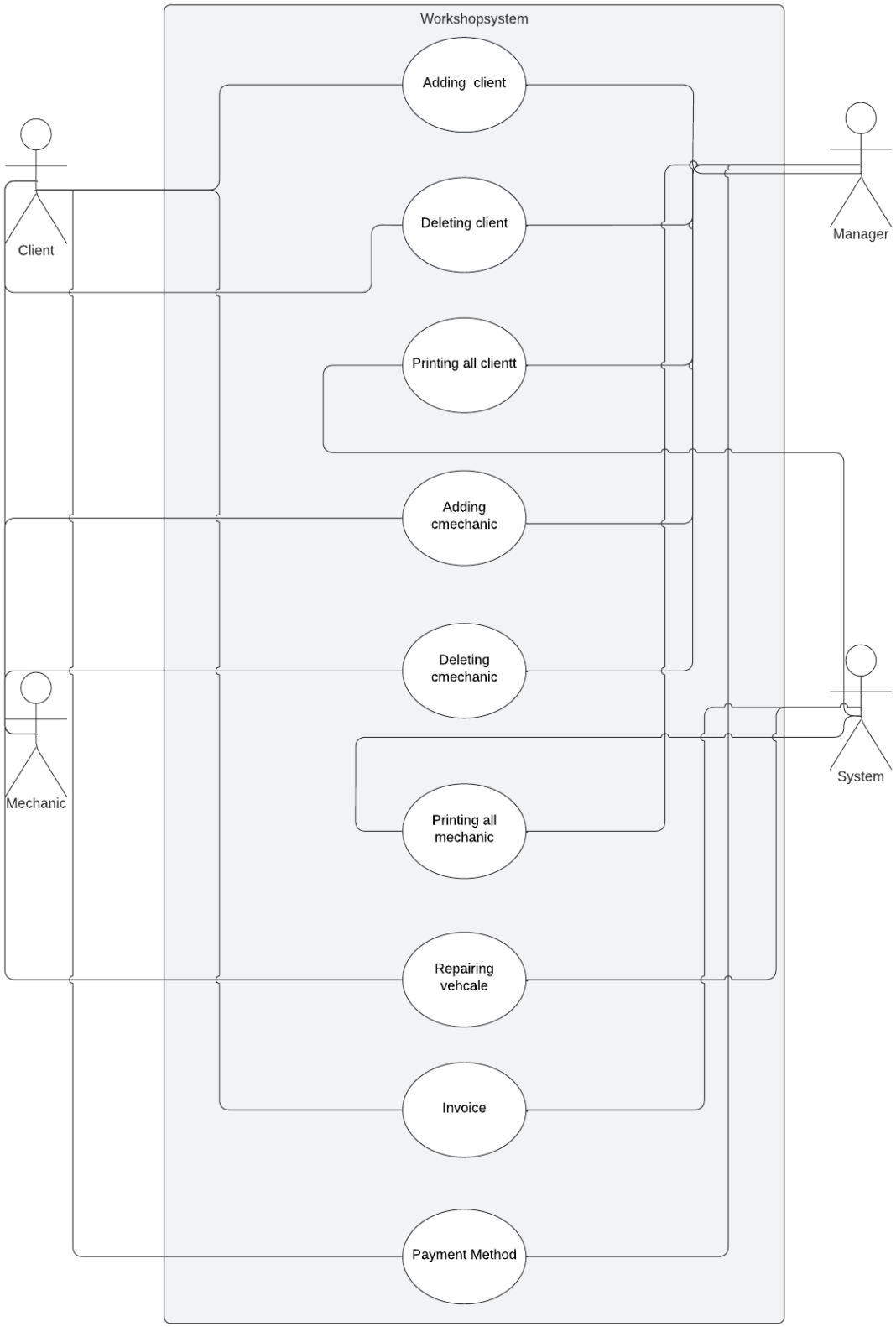
13.2.1 Functional requirements:

- Only the receptionist has the authority to make changes to the database's information.
- The system should collect all of a new client's information and register him by putting his information in the database.
- Do not recreate the client if one already exists.
- The system must provide the user with all the information about their client's appointments.

13.2.2 Non-Functional requirements:

- All the users of the software must be connected to an active internet connection.
- The system shall be available to all clients and managers.
- System's information must be secured.
- Python Idle is required to run this software.
- The system shall be fully operational and in working condition.
- The software should take adequate storage space depending on the number of clients.
- The system shall allow users to book appointments quickly.

14. Use Case diagram



Automating Workshop Software

15. Fully dressed Use Cases

15.1 Adding client

ACTORS, GOALS AND USE CASE BRIEFS

Actors	Task-level Goal	Priority	Brief
Secondary Actor is Receptionist in the Workshop and Primary Actor is the Client.	Goal is to add the new client in the database	High	Client will ask receptionist to add his details for Workshop. Receptionist will use program to Add this client in the database.

USE CASE SPECIFICATIONS

Use Case Identification and History			
Use Case ID:	PRJWRKSP 1.0		
Use Case Name:	Adding a Client	**Version No:**	
End Objective:	This will add the client into the database.		
Created by:	Rafay	**On (date):**	
Last Update by:		**On (date):**	
Approved by:		**On (date):**	
User/Actor:	Receptionist		
Business Owner Name:	Workshop Owner Fahad	**Contact Details:**	
Trigger:	It is triggered by clicking add client in Client section		
Frequency of Use:	3-5 Times a week		

Preconditions
User clicks Clients in main menu and systems takes user to Client's section

Basic Flow:

- **User initializes the program which takes user to a welcome screen and after that main menu will appear from where use selects 'Clients' option and then system will take user to Clients section. User selects the option of 'Add Client' in Client section. After adding a client program will prompt a message "Client has been successfully added" and take user back to main menu.**

Step	User Actions	System Actions
1	User initializes the program	System prompt with Welcome message and then displays the main menu.
2	User selects the option Client	System takes user to Client section
3	User clicks Add Client	System takes user to Add Client section
4	User types the client details	System stores client data in database and prompt the message "Client has been successfully added"

Alternate Flow		
Step	**User Actions**	**System Actions**
3.1	User clicks Client List	System prints the List of Clients
3.2	User clicks Delete Client	System takes user to Delete Client section to delete the desired client.

Exception Flow		
1	User leaves any slot empty	System prompts the message "Incomplete detail" and take back to client data input section.

Post conditions
System prompts message and take user back to main menu.

Includes or Extension Points

Special Requirements

Business Rules

Other Notes (Assumptions, Issues,)
Assumptions: N/A • **Issues: N/A**

15.2 Printing client list

ACTORS, GOALS AND USE CASE BRIEFS

Actors	Task-level Goal	Priority	Brief
Actor is Receptionist in the Workshop.	Goal is to print the client list from database	Medium	Receptionist will print the client list from the internal database

USE CASE SPECIFICATIONS

Use Case Identification and History			
Use Case ID:	PRJWRKSP 1.1		
Use Case Name:	Printing a client list	**Version No:**	
End Objective:	This will print the client list form the internal database		
Created by:	Rafay	**On (date):**	
Last Update by:		**On (date):**	
Approved by:		**On (date):**	
User/Actor:	Receptionist		
Business Owner	Workshop Owner	**Contact**	

Name:	Fahad	Details:	
Trigger:	It is triggered by clicking Client List in Clients section		
Frequency of Use:	Once a Month		

Preconditions
User clicks Clients in main menu and systems takes user to Clients section

Basic Flow:

1. **User initializes the program which takes user to a welcome screen and after that main menu will appear from where use selects 'Clients' option and then system will take user to Clients section. User selects the option of 'Client List' in Client section. Systems prints the Client List and take user back to main menu.**

Step	User Actions	System Actions
1	User initializes the program	System prompt with Welcome message and then displays the main menu.
2	User selects the option Client	System takes user to Client section
3	User clicks Client List	System prints the Client List from the internal database

Alternate Flow		
Step	**User Actions**	**System Actions**
3.1	User clicks Add Client	System takes user to Add Client section
3.2	User clicks Delete Client	System takes user to Delete Client section to delete the desired client.

Exception Flow		
1	User command to print Client List without any Client in database	System prompts the message of 'Empty Client List'

Post conditions

System prints the Client List and take user back to main menu.

Includes or Extension Points

Special Requirements

Business Rules

Other Notes (Assumptions, Issues,)
Assumptions: N/A **Issues: N/A**

15.3 Deleting client

ACTORS, GOALS AND USE CASE BRIEFS

Actors	**Task-level Goal**	**Priority**	**Brief**
Secondary Actor is Receptionist in the Workshop and Primary Actor is the Client.	Goal is deleting the existing client in the database	Low	Receptionist deletes the desired client from the database

USE CASE SPECIFICATIONS

Use Case Identification and History

Use Case ID:	PRJWRKSP 1.2		
Use Case Name:	Deleting a Client	**Version No:**	
End Objective:	This will delete the client into the database.		
Created by:	Rafay	**On (date):**	
Last Update by:		**On (date):**	
Approved by:		**On (date):**	
User/Actor:	Receptionist		
Business Owner Name:	Workshop Owner Fahad	**Contact Details:**	
Trigger:	It is triggered by clicking Delete client in Client section		
Frequency of Use:	5-10 times a year		

Preconditions
User clicks Clients in main menu and systems takes user to Clients section

Basic Flow:

1. **User initializes the program which takes user to a welcome screen and after that main menu will appear from where use selects 'Clients' option and then system will take user to Clients section. User selects the option of 'Delete Client' in Client section. After deleting a client program will prompt a message "Client has been successfully deleted" and take user back to main menu.**

Step	User Actions	System Actions
1	User initializes the program	System prompt with Welcome message and then displays the main menu.
2	User selects the option Client	System takes user to Client section
3	User clicks Delete Client	System takes user to Delete Client section
4	User selects the desired client	System deletes client data in database and prompt the message "Client has been successfully deleted"

Alternate Flow		
Step	**User Actions**	**System Actions**
3.1	User clicks Client List	System prints the List of Clients
3.2	User clicks Add Client	System takes user to Add Client section

Exception Flow		
1	User command to delete empty list	System prompts the message “Empty List” and take back to client data input section.

Post conditions
System prompts message and take user back to main menu.

Includes or Extension Points

Special Requirements

Business Rules

Other Notes (Assumptions, Issues,)
Assumptions: N/A **Issues: N/A**

15.4 Adding mechanic

ACTORS, GOALS AND USE CASE BRIEFS

Actors	Task-level Goal	Priority	Brief
Secondary Actor is Receptionist in the Workshop and Primary Actor is the Mechanic.	Goal is to add the new Mechanic in the database	High	Mechanic will ask receptionist to add his details for Workshop. Receptionist will use program to Add this mechanic in the database.

USE CASE SPECIFICATIONS

Use Case Identification and History			
Use Case ID:	PRJWRKSP 2.0		
Use Case Name:	Adding a Mechanic	**Version No:**	
End Objective:	This will add the Mechanic into the database.		
Created by:	Rafay	**On (date):**	
Last Update by:		**On (date):**	
Approved by:		**On (date):**	
User/Actor:	Receptionist		
Business Owner Name:	Workshop Owner Fahad	**Contact Details:**	
Trigger:	It is triggered by clicking Add Mechanic in Mechanic section		
Frequency of Use:	Once a month		

Preconditions
User clicks Mechanic in main menu and systems takes user to Mechanic section

Basic Flow:

1. **User initializes the program which takes user to a welcome screen and after that main menu will appear from where use selects 'Mechanics' option and then system will take user to Mechanics section. User selects the option of 'Add Mechanic' in Mechanics section. After adding a Mechanic program will prompt a message "Mechanic has been successfully added" and take user back to main menu.**

Step	User Actions	System Actions
1	User initializes the program	System prompt with Welcome message and then displays the main menu.
2	User selects the option Mechanic	System takes user to Mechanic section
3	User clicks Add Mechanic	System takes user to Add Mechanic section
4	User types the Mechanic details	System stores Mechanic data in database and prompt the message "Mechanic has been successfully added"

Alternate Flow

Step	User Actions	System Actions
3.1	User clicks Mechanic List	System prints the List of Mechanic
3.2	User clicks Delete Mechanic	System takes user to Delete Mechanic section to delete the desired Mechanic

Exception Flow

1	User leaves any slot empty	System prompts the message "Incomplete detail" and take back to Mechanic data input section.

Post conditions

System prompts message and take user back to main menu.

Includes or Extension Points

Special Requirements

Business Rules

Other Notes (Assumptions, Issues,)
Assumptions: N/A **Issues: N/A**

15.5 Printing mechanic list

ACTORS, GOALS AND USE CASE BRIEFS

Actors	Task-level Goal	Priority	Brief
Actor is Receptionist in the Workshop and Secondary Actor is the internal database.	Goal is to print the Mechanic list from database	Medium	Receptionist will print the Mechanic list from the internal database

USE CASE SPECIFICATIONS

Use Case Identification and History			
Use Case ID:	PRJWRKSP 2.1		
Use Case Name:	Printing a Mechanic list	**Version No:**	
End Objective:	This will print the Mechanic list form the internal database		
Created by:	Rafay	**On (date):**	
Last Update by:		**On (date):**	

Approved by:		**On (date):**	
User/Actor:	Receptionist		
Business Owner Name:	Workshop Owner Fahad	**Contact Details:**	
Trigger:	It is triggered by clicking Mechanic List in Mechanic section		
Frequency of Use:	Once a Month		

Preconditions
User clicks Mechanic in main menu and systems takes user to Mechanic section

Basic Flow:

1. **User initializes the program which takes user to a welcome screen and after that main menu will appear from where use selects 'Clients' option. System takes user to Client section where user selects 'Mechanics List'. System prints the Mechanics list and take user back to main menu.**

Step	User Actions	System Actions
1	User initializes the program	System prompt with Welcome message and then displays the main menu.
2	User selects the option Mechanic	System takes user to Mechanic section
3	User clicks Mechanic List	System prints the Mechanic List from the internal database

Alternate Flow		
Step	**User Actions**	**System Actions**
3.1	User clicks Add Mechanic	System takes user to Add Mechanic section
3.2	User clicks Delete Mechanic	System takes user to Delete Mechanic section to delete the desired Mechanic

a.

Exception Flow

1	User command to print Mechanic List without any Mechanic in database	System prompts the message of 'Empty Mechanic List'

Post conditions
System prints the Mechanic List and take user back to main menu.

Includes or Extension Points

Special Requirements

Business Rules

Other Notes (Assumptions, Issues,)
Assumptions: N/A **Issues: N/A**

15.6 Deleting mechanic

ACTORS, GOALS AND USE CASE BRIEFS

Actors	**Task-level Goal**	**Priority**	**Brief**
Actor is Receptionist in the Workshop and Secondary Actor is the internal	Goal is deleting the existing Mechanic in the database	Low	Receptionist deletes the desired Mechanic from the database

database			

USE CASE SPECIFICATIONS

Use Case Identification and History			
Use Case ID:	PRJWRKSP 2.2		
Use Case Name:	Deleting a Mechanic	**Version No:**	
End Objective:	This will delete the Mechanic into the database.		
Created by:	Rafay	**On (date):**	
Last Update by:		**On (date):**	
Approved by:		**On (date):**	
User/Actor:	Receptionist		
Business Owner Name:	Workshop Owner Fahad	**Contact Details:**	
Trigger:	It is triggered by clicking Delete Mechanic in Mechanic section		
Frequency of Use:	5-10 times a year		

Preconditions
User clicks Mechanic in main menu and systems takes user to Mechanic section

Basic Flow:

1. **User initializes the program which takes user to a welcome screen and after that main menu will appear from where use selects 'Mechanics' option and then system will take user to Mechanics section. User selects the option of 'Delete Mechanics' in Mechanics section. After deleting a Mechanic program will prompt a message "Mechanic has been successfully deleted" and take user back to main menu.**

Step	User Actions	System Actions
1	User initializes the program	System prompt with Welcome message and then displays the main menu.

2	User selects the option Mechanics	System takes user to Mechanics section
3	User clicks Delete Mechanic	System takes user to Delete Mechanic section
4	User selects the desired Mechanic	System deletes Mechanic data in database and prompt the message “Mechanic has been successfully deleted”

Alternate Flow		
Step	**User Actions**	**System Actions**
3.1	User clicks Mechanic List	System prints the List of Mechanics
3.2	User clicks Add Mechanic	System takes user to Add Mechanic section

Exception Flow		
1	User command to delete empty list	System prompts the message “Empty List” and take back to Mechanic data input section.

Post conditions
System prompts message and take user back to main menu.

Includes or Extension Points

Special Requirements

Business Rules

Other Notes (Assumptions, Issues,)
Assumptions: N/A **Issues: N/A**

15.7 Repairing vehicle

ACTORS, GOALS AND USE CASE BRIEFS

Actors	Task-level Goal	Priority	Brief
Actor is Receptionist in the Workshop and Primary Actor is the Client.	Goal is to repair the client's Vehicle	High	Client will ask receptionist to repair his vehicle in Workshop. Receptionist will ask the issues and inform mechanic about that issues.

USE CASE SPECIFICATIONS

Use Case Identification and History			
Use Case ID:	PRJWRKSP 3.0		
Use Case Name:	Repairing a Vehicle	**Version No:**	
End Objective:	This will gather the info about issue of vehicle and send the details to desired mechanic		
Created by:	Rafay	**On (date):**	
Last Update by:		**On (date):**	
Approved by:		**On (date):**	
User/Actor:	Receptionist		
Business Owner Name:	Workshop Owner Fahad	**Contact Details:**	
Trigger:	It is triggered by clicking Repair in Main Menu section		

Frequency of Use:	20 times a week

Preconditions
User initializes the program and system takes user to main menu

Basic Flow:

1. **User initializes the program which takes user to a welcome screen and after that main menu will appear from where use selects 'Repair' option and then system will take user to Repair section. User selects the desired client from client list. After selecting a client program will take user to Repair section where user will gather info about issues and send it to Mechanic.**

Step	User Actions	System Actions
1	User initializes the program	System prompt with Welcome message and then displays the main menu.
2	User selects the option Repair	System prompts user with the list of client to select from
3	User selects client	System takes user to Repair section
4	User selects from types of issue that has to be resolved	System gathers the data and send it to the Mechanic

Alternate Flow		
Step	**User Actions**	**System Actions**
3.1	User clicks Oil Change	System gathers the info and send it to the Mechanic
3.2	User clicks Air Refill	System gathers the info and send it to the Mechanic

Exception Flow		
1	N/A	N/A

Post conditions

System prompt the message to take user to Invoice section

Includes or Extension Points

Special Requirements

Business Rules

Other Notes (Assumptions, Issues,)
Assumptions: N/A **Issues: N/A**

15.8 Invoice

ACTORS, GOALS AND USE CASE BRIEFS

Actors	**Task-level Goal**	**Priority**	**Brief**
Secondary Actor is Receptionist in the Workshop and Primary	Goal is to make an Invoice of the client's repair job	High	Client will ask receptionist to make invoice of the repair job. Receptionist will add the repair jobs in the invoice and calculate the total.

Actor is the Client.			

USE CASE SPECIFICATIONS

Use Case Identification and History			
Use Case ID:	PRJWRKSP 3.0		
Use Case Name:	Making an Invoice	**Version No:**	
End Objective:	This will gather the info about repair job and add it to Invoice with charges and Total Amount		
Created by:	Rafay	**On (date):**	
Last Update by:		**On (date):**	
Approved by:		**On (date):**	
User/Actor:	Receptionist		
Business Owner Name:	Workshop Owner Fahad	**Contact Details:**	
Trigger:	It is triggered by clicking Invoice in Main Menu section		
Frequency of Use:	20 times a week		

Preconditions
System sends the repair job info to mechanic

Basic Flow:

1. **User initializes the program which takes user to a welcome screen and after that main menu will appear from where use selects 'Repair' option and then system will take user to Repair section. User selects the issues in Repair section. After sending the info to the Mechanic system prompts to make an Invoice and takes user back to main menu from where user selects Invoice option. Systems gathers repair job info and its charges and calculate the total and create an invoice in specific format**

Step	User Actions	System Actions
1	User initializes the program	System prompt with Welcome message and then displays the main menu.
2	User selects the option Repair	System prompts user with the list of client to select from
3	User selects client	System takes user to Repair section
4	User selects from types of issue that has to be resolved	System gathers the data and send it to the Mechanic
5	User clicks the option Make Invoice	System takes user back to Main Menu
6	User selects Invoice	System gathers info about repair jobs and its charges and total bill and create and invoice in specific format

Alternate Flow		
Step	**User Actions**	**System Actions**
	N/A	N/A
	N/A	N/A

Exception Flow		
1	N/A	N/A

Post conditions
System prompt to select the Payment Method

Includes or Extension Points

Special Requirements

Business Rules

Other Notes (Assumptions, Issues,)
Assumptions: N/A **Issues: N/A**

15.9 Payment method

ACTORS, GOALS AND USE CASE BRIEFS

Actors	**Task-level Goal**	**Priority**	**Brief**
Secondary Actor is Receptionist in the Workshop and Primary Actor is the Client.	Goal is to select the payment method the client desires	High	Receptionist will ask client for desired payment method and add it to the invoice

USE CASE SPECIFICATIONS

Use Case Identification and History			
Use Case ID:	PRJWRKSP 4.0		
Use Case Name:	Payment Method	**Version No:**	
End Objective:	This will select the payment method and add it to the invoice		
Created by:	Haris	**On (date):**	
Last Update by:	Rafay	**On (date):**	

Approved by:		**On (date):**	
User/Actor:	Receptionist		
Business Owner Name:	Workshop Owner Fahad	**Contact Details:**	
Trigger:	It is triggered by clicking Payment Option in Invoice section		
Frequency of Use:	20 times a week		

Preconditions
System creates an Invoice

Basic Flow:

1. **User initializes the program which takes user to a welcome screen and after that main menu will appear from where use selects 'Repair' option and then system will take user to Repair section. User selects the issues in Repair section. After sending the info to the Mechanic system prompts to make an Invoice and takes user back to main menu from where user selects Invoice option. Systems gathers repair job info and its charges and calculate the total and create an invoice in specific format. System then prompt for Payment Method. User selects the payment method as the client desires.**

Step	User Actions	System Actions
1	User initializes the program	System prompt with Welcome message and then displays the main menu.
2	User selects the option Repair	System prompts user with the list of client to select from
3	User selects client	System takes user to Repair section
4	User selects from types of issue that has to be resolved	System gathers the data and send it to the Mechanic
5	User clicks the option Make Invoice	System takes user back to Main Menu
6	User selects Invoice	System gathers info about repair jobs and its charges and total bill and create and invoice in specific format and prompt to select Payment Method

7	User selects Payment Method Option	System add the Payment Method option to the Invoice

Alternate Flow		
Step	**User Actions**	**System Actions**
7.1	User selects Hard Cash	System add "Payment Via Hard Cash" to the Invoice
7.2	User selects Online Transfer	Systems Prompt with Workshop AC Number.

Exception Flow		
1	N/A	N/A

Post conditions
System prompt to Print the Invoice

Includes or Extension Points

Special Requirements

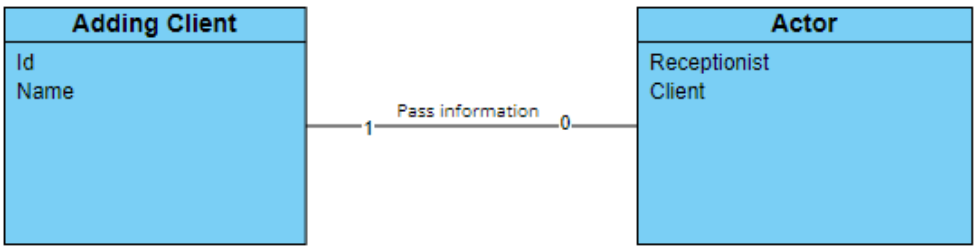
Business Rules

Other Notes (Assumptions, Issues,)
Assumptions: N/A

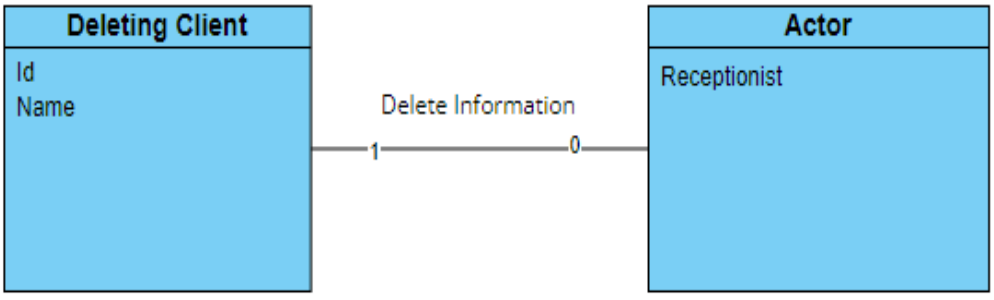
- **Issues: Should there be PayPal Option?**

16. Domain Model

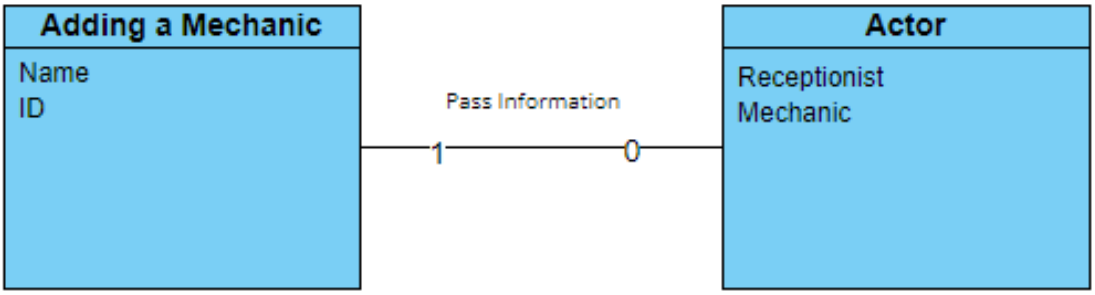
16.1 Adding A Client



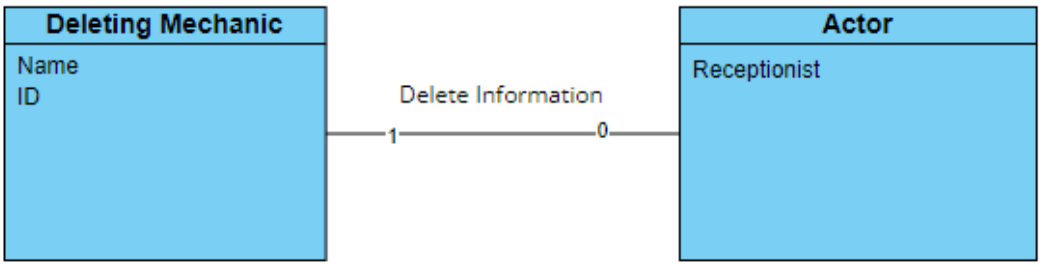
16.2 Deleting A Client



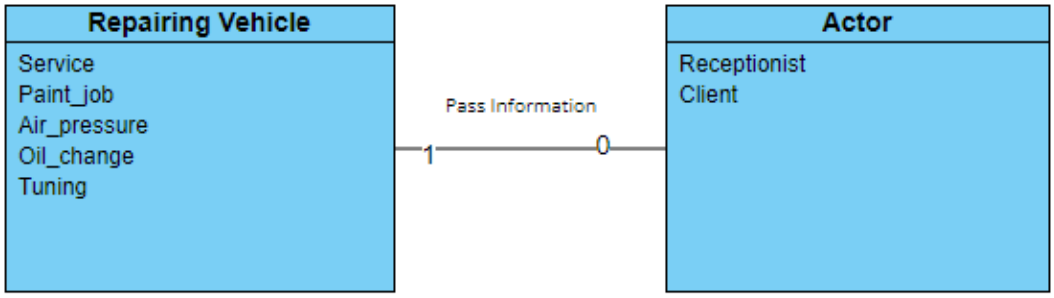
16.3 Adding a Mechanic



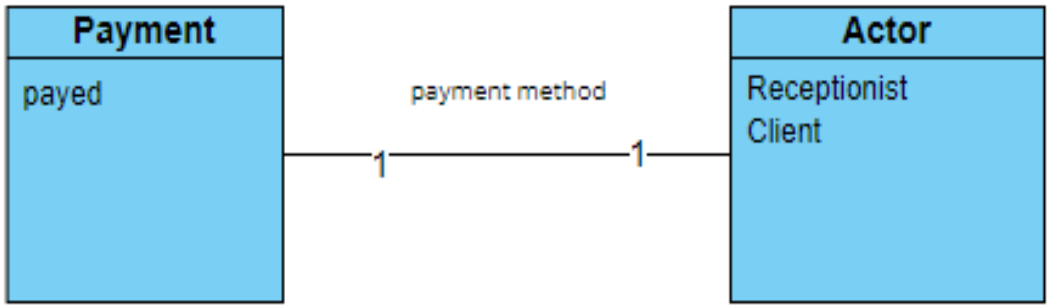
16.4 Deleting a Mechanic



16.5 Repairing Vehicle

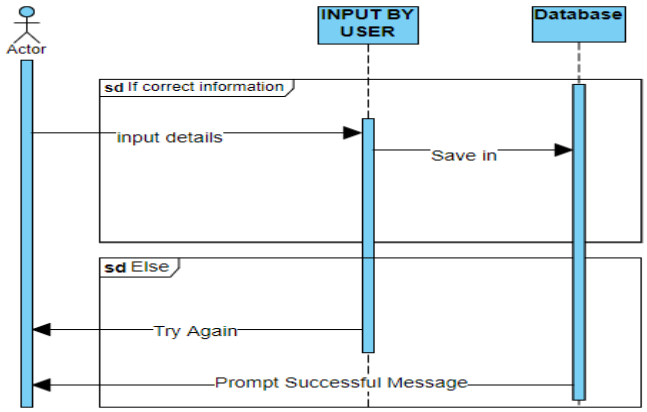


16.6 Payment

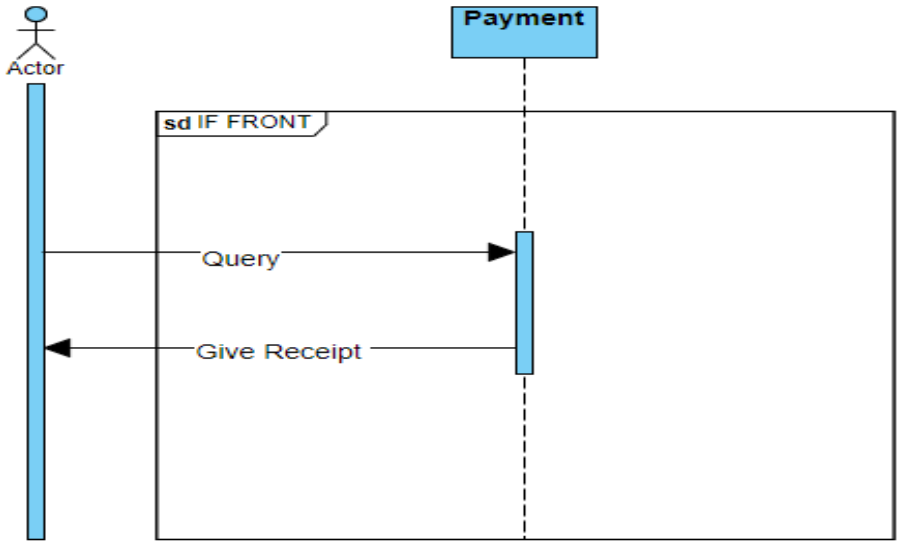


17. Sequence diagram

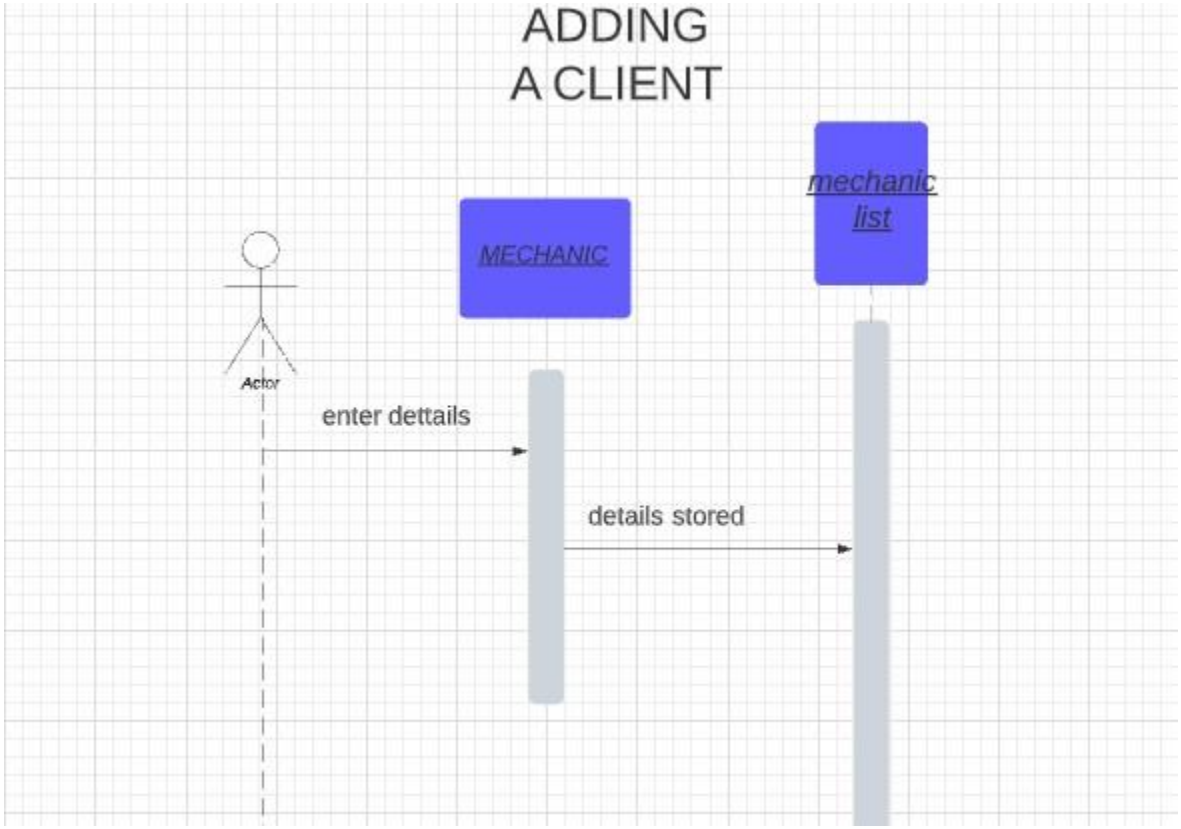
17.1 Adding a client



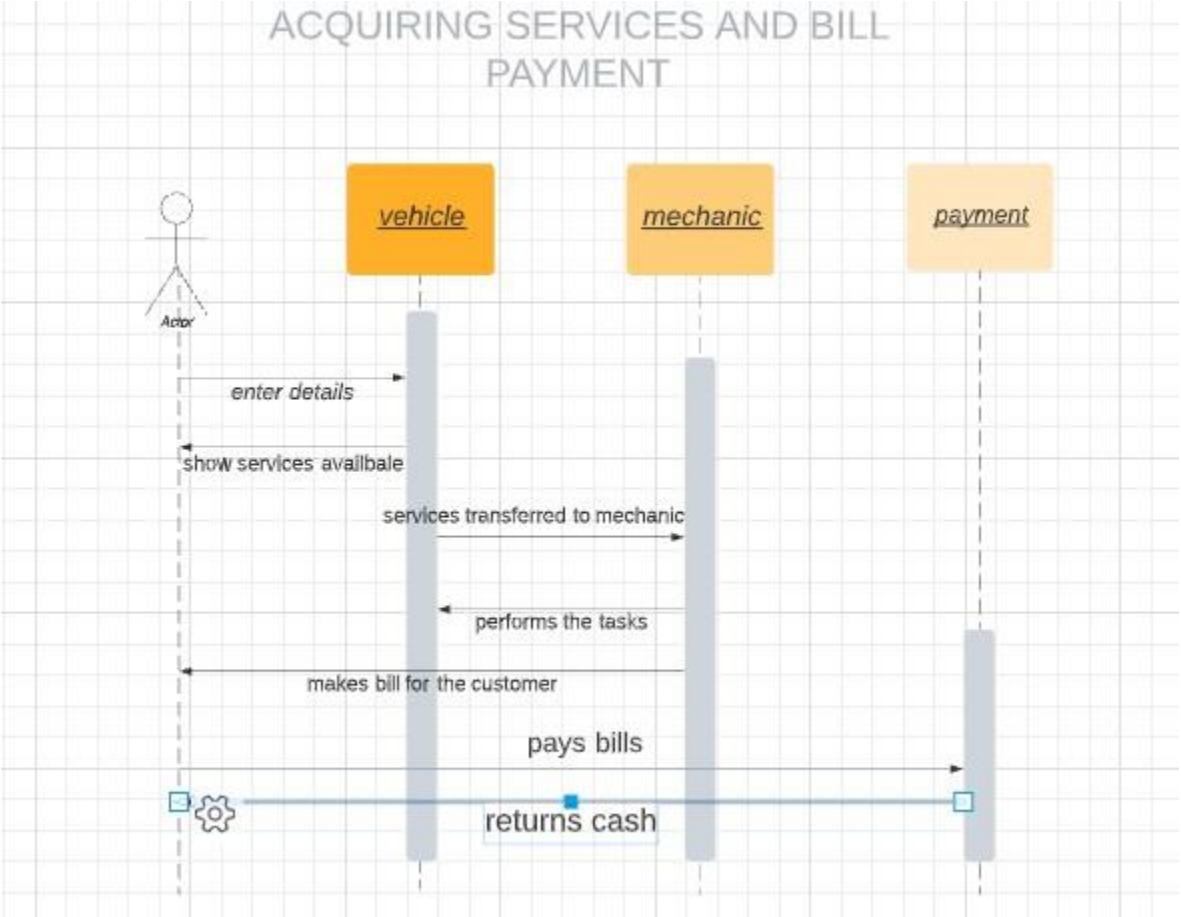
17.2 Payment



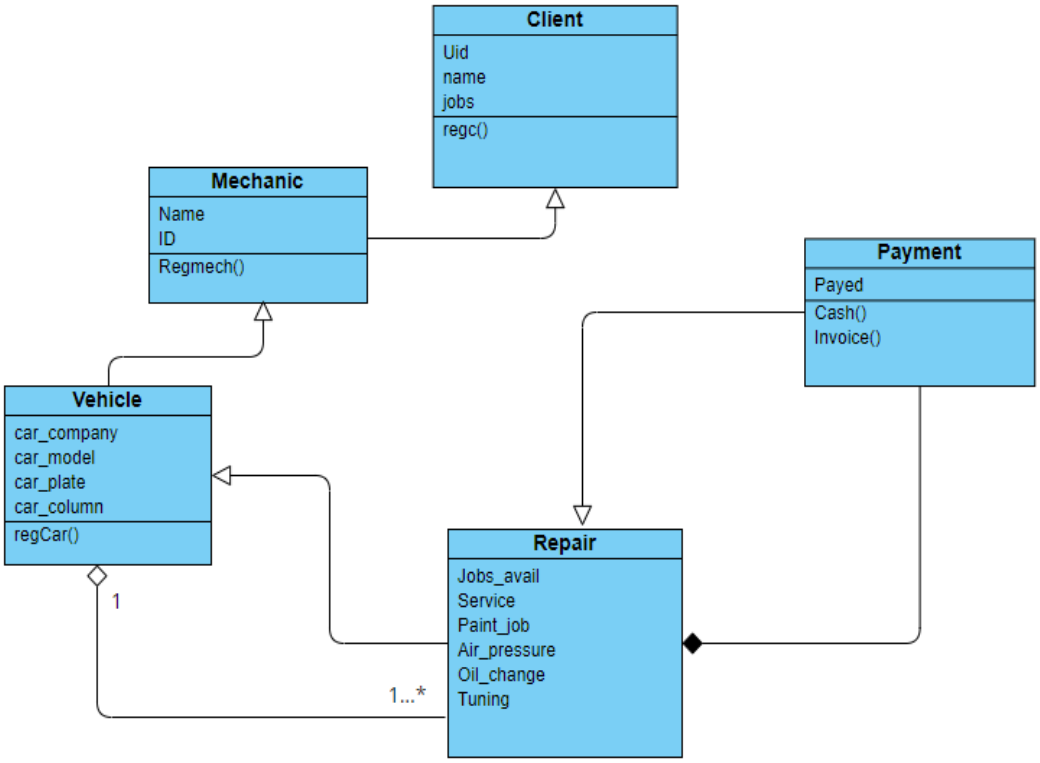
17.3 Adding mechanic



17.4 Services and payment

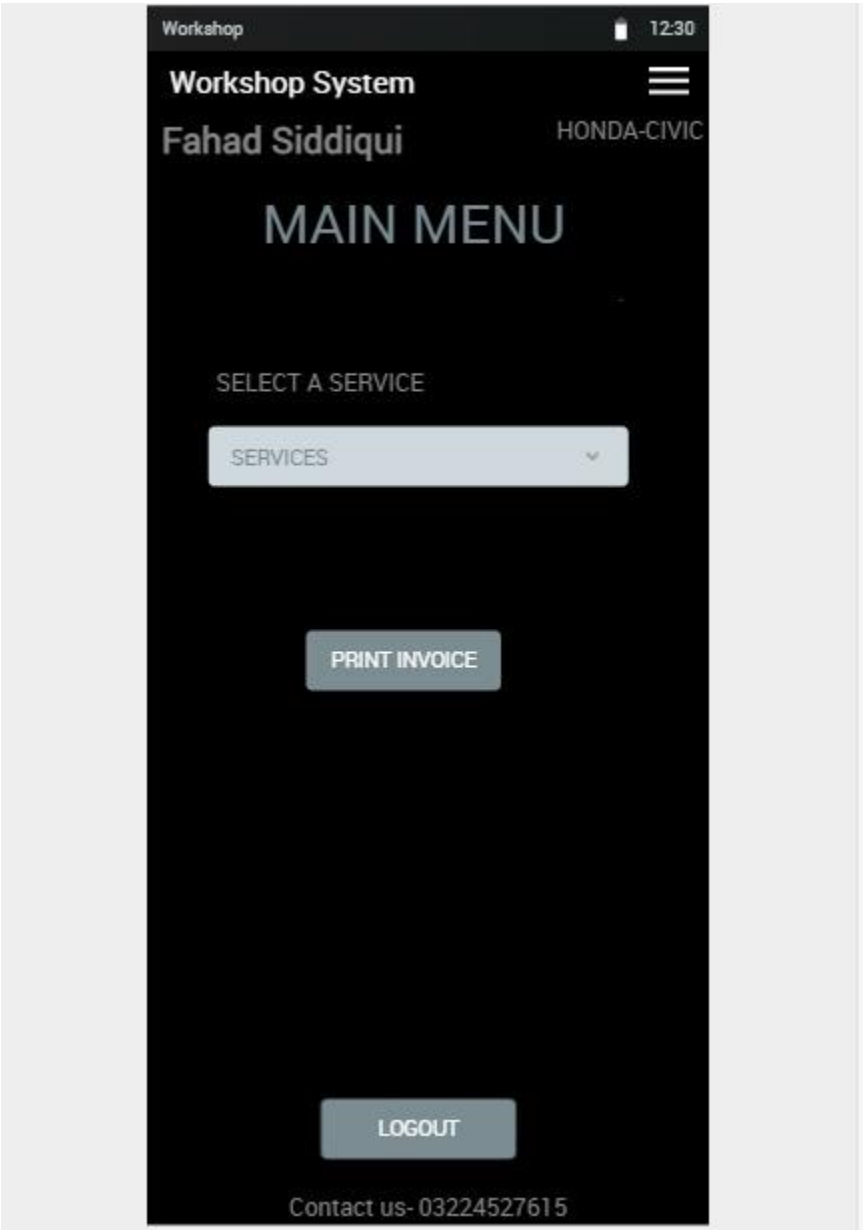


18. Class diagram

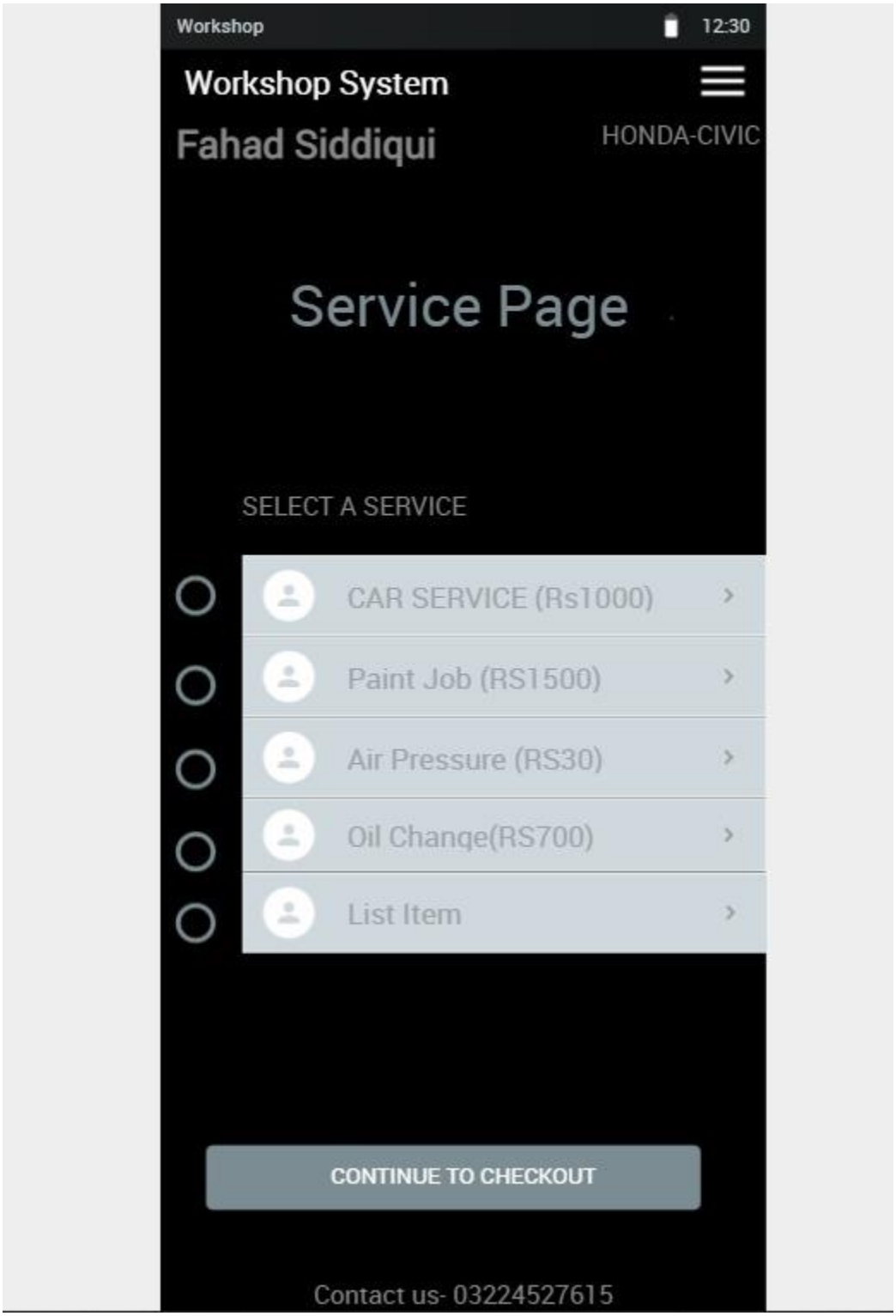


19. User Interface

In these use cases at the top information about the client will be displayed for example their name of the vehicle they're using.



This UI basically Mount this when you have selected the customer or client menu, this would allow the customer to choose the services that are being provided. the customer can choose from the dropdown menu when you click the services option it would display the services being provided at the workshop. You can also print the invoice if you already have selected the services that you want. a logout button also exist at the bottom if the does not want to acquire any services from the workshop.



Once you have selected the service option it would display the number of services being provided with the charges off the workshop you can check the circle if you want to acquire a

service the total bill of all the services that you have selected will be added in your invoice and when you continue to check out it would have your invoice ready.

20. Test cases

20.1 TEST CASE 1:

Test Case ID:	100 a
Test Case Name:	Adding client
Test Case Priority:	High
Test Case Pre-requisite:	User should click Clients in main menu.
Test Case Post-requisite:	System prompts message and take user back to main menu.

Sr No	Action	Input	Expected Output	Actual Output	Result	Comment
1	User clicks Add Client. Users type the Name, ph-no, address.	Rafay 022920239302 Gulberge 3, Lahore Pakistan.	Client will be added into database.	Client is be added into database.	Add client and takes user to Client section.	The system should prompt a welcome message.

20.2 TEST CASE 2:

Test Case ID:	101 a
Test Case Name:	Deleting client
Test Case Priority:	Low
Test Case Pre-requisite:	User clicks Clients in main menu and should have client in database.
Test Case Post-requisite:	This will delete the client from database.

Sr No	Action	Input	Expected Output	Actual Output	Result	Comment
2	User clicks deleting Client. User selects the desired client or	Ammar	Selected Client will be deleted from database.	Selected Client is be deleted.	Delete client and takes user to Client section.	The system should prompt a GOODBYE message.

	enter Name to search.					

20.3 TEST CASE 3:

Test Case ID:	102 a
Test Case Name:	Printing all clients
Test Case Priority:	Medium
Test Case Pre-requisite:	User clicks Clients in main menu
Test Case Post-requisite:	Print the client list from database

Sr No	Action	Input	Expected Output	Actual Output	Result	Comment
3	User clicks printing Clients.	No such input	Print the client list from database	All clients in database are printed.	List of clients printed.	The system should print all client name according to time earliest first.

20.4 TEST CASE 4:

Test Case ID:	103 a
Test Case Name:	Adding mechanic
Test Case Priority:	High
Test Case Pre-requisite:	Clicks Mechanic in menu and systems takes user to Mechanic section.
Test Case Post-requisite:	System prompts message and take user back to main menu.

Sr No	Action	Input	Expected Output	Actual Output	Result	Comment
4	User clicks Add Mechanic. Users type the Name, ph-no, address, salary.	Ahmed 022920239302 Gulberge 3, Lahore Pakistan 10000.	Mechanic will be added into database.	Mechanic is be added into database.	Add Mechanic and takes user to Mechanic section.	The system should prompt a welcome message.

20.5 TEST CASE 5:

Test Case ID:	104 a
Test Case Name:	Deleting Mechanic
Test Case Priority:	Low
Test Case Pre-requisite:	Clicks Mechanic in main menu and systems takes user to Mechanic section
Test Case Post-requisite:	This will delete the Mechanic from database.

Sr No	Action	Input	Expected Output	Actual Output	Result	Comment
5	User clicks deleting Mechanic. User selects the desired Mechanic or enter Name to search.	Zain	Selected mechanic will be deleted from database.	Selected Mechanic is deleted.	Delete mechanic and takes user to Mechanic section.	The system should prompt a best wishes message.

20.6 TEST CASE 6:

Test Case ID:	105 a
Test Case Name:	Printing all mechanic
Test Case Priority:	Medium
Test Case Pre-requisite:	User clicks mechanic in main menu
Test Case Post-requisite:	Print the client list from database

Sr No	Action	Input	Expected Output	Actual Output	Result	Comment
6	User clicks printing mechanic.	No such input	Print the mechanics list from database	All mechanic in database is printed.	List of mechanic printed.	The system should print all client name according to highest salary.

20.7 TEST CASE 7:

Test Case ID:	106 a
Test Case Name:	Repairing vehicle

Test Case Priority:	High
Test Case Pre-requisite:	User initializes the program and system takes user to main menu
Test Case Post-requisite:	System prompt the message to take user to Invoice section

Sr No	Action	Input	Expected Output	Actual Output	Result	Comment
7	User selects the option Repair User selects client User selects from types of issue that must be resolved	Rafay Oil change	Desired services of vehicle should send mechanics.	The info about issue of vehicle is sent to desired mechanic.	Take user to Invoice section.	System prompts the message that "Your request is sent to mechanic".

20.8 TEST CASE 8:

Test Case ID:	107 a
Test Case Name:	Payment method
Test Case Priority:	High
Test Case Pre-requisite:	System should create an Invoice
Test Case Post-requisite:	System prompt to Print the Invoice

Sr No	Action	Input	Expected Output	Actual Output	Result	Comment
8	User selects Hard Cash. User selects Online Transfer	Hard cash	Show invoice	Invoice is printed.	Cash is received and data enter to database.	It should prompt a good wishes message.

21. Future Work

- Clients can call mechanics at their home for repairing their cars
- Payment to mechanics can be observed
- More services can be added to the services option
- Selling and buying of cars and spare parts can be added
- Loyalty discounts for oldest customers

22. Conclusion

This app would help the client and the workshop staff a lot. Would not require customers to wait for their turn and would make it impossible for any mechanic to do any frauds such as taking extra money for a service. Hope this software makes the life of workshop owner and customer coming to the workshop. The owner can keep a record of all his customers.

23. Code

```
def Printmainmenu():
    print("Welcome to the Workshop system"+'\n'+"Choose Menu:"+'\n'+'1. Client'+'\n'+'2. Mechanic'+'\n'+"3. Exit")


def Printclientmenu():
    print("Welcome to the Client menu."+'\n'+"Choose Option:"+'\n'+"1. Enter Client data"+'\n'+"2. Services"+'\n'+"3. Print invoice"+'\n'+"4. Go back to previous menu")


def Printmechmenu():
    print("Welcome to the Mechanic menu."+'\n'+"Choose Option:"+'\n'+"1. Enter new mechanic"+'\n'+"2. List of mechanics"+'\n'+"3. Go back to previous menu")


def Printservices():
    print("")


def Printservicemenu():
     print("Welcome to the Services menu."+'\n'+"Choose Option:"+'\n'+"1. Car service    1000"+'\n'+"2. Paint Job      1500"+'\n'+"3. Air Pressure    30"+'\n'+"4. Oil change      700"+'\n'+"5. Tuning      500"+'\n'+"6. Return")


Mechanic_list= {"1" : "Ahmed", "2" : "Ali"}
Client_list= {}


def Workshop():
```

```
Endrun = False
while not Endrun:

    Printmainmenu()

    x=int(input())
    new=Payment()
    new2=Mechanic()
    JL= new.Jobs_avail



    if x==1:
        clientmenudone = False
        while not clientmenudone:
            Printclientmenu()
            y=int(input())
            if y==1:

                candatadone = False
                while not candatadone:
                    new.regc()
                    new.regcar()
                    candatadone = True

            if y==2:
                jobdone= False
                while not jobdone:
                    Printservicemenu()
                    o= int(input())

                    if o==1:
```

```
                new.jobs.append("Car service")
                new.total+= JL["Service"]

            if o==2:
                new.jobs.append("Paint Job")
                new.total+= JL["Paint Job"]
            if o==3:
                new.jobs.append("Air Pressure")
                new.total+= JL["Air Pressure"]
            if o==4:
                new.jobs.append("Oil Change")
                new.total+= JL["Oil Change"]
            if o==5:
                new.jobs.append("Tuning")
                new.total+= JL["Tuning"]
            if o==6:
                jobdone = True

        if y==3:
            invodone= False
            while not invodone:
                new.Invoice()
                invodone = True

        if y==4:
            clientmenudone = True

if x==2:
    mechmenudone= False
    while not mechmenudone:
        Printmechmenu()
```

```
            v=int(input())
            if v==1:
                mechdatadone = False
                while not mechdatadone:
                    new.Regmech()
                    mechdatadone = True
            if v==2:
                print(Mechanic_list)
            if v==3:
                mechmenudone = True

        if x==3:
            Endrun = True

class Client:
    def __init__(self):
        self.Uid=0
        self.name=''
        self.total = 0
        self.jobs=[]

    def regc(self):
        self.Uid+=1
        print(self.Uid)
        print("Enter name :")
        self.name=input()


class Mechanic(Client):
    def __init__(self):
        Client.__init__(self)
```

```
        self.Name=''
        self.ID=2


    def Regmech(self):
        self.ID+=1
        print(self.ID)
        print("Enter name :")
        self.Name=input()
        Mechanic_list[self.ID]=self.Name
        print(Mechanic_list)

class Vehicle(Mechanic):
    def __init__(self):
        Mechanic.__init__(self)
        self.car_company= ''
        self.car_model=''
        self.car_plate=''
        self.car_colour=''


    def regcar(self):
        print("Enter Car Company :")
        self.car_company=input()
        print("Enter Car model :")
        self.car_model=input()
        print("Enter Car No.plate :")
        self.car_plate=input()
        print("Enter Car colour :")
        self.car_colour= input()

Client_list[self.Uid]=self.name,self.car_company,self.car_model,self.car_plate,self.car_colour,self.total
```

```
        print(Client_list)


class Repair(Vehicle):
    def __init__(self):
        Vehicle.__init__(self)
        self.Jobs_avail= {"Service" : 1000, "Paint Job": 1500, "Air Pressure": 30, "Oil Change": 700,
"Tuning": 500}
        self.Service= 1000
        self.Paint_job=1500
        self.Air_pressure= 30
        self.Oil_change= 700
        self.Tuning= 500


class Payment(Repair):
    def __init__(self):
        Repair.__init__(self)
        self.Payed= 0


    def Cash(self):
        change= 0
        Payed= 0
        total= self.total
        print("Amount Payed =")
        x=int(input())
        if x > total:
            change= x-total
            print("Your change = ",change,'\n',"Thank You.")


        if x== total:
            print("No change."+'\n'+"Thank You.")
        if x < total:
            amount_due=total-x
```

```
        print("Amount due= ",amount_due,'\n',"Please, pay the remaining amount")
        self.total=amount_due


    def Invoice(self):
        print("INVOICE",'\n',"Client name:",",self.name,'\n',"Jobs Done:",'\n',self.jobs,'\n','\n',"Total = ",
self.total)
        print("Please select mode of payment:"+'\n'+"Choose Option:"+'\n'+"1. Cash"+'\n'+"2.
EasyPaisa"+'\n'+"3. Debit")
        y=int((input()))
        if y==1:
            self.Cash()
        elif y==2:
            print("Thank You.")


        elif y==3:
            print("Thank You.")


Workshop()
```