



Sri Lanka Institute of Information Technology

Vehicle Repair Management System

Project Report

Information Technology Project 2021

Project ID: ITP2021_S1_02_G7

Submitted by:

1. IT19968216 -Balasooriya D.P.K.D
2. IT19961590-Dilmika B.G.N
3. IT1966618-Siriwardana H.D.T.H
4. IT19966922-Wimukthi H.R.Y
5. IT19958620-Wickramasinghe T.L
6. IT19960364-Madushanka G.T
7. IT19962580-Sansala M.G.N
8. IT19970332-Priyal N.C.I

Submitted Date :

30.05.20

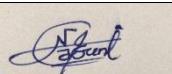
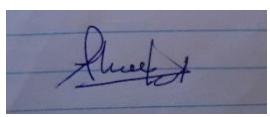
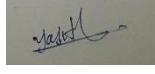
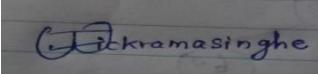
Declaration

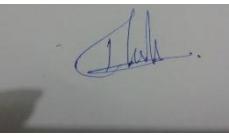
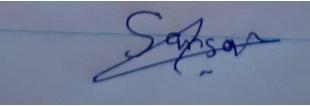
We declare that this project report or part of it was not a copy of a document done by any organization, university any other institute or a previous student project group at SLIIT and was not copied from the Internet or other sources.

Project Details

Project Title	RASA vehicle repair service management system
Project ID	ITP2021_S1_02_G7

Group Members

Reg. No	Name	Signature
IT19968216	Balasooriya D.P.K.D	
IT19961590	Dilmika B.G.N	
IT19966618	Siriwardhane H.D.T.H	
IT19966922	Wimukthi H.R.Y	
IT19958620	Wickramasinghe T.L	

IT19960364	Madushanka G.T	
IT19962580	Sansala M.G.N	
It19970332	Priyal N.C.I	

Abstract

The vehicle repair company "RASA" is based in Malabe, Sri Lanka. The company's job is to repair vehicles that come in to repair with or without an accident. The company does it for the customer when a vehicle arrives. The company runs and maintains its network manually and holds records to document the specifics of its inventory, service details, client data, and account details and also employee details, they still have no computerized program as it poses several problems such as data incoherence, data unreliability, and so on. The approach is a repetitive and inefficient way and could have resulted in a serious loss of data. As a client requirement, we proposed a vehicle repair management system for "RASA" company. Our team chose a set of features to build for this system after discussing it with the client and agreed to create a java-based web application as a response to the existing problems confronting the company. Our team has agreed to use Java as their programming languages and MySQL as a database tool for backend and HTML, CSS, JS, Sementic UI as frontend tools.

Acknowledgement

We express our sincere thanks firstly to our lecturers who stood as pillars behind our team and guide us to success. Our team acknowledge the indebtedness to our teaching staff who encourage us with their valuable guidance and kind supervision throughout every task of this project.

Special thanks for Ms. Janani Tharmaseelan, Ms. Archchana, Ms. Uthpala Samarakoon for given their useful time to explain our mistakes and educate about those errors.

We pay our deep sense of gratitude to the company that offered us this opportunity to build this project. Our team express our heartiest thanks to the company staff who supported us with providing relevant information that are crucial to build this process.

Lastly, we are thankful to all those who helped us in any way to make this project a successful outcome.

Table of Contents

Declaration	ii
Abstract	iii
Table of Contents	v
List of Figures	vi
List of Tables	ix
List of Acronyms and Abbreviations.....	x
1. Introduction	1
1.1 Problem Statement	1
1.2 Product Scope	2
1.3 Project Report Structure	2
2. Methodology.....	3
2.1 Requirements and Analysis	3
2.2 Design.....	19
2.3 Implementations.....	42
2.4 Testing	70
3. Conclusion	67
4. References.....	88
[1] Lecture 5 - Final Report.....	88
[2] IEEE Citation Guidelines.	88

List of Figures

<i>Figure 1:Logo-RASA</i>	1
<i>Figure 2: UD – employee details management</i>	4
<i>Figure 3: UD – employee payments management</i>	5
<i>Figure 4: UD –customer details management</i>	6
<i>Figure 5: UD –work progress management</i>	7
<i>Figure 6: UD –financial management</i>	8
<i>Figure 7: UD –car records management</i>	9
<i>Figure 8: UD –car rental management</i>	10
<i>Figure 9: AD – employee details management</i>	11
<i>Figure 10: AD – Employee payment management (attendance management)</i>	12
<i>Figure 11: AD – Employee payment management (salary management)</i>	13
<i>Figure 12: AD – Employee payment management (advance management)</i>	14
<i>Figure 13: AD--customer details management</i>	15
<i>Figure 14: AD-- work progress management</i>	16
<i>Figure 15: AD financial management (payment management)</i>	17
<i>Figure 16: AD financial management (Budget management)</i>	18
<i>Figure 17: AD car records management</i>	19
<i>Figure 18: AD car rental management</i>	20
<i>Figure 19: High level architecture diagram</i>	21
<i>Figure 20: Class Diagram</i>	22
<i>Figure 21: Entity Relationship diagram</i>	23
<i>Figure 22: customer registration</i>	24
<i>Figure 23: vehicle registration</i>	24
<i>Figure 24: add service</i>	25
<i>Figure 25: search service entry</i>	25
<i>Figure 26: service entry page</i>	26
<i>Figure 27: work progress management System</i>	27
<i>Figure 28: add new service</i>	27
<i>Figure 29: assign employee</i>	28
<i>Figure 30: add repair components</i>	28

<i>Figure 31: estimate report overview</i>	29
<i>Figure 32: financial management</i>	29
<i>Figure 33: Latest payment</i>	30
<i>Figure 34: all payment</i>	31
<i>Figure 35: Budget UI</i>	31
<i>Figure 36: car record retrieve</i>	32
<i>Figure 37: car record register</i>	33
<i>Figure 38,39,40:car record edit</i>	34
<i>Figure 41:available cars</i>	35
<i>Figure 42:Rent form</i>	36
<i>Figure 43:Rent Details</i>	37
<i>Figure 44:update rental details</i>	38
<i>Figure 45:employee registration</i>	39
<i>Figure 46:employee details</i>	40
<i>Figure 47:employee update form</i>	41
<i>Figure 48: employee profile</i>	42
<i>Figure 49: employee salary and attendance management</i>	43
<i>Figure 50: latest advance record management</i>	43
<i>Figure 51:Add Advance</i>	44
<i>Figure 52: customer class</i>	45
<i>Figure 53: Employee class</i>	45
<i>Figure 54 Employee advance Class:</i>	46
<i>Figure 55:repair class</i>	46
<i>Figure 56: work progress class</i>	47
<i>Figure 57:Vehcile class</i>	47
<i>Figure 58 payment class:</i>	48
<i>Figure 59:Budget class</i>	48
<i>Figure 60 :client service</i>	49
<i>Figure 61:employee service</i>	49
<i>Figure 62: Employee Advance service</i>	50
<i>Figure 63:workprogress service</i>	50
<i>Figure 64:Vehicle service</i>	51
<i>Figure 65:service entry service</i>	51
<i>Figure 66:budget service</i>	52
<i>Figure 67:payment service</i>	52

<i>Figure 68 add customer:</i>	53
<i>Figure 69:payement servlet</i>	53
<i>Figure 70:add service entry</i>	54
<i>Figure 71:create work progress</i>	54
<i>Figure 72:upload vehicle</i>	55
<i>Figure 73: get service details</i>	55
<i>Figure 74 Get budget amount :</i>	56
<i>Figure 75: Update payment</i>	56
<i>Figure 76:add payment</i>	57
<i>Figure 77:display latest payment</i>	57
<i>Figure 78:car record model</i>	58
<i>Figure 79:car record service</i>	58
<i>Figure 80:car record delete servlet</i>	59
<i>Figure 81:car record insert servlet</i>	60
<i>Figure 82:car record retrieve servlet</i>	60
<i>Figure 83:car record update edit servlet</i>	61
<i>Figure 84:car record update servlet</i>	62
<i>Figure 85:car record delete servlet</i>	62

List of Tables

Table 1: Testing- Add service _____ 65

List of Acronyms and Abbreviations

ER Entity Relationship

CD Class diagram

AD Activiti Diagram

UI User Interface

1. Introduction

1.1 Problem Statement

RASA is a car repair company that has been well known in the Kaduwela area for many years. It does all kinds of repair work. As a company, RASA wants to manage its employees' information and payments, inventory information such as vehicles coming into repair and their owners, distribution information, and auto parts. Also, you need to manage the registrations and payments related to the rental car service provided by the company. Finally, the profit should be compared with the income and expenses of the company and records should be kept for all the above. After discussing all these issues, our team came up with solutions to the problems faced by the company and a system that can provide them.



Figure 1:Logo

1.2 Product Scope

Web base application was developed by us for event planning company to deal with vehicle repairs, manage inner company works. This scope covers the accompanying capacities through online application. Such as Finance Management System, Customer Management System, Employee Management System, Service Management System, Inventory Management system, Work Progress Management System, Rent Management System. This scope gives an easy to use system and effective information preparing.

1.3 Project Report Structure

First Section:

As the first section it mainly focused on client's problems which they have while using manual system and client's requirements. Therefore, in this part gave the solutions for that problem and include the overall project plan and the scope of the project by us.

Second Section:

In this section focused on methodology part which contain the clarification of Requirement Analysis, Designs, Implementation, and the Testing. Use case Diagrams and the Activity Diagrams are used for describing Requirement Analysis part. The Design part is classified utilizing the guide of the ER diagram, Class Diagram, High-Level Architecture Diagram and the snapshots of user Interfaces. For the Implementation part there are Snapshots of the Implementations and finally there are test cases for the test areas.

Third Section:

This section indicates the information of evaluation and accomplishment of the whole project and the new knowledge which are learned through this project.

Forth Section:

This section indicates all the references which used to full fill this project.

Methodology

1.4 Requirements and Analysis

The Objective of the process is to convert a manual vehicle repair processes into an automated system. After a thorough analysis these manual process was divide into eight main functions. Which included Customer Details management, Work progress management, Financial management, Employee Details Management, Employee Payment Management, Car Records Management, Car Rental Management and Inventory Management. In previously this company use paper based system to handle documents. In this company documents are really important to handle. So, with the implementation of the system, the documents are electronically stored and can be downloaded as PDF. Furthermore, previously all the calculations were done by manually. But from this newly created system payment calculation and estimate amount calculations also automated. The aims of this system are to reduce the time that consume for paper works, increase accuracy of the calculations and handle files perfectly.

Monthly analyzed reports can be downloaded from the system which can be used in the decision-making process of the system. Which gives a chance to the company to increase profit and performance of it.

1.1 Functional Requirement

Reliability - Consisted of good performance and able to trust the content. The system is developed involving the latest technologies.

Efficiency - The highest or the peak level of performance when using the least amount of inputs in order to generate the highest output.

The system will handle all the input data without any problems. So that every task that done using this system is very fast. So user can gain huge efficiency from this system.

Accuracy - The correctness or the precision of the stored or produced data in the system. Accuracy of the data is very important in this system because this company always deals with sensitive data of the customer and also with lots of calculations. So inputs data need to be correct. If it is not all the calculations become wrong. Because of that in this system all the input data will be validating in entry level as well as database level.

Ease of use - Easy to handle and understand the concepts of a process. All the process of this system is break down to small processes and implement. So then any user can interact with this system very easily. Every part of this system can be handle by any user who has basic knowledge of handling a computer.

Usability - How easy it is to use and handle the system for the client can be defined as usability. In the implemented system a side navigational bar consists of easy keywords with images to access any page. All the details are displayed and described in a simple form which will be easy to understand to any user.

1.1.1 Use Case Diagrams

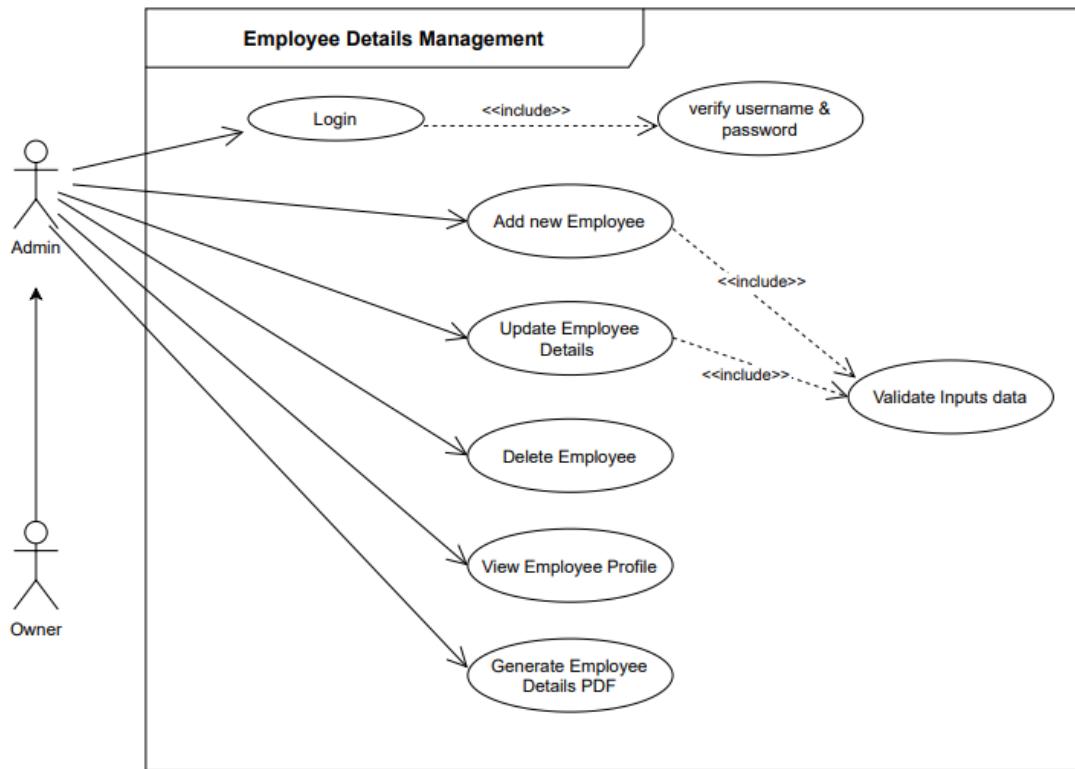


Figure 2: UD – employee details management

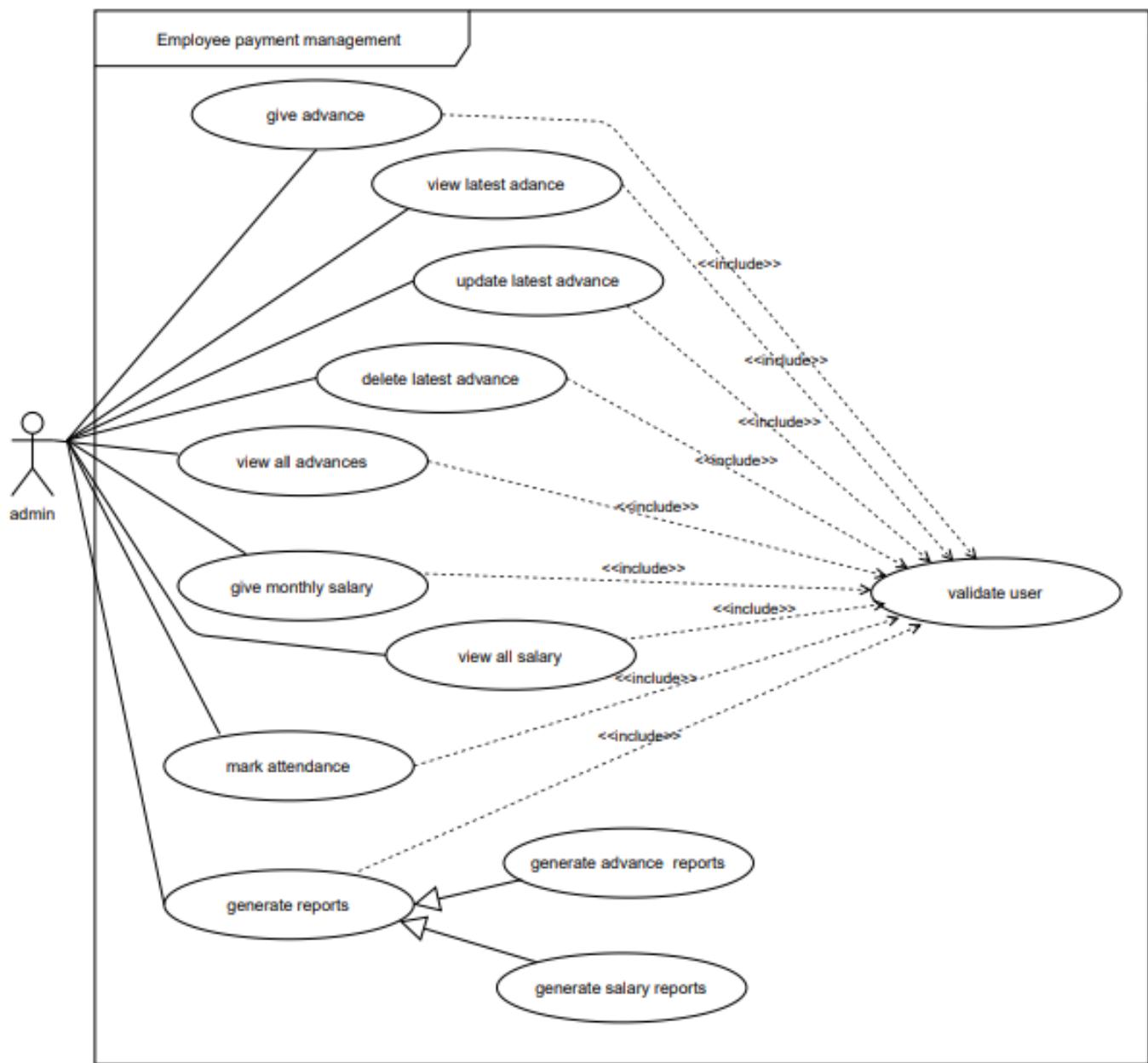


Figure 3: UD – employee payments management

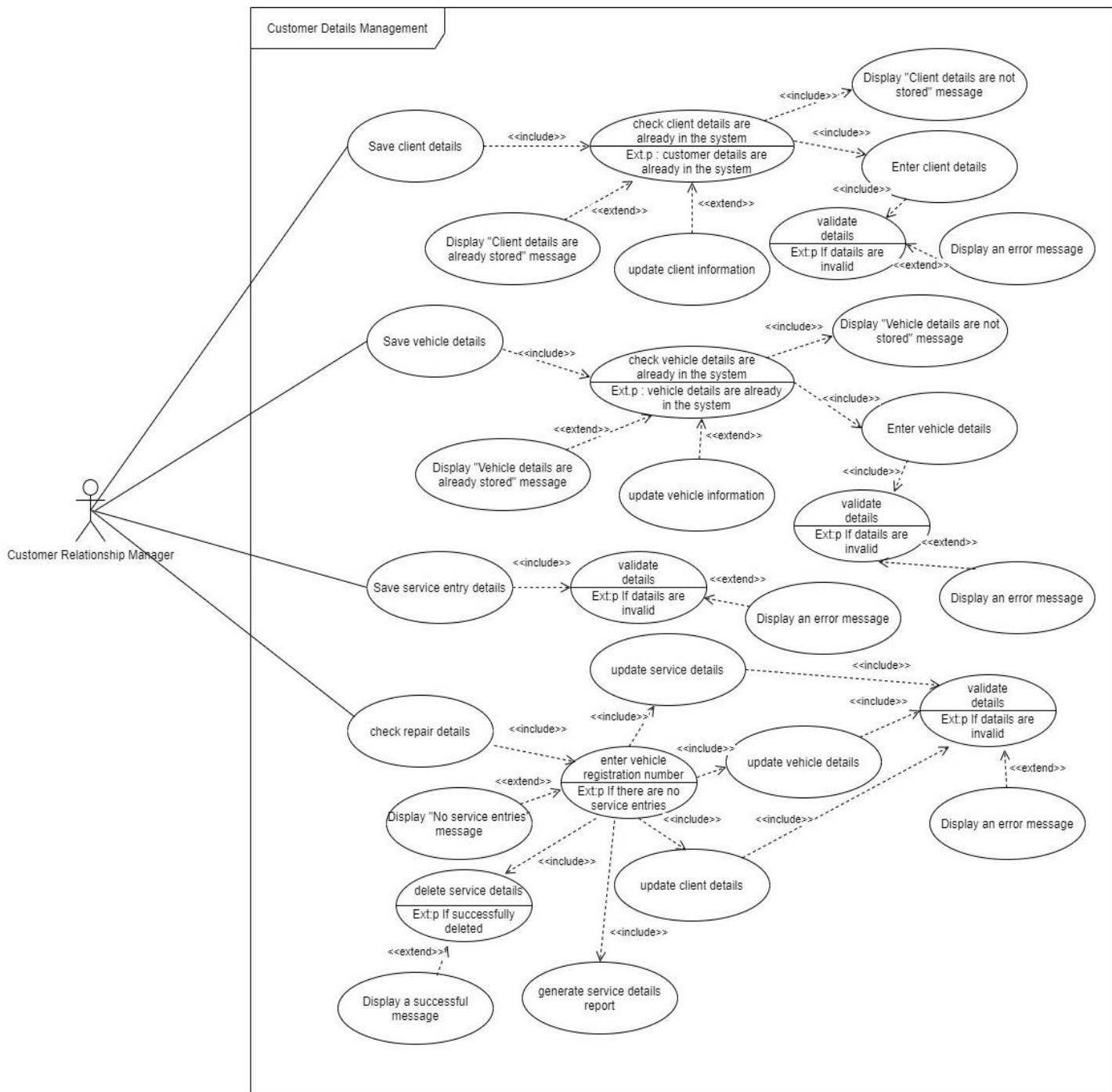


Figure 4: UD –customer details management

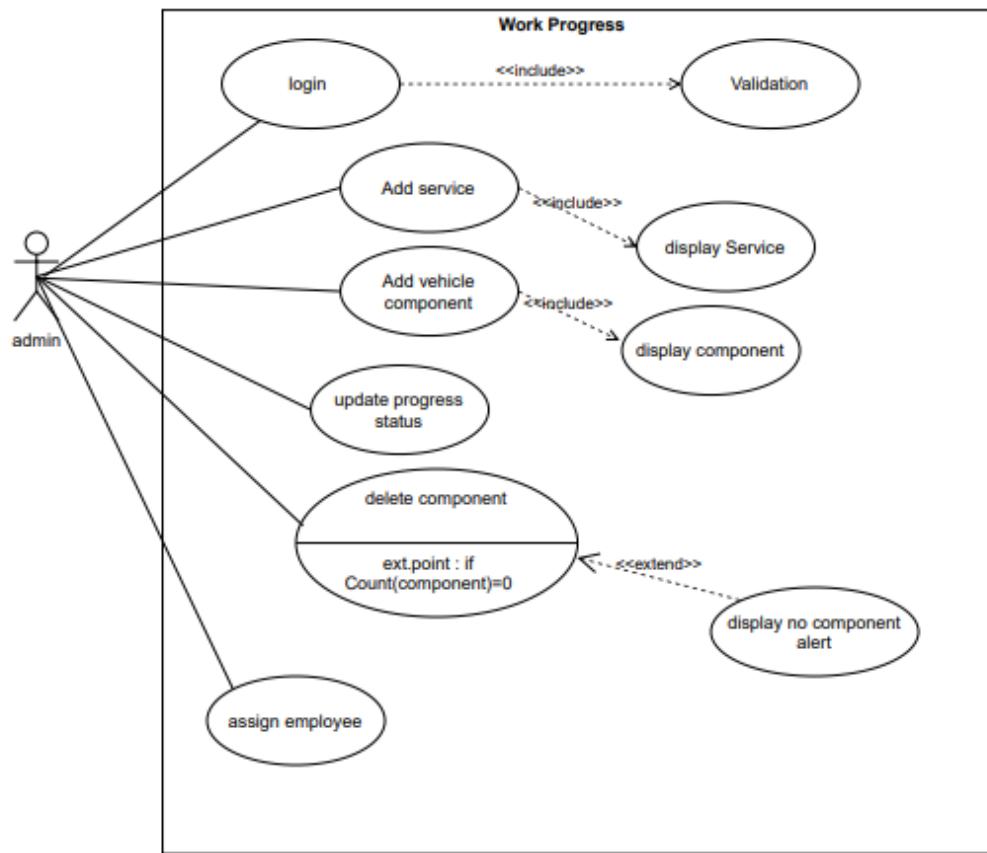


Figure 5: UD –work progress management

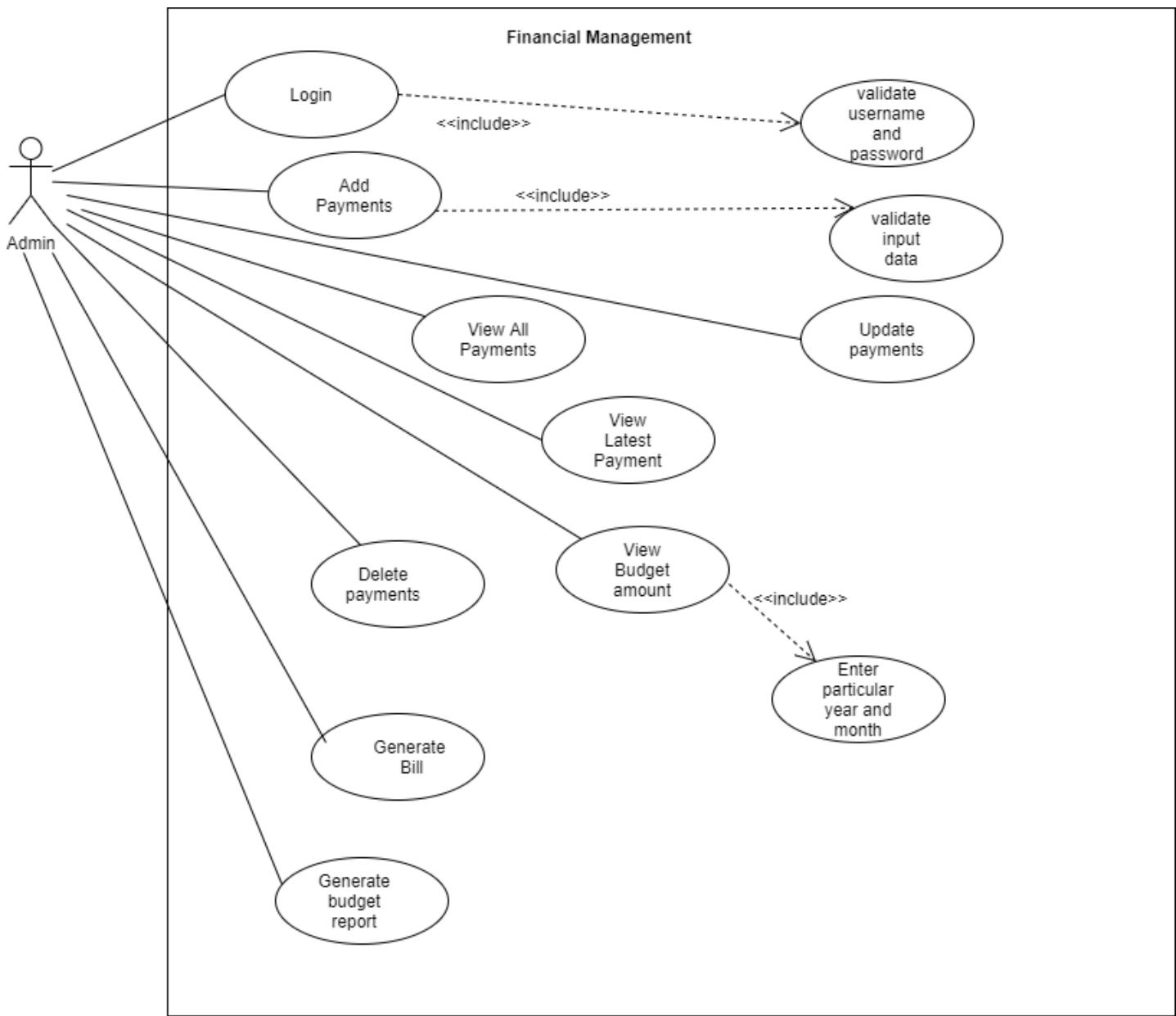


Figure 6: UD –financial management

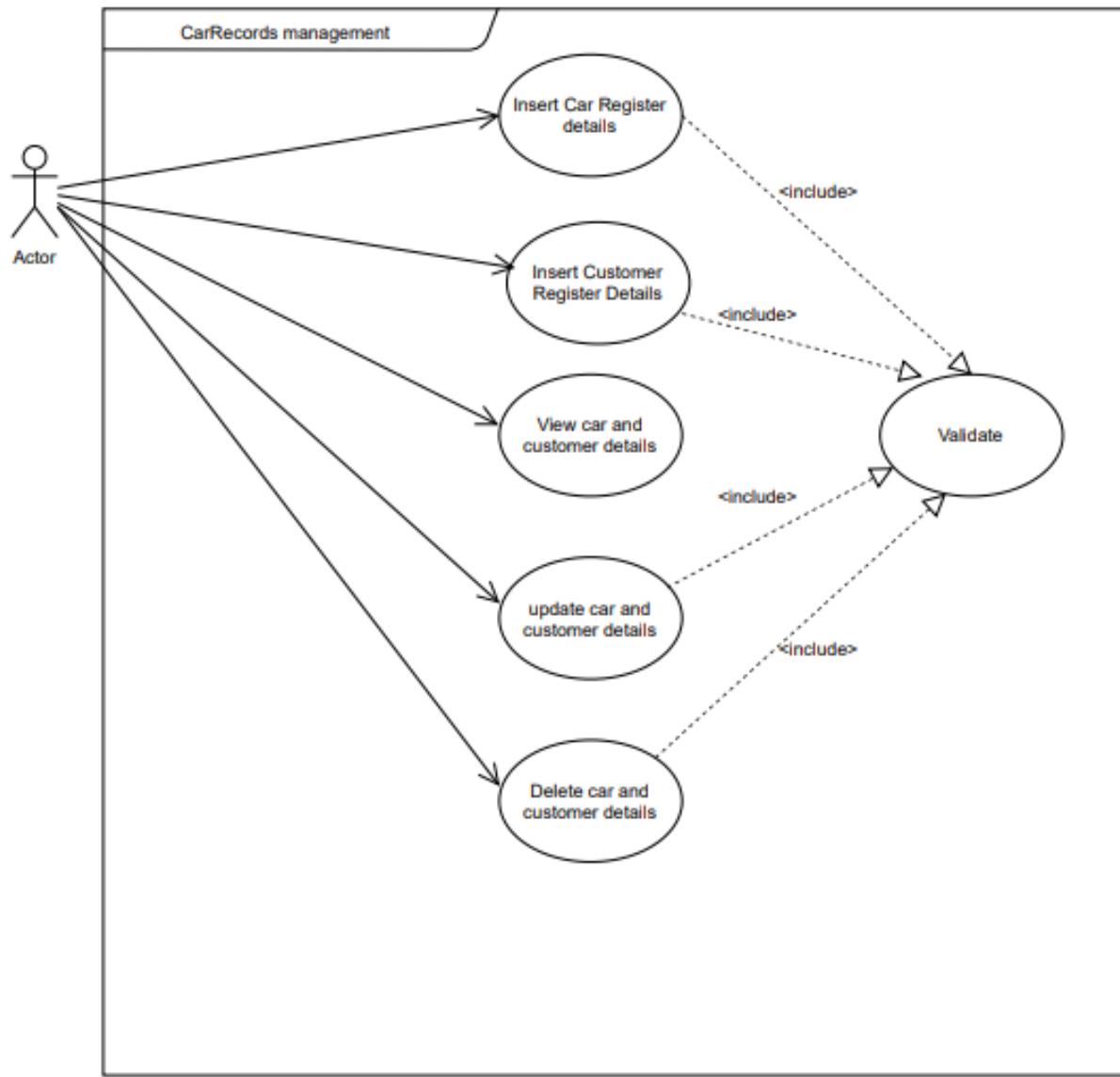


Figure 7: UD –car records management

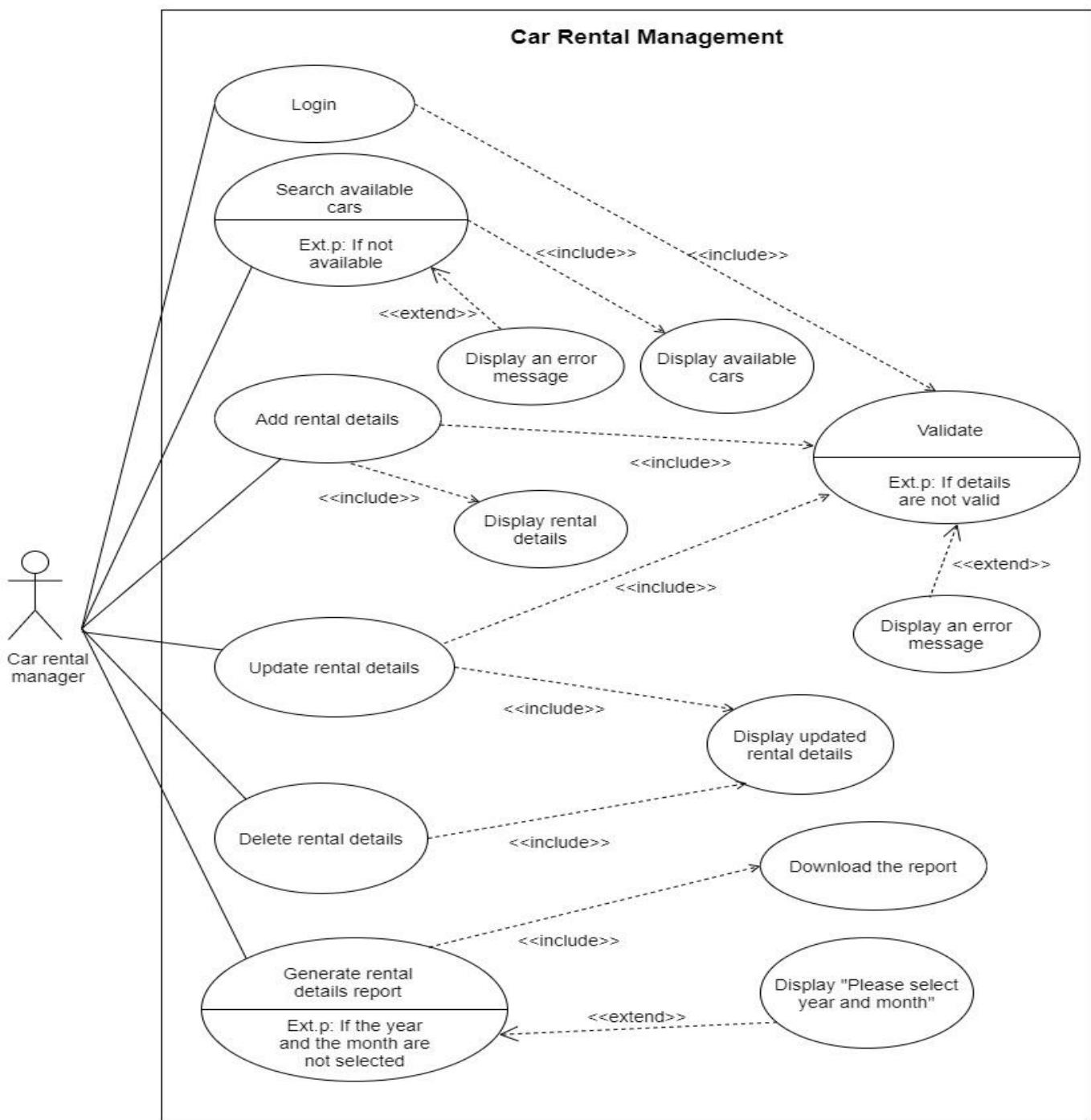
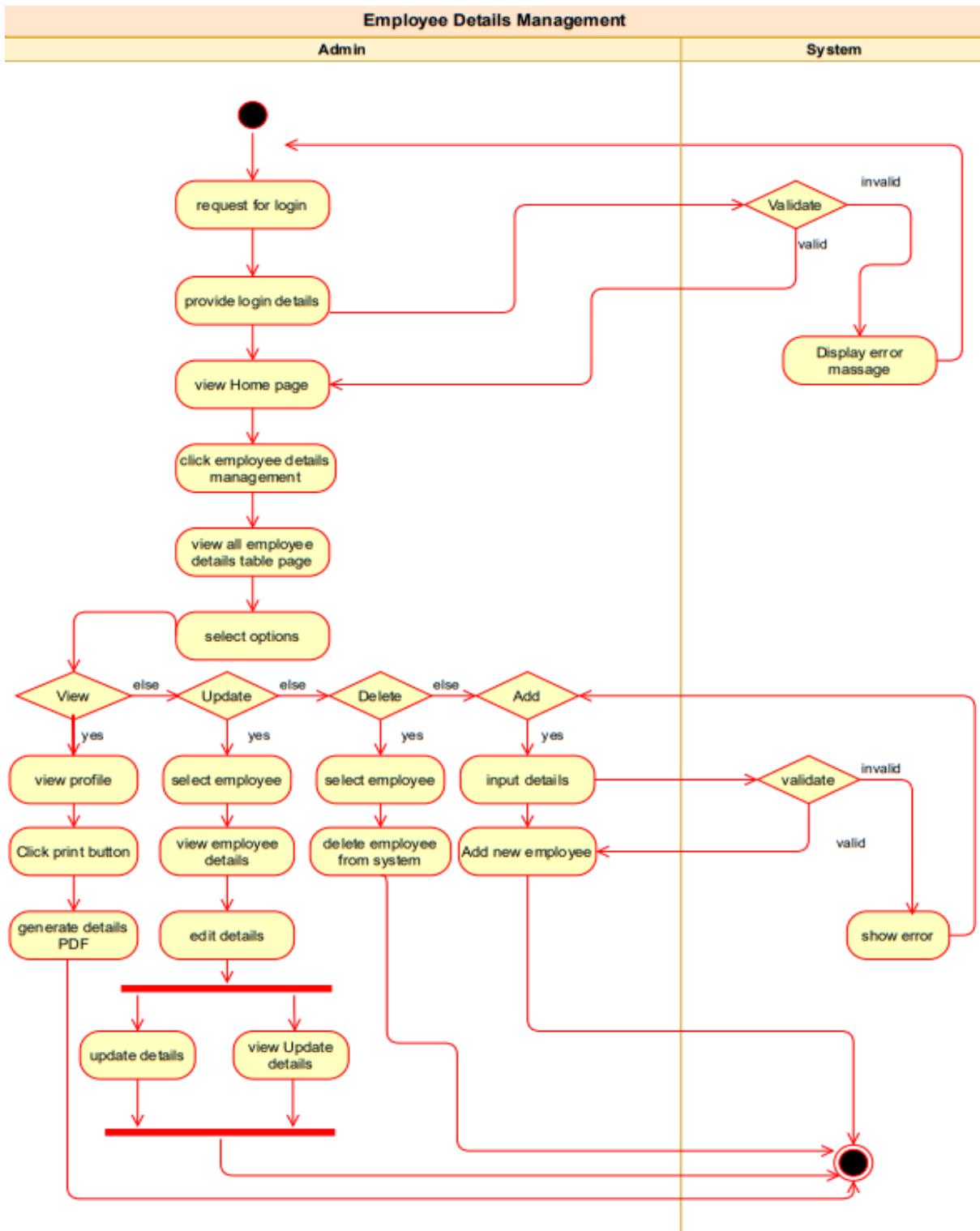


Figure 8: UD –car rental management

1.1.2 Activity Diagrams

Figure 9: AD – employee details management



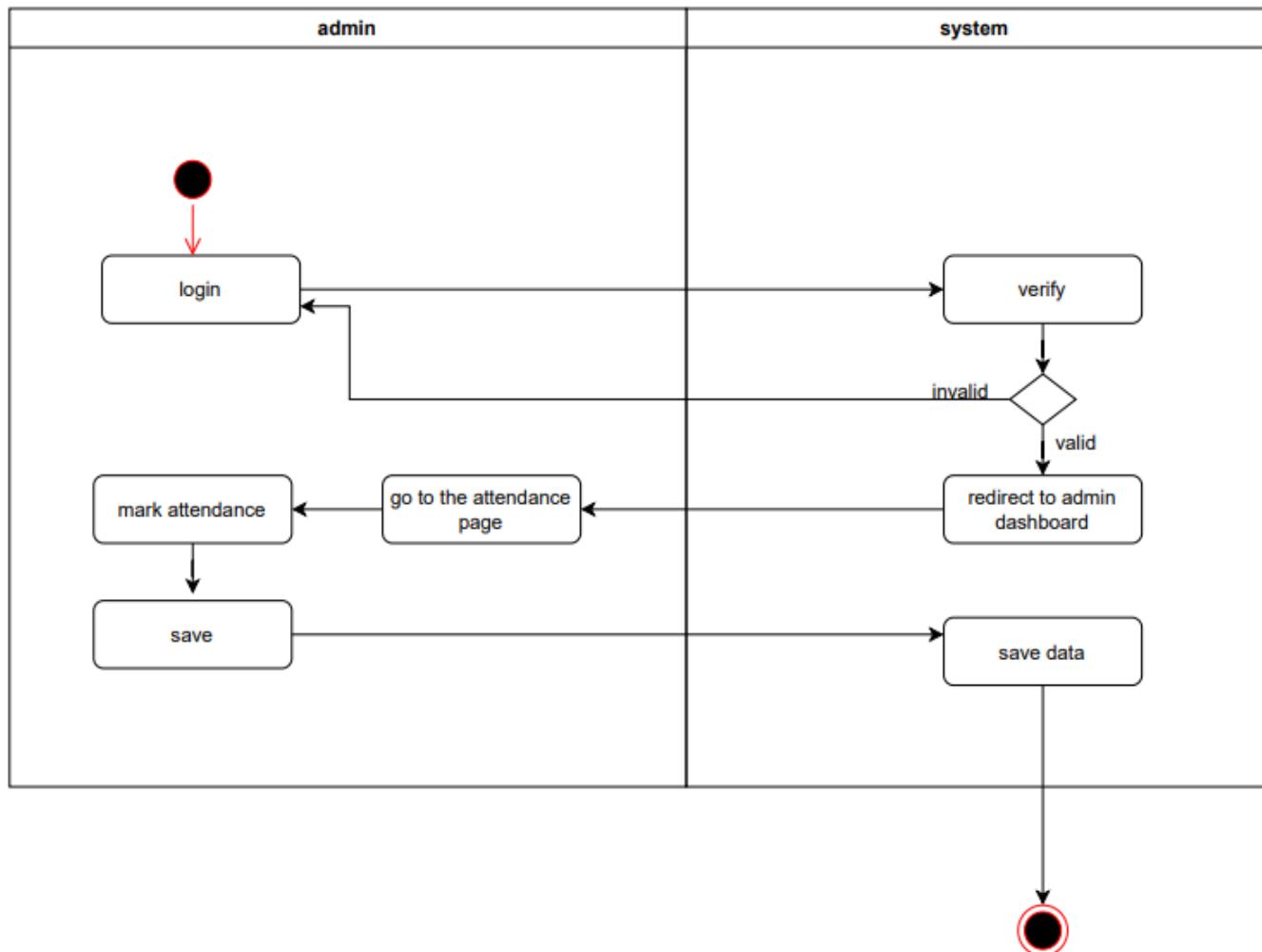


Figure 10: AD – Employee payment management (attendance management)

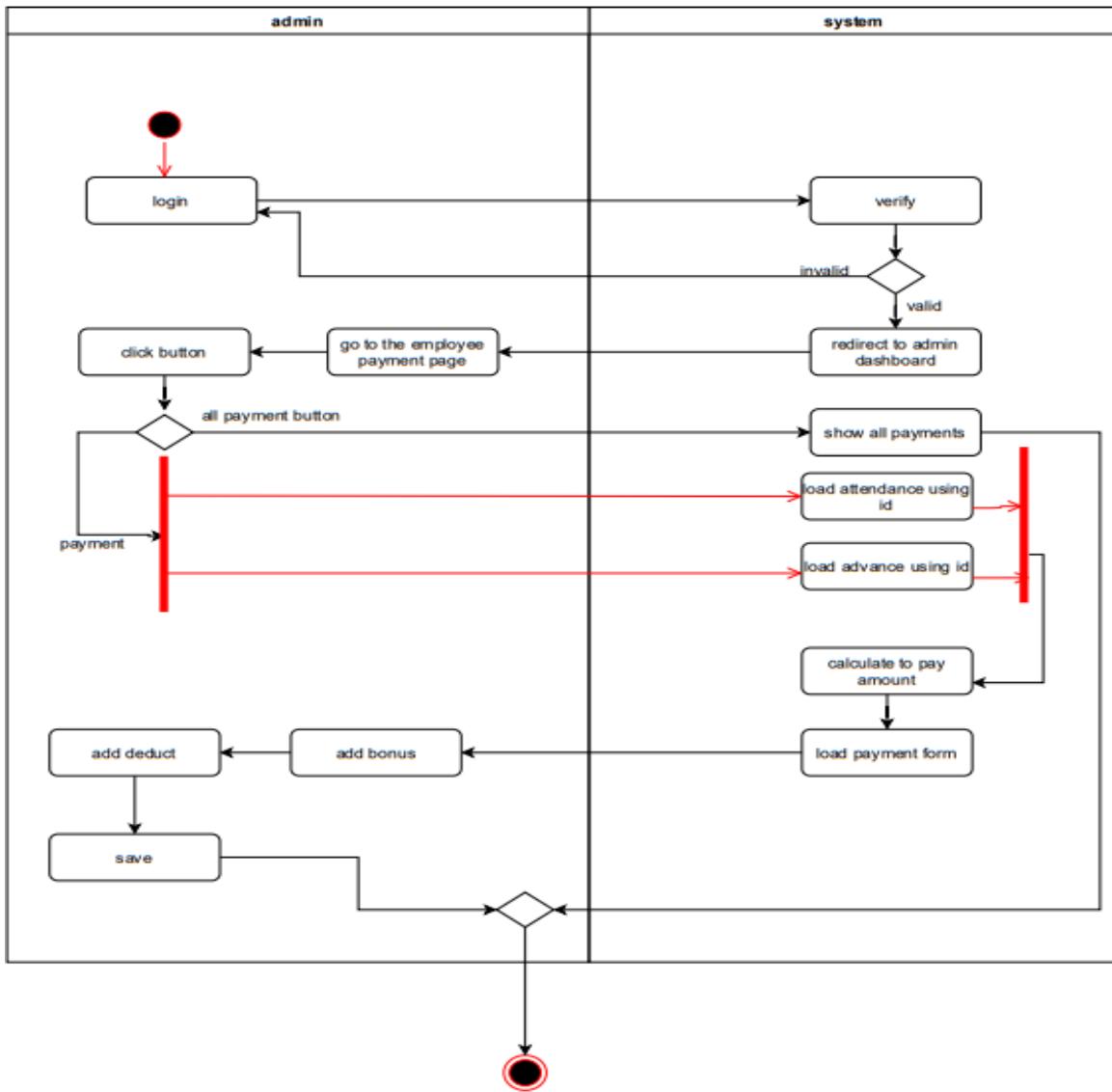


Figure 11: AD – Employee payment management (salary management)

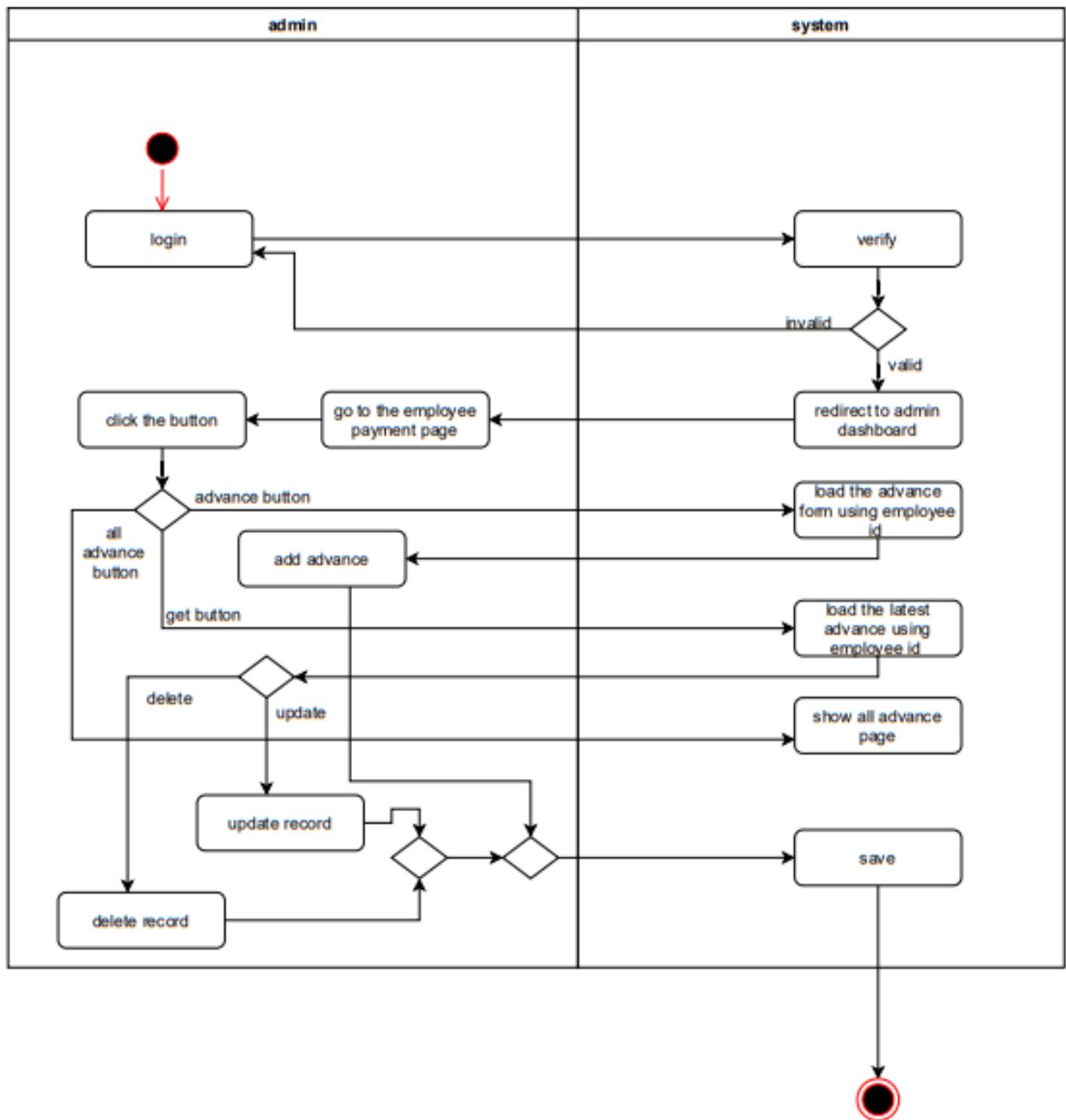


Figure 12: AD – Employee payment management (advance management)

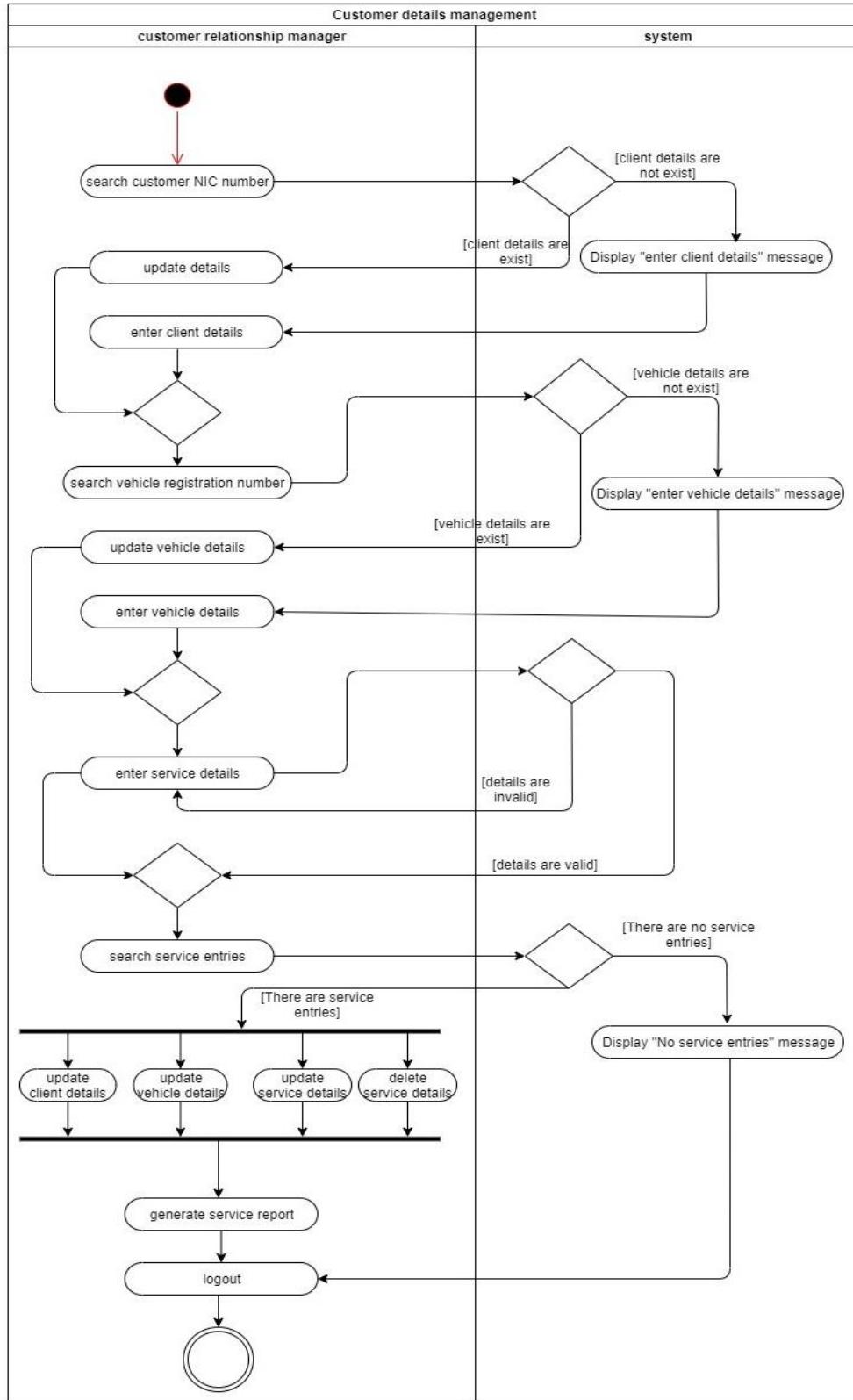


Figure 13: AD-customer details management

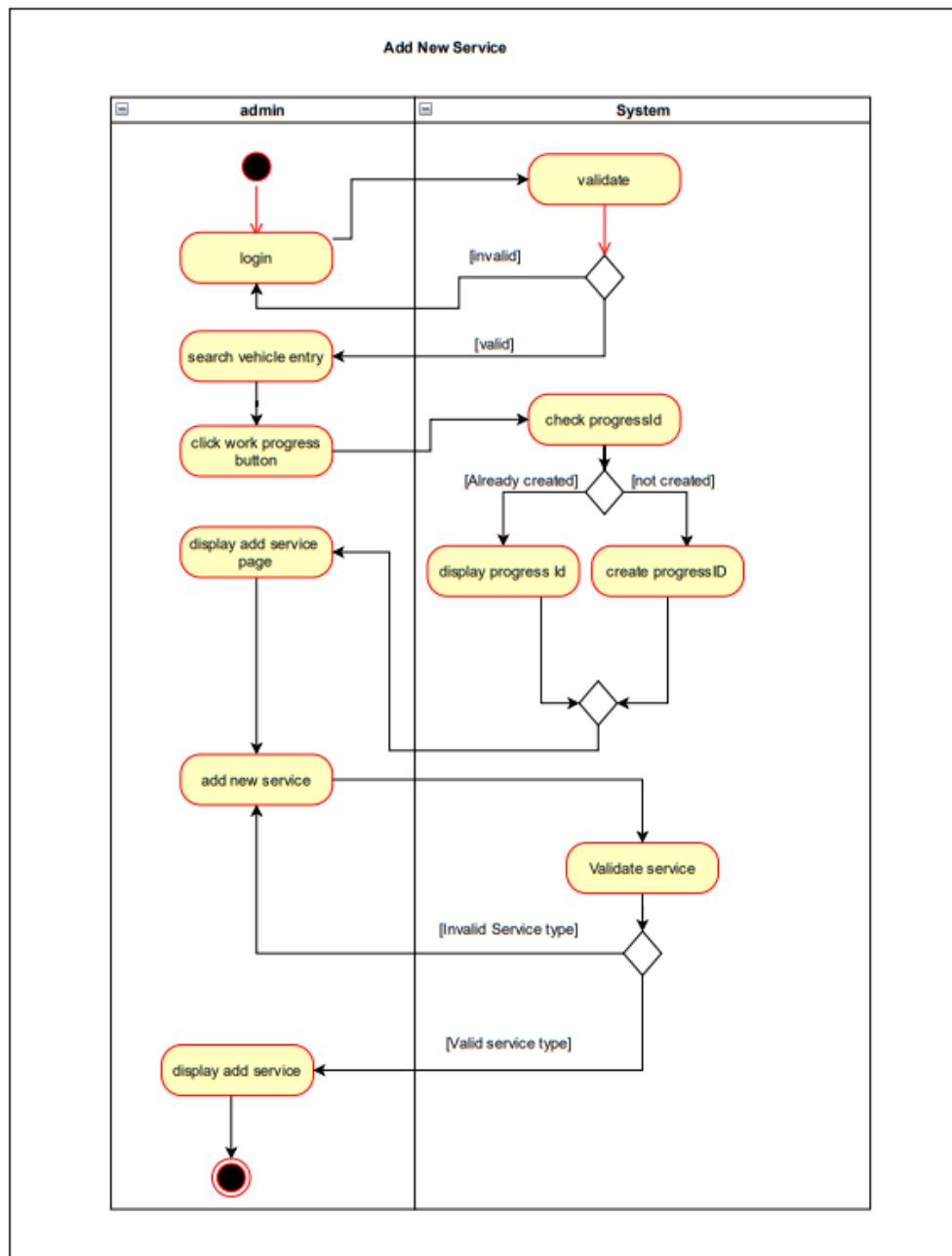


Figure 14:AD work progress management

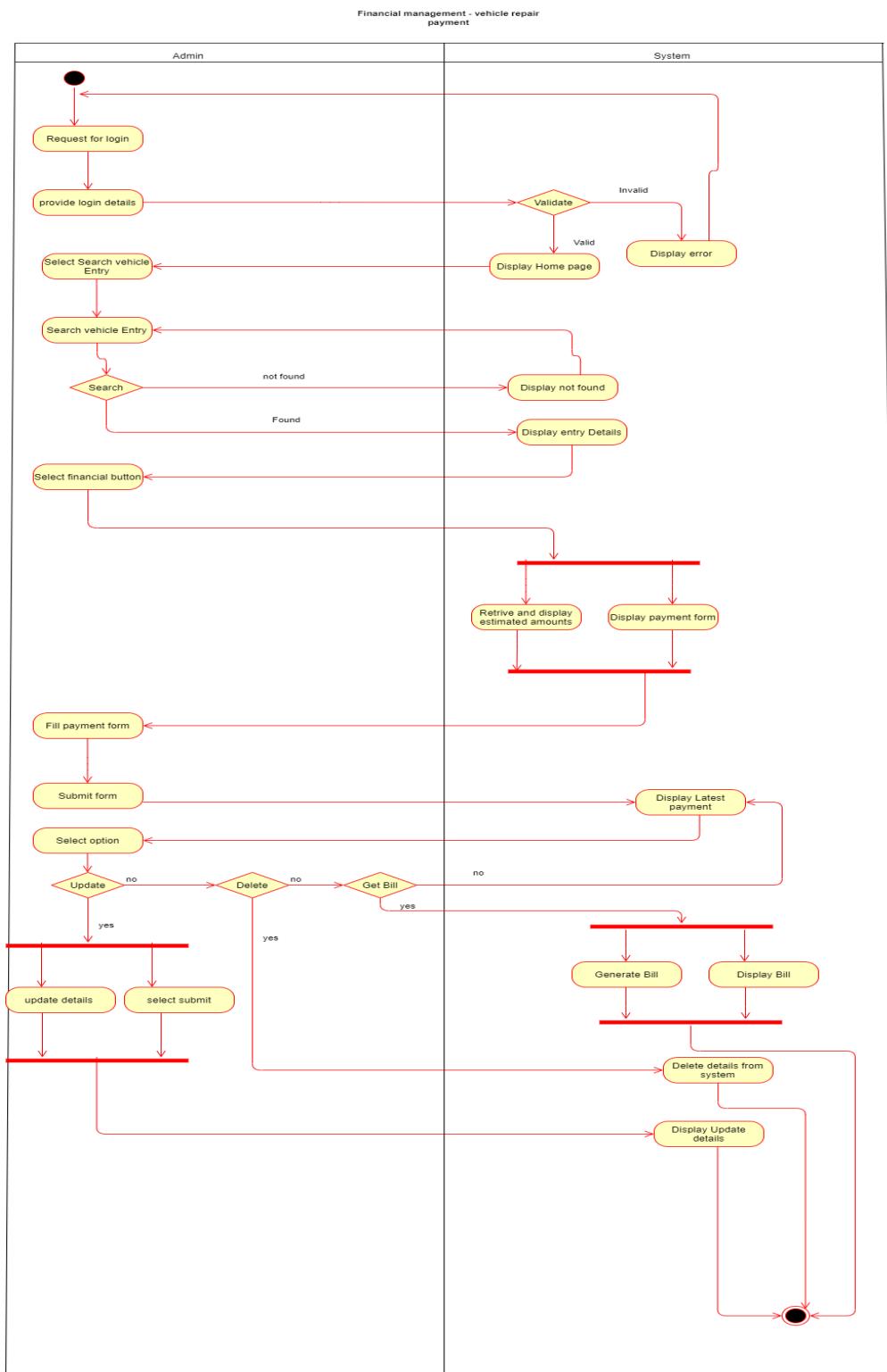


Figure 15: AD financial management (payment management)

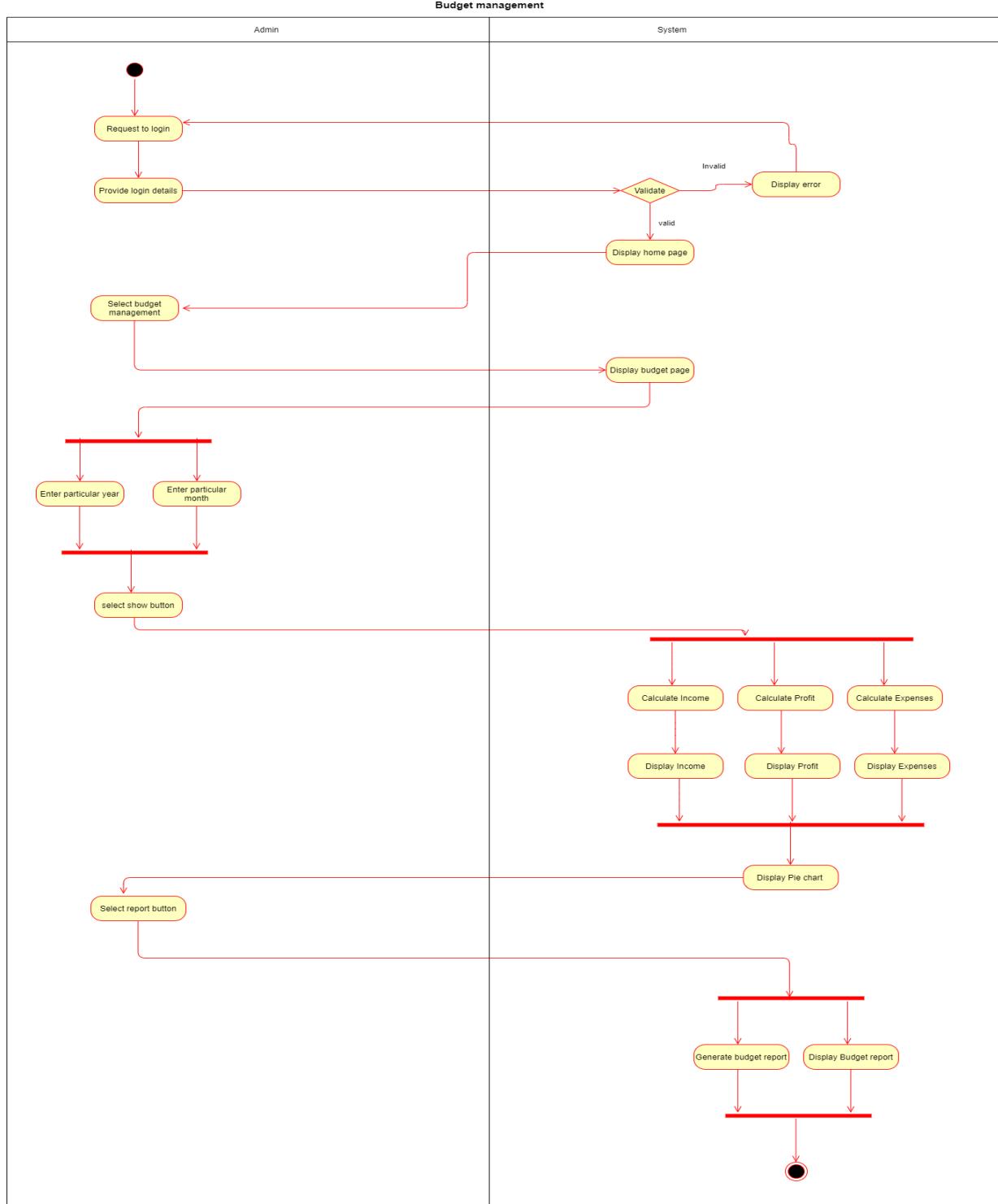


Figure 16: AD financial management (Budget management)

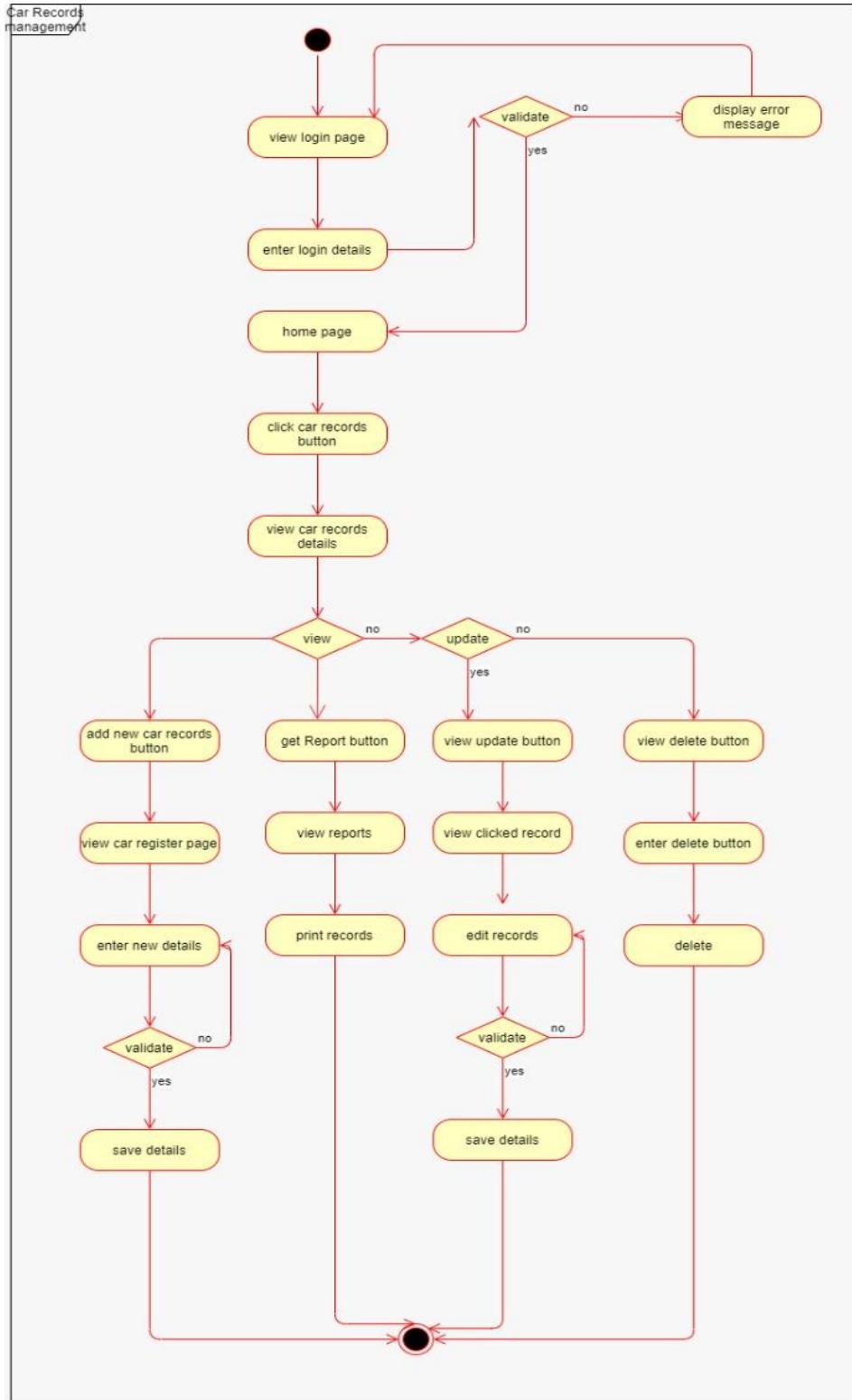
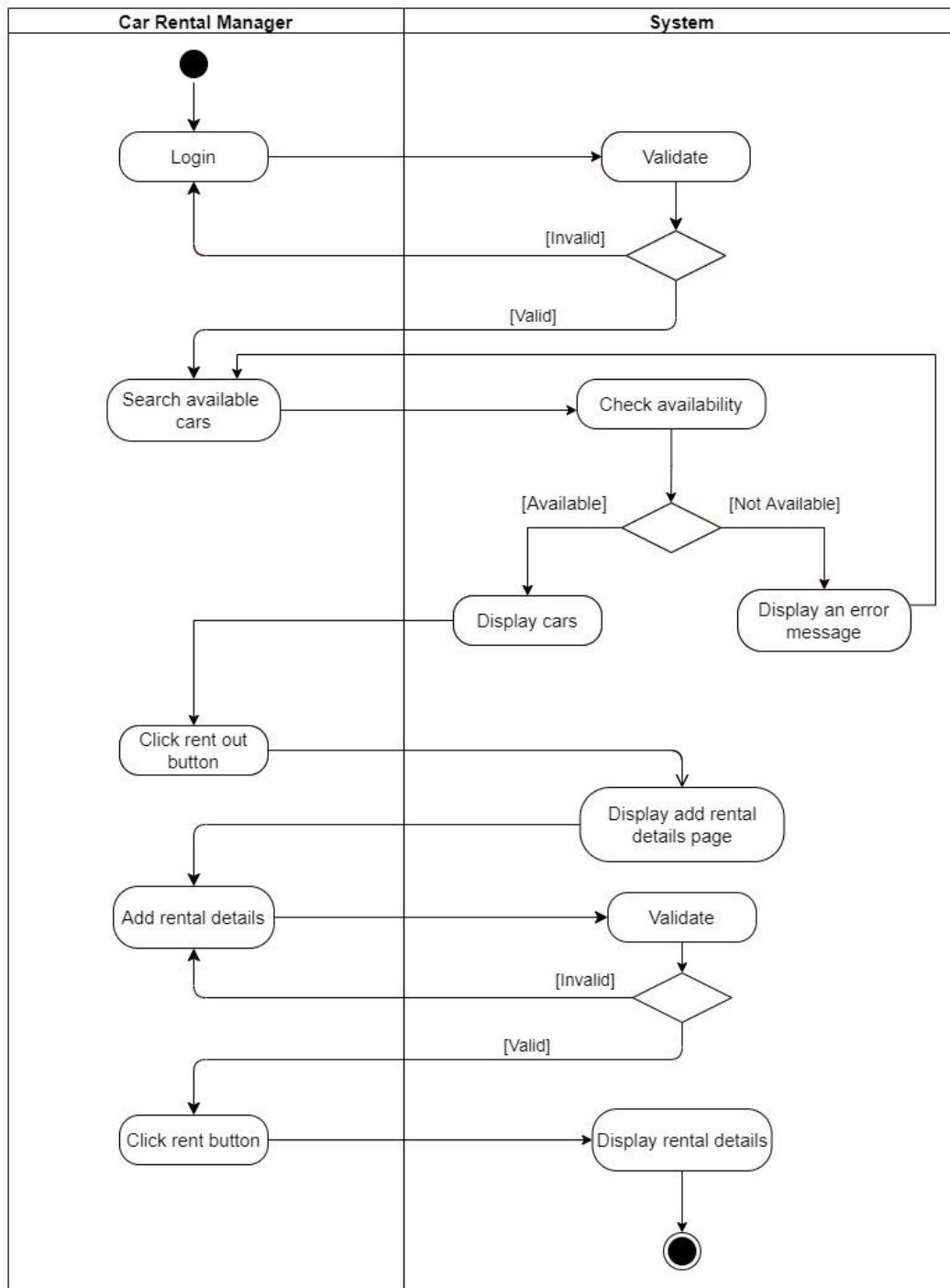


Figure 17:AD car records management

Figure 18: AD car rental management



1.2 Design

1.2.1 High Level Architecture Diagram

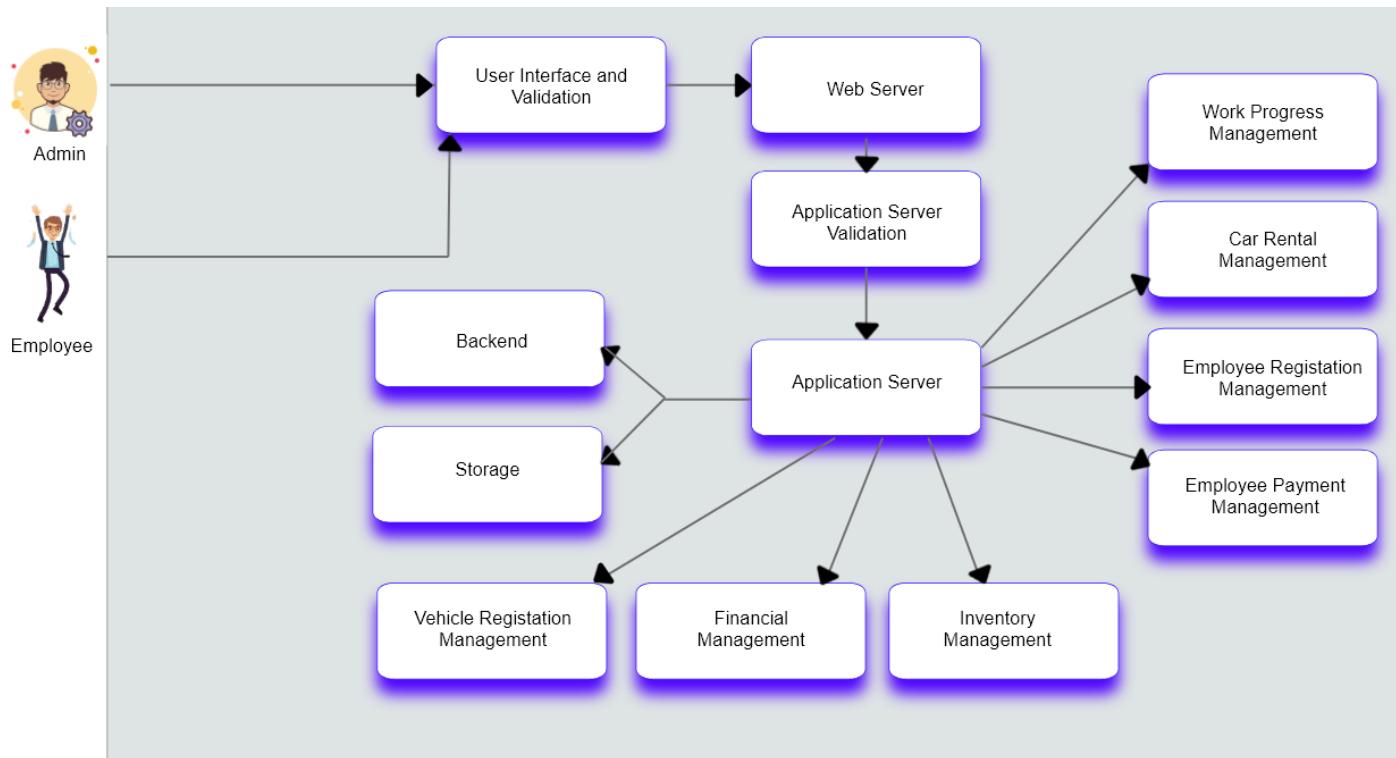
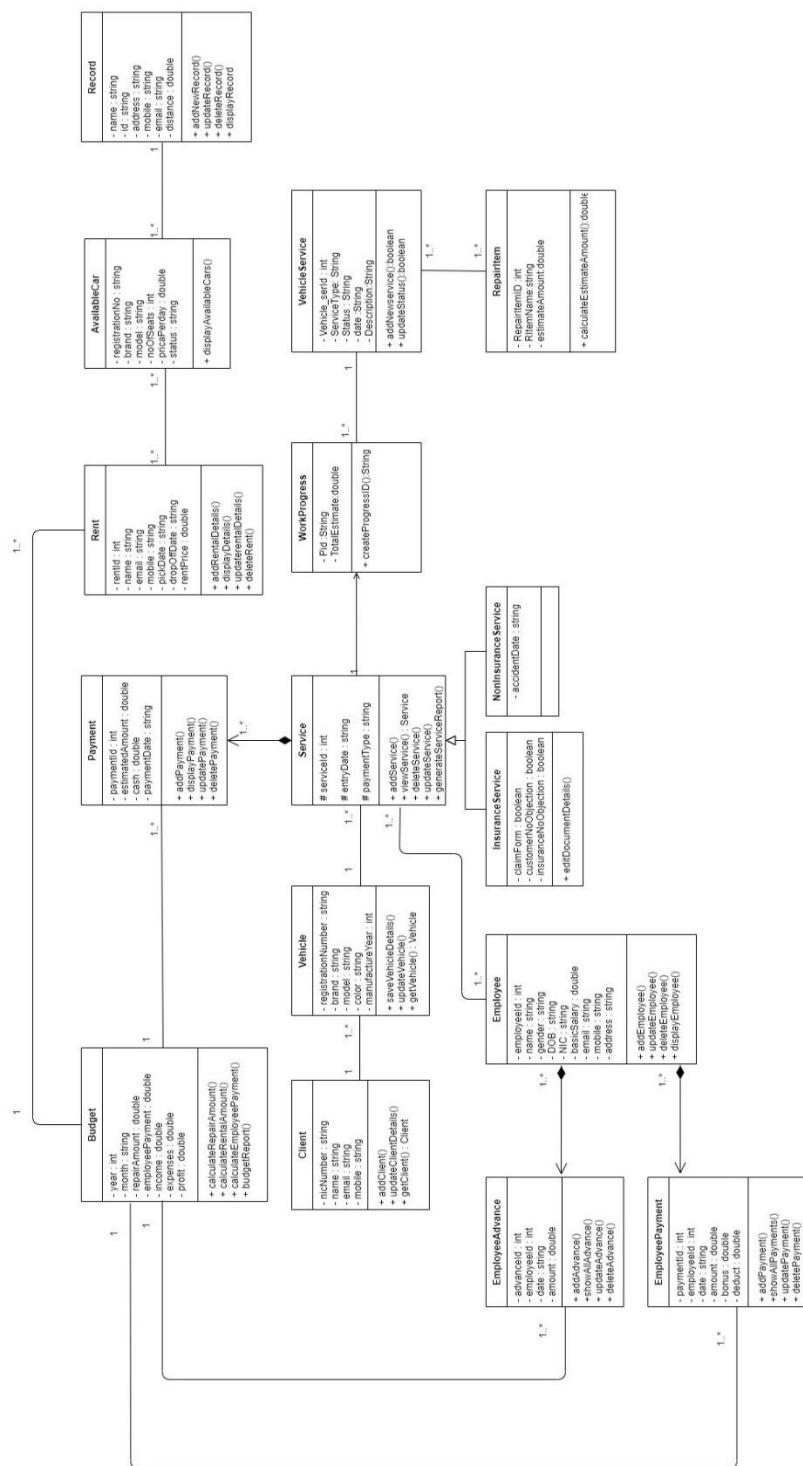
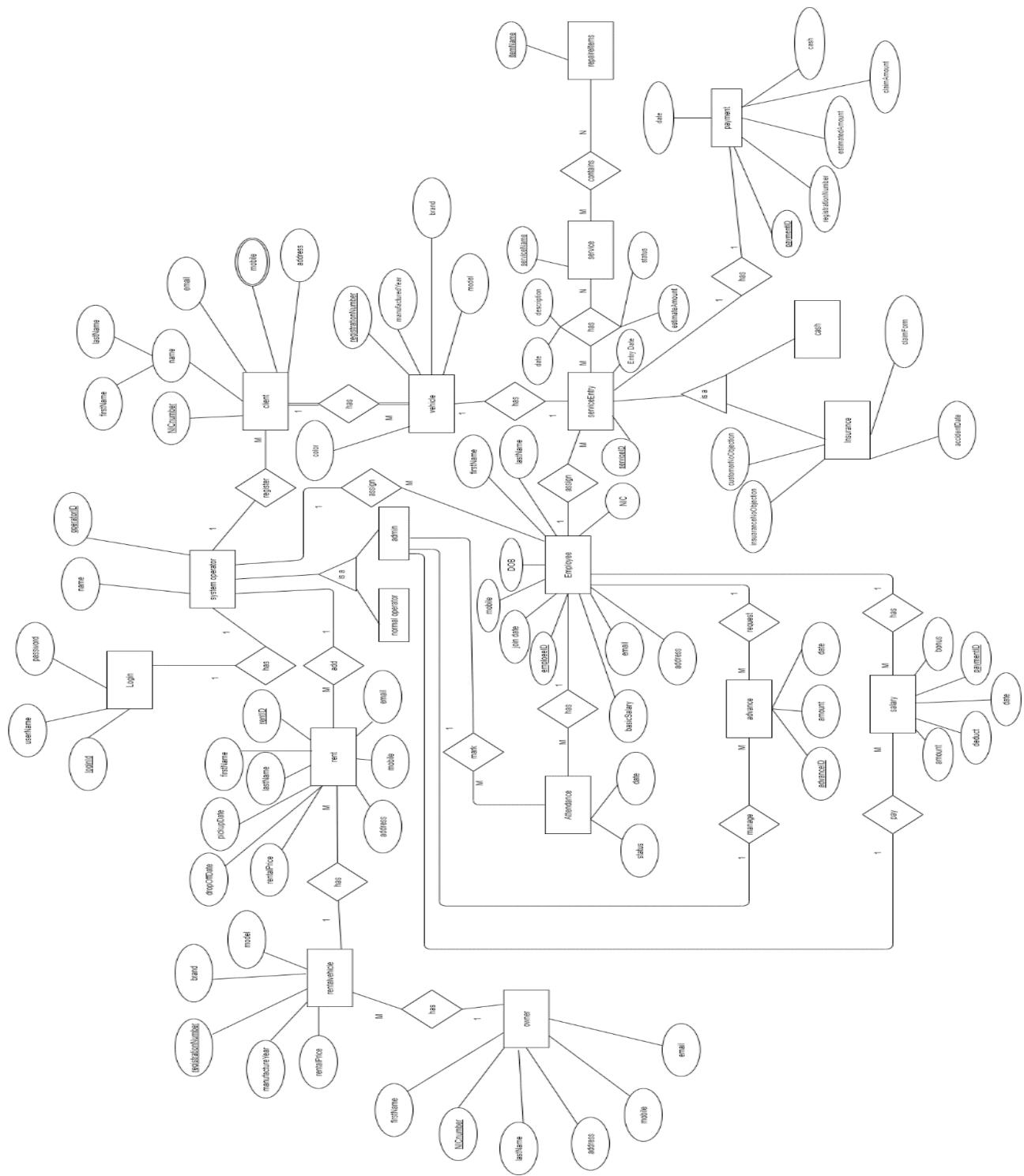


Figure 19: High level architecture diagram

1.2.2 Class Diagram



Entity Relationship Diagram - Vehicle Repair Management System



Customer Registration

localhost:8080/web_war_exploded/customerreg.jsp

DATA | Atlas: Mongo... Welcome | AWS Trai... Pathways Question Library: Le... Java Learning Subs... Software Testing Str... Judge System | Has... PCAP - Programmi...

ADD SERVICE ENTRY SEARCH ENTRY BUDGET MANAGEMENT INVENTORY MANAGEMENT EMPLOYEE MANAGEMENT CAR RENT CAR RECORD

Enter NIC Number...

Customer Registration Enter client details **Vehicle Registration** Enter Vehicle Information **Service Entry** Enter Service Entry Information

First Name Last Name

NIC Number Phone Number

Address

Email

Figure 22: customer registration

vehicle registration

localhost:8080/web_war_exploded/AddCustomerServlet

DATA | Atlas: Mongo... Welcome | AWS Trai... Pathways Question Library: Le... Java Learning Subs... Software Testing Str... Judge System | Has... PCAP - Programmi...

ADD SERVICE ENTRY SEARCH ENTRY BUDGET MANAGEMENT INVENTORY MANAGEMENT EMPLOYEE MANAGEMENT CAR RENT CAR RECORD

Enter Registration Number...

Customer Registration Enter client details **Vehicle Registration** Enter Vehicle Information **Service Entry** Enter Service Entry Information

Vehicle Registration Number

Brand Name
 Toyota

Model Color

Manufactured Year

Next

Figure 23: vehicle regisraion

The screenshot shows a web application interface for adding a service entry. On the left, a sidebar menu lists various management options: ADD SERVICE ENTRY, SEARCH ENTRY, BUDGET MANAGEMENT, INVENTORY MANAGEMENT, EMPLOYEE MANAGEMENT, CAR RENT, and CAR RECORD. The main content area is titled 'Vehicle Registration' and contains fields for Vehicle Registration Number (AAA-405), NIC Number (980971422V), Entry Date (mm/dd/yyyy), and Accident Date (mm/dd/yyyy). Below these fields is a radio button group for Repair Type, with 'Non-Insurance' selected. A green checkmark icon and the word 'complete' are displayed at the bottom right of the form.

Figure 24: add service

The screenshot shows a search results page for service entries. The sidebar menu is identical to Figure 24. The main area features a search bar with a dropdown for 'Registration Number' and a search icon. Below the search bar is a table listing six service entries. Each entry includes the Registration Number (all listed as AAA-1000), Entry Date, Type (Non Insurance or Insurance), and a blue 'view' button.

Registration Number	Entry Date	Type	
AAA-1000	2021-04-09	Non Insurance	<button>view</button>
AAA-1000	2021-04-21	Insurance	<button>view</button>
AAA-1000	2021-04-11	Non Insurance	<button>view</button>
AAA-1000	2021-04-21	Insurance	<button>view</button>
AAA-1000	2021-04-13	Insurance	<button>view</button>
AAA-1000	2021-04-21	Insurance	<button>view</button>

Figure 25: search service entry

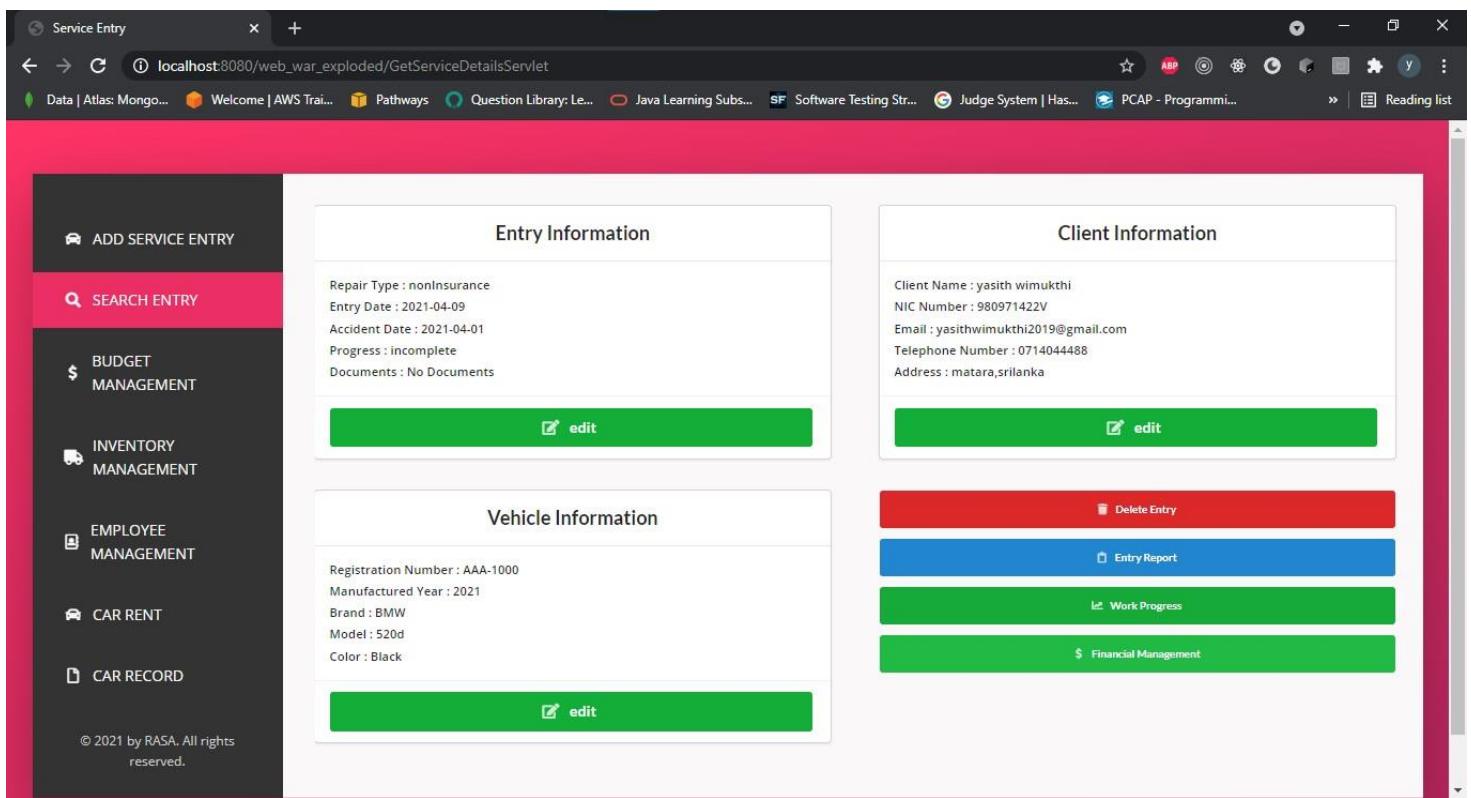


Figure 26: service entry page

Figure 27: Work progress management

The screenshot shows a web browser window titled "progress" with the URL "localhost:8080/web_war_exploded/NavigateToWorkprogressServlet". The main content area has a pink header with the title "Vehicle Service Onprogress..!". Below the title are two green buttons: "Show Estimate" and "+ Add Services". A table titled "Current total estimate" displays the following data:

Service type	Total estimate Amount	Total Components
Painting	3000.0	3
Remove and Refitting	5000.0	1
Repair Items	1000.0	1
Item to be replace	1000.0	1

Below the table is a section titled "Add Components" with four teal buttons: "Painting", "Remove and Refitting", "Repair Items", and "Item to be replace". On the left side of the page is a sidebar with icons for "SERVICE ENTRY", "SEARCH ENTRY", and "WORK PROGRESS". At the bottom left, there is a copyright notice: "© 2021 by RASA. All rights reserved." The bottom of the screen shows a Windows taskbar with various pinned icons and system status.

Figure 28: Add new Service

The screenshot shows a web browser window titled "Work Progress" with the URL "localhost:8080/web_war_exploded/CreateWorkProgressServlet". The main content area has a pink header with the title "Add New Services and Items". Below the title is a form with fields for "Select service" (dropdown menu "Services"), "Description (Optional)" (text input field with placeholder "..."), and a "Add Service" button. To the right of the form is a modal dialog box with the following content:

1000D
Total estimate amount
Assign Employee Id : 110922
Change Employee

At the bottom of the main content area is a table titled "Back" showing a list of services:

Service Name	Date	description	Status	Action
Painting	2021-05-19	--	Finished	
Remove and Refitting	2021-05-19	--	Finished	

On the left side of the page is a sidebar with icons for "ADD SERVICE ENTRY", "SEARCH ENTRY", "BUDGET MANAGEMENT", "INVENTORY MANAGEMENT", "EMPLOYEE MANAGEMENT", "CAR RENT", and "CAR RECORD". At the bottom left, there is a "Type here to search" bar and a Windows taskbar with various pinned icons and system status.

Figure 29: Assign employee

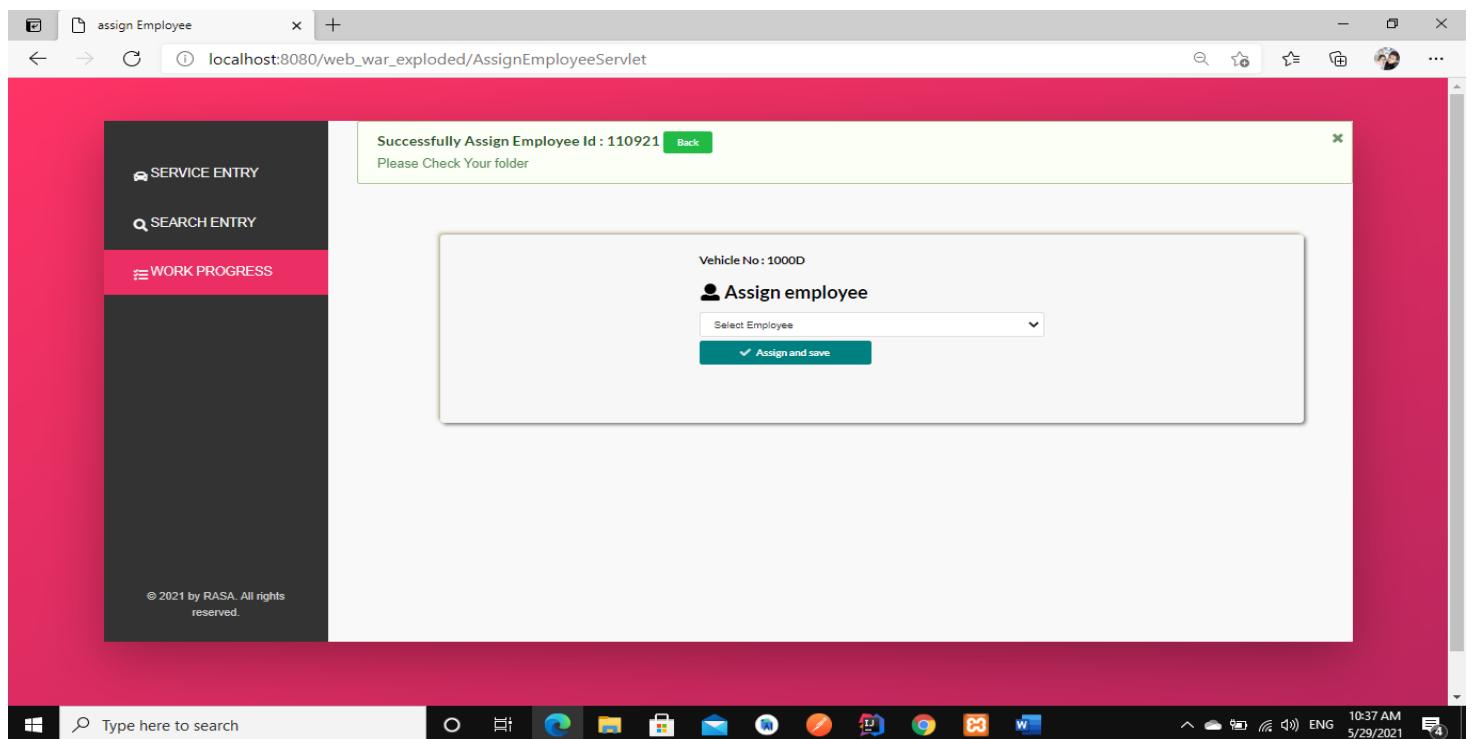


Figure 30: Add repair components

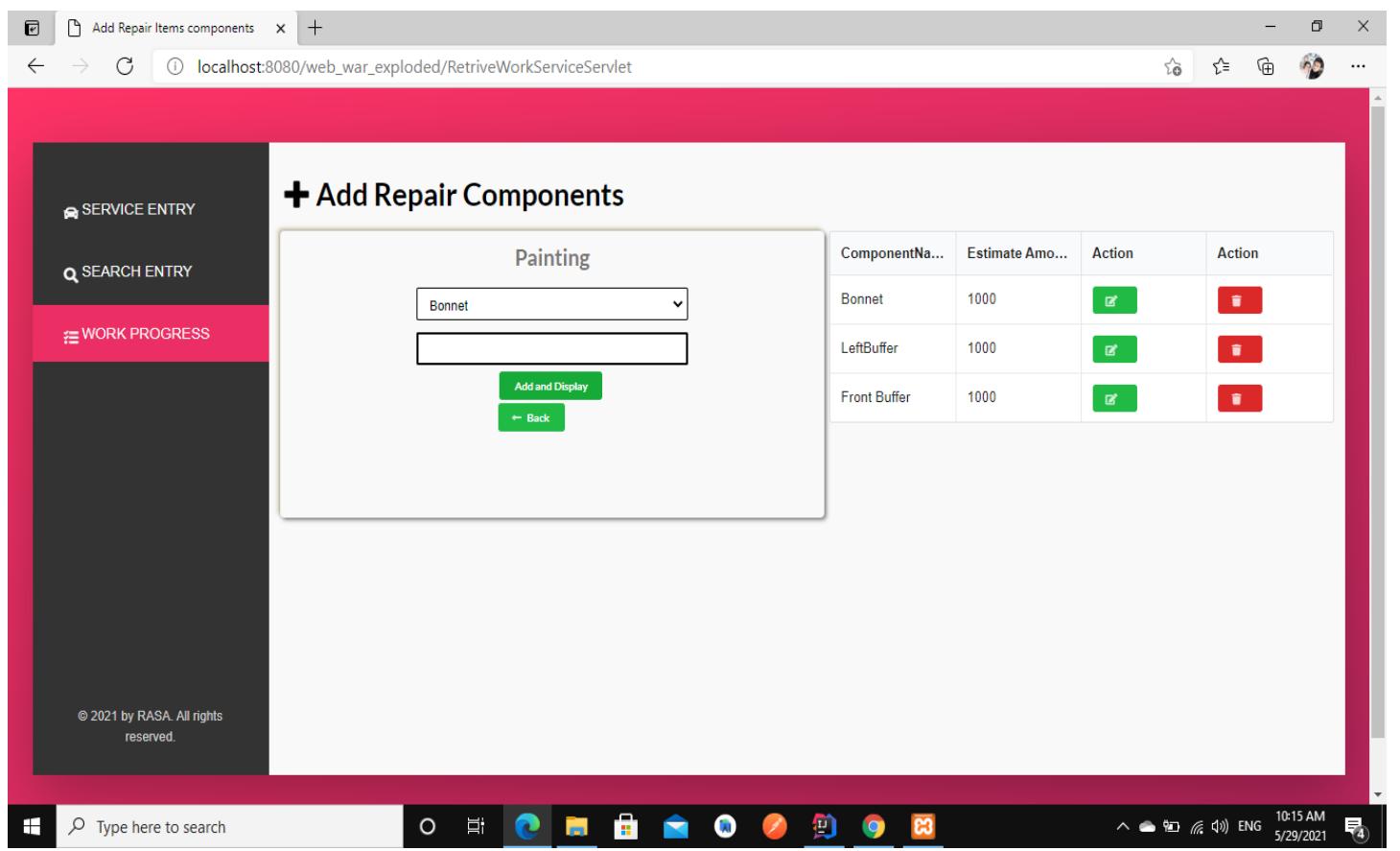


Figure 31: Estimate report overview

The screenshot shows a web browser window titled "Report handler" displaying an "Estimate Report" for Vehicle NO : 1000D. The left sidebar has three items: "SERVICE ENTRY", "SEARCH ENTRY", and "WORK PROGRESS". The main content area is titled "Estimate Report" and shows four sections of repair items:

- Painting**:

Repair Component	Estimate Amount
Bonnet	RS: 1000
LeftBuffer	RS: 1000
Front Buffer	RS: 1000
- Remove and Refitting**:

Repair Component	Estimate Amount
EngineCompartment	RS: 5000
- Repair Items**:

Repair Component	Estimate Amount
Front Buffer	RS: 1000
- Item to be replace**:

Repair Component	Estimate Amount

Each section has a "Change Estimate" button to its right. The browser address bar shows "localhost:8080/web_war_exploded/estimatereportview.jsp". The taskbar at the bottom includes icons for File Explorer, Mail, Edge, Task View, and others, along with system status indicators like battery level and network connection.

Figure 32: Financial Management

The screenshot shows a web-based financial management system. On the left, a sidebar menu lists various modules: ADD SERVICE ENTRY, SEARCH ENTRY, BUDGET MANAGEMENT, INVENTORY MANAGEMENT, EMPLOYEE MANAGEMENT, CAR RENT, and CAR RECORD. The main content area displays a table titled "ESTIMATED AMOUNT" with the following data:

ID	Services	Amount
1	Painting	2000.0
2	Remove and Refitting	11000.0
3	Replace items	10000.0
4	items to be replace	8000.0
Total Amount		30000.0

Below this is a form titled "Add payment" with fields for Vehicle Registration number, Estimation Amount, Cash, and Date of Payment (mm/dd/yyyy). A "submit" button is at the bottom.

The screenshot shows a "Latest Payment" page. The sidebar menu is identical to Figure 32. The main content area displays a table titled "Latest Payment" with the following data:

Payment ID	Registration Number	Estimate Amount	Cash Amount	Payment Date	Action
29	AAA-104	250000.0	230000.0	2021-05-14	

A green "All Payments" button is located in the top right corner of the main content area.

Figure 33: Latest Payments

The screenshot shows a web application interface titled "Payments". On the left, there is a sidebar with various management options: ADD SERVICE ENTRY, SEARCH ENTRY, BUDGET MANAGEMENT, INVENTORY MANAGEMENT, EMPLOYEE MANAGEMENT, CAR RENT, and CAR RECORD. Below the sidebar, a copyright notice reads: "© 2021 by RASA. All rights reserved." The main content area is titled "Payments" and displays a table of transaction data:

Payment ID	Registration Number	Estimate Amount	Cash	Date
1	AAA-102	100000.0	100000.0	2021-05-13
5	AAA-102	10000.0	10000.0	2021-05-21
6	AAA-102	250000.0	250000.0	2021-05-20
25	AAA-002	15000.0	15000.0	2021-04-09
26	AAA-102	10000.0	10000.0	2021-05-29
27	AAA-102	15000.0	10000.0	2021-05-21
29	AAA-104	250000.0	230000.0	2021-05-14

Figure 34: All payments

The screenshot shows a web application interface titled "Budget/month". On the left, there is a sidebar with various management options: ADD SERVICE ENTRY, SEARCH ENTRY, BUDGET MANAGEMENT, INVENTORY MANAGEMENT, EMPLOYEE MANAGEMENT, CAR RENT, and CAR RECORD. Below the sidebar, a copyright notice reads: "© 2021 by RASA. All rights reserved." The main content area is titled "Budget/month" and displays three colored boxes: "Income" (green), "Profit" (blue), and "Expenses" (red). To the right, there is a dropdown menu set to "January" with a "Show" button, and a "Report" button. Below the boxes, there is a pie chart titled "Monthly Budget" with the following distribution:

- Income: 50%
- Expenses: 8%
- Profit: 42%

Figure 35: Budget Ui

Rent Car Records Management

Figure 36: Car Records retrieve

RegNumber	fname	lname	ID	Address	Phone	Email	BookNumber	Model	SeatAmount	Distance	CarType	Action	
29	Thilina	Madushanka	971420464V	malabbe	54645645	tm@gmail.com	23-6797	toyota-axio	6	12000.0	gfgt		
31	Thilina	kalum	971420464	malabbe	774565123	t4ka@gmail.com	23-6797	nissan	6	43000.0	good		
32	Thilina	Madushanka	971420464V	dematagoda	774546512	th9@gmail.com	yt6754	toyota-corolla	5	45000.0	low condition		
35	Thilina	Madushanka	971420464V	dematagoda	774546512	t4ka@gmail.com	yt6754	toyota-corolla	8	56000.0	zar		

Figure 37: Car Records Register

Car Records Registration form

localhost:8081/web_war_exploded/CarRecordsReg.jsp?

Apps youtube - Google S... Operating System... Introduction To Op... New Tab Reading list

Car Registration

First Name
Enter First Name..!!

Last name
Enter Last Name..!!

NIC
Enter ID number..!!

Address
Enter ID number..!!

Phone number
Enter ID number..!!

ADD SERVICE ENTRY
SEARCH ENTRY
BUDGET MANAGEMENT
INVENTORY MANAGEMENT
EMPLOYEE MANAGEMENT
CAR RENT
CAR RECORD

Type here to search

3:00 PM 5/29/2021

Car Records Registration form

localhost:8081/web_war_exploded/CarRecordsReg.jsp?

Apps youtube - Google S... Operating System... Introduction To Op... New Tab Reading list

email
Enter Email Address..!!

Vehicle Book number
Enter Car Book Number..!!

Vehicle Model
Enter Vehicle Model..!!

Vehicle seat amount
Enter Seat Amount..!!

Distance
Enter Distance before Registration..!!

Car Condition
Enter Car Condition..!!

Register

© 2021 by RASA. All rights reserved.

Type here to search

3:00 PM 5/29/2021

Figure 39,40: Car Records Edit Jsp

The screenshot shows a web browser window titled "Edit Registration Form" with the URL localhost:8081/web_war_exploded/CarRecordUpdateEditServlet?regNumber=35. The page has a pink header bar. On the left, there is a sidebar with icons and labels: ADD SERVICE ENTRY, SEARCH ENTRY, BUDGET MANAGEMENT, INVENTORY MANAGEMENT, EMPLOYEE MANAGEMENT, CAR RENT, and CAR RECORD. The main content area is titled "Car Registration Edit Page". It contains the following form fields:

Registration Number	35
First Name	Thilina
Last name	Madushanka
NIC	971420464V
Address	dematagoda

At the bottom right of the page, there is a timestamp: 2:58 PM 5/29/2021.

The screenshot shows a web browser window titled "Edit Registration Form" with the URL localhost:8081/web_war_exploded/CarRecordUpdateEditServlet?regNumber=35. The page has a pink header bar. The main content area contains the following form fields:

email	t4ka@gmail.com
Vehicle Book number	yf6754
Vehicle Model	toyota-corolla
Vehicle seat amount	8
Distance	56000.0
Car Condition	zar

At the bottom right of the page, there is a blue button labeled "Update Details". At the bottom left, there is a copyright notice: "© 2021 by RASA. All rights reserved." The bottom of the screen shows a taskbar with various application icons and a timestamp: 2:58 PM 5/29/2021.

Car Rental Management

Figure 41: Available Cars

The screenshot shows a web-based car rental management system. On the left, a vertical sidebar menu lists several management modules: ADD SERVICE ENTRY, SEARCH ENTRY, BUDGET MANAGEMENT, INVENTORY MANAGEMENT, EMPLOYEE MANAGEMENT, CAR RENT, and CAR RECORD. The CAR RENT module is currently selected, indicated by a pink background. The main content area is titled "Cars Available" and features a search bar with placeholder text "Search.." and a magnifying glass icon. Below the search bar is a table with the following columns: Registration Number, Brand, Model, No of Seats, and Price Per Day. The table contains six rows of data, each with a "Rent Out" button on the right. The data is as follows:

Registration Number	Brand	Model	No of Seats	Price Per Day
1	Toyota	Corolla	4	4000.0
2	BMW	i8	2	10000.0
3	Suzuki	Swift	4	3000.0
4	Nissan	Altima	4	2000.0
5	Suzuki	Alto	4	2500.0
6	Honda	Vezel	4	3500.0

At the bottom of the sidebar, there is a copyright notice: "© 2021 by RASA. All rights reserved." The browser's address bar shows the URL "localhost:8080/web_war_exploded/availableCars.jsp". The system interface includes a standard Windows-style taskbar at the bottom with various icons and a system tray showing the date and time.

Figure 42: Rent Form

The screenshot shows a web browser window titled "Car Rental Form" with the URL "localhost:8080/web_war_exploded/RetrieveRegNoServlet". The page has a sidebar on the left with various management options: ADD SERVICE ENTRY, SEARCH ENTRY, BUDGET MANAGEMENT, INVENTORY MANAGEMENT, EMPLOYEE MANAGEMENT, CAR RENT (which is highlighted in pink), and CAR RECORD. The main content area is titled "Rent the Car" and contains a form with the following fields:

First name	<input type="text"/>
Last name	<input type="text"/>
E-mail	<input type="text"/> abc@gmail.com
Mobile Number	<input type="text"/> 1234567890
Address	<input type="text"/>
Pick up Date	<input type="text"/> mm/dd/yyyy <input type="button" value="..."/>
Drop off Date	<input type="text"/> mm/dd/yyyy <input type="button" value="..."/>
Rental Price	<input type="text"/>
Registration Number	<input type="text"/> 1

At the bottom of the form are two buttons: "Rent" and "Reset".

At the bottom of the browser window, the status bar shows "8:04 PM" and the date "5/29/2021".

Figure 43: Rental Details

The screenshot shows a web-based application titled "Rental Details" running on a local host. The interface includes a sidebar with various management options and a main content area displaying a table of rented car details.

Sidebar (Left):

- ADD SERVICE ENTRY
- SEARCH ENTRY
- BUDGET MANAGEMENT
- INVENTORY MANAGEMENT
- EMPLOYEE MANAGEMENT
- CAR RENT** (highlighted in pink)
- CAR RECORD

Main Content Area:

Rented Cars

Rent ID	First Name	Last Name	Email	Phone Number	Address	Pick up Date	Drop off Date	Rental Price	Registration Number	Update	Delete
25	Nimal	Perera	nimal@gmail.com	0789634247	Demuwatha, Rakwana	2021-04-28	2021-04-30	5000.0	5	<button>Update</button>	<button>Delete</button>
27	Sadun	Nalaka	sadun@gmail.com	0711540333	Madampe, Nugawela, Kahawatta	2021-05-29	2021-05-31	6000.0	1	<button>Update</button>	<button>Delete</button>
31	Kasun	Perera	kasun@gmail.com	0789624249	Kauduwawa, Atakalapanna	2021-05-18	2021-05-20	8000.0	1	<button>Update</button>	<button>Delete</button>
33	Nethmini	Gamage	nethmini@gmail.com	0777756777	Sannasgama, Ratnapura	2021-05-28	2021-05-30	3000.0	1	<button>Update</button>	<button>Delete</button>
34	Nadun	Weerathunga	nadun@gmail.com	0761234567	Kauduwawa, Atakalapanna	2021-05-18	2021-05-20	7000.0	5	<button>Update</button>	<button>Delete</button>
45	Saman	Samaranayaka	saman@gmail.com	0761234567	Kauduwawa, Atakalapanna	2021-05-27	2021-05-29	7000.0	3	<button>Update</button>	<button>Delete</button>

Select the year and month to generate the report

© 2021 by RASA. All rights reserved.

Windows taskbar icons: File Explorer, Mail, Edge, Google Chrome, Microsoft Word, Microsoft Excel, Microsoft Powerpoint, Firefox. System tray: 8:06 PM, 5/29/2021, battery level, signal strength.

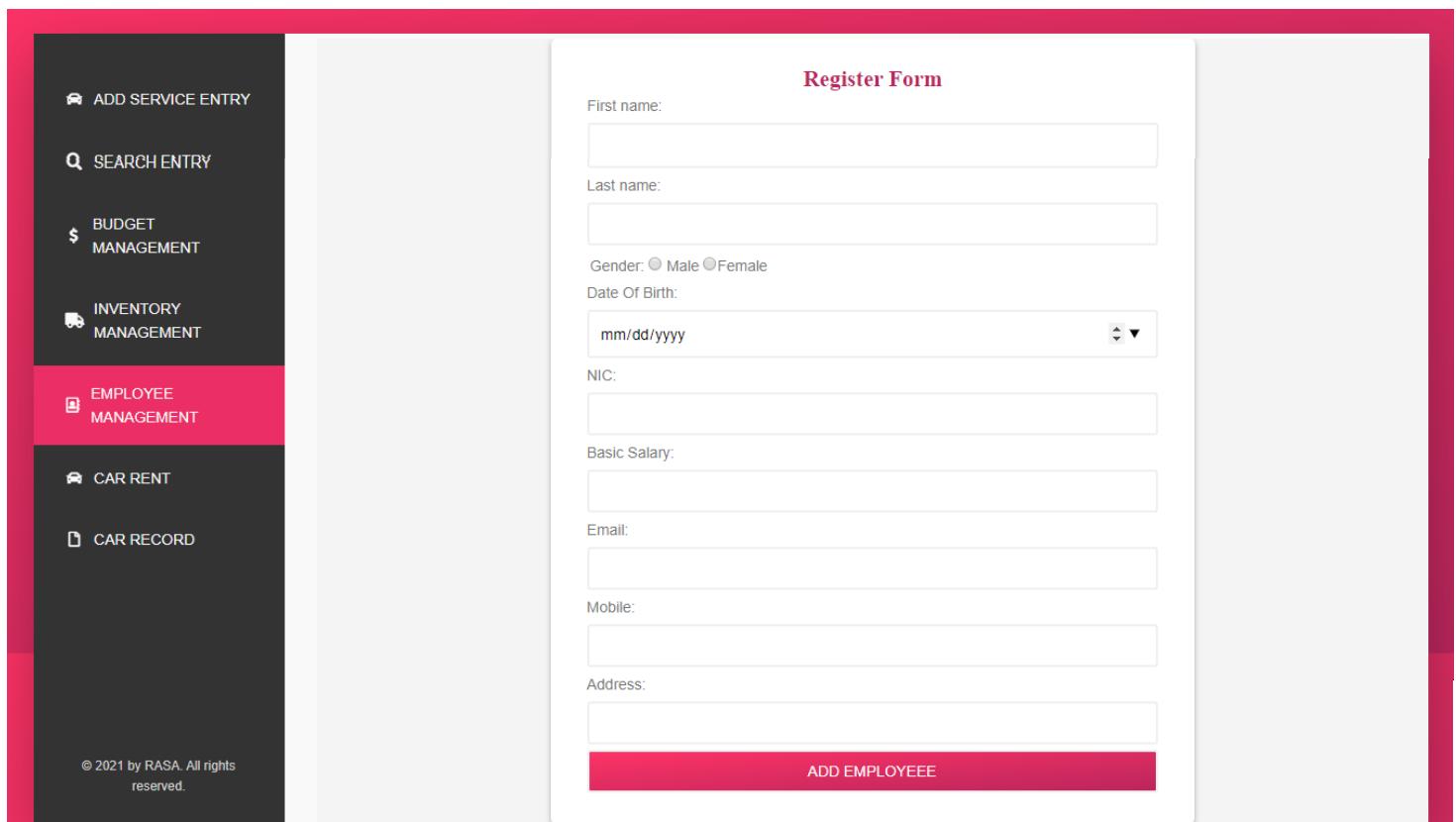
Figure 44: Update Rental Details

The screenshot shows a web application titled "Update Rental Details" running on a local host. The sidebar on the left contains links for various management tasks: ADD SERVICE ENTRY, SEARCH ENTRY, BUDGET MANAGEMENT, INVENTORY MANAGEMENT, EMPLOYEE MANAGEMENT, CAR RENT (which is highlighted in pink), and CAR RECORD. The main content area displays a form with the following fields and values:

First name	Nimal
Last name	Perera
E-mail	nimal@gmail.com
Mobile Number	0789634247
Address	Demuwatha, Rakwana
Pick up Date	04/28/2021
Drop off Date	04/30/2021
Rental Price	5000.0
Registration Number	5

At the bottom of the form are two buttons: "Update" and "Reset". The status bar at the bottom right of the browser window shows the time as 8:06 PM and the date as 5/29/2021.

Figure 45: employee registration



The image shows a user interface for employee registration. On the left, there is a vertical navigation menu with the following items:

- ADD SERVICE ENTRY
- SEARCH ENTRY
- BUDGET MANAGEMENT
- INVENTORY MANAGEMENT
- EMPLOYEE MANAGEMENT** (highlighted in pink)
- CAR RENT
- CAR RECORD

At the bottom of the menu, there is a copyright notice: © 2021 by RASA. All rights reserved.

The main content area is titled "Register Form". It contains the following fields:

- First name: [Text input field]
- Last name: [Text input field]
- Gender: Male Female
- Date Of Birth: [Text input field with a date picker icon]
- NIC: [Text input field]
- Basic Salary: [Text input field]
- Email: [Text input field]
- Mobile: [Text input field]
- Address: [Text input field]

A large pink button at the bottom right of the form area is labeled "ADD EMPLOYEE".

Figure 46: Employee details

EMPLOYEE DETAILS

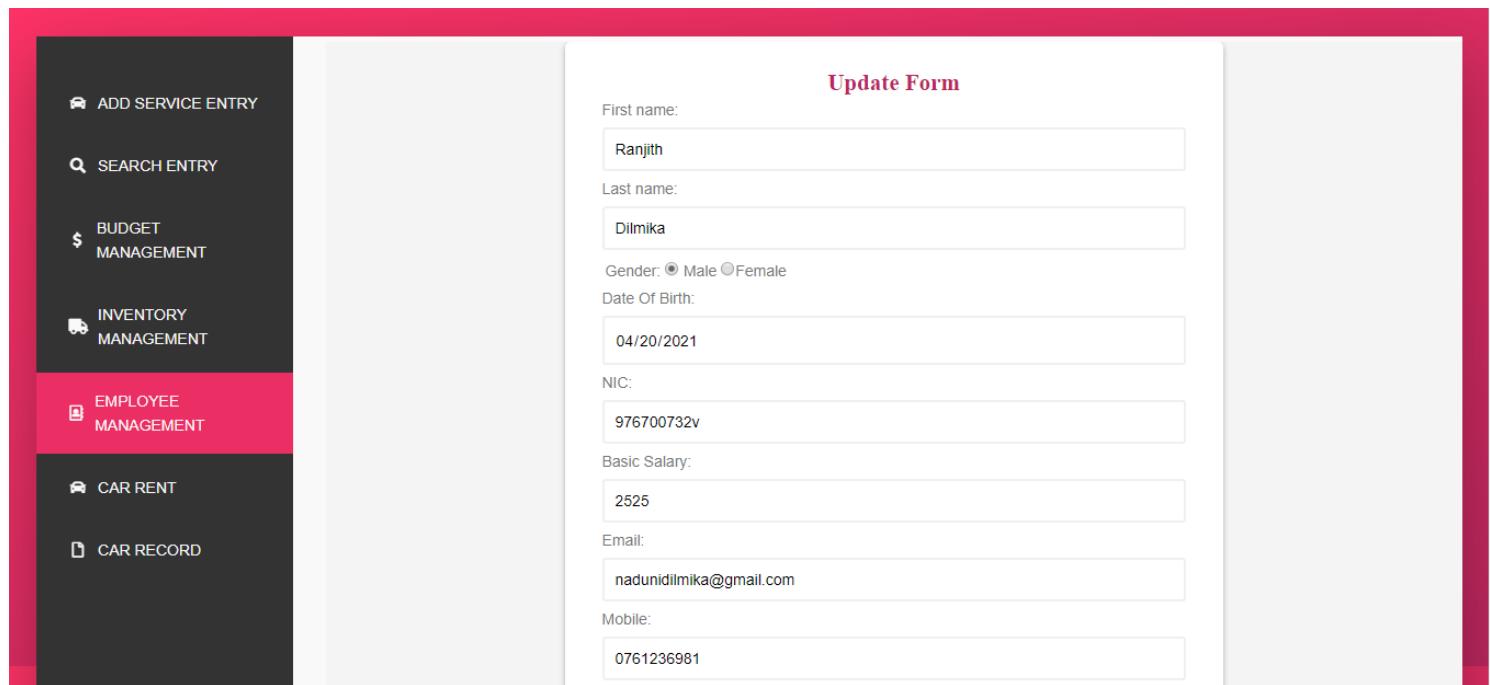
EMPLOYEE ID	EMPLOYEE NAME	MANAGE
1	Ranjith	
2	sanath	
3	vihangi	
4	Naduni	
5	kalana	
84	kithmini	
93	Tharushi	
95	Kamal	

ADD EMPLOYEE

ADD SERVICE ENTRY
SEARCH ENTRY
\$ BUDGET MANAGEMENT
INVENTORY MANAGEMENT
EMPLOYEE MANAGEMENT
CAR RENT
CAR RECORD

© 2021 by RASA. All rights reserved.

Figure 47: Employee update form



The image shows a user interface for an employee management system. On the left, there is a vertical navigation menu with the following items:

- ADD SERVICE ENTRY
- SEARCH ENTRY
- BUDGET MANAGEMENT
- INVENTORY MANAGEMENT
- EMPLOYEE MANAGEMENT** (highlighted in pink)
- CAR RENT
- CAR RECORD

The main content area is titled "Update Form". It contains the following fields:

- First name: Ranjith
- Last name: Dilmika
- Gender: Male Female
- Date Of Birth: 04/20/2021
- NIC: 976700732v
- Basic Salary: 2525
- Email: nadunidilmika@gmail.com
- Mobile: 0761236981

Figure 48: employee profile

The screenshot shows a mobile application interface. On the left is a vertical navigation bar with the following items:

- ADD SERVICE ENTRY
- SEARCH ENTRY
- BUDGET MANAGEMENT
- INVENTORY MANAGEMENT
- EMPLOYEE MANAGEMENT** (highlighted with a pink background)
- CAR RENT
- CAR RECORD

On the right is a detailed view of an employee profile titled "Employee Profile". The profile information is as follows:

Attribute	Value
ID	95
fname	Kamal
Iname	De Silva
Gender	male
Date of Birth	2021-05-14
NIC	123456789v
Basic Salary	25000.00
Mobile Number	1234567891
Email	kamal@gmail.com
Address	Kaduwella

Figure 49: employee salary and attendance management

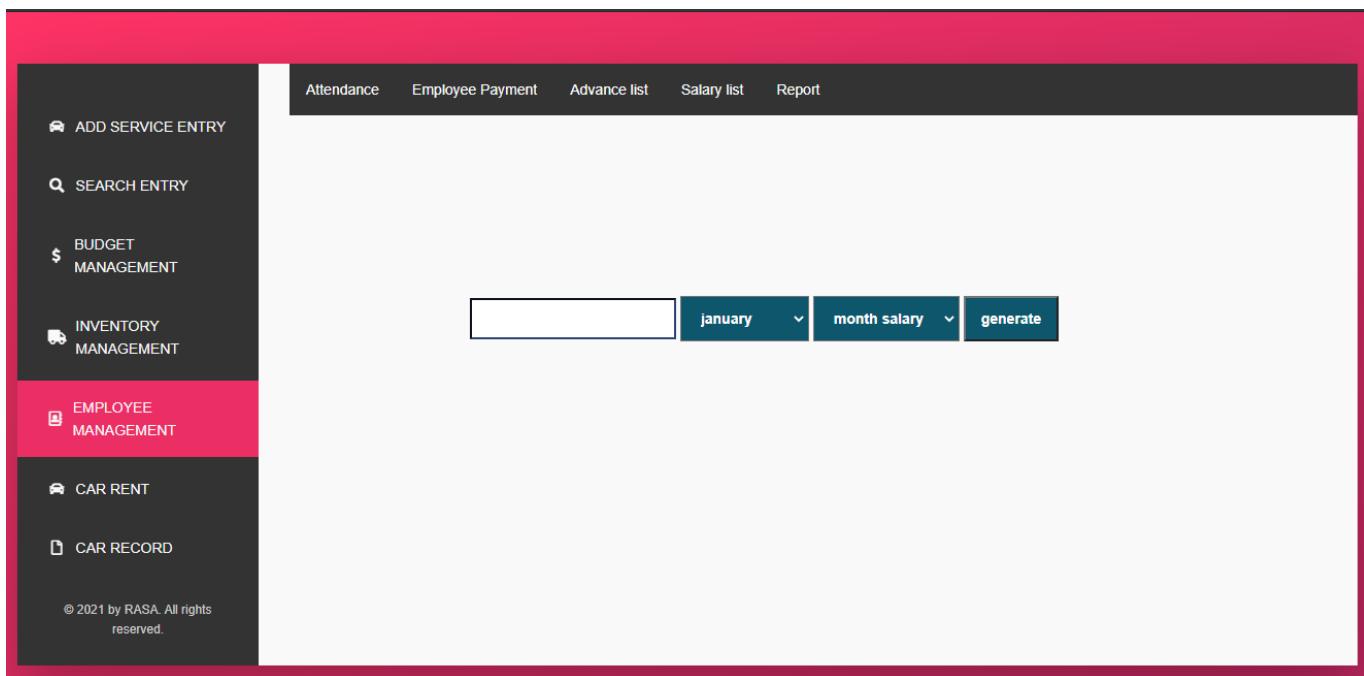


Figure 50: latest advance record

The screenshot shows a detailed view of the 'Latest advance record' for an employee. The main table displays a list of advance records with columns: ID, fname, lname, Advance, and payment. The data is as follows:

ID	fname	lname	Advance	payment
1	saman	kumara	advance	payment
2	kavindu	balasooriya	advance	payment
3	naduni	dilmika	advance	payment
4	saduni	dilmika	advance	payment
5	chanuka	dilum	advance	payment
6	naduni	dilmika	advance	payment
7	maalan	samual	advance	payment

To the right of the table, a modal window titled 'Latest advance record' shows the details of the last record: 'Id=null' and 'null'. It includes buttons for 'get', 'update', and 'delete'.

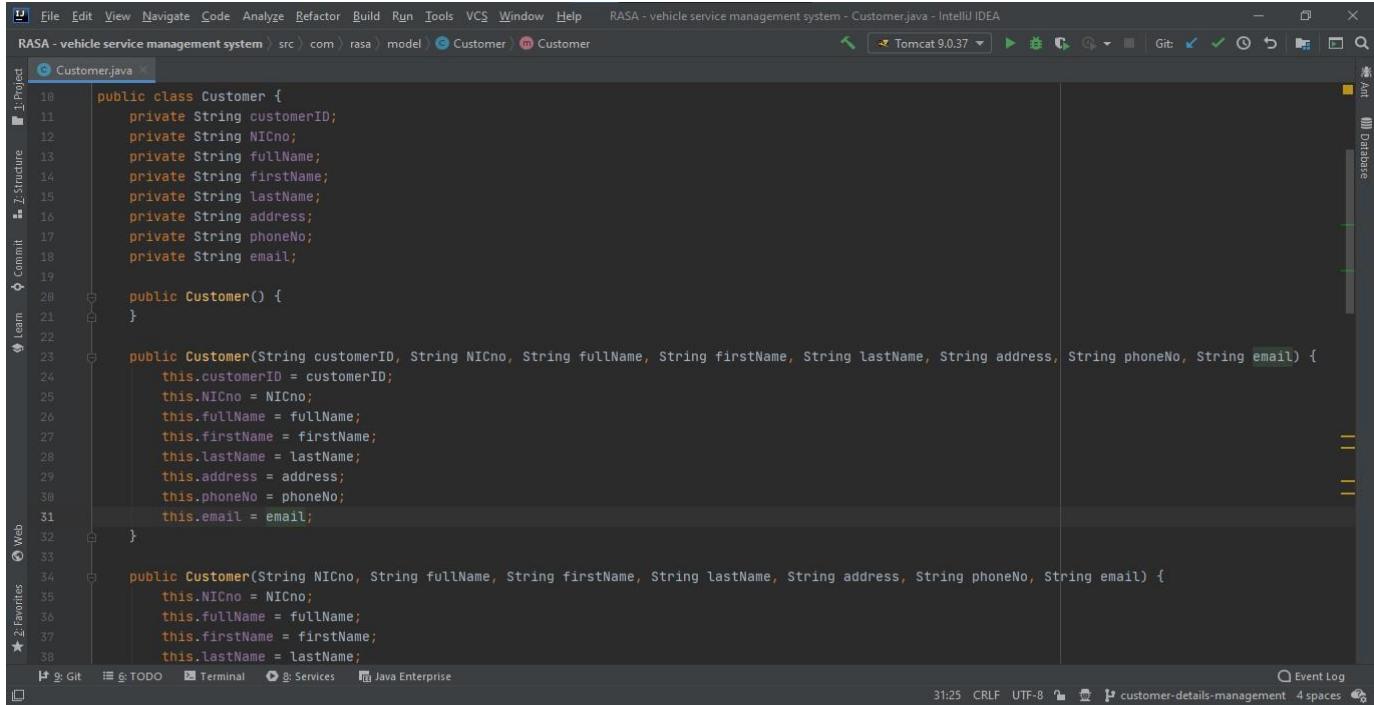
Figure 51: add Advance

The screenshot shows the RASA application interface. On the left is a sidebar with various management options: ADD SERVICE ENTRY, SEARCH ENTRY, BUDGET MANAGEMENT, INVENTORY MANAGEMENT, EMPLOYEE MANAGEMENT (which is highlighted in pink), CAR RENT, and CAR RECORD. At the bottom of the sidebar is a copyright notice: © 2021 by RASA. All rights reserved. The main content area has a dark header bar with links: Attendance, Employee Payment, Advance list, Salary list, and Report. A central modal window titled "ADD NEW ADVANCE" contains two input fields: "ID" (with value "null") and "advance" (with value "null"). Below the inputs is a teal-colored "save" button.

This screenshot shows the same RASA application interface as the first one. The sidebar and header are identical. A new modal window titled "ADD NEW PAYMENT" is open. It contains several input fields and labels: "paid advance" (with value "null"), "to pay" (with value "null"), "add bouns" (with value "null"), and "deduct" (with value "null"). Below these fields is a teal-colored "save" button.

Back- End Development

Model Classes



The screenshot shows the IntelliJ IDEA interface with the Customer.java file open. The code defines a Customer class with fields for customerID, NICno, fullName, firstName, lastName, address, phoneNo, and email. It includes a constructor taking all these parameters and two other constructors that initialize NICno and either fullName or firstName and lastName.

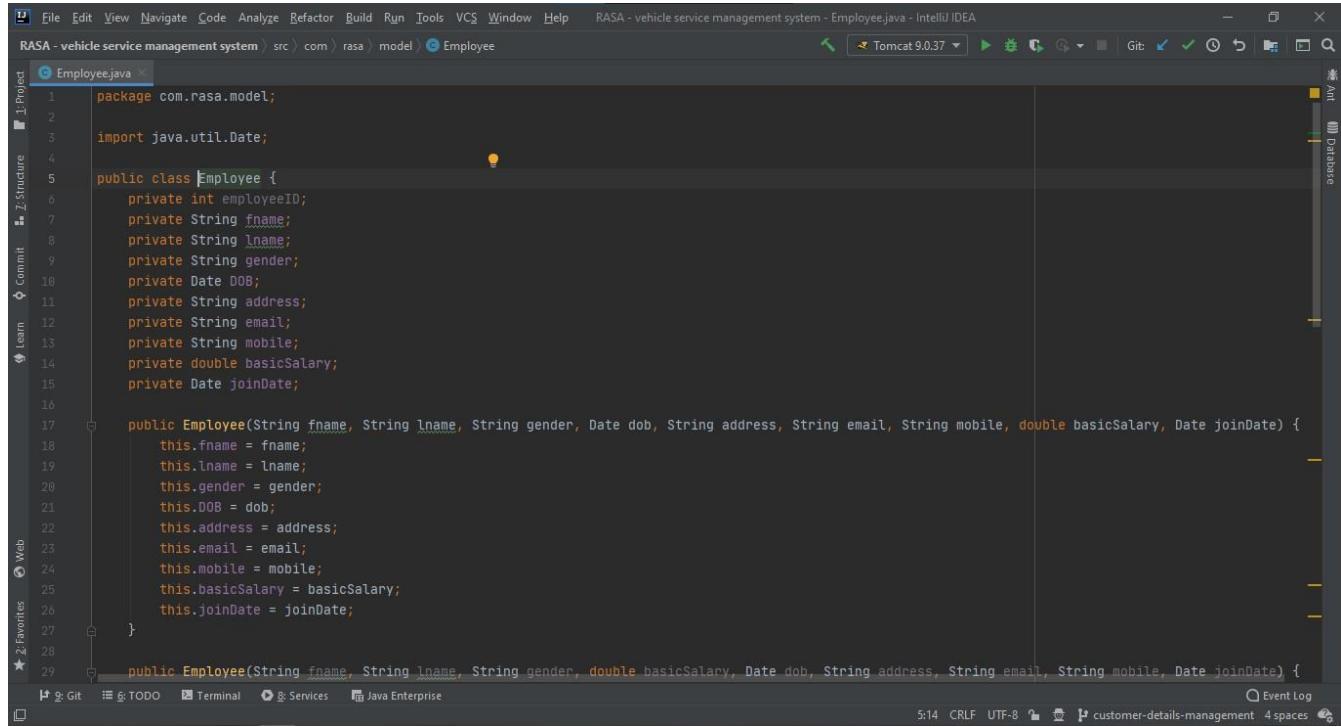
```
public class Customer {
    private String customerID;
    private String NICno;
    private String fullName;
    private String firstName;
    private String lastName;
    private String address;
    private String phoneNo;
    private String email;

    public Customer() {
    }

    public Customer(String customerID, String NICno, String fullName, String firstName, String lastName, String address, String phoneNo, String email) {
        this.customerID = customerID;
        this.NICno = NICno;
        this.fullName = fullName;
        this.firstName = firstName;
        this.lastName = lastName;
        this.address = address;
        this.phoneNo = phoneNo;
        this.email = email;
    }

    public Customer(String NICno, String fullName, String firstName, String lastName, String address, String phoneNo, String email) {
        this.NICno = NICno;
        this.fullName = fullName;
        this.firstName = firstName;
        this.lastName = lastName;
    }
}
```

Figure 52: customer class



The screenshot shows the IntelliJ IDEA interface with the Employee.java file open. The code defines an Employee class with fields for employeeID, fname, lname, gender, DOB, address, email, mobile, basicSalary, and joinDate. It includes a constructor taking fname, lname, gender, DOB, address, email, mobile, basicSalary, and joinDate, and another constructor taking fname, lname, gender, basicSalary, DOB, address, email, mobile, and joinDate.

```
package com.rasa.model;

import java.util.Date;

public class Employee {
    private int employeeID;
    private String fname;
    private String lname;
    private String gender;
    private Date DOB;
    private String address;
    private String email;
    private String mobile;
    private double basicSalary;
    private Date joinDate;

    public Employee(String fname, String lname, String gender, Date dob, String address, String email, String mobile, double basicSalary, Date joinDate) {
        this.fname = fname;
        this.lname = lname;
        this.gender = gender;
        this.DOB = dob;
        this.address = address;
        this.email = email;
        this.mobile = mobile;
        this.basicSalary = basicSalary;
        this.joinDate = joinDate;
    }

    public Employee(String fname, String lname, String gender, double basicSalary, Date dob, String address, String email, String mobile, Date joinDate) {
    }
}
```

Figure 53: Employee class

The screenshot shows the IntelliJ IDEA interface with the EmployeeAdvance.java file open. The code defines a class EmployeeAdvance with private fields advanceID, empID, adminID, date, and amount, and corresponding getters and setters.

```
1 package com.rasa.model;
2
3 import java.sql.Date;
4
5 public class EmployeeAdvance {
6
7     private int advanceID;
8     private int empID;
9     private int adminID;
10    private Date date;
11    private double amount;
12
13    public EmployeeAdvance(int empID, int adminID, Date date, double amount) {
14        this.empID = empID;
15        this.adminID = adminID;
16        this.date = date;
17        this.amount = amount;
18    }
19
20    //getters
21
22    public int getEmpID() { return empID; }
23
24    public int getAdminID() { return adminID; }
25
26    public Date getDate() { return date; }
27
28    public double getAmount() { return amount; }
29
30
31
32
33
34
35
36
37 }
```

Figure 54: Employee Advance class

The screenshot shows the IntelliJ IDEA interface with the Repair.java file open. The code defines a class Repair with private fields repairID, NICno, vehicleRegistrationNo, entryDate, accidentDate, paymentType, customerNoObjection, and leasingNoObjection, and corresponding getters and setters.

```
1 package com.rasa.model;
2
3 public class Repair {
4
5     private int repairID;
6
7     public Repair() {
8     }
9
10    public Repair(String NICno, String vehicleRegistrationNo, String entryDate, String accidentDate, String paymentType, boolean customerNoObjection, boolean leasingNoObjection) {
11        this.NICno = NICno;
12        this.vehicleRegistrationNo = vehicleRegistrationNo;
13        this.entryDate = entryDate;
14        this.accidentDate = accidentDate;
15        this.paymentType = paymentType;
16        this.customerNoObjection = customerNoObjection;
17        this.leasingNoObjection = leasingNoObjection;
18        this.claimForm = claimForm;
19    }
20
21    public String getNICno() { return NICno; }
22
23    public void setNICno(String NICno) { this.NICno = NICno; }
24
25    public String getVehicleRegistrationNo() { return vehicleRegistrationNo; }
26
27    public void setVehicleRegistrationNo(String vehicleRegistrationNo) {
28        this.vehicleRegistrationNo = vehicleRegistrationNo;
29    }
30
31    public String getEntryDate() { return entryDate; }
32
33    public void setEntryDate(String entryDate) { this.entryDate = entryDate; }
34
35
36
37 }
```

Figure 55: Repair class

The screenshot shows the IntelliJ IDEA interface with the 'WorkProgress.java' file open. The code defines a class 'WorkProgress' with private fields for pid, sid, and tot_EstimateAmount, and methods for setting and getting these values.

```
1 package com.rasa.model;
2
3 public class WorkProgress {
4     private String pid;
5     private int sid;
6     private double tot_EstimateAmount;
7
8     public WorkProgress() {
9     }
10
11    public WorkProgress(String pid, int sid, double tot_EstimateAmount) {
12        this.pid = pid;
13        this.sid = sid;
14        this.tot_EstimateAmount = tot_EstimateAmount;
15    }
16
17    public WorkProgress(String pid, int sid) {
18        this.pid = pid;
19        this.sid = sid;
20    }
21
22    public String getPid() { return pid; }
23
24    public int getSid() { return sid; }
25
26    public double getTot_EstimateAmount() { return tot_EstimateAmount; }
27
28    public void setPid(String pid) { this.pid = pid; }
29
30    public void setSid(int sid) { this.sid = sid; }
31
32    public void setTot_EstimateAmount(double tot_EstimateAmount) { this.tot_EstimateAmount = tot_EstimateAmount; }
33
34}
```

Figure 56: work progress class

The screenshot shows the IntelliJ IDEA interface with the 'Vehicle.java' file open. The code defines a class 'Vehicle' with private fields for NICno, registrationNo, year, model, brand, and color, and methods for setting and getting these values.

```
1 package com.rasa.model;
2
3 public class Vehicle {
4     private String NICno;
5     private String registrationNo;
6     private int year;
7     private String model;
8     private String brand;
9     private String color;
10
11    public Vehicle() {
12    }
13
14    public Vehicle(
15        String NICno,
16        String registrationNo,
17        int year,
18        String model,
19        String brand,
20        String color
21    ) {
22        this.NICno = NICno;
23        this.registrationNo = registrationNo;
24        this.year = year;
25        this.model = model;
26        this.brand = brand;
27        this.color = color;
28    }
29
30    public Vehicle(String registrationNo, int year, String model, String brand, String color) {
31    }
32
33    public String getNICno() { return NICno; }
34    public void setNICno(String NICno) { this.NICno = NICno; }
35
36    public String getRegistrationNo() { return registrationNo; }
37    public void setRegistrationNo(String registrationNo) { this.registrationNo = registrationNo; }
38
39    public int getYear() { return year; }
40    public void setYear(int year) { this.year = year; }
41
42    public String getModel() { return model; }
43    public void setModel(String model) { this.model = model; }
44
45    public String getBrand() { return brand; }
46    public void setBrand(String brand) { this.brand = brand; }
47
48    public String getColor() { return color; }
49    public void setColor(String color) { this.color = color; }
50
51}
```

Figure 57: Vehicle class

The screenshot shows the IntelliJ IDEA interface with the file `paymentList.java` open. The code defines a class `paymentList` with various fields and methods. The class has a constructor that takes parameters like `payId`, `registrationNumber`, `estimateAmount`, `cash`, `paymentDate`, and `sid`. It also has a method `paymentList` that takes a `String registrationNumber`, `double estimateAmount`, `double cash`, `Date paymentDate`, and `int sid`.

```
package com.rasa.model;

import java.sql.Date;

public class paymentList {

    int payId;
    String registrationNumber;
    String customerName;
    double estimateAmount;
    double cash;
    Date paymentDate;
    int sid;

    public paymentList(){

    }

    public paymentList(int payId, String registrationNumber, double estimateAmount, double cash, Date paymentDate, int sid) {
        this.payId = payId;
        this.registrationNumber = registrationNumber;

        this.estimateAmount = estimateAmount;

        this.cash = cash;
        this.paymentDate = paymentDate;
        this.sid = sid;
    }

    public paymentList(String registrationNumber, double estimateAmount, double cash, Date paymentDate, int sid) {
        this.registrationNumber = registrationNumber;
    }
}
```

Figure 58: payment class

The screenshot shows the IntelliJ IDEA interface with the file `Budget.java` open. The code defines a class `Budget` with various fields and methods. The class has a constructor that takes a `String year`, `String month`, and `double totIncome`. It also has a method `Budget` that takes a `double repairAmount`.

```
package com.rasa.model;

import java.util.Date;

public class Budget {

    String year;
    String month;
    double totIncome;

    double repairAmount;
    double rentalAmount;
    double inventoryExpenses;
    double empPayments;
    double totExpenses;
    double profit;

    public Budget(){

    }

    public Budget(double repairAmount) { this.repairAmount = repairAmount; }

    public Budget(String year, String month, double totIncome, double repairAmount, double rentalAmount, double totExpenses, double inventoryExpenses) {
        this.year = year;
        this.month = month;
        this.totIncome = totIncome;

        this.repairAmount = repairAmount;
        this.rentalAmount = rentalAmount;
        this.totExpenses = totExpenses;
        this.inventoryExpenses = inventoryExpenses;
        this.empPayments = empPayments;
    }
}
```

Figure 59: Budget class

Service classes

The screenshot shows the IntelliJ IDEA interface with the file `ClientService.java` open. The code implements the `IClientService` interface and contains a method `searchByNic` which performs a database query to find a customer by their NIC number.

```
18  public class ClientService implements IClientService{  
19  
20  
21     /** REFERENCE FOR CONNECTION **/  
22     private static Connection conn;  
23  
24     private PreparedStatement preparedStatement;  
25  
26  
27     @Override  
28     public Customer searchByNic(String nic) {  
29         Customer customer = new Customer();  
30  
31         try{  
32             conn = DBConnectionUtil.getConnection();  
33             String sql = CustomerManagementQuery.SEARCH_CLIENT_BY_NIC;  
34             preparedStatement = conn.prepareStatement(sql);  
35  
36             preparedStatement.setString(QueryConstants.COLUMN_ONE,nic.toUpperCase());  
37  
38             ResultSet resultSet = preparedStatement.executeQuery();  
39  
40             while(resultSet.next()){  
41                 customer.setNIC(resultSet.getString( columnLabel: "NICnumber"));  
42                 customer.setFirstName(resultSet.getString( columnLabel: "firstName"));  
43                 customer.setLastName(resultSet.getString( columnLabel: "lastName"));  
44                 customer.setAddress(resultSet.getString( columnLabel: "address"));  
45                 customer.setPhoneNo(resultSet.getString( columnLabel: "mobile"));  
46                 customer.setEmail(resultSet.getString( columnLabel: "email"));  
47             }  
48         } catch (SQLException e) {  
49             e.printStackTrace();  
50         }  
51     }  
52  
53 }
```

Figure 60: Client Service

The screenshot shows the IntelliJ IDEA interface with the file `employeeService.java` open. The code defines two methods: `insertEmployee` which inserts a new employee record into the database, and `viewEmployee` which retrieves all employee records.

```
41  public boolean insertEmployee(Employee employee) throws SQLException {  
42      String sql = "INSERT INTO employee (fname, lname, gender,DOB,address,email,mobile,basicSalary,joinDate) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);";  
43      connect();  
44  
45      PreparedStatement statement = jdbcConnection.prepareStatement(sql);  
46      statement.setString( parameterIndex: 1, employee.getFname());  
47      statement.setString( parameterIndex: 2, employee.getName());  
48      statement.setString( parameterIndex: 3, employee.getGender());  
49      statement.setDate( parameterIndex: 4, employee.getDOB());  
50      statement.setString( parameterIndex: 5, employee.getAddress());  
51      statement.setString( parameterIndex: 6, employee.getEmail());  
52      statement.setString( parameterIndex: 7, employee.getMobile());  
53      statement.setDouble( parameterIndex: 8, employee.getBasicSalary());  
54      statement.setDate( parameterIndex: 9, employee.getJoinDate());  
55  
56      boolean rowInserted = statement.executeUpdate() > 0;  
57      statement.close();  
58      disconnect();  
59      return rowInserted;  
60  }  
61  
62  public List<Employee> viewEmployee() throws SQLException {  
63      List<Employee> viewEmployee = new ArrayList<>();  
64  
65      String sql = "SELECT * FROM employee";  
66  
67      connect();  
68  
69      Statement statement = jdbcConnection.createStatement();
```

Figure 61: employee service

The screenshot shows the IntelliJ IDEA interface with the EmployeeAdvanceService.java file open. The code implements the IEmpAdv interface and adds advances to a database. It uses try-catch blocks to handle SQLExceptions and ClassNotFoundExceptions.

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA - vehicle service management system - EmployeeAdvanceService.java - IntelliJ IDEA
RASA - vehicle service management system > src > com > rasa > service > EmployeeAdvanceService
EmployeeAdvanceService.java
1 package com.rasa.service;
2
3
4 import ...
5
6
7 public class EmployeeAdvanceService implements IEmpAdv {
8
9     private static Connection con;
10    private PreparedStatement preparedStatement;
11
12    @Override
13    public boolean addAdvance(int empID, int adminID, Date date, double amount) {
14
15        try{
16            con = DBConnectionUtil.getConnection();
17            String sql= EmpQuery.add_adv;
18            preparedStatement=con.prepareStatement(sql);
19
20            preparedStatement.setInt( parameterIndex: 1,empID);
21            preparedStatement.setInt( parameterIndex: 2,adminID);
22            preparedStatement.setDate( parameterIndex: 3,date);
23            preparedStatement.setDouble( parameterIndex: 4,amount);
24
25            preparedStatement.execute();
26
27        } catch (SQLException | ClassNotFoundException throwables) {
28            throwables.printStackTrace();
29            return false;
30        }
31
32    }
33
34 }
35
36
37
38
39
40
41
42
43
```

Figure 62: Employee Advance service

The screenshot shows the IntelliJ IDEA interface with the Workprogress_service.java file open. It contains methods for getting and creating progress IDs, and an InsertService function.

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA - vehicle service management system - Workprogress_service.java - IntelliJ IDEA
RASA - vehicle service management system > src > com > rasa > service > Workprogress_service
EmployeeAdvanceService.java | employeeService.java | Workprogress_service.java
Workprogress_service.java
13
14    @Override
15    public String getProgressId(int sid) throws SQLException, ClassNotFoundException {
16        //initialized pid
17        String pid = null;
18        Connection con = DBConnectionUtil.getConnection();
19        String sql = "Select pid from workprogress where sid = ? ";
20
21        PreparedStatement preparedStatement = con.prepareStatement(sql);
22        preparedStatement.setInt( parameterIndex: 1,sid);
23
24        ResultSet S_pid = preparedStatement.executeQuery();
25        while(S_pid.next()){
26            pid = S_pid.getString( columnIndex: 1);
27
28        }
29
30        return pid;
31    }
32
33    @Override
34    public String createProgressId(int sid) { return "p"+sid; }
35
36    //insert function
37    @Override
38    public void InsertService(RemoteService repairService) {
39        //I have to implement
40    }
41
42
43
```

Figure 63:work progress service

The screenshot shows the IntelliJ IDEA interface with the 'VehicleService.java' file open. The code implements a service method to search for vehicles by registration number. It uses JDBC to execute a prepared statement and map the results to a vehicle object. Error handling includes catching SQLException and ClassNotFoundException, and closing the connection.

```
26  
27     @Override  
28     public Vehicle searchByRegistrationNumber(String registrationNum) {  
29  
30         Vehicle vehicle = new Vehicle();  
31  
32         try {  
33             conn = DBConnectionUtil.getConnection();  
34             String sql = CustomerManagementQuery.SEARCH_VEHICLE_BY_REGISTRATION_NUMBER;  
35             preparedStatement = conn.prepareStatement(sql);  
36  
37             preparedStatement.setString(QueryConstants.COLUMN_ONE, registrationNum.toUpperCase());  
38             ResultSet resultSet = preparedStatement.executeQuery();  
39  
40             while (resultSet.next()) {  
41                 vehicle.setRegistrationNo(resultSet.getString(columnLabel: "registrationNumber"));  
42                 vehicle.setBrand(resultSet.getString(columnLabel: "brand"));  
43                 vehicle.setColor(resultSet.getString(columnLabel: "color"));  
44                 vehicle.setModel(resultSet.getString(columnLabel: "model"));  
45                 vehicle.setYear(resultSet.getInt(columnLabel: "manufacturedYear"));  
46             }  
47         }  
48  
49         }catch (SQLException | ClassNotFoundException e){  
50             e.printStackTrace();  
51         }finally {  
52             DBConnectionUtil.closeConnection(preparedStatement, conn);  
53         }  
54     }
```

Figure 64: vehicle service

The screenshot shows the IntelliJ IDEA interface with the 'ServiceEntry.java' file open. The code implements a service method to add a service entry. It checks the service type and uses different SQL queries for insurance or cash services, setting various parameters like registration number, entry date, accident date, and service type.

```
38  
39     @Override  
40     public boolean addServiceEntry(String registrationNumber, String nic, String ServiceType, String entryDate, String accidentDate, boolean customerNoObject)  
41     {  
42         try {  
43             conn = DBConnectionUtil.getConnection();  
44             String sql = "";  
45  
46             if (ServiceType.equals("insurance")){  
47                 sql = CustomerManagementQuery.ADD_CASH_SERVICE;  
48                 preparedStatement = conn.prepareStatement(sql);  
49                 preparedStatement.setString(QueryConstants.COLUMN_ONE, registrationNumber);  
50                 preparedStatement.setString(QueryConstants.COLUMN_TWO, entryDate);  
51                 preparedStatement.setString(QueryConstants.COLUMN_THREE, accidentDate);  
52                 preparedStatement.setString(QueryConstants.COLUMN_FOUR, ServiceType);  
53                 preparedStatement.setString(QueryConstants.COLUMN_FIVE, nic);  
54                 preparedStatement.setString(QueryConstants.COLUMN_SIX, "Incomplete");  
55             }else {  
56                 sql = CustomerManagementQuery.ADD_INSURANCE_SERVICE;  
57                 preparedStatement = conn.prepareStatement(sql);  
58                 preparedStatement.setString(QueryConstants.COLUMN_ONE, registrationNumber);  
59                 preparedStatement.setString(QueryConstants.COLUMN_TWO, entryDate);  
60                 preparedStatement.setString(QueryConstants.COLUMN_THREE, accidentDate);  
61                 preparedStatement.setBoolean(QueryConstants.COLUMN_FOUR, customerNoObject);  
62                 preparedStatement.setBoolean(QueryConstants.COLUMN_FIVE, insuranceNoObject);  
63                 preparedStatement.setString(QueryConstants.COLUMN_SIX, claimForm);  
64                 preparedStatement.setString(QueryConstants.COLUMN_SEVEN, ServiceType);  
65                 preparedStatement.setString(QueryConstants.COLUMN_EIGHT, nic);  
66                 preparedStatement.setString(QueryConstants.COLUMN_NINE, "Incomplete");  
67             }  
68         }  
69     }
```

Figure 65: service entry service

```

public class budgetService {

    static Connection connection;

    //calculate total repair amount
    public double CalRepairAmount(String year, String month) {
        double repairAmount = 0;
        double totalAmount = 0;
        try {
            connection = DBConnectionUtil.getConnection();
            PreparedStatement preparedStatement = connection.prepareStatement(sql:"SELECT cash FROM rasa.payment WHERE EXTRACT(year FROM paymentDate) = ? AND EXTRACT(month FROM paymentDate) = ?");
            preparedStatement.setString(parameterIndex: 1, year);
            preparedStatement.setString(parameterIndex: 2, month);

            ResultSet resultSet = preparedStatement.executeQuery();

            while (resultSet.next()) {
                repairAmount = resultSet.getDouble(columnIndex: 1);
                totalAmount = totalAmount + repairAmount;
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return totalAmount;
    }
}

```

Figure 66: budget service

```

import ...

public class paymentService implements ipayment{

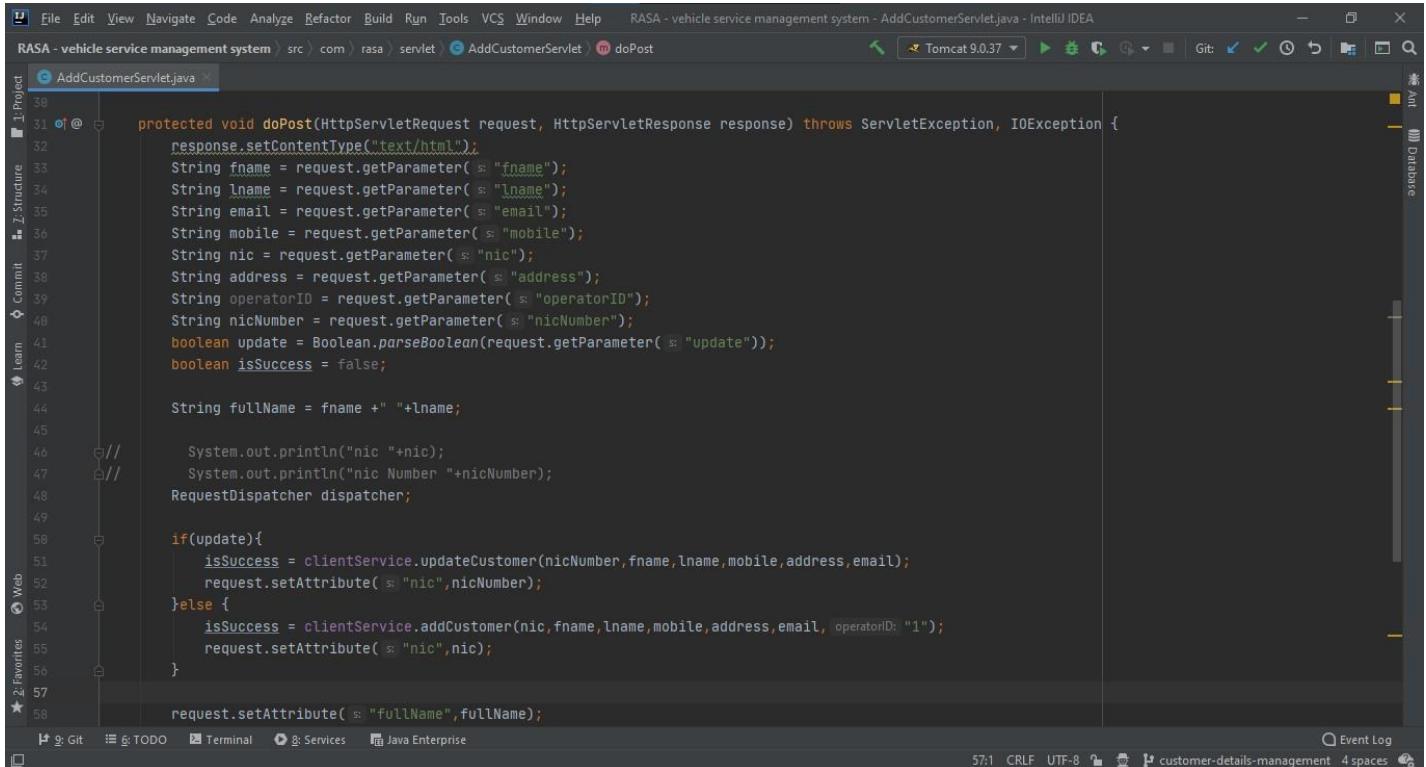
    public paymentService() {
    }

    public Connection connection;
    // method for insert payments
    public void addPayment(paymentList paymentlist) throws SQLException {
        try{
            connection = DBConnectionUtil.getConnection();
            PreparedStatement preparedStatement = connection.prepareStatement(sql: "INSERT INTO rasa.payment "+ "(registrationNumber,estimateAmount,actualAmount,cash,paymentDate,sid)");
            preparedStatement.setString(parameterIndex: 1,paymentlist.getRegistrationNumber());
            preparedStatement.setDouble(parameterIndex: 2,paymentlist.getEstimateAmount());
            preparedStatement.setDouble(parameterIndex: 3,paymentlist.getCash());
            preparedStatement.setDate(parameterIndex: 4, paymentlist.getPaymentDate());
            preparedStatement.setInt(parameterIndex: 5, paymentlist.getSid());
            System.out.println(preparedStatement);
            preparedStatement.executeUpdate();
        }catch(SQLException | ClassNotFoundException e){
            printSQLException((SQLException) e);
        }
    }
}

```

Figure 67: payment service

Servlet classes



```

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html");
    String fname = request.getParameter("fname");
    String lname = request.getParameter("lname");
    String email = request.getParameter("email");
    String mobile = request.getParameter("mobile");
    String nic = request.getParameter("nic");
    String address = request.getParameter("address");
    String operatorID = request.getParameter("operatorID");
    String nicNumber = request.getParameter("nicNumber");
    boolean update = Boolean.parseBoolean(request.getParameter("update"));
    boolean isSuccess = false;

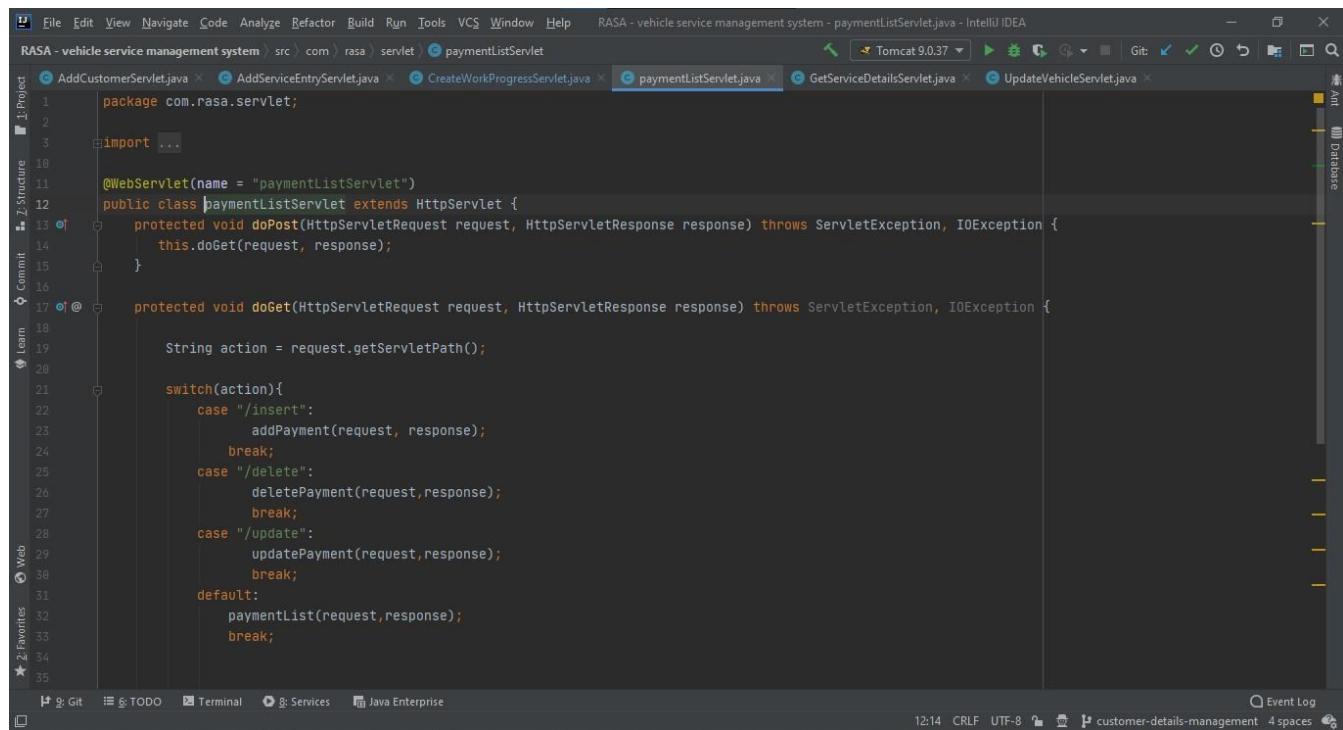
    String fullName = fname + " " + lname;
    System.out.println("nic " + nic);
    System.out.println("nic Number " + nicNumber);
    RequestDispatcher dispatcher;

    if(update){
        isSuccess = clientService.updateCustomer(nicNumber, fname, lname, mobile, address, email);
        request.setAttribute("nic", nicNumber);
    }else {
        isSuccess = clientService.addCustomer(nic, fname, lname, mobile, address, email, operatorID: "1");
        request.setAttribute("nic", nic);
    }

    request.setAttribute("fullName", fullName);
}

```

Figure 68: Add Customer



```

package com.rasa.servlet;

import ...

@WebServlet(name = "paymentListServlet")
public class paymentListServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        this.doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        String action = request.getServletPath();

        switch(action){
            case "/insert":
                addPayment(request, response);
                break;
            case "/delete":
                deletePayment(request, response);
                break;
            case "/update":
                updatePayment(request, response);
                break;
            default:
                paymentList(request, response);
                break;
        }
    }
}

```

Figure 69: Payment Servlet

The screenshot shows the IntelliJ IDEA interface with the file `AddServiceEntryServlet.java` open. The code implements a `HttpServlet` for handling POST requests. It retrieves parameters from the request, including registration number, entry date, accident date, repair status, customer no objection, insurance no objection, claim form, and NIC. It then prints the NIC number to the console. The code handles exceptions and attempts to catch `NullPointerException`. The IDE's navigation bar shows other files like `AddCustomerServlet.java`, `CreateWorkProgressServlet.java`, etc.

```
29 }  
30 }  
31 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
32     response.setContentType("text/html");  
33     String registrationNumber = request.getParameter("registrationNumber");  
34     String entryDate = request.getParameter("entryDate");  
35     String accidentDate = request.getParameter("accidentDate");  
36     String repair = request.getParameter("repair");  
37     String customerNoObjection = request.getParameter("customerNoObjection");  
38     String insuranceNoObjection = request.getParameter("insuranceNoObjection");  
39     String claimForm = request.getParameter("claimForm");  
40     String nic = request.getParameter("nicNumber");  
41     boolean hasCustomerNoObjection = false;  
42     boolean hasInsuranceNoObjection = false;  
43     boolean hasClaimForm = false;  
44     boolean isSuccess = false;  
45  
46     System.out.println("nic " + nic);  
47  
48     try {  
49         if (customerNoObjection.equals("on")){  
50             hasCustomerNoObjection = true;  
51         }  
52     }catch (NullPointerException e){  
53         hasCustomerNoObjection = false;  
54     }  
55  
56     try {  
57         if (insuranceNoObjection.equals("on")){  
58             hasInsuranceNoObjection = true;  
59         }  
60     }catch (NullPointerException e){  
61         hasInsuranceNoObjection = false;  
62     }  
63  
64     try {  
65         if (claimForm.equals("on")){  
66             hasClaimForm = true;  
67         }  
68     }catch (NullPointerException e){  
69         hasClaimForm = false;  
70     }  
71  
72     if (hasCustomerNoObjection && hasInsuranceNoObjection && hasClaimForm){  
73         isSuccess = true;  
74     }  
75  
76     if (isSuccess){  
77         response.sendRedirect("success.jsp");  
78     }  
79     else{  
80         response.sendRedirect("failure.jsp");  
81     }  
82 }  
83  
84 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
85 }
```

Figure 70: Add service Entry

The screenshot shows the IntelliJ IDEA interface with the file `CreateWorkProgressServlet.java` open. This servlet handles both POST and GET requests. For POST, it gathers a sid (service ID), creates a new progress ID, and forwards the request to `add_services.jsp`. For GET, it simply forwards the request to the same page. The code includes annotations for `@WebServlet` and `HttpServlet`.

```
1 package com.rasa.servlet;  
2  
3 import ...  
4  
5 @WebServlet("/CreateWorkProgressServlet")  
6 public class CreateWorkProgressServlet extends HttpServlet {  
7     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
8         //gather sid for create new progress Id  
9  
10        try{  
11            int sid = Integer.parseInt(request.getParameter("sid"));  
12            Iworkprogress_service iworkprogress_service = new Workprogress_service();  
13            String pid = iworkprogress_service.createProgressId(sid);  
14            request.setAttribute("pid",pid);  
15            request.getRequestDispatcher("add_services.jsp").forward(request,response);  
16  
17        }catch (NullPointerException n){  
18            System.out.println("something wrong !");  
19        }  
20    }  
21  
22    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
23    }  
24 }
```

Figure 71: create work progress

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA - vehicle service management system - UpdateVehicleServlet.java - IntelliJ IDEA
RASA - vehicle service management system > src > com > rasa > servlet > UpdateVehicleServlet > init
Tomcat 9.0.37
AddCustomerServlet.java AddServiceEntryServlet.java CreateWorkProgressServlet.java paymentListServlet.java GetServiceDetailsServlet.java UpdateVehicleServlet.java
Project Z-Structure Commit Learn Favorites Web
17
18     @WebServlet("/UpdateVehicleServlet")
19     public class UpdateVehicleServlet extends HttpServlet {
20
21         private VehicleService vehicleService;
22         private Repair repair;
23         private Vehicle vehicle;
24         private Customer customer;
25         private ServiceEntry serviceEntry;
26         private ClientService clientService;
27
28         @Override
29         public void init() throws ServletException {
30             super.init();
31             vehicleService = new VehicleService();
32             serviceEntry = new ServiceEntry();
33             clientService = new ClientService();
34             repair = new Repair();
35             vehicle = new Vehicle();
36             customer = new Customer();
37         }
38
39         protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
40             response.setContentType("text/html");
41             String regID = request.getParameter("registrationNumber");
42             String brand = request.getParameter("brand");
43             String model = request.getParameter("model");
44             String color = request.getParameter("color");
45             int serviceID = Integer.parseInt(request.getParameter("serviceID"));
46
47             serviceEntry = new ServiceEntry();
48             clientService = new ClientService();
49             vehicleService = new VehicleService();
50             repair = new Repair();
51             vehicle = new Vehicle();
52             customer = new Customer();
53
54             vehicle.setRegistrationNumber(regID);
55             vehicle.setBrand(brand);
56             vehicle.setModel(model);
57             vehicle.setColor(color);
58
59             vehicleService.updateVehicle(vehicle);
60
61             request.setAttribute("repair", repair);
62
63             response.sendRedirect("index.jsp");
64         }
65     }

```

Figure 72: Upload vehicle

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA - vehicle service management system - GetServiceDetailsServlet.java - IntelliJ IDEA
RASA - vehicle service management system > src > com > rasa > servlet > GetServiceDetailsServlet > vehicle
Tomcat 9.0.37
AddCustomerServlet.java AddServiceEntryServlet.java CreateWorkProgressServlet.java paymentListServlet.java GetServiceDetailsServlet.java UpdateVehicleServlet.java
Project Z-Structure Commit Learn Favorites Web
18
19     @WebServlet("/GetServiceDetailsServlet")
20     public class GetServiceDetailsServlet extends HttpServlet {
21
22         private ServiceEntry serviceEntry;
23         private ClientService clientService;
24         private VehicleService vehicleService;
25         private Repair repair;
26         private Vehicle vehicle;
27         private Customer customer;
28
29         @Override
30         public void init() throws ServletException {
31             super.init();
32             serviceEntry = new ServiceEntry();
33             clientService = new ClientService();
34             vehicleService = new VehicleService();
35             repair = new Repair();
36             vehicle = new Vehicle();
37             customer = new Customer();
38
39         }
40
41         protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
42             response.setContentType("text/html");
43             String serviceID = request.getParameter("serviceID");
44             repair = serviceEntry.getRepairByServiceID(Integer.parseInt(serviceID));
45             customer = clientService.searchByNic(repair.getNICno().toUpperCase());
46             vehicle = vehicleService.searchByRegistrationNumber(repair.getVehicleRegistrationNo().toUpperCase());
47
48             request.setAttribute("repair", repair);
49
50             response.sendRedirect("index.jsp");
51         }
52     }

```

Figure 73: Get Service Details

Figure 74: Get budget amount

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA-vehicle-service-management-system(final-final) - budgetServlet.java - IntelliJ IDEA
RASA-vehicle-service-management-system(final-final) > src > com > rasa > servlet > budgetServlet > doPost
Project Files Z: Structure Commit Services Web Favorites
1 package com.rasa.servlet;
2 import ...
3
4 @WebServlet("/budgetServlet")
5 public class budgetServlet extends HttpServlet {
6
7     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
8         String year = request.getParameter("year");
9         String month = request.getParameter("month");
10        HttpSession session = request.getSession();
11        session.setAttribute("year", year);
12        session.setAttribute("month", month);
13        budgetService budgetservice = new budgetService();
14        Budget b = budgetservice.AllMethods(year,month);
15        System.out.println(b.getRepairAmount());
16        System.out.println(b.getRentalAmount());
17        System.out.println(b.getProfit());
18        System.out.println(b.getInventoryExpenses());
19        System.out.println(b.getEmpPayments());
20
21        request.setAttribute("income", b.getTotIncome());
22
23        request.setAttribute("Ramount", b.getRepairAmount());
24        request.setAttribute("rentalAmount", b.getRentalAmount());
25        request.setAttribute("expenses", b.getTotExpenses());
26
27        request.setAttribute("inventoryExpenses", b.getInventoryExpenses());
28        request.setAttribute("empPayments", b.getEmpPayments());
29        request.setAttribute("profit", b.getProfit());
30
31        request.getRequestDispatcher("budget.jsp").forward(request, response);
32    }
33}

```

Figure 75: Update payment

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA-vehicle-service-management-system(final-final) - paymentEditServlet.java - IntelliJ IDEA
RASA-vehicle-service-management-system(final-final) > src > com > rasa > servlet > paymentEditServlet > doPost
Project Files Z: Structure Commit Services Web Favorites
1 package com.rasa.servlet;
2 import ...
3
4 @WebServlet("/paymentEditServlet")
5 public class paymentEditServlet extends HttpServlet {
6     private paymentService sv;
7
8     public paymentEditServlet() {
9         super();
10        sv = new paymentService();
11    }
12
13    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
14        int payId = Integer.parseInt(request.getParameter("payId"));
15        String registrationNumber = request.getParameter("registrationNumber");
16
17        double estimateAmount = Double.parseDouble(request.getParameter("estimateAmount"));
18
19        double cash = Double.parseDouble(request.getParameter("cash"));
20        Date paymentDate = Date.valueOf(request.getParameter("paymentDate"));
21
22
23        paymentList existingPayment = new paymentList(payId, registrationNumber, estimateAmount, cash, paymentDate);
24        try {
25            sv.updatePayment(existingPayment);
26            response.sendRedirect("paymentlist.jsp");
27        } catch (SQLException throwables) {
28            throwables.printStackTrace();
29        }
30    }
31}

```

Figure 76: Add payment

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA-vehicle-service-management-system(final-final) - paymentListServlet.java - IntelliJ IDEA
RASA-vehicle-service-management-system(final-fin... > src > com > rasa > servlet > paymentListServlet.java > doPost
Project Files 1 Project Structure 2 Commit 3 Favorites 4 Services 5 Git 6 TODO 7 Services 8 Java Enterprise 9 Terminal 10 Database 11 Art 12 Event Log 13
14 package com.rasa.servlet;
15
16 import ...
17
18 @WebServlet("/paymentListServlet")
19 public class paymentListServlet extends HttpServlet {
20     private paymentService sv;
21
22     public paymentListServlet() {
23         super();
24         sv = new paymentService();
25     }
26
27     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
28
29         String registrationNumber = request.getParameter("registrationNumber");
30
31         double estimateAmount = Double.parseDouble(request.getParameter("estimateAmount"));
32
33         double cash = Double.parseDouble(request.getParameter("cash"));
34         Date paymentDate = Date.valueOf(request.getParameter("paymentDate"));
35         int sid = Integer.parseInt(request.getParameter("sid"));
36
37
38         paymentList newPayment = new paymentList(registrationNumber, estimateAmount, cash, paymentDate, sid);
39         try {
40             sv.addPayment(newPayment);
41             response.sendRedirect("paymentlist.jsp");
42         } catch (SQLException throwable) {
43             throwables.printStackTrace();
44         }
45     }
46
47 }

```

Figure 77: Display latest payment

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA-vehicle-service-management-system(final-final) - paymentRetrieveServlet.java - IntelliJ IDEA
RASA-vehicle-service-management-system(final-fin... > src > com > rasa > servlet > paymentRetrieveServlet.java > doPost
Project Files 1 Project Structure 2 Commit 3 Favorites 4 Services 5 Git 6 TODO 7 Services 8 Java Enterprise 9 Terminal 10 Database 11 Art 12 Event Log 13
14
15
16 @WebServlet(name = "paymentRetrieveServlet")
17 public class paymentRetrieveServlet extends HttpServlet {
18
19     private static final long serialVersionUID = 1L;
20
21     private paymentService payday;
22
23     public void init() { this.payday = new paymentService(); }
24
25     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
26
27         try {
28             paymentService load = new paymentService();
29             paymentList pay= load.showLatestPayment();
30
31             request.setAttribute("payId", pay.getPayId());
32             request.setAttribute("registrationNumber", pay.getRegistrationNumber());
33             request.setAttribute("estimateAmount", pay.getEstimateAmount());
34             request.setAttribute("cash", pay.getCash());
35             request.setAttribute("paymentDate", pay.getPaymentDate());
36
37             request.getRequestDispatcher("paymentList.jsp").forward(request, response);
38
39         } catch (ServletException e) {
40             e.printStackTrace();
41         }
42     }
43
44 }

```

Car records model class *Figure 78*

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA-vehicle-service-management-system - CarRecords.java - IntelliJ IDEA
RASA-vehicle-service-management-system > src > com > rasa > model > CarRecords > CarRecords.java
  CarRecords_Reg.css CarRecords_Edit.css CarRecords.java CarRecordsReg.jsp CarRecordsEdit.jsp CarRecords.css CRecords Ant Database
  Tomcat 8.5.58 Git
  Project Files idea lib out src com rasa model CarRecords service CarRecordService servlet CarRecordDeleteServlet CarRecordInsertServlet CarRecordRetreiveServlet CarRecordUpdateEditServlet CarRecordUpdateServlet ReportGenerateServlet util DBConnectionUtil EmpQuery web scripts carRecords.js customerManagement.js styles Semantic-UI-CSS-master CarRecords.css CarRecords_Edit.css CarRecords_Reg.css style.css
  package com.rasa.model;
  public class CarRecords {
    private int regNumber;
    private String fname;
    private String lname;
    private String id;
    private String address;
    private int phone;
    private String email;
    private String bookNumber;
    private String model;
    private int seatAmount;
    private float distance;
    private String carType;
    public CarRecords(int regNumber, String fname, String lname, String id, String address, int phone, String email, String bookNumber, String model, int seatAmount, float distance, String carType) {
      this.regNumber = regNumber;
      this.fname = fname;
      this.lname = lname;
      this.id = id;
      this.address = address;
      this.phone = phone;
      this.email = email;
      this.bookNumber = bookNumber;
      this.model = model;
      this.seatAmount = seatAmount;
      this.distance = distance;
      this.carType = carType;
    }
  }
  
```

Event Log
Build completed successfully in 4 s 351 ms (yesterday 2:51 PM)
1:35 PM 5/30/2021

Car records service *Figure 79*

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA-vehicle-service-management-system - CarRecordService.java - IntelliJ IDEA
RASA-vehicle-service-management-system > src > com > rasa > service > CarRecordService > UpdateCarRecords > CarRecordService.java
  CarRecords_Reg.css CarRecords_Edit.css CarRecords.java CarRecordService.java CarRecordsReg.jsp CarRecordsEdit.jsp C Ant Database
  Tomcat 8.5.58 Git
  Project Files idea lib out src com rasa model CarRecords service CarRecordService servlet CarRecordDeleteServlet CarRecordInsertServlet CarRecordRetreiveServlet CarRecordUpdateEditServlet CarRecordUpdateServlet ReportGenerateServlet util DBConnectionUtil EmpQuery web scripts carRecords.js customerManagement.js styles Semantic-UI-CSS-master CarRecords.css CarRecords_Edit.css CarRecords_Reg.css style.css
  package com.rasa.service;
  import com.itextpdf.text.Document;
  import com.itextpdf.text.DocumentException;
  import com.itextpdf.text.Paragraph;
  import com.itextpdf.text.pdf.PdfCell;
  import com.itextpdf.text.pdf.PdfTable;
  import com.itextpdf.text.pdf.PdfWriter;
  import com.rasa.model.CarRecords;
  import com.rasa.util.DBConnectionUtil;
  import java.io.FileOutputStream;
  import java.io.IOException;
  import java.sql.*;
  import java.util.ArrayList;
  import java.util.List;
  //Retrieve car details
  public class CarRecordService {
    private static Connection conn;
    private static PreparedStatement preparedStatement;
    public static List<CarRecords> CarRecordRetrieve() {
      ArrayList<CarRecords> CR = new ArrayList<>();
      try {
        conn = DBConnectionUtil.getConnection();
        
```

Event Log
Build completed successfully in 4 s 351 ms (yesterday 2:51 PM)
18 chars 121:33 CRLF UTF-8 4 spaces 1:35 PM 5/30/2021

Car Records delete servlet *Figure 80*

The screenshot shows the IntelliJ IDEA interface with the project 'RASA-vehicle-service-management-system' open. The 'CarRecordDeleteServlet.java' file is the active editor. The code implements a servlet for deleting car records.

```
package com.rasa.servlet;

import com.rasa.service.CarRecordService;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;

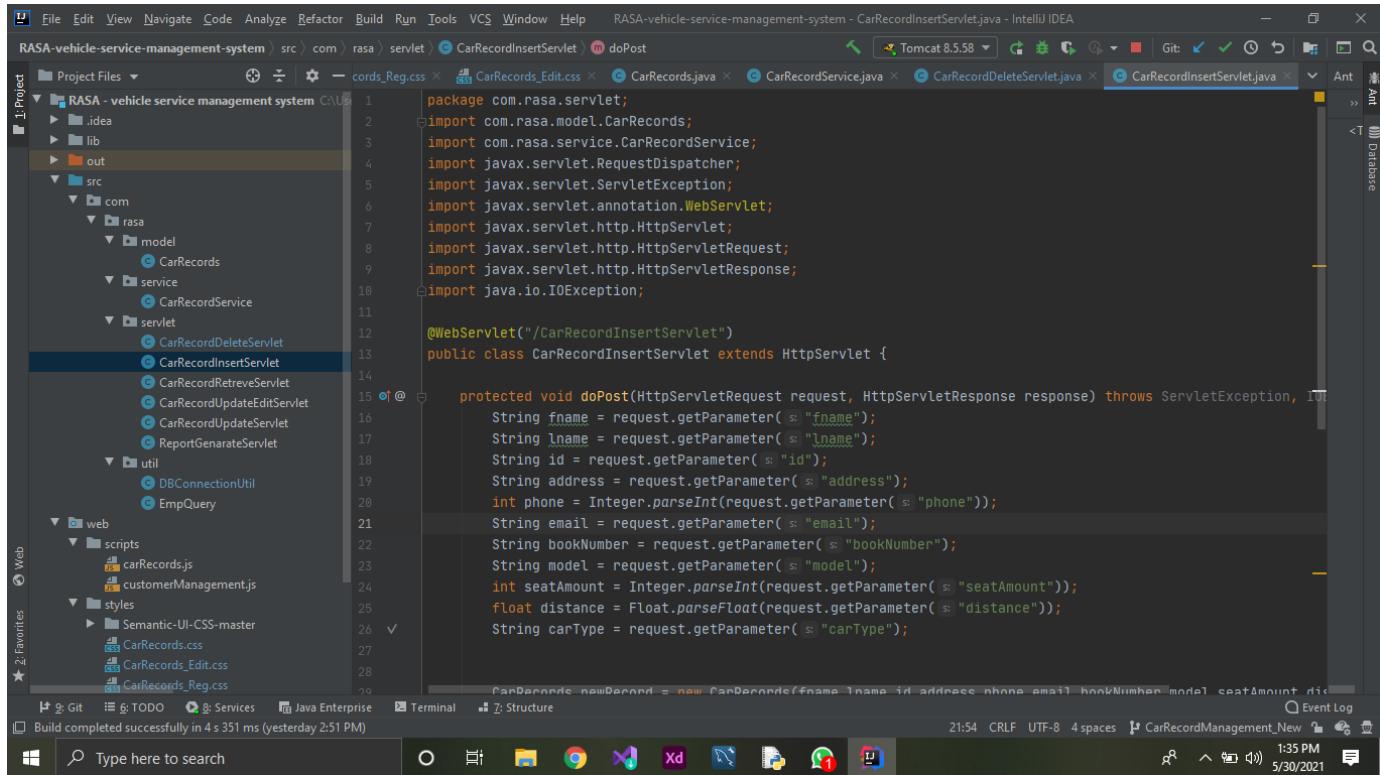
@WebServlet("/CarRecordDeleteServlet")
public class CarRecordDeleteServlet extends HttpServlet {

    private CarRecordService carRecordService;
    public CarRecordDeleteServlet() {
        super();
        carRecordService = new CarRecordService();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
        int regNumber = Integer.parseInt(request.getParameter("regNumber"));
        try {
            carRecordService.DeleteCarRecords(regNumber);
            response.sendRedirect("CRecords.jsp");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

The project structure on the left includes packages for com, rasa, service, and servlet, along with utility classes like DBConnectionUtil and EmpQuery. The web directory contains scripts (carRecords.js, customerManagement.js) and styles (Semantic-UI-CSS-master, CarRecords.css, CarRecords_Edit.css, CarRecords_Reg.css).

Car records Insert servlet *Figure 81*

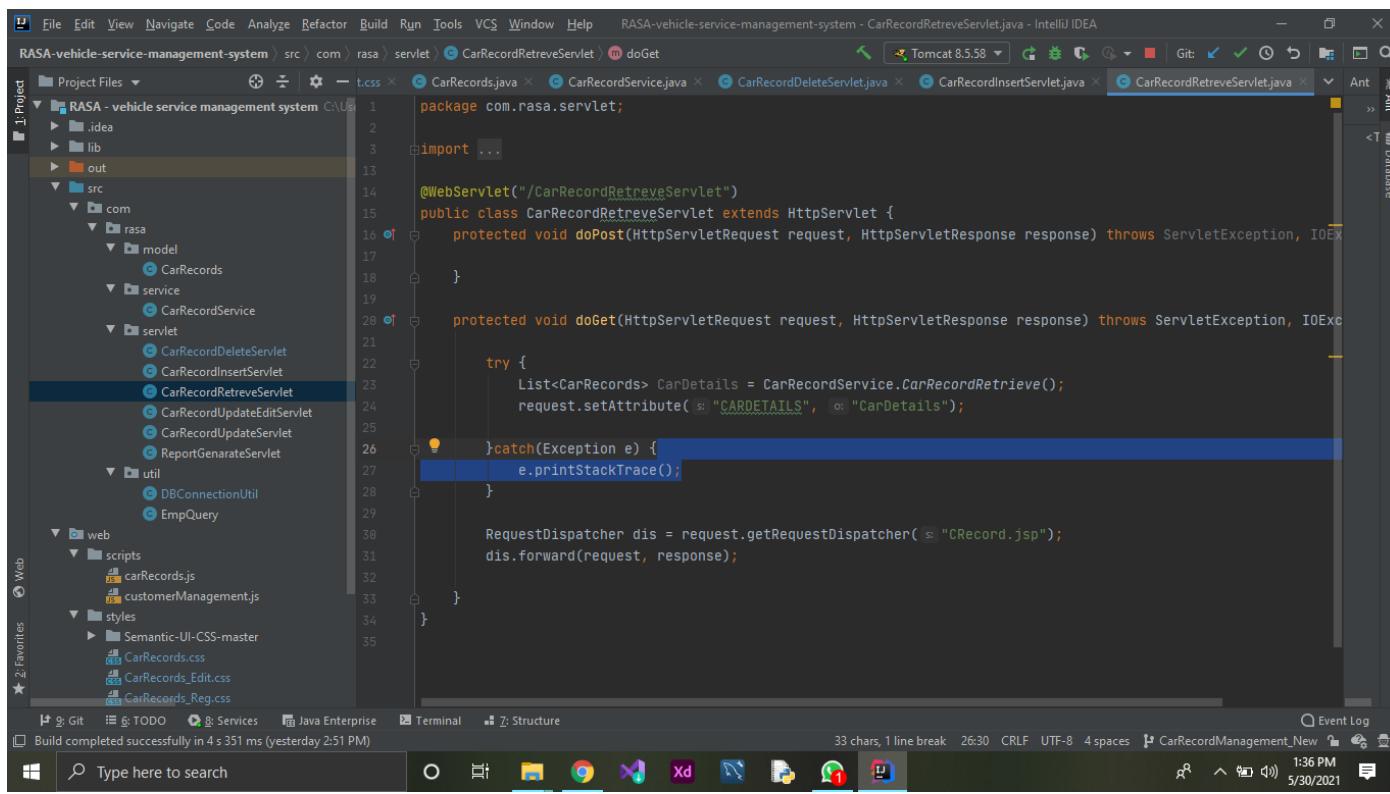


```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA-vehicle-service-management-system - CarRecordInsertServlet.java - IntelliJ IDEA
RASA-vehicle-service-management-system > src > com > rasa > servlet > CarRecordInsertServlet.java doPost Tomcat 8.5.58 Git: 
Project Files 1 package com.rasa.servlet;
2 import com.rasa.model.CarRecords;
3 import com.rasa.service.CarRecordService;
4 import javax.servlet.RequestDispatcher;
5 import javax.servlet.ServletException;
6 import javax.servlet.annotation.WebServlet;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import java.io.IOException;
11
12 @WebServlet("/CarRecordInsertServlet")
13 public class CarRecordInsertServlet extends HttpServlet {
14
15     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
16         String fname = request.getParameter("fname");
17         String lname = request.getParameter("lname");
18         String id = request.getParameter("id");
19         String address = request.getParameter("address");
20         int phone = Integer.parseInt(request.getParameter("phone"));
21         String email = request.getParameter("email");
22         String bookNumber = request.getParameter("bookNumber");
23         String model = request.getParameter("model");
24         int seatAmount = Integer.parseInt(request.getParameter("seatAmount"));
25         float distance = Float.parseFloat(request.getParameter("distance"));
26         String carType = request.getParameter("carType");
27
28
29         CarRecords newRecord = new CarRecords(fname, lname, id, address, phone, email, bookNumber, model, seatAmount, distance);
30         CarRecordService.insertRecord(newRecord);
31
32         RequestDispatcher dis = request.getRequestDispatcher("CRecord.jsp");
33         dis.forward(request, response);
34     }
35
36 }

```

Car records retrieve servlet *Figure 82*



```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA-vehicle-service-management-system - CarRecordRetreveServlet.java - IntelliJ IDEA
RASA-vehicle-service-management-system > src > com > rasa > servlet > CarRecordRetreveServlet.java doGet Tomcat 8.5.58 Git: 
Project Files 1 package com.rasa.servlet;
2 import ...
3
4 @WebServlet("/CarRecordRetreveServlet")
5 public class CarRecordRetreveServlet extends HttpServlet {
6
7     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
8
9     }
10
11     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
12
13         try {
14             List<CarRecords> CarDetails = CarRecordService.CarRecordRetrieve();
15             request.setAttribute("CARDETAILS", CarDetails);
16
17         } catch (Exception e) {
18             e.printStackTrace();
19
20         }
21
22         RequestDispatcher dis = request.getRequestDispatcher("CRecord.jsp");
23         dis.forward(request, response);
24
25     }
26
27 }

```

Car record update edit servlet *Figure 83*

The screenshot shows the IntelliJ IDEA interface with the project 'RASA - vehicle service management system' open. The code editor displays the `CarRecordUpdateEditServlet.java` file, which extends `HttpServlet`. It imports various classes and uses annotations like `@WebServlet` and `@RequestDispatcher`. The code handles a GET request by setting attributes from a car record object `p` to the response. The project structure on the left shows packages for com.rasa.model, com.rasa.service, and com.rasa.servlet, along with web resources like scripts and styles.

```
package com.rasa.servlet;
import ...
@WebServlet("/CarRecordUpdateEditServlet")
public class CarRecordUpdateEditServlet extends HttpServlet {
    private CarRecordService CarRec1;
    public void init() { this.CarRec1 = new CarRecordService(); }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        int regNumber = Integer.parseInt(request.getParameter("regNumber"));
        CarRecords p = CarRec1.selectRecord(regNumber);
        request.setAttribute("regNumber", p.getRegNumber());
        request.setAttribute("fname", p.getFname());
        request.setAttribute("lname", p.getLname());
        request.setAttribute("id", p.getId());
        request.setAttribute("address", p.getAddress());
        request.setAttribute("phone", p.getPhone());
        request.setAttribute("email", p.getEmail());
        request.setAttribute("bookNumber", p.getBookNumber());
        request.setAttribute("model", p.getModel());
        request.setAttribute("seatAmount", p.getSeatAmount());
        request.setAttribute("distance", p.getDistance());
        request.setAttribute("carType", p.getCarType());
        request.getRequestDispatcher("CarRecordEdit.jsp").forward(request, response);
    }
}
```

Car record update servlet *Figure 84*

RASA-vehicle-service-management-system - CarRecordUpdateServlet.java - IntelliJ IDEA

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA-vehicle-service-management-system - CarRecordUpdateServlet.java - IntelliJ IDEA
RASA-vehicle-service-management-system > src > com > rasa > servlet > CarRecordUpdateServlet > doPost
Project Files
RASA - vehicle service management system
  .idea
  lib
  out
  src
    com
      rasa
        model
          CarRecords
        service
          CarRecordService
        servlet
          CarRecordDeleteServlet
          CarRecordInsertServlet
          CarRecordRetreiveServlet
          CarRecordUpdateEditServlet
          CarRecordUpdateServlet
          ReportGenerateServlet
        util
          DBConnectionUtil
          EmpQuery
    web
      scripts
        carRecords.js
        customerManagement.js
      styles
        Semantic-UI-CSS-master
        CarRecords.css
        CarRecords_Edit.css
        CarRecords_Reg.css
src/com/rasa/servlet/CarRecordUpdateServlet.java
1 package com.rasa.servlet;
2
3 import ...
4
5 @WebServlet("/CarRecordUpdateServlet")
6 public class CarRecordUpdateServlet extends HttpServlet {
7
8     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
9
10         int regNumber = Integer.parseInt(request.getParameter("regNumber"));
11         String fname = request.getParameter("fname");
12         String lname = request.getParameter("lname");
13         String id = request.getParameter("id");
14         String address = request.getParameter("address");
15         int phone = Integer.parseInt(request.getParameter("phone"));
16         String email = request.getParameter("email");
17         String bookNumber = request.getParameter("bookNumber");
18         String model = request.getParameter("model");
19         int seatAmount = Integer.parseInt(request.getParameter("seatAmount"));
20         Float distance = Float.parseFloat(request.getParameter("distance"));
21         String carType = request.getParameter("carType");
22
23         CarRecords newCar = new CarRecords(regNumber, fname, lname, id, address, phone, email, bookNumber, model);
24         try {
25             CarRecordService.UpdateCarRecords(newCar);
26             response.sendRedirect("Crecords.jsp");
27         } catch (SQLException throwables) {
28             throwables.printStackTrace();
29         }
30     }
31
32     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
33
34     }
35
36 }
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
99
```

File: CarRecordUpdateServlet.java

Content:

```
package com.rasa.servlet;

import ...

@WebServlet("/CarRecordUpdateServlet")
public class CarRecordUpdateServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        int regNumber = Integer.parseInt(request.getParameter("regNumber"));
        String fname = request.getParameter("fname");
        String lname = request.getParameter("lname");
        String id = request.getParameter("id");
        String address = request.getParameter("address");
        int phone = Integer.parseInt(request.getParameter("phone"));
        String email = request.getParameter("email");
        String bookNumber = request.getParameter("bookNumber");
        String model = request.getParameter("model");
        int seatAmount = Integer.parseInt(request.getParameter("seatAmount"));
        Float distance = Float.parseFloat(request.getParameter("distance"));
        String carType = request.getParameter("carType");

        CarRecords newCar = new CarRecords(regNumber, fname, lname, id, address, phone, email, bookNumber, model);
        try {
            CarRecordService.UpdateCarRecords(newCar);
            response.sendRedirect("Crecords.jsp");
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }
}
```

Car record delete servlet Figure 85

RASA-vehicle-service-management-system - ReportGenerateServlet.java - IntelliJ IDEA

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help RASA-vehicle-service-management-system - ReportGenerateServlet.java - IntelliJ IDEA
RASA-vehicle-service-management-system > src > com > rasa > servlet > ReportGenerateServlet
Project Files
RASA - vehicle service management system
  .idea
  lib
  out
  src
    com
      rasa
        model
          CarRecords
        service
          CarRecordService
        servlet
          CarRecordDeleteServlet
          CarRecordInsertServlet
          CarRecordRetreiveServlet
          CarRecordUpdateEditServlet
          CarRecordUpdateServlet
          ReportGenerateServlet
        util
          DBConnectionUtil
          EmpQuery
    web
      scripts
        carRecords.js
        customerManagement.js
      styles
        Semantic-UI-CSS-master
        CarRecords.css
        CarRecords_Edit.css
        CarRecords_Reg.css
src/com/rasa/servlet/ReportGenerateServlet.java
1 package com.rasa.servlet;
2
3 import com.itextpdf.text.DocumentException;
4 import com.rasa.service.CarRecordService;
5 import javax.servlet.ServletException;
6 import javax.servlet.annotation.WebServlet;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import java.io.IOException;
11
12
13 @WebServlet("/ReportGenerateServlet")
14 public class ReportGenerateServlet extends HttpServlet {
15
16     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
17
18         try {
19             CarRecordService.ReportGenerate();
20         } catch (DocumentException e) {
21             // TODO Auto-generated catch block
22             e.printStackTrace();
23         }
24
25     }
26
27     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
28
29     }
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
```

File: ReportGenerateServlet.java

Content:

```
package com.rasa.servlet;

import com.itextpdf.text.DocumentException;
import com.rasa.service.CarRecordService;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/ReportGenerateServlet")
public class ReportGenerateServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        try {
            CarRecordService.ReportGenerate();
        } catch (DocumentException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }
}
```

Testing

Test ID	Test Inputs	Expected Outputs	Actual Outputs	Result (Pass/ Fail)	Comments
1	Input present as the attendance status	Save the attendance and employee disappear from the attendance table	Expected Output	Pass	Add attendance works 100% accurately.
2	Real amount as Advance	Save the Advance and redirect to the employee payment page	Expected Output	Pass	Add Advance function works 100% accurately
3	Real values as bonus and deduct	Save the salary in database and redirect to the Employee Payment page	Expected output	Pass	Adding new Salary function worked 100% accurately

4	Assign employee name, Gender, Date of Birth, Basic Salary, NIC, Email, mobile, Address Employee ID is automatically generated by the system	A new employee should be added and make an employee profile.	A new employee is added, and in the database all the employee's information is kept. In the view employee tab, current employee information and the newly added can be viewed	Pass	Adding of a new employee was executed successfully.
5	Update employee name, Gender, Date of Birth, NIC, Basic Salary, Email, contact number, Address	Relevant employee details are updated	Details of workers are updated and maintained in the database. In the view employee tab, current employee information and the newly updated employee details can be viewed.	Pass	Updating employee details functions was executed successfully.
6-Work Progress Management	Service name = remove and refitting status = on progress	Data display successfully	Expected output	Pass	Service details added successfully and validated successfully
7-Work Progress Management	Component name = bonnet Estimate amount = 1000.00 Component name = front buffer	Total estimate amount = 3000.00 Total component = 2	Expected output	Pass	Calculation works 100% accurately

	Estimate amount = 2000.00				
8-Car Record Management	Assign rental details first name, last name, email, mobile, address, pick up date, drop off date, rental price.	A new rental details record should be added	Expected output	Pass	Rental details added 100% accurately
9- Car Rental Management	Update first name, last name, email, mobile, address, pick up date, drop off date, rental price.	Relevant rental details are updated	Expected output	Pass	Rental details updated successfully
10 – Financial Management	Year = 2021 Month = "May"	Profit = Rs:5000	Profit = Rs:5000	Pass	Profit is correctly calculated according to given year and month

11- Financial Management	Registration number= CBD123 Estimate amount = 20000 Cash = 20000 Payment date = 2021/05/20	Data display successfully	Expected output	pass	Payment details added and display correctly and validated succesfully

2 Conclusion

The web-based system of Vehicle Repair Management System was implemented as a replacement to the previously practiced manual process. The new system was implemented by reducing the complexity of the manual process while breaking into eight functions. The functions Customer Details management, Work progress management, Financial management, Employee Details Management,

Employee Payment Management, Car Records Management, Car Rental Management and Inventory Management. Through the system, reports can be downloaded as an electronic document which can be used to analyzed data.

In this system main objective is to handle documents of the system because this company handle lots of documents per day. From this system documents are handle more effiency.In Customer Details Management Function manage all details of the customer in accuracy. In Work progress management Function deals with services that use to repair a vehicle and estimated amount as well as with employees who are assign to for each service. In Financial Management Function manage vehicle repair payments and show monthly profit to the user with using diagrams and amounts.

From Employee Details Management will handle all the details of the employees of the company and from using Employee Payment Management function help to user to take employee attendance and store them and calculate employee's payments.

Furthermore, from Car Records Management function user can keep details of the vehicle and vehicle owner's data who given their vehicle to rent. And also Car Rental Management Function helps to handle all rental processes and check availability of the vehicles. Moreover, Inventory Management Function use to manage inventory section of the company storing supplier's data, prices of the items and quantity.

