

STQA MINI PROJECT 1

PARKING MANAGEMENT SYSTEM

Group Members

Bhavansh Gupta	BECOA134
Jitendra Joshi	BECOA139
Ganesh Shinde	BECOA164
Vaishali Avhale	BECOA104

Guide

Prof. Manjiri Ranjanikar

Introduction

- ❖ Parking management system for managing the records of the incoming and outgoing vehicles in an parking. It's an easy for Admin to retrieve the data if the vehicle has been visited through number he can get that data .
- ❖ Now days in many public places such as malls, multiplex system, hospitals, offices, market areas there is a crucial problem of vehicle parking. The vehicle parking area has many lanes/slots for car parking. So to park a vehicle one has to look for all the lanes. Moreover this involves a lot of manual labor and investment. Instead of vehicle caught in towing the vehicle can park on safe and security with low cost.
- ❖ The objective of this project is to build a Vehicle Parking management system that enables the time management and no parking disputes. The system that will track the entry and exit of cars, maintain a listing of cars within the parking lot, and determine if the parking lot is full or not. It will determine the cost of per vehicle according to their time consumption.

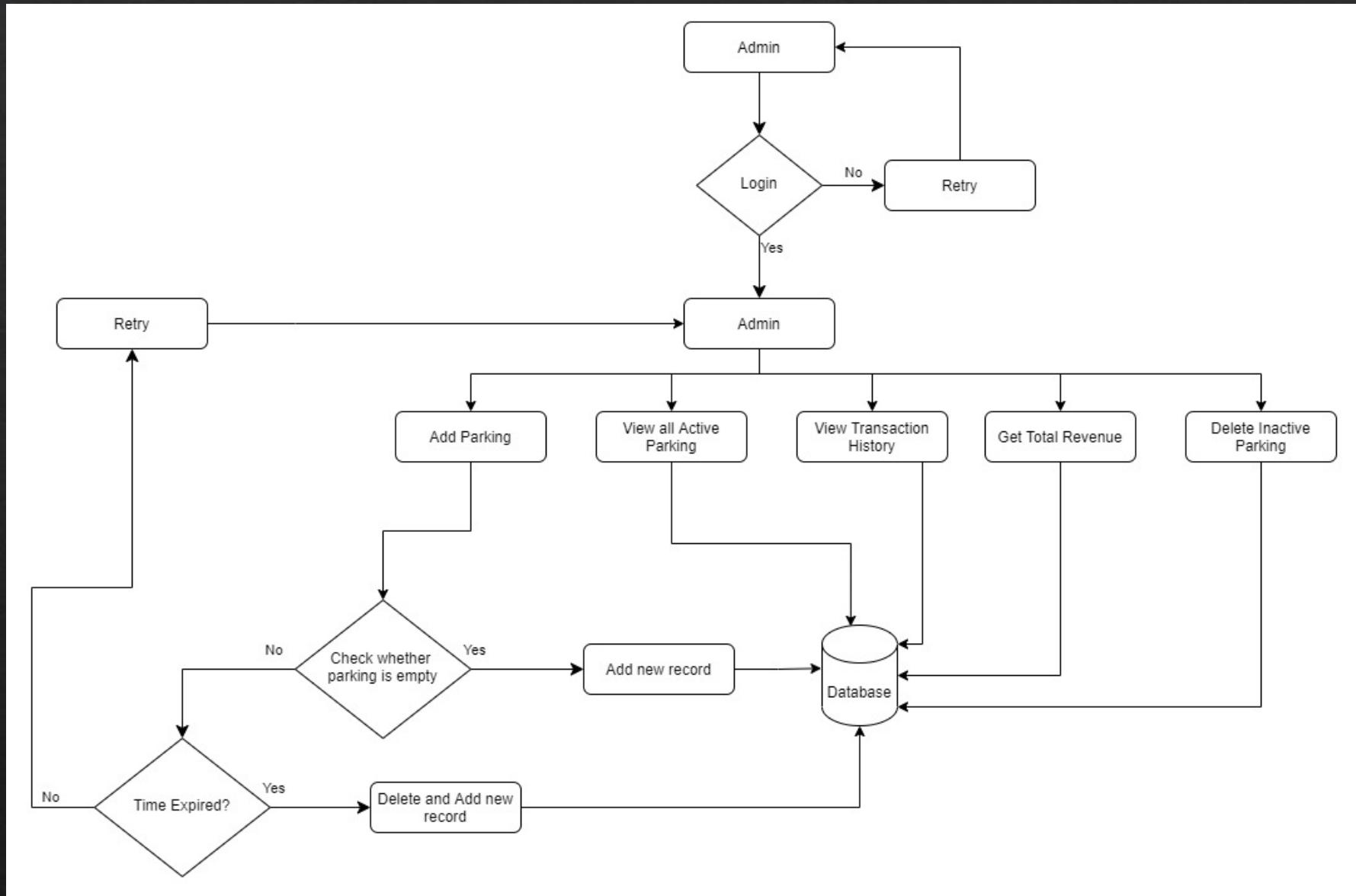
Objectives

- ❖ We can park our vehicle in our own slot by paying.
- ❖ Because of that there are no towing problems.
- ❖ And our vehicle has been parked as a secure condition.
- ❖ There is no risk for vehicle owner for parking the car.
- ❖ In case of any damages and problem of vehicle that will claim by parking management.
- ❖ Maintain records in short time of period.
- ❖ Determines the parking area is full or not.
- ❖ Enhances the visitor's experience.
- ❖ Generate Total Revenue Report

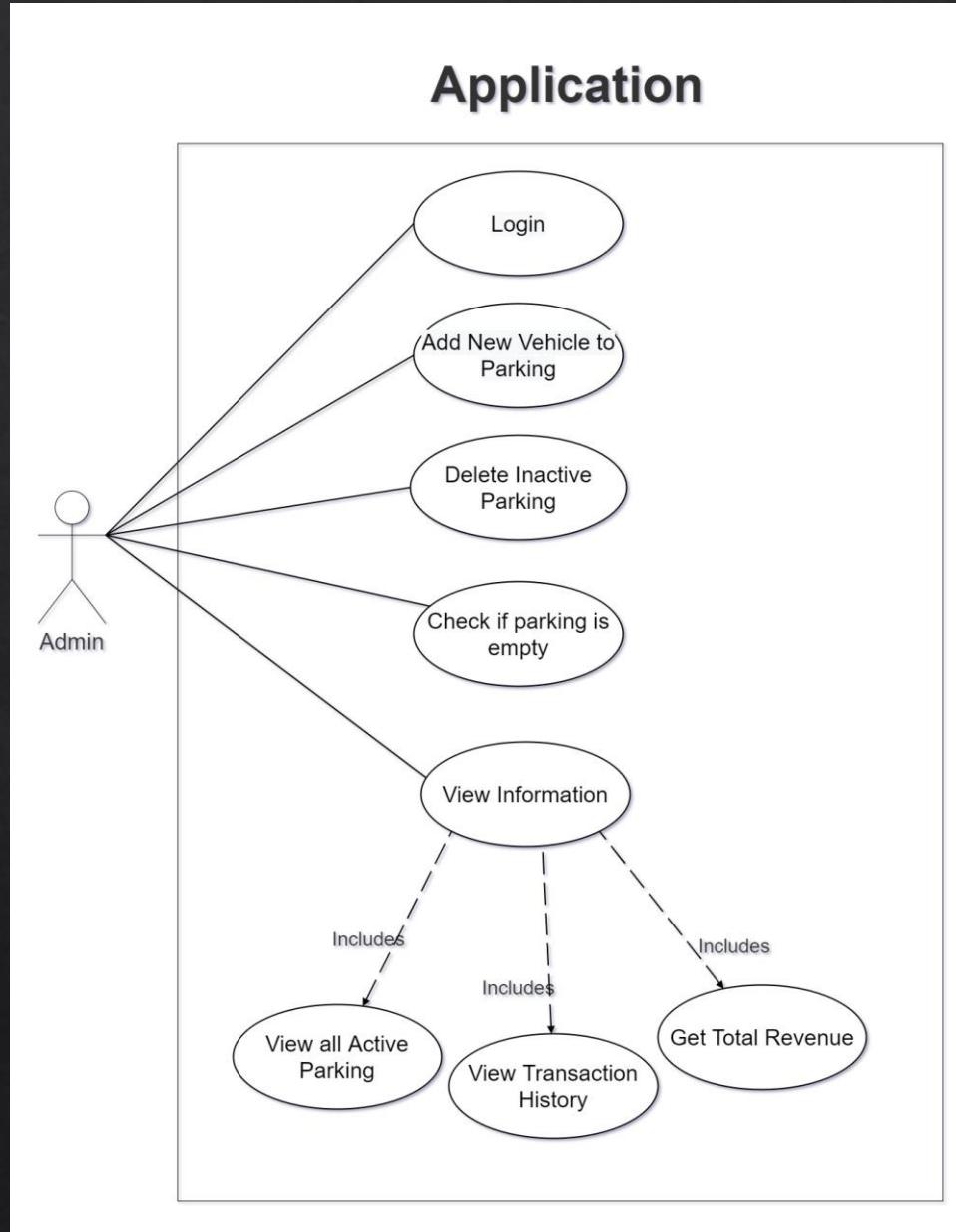
Motivation

- ❖ Parking control system has been generated in such a way that it is filled with many secure devices such as time and attendance machine, car counting system etc. These features are hereby very necessary nowadays to secure your car and also to evaluate the fee structure for every vehicle's entry and exit.

Block Diagram



Use Case Diagram



Functional Requirements

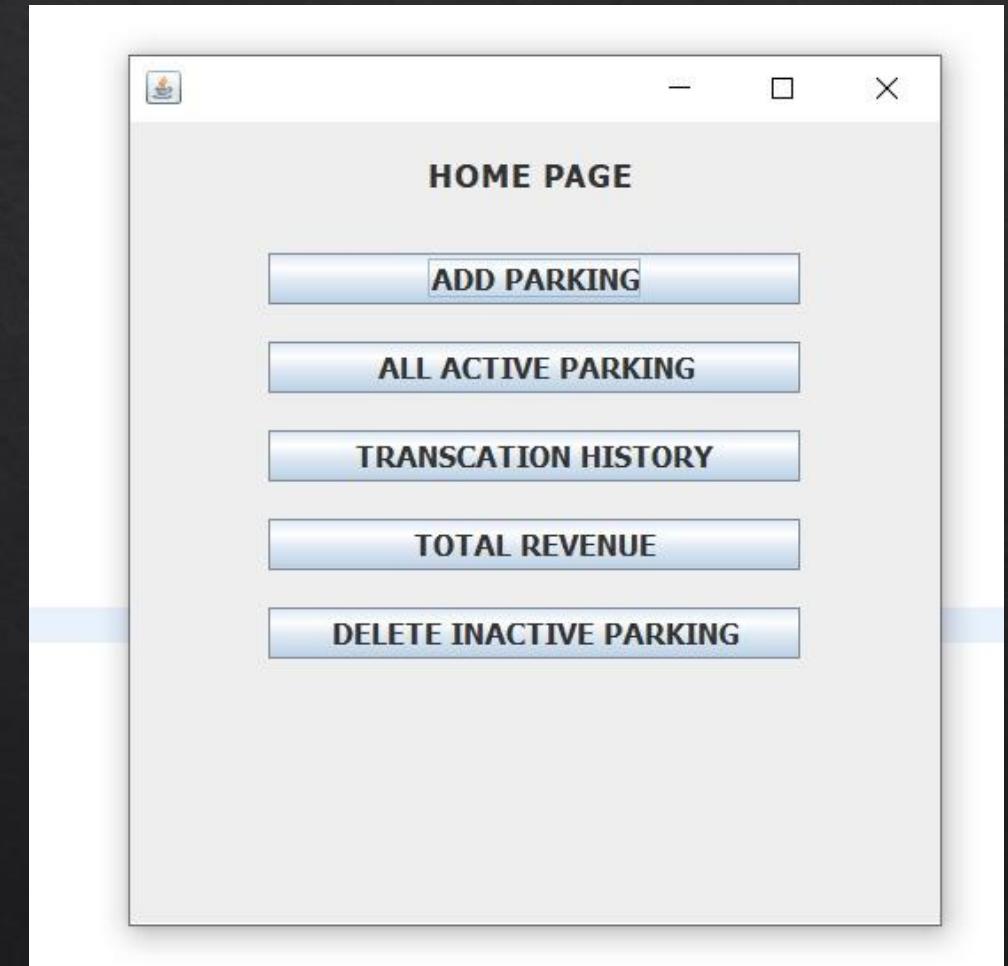
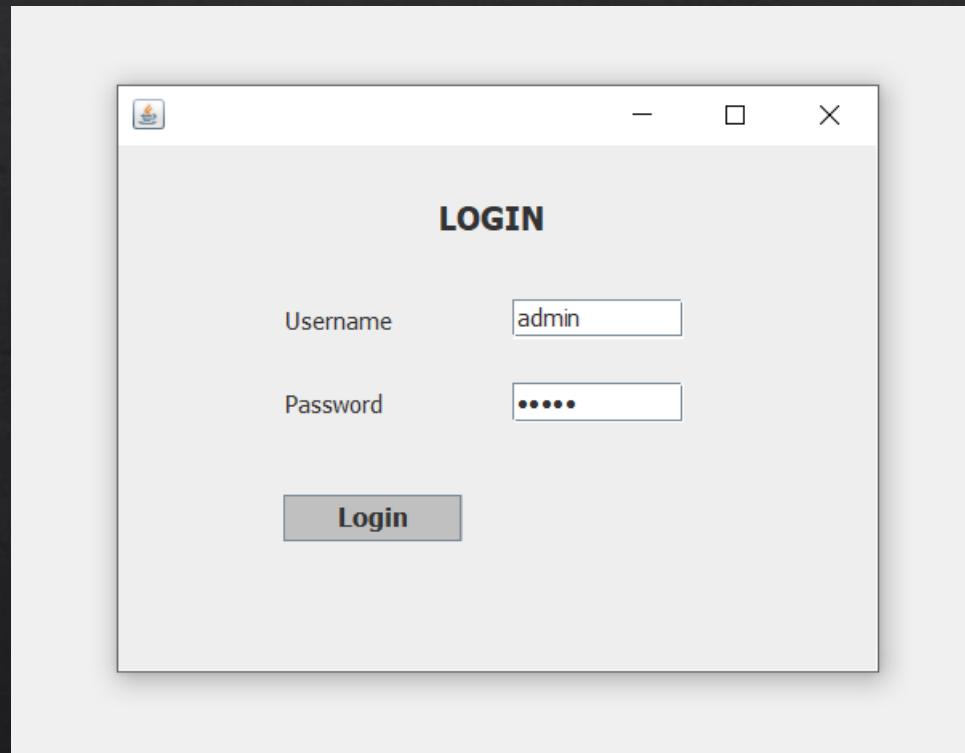
This is how the system shall look like and do when successfully completed. The system shall meet the following functional requirements:

- ♦ Admin need to enter all details for registration.
- ♦ Admin need to insert all details about customer and vehicle.
- ♦ Admin need to save all the details of customer and vehicle.
- ♦ Admin can retrieve the details of customer.
- ♦ Admin must generate a report of revenue collected.
- ♦ Admin can check for all parking spots
- ♦ Admin can delete remove vehicles after time expires from the record in a single click
- ♦ Admin can check if a specific parking spot is free or not.

Non Functional Requirements

- Performance should be fast.
- Software should be easy to use.
- Databases should be consistent and atomic.
- Errors should be handled properly.
- Software should pass all the tests during testing.
- These Software is capable to secure the data and easily retrieve the data
- This software provide user and authentication so that only the legitimate user are allowed to use the software

Implementation Screenshots



Implementation Screenshots

ADD VEHICLE

Parking ID	<input type="text" value="1"/>	SEARCH
Name	<input type="text" value="Bhavansh"/>	Vehicle No <input type="text" value="MH14YY1234"/>
Hours Parked	<input type="text" value="1"/>	Entry Time <input type="text" value="16:06:37"/>
Payment Mode	<input type="text" value="NetBanking"/>	Entry Date <input type="text" value="2021-10-10"/>
Phone No	<input type="text" value="8888888888"/>	

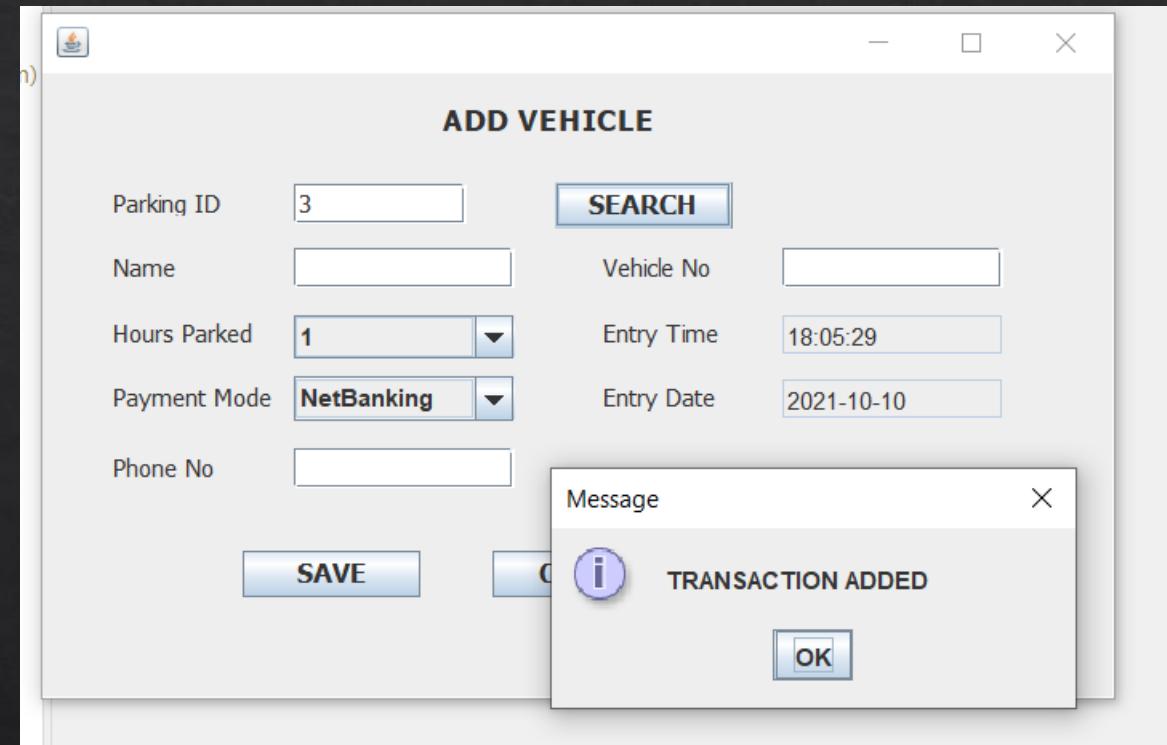
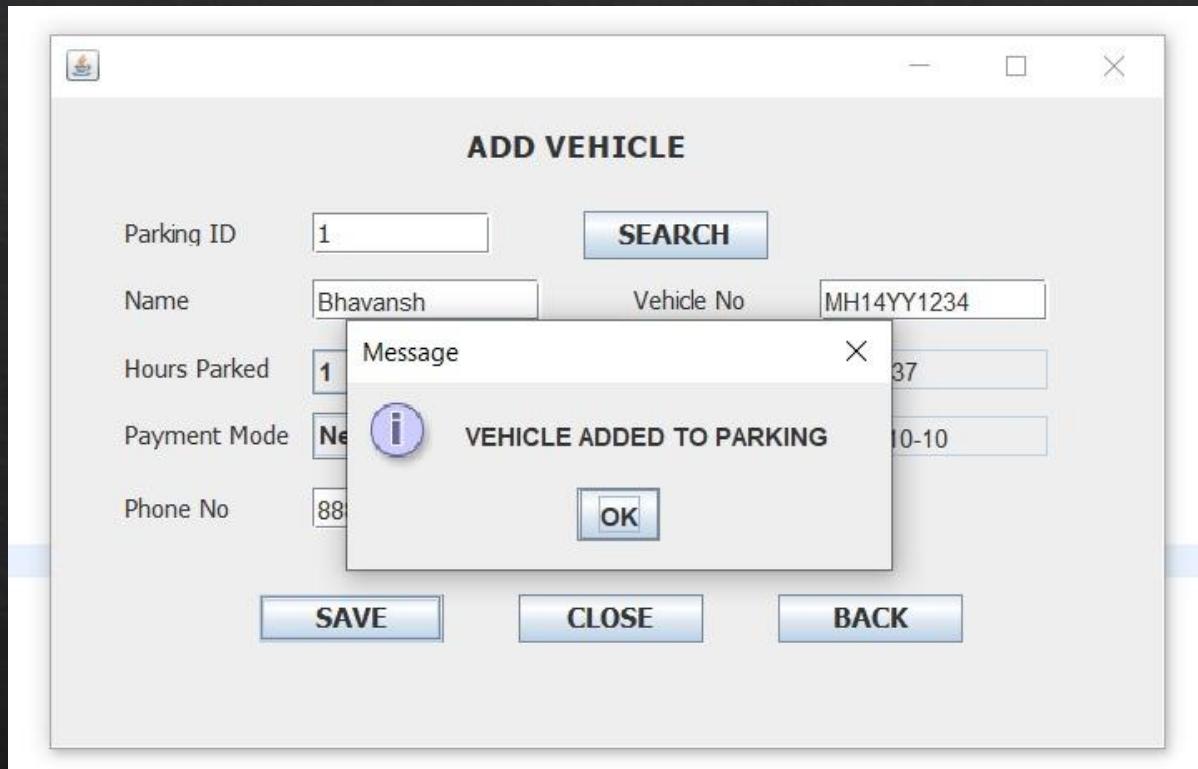
SAVE **CLOSE** **BACK**

ADD VEHICLE

Parking ID	<input type="text" value="1"/>	DELETE
Name	<input type="text" value="Bhavansh"/>	Vehicle No <input type="text" value="MH14YY1234"/>
Hours Parked	<input type="text" value="1"/>	Entry Time <input type="text" value="16:06:37"/>
Payment Mode	<input type="text" value="NetBanking"/>	Entry Date <input type="text" value="2021-10-10"/>
Phone No	<input type="text" value="8888888888"/>	

SAVE **CLOSE** **BACK**

Implementation Screenshots



Implementation Screenshots

A screenshot of a Java Swing application window titled "ACTIVE PARKING SPOTS". The window contains a table with the following data:

parking_id	person_na..	vehicle_nu..	time_of_en..	date_of_en..	payment_...	hours_par...	phone_num
1	Bhavansh	MH14YY1...	16:06:37	2021-10-10	NetBanking	1	88888888...
2	Ganesh	MH12BG9...	16:09:42	2021-10-10	Cash	2	98765432...
3	Jitendra	Joshi	16:10:36	2021-10-10	UPI	4	77665544...

At the bottom of the window are two buttons: "BACK" and "CLOSE".

A screenshot of a Java Swing application window titled "ALL TRANSACTIONS". The window contains a table with the following data:

parking_id	person_na..	vehicle_nu..	time_of_en..	date_of_en..	payment_...	hours_par...	phone_num
1	Bhavansh	MH14YY1...	16:06:37	2021-10-10	NetBanking	1	88888888...

At the bottom of the window are two buttons: "BACK" and "CLOSE".

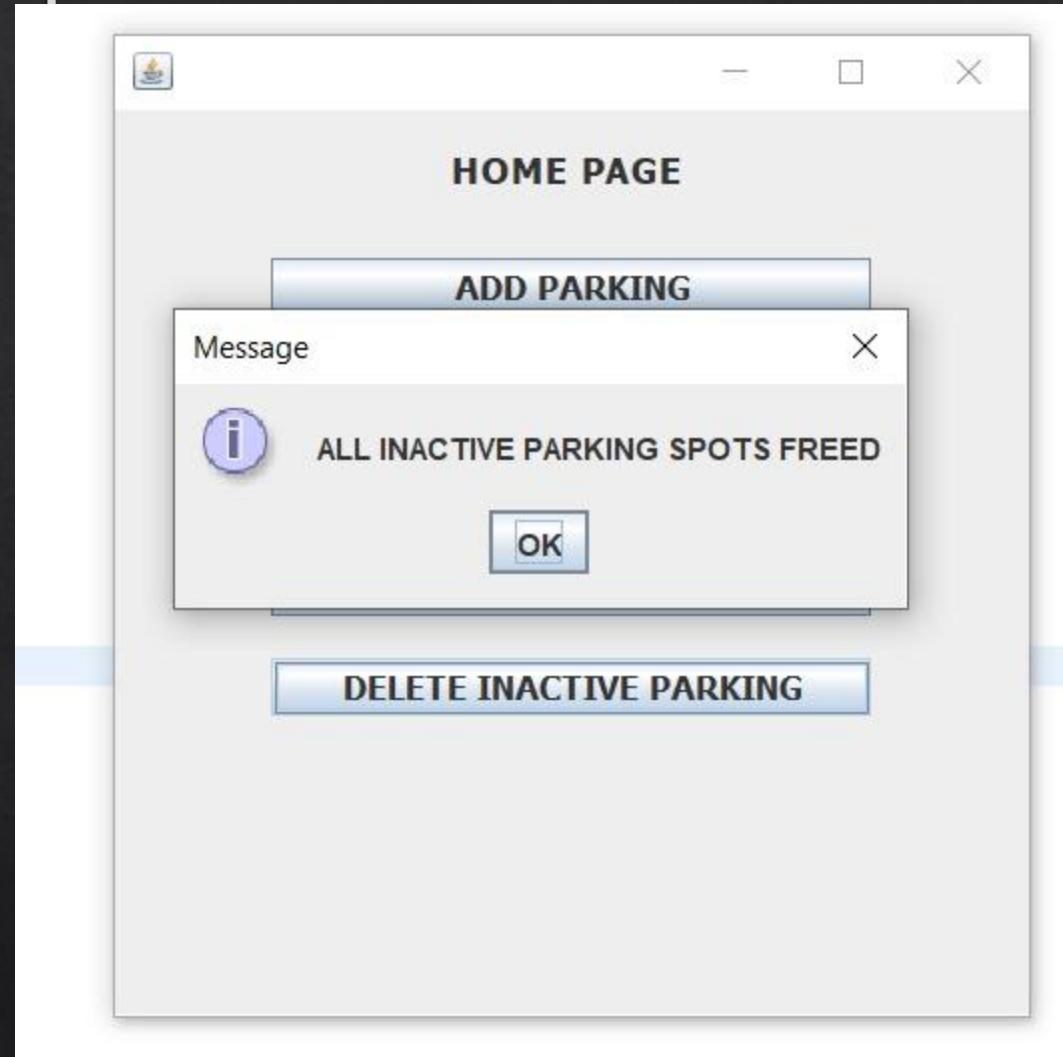
Implementation Screenshots

The image displays two side-by-side screenshots of a Windows application window titled "TOTAL AMOUNT".

Screenshot 1 (Left): Shows the initial state of the application. It has a title bar "TOTAL AMOUNT", a label "Enter Fees Per Hour", and a text input field containing "20". At the bottom are "BACK" and "CLOSE" buttons.

Screenshot 2 (Right): Shows the application after processing. The "Enter Fees Per Hour" label and "20" in the input field remain. A message dialog box titled "Message" is displayed, containing the text "Total Hours Parked: 7" and "Total Amount Recieved:140", preceded by an information icon. An "OK" button is at the bottom of the dialog. At the bottom of the main window are "BACK" and "CLOSE" buttons.

Implementation Screenshots



Test Plan Details

Introduction:

- ❖ This system is mainly aimed at reducing manual errors and solving simple time management problems. For each day a new schedule can be generated if intended by user. This system is developed using JavaFX and is tested using Junit testing for several test conditions.

❖ Assumptions of Testing:

- ❖ Application will be working on Windows 7/8/10 or Ubuntu. Other operating systems, if any, are compatible.
- ❖ Application is tested on Intel(R) Core (TM) i5 9th Gen, 2.4 GHz processor or higher, 8GB RAM. Other configurations used, if any, are compatible

❖ Test Scenarios

❖ Login:

1. To check whether the inserted username and password fields are not blank?
2. To check if both the fields are filled or not?
3. to check if any username and password is correct or not?
4. To check if the system notifies or not on wrong input details?
5. To check if any field remains empty or not corrected?

❖ Add Vehicle:

1. To check if the data is in correct format or not?
2. To check if any of the fields are empty or not?
3. To check if the data is found or not?
4. To check if the data already exists?
5. To check if a validation message pops up on the completion of addition or not?

❖ Test Scenarios

❖ Search and Delete Vehicle:

1. To check if the parking spot is occupied ?
2. To check if hours allotted are finished or not ?
3. To check if the data is deleted or not?
4. To check if a validation message pops up on the completion of deletion or not?

❖ Active Parking Spots :

1. To check if the parking spot is occupied ?
2. To check if all occupied parking spots are listed or not ?

❖ Test Scenarios

❖ Transaction History:

1. To check if the Transactions are added or not?
2. To check if post time expiry are the parking spots freed and data is added to transactions or not?

❖ Total Revenue:

1. To check if all the total hour count of active parking spots and transactions included or not?
2. To check if the input for fees per hour is working ?
3. To check if a message pops up with the total revenue calculated?

❖ Delete all inactive parking spots:

1. To check if all parking spots whose time is expired are freed or not?

Conclusion

- ❖ We have successfully built and Tested the Parking Management System Software and considering the Test Scenarios have Tested the software Manually and Using Junit Testcases.

THANK YOU !!!