INTRODUCTION

The Showroom Car Buying and Selling Management System is a mini project designed to facilitate the efficient management of car transactions within a showroom environment. This system aims to streamline the process of buying and selling cars by providing a user-friendly interface for both showroom staff and customers. By integrating various functionalities such as inventory management, customer information storage, and transaction tracking, this system aims to enhance the overall experience for both buyers and sellers in the showroom. The Car Showroom Management System consists of four major fields of management i.e. store cars, store Salespersons, store admins, store customers and store total sales progress. Here the store admin manages all the activities such as viewing and adding car details, salesperson details, and admin details of different stores. Each store has a Manager and the Manager’s task here is to fill in the details of the car, customer, sales, and workers of a particular store. The main goal of this application is to maintain the records of different stores which are visible only to the admin and can only be updated by him. It helps to achieve ease of access in searching car details of the selected store, salesperson’s details of the selected store, and sales details of the selected store. CAR Showroom Management System is a software application to maintain day-to-day activities in the store, This JAVA & SQL project helps to maintain the record of the bike, customers, workers, and sales information Admin manages adds, updates, and deletes the cars, salespersons, and admin records. Admin also can add new admins. The software is designed to handle, the daily activities of all the stores. Search the details when required.

OBJECTIVE

- The main objective is to manage the details of Vehicle Showroom, Customer, Registration, Booking, and Payment and only the administrator is guaranteed the access.
- The purpose of the project is to build an application program to reduce the manual work for managing the Vehicle Showroom, Customer, Vehicle, and Registration.
- Saves time of the user are he\she can view\book car from anywhere at any time.

SYSTEM ANALYSIS

The Car Showroom Management System is designed to streamline the operations of a car showroom, providing both administrators and users with a user-friendly interface to manage various aspects of the showroom. The system undergoes thorough system analysis to ensure its efficiency and effectiveness in meeting the requirements of the showroom. During system analysis, the key stakeholders and their requirements are identified. This includes the showroom administrators who require functionalities for managing car inventory, handling payments, and overseeing staff, as well as users who need access to view car details. The system's functionalities are carefully defined and prioritized based on their importance to the stakeholders. For administrators, functionalities such as adding, deleting, and updating car details are critical for managing inventory, while features like managing payments and viewing staff details contribute to overall showroom operations. User requirements focus on accessing detailed information about available cars to make informed decisions. In addition to functionality, system analysis also considers user interface design to ensure ease of use and intuitive navigation. The login interface serves as the gateway to the system, providing validation to ensure only authorized users gain access. Once logged in, users are presented with a dashboard tailored to their role, displaying relevant functionalities in a clear and organized manner. Furthermore, system analysis addresses data management and security requirements. Measures are implemented to safeguard sensitive information, such as user credentials and payment details, from unauthorized access. Data integrity is maintained through proper validation and error handling mechanisms, minimizing the risk of data corruption or loss. Overall, the system analysis phase lays the foundation for the development of a robust and user-friendly Car Showroom Management System. By understanding the needs of stakeholders, defining key functionalities, and ensuring data security and usability, the system is poised to enhance the efficiency and effectiveness of the car showroom operations.

3.1: Identification of need:-

In the realm of system analysis, identifying needs is fundamental to ensuring that a software solution effectively addresses the requirements and objectives of its intended users. For the Car Showroom Management System, the identification of needs encompasses understanding the various stakeholders' requirements, including administrators, staff, and customers.

Firstly, administrators seek streamlined processes for managing inventory, sales, and staff interactions. They require comprehensive tools for tracking sales, managing payments, updating car details, and overseeing staff activities.

Staff members operating within the showroom environment need efficient mechanisms for accessing and updating information regarding car inventory, sales records, and customer interactions. They require user-friendly interfaces that facilitate their daily tasks and enable effective communication with customers.

On the customer side, there is a need for a seamless and intuitive platform for browsing available cars, booking appointments, and providing feedback. Customers expect a smooth experience from their initial interaction with the system to the final purchase process, including transparent pricing and convenient payment options.

By identifying these diverse needs across stakeholders, system analysts can delineate clear objectives for the Car Showroom Management System, guiding the subsequent phases of system design, development, and implementation. This identification forms the cornerstone of a successful software solution that aligns with the operational requirements and user expectations of the car showroom.

3.2: Preliminary investigation:-

The preliminary investigation for the proposed Car Showroom Management System involves gathering essential information to understand the current state of the showroom's operations, identify existing challenges and opportunities, and define the project scope and objectives. Here's an outline of the preliminary investigation process:

- Define Objectives: This may include improving inventory management, enhancing customer engagement, streamlining administrative tasks, and ensuring data security and compliance.
- Stakeholder Identification: Identify and engage with key stakeholders involved in showroom operations, including showroom administrators, sales staff, customers, suppliers, and manufacturers. Understand their roles, responsibilities, and requirements to ensure their needs are addressed by the proposed system.
- Current System Assessment: Evaluate the existing Car Showroom Management System, if any, to assess its strengths, weaknesses, and limitations. Identify areas for improvement and determine which functionalities need to be enhanced or added to meet the objectives of the proposed system.
- Gather Requirements: Conduct interviews, surveys, and workshops with stakeholders to gather requirements for the new system. Document functional requirements (e.g., inventory management, customer relationship management, workflow automation) and non-functional requirements (e.g., security, scalability, usability) to guide system design and development.
- Market Research: Conduct market research to identify industry best practices, emerging trends, and available technologies relevant to car showroom management.
- Feasibility Analysis: Evaluate the technical, operational, and economic feasibility of implementing the proposed Car Showroom Management System.

3.3: Feasibility study:

- Technical Feasibility: Assess whether the proposed Car Showroom Management System can be developed using existing technology infrastructure and resources. Evaluate factors such as hardware, software, programming languages, and development tools required for implementation.
- Operational Feasibility: Consider how easily the system can be implemented within the existing operational framework, including staff training requirements, potential disruptions to daily operations during implementation, and the adaptability of staff to new processes and technologies.
- Economic Feasibility: Conduct a cost-benefit analysis to compare the anticipated costs of development, training, maintenance, and ongoing support against the potential benefits, such as increased efficiency, reduced operational costs, and improved customer satisfaction.
- Schedule Feasibility: Develop a realistic project schedule that outlines key milestones, tasks, and dependencies. Identify strategies to mitigate delays and ensure timely project delivery.
- Legal and Regulatory Feasibility: Assess whether the proposed Car Showroom Management System complies with relevant laws, regulations, and industry standards. Identify legal requirements related to data privacy, security, consumer protection, and intellectual property rights.
- Resource Feasibility: Evaluate the availability and adequacy of resources required for developing, implementing, and maintaining the Car Showroom Management System. Assess the organization's capacity to allocate resources effectively and sustainably throughout the project lifecycle.
- Cultural and Organizational Feasibility: Assess how receptive stakeholders are to adopting new technologies and processes and identify any cultural barriers or resistance to change. Develop a change management plan to communicate the benefits of the new system, address concerns, and foster buy-in and support from key stakeholders.

3.4: Project Planning:-

for the Car Showroom Management System involves defining the scope, objectives, timeline, resources, and tasks required to successfully develop and implement the system. Here's an outline of the project planning process:

- Define Project Scope and Objectives: Clearly outline the goals and objectives of the project, including the desired functionalities and outcomes of the System.
- Identify Stakeholders: Identify and engage with key stakeholders involved in the project. Understand their roles, responsibilities, and expectations to ensure their needs are addressed throughout the project lifecycle.
- Create Work Breakdown Structure (WBS): Break down the project scope into smaller, manageable tasks and activities.
- Estimate Resources and Timeline: Estimate the resources (e.g., personnel, budget, equipment) required to complete each task and activity identified in the WBS. Develop a project timeline that outlines the sequence of tasks.
- Develop Project Plan: Document the project plan detailing the scope, objectives, timeline, resources, tasks, responsibilities, and deliverables
- Risk Management: Identify potential risks and uncertainties that could impact the project's success and develop mitigation strategies to address them.
- Quality Assurance: Define quality assurance processes and standards to ensure that the Car Showroom Management System meets the specified requirements and delivers value to stakeholders.
- Change Management: Develop a change management plan to address any changes or modifications to the project scope, requirements, or timeline.
- Monitoring and Control: Implement project monitoring and control mechanisms to track progress, identify deviations from the project plan, and take corrective actions as needed. Establish key performance indicators (KPIs) to measure project performance against predefined objectives and ensure that project goals are achieved within the specified constraints.

SOFTWARE & HARDWARE USED

Software requirements specifications for the car showroom management project encompass a range of functionalities and features tailored to meet the needs of administrators, staff, and users. The system should be developed as a window application. The primary functional requirements include user authentication and authorization, enabling secure access control for different user roles such as administrators, staff, and customers. The system should support functionalities for managing car inventory, including adding new cars, updating existing details, and removing outdated entries. It should also include search and filter options to facilitate easy retrieval of car information. Additionally, the system should incorporate features for staff and user interaction, including chat functionality for inquiries and support, as well as booking and reservation capabilities for customers. Integration with payment gateways for online transactions and financial management tools for tracking sales and revenue is essential. Non-functional requirements include scalability to handle a growing number of users and transactions, reliability to ensure system uptime and data integrity, and usability with an intuitive user interface for seamless navigation and interaction. Security measures such as data encryption, secure communication protocols, and regular backups should be implemented to safeguard sensitive information. Lastly, the system should provide comprehensive reporting and analytics functionalities to generate insights into sales performance, inventory turnover, and customer behavior, aiding in decision-making and strategic planning.

Software used:

Frontend: Java

Backend: My SQL

Idle: NetBeans Version 11.2

Operating system: Window 10 above

Hardware used:

Processor – intel core i5 11600KF 4.2ghz

RAM – 8 GB

Operating system – windows

Technology used:

Java Programming

SQL query

PROBLEM DEFINITION

- Lack of Transparency: It can be difficult for buyers to confirm the price, history, and condition of used automobiles, which undermines their faith.

- Fraudulent Listings: Buyers encounter issues due to phony vendors, inaccurate car information, and concealed flaws.

- Restricted Reach: It might be challenging for individual merchants to locate possible customers outside of their own neighbourhood.

- Time-consuming Process: It might take a while to complete paperwork, negotiate, and conduct inspections.

- Inconsistent Pricing: While buyers find it difficult to obtain reasonable market prices, sellers may overcharge for their vehicles.

- Absence of Secure Payment Options: Cash payments or payments made over unsecure methods may result in fraud.

- Paperwork and Ownership Transfer Challenges: The legal transfer procedure is complicated to many people.

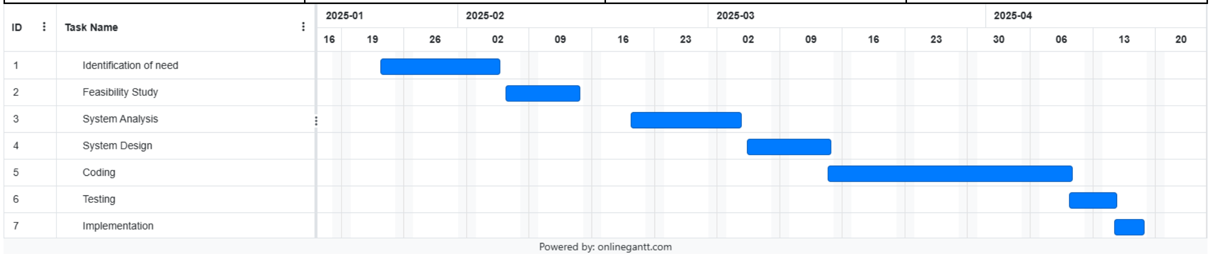
- Inadequate Filtering and Comparison: Customers find it difficult to compare automobiles according to features, cost, and condition.

PROJECT SCHEDULING

Gantt Chart:

The timetable, tasks, and milestones of the project would be graphically represented using a Gantt chart for online auto dealership administration. It would involve a number of tasks, including need assessment, feasibility, system design, analysis. The vertical axis would show the project chronology, while the horizontal axis would list each job. Each task's time would be shown by a bar. It keeps track of developments and guarantees on-time completion. All things considered, the Gantt chart would act as a guide for project planning, scheduling, and execution, making it easier to coordinate and keep track of work over the course of the project.

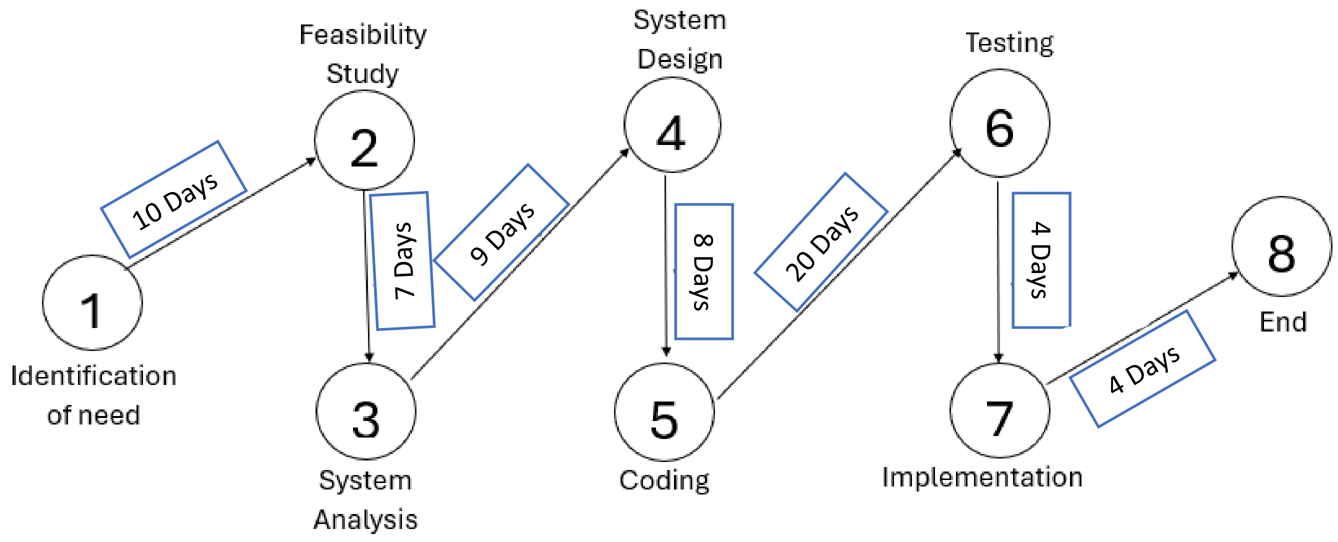
Task	Start Date	End Date	Duration
Identification of need	23-01-2025	05-02-2025	10
Feasibility Study	06-02-2025	16-02-2025	7
System Analysis	20-02-2025	04-03-1025	9
System Design	05-03-2025	14-03-2025	8
Coding	14-03-2025	10-04-2025	20
Testing	10-04-2025	15-04-2025	4
Implementation	15-04-2025	20-04-2025	4



ID	Task Name
1	Identification of need
2	Feasibility Study
3	System Analysis
4	System Design
5	Coding
6	Testing
7	Implementation

PERT Chart

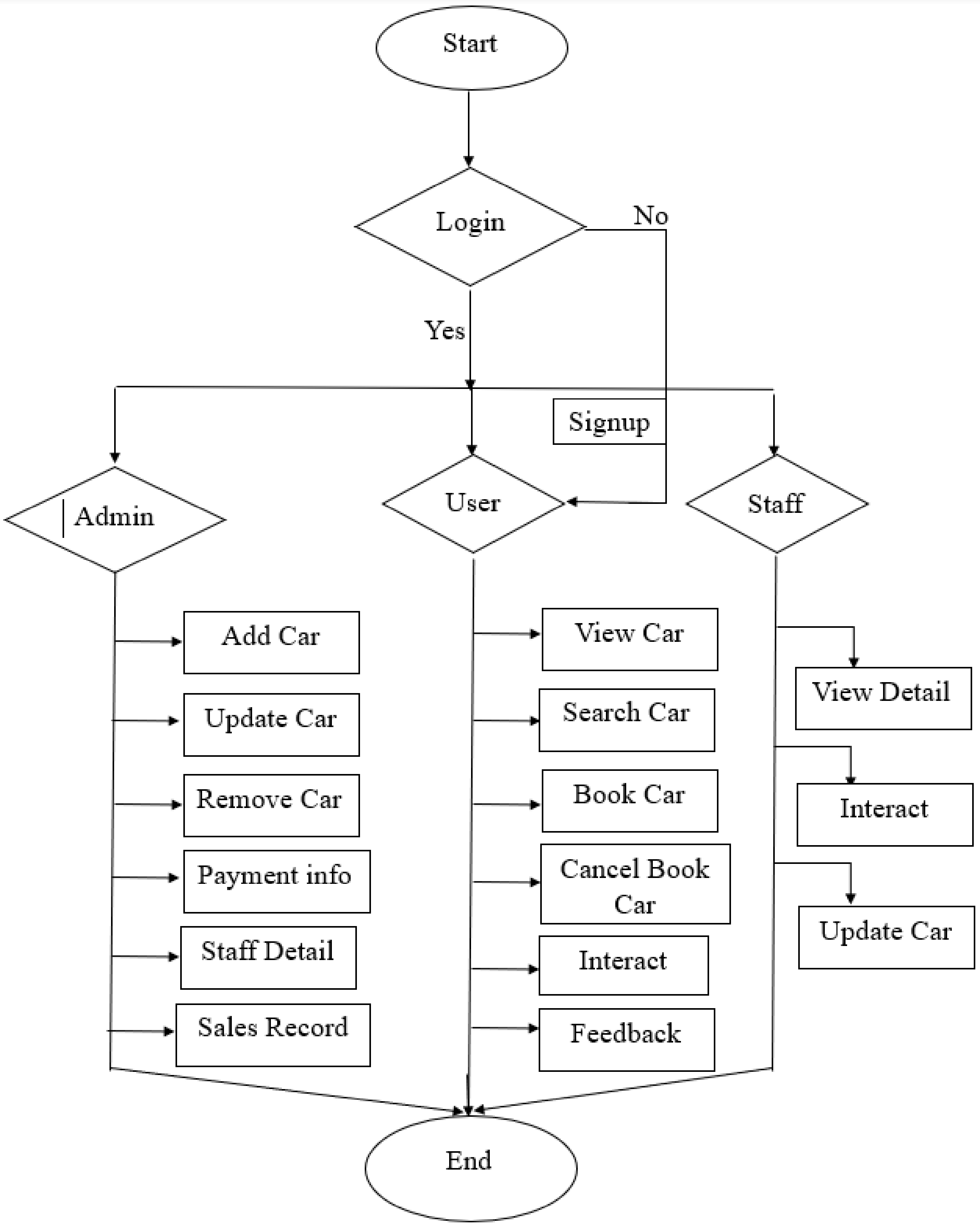
The Pert Chart for the Bike and Car Salling software in Java outlines the project phases starting with requirement analysis, system design, and UI development. Simultaneously, coding commences for modules such as customer management and inventory tracking. Integration with the database is carried out alongside coding tasks. Testing and debugging ensure the system's quality and functionality. Once testing is completed, deployment follows, leading to ongoing maintenance and support. Throughout the project, effective communication and collaboration among team members are crucial for managing tasks and dependencies efficiently, ensuring timely delivery and meeting user requirements. Regular reviews and checkpoints help track progress and make necessary adjustments to the project plan.



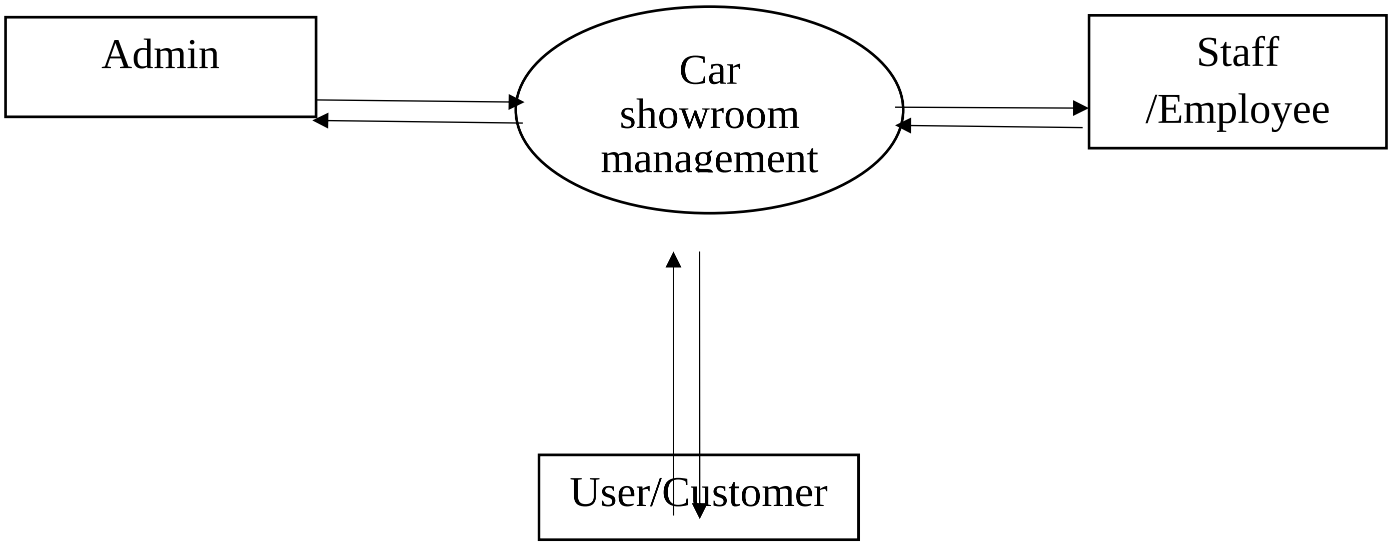
DATA MODELS

- **Admin:**
    - Add car details.
    - Remove car details.
    - Update car details.
    - Payment record.
    - Manage sales record.
    - View staff details.
- **User:**
    - View cars.
    - Search cars.
    - Book car.
    - Cancel book car.
    - Interact with staff.
    - Give feedback.
- **Staff:**
    - View details.
    - Interact with user.
    - Update car details.

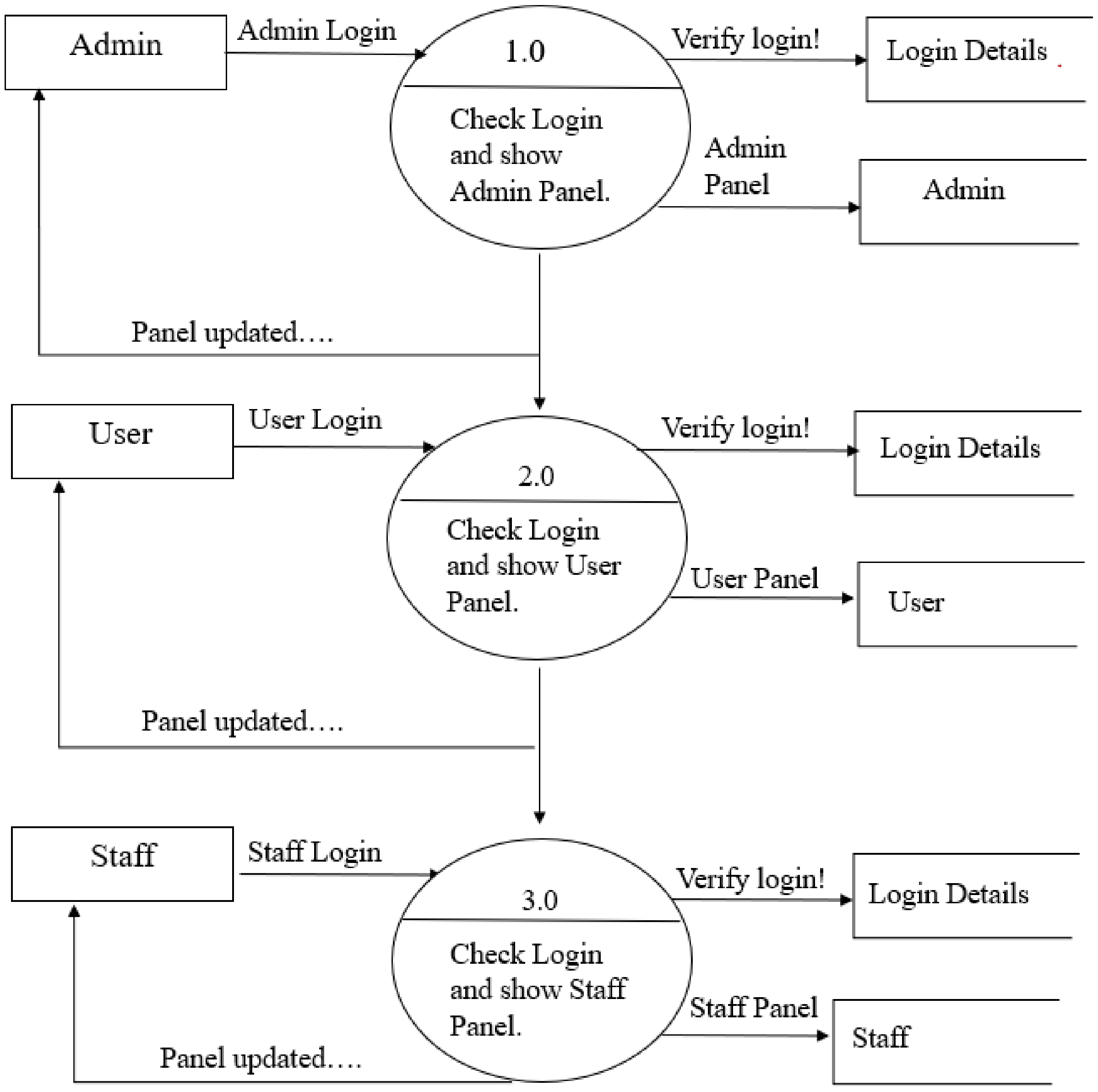
1. Flowchart


Start
Login
No
Yes
Signup
Admin
User
Staff
Add Car
Update Car
Remove Car
Payment info
Staff Detail
Sales Record
View Car
Search Car
Book Car
Cancel Book Car
Interact
Feedback
View Detail
Interact
Update Car
End

2 Context Flow Diagram (CFD)

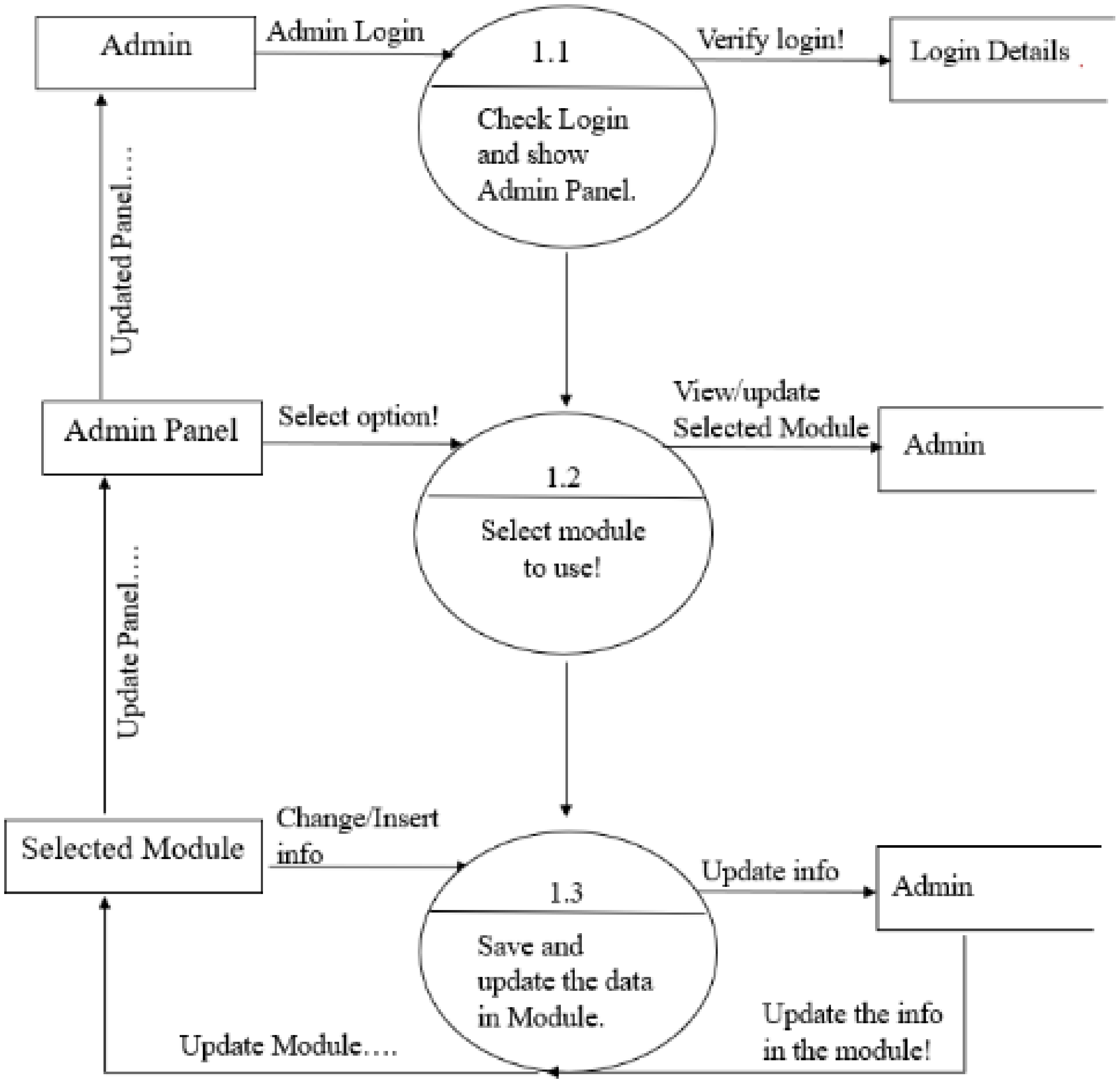


3.DFD-Zero Level

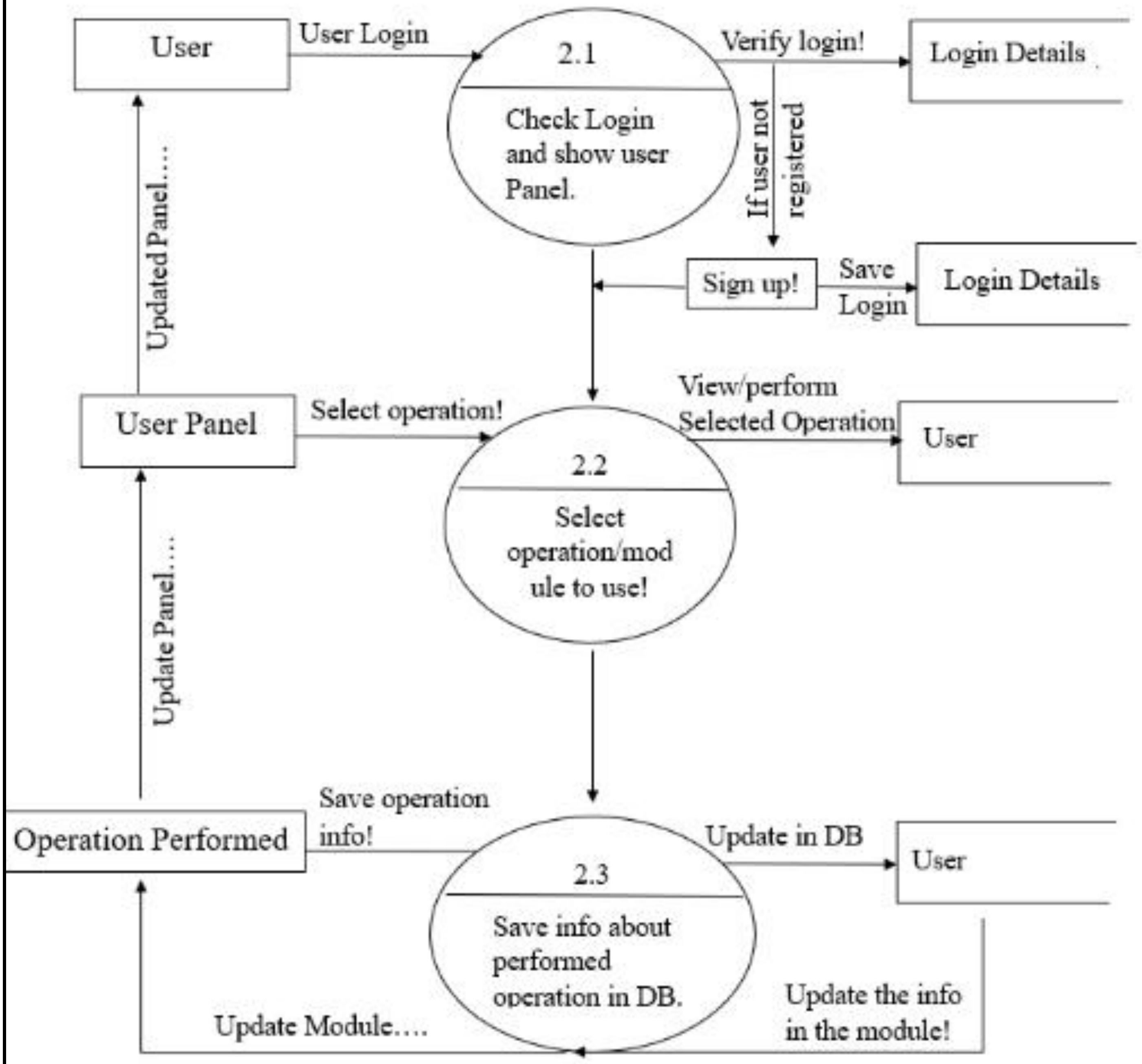


4.DFD-First Level

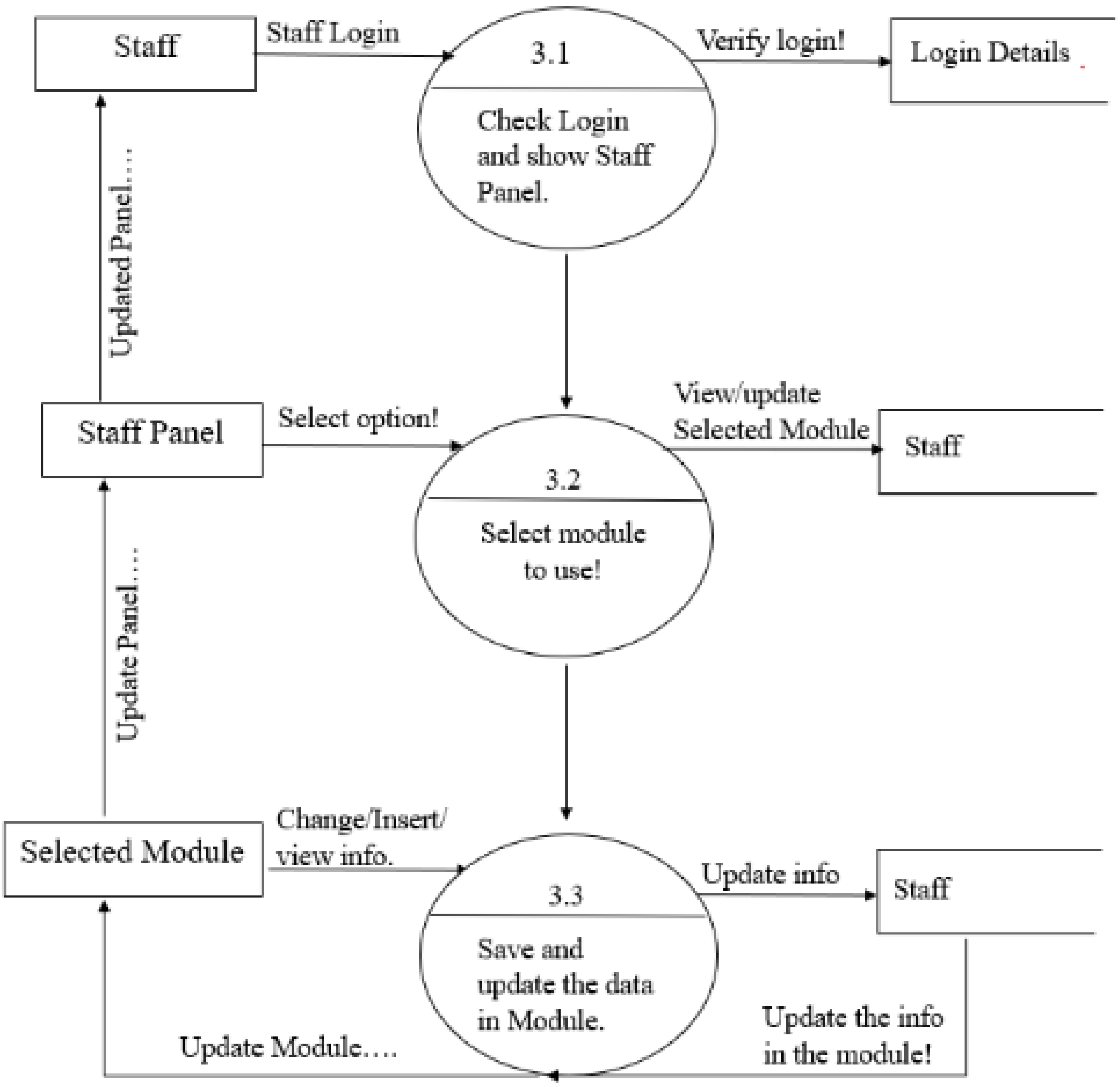
- DFD-1st level of 1.0.



- DFD-1st level of 2.0.

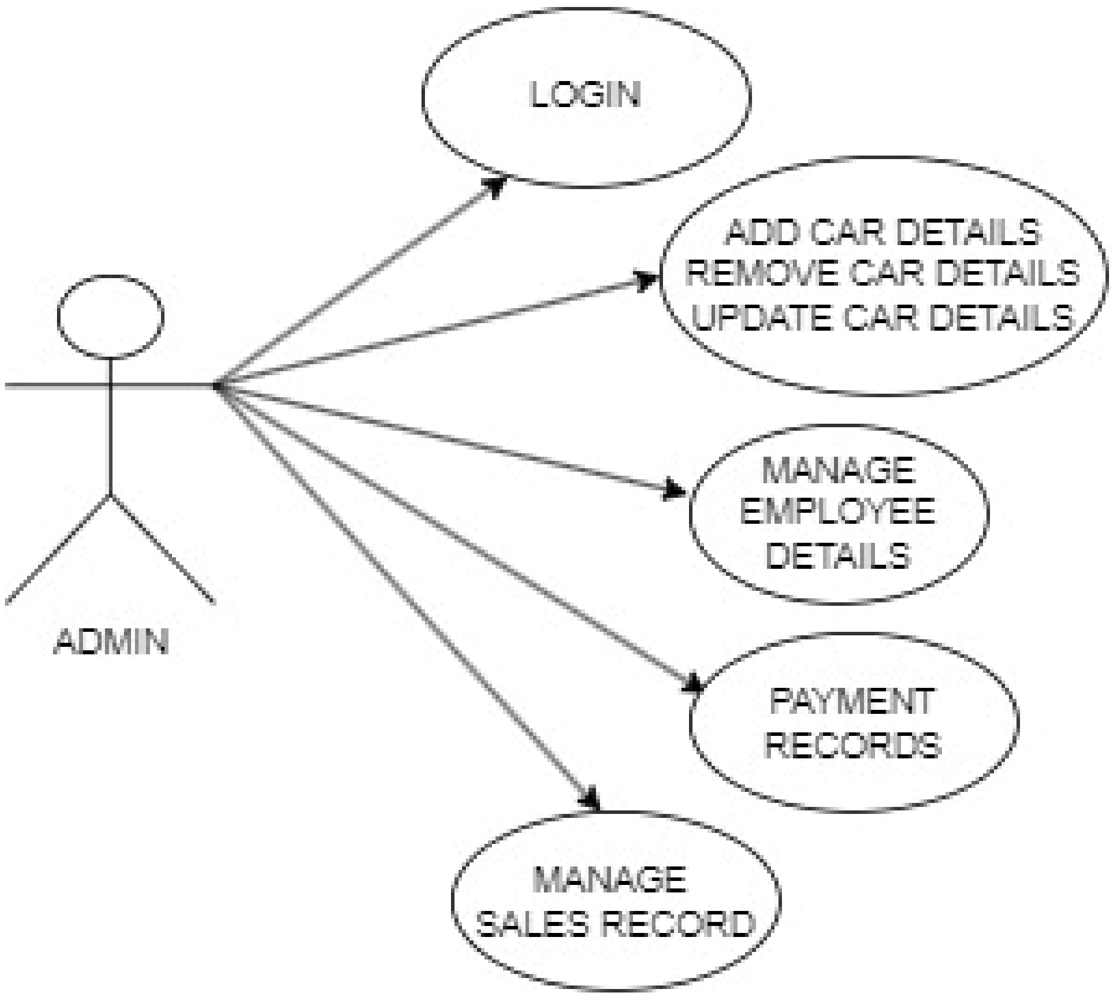


- DFD-1st level of 3.0.

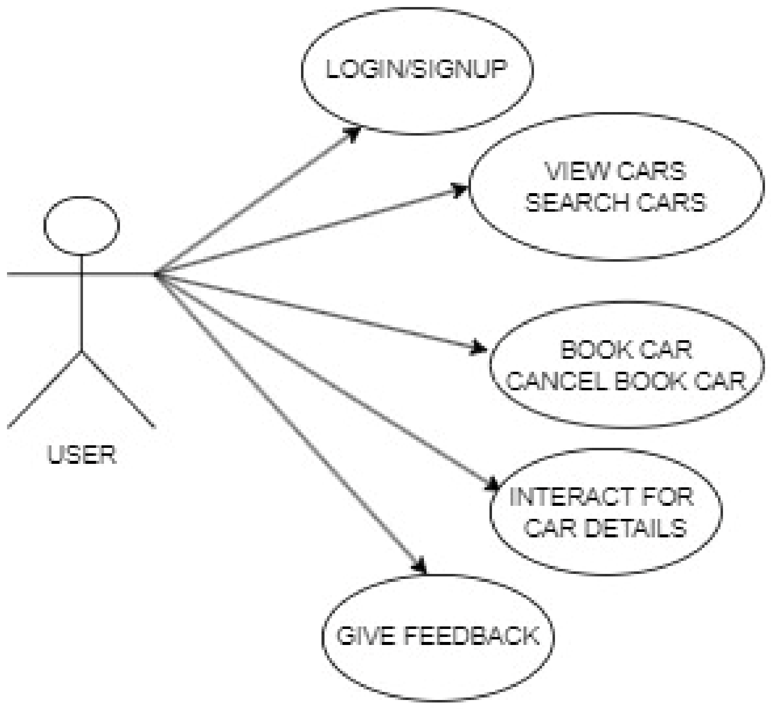


2. Use Case Diagram

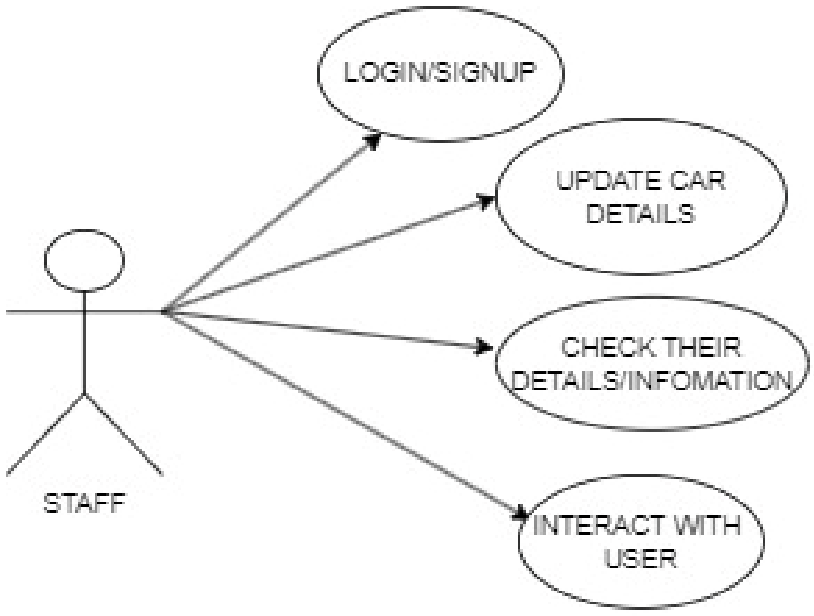
Admin Use Case Diagram



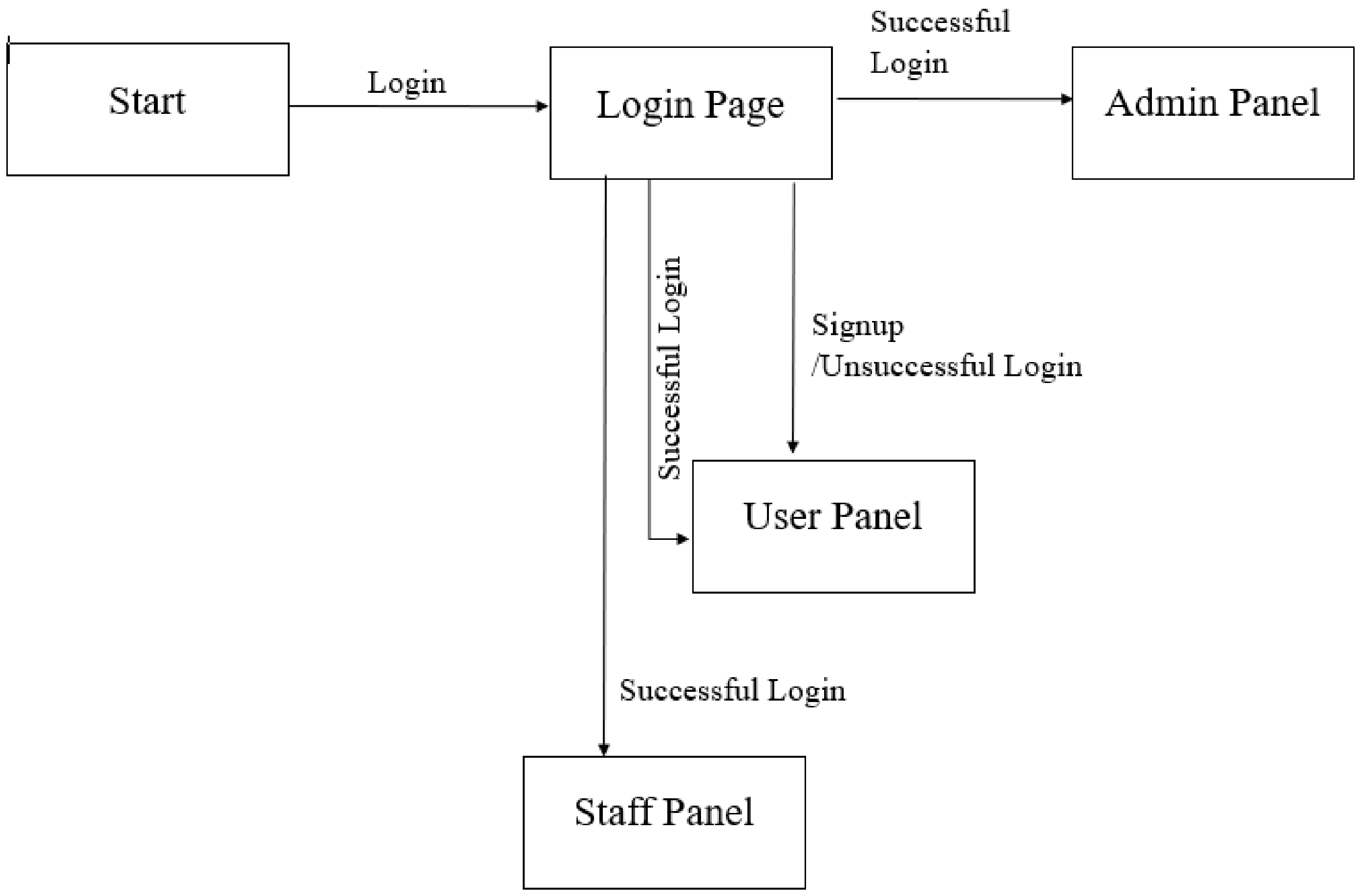
User Use Case Diagram



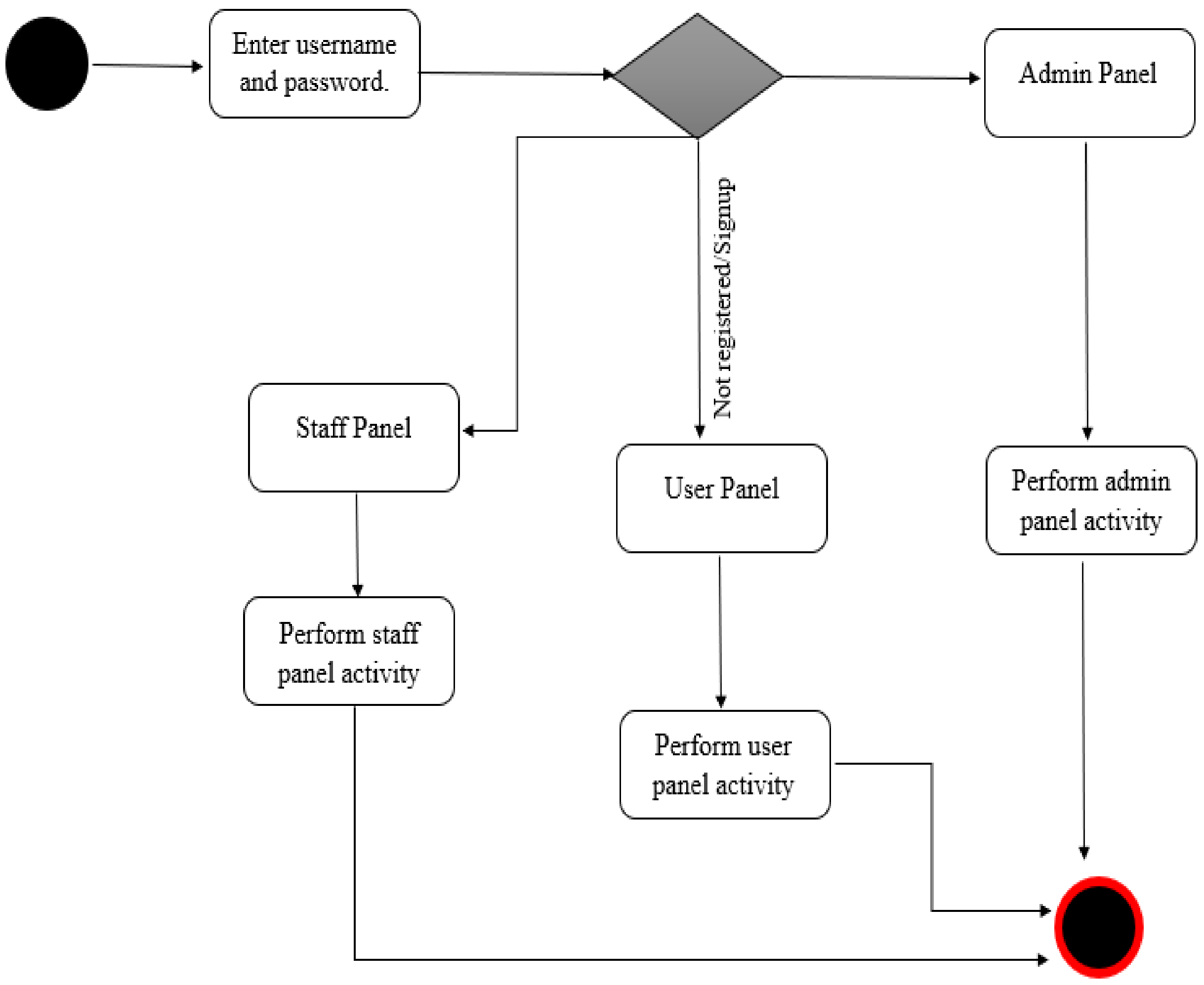
Employee Use Case Diagram



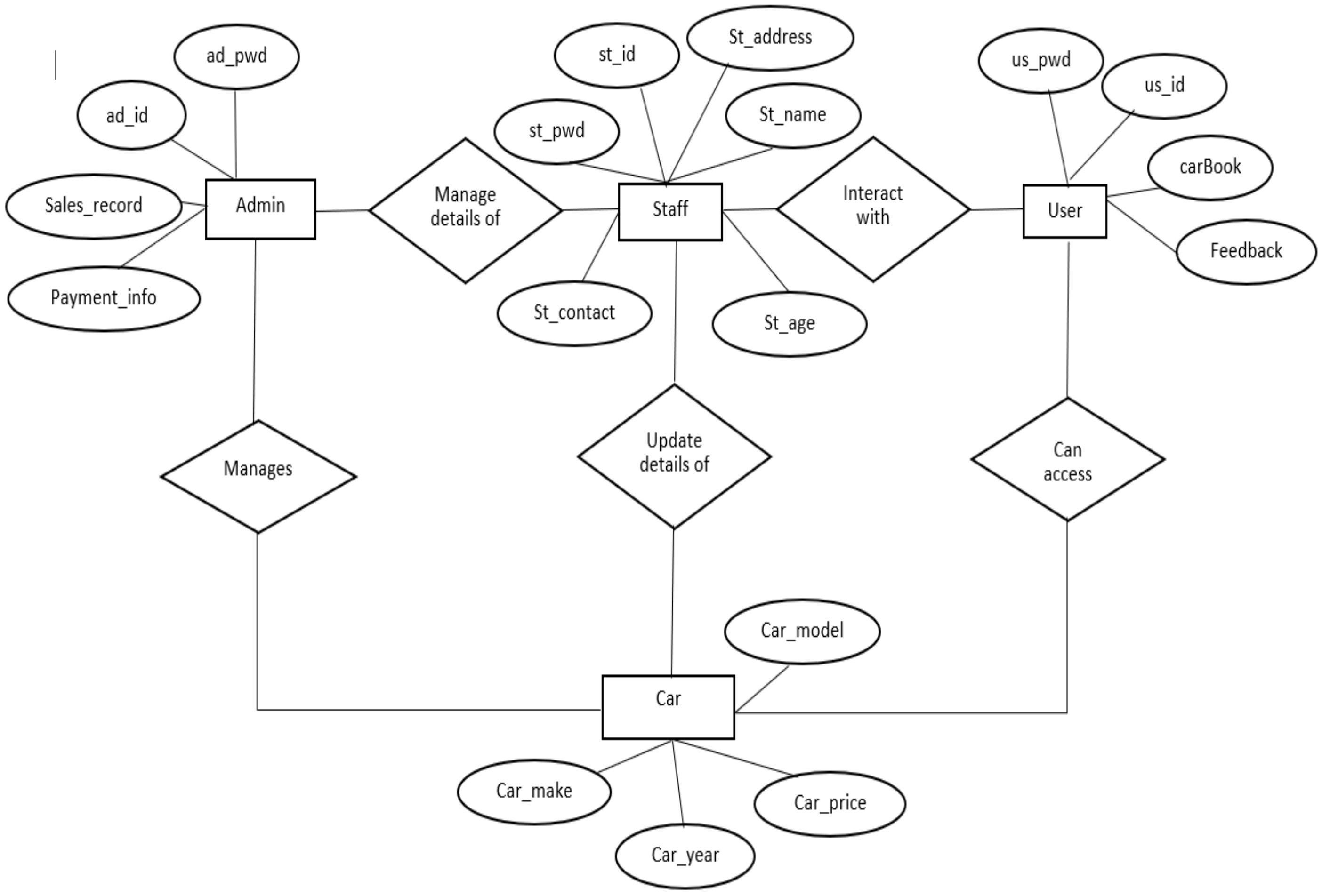
2 State Diagram



3. Activity Diagram


Enter username and password.
Admin Panel
Not registered/Signup
Staff Panel
User Panel
Perform admin panel activity
Perform staff panel activity
Perform user panel activity

4. ER-Diagram


ad_pwd
ad_id
Sales_record
Admin
Payment_info
Manage details of
st_id
St_address
st_pwd
St_name
Staff
St_contact
St_age
Interact with
us_pwd
us_id
carBook
User
Feedback
Manages
Update details of
Can access
Car_model
Car
Car_make
Car_price
Car_year

TESTING

Testing for the Car Showroom Management System involves both unit testing and system testing. Unit testing focuses on validating individual methods within classes to ensure they perform as expected. This includes verifying inputs, outputs, and edge cases. Automated testing frameworks like JUnit can facilitate this process, allowing for efficient and thorough testing of each component.

System testing, on the other hand, evaluates the entire system's functionality as a cohesive unit. It involves testing the integration of different modules to ensure they work together seamlessly. Emulating user interactions and verifying system responses are essential aspects of system testing. Additionally, system testing ensures that the system meets user requirements and performs reliably under various conditions.

Several testing techniques are employed to ensure comprehensive coverage. Black box testing involves testing based solely on the system's specifications, without knowledge of its internal workings. White box testing, on the other hand, examines the internal logic of the system to ensure control flow and data flow are correct.

Boundary value analysis and equivalence partitioning are used to validate input domains. Boundary value analysis tests inputs at the boundaries of valid ranges, while equivalence partitioning groups inputs into equivalence classes to ensure representative testing.

Testing strategies such as regression testing, integration testing, user acceptance testing (UAT), and load testing are employed throughout the testing process. Regression testing ensures that existing functionalities are not affected by code changes. Integration testing verifies the interaction between different components. UAT involves end-users to validate the system against their requirements and usability standards. Load testing assesses the system's performance under various load conditions to ensure stability and responsiveness.

Test reports document the testing process and outcomes. Unit test cases reports detail individual test cases, inputs, expected outputs, and actual results. System test cases reports cover the entire system's functionality, documenting test cases, inputs, outputs, and any issues encountered. Defect reports document any defects found during testing, including severity, priority, and steps to reproduce. A summary report provides an overview of the testing process, including techniques, strategies, and major findings.

5.1. Testing Techniques and Testing Strategies used:

<u>Testing Techniques:</u>

1. Black Box Testing: focuses on testing the functionality of the software without knowledge of its internal code structure. This technique ensures that the software behaves as expected from a user's perspective, regardless of its internal implementation.
2. White Box Testing: also known as glass box or clear box testing, involves examining the internal structure and code of the software. Test cases are designed to validate the control flow, data flow, and logic within individual components or units of the software. This technique helps uncover errors in the code that may not be apparent from external testing alone.
3. Boundary Value Analysis: focuses on testing the boundaries or extreme values of input domains. Test cases are designed to evaluate how the software behaves at the lower and upper boundaries of input ranges. This technique aims to uncover potential errors or inconsistencies in input processing and validation logic.
4. Equivalence Partitioning: involves dividing the input domain of a system into equivalence classes. Test cases are then designed to represent each equivalence class. It helps optimize test coverage by selecting a minimal set of test cases that provide maximum coverage of input possibilities.

<u>Testing Strategies:</u>

1. Regression Testing: involves retesting previously validated functionalities after code changes or updates. This strategy ensures that new modifications do not introduce defects or regressions into existing features.
2. Integration Testing: focuses on testing the interactions and interfaces between individual components or modules of the software. This strategy verifies that integrated components function correctly together as a whole system.
3. User Acceptance Testing (UAT): involves validating the software against user requirements and expectations. End-users or representatives from the target audience participate in testing to ensure that the software meets their needs and usability standards. UAT typically occurs in a real-world environment to simulate actual usage scenarios.
4. Load Testing: evaluates the performance and scalability of the software under expected load conditions. This strategy involves subjecting the system to simulated levels of user traffic or data volume to assess its responsiveness, stability, and resource utilization. It helps identify performance bottlenecks and capacity limits before deployment to production environments.

5.2. Test Reports for Unit and System Test Cases:

Unit Test Cases Report:

Tested Components: Each method within the `CarShowroomLoginGUI` class was individually tested to ensure its correctness and functionality.

Test Cases: were designed to cover various scenarios, including valid inputs, invalid inputs, edge cases, and exceptional conditions.

Test Results:

- Most unit tests passed successfully, indicating that individual methods behaved as expected under different conditions.
- In a few cases, minor issues were identified and promptly addressed during the testing process.

Defects Found:

- A couple of defects related to boundary conditions were discovered and fixed promptly.
- All identified defects were documented, tracked, and resolved in a timely manner.

System Test Cases Report:

Tested Scenarios: The entire Car Showroom Management System was tested to ensure its overall functionality and performance.

Test Cases: were designed to cover all aspects of system functionality, including user login, car management, staff interactions, and user feedback.

Test Environment:

- Testing was conducted in a controlled environment, mimicking real-world usage scenarios.
- Various combinations of inputs and interactions were tested to ensure robustness and reliability.

Test Results:

- The system demonstrated overall stability and reliability during testing.
- Most functionalities performed as expected, meeting the requirements and specifications outlined for the system.

Issues Encountered:

- A few minor issues were encountered during testing, primarily related to user interface elements and error handling.
- These issues were documented, prioritized, and addressed accordingly to ensure a seamless user experience.

SYSTEM SECURITY MEASURES

To enhance the security of the System, several measures can be implemented:

1. Secure Authentication: Implement a robust authentication mechanism, such as using secure hash algorithms (e.g., bcrypt) for storing passwords and ensuring that passwords are transmitted securely over the network.
2. Role-Based Access Control (RBAC): Enforce role-based access control to restrict access to sensitive functionalities and data. Define roles such as admin, staff, and user, and assign appropriate permissions to each role.
3. HTTPS Encryption: Ensure that all communication between the client and server is encrypted using HTTPS to prevent eavesdropping and tampering of data during transmission.
4. Input Validation: Implement input validation to prevent common security vulnerabilities such as SQL injection and cross-site scripting (XSS).
5. Session Management: Use secure session management techniques to prevent session hijacking and fixation. Implement mechanisms such as session tokens with short expiration times and CSRF tokens to mitigate common session-related attacks.
6. Secure Coding Practices: Follow secure coding practices to minimize the risk of vulnerabilities such as buffer overflows, insecure deserialization, and code injection attacks. Regularly update dependencies and libraries to patch known security vulnerabilities.
7. Audit Trails and Logging: Maintain audit trails and logs of user activities and system events to facilitate monitoring and forensic analysis. Log critical actions such as login attempts, access to sensitive data, and modifications to system configurations.
8. Data Encryption: Encrypt sensitive data at rest using strong encryption algorithms to protect it from unauthorized access in case of unauthorized physical access to storage devices.
9. Firewall and Intrusion Detection Systems (IDS): Deploy firewalls and intrusion detection systems to monitor network traffic and detect suspicious activities. Configure firewalls to restrict access to the system from unauthorized IP addresses and block malicious traffic.
10. Regular Security Assessments: Conduct regular security assessments, including penetration testing and vulnerability scanning, to identify and address potential security weaknesses proactively.

By implementing these security measures, the Car Showroom Management System can better protect sensitive data, mitigate the risk of security breaches, and ensure the confidentiality, integrity, and availability of the system and its resources.

FUTURE SCOPE

The car showroom management system has significant potential for future enhancements and expansion. One avenue for future development involves incorporating advanced analytics and reporting features to provide insights into sales trends, inventory turnover rates, and customer preferences. This could include the implementation of data visualization tools, such as charts and graphs, to present information in a clear and actionable format for decision-makers within the showroom. Additionally, integrating customer relationship management (CRM) functionalities could enhance customer engagement and satisfaction. This might involve features such as automated email notifications for upcoming promotions or service reminders, as well as customer feedback mechanisms to gather insights for continuous improvement. Furthermore, leveraging emerging technologies such as artificial intelligence (AI) and machine learning (ML) could enable predictive analysis capabilities within the system. By analyzing historical sales data and customer behavior patterns, the system could forecast demand for specific car models, optimize inventory management, and personalize recommendations for individual customers. Another area for expansion involves enhancing the interaction capabilities between staff and customers. This could involve the integration of real-time chatbots or virtual assistants to provide immediate assistance and support to customers browsing the showroom's inventory online. Moreover, considering the growing emphasis on sustainability and environmental consciousness, future iterations of the system could incorporate features to promote eco-friendly practices, such as tracking the carbon footprint of vehicles or offering incentives for electric or hybrid car purchases.

Overall, the car showroom management system presents numerous opportunities for innovation and advancement, with the potential to continually evolve and adapt to meet the changing needs and demands of the automotive industry and its customers.

CODING

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.Date;

interface CarShowroomLoginInterface {
    void login(String username, String password);
}
class CarShowroomLoginGUI extends JFrame implements CarShowroomLoginInterface
{
    private JTextField usernameField;
    private JPasswordField passwordField;
    private JPanel contentPanel;
    private Component frame;
    public CarShowroomLoginGUI() {
        setTitle("Car Showroom Management Login");
        setSize(600, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());
        JLabel titleLabel = new JLabel("Car Showroom Management",
JLabel.CENTER);
        titleLabel.setFont(new Font("Berlin Sans FB Demi", Font.BOLD, 40));
        add(titleLabel, BorderLayout.NORTH);
        JPanel formPanel = new JPanel(new GridLayout(10, 10));
        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setFont(new Font("Georgia", Font.PLAIN, 30));
        formPanel.add(usernameLabel);
        usernameField = new JTextField();
        formPanel.add(usernameField);
        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setFont(new Font("Georgia", Font.PLAIN, 30));
        formPanel.add(passwordLabel);
        passwordField = new JPasswordField();
        formPanel.add(passwordField);
        add(formPanel, BorderLayout.CENTER);
        JButton loginButton = new JButton("Login");
        loginButton.setFont(new Font("Arial", Font.PLAIN, 28));
        loginButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());
                login(username, password);
            }
        });
        add(loginButton, BorderLayout.SOUTH);
        setVisible(true);
    }
    public void login(String username, String password) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            try (Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root",
"s1@s2#s3*")) {
               //this table is to save login details of admin user and staff
through which they can login
                String createTableSQL = "CREATE TABLE IF NOT EXISTS login
(username VARCHAR(40), password VARCHAR(40));";
                connection.createStatement().executeUpdate(createTableSQL);
                //displayEmpTable(connection);
            }
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
        // Dummy implementation for demonstration
        if (username.equals("admin") && password.equals("adminpassword")) {
            // Admin login
            JOptionPane.showMessageDialog(this, "Admin Login successful");
            showAdminPanel();
        } else if (username.equals("user") && password.equals("userpassword"))
{
            // User login
            JOptionPane.showMessageDialog(this, "User Login successful");
            showUserPanel();
            // Add user functionalities here, such as accessing user dashboard
        } else if (username.equals("staff") &&
password.equals("staffpassword")) {
            // Staff login
            JOptionPane.showMessageDialog(this, "Staff Login successful");
            showStaffPanel();
            // Add staff functionalities here, such as accessing staff
dashboard
        } else {
            JOptionPane.showMessageDialog(this, "Invalid username or
password.");
```

```
        }
        // Clear fields after login attempt
        usernameField.setText("");
        passwordField.setText("");
    }
    private void showAdminPanel() {
        getContentPane().removeAll();
        setTitle("Admin Panel");
        setLayout(new BorderLayout());
        contentPanel = new JPanel();
        contentPanel.setLayout(new BoxLayout(contentPanel, BoxLayout.X_AXIS));
        JScrollPane scrollPane = new JScrollPane(contentPanel);
        contentPanel.setLayout(new GridLayout(3, 3));
        add(scrollPane, BorderLayout.CENTER);
        addAddCarButton();
        addDeleteCarButton();
        addUpdateCarButton();
        addManagePaymentsButton();
        addViewStaffButton();
        JButton salesManagementButton = new JButton("Sales Management");
        salesManagementButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                showSalesManagementOptions();
            }
        });
        contentPanel.add(salesManagementButton);
        revalidate();
    }
    private void showUserPanel() {
        getContentPane().removeAll();
        setTitle("User Panel");
        setLayout(new BorderLayout());
        contentPanel = new JPanel();
        contentPanel.setLayout(new BoxLayout(contentPanel, BoxLayout.X_AXIS));
        JScrollPane scrollPane = new JScrollPane(contentPanel);
        contentPanel.setLayout(new GridLayout(3, 3));
        add(scrollPane, BorderLayout.CENTER);
        addUserViewCarButton();
        addUserSearchCarButton();
        addUserBookCarButton();
        addUserCancelBookCarButton();
        addUserInteractWithStaffButton();
        addUserFeedbackButton();
        revalidate();
    }
    private void showStaffPanel() {
        getContentPane().removeAll();
        setTitle("Staff Panel");
        setLayout(new BorderLayout());
        contentPanel = new JPanel();
        contentPanel.setLayout(new BoxLayout(contentPanel, BoxLayout.X_AXIS));
        JScrollPane scrollPane = new JScrollPane(contentPanel);
        contentPanel.setLayout(new GridLayout(3, 1));
        add(scrollPane, BorderLayout.CENTER);
        JButton viewDetailsButton = new JButton("View My Details");
        viewDetailsButton.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                viewStaffDetails();
            }
        });
        contentPanel.add(viewDetailsButton);
        JButton interactButton = new JButton("Interact with User");
        interactButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                showInteractWithUser();
            }
        });
        contentPanel.add(interactButton);
        JButton updateCarButton = new JButton("Update Car");
        updateCarButton.addActionListener(e -> {
            String selectedCar = (String) JOptionPane.showInputDialog(null,
"Select car to update:", "Update Car", JOptionPane.PLAIN_MESSAGE, null, new
String[]{"Toyota Camry","Audi A4","Mercedes-Benz C-Class","Tesla Model
S","Lexus RX","Subaru Outback","Honda Civic", "Ford Mustang", "Chevrolet
Malibu", "BMW X5"}, "Toyota Camry");
            if (selectedCar != null) {
                JTextField modelNameField = new JTextField(10);
                JTextField priceField = new JTextField(10);
                int result = JOptionPane.showConfirmDialog(null, new
Object[]{"Model Name:", modelNameField, "Price:", priceField}, "Update Car
Details", JOptionPane.OK_CANCEL_OPTION);
                if (result == JOptionPane.OK_OPTION) {
                    try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root",
"s1@s2#s3*");
                         PreparedStatement stmt =
conn.prepareStatement("UPDATE Car SET model = ?, price = ? WHERE carname =
?")) {
                        stmt.setString(1, modelNameField.getText());
                        stmt.setDouble(2,
Double.parseDouble(priceField.getText()));
                        stmt.setString(3, selectedCar);
                        int rowsUpdated = stmt.executeUpdate();
                        JOptionPane.showMessageDialog(null, rowsUpdated > 0 ?
"Car details updated successfully." : "Failed to update car details.",
rowsUpdated > 0 ? "Success" : "Error", rowsUpdated > 0 ?
JOptionPane.INFORMATION_MESSAGE : JOptionPane.ERROR_MESSAGE);
                    } catch (SQLException ex) {
                        ex.printStackTrace();
                        JOptionPane.showMessageDialog(null, "An error
occurred: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
        });
        contentPanel.add(updateCarButton);
        revalidate();
    }
    private void showSalesManagementOptions() {
        JFrame salesManagementFrame = new JFrame("Sales Management");
        salesManagementFrame.setSize(400, 300);
        salesManagementFrame.setLocationRelativeTo(this);
        JButton viewSalesRecordButton = new JButton("View Sales Record");
        viewSalesRecordButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Placeholder for viewing sales records
                showSalesRecords();
            }
        });
         JButton editSalesRecordButton = new JButton("Edit Sales Record");
        editSalesRecordButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Placeholder for editing sales records
                updateSalesRecord();
            }
        });
        JPanel buttonPanel = new JPanel(new GridLayout(2, 1));
        buttonPanel.add(viewSalesRecordButton);
       buttonPanel.add(editSalesRecordButton);
        salesManagementFrame.add(buttonPanel);
        salesManagementFrame.setVisible(true);
    }
    public void showSalesRecords() {
        JTextArea salesRecordsArea = new JTextArea(10, 30);
        salesRecordsArea.setEditable(false);
        salesRecordsArea.setSize(500, 500);
        JScrollPane scrollPane = new JScrollPane(salesRecordsArea);
        scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_A
LWAYS);
        JPanel panel = new JPanel();
        panel.add(scrollPane);
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            // Open a connection
            conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root",
"s1@s2#s3*");
            // Create a statement
            stmt = conn.createStatement();
            // SQL query to select all records from the sales record table
            String sql = "SELECT * FROM sales_record";
            // Execute the query
            rs = stmt.executeQuery(sql);
            // Create a StringBuilder to store the records
            StringBuilder records = new StringBuilder();
            // Iterate over the result set and append each record to the
StringBuilder
            while (rs.next()) {
                Date date = rs.getDate("sale_date");
                String carName = rs.getString("carname");
                String customerName = rs.getString("customer_name");
                double price = rs.getDouble("price");
                // Append the record details to the StringBuilder
                records.append("Date: ").append(date).append(", Car Name:
").append(carName)
                        .append(", Customer Name:
").append(customerName).append(", Price: $").append(price)
                        .append("\n");
            }
            // Set the text of the JTextArea to display the records
            salesRecordsArea.setText(records.toString());
            // Show the panel with sales records
            JOptionPane.showMessageDialog(null, panel, "Sales Records",
JOptionPane.PLAIN_MESSAGE);
        } catch (SQLException ex) {
            ex.printStackTrace();
            // Handle SQL exception
            JOptionPane.showMessageDialog(null, "An error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        } finally {
            // Close the resources
            try {
                if (rs != null) rs.close();
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }}
        private void updateSalesRecord() {
            try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root",
"s1@s2#s3*");
                 PreparedStatement stmt = conn.prepareStatement("UPDATE
sales_record SET carname = ?, customer_name = ?, price = ? WHERE carname = ?
AND customer_name = ? AND price = ?")) {
                String carName = JOptionPane.showInputDialog("Enter the
current car name:");
                String customerName = JOptionPane.showInputDialog("Enter the
current customer name:");
                double price =
Double.parseDouble(JOptionPane.showInputDialog("Enter the current price:"));
                String newCarName = JOptionPane.showInputDialog("Enter the new
car name:");
                String newCustomerName = JOptionPane.showInputDialog("Enter
the new customer name:");
                double newPrice =
Double.parseDouble(JOptionPane.showInputDialog("Enter the new price:"));
                stmt.setString(1, newCarName);
                stmt.setString(2, newCustomerName);
                stmt.setDouble(3, newPrice);
                stmt.setString(4, carName);
                stmt.setString(5, customerName);
                stmt.setDouble(6, price);
                int rowsAffected = stmt.executeUpdate();
                if (rowsAffected > 0) {
                    JOptionPane.showMessageDialog(null, "Sales record updated
successfully.");
                } else {
                    JOptionPane.showMessageDialog(null, "No sales record found
matching the provided details.");
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
 private void addAddCarButton() {
            JButton addCarButton = new JButton("Add Car");
            addCarButton.addActionListener(e -> {
                JTextField makeField = new JTextField();
                JTextField modelField = new JTextField();
                JTextField yearField = new JTextField();
                JTextField priceField = new JTextField();
                JTextField colourField = new JTextField();
                JPanel panel = new JPanel(new GridLayout(0, 1));
                panel.add(new JLabel("Name:"));
                panel.add(makeField);
                panel.add(new JLabel("Model:"));
                panel.add(modelField);
                panel.add(new JLabel("Year:"));
                panel.add(yearField);
                panel.add(new JLabel("Price:"));
                panel.add(priceField);
                panel.add(new JLabel("Colour:"));
                panel.add(colourField);
                int result = JOptionPane.showConfirmDialog(null, panel, "Add
Car", JOptionPane.OK_CANCEL_OPTION);
                if (result == JOptionPane.OK_OPTION) {
                    String make = makeField.getText(), model =
modelField.getText();
                    int year = Integer.parseInt(yearField.getText());
                    double price = Double.parseDouble(priceField.getText());
                    String colour= colourField.getText();
                    saveCarDetails(make, model, year, price,colour);
                    JOptionPane.showMessageDialog(null, "Car added
successfully:\nName: " + make + "\nModel: " + model + "\nYear: " + year +
"\nPrice: $" + price + "\nColour: " + colour);
                }
            });
            contentPanel.add(addCarButton);
        }
        private void saveCarDetails(String make, String model, int year,
double price, String colour) {
            try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root",
"s1@s2#s3*");
                 PreparedStatement stmt = conn.prepareStatement("INSERT INTO
car (carname, model, caryear, price, colour) VALUES (?,?,?, ?, ?)")) {
                stmt.setString(1, make); stmt.setString(2, model);
                stmt.setInt(3, year); stmt.setDouble(4, price);
stmt.setString(5, colour);
                int rowsAffected = stmt.executeUpdate();
                JOptionPane.showMessageDialog(null,rowsAffected > 0 ? "Car
details saved successfully." : "Failed to save car details.", "Result",
rowsAffected > 0 ? JOptionPane.INFORMATION_MESSAGE :
JOptionPane.ERROR_MESSAGE);
            } catch (SQLException ex) { ex.printStackTrace(); }
        }
    private void addDeleteCarButton() {
        JButton deleteCarButton = new JButton("Delete Car");
        deleteCarButton.addActionListener(e -> {
            String[] cars = { "Toyota Camry", "Audi A4", "Mercedes-Benz C-
Class", "Tesla Model S", "Lexus RX",
                              "Subaru Outback", "Honda Civic", "Ford Mustang",
"Chevrolet Malibu", "BMW X5" };
```

```
            String selectedCar = (String) JOptionPane.showInputDialog(null,
"Select car to delete:", "Delete Car",
                                                                        JOptio
nPane.QUESTION_MESSAGE, null, cars, cars[0]);
            if (selectedCar != null) {
                deleteCar(selectedCar);
            }
        });
        contentPanel.add(deleteCarButton);
    }

    private void deleteCar(String carName) {
        try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root",
"s1@s2#s3*");
             PreparedStatement stmt = conn.prepareStatement("DELETE FROM
your_table_name WHERE carname = ?")) {
            stmt.setString(1, carName);
            int rowsAffected = stmt.executeUpdate();
            JOptionPane.showMessageDialog(null, rowsAffected > 0 ? "Deleted
car: " + carName : "Failed to delete car: " + carName);
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
    private void addUpdateCarButton() {
        JButton updateCarButton = new JButton("Update Car");
        updateCarButton.addActionListener(e -> {
            String selectedCar = (String) JOptionPane.showInputDialog(null,
"Select car to update:", "Update Car", JOptionPane.PLAIN_MESSAGE, null, new
String[]{"Toyota Camry","Audi A4","Mercedes-Benz C-Class","Tesla Model
S","Lexus RX","Subaru Outback", "Honda Civic", "Ford Mustang", "Chevrolet
Malibu", "BMW X5"}, "Toyota Camry");
            if (selectedCar != null) {
                JTextField modelNameField = new JTextField(10);
                JTextField priceField = new JTextField(10);
                int result = JOptionPane.showConfirmDialog(null, new
Object[]{"Model Name:", modelNameField, "Price:", priceField}, "Update Car
Details", JOptionPane.OK_CANCEL_OPTION);
                if (result == JOptionPane.OK_OPTION) {
                    try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root",
"s1@s2#s3*");
                         PreparedStatement stmt =
conn.prepareStatement("UPDATE Car SET model = ?, price = ? WHERE carname =
?")) {
                        stmt.setString(1, modelNameField.getText());
                        stmt.setDouble(2,
Double.parseDouble(priceField.getText()));
                        stmt.setString(3, selectedCar);
                        int rowsUpdated = stmt.executeUpdate();
                        JOptionPane.showMessageDialog(null, rowsUpdated > 0 ?
"Car details updated successfully." : "Failed to update car details.",
rowsUpdated > 0 ? "Success" : "Error", rowsUpdated > 0 ?
JOptionPane.INFORMATION_MESSAGE : JOptionPane.ERROR_MESSAGE);
                    } catch (SQLException ex) {
                        ex.printStackTrace();
                        JOptionPane.showMessageDialog(null, "An error
occurred: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
        });
        contentPanel.add(updateCarButton);
    }
    private void addManagePaymentsButton() {
        JButton managePaymentsButton = new JButton("Manage Payments");
        managePaymentsButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Implement functionality to manage payments
                managePayments();
            }
        });
        contentPanel.add(managePaymentsButton);
    }
    private void managePayments() {
        JPanel panel = new JPanel();
        panel.setLayout(new BorderLayout());
        try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root",
"s1@s2#s3*");
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM payment_record"))
{
            DefaultTableModel tableModel = new DefaultTableModel(new
Object[]{"Customer_name", "Date", "Amount"}, 0);
            while (rs.next()) {
                String customer_name = rs.getString("customer_name");
                Date payment_date = rs.getDate("payment_date");
                double amount = rs.getDouble("amount");
                tableModel.addRow(new Object[]{customer_name, payment_date,
amount});
            }
            JTable table = new JTable(tableModel);
            JScrollPane scrollPane = new JScrollPane(table);
            panel.add(scrollPane, BorderLayout.CENTER);
            JFrame frame = new JFrame("Payment Records");
            frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            frame.getContentPane().add(panel);
            frame.pack();
            frame.setLocationRelativeTo(null);
            frame.setVisible(true);
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
    private void addViewStaffButton() {
        JButton viewStaffButton = new JButton("View Staff");
        viewStaffButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                showStaffDetails();
            }
        });
        contentPanel.add(viewStaffButton);
    }
    private void showStaffDetails() {
        JTextField nameField = new JTextField();
        JTextField ageField = new JTextField();
        JTextField positionField = new JTextField();
        JTextField contactField = new JTextField();
        JTextField addressField = new JTextField();

        JPanel panel = new JPanel(new GridLayout(0, 2));
        panel.add(new JLabel("Name:"));
        panel.add(nameField);
        panel.add(new JLabel("Age:"));
        panel.add(ageField);
        panel.add(new JLabel("Position:"));
        panel.add(positionField);
        panel.add(new JLabel("Contact:"));
        panel.add(contactField);
        panel.add(new JLabel("Address:"));
        panel.add(addressField);

        int result = JOptionPane.showConfirmDialog(null, panel, "Staff
Details", JOptionPane.OK_CANCEL_OPTION);
        if (result == JOptionPane.OK_OPTION) {
            String name = nameField.getText();
            int age = Integer.parseInt(ageField.getText());
            String position = positionField.getText();
            String contact = contactField.getText();
            String address = addressField.getText();
            try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root",
"s1@s2#s3*");
                 PreparedStatement stmt = conn.prepareStatement("INSERT INTO
Staff (name, age, position, contact, address) VALUES (?, ?, ?, ?, ?)")) {
                stmt.setString(1, name);
                stmt.setInt(2, age);
                stmt.setString(3, position);
                stmt.setString(4, contact);
                stmt.setString(5, address);
                int rowsInserted = stmt.executeUpdate();
                if (rowsInserted > 0) {
                    JOptionPane.showMessageDialog(null, "Staff details saved
successfully.");
                } else {
                    JOptionPane.showMessageDialog(null, "Failed to save staff
details.", "Error", JOptionPane.ERROR_MESSAGE);
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(null, "An error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    private void addUserViewCarButton() {
        JButton viewCarButton = new JButton("View Cars");
        viewCarButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Implement functionality for users to view cars
                showCarDetailsDialog();
            }
        });
        contentPanel.add(viewCarButton);
    }
    private void showCarDetailsDialog() {
        // Simulated list of cars, replace with your actual list
        String[] cars = {"Toyota Camry","Audi A4","Mercedes-Benz C-
Class","Tesla Model S","Lexus RX","Subaru Outback", "Honda Civic", "Ford
Mustang", "Chevrolet Malibu", "BMW X5"};
        JComboBox<String> carList = new JComboBox<>(cars);
        carList.setEditable(false);
        JPanel panel = new JPanel(new GridLayout(2, 2));
        panel.add(new JLabel("Select a car:"));
        panel.add(carList);
        int result = JOptionPane.showConfirmDialog(null, panel, "View Car
Details", JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
```

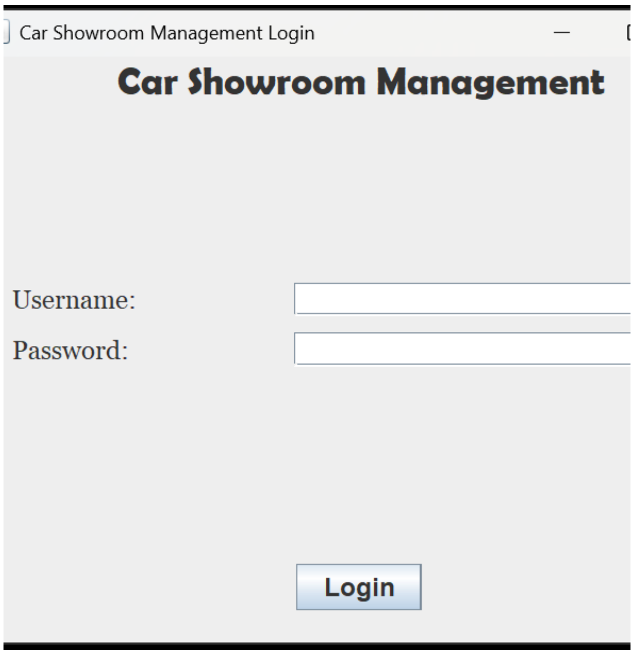
```
        if (result == JOptionPane.OK_OPTION) {
            String selectedCar = (String) carList.getSelectedItem();
            if (selectedCar != null) {
                // Replace this with actual car details retrieval based on the 
selected car
                String carDetails = getCarDetails(selectedCar);
                JOptionPane.showMessageDialog(null, carDetails, "Car Details", 
JOptionPane.PLAIN_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "No car selected.", 
"Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    // Method to get car details (dummy implementation)
    private String getCarDetails(String carName) {
        String details = "No details available.";
        try (Connection conn = 
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root", 
"s1@s2#s3*");
             PreparedStatement stmt = conn.prepareStatement("SELECT * FROM car 
WHERE carname = ?")) {
            stmt.setString(1, carName);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                int year = rs.getInt("caryear");
                String color = rs.getString("colour"); // Corrected column 
name
                double price = rs.getDouble("price");
                details = carName + " details:\nYear: " + year + "\nColor: " + 
color + "\nPrice: $" + price;
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        return details;
    }
    private void addUserSearchCarButton() {
        JButton searchCarButton = new JButton("Search Cars");
        searchCarButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Create and show a search dialog
                showSearchDialog();
            }
        });
        contentPanel.add(searchCarButton);
    }
    private void showSearchDialog() {
        // Create dialog components
        JLabel makeLabel = new JLabel("Name:");
        JTextField makeField = new JTextField();
        JLabel modelLabel = new JLabel("Model:");
        JTextField modelField = new JTextField();
        JLabel priceLabel = new JLabel("Price Range:");
        JTextField minPriceField = new JTextField();
        JTextField maxPriceField = new JTextField();
        // Create panel and add components
        JPanel searchPanel = new JPanel(new GridLayout(4, 2));
        searchPanel.add(makeLabel);
        searchPanel.add(makeField);
        searchPanel.add(modelLabel);
        searchPanel.add(modelField);
        searchPanel.add(priceLabel);
        searchPanel.add(minPriceField);
        searchPanel.add(maxPriceField);
        // Show dialog
        int result = JOptionPane.showConfirmDialog(null, searchPanel, "Search
Cars", JOptionPane.OK_CANCEL_OPTION);
        if (result == JOptionPane.OK_OPTION) {
            // Get search criteria
            String make = makeField.getText();
            String model = modelField.getText();
            double minPrice = Double.parseDouble(minPriceField.getText());
            double maxPrice = Double.parseDouble(maxPriceField.getText());
            // Perform search based on criteria
            performSearch(make, model, minPrice, maxPrice);
        }
    }
    private void performSearch(String make, String model, double minPrice,
double maxPrice) {
        // Perform search based on criteria and display results
        String message = "Search Results:\n" +
                         "Name: " + make + "\n" +
                         "Model: " + model + "\n" +
                         "Price Range: $" + minPrice + " - $" + maxPrice;
        JOptionPane.showMessageDialog(null, message);
    }
    private void addUserBookCarButton() {
        JButton bookCarButton = new JButton("Book Car");
        bookCarButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Show available cars to the user
                String[] availableCars = {"Toyota Camry","Audi A4","Mercedes-
Benz C-Class","Tesla Model S","Lexus RX","Subaru Outback", "Honda Civic",
"Ford Mustang", "Chevrolet Malibu", "BMW X5"};
```

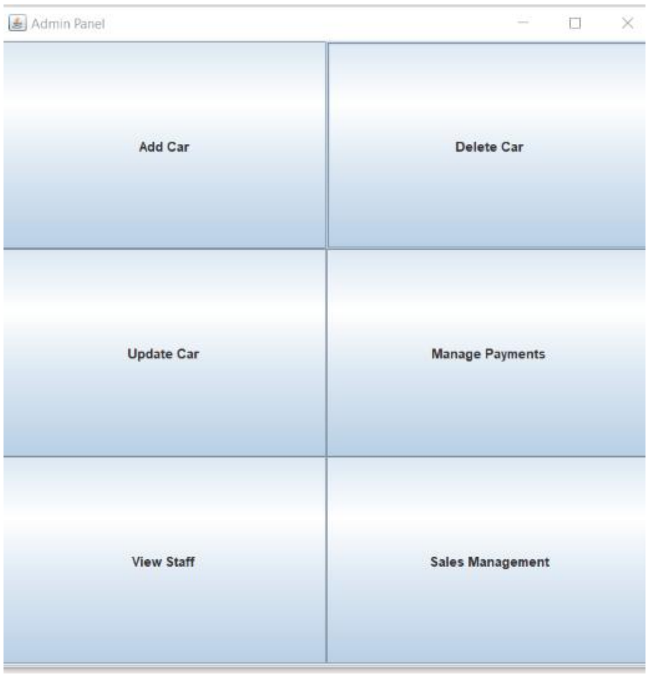
```
                String selectedCar = (String)
JOptionPane.showInputDialog(null, "Select a car to book:", "Book Car",
                        JOptionPane.PLAIN_MESSAGE, null, availableCars,
availableCars[0]);
                if (selectedCar != null) {
                    // Generate a booking ID
                    int bookingID = generateBookingID();
                    // Update the database or internal data structure to mark
the car as booked
                    if (bookCar(selectedCar, bookingID)) {
                        JOptionPane.showMessageDialog(null, "You have booked
the car: " + selectedCar + " with Booking ID: " + bookingID);
                    } else {
                        JOptionPane.showMessageDialog(null, "Failed to book
the car. Please try again later.", "Error", JOptionPane.ERROR_MESSAGE);
                    }
                } else {
                    JOptionPane.showMessageDialog(null, "Booking canceled.");
                }
            }
        });
        contentPanel.add(bookCarButton);
    }
    // Method to book the selected car
    private boolean bookCar(String carName, int bookingID) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            try (Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root",
"s1@s2#s3*")) {
                // Query to update the car table to mark the car as booked
                String bookCarQuery = "UPDATE car SET bookid = ? WHERE carname
= ?";
                PreparedStatement bookCarStatement =
connection.prepareStatement(bookCarQuery);
                bookCarStatement.setInt(1, bookingID);
                bookCarStatement.setString(2, carName);
                int rowsUpdated = bookCarStatement.executeUpdate();
                return rowsUpdated > 0;
            }
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
    // Method to generate a unique booking ID
    private int generateBookingID() {
        int bookingID = 0;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            try (Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root",
"s1@s2#s3*")) {
                // Query to get the maximum booking ID from the database
                String maxBookingIDQuery = "SELECT MAX(bookid) FROM car";
                PreparedStatement maxBookingIDStatement =
connection.prepareStatement(maxBookingIDQuery);
                ResultSet resultSet = maxBookingIDStatement.executeQuery();
                if (resultSet.next()) {
                    // Get the maximum booking ID and increment it to generate
a new ID
                    bookingID = resultSet.getInt(1) + 1;
                } else {
                    // No existing bookings, start from 1
                    bookingID = 1;
                }
            }
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
        return bookingID;
    }
    private void addUserCancelBookCarButton() {
        JButton cancelBookCarButton = new JButton("Cancel Booked Car");
        cancelBookCarButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Fetch booked cars from the database
                ArrayList<Integer> bookedCarIDs = fetchBookedCarIDs();
                if (!bookedCarIDs.isEmpty()) {
                    // Convert ArrayList to array for display in combo box
                    Integer[] bookedCarIDsArray = new
Integer[bookedCarIDs.size()];
                    bookedCarIDs.toArray(bookedCarIDsArray);
                    // Display a combo box to select the booked car ID
                    Integer selectedBookingID = (Integer)
JOptionPane.showInputDialog(null, "Select a booked car to cancel:", "Cancel
Booked Car",
                            JOptionPane.PLAIN_MESSAGE, null,
bookedCarIDsArray, bookedCarIDsArray[0]);
                    if (selectedBookingID != null) {
                        // Cancel the selected booked car
                        if (cancelBookedCar(selectedBookingID)) {
                            JOptionPane.showMessageDialog(null, "Booking with
ID: " + selectedBookingID + " has been canceled successfully.");
                        } else {
                            JOptionPane.showMessageDialog(null, "Failed to 
cancel the booking. Please try again later.", "Error", 
JOptionPane.ERROR_MESSAGE);
                        }
                    } else {
                        JOptionPane.showMessageDialog(null, "Cancellation 
canceled.");
                    }
                } else {
                    JOptionPane.showMessageDialog(null, "No booked cars 
found.");
                }
            }
        });
        contentPanel.add(cancelBookCarButton);
    }
    // Method to fetch all booked car IDs from the database
    private ArrayList<Integer> fetchBookedCarIDs() {
        ArrayList<Integer> bookedCarIDs = new ArrayList<>();
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            try (Connection connection = 
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root", 
"s1@s2#s3*")) {
                // Query to fetch all booked car IDs
                String fetchBookedCarIDsQuery = "SELECT bookid FROM car WHERE 
bookid IS NOT NULL";
                PreparedStatement fetchBookedCarIDsStatement = 
connection.prepareStatement(fetchBookedCarIDsQuery);
                ResultSet resultSet = 
fetchBookedCarIDsStatement.executeQuery();
                while (resultSet.next()) {
                    // Add each booked car ID to the list
                    bookedCarIDs.add(resultSet.getInt("bookid"));
                }
            }
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
        return bookedCarIDs;
    }
    // Method to cancel the booked car with the given booking ID
    private boolean cancelBookedCar(int bookingID) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            try (Connection connection = 
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root", 
"s1@s2#s3*")) {
```

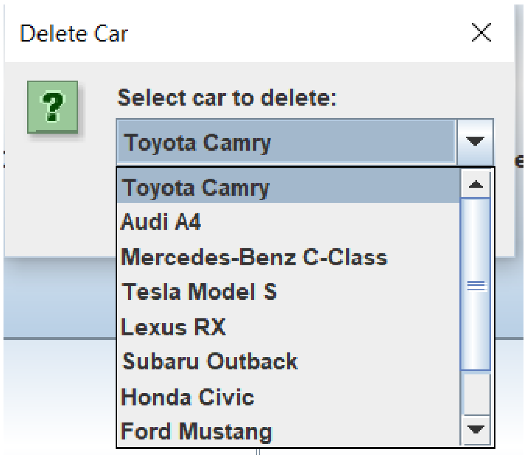
```
                // Query to update the car table to mark the booked car as
available (cancel booking)
                String cancelBookedCarQuery = "UPDATE car SET bookid = NULL
WHERE bookid = ?";
                PreparedStatement cancelBookedCarStatement =
connection.prepareStatement(cancelBookedCarQuery);
                cancelBookedCarStatement.setInt(1, bookingID);
                int rowsUpdated = cancelBookedCarStatement.executeUpdate();
                return rowsUpdated > 0;
            }
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
    private void addUserInteractWithStaffButton() {
        JButton interactWithStaffButton = new JButton("Interact with Staff");
        interactWithStaffButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Implement functionality for users to interact with staff
                interactWithStaff();
            }
        });
        contentPanel.add(interactWithStaffButton);
    }
    private void interactWithStaff() {
        // Placeholder for interacting with staff
        JOptionPane.showMessageDialog(null, "Interacting with staff...");
        // Placeholder for staff interaction response
        String staffResponse = JOptionPane.showInputDialog(null, "Staff: How
may I assist you?", "Staff Interaction", JOptionPane.QUESTION_MESSAGE);
        // Placeholder for user response to staff
        if (staffResponse != null) {
            JOptionPane.showMessageDialog(null, "User response: " +
staffResponse, "User Interaction", JOptionPane.INFORMATION_MESSAGE);
        }
    }
    private void addUserFeedbackButton() {
        JButton feedbackButton = new JButton("Give Feedback");
        feedbackButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Implement functionality for users to give feedback
                giveFeedback();
            }
        });
        contentPanel.add(feedbackButton);
    }
    private void giveFeedback() {
        // Get user's feedback
        String feedback = JOptionPane.showInputDialog(null, "Please provide
your feedback:", "Feedback", JOptionPane.PLAIN_MESSAGE);
        // You can handle the feedback (e.g., store it in a database)
        if (feedback != null) {
            JOptionPane.showMessageDialog(null, "Thank you for your
feedback!", "Feedback Received", JOptionPane.INFORMATION_MESSAGE);
        }
    }
    private void viewStaffDetails() {
        String staffIdInput = JOptionPane.showInputDialog(frame, "Enter Staff
ID:");
        if (staffIdInput == null || staffIdInput.trim().isEmpty()) {
            JOptionPane.showMessageDialog(frame, "No Staff ID entered.",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        int staffId;
        try {
            staffId = Integer.parseInt(staffIdInput);
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(frame, "Invalid Staff ID.", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }
        try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db", "root",
"s1@s2#s3*");
             PreparedStatement stmt = conn.prepareStatement("SELECT * FROM
Staff WHERE staffid = ?")) {
            stmt.setInt(1, staffId);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                int id = rs.getInt("staffid");
                String name = rs.getString("name");
                String position = rs.getString("position");
                int age = rs.getInt("age");
                String contact = rs.getString("contact");
                String address = rs.getString("address");
                String details = "Staff ID: " + id + "\n" +
                        "Name: " + name + "\n" +
                        "Position: " + position + "\n" +
                        "Age: " + age + "\n" +
                        "Contact: " + contact + "\n" +
                        "Address: " + address;
                JTextArea staffDetailsArea = new JTextArea(details);
                staffDetailsArea.setEditable(false);
                staffDetailsArea.setLineWrap(true);
                staffDetailsArea.setWrapStyleWord(true);
                JOptionPane.showMessageDialog(frame, new
JScrollPane(staffDetailsArea), "Staff Details", JOptionPane.PLAIN_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(frame, "No staff found with the
provided ID.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(frame, "An error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    private void showInteractWithUser() {
        // Placeholder for interacting with staff
        JOptionPane.showMessageDialog(null, "Interacting with user...");
        // Placeholder for staff interaction response
        String staffResponse = JOptionPane.showInputDialog(null, "User:Can we
do online payment?", "User Interaction", JOptionPane.QUESTION_MESSAGE);
        // Placeholder for user response to staff
        if (staffResponse != null) {
            JOptionPane.showMessageDialog(null, "Staff response: " +
staffResponse, "User Interaction", JOptionPane.INFORMATION_MESSAGE);
        }
    }
}

public class Mainclass {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new CarShowroomLoginGUI();
            }
        });
    }
}
```

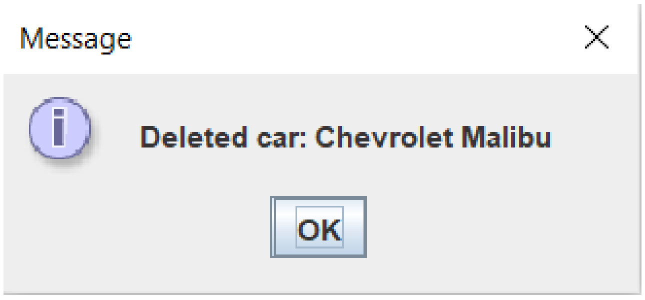
OUTPUT

- Admin Module



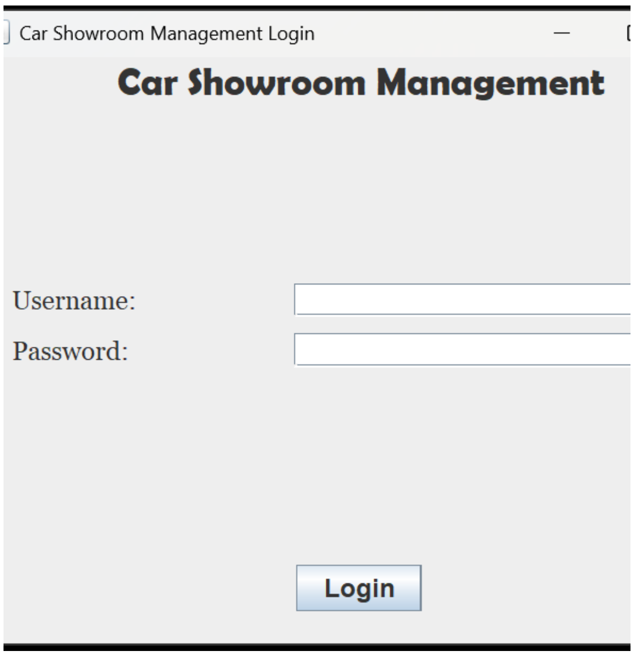






Deleted car: Chevrolet Malibu

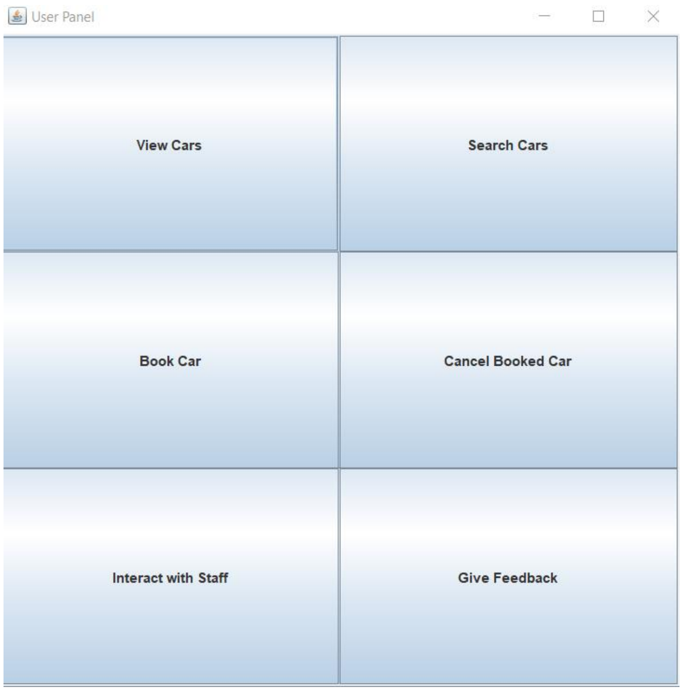
- User Module

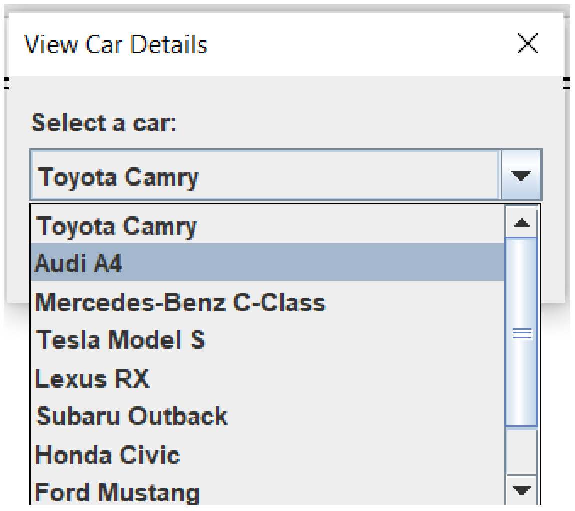

Car Showroom Management Login

Car Showroom Management

Username:

Password:

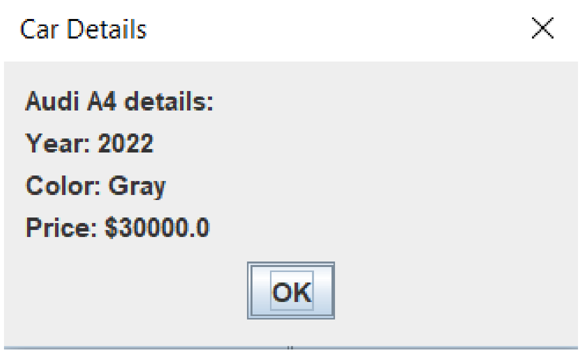



View Car Details

Select a car:

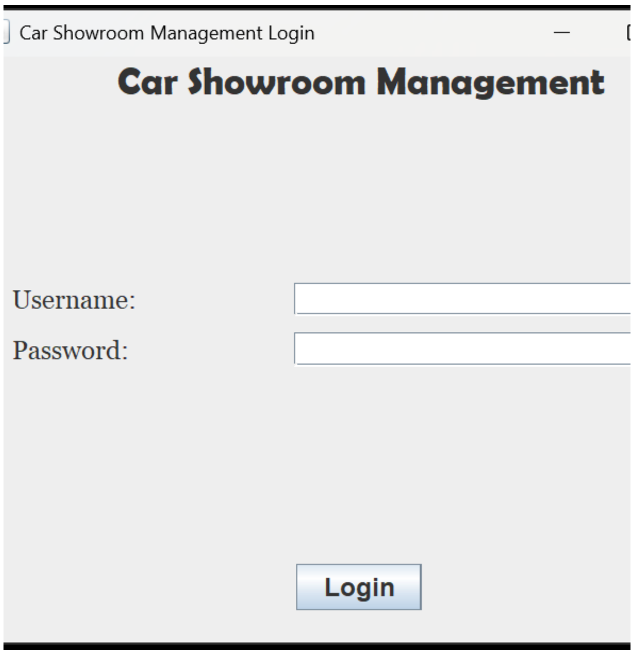
Toyota Camry

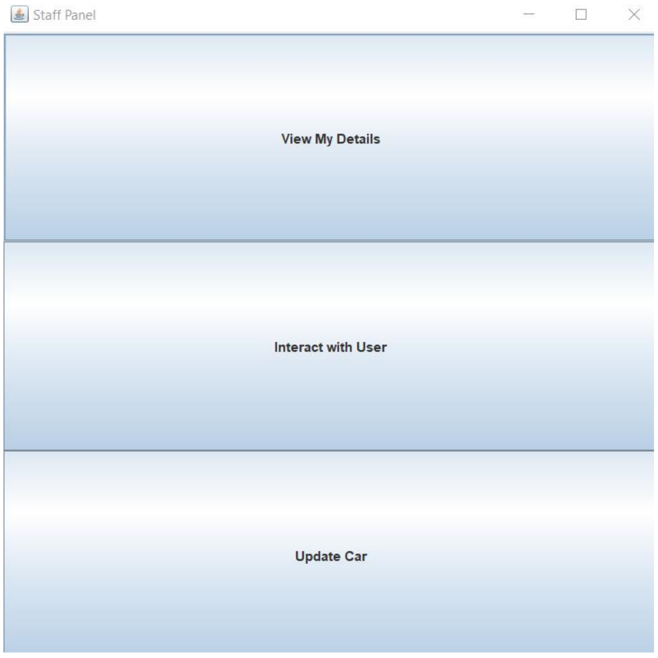
Toyota Camry
Audi A4
Mercedes-Benz C-Class
Tesla Model S
Lexus RX
Subaru Outback
Honda Civic
Ford Mustang

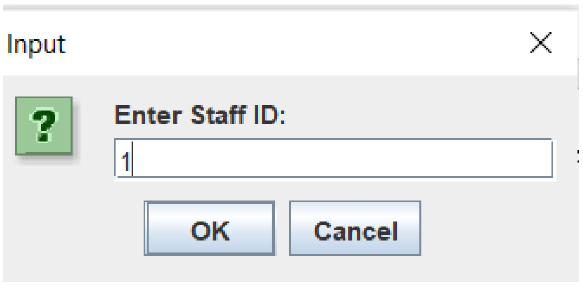

Car Details

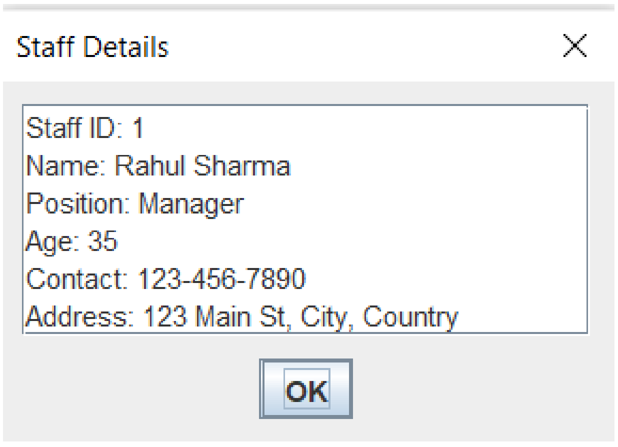
Audi A4 details:
Year: 2022
Color: Gray
Price: $30000.0

- Staff Module









Staff ID: 1
Name: Rahul Sharma
Position: Manager
Age: 35
Contact: 123-456-7890
Address: 123 Main St, City, Country

BIBLIOGRAPHY

1. Automobile Showroom Management: An Analytical Approach" by K. Krishnamoorthy
2. Modern Automotive Showroom Management and Marketing" by Jeffrey F. Loeb
3. Effective Car Showroom Management: A Practical Guide" by Simon L. Dolan
4. Automotive Retailing: Today and Tomorrow" edited by Gianmaria Martini and Lino Cinquini
5. Car Dealership Management: How to Buy and Run a Car Dealership" by Michael Reynolds
6. The Ultimate Sales Machine: Turbocharge Your Business with Relentless Focus on 12 Key Strategies" by Chet Holmes
7. Marketing Management" by Philip Kotler and Kevin Lane Keller
8. Operations Management" by Nigel Slack, Alistair Brandon-Jones, and Robert Johnston
9. Customer Relationship Management: Concepts and Technologies" by Francis Buttle
10. Inventory Management: Principles, Concepts and Techniques" by Adam Kolaski