Faculty of Engineering Department of Informatics Engineering- Software and Information Systems Engineering

الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Integrated Car Center Management System

Prepared by

Badr Askar - Mashal Alhaj - Mohamad Moneer Kurdi

Supervised By

Eng. Anas Abdulaziz

الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

كلية الهندسة
قسم الهندسة المعلوماتية
هندسة البرمجيات ونظم المعلومات

نظام إدارة المركز المتكامل للسيارات

اعداد

بدر عسكر - مشعل الحاج - محمد منير كردي

اشراف

الدكتور انس عبدالعزيز

المحتويات

Abstract

This project focuses on the design and development of an integrated electronic system for car center management, aiming to organize and streamline the processes of buying and selling new and used cars, car rental services, original spare parts sales, and maintenance and repair services. Traditional approaches in this field often suffer from poor organization, slow procedures, and increased risks of fraud, which negatively impact customer experience. The proposed system seeks to enhance operational efficiency and reliability through a secure and user-friendly digital platform that allows users to access all services from a single place. The system adopts structured user management, customized dashboards, and efficient data organization, enabling car center owners to monitor operations and make informed decisions. This solution helps save time and effort, reduce risks, and improve customer satisfaction by providing a smooth and transparent user experience.

الملخص

يركز هذا المشروع على تصميم وتطوير نظام إلكتروني متكامل لإدارة مراكز السيارات، يهدف إلى تسهيل وتنظيم عمليات بيع وشراء السيارات الجديدة والمستعملة، وتأجير السيارات، وبيع قطع الغيار الأصلية، بالإضافة إلى إدارة خدمات الصيانة والتصليح. تعاني الطرق التقليدية في هذا المجال من ضعف التنظيم، بطء الإجراءات، وزيادة مخاطر الاحتيال، مما يؤثر سلباً على تجربة العملاء. يسعى هذا المشروع إلى تحسين كفاءة العمليات وتعزيز موثوقية التعامل من خلال منصة رقمية آمنة وسهلة الاستخدام، تتيح للمستخدمين الوصول إلى جميع الخدمات من مكان واحد. يعتمد النظام على إدارة المستخدمين، ولوحات تحكم مخصصة، وتنظيم البيانات بشكل فعال، مما يساعد أصحاب المراكز على متابعة العمليات واتخاذ القرارات المناسبة. يساهم هذا الحل في توفير الوقت والجهد، وتقليل المخاطر، ورفع مستوى رضا العملاء من خلال تجربة استخدام سلسة وشفافة

Chapter 1 Introduction

1. Introduction :

This chapter introduces the core elements of the Car Marketplace System project, laying the foundation for understanding the problem it addresses, the project goals, the proposed system, the organization of the report, and a brief summary of key points.

2. Problem Statement :

The project addresses the current limitations in traditional car marketplaces, where existing models often struggle with scattered services, limited access to reliable information, and inefficient management of car sales, rentals, spare parts, and repair services. These limitations create difficulties for customers and reduce overall satisfaction, highlighting the need for a more integrated solution. This project proposes a unified platform that simplifies buying, selling, and renting cars, streamlines spare parts purchasing, and enables efficient booking of repair services to enhance customer convenience and business operations.

3. Project Objective :

The main objective of this project is to develop an integrated and efficient car marketplace system that simplifies the processes of buying, selling, renting, and maintaining cars. Key project goals include improving customer convenience, enabling smooth management of vehicles and spare parts, and ensuring that repair services can be booked and tracked easily. The project scope focuses on creating a platform that incorporates user management, car listings, rental management, spare parts management, and workshop booking. The methodology includes an analytical approach to defining system requirements and workflows, supported by structured databases and tools that enable seamless operations and role-based access control.

4. Proposed System :

The AI-Powered Car Marketplace System is designed to function as a high-level, automated solution for vehicle trading and services management. By leveraging intelligent technologies, it offers advanced capabilities, including smart search and filtering, which allow users to quickly find suitable cars, rentals, or spare parts. Additionally, it provides role-based management tools for users and administrators, enabling efficient control of listings, transactions, and workshop services across the platform.

5. Report Organization :

The report is structured as follows:

Chapter 1: Introduction

Chapter 2: Basic Concepts and Reference Studies in the Automotive Field

Chapter 3: Project Management

Chapter 4: System Analysis

Chapter 5: System Design

Chapter 6: Practical Implementation of the Platform

Chapter 7: Report Overview

6. Summary :

This chapter has outlined the need for a smart car marketplace platform, explained the objectives of the project, introduced the proposed solution at a high level, and provided an overview of the report's structure. This introduction sets the stage for a deeper exploration of each aspect of the project in the following chapters

Chapter 2 Fundamental Concepts and Literature Review

1. Introduction :

This chapter provides a foundation for understanding the Car Dealership and Service System by exploring its core concepts industry. It highlights the integration of car sales, rentals, spare parts management, and repair services within a unified platform. The aim is to present an overview of modern automotive business solutions and digital management approaches, along with insights from existing systems to guide and support the development of the proposed project.

2. Fundamental Concepts :

This section explains the core concepts, terms, and theoretical principles relevant to the project, including:

• **Car Sales Management**: An overview of the processes involved in displaying ,managing, and selling vehicles through the system. Concepts such as inventory tracking, pricing models, and promotional offers are included to explain how the platform supports efficient car sales operations.

• **Car Rental System**: Explanation of rental management and its importance in providing short-term or long-term mobility solutions. Key aspects such as booking, availability tracking, rental duration, and return policies are covered to highlight the role of digital automation in managing rentals.

• **Spare Parts Management**: Discussion on managing and selling spare parts ,including categorization, search functions by brand and model, and inventory updates. This concept emphasizes how digital systems enhance accessibility and availability of vehicle components for both customers and the workshop.

• **Workshop and Repair Services:** An introduction to repair service management, outlining the process of booking maintenance appointments, tracking service history, and ensuring quality repairs. The concept highlights the integration of workshop operations with customer management for seamless service delivery.

• **Customer-to-Business Vehicle Transactions**: Explanation of the process where customers can sell their vehicles to the dealership. This includes

submission of vehicle details, valuation methods, and approval processes, showcasing how the system facilitates two-way transactions.

• **Role-Based Access Control (RBAC):** An overview of RBAC and its significance in ensuring data security within the system. Different roles such as Admin, Customer , and Workshop Staff are defined to ensure controlled access, secure workflows, and proper management of business operations.

3. Literature Review :

The purpose of this Literature Review is to analyze and evaluate existing systems and approaches related to the development of integrated car dealership and service platforms.

This study aims to provide a comprehensive overview of current practices in car sales,rental management, spare parts distribution, and workshop services, focusing on how digital solutions and automation are utilized to enhance efficiency and customer experience. The analysis includes comparisons between several systems and their key functionalities to identify strengths, limitations, and opportunities for the proposed project.

- ### Motorway (AI-powered car buying and selling platform)

Advantages:

1. Smart and accurate pricing using AI technologies.
2. Digital platform that saves time and effort for customers.
3. Simple and user-friendly interface.
4. Access to a wide network of buyers and sellers.

Disadvantages:

1. Heavy reliance on technology requires high infrastructure investment.
2. Non-tech-savvy customers may find it difficult to use.
3. Limited applicability in markets with low digital adoption.
4. Data security risks associated with cloud services.

Main Features:

• AI-based car pricing algorithms.

• Automated sales and purchase processes.

• Scalable cloud-based platform.

• Personalized user experience.

- ## Online Auto Parts Store (Ukraine case study)

Advantages:

1. Enhanced user experience that boosted sales.
2. Improved SEO attracted higher website traffic.
3. Clear product categorization for easy search.
4. Significant revenue growth in a short period.

Disadvantages:

1. Heavy dependence on digital marketing.
2. Inventory management challenges with high order volumes.
3. Difficulties in shipping and after-sales support.
4. Strong competition from other online stores.

Main Features:

• Organized database of spare parts by type, brand, and model.

• SEO optimization for customer acquisition.

• Fast and user-friendly interface.

• Sales and performance tracking reports.

- ## Catch-e (Dealer-based leasing and fleet management software)

Advantages:

1. Comprehensive system for leasing and maintenance management.
2. Accurate reporting on fleet performance.
3. Supports large dealerships and automotive enterprises.
4. Facilitates internal maintenance and service tracking.

Disadvantages:

1. Relatively complex system requiring staff training.
2. High operational and maintenance costs.
3. Less suitable for small businesses.
4. Integration challenges with other systems.

Main Features:

• Fleet management (leasing, monitoring, maintenance).

• Customer booking system.

• Detailed performance analytics and reporting.

• Digital tracking of car status and workshop operations.

System Feature	*Motorway*	*Online Auto Parts Store*	*Catch-e*	*Our system*
Car Sales Management	✓	✗	✗	✓
Car Rental Management	✓	✗	✓	✓
Spare Parts Catalog	✗	✓	✗	✓
Workshop / Maintenance Management	✗	✗	✓	✓
Customer Request Tracking	✓	✓	✓	✓
AI-Powered Pricing / Recommendations	✓	✗	✗	✓
Reporting & Analytics	✓	✓	✓	✓
User-Friendly Interface	✓	✓	✓	✓
Inventory Management	✓	✓	✓	✓
Multi-Language Support	✗	✗	✗	✓
(✓ means feature is present, ✗ means feature is absent)				

Chapter 3 Project Management

1. Introduction :

In this chapter, we will dive into the management phase of the Car Showroom Project, which is a vital component in ensuring the project's success and long-term sustainability. We will examine the project charter, project plan, Statement of Work (SOW) document, stakeholder analysis, and risk management strategies to effectively manage and control all aspects of the business. This includes car sales, car rentals, spare parts sales, and repair workshop services, ensuring smooth operations from initiation to completion.

2. Project charter :

A project charter is a formal document that serves as an official authorization for the start of a project. It acts as a reference point throughout the project, providing a clear understanding of the project's purpose and establishing a foundation for decision making and project governance.

- **Project Title**: cars website
- **Project Start Date**: September 9, 2025
- **Projected Finish Date:**
- **Project Manager:** Eng.Anas Abdulaziz
- **Project background:** With the increasing reliance on digital services, it has become necessary to create an electronic platform for the car store that makes it easier for customers to buy cars, rent them, book maintenance services, and order spare parts online.
- **Project Objectives :**

  ❖ Development of a comprehensive website for store services (sale-buy - rent - maintenance - spare parts) .

  ❖ Enable customers to book and buy electronically easily .

  ❖ Manage inventory and ads in a structured way via a control pane .

  ❖ Enhance the customer experience through smooth and fast service.

  ❖ Enhancing customer confidence in providing transparent information and after-sales service (follow-up of maintenance and spare parts).

  ❖ Easily access to a large customer base at the local level and connect them to the store

  ❖ Facilitate the customer experience through an easy user interface and advanced search by price, model, type and service

❖ Enable customers to pay online in a safe and fast way.

- **Approach:**

❖ Conduct a needs assessment and gather user requirements.

❖ Develop a system architecture covering listings, reservations, inventory management, and payment solutions.

❖ Utilize internal development teams to build a modern front-end and an admin dashboard.

• Create a database for cars, users, and inventory management.

❖ Perform functional testing (search, registration, purchasing, booking, and payment).

• Conduct performance and security testing to ensure speed and data protection.

❖ Integrate with APIs and payment gateways.

❖ Develop an interactive dashboard for Admins.

▪ **Roles and Responsibilities:**

Name	*Role*	*Responsibility*
Eng.Anas Abdulaziz	Project Manager	Manage day-to-day project execution, coordinate with team members, track progress, manage risks.
Mohamed Moneer Kurdi	SE& Backend(Django)& frontend(react)	Design and implement user interfaces, ensure a seamless user experience Collaborate on backend development, handle database integrations, manage API development.
Bader Askar	SE& Backend(Django)& frontend(react)	Design and implement user interfaces, ensure a seamless user experience Collaborate on backend development, handle database integrations, manage API development.
Mashal Alhaj	SE& frontend(react)& Backend(Django)	Design and implement user interfaces, ensure a seamless user experience Collaborate on backend development, handle database integrations, manage API development.

3. The SOW document :

Statement of Work is a comprehensive document that defines the scope of work for a project. It outlines the specific tasks, deliverables, timeline, and responsibilities. The SOW document provides a clear understanding of what needs to be accomplished, the project's objectives, and the criteria for success.

3.1 Project Description and Objectives:

The project aims to develop a Car Showroom Management System that includes car sales, rentals, purchasing cars from customers, spare parts sales, and repair workshop services. The system will facilitate the processes of selling, renting, and maintenance through an integrated platform, helping to enhance customer experience, improve operational efficiency, and ensure smooth management of all aspects of the project.

3.2 Project Goals:

• Develop a software system that simplifies the management of car sales, rentals, spare parts, and repair services.

• Enable easy addition, deletion, and modification of cars and services through an admin dashboard.

• Provide a user-friendly interface for customers to browse, buy, rent, or sell cars seamlessly.

• Integrate reporting and dashboards to track sales performance, rental activities, workshop operations, and customer interactions.

Deliverables:

• Project plan.

• SRS (Software Requirements Specification) documentation.

• Final project report.

• Fully functional system for managing car sales, rentals, spare parts, and repair services.

• Backend and frontend components – the complete Car Showroom Management System with admin dashboard and customer interface.

3.3 Project Requirements:

Technology and Tools:

• **Programming Languages:** Python, JavaScript, HTML, CSS.

• **Frameworks:** Django (backend), React (frontend).

• **Database:** MySQL.

3.4 Assumptions:

• Regular availability of project team members and stakeholders for collaboration and feedback.

• Continuous guidance and supervision from the project manager to ensure smooth progress.

• Availability of necessary hardware and software resources for the development and deployment of the Car Showroom Management System.

• Customer willingness to adopt and use the system for car sales, rentals, spare parts purchases, and repair services

3.5 Project Resources:

Human Resources:

•: Project Manager

• Eng.Anas-Abdulaziz: Project Manager

• Mohamed Moneer Krdi:(Backend – frontend)

•Bader Asker: SE-Developer (Backend – frontend)

• Mashal Alhaj: SE-Developer (backend – frontend)

3.6 Schedule:

• **Project Start Date:** Septemper 9, 2025

• **First Seminar:**

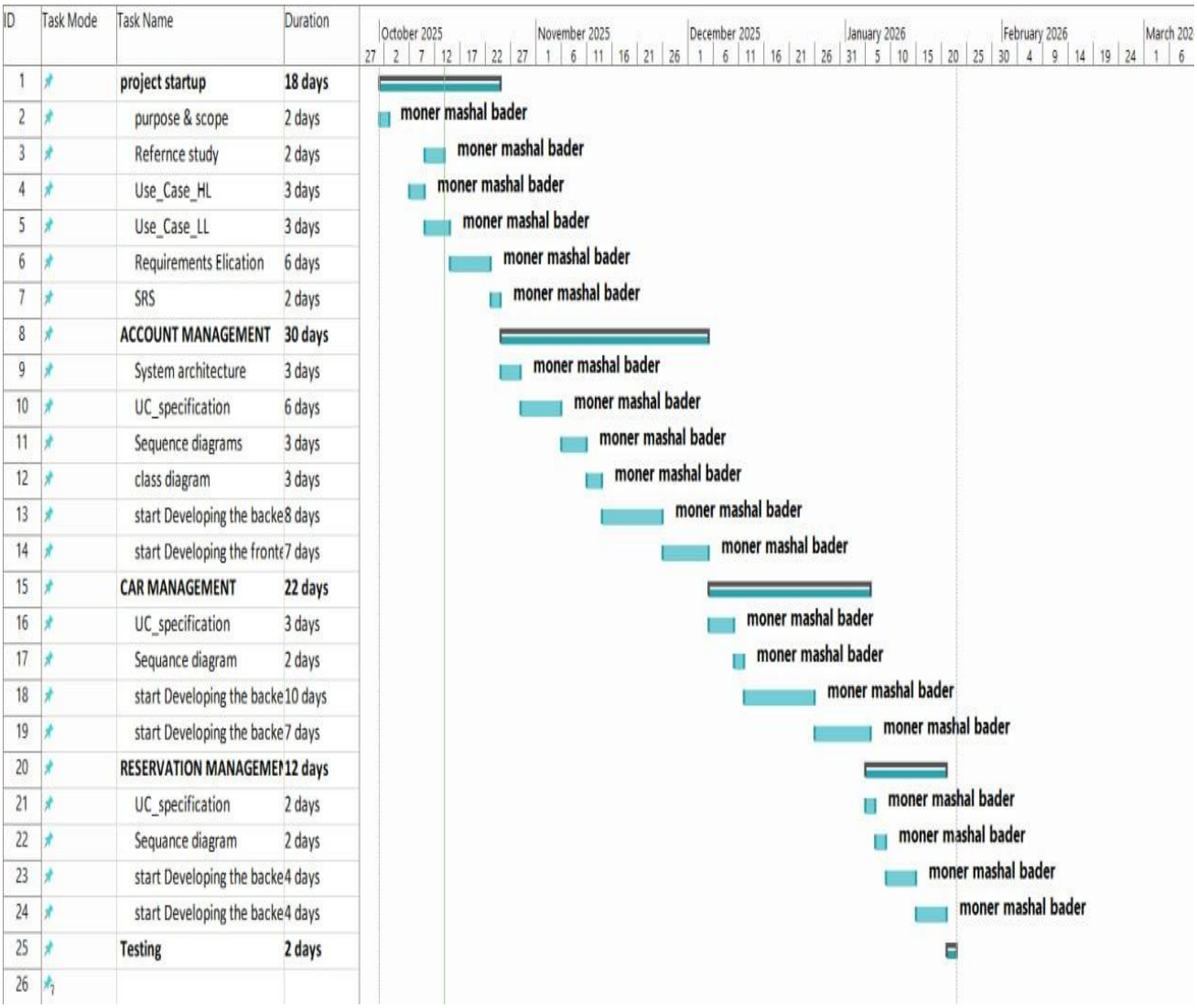
• **Second Seminar:**

• **Final Seminar:**

• **Project Finish Date:**

4. Project plan _ Gantt Chart:

A document outlining tasks, deadlines, and resources needed to achieve project objectives, serving as a roadmap for successful project execution.



ID	Task Mode	Task Name	Duration
1		**project startup**	**18 days**
2		purpose & scope	2 days
3		Refernce study	2 days
4		Use_Case_HL	3 days
5		Use_Case_LL	3 days
6		Requirements Elication	6 days
7		SRS	2 days
8		**ACCOUNT MANAGEMENT**	**30 days**
9		System architecture	3 days
10		UC_specification	6 days
11		Sequence diagrams	3 days
12		class diagram	3 days
13		start Developing the backe	8 days
14		start Developing the fronte	7 days
15		**CAR MANAGEMENT**	**22 days**
16		UC_specification	3 days
17		Sequance diagram	2 days
18		start Developing the backe	10 days
19		start Developing the backe	7 days
20		**RESERVATION MANAGEMEI**	**12 days**
21		UC_specification	2 days
22		Sequance diagram	2 days
23		start Developing the backe	4 days
24		start Developing the backe	4 days
25		**Testing**	**2 days**
26			

5. Risk management :

Risk Title	Risk Description	Raised Date	Tracking Frequency	state	Impact	Mitigation Plan
Team of only three students	If one of the students stops working for some reason, the whole project progress will be impacted.	16/9/2025	Weekly	Active	High	The one who finishes early will assist the other with their tasks.
Learning new technologies (frontend)	The framework is new to us in the frontend	9/9/2025	Daily	Under Mitigation	High	Allocate dedicated learning sessions and use online resources to build understanding.
Learning new technologies (Backend)	The framework is new to us in the backend as well as the programming language	9/9/2025	Daily	Under Mitigation	High	Allocate dedicated learning sessions and use online resources to build understanding.
Fuzzy project scope	Misunderstanding the project scope at the beginning of the work can lead to major mistakes in implementation.	10/11/2025	Daily	Closed	Medium	Conduct regular team discussions and meetings at the end of each milestone to clarify objectives.
Separation of work	One member is responsible for the frontend and the other for the backend, which can cause		Daily	Active	Medium	Regular integration testing and progress sync-ups to ensure compatibility

	coordination gaps.					between backend and frontend.
Deadline pressure	Tight deadlines may lead to shortcuts in development, causing potential quality issues.	1/1/2026	Weekly	Active	Medium	Break tasks into smaller deliverables and prioritize critical functionality to ensure quality work within deadlines.
Requirement changes	Late changes in requirements from instructors or stakeholders can disrupt the project timeline and scope	1/1/2026	Weekly	Active	High	Allocate buffer time in the schedule and ensure effective communication with stakeholders to manage changes.
Deployment challenges	Lack of experience with deployment might delay the final delivery of the project.		Weekly	Active	High	Research deployment best practices and run a mock deployment to identify potential blockers early.
Sentiment analysis accuracy	Challenges in achieving acceptable sentiment analysis accuracy due to limitations in AI training data or inappropriate model selection.		Weekly	Active	High	Train the sentiment analysis model with a diverse dataset and validate the performance regularly with test cases.

6. Summary :

In conclusion, effective project management plays a crucial role in the successful development of software systems. It provides a structured framework that helps ensure tasks are completed within the defined scope, time, and quality constraints. By applying proper planning, monitoring, and risk management practices, the project team can coordinate efforts efficiently and address challenges proactively. Overall, strong project management contributes significantly to delivering a high-quality software system that fulfills project objectives and meets user requirements.

Chapter 4 System Analysis

1. Introduction :

This chapter focuses on the detailed analysis of the Integrated Car Center Management System (ICCMS). It defines the system requirements, functionality, and overall architecture by analyzing user needs and the operational environment of car sales, rentals, and service management. The chapter covers both functional and non-functional requirements, use cases, and system workflows to provide a comprehensive understanding of how the proposed system supports buying and selling cars, renting vehicles, managing reservations, and administering user accounts. This analysis establishes the foundation for designing and implementing a reliable, efficient, and scalable solution that achieves the project objectives.

2. Software Requirement Specification document :

Revision history:

Date	Reason for change	version
7/10/2025	Add more details about the reports and actors.	0.1
11/10/2025	Add revision history.	0.2
25/12/2025	Updated functional requirements	0.3

1. Introduction

1.1 Purpose :

This document specifies the software requirements and design for the Integrated Car Center Management System (ICCMS), a solution designed to streamline and manage car center operations, including buying and selling vehicles, car rental services, reservation handling, and user administration. The document serves as a reference for stakeholders, developers, and project supervisors to understand the system's functionality, constraints, and overall structure.

1.2 Project Scope :

The Integrated Car Center Management System (ICCMS) is designed to automate and enhance the management of car center operations through a centralized digital platform. The system focuses on improving the efficiency and reliability of processes related to buying and selling cars, vehicle rental services, reservation handling, and user account management. This specification covers the high-level system requirements and identifies the main actors involved in the system, including customers and administrators, to ensure clear understanding of system responsibilities and interactions

1.2.1 High-Level Requirements

- User Management:

  This functionality allows administrators to manage system users by adding, updating, viewing, and deleting user accounts. The system supports role-based access control, automatically assigning roles such as Admin or Customer based on the registration process or administrative actions.

- Car Management (Sale):

  The system provides tools to manage cars available for sale, including adding new listings, updating car details, deleting listings, viewing available cars, and marking cars as sold after successful purchase.

- Car Management (Rental):

  This functionality enables the management of cars available for rent. It includes adding, updating, deleting, and viewing rental car listings, as well as managing availability and rental pricing information.

- **Rental Management:**

  The system supports the complete rental process, including selecting rental dates, calculating rental costs, creating rental records, and tracking ongoing and completed rentals.

- **Reservation Management:**

  The system allows customers to request reservations for rental cars by specifying rental periods. Administrators can review, approve, or reject reservation requests, while the system prevents overlapping reservations for the same vehicle.

- **Payment Information Handling:**

  The system enables users to submit required payment-related data during car purchase or rental processes. This functionality ensures that transactions are recorded and linked to the corresponding sale or rental operation.

- **Account Management:**

  Users can manage their personal accounts, including updating profile information, changing passwords, and viewing their own activity such as rentals and reservations.

- **Authentication and Authorization:**

  The system provides secure registration, login, and logout functionality. Access to system features is restricted based on user roles to ensure data security and proper authorization.

- **Administrative Dashboard:**

  The system provides administrators with tools to monitor system activities, manage users, manage car listings, and oversee rental and reservation operations through a centralized interface.

1.2.2 Actors

- **Admin (Administrator):**

  The Admin is responsible for overseeing and managing the entire Integrated Car Center Management System. This actor has the highest level of access and control over the platform. Admins can manage user accounts, add, update, and delete cars for sale and rent, review and manage reservations, approve or reject booking

requests, and monitor overall system activities to ensure smooth and secure operations.

- Customer (Client):

  Customers are the end users of the system who interact with the platform to browse available cars for sale or rent. They can register and manage their accounts, submit car purchase requests, rent vehicles, request reservations, view the status of their rentals and reservations, and update their personal information. Customers primarily use the system to access car-related services in an efficient and reliable manner.

2. Overall Description

2.1 Product Perspective:

The Integrated Car Center Management System (ICCMS) specified in this document is a new, self-contained software product designed to organize and streamline car center operations through a unified digital platform. The system supports core services such as buying and selling vehicles, car rental management, reservation handling, and user account administration. It is not part of an existing product family, nor is it intended to replace any current system. Instead, ICCMS represents a standalone solution developed to address the limitations of traditional car center management methods by improving efficiency, transparency, and overall service quality.

2.2 Product Features

- User Management:

  Create,update, and delete user profiles.

  Support role-based access control with defined roles such as Admin and Customer, ensuring secure and authorized access to system functionalities.

- Car Management (Sale):

  Manage cars available for sale by adding, updating, deleting, and viewing car listings.Allow customers to browse available cars and complete purchase operations, with automatic status updates for sold vehicles.

- Car Management (Rental):

  Manage cars available for rent, including adding, updating, deleting, and viewing rental car listings.

  Maintain car availability and rental pricing information.

- **Rental and Reservation Management:**

  Enable customers to rent cars by selecting rental periods and submitting required information.

  Allow customers to request reservations, while enabling administrators to approve or reject requests and track reservation status.

- **Payment Data Handling:**

  Allow users to submit payment-related information during car purchase and rental processes, ensuring transactions are recorded and linked to corresponding operations.

- **Account Management:**

  Enable users to manage their personal accounts, including updating profile information, changing passwords, and viewing their own rentals and reservations.

- **Administrative Control and Monitoring:**

  Provide administrators with tools to monitor system activity, manage users, oversee car listings, and control rental and reservation workflows through a centralized platform.

- **Integration with External Systems:**

  Support integration with external systems such as frontend applications, reporting tools, or future payment gateways through RESTful APIs.

2.3 User Classes and Characteristics

2.3.1 Admin

Responsibilities:

The Admin is responsible for managing and supervising the entire Integrated Car Center Management System. This includes managing user accounts, adding, updating, and deleting cars for sale and rent, reviewing rental and reservation requests, approving or rejecting reservations, and monitoring overall system operations to ensure data accuracy and proper workflow execution.

Privileges:
Full access to user management, car management (sale and rental), reservation management, and all administrative configuration features within the system.

2.3.2 Customer (Client)

Responsibilities:
The Customer interacts with the system to browse cars available for sale or rent, purchase cars, request car rentals, submit reservation requests, and track the status of their rentals and reservations. Customers are also responsible for managing their personal account information.

Privileges:
Limited access to browsing car listings, purchasing cars, renting vehicles, requesting reservations, viewing their own rental and reservation history, and managing their own account information.

3. System Features

3.1 Functional Requirements

3.1.1 Account Management

REQ-01:
The system must allow the user to create a new account.
Required data: phone number, email, username, and a strong password according to the password policy.

REQ-02:
The system must provide an introductory interface explaining the platform's purpose and features to new users.

REQ-03:
The system must allow the user to log in using email and password.

REQ-04:
The system must allow the user to log out of the system.

REQ-05:
The system must allow the user to change their password after entering the current password.

REQ-06:
The system must allow the customer to delete their account after providing the password.

REQ-07:
The system must allow the administrator to delete user accounts.

REQ-08:
The system must allow the customer to modify their account information, including phone number, email, and username.

3.1.2 Car Management

REQ-09:
The system must allow the administrator to add a car for sale.
Required data: type, description, price, model, picture, and year.

REQ-10:
The system must allow the administrator to add a car for rent.
Required data: type, description, model, picture, year, and price per day.

REQ-11:
The system must allow the administrator to modify information of cars offered for sale.

REQ-12:
The system must allow the administrator to modify information of cars offered for rent.

REQ-13:
The system must allow the administrator to delete a car.

REQ-14:
The system must allow users to view cars available for sale.

REQ-15:
The system must allow users to view cars available for rent.

REQ-16:
The system must allow the customer to purchase a car.

REQ-17:
The system must allow the customer to rent a car.

3.1.3 Reservation Management

REQ-18:
The system must allow the customer to search for a car.

REQ-19:
The system must allow users to filter cars by type, price, and year of manufacture.

REQ-20:
The system must allow the administrator to update the vehicle status (rented or not rented).

REQ-21:
The system must allow the customer to request a car reservation, provided that the car is available**.**

REQ-22:
The system must allow the administrator to approve a reservation request.

REQ-23:
The system must allow the administrator to reject a reservation request.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

NF-1.1: The system shall respond to user requests within an acceptable time under normal operating conditions.

NF-1.2: The system shall support multiple concurrent users without performance degradation.

3.2.2 Security Requirements

NF-2.1: The system shall protect user data from unauthorized access.

NF-2.2: Passwords shall be stored using secure hashing algorithms.

NF-2.3: Access to system features shall be restricted based on user roles.

3.2.3 Usability Requirements

NF-3.1: The system shall provide an intuitive and user-friendly interface.

NF-3.2: The system shall be usable by users with limited technical experience.

3.2.4 Reliability Requirements

NF-4.1: The system shall operate reliably with minimal failures.

NF-4.2: The system shall maintain data integrity in case of errors or unexpected interruptions.

3.2.5 Availability Requirements

NF-5.1: The system shall be available for use at all times except during scheduled maintenance.

3.2.6 Maintainability Requirements

NF-6.1: The system shall be easy to maintain and update.

NF-6.2: The system shall support future enhancements without major modifications.

3.2.7 Compatibility Requirements

NF-7.1: The system shall be compatible with commonly used web browsers.

NF-7.2: The system shall support different screen sizes.

4. System Requirements

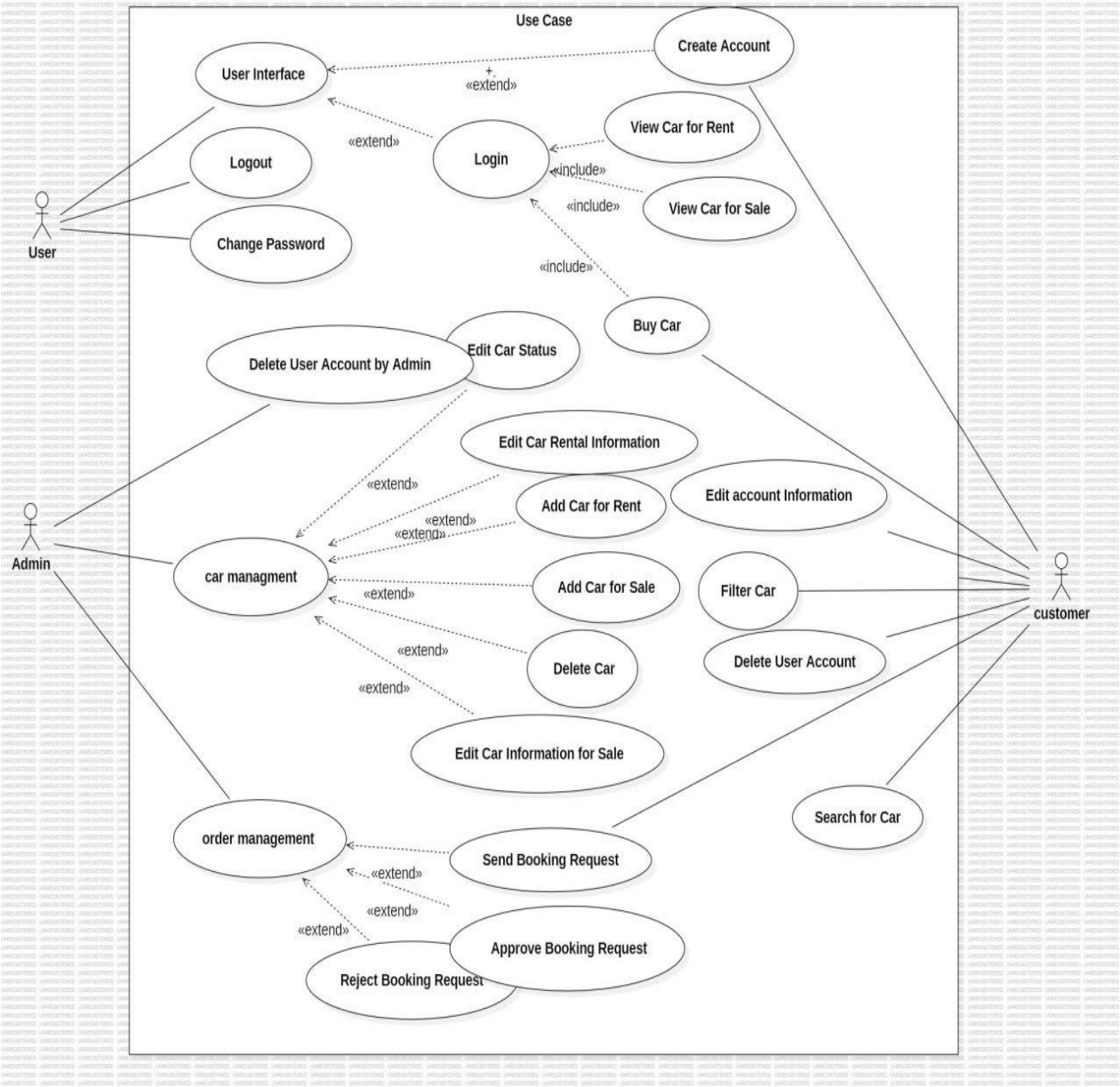
Req-ID	Req Title	Category	Priority
FR-CUS 01	The system must allow the user to create a new account.	Account Management	High
FR-USER 03	The system must allow the user to log in.	Account Management	High
FR-USER 04	The system must allow the user to log out	Account Management	Low
FR-USER 05	The system must allow the user to change their password	Account Management	Low
FR-CUS 06	The system must allow the customer to delete his account.	Account Management	Medium
FR-ADM 07	The system must allow the administrator to delete user accounts.	Account Management	Medium
FR-CUS 08	The system must allow the customer to modify his account information.	Account Management	Low
FR-ADM 09	The system should allow the administrator to add a car for sale.	Car Management	High
FR-ADM 10	The system should allow the manager to add a car for rent.	Car Management	High

FR-ADM 011	The system must allow the administrator to modify the information of the vehicle offered for sale.	Car Management	Low
FR-ADM 012	The system must allow the manager to modify the information of the car offered for rent.	Car Management	Low
FR-ADM 013	The system should allow the administrator to delete a car.	Car Management	Low
FR-USER 014	The system must allow the user to display cars for sale.	Car Management	Low
FR-USER 015	The system must allow the user to view cars offered for rent.	Reservation Management	Low
FR-CUS 016	The system must allow the customer to purchase a car	Car Management	High
FR-CUS 017	The system must allow the customer to rent the car.	Car Management	High
FR-CUS 018	The system should allow the customer to search for the car	Reservation Management	Medium
FR-USER 019	The system should allow the user to filter.	Reservation Management	High
FR-ADM 020	The system should allow the manager to update the vehicle status.	Reservation Management	High

FR-CUS 021	The system should allow the user to request a car reservation.	Reservation Management	High
FR-ADM 022	The system must allow the administrator to approve the reservation request.	Reservation Management	High
FR-ADM 023	The system should allow the administrator to reject the booking request.	Reservation Management	High
NF-1.1	The system shall respond to user requests within an acceptable time under normal operating conditions	Performance	High
NF-2.1	The system shall protect user data from unauthorized access.	Security	High
NF-3.1	The system shall provide an intuitive and user-friendly interface.	Usability	Medium
NF-4.1	The system shall operate reliably with minimal failures.	Reliability	High
NF-5.1	The system shall be available for use at all times except during scheduled maintenance.	Availability	High
NF-6.1	The system shall be easy to maintain and update.	Maintainability	High
NF-7.1	The system shall be compatible with commonly used web browsers.	Compatibility	Medium

5. Requirements modeling

5.1 Basic UML Diagrams



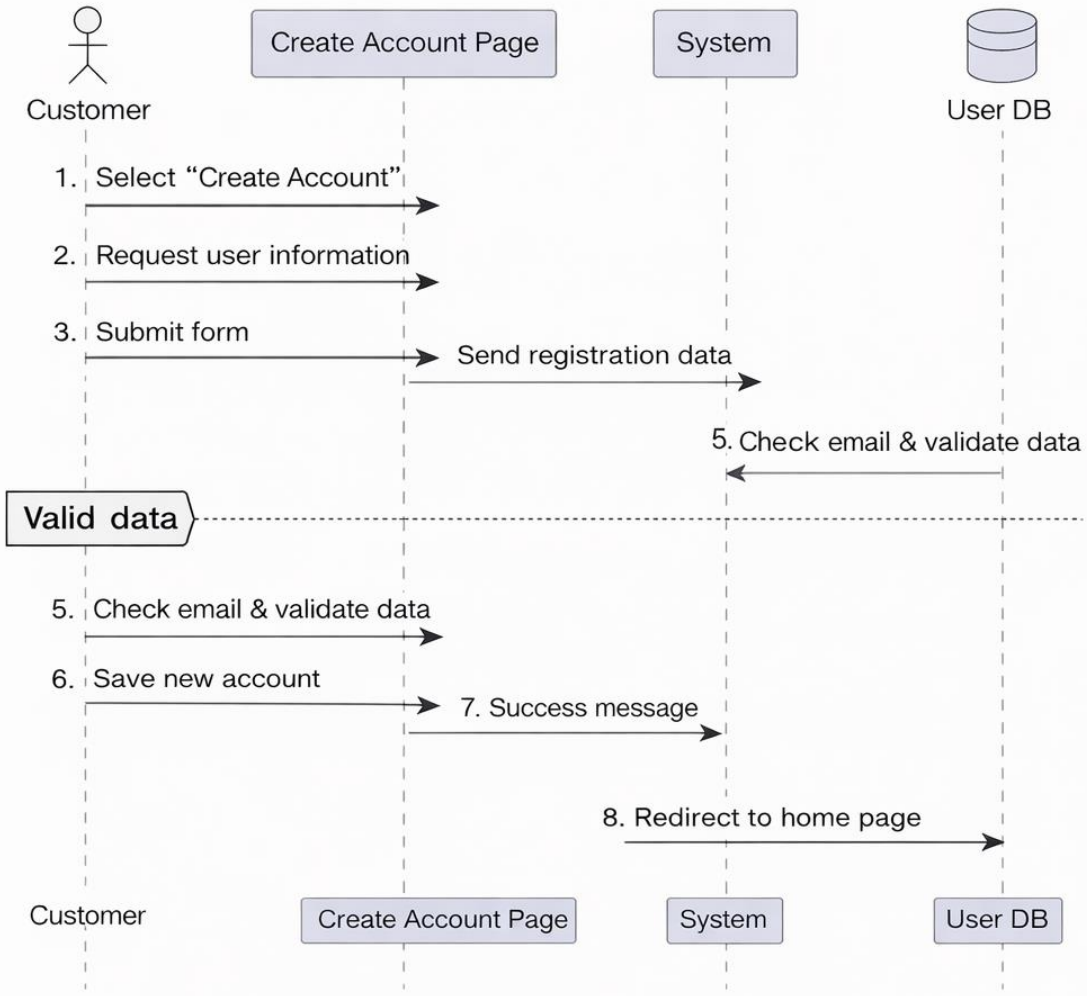
6. System features (use case specifications - Sequence Diagrams-Activity Diagrams):

6.1 Create User account:

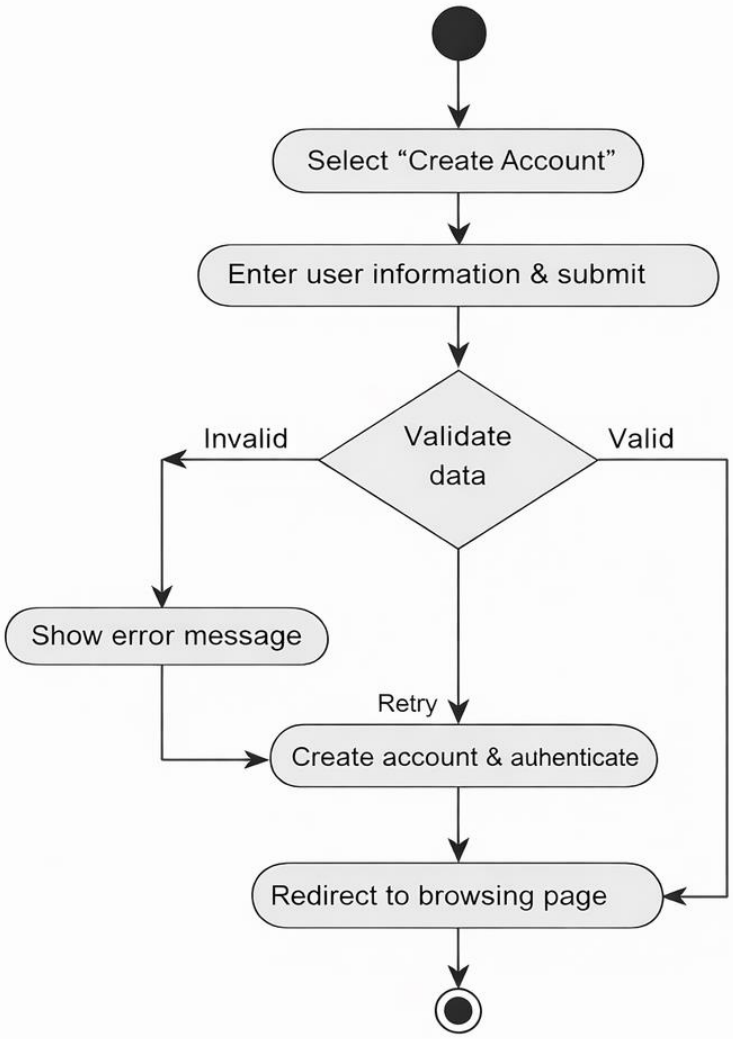
item	Description
Use Case Name	Create a New User Account
Actors	Customer
Goal	- User: Obtain a valid account to access the platform's services. - System: Ensure valid input and prevent fake registrations.
Preconditions	- The user has navigated to the "Create Account" page. - The system is ready to accept new registrations.
Main Flow	1. User selects “Create Account”. 2. System requests full user information. 3. User fills the form and submits. 4. System validates the data. 5. System displays a success message.
Alternative Flow 1	Empty Fields: 1. System detects missing required fields. 2. Shows “Please fill out this field” next to empty fields. 3. User fills missing fields.
Alternative Flow 2	Email Already Used: 1. System finds that the email already exists. 2. Shows “This email is already in use.” 3. User modifies the email.
Alternative Flow 3	Short Password: 1. Password is less than 8 characters. 2. System shows “Password is too short.” 3. User adjusts the password.
Alternative Flow 4	Invalid Email Format: 1. Email format is incorrect (missing @gmail.com). 2. System displays “Enter a valid Gmail address.” 3. User corrects the email.

Postconditions	- Success: Account is created, user is authenticated, redirected to project browsing page. - Failure: Errors shown, user remains on registration page.

Sequence Diagram: Create New Account

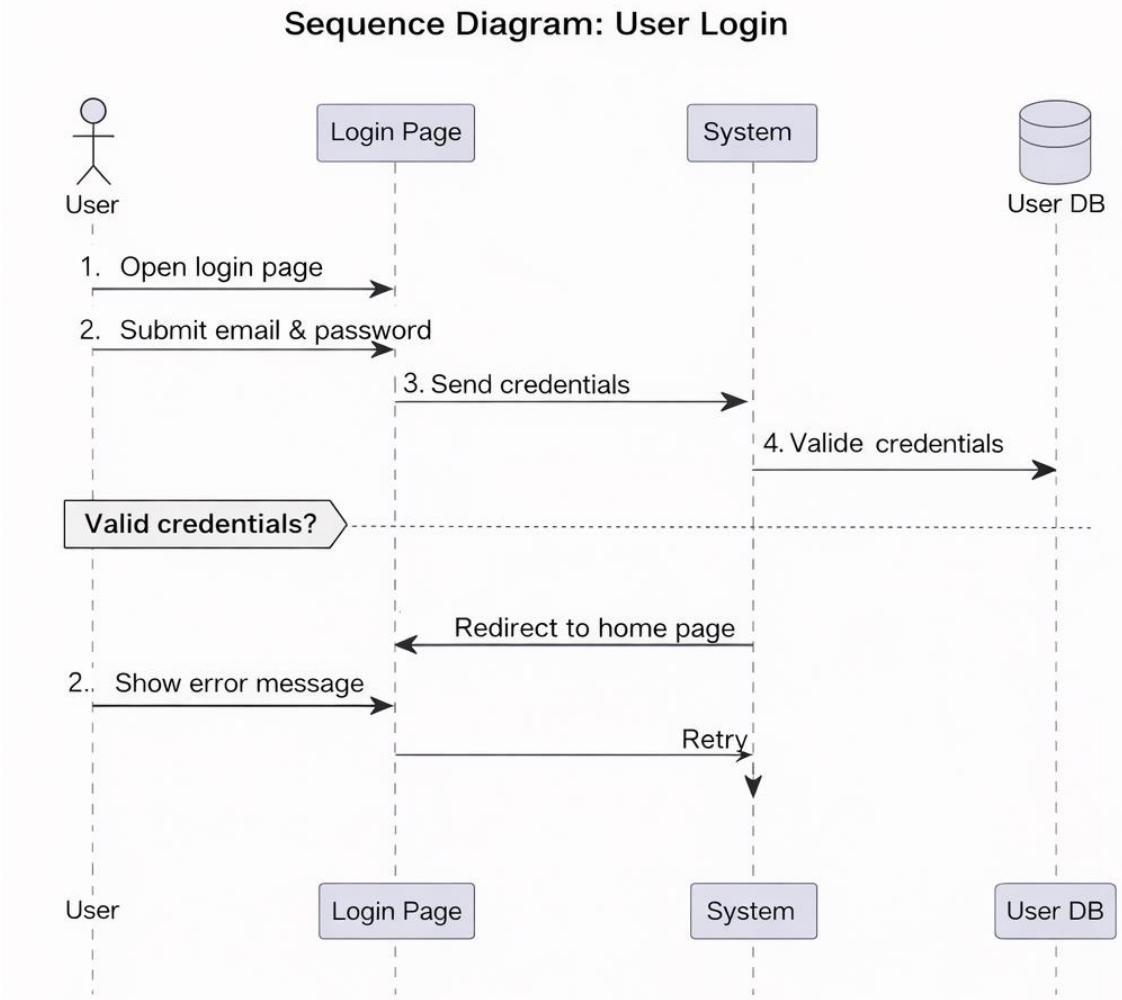


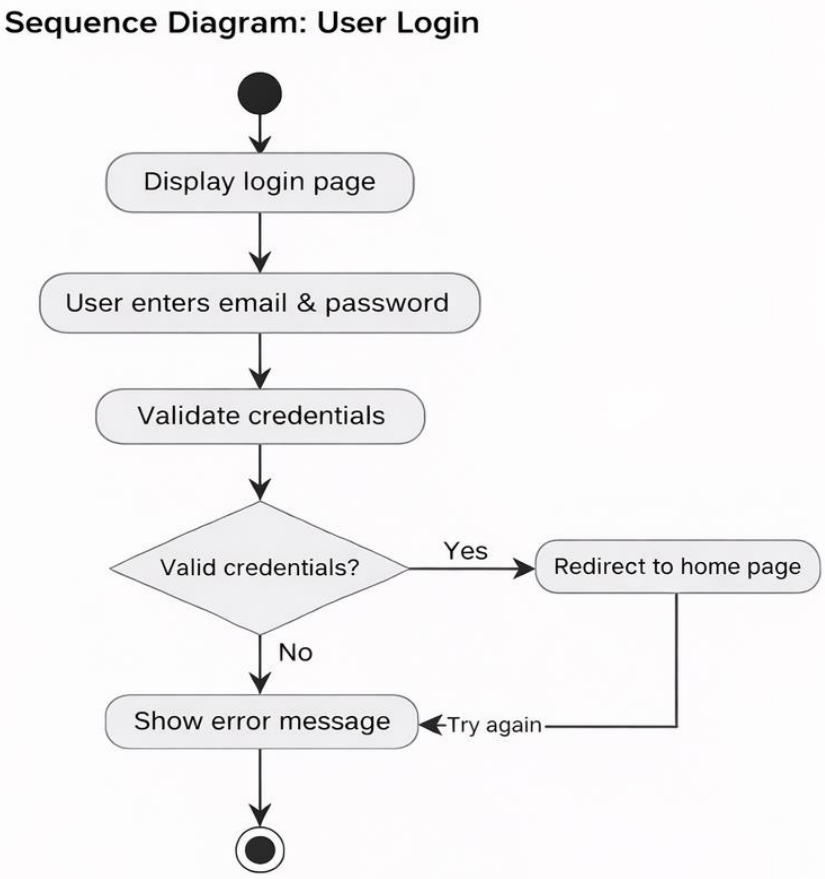
Activity Diagram: Create New User Account



6.2 Login

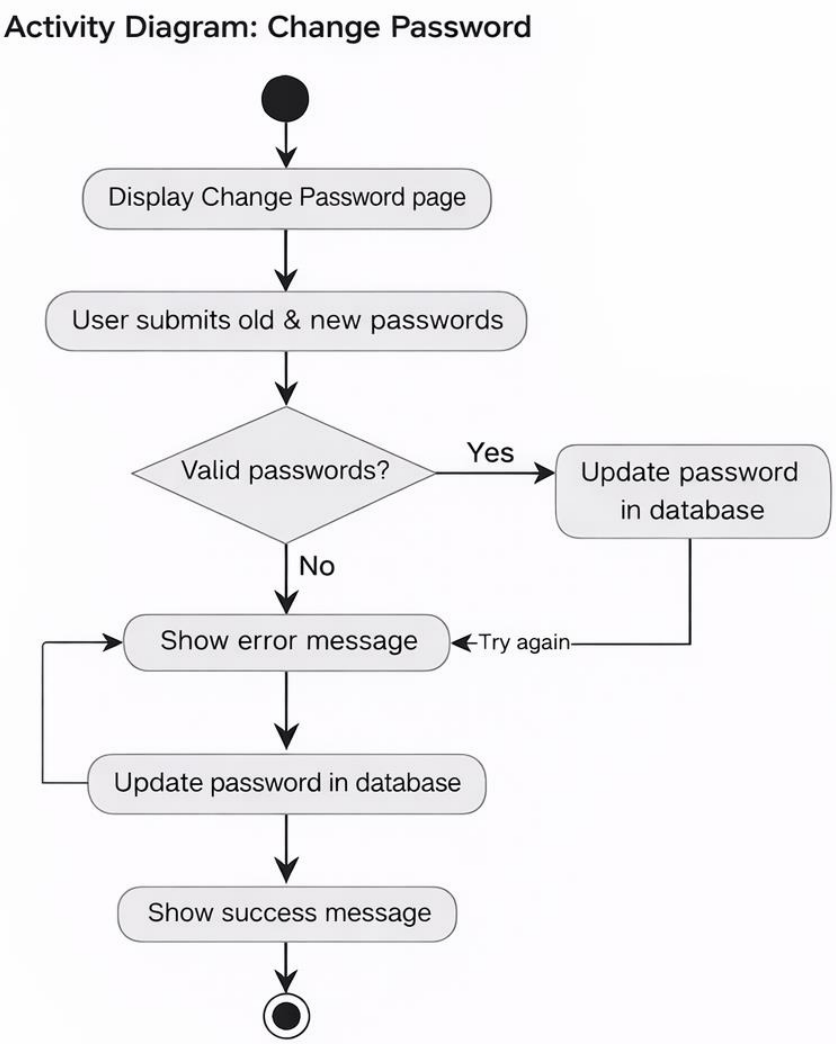
Item	Description
Use Case Name	User Login
Actors	Admin/customer
Goal	Allow the user to access the system using a valid Gmail email and password, and navigate to the home page.
Preconditions	1. Login page is displayed. 2. A user account with a registered @gmail.com email and password exists in the database.
Main Flow	1. System displays the login page with email and password fields, and a "Login" button. 2. User enters a valid Gmail and correct password, then clicks "Login". 3. System validates credentials against the database. 4. If credentials are correct, the system redirects the user to the home page.
Alternative Flow 1	Wrong Password: 1. System detects password mismatch. 2. Displays “Incorrect password.” 3. Returns to the login page.
Alternative Flow 2	Unregistered Email: 1. System detects email is not found in the database. 2. Displays “Invalid email address.” 3. Returns to login page.
Alternative Flow 3	Invalid Email Format: 1. System detects missing @gmail.com in the email. 2. Displays “Email must include ‘@gmail.com’.” 3. Returns to login page.
Postconditions	Success: User is authenticated and redirected to the home page. - Failure: User remains on login page with an appropriate error message.

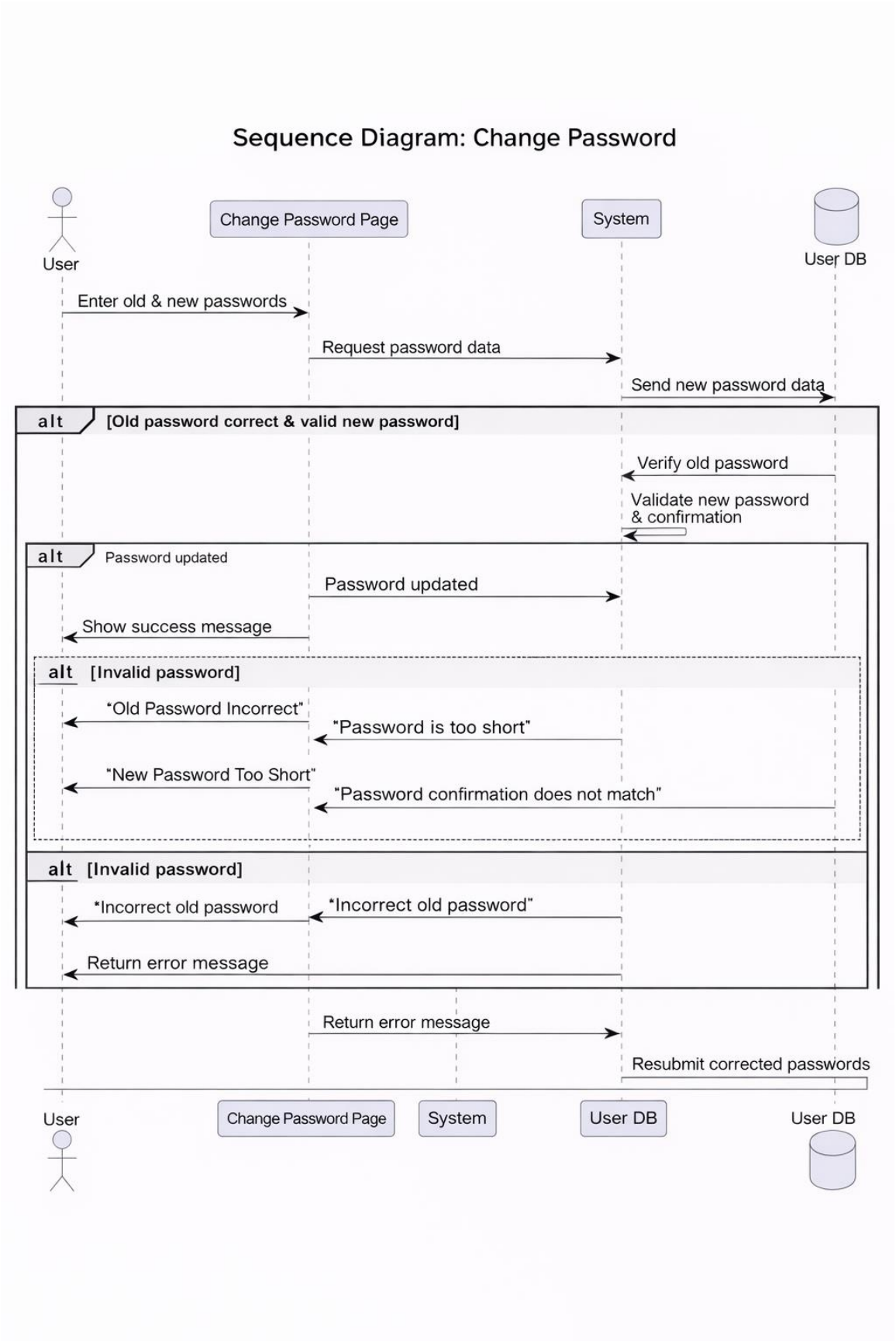

Sequence Diagram: User Login
User
Login Page
System
User DB
1. Open login page
2. Submit email & password
3. Send credentials
4. Valide credentials
Valid credentials?
Redirect to home page
2.. Show error message
Retry
User
Login Page
System
User DB


Sequence Diagram: User Login
Display login page
User enters email & password
Validate credentials
Valid credentials?
Yes
Redirect to home page
No
Show error message
Try again

6.3 Change password

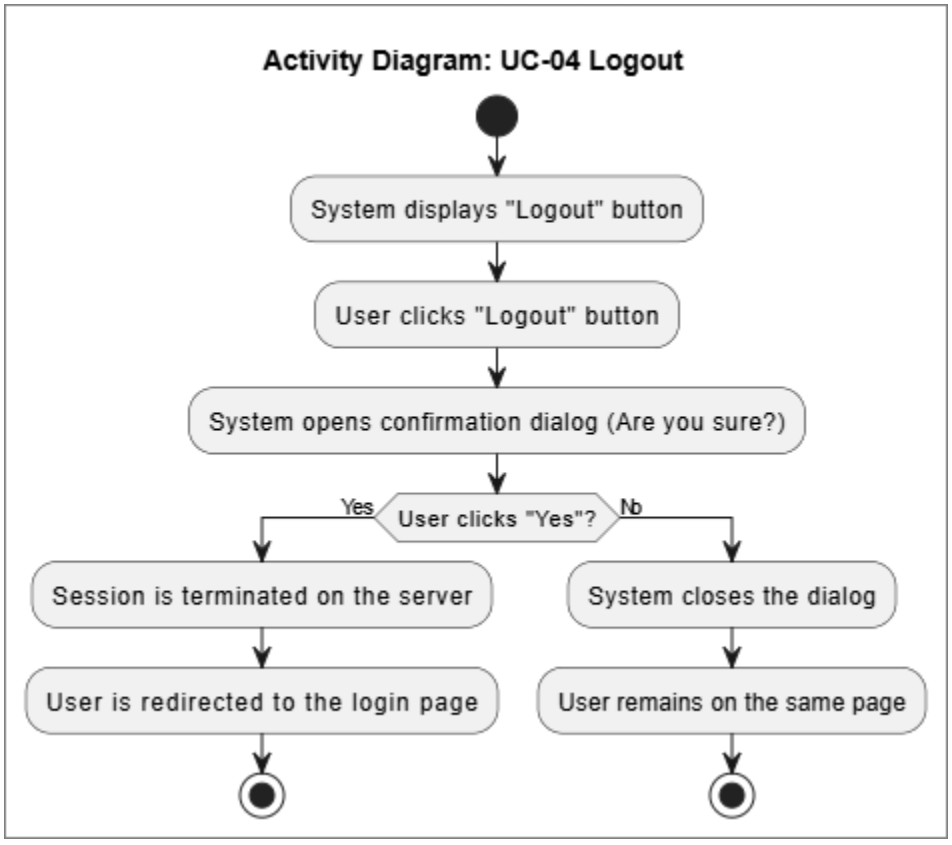
Item	Description
Use Case Name	Change Password
Actors	Admin/customer
Goal	The user requests to change their current password to a new confirmed password to ensure account security.
Preconditions	1. The user is logged in (authenticated session). 2. The system displays the "Change Password" page. 3. The account is not blocked or under maintenance.
Main Flow	1. System displays the "Change Password" page with fields: Old Password, New Password, Confirm New Password, and a "Change" button. 2. User enters the correct old password, new password, and confirms it. 3. System verifies the information and updates the password.
Alternative Flow 1	Incorrect Old Password: 1. System detects mismatch with the stored password. 2. Displays “Incorrect old password.” 3. Returns to the change password page, preserving the other fields.
Alternative Flow 2	New Password Too Short: 1. System detects that the new password is less than 8 characters. 2. Displays “Password is too short – must be at least 8 characters.” 3. Clears the New Password and Confirm Password fields. 4. Returns to the change password page.
Alternative Flow 3	Password Confirmation Mismatch: 1. System detects that confirmation does not match the new password. 2. Displays “Password confirmation does not match.” 3. Clears the New Password and Confirm Password fields. 4. Returns to the change password page.
Postconditions	- Success: Password is successfully updated and “Password changed successfully” message is displayed. - Failure: User remains on the change password page with an appropriate error message.

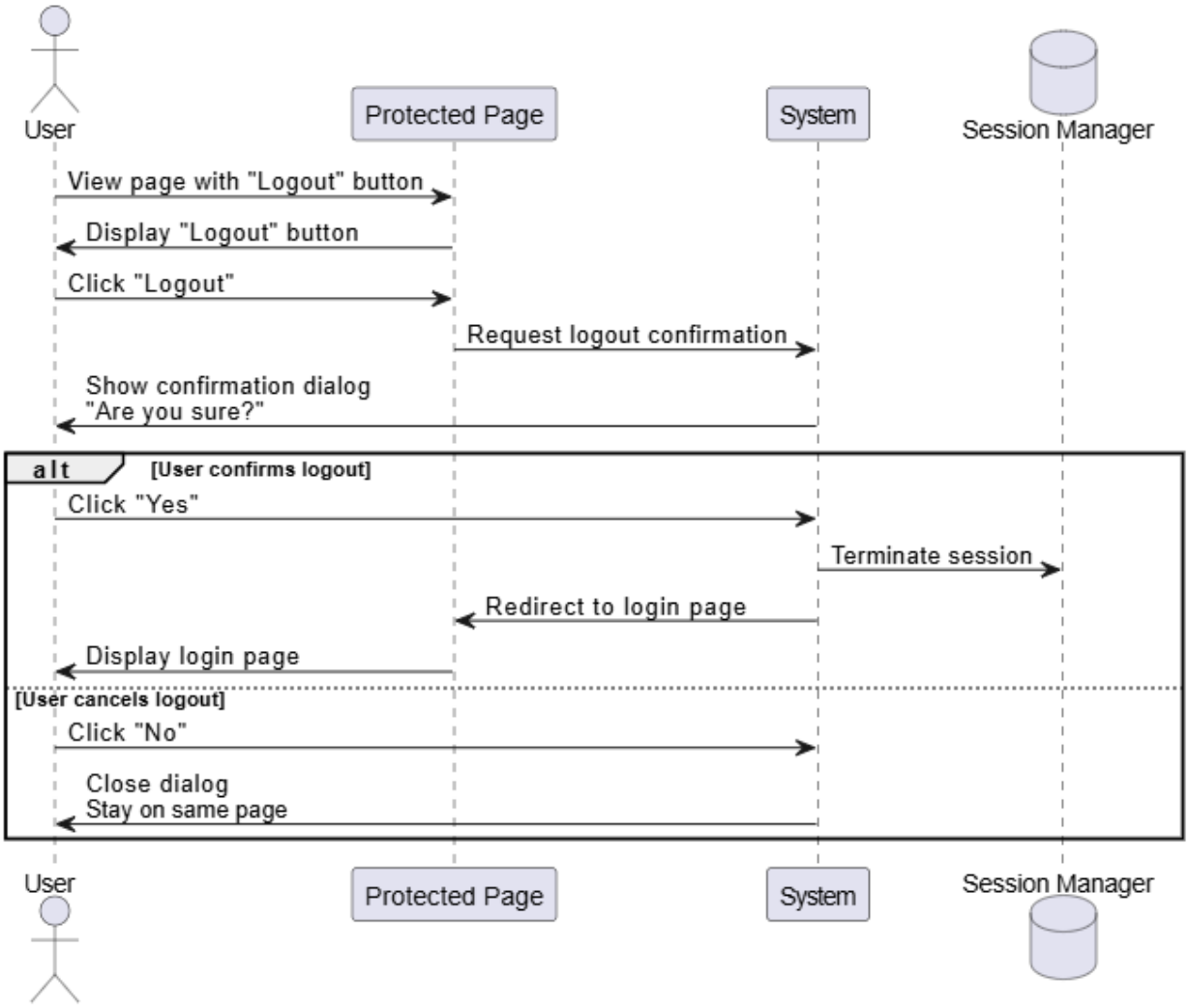

Activity Diagram: Change Password
Display Change Password page
User submits old & new passwords
Valid passwords?
Yes
Update password in database
No
Show error message
Try again
Update password in database
Show success message


Sequence Diagram: Change Password
User
Change Password Page
System
User DB
Enter old & new passwords
Request password data
Send new password data
alt
[Old password correct & valid new password]
Verify old password
Validate new password
& confirmation
alt
Password updated
Password updated
Show success message
alt
[Invalid password]
"Old Password Incorrect"
"Password is too short"
"New Password Too Short"
"Password confirmation does not match"
alt
[Invalid password]
*Incorrect old password
*Incorrect old password"
Return error message
Return error message
Resubmit corrected passwords
User
Change Password Page
System
User DB
User DB

6.4 Logout :

Item	Description
Use Case Name	Logout
Actors	Admin/customer
Goal	Terminate the authenticated session and redirect the user to the login page.
Preconditions	1. The user is authenticated and has an active session. 2. The user is on a protected page with a "Logout" button available.
Main Flow	1. The system displays the “Logout” button on the current page 2. The user clicks the “Logout” button. 3. The system opens a confirmation dialog with the message “Are you sure you want to logout?” and options “Yes” and “No”. 3. The user clicks “Yes”. 4. The system redirects the user to the login page.
Alternative Flow 1	User Cancels Logout: 1. Steps 1–3 of the main flow are executed. 2. The user clicks “No” in the confirmation dialog. 3. The system closes the dialog. 4. The user remains on the same page; the session stays active.
Postconditions	If logout confirmed: • Session is terminated on the server. • User is redirected to the login page. - If logout canceled: • User remains on the same page with the session still active

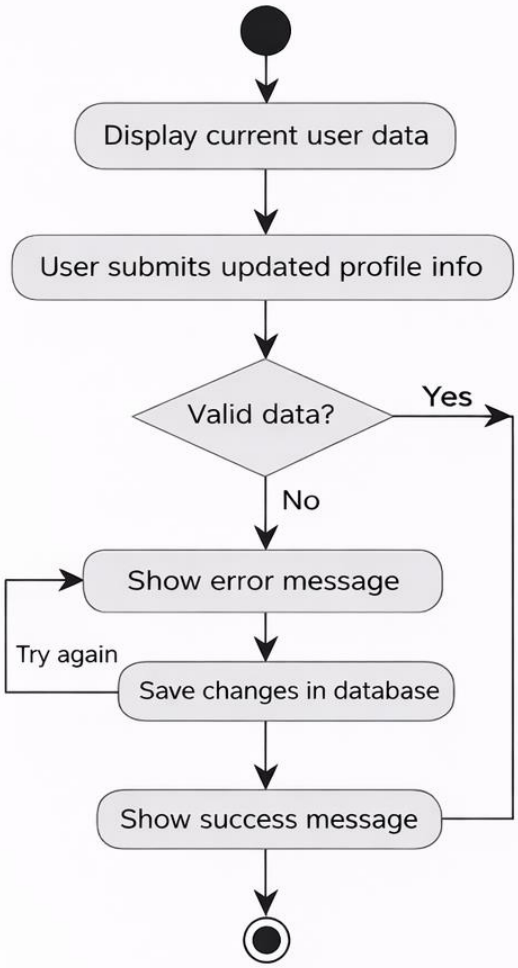

Activity Diagram: UC-04 Logout
System displays "Logout" button
User clicks "Logout" button
System opens confirmation dialog (Are you sure?)
User clicks "Yes"?
Yes
No
Session is terminated on the server
User is redirected to the login page
System closes the dialog
User remains on the same page

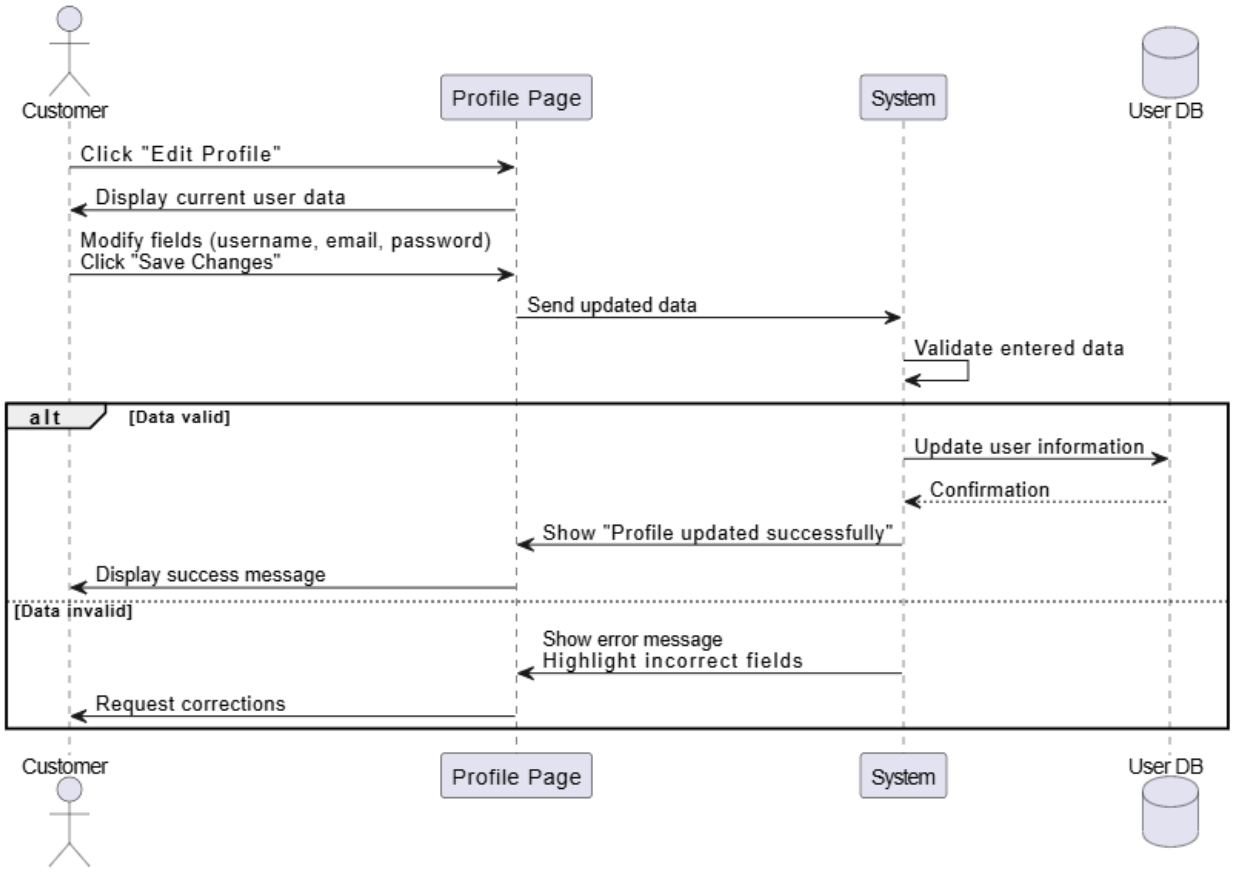

User
Protected Page
System
Session Manager
View page with "Logout" button
Display "Logout" button
Click "Logout"
Request logout confirmation
Show confirmation dialog
"Are you sure?"
alt
[User confirms logout]
Click "Yes"
Terminate session
Redirect to login page
Display login page
[User cancels logout]
Click "No"
Close dialog
Stay on same page
User
Protected Page
System
Session Manager

6.5 Edit customer profile information

Item	Description
Use Case Name	Edit customer profile information
Actors	customer
goal	To ensure that personal information is accurate and up-to-date, such as name, email address, phone number, and password.
Preconditions	login
Main Flow	1. When the user clicks on the option to modify personal data 2. The system displays user data 3. The user fills in the fields with the data he wishes to modify (username - password - email) and presses the save changes button 4. The system verifies the entered data 5. If the data entered is correct, the system saves the changes and updates the database
Alternative Flow 1	• If there is an error in the entered data (not matching the pattern), an error message will be displayed requesting the user to modify the fields indicated in red.
Postconditions	Completed the process successfully

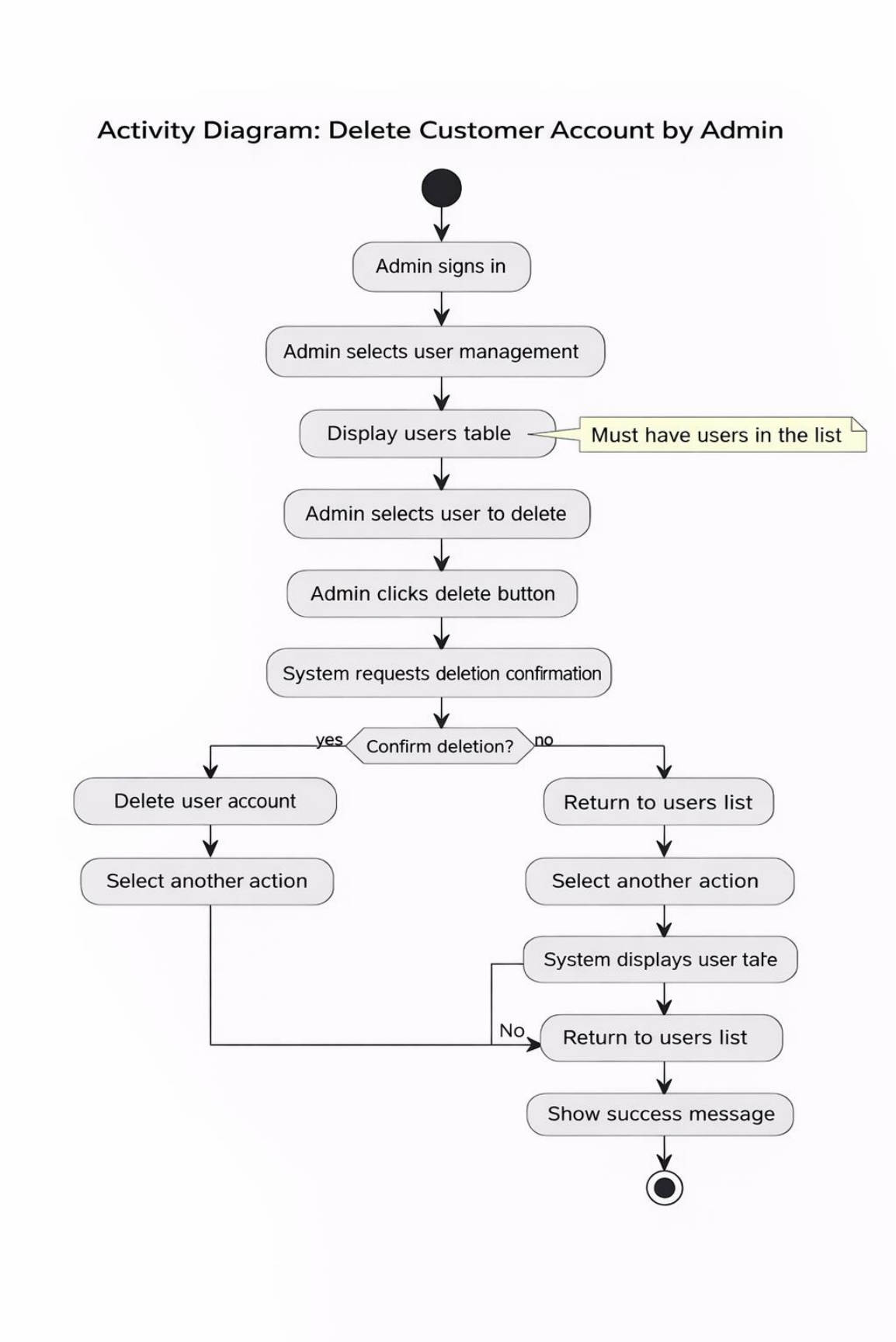
Activity Diagram: Edit Customer Profile Information

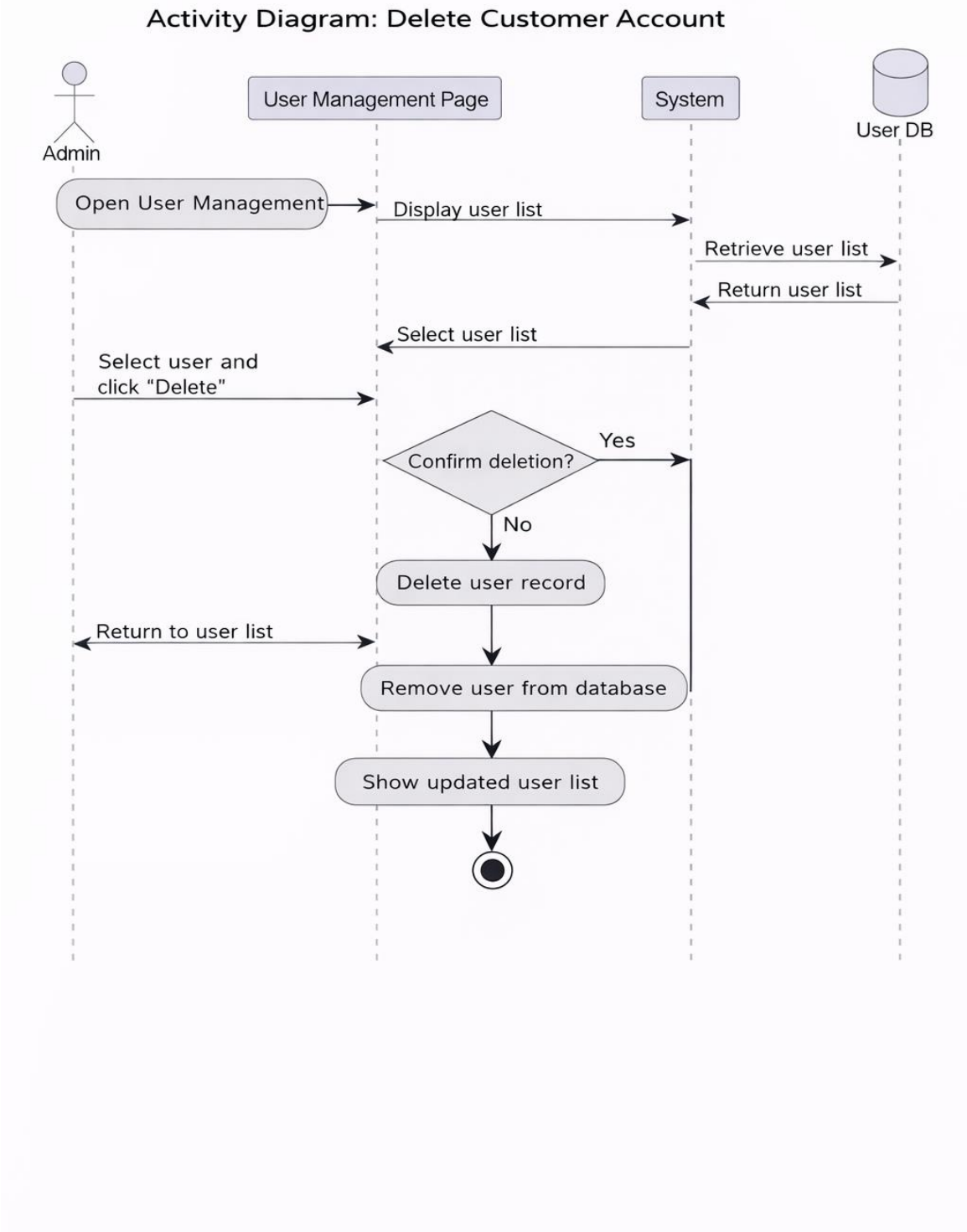



Customer
Profile Page
System
User DB
Click "Edit Profile"
Display current user data
Modify fields (username, email, password)
Click "Save Changes"
Send updated data
Validate entered data
alt
[Data valid]
Update user information
Confirmation
Show "Profile updated successfully"
Display success message
[Data invalid]
Show error message
Highlight incorrect fields
Request corrections
Customer
Profile Page
System
User DB

6.6 Delete customer account by admin

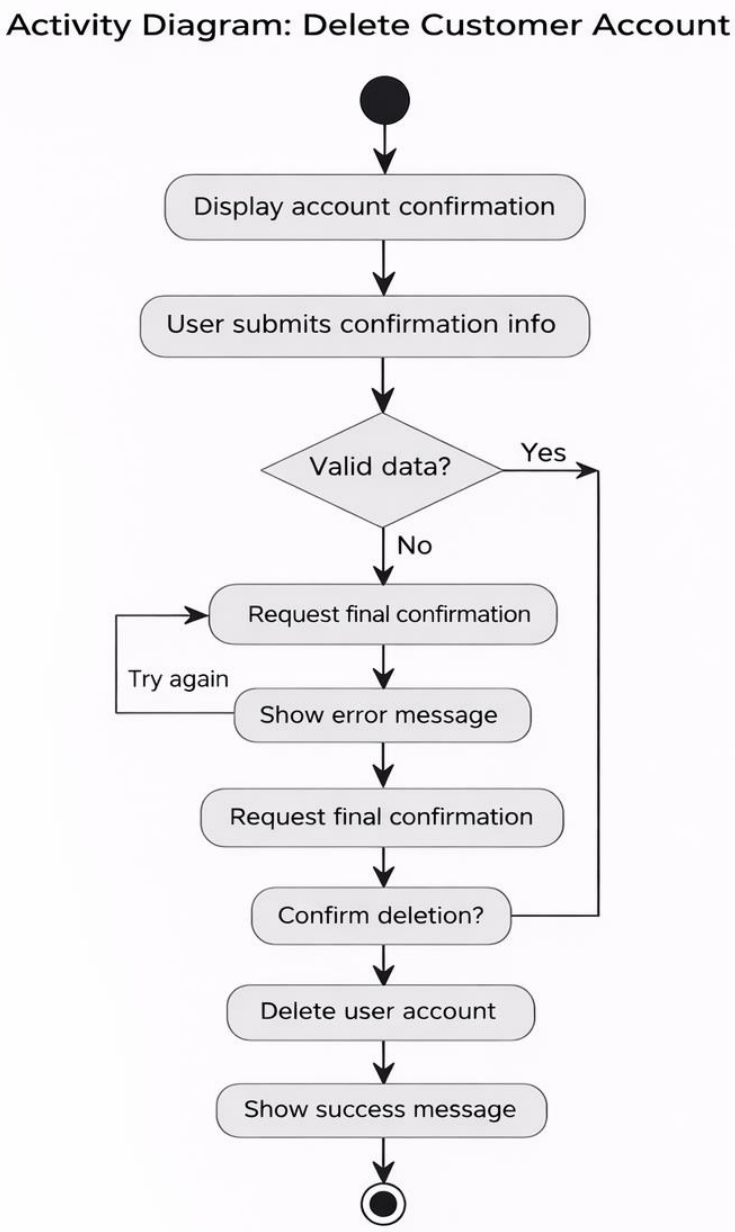
Item	Description
Use Case Name	Delete customer account by admin
Actors	admin
goal	Delete user data for which a file has already been created
Preconditions	-Sign in - The presence of users in the list of users
Main Flow	1. The administrator chooses user management 2. the system shows a table containing the list of users 3. The manager chooses the user he wants to delete his data,When you press the delete button 4. the system asks to confirm the deletion process. 5. the manager confirms the deletion process. 6. the system saves the changes and updates the database
Alternative Flow 1	- If the deletion process is not confirmed, return to the list of users view
Postconditions	Successful completion of the process

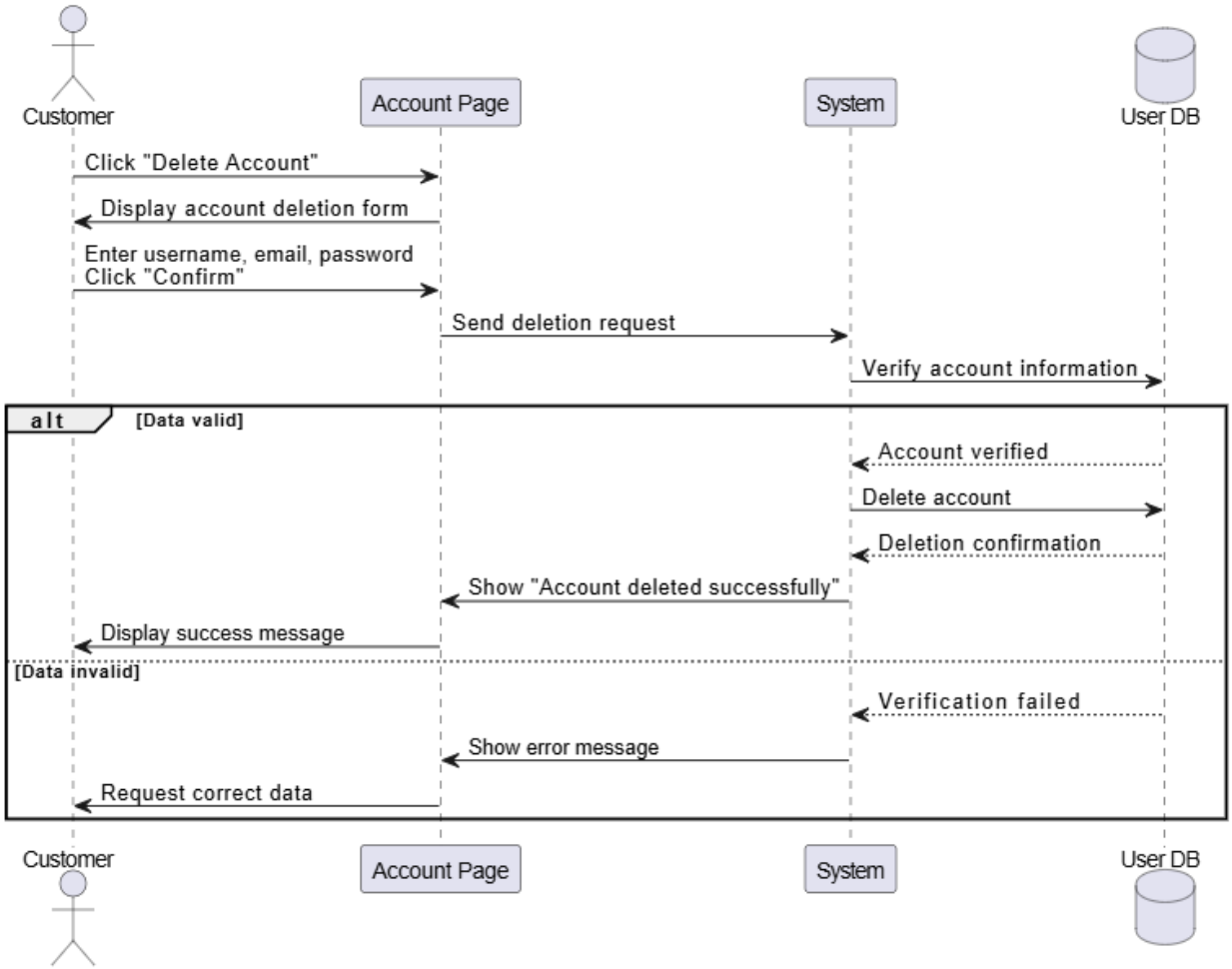

Activity Diagram: Delete Customer Account by Admin
Admin signs in
Admin selects user management
Display users table
Must have users in the list
Admin selects user to delete
Admin clicks delete button
System requests deletion confirmation
Confirm deletion?
yes
no
Delete user account
Select another action
Return to users list
Select another action
System displays user tale
No
Return to users list
Show success message


Activity Diagram: Delete Customer Account
Admin
User Management Page
System
User DB
Open User Management
Display user list
Retrieve user list
Return user list
Select user list
Select user and
click "Delete"
Confirm deletion?
Yes
No
Delete user record
Return to user list
Remove user from database
Show updated user list

6.7 Delete customer account by custome

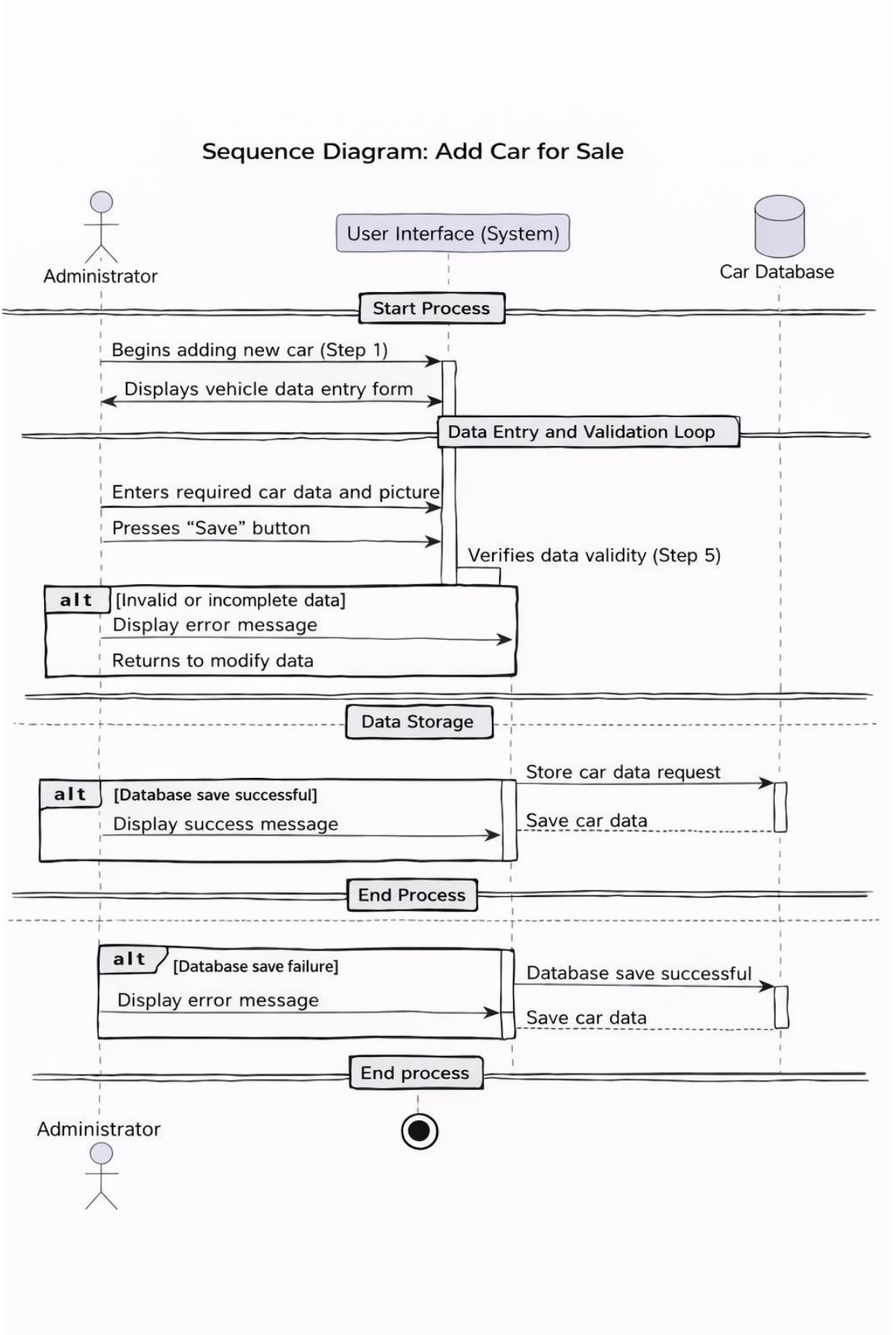
Item	Description
Use Case Name	Delete customer account by customer
Actors	customer
goal	Delete user data for which a file has already been created
Preconditions	-Sign in
Main Flow	1. When the user clicks on the option to delete personal account 2. The system displays an interface for the user to enter account information. 3. The user fills in the fields with data (username, password, email) and clicks the confirm button. 4. The system verifies the validity of the entered data; if it is correct, it deletes the account, and if it is incorrect, it prompts the user to re-enter the data correctly.
Alternative Flow 1	- If the deletion process is not confirmed, return to the
Postconditions	Successful completion of the process

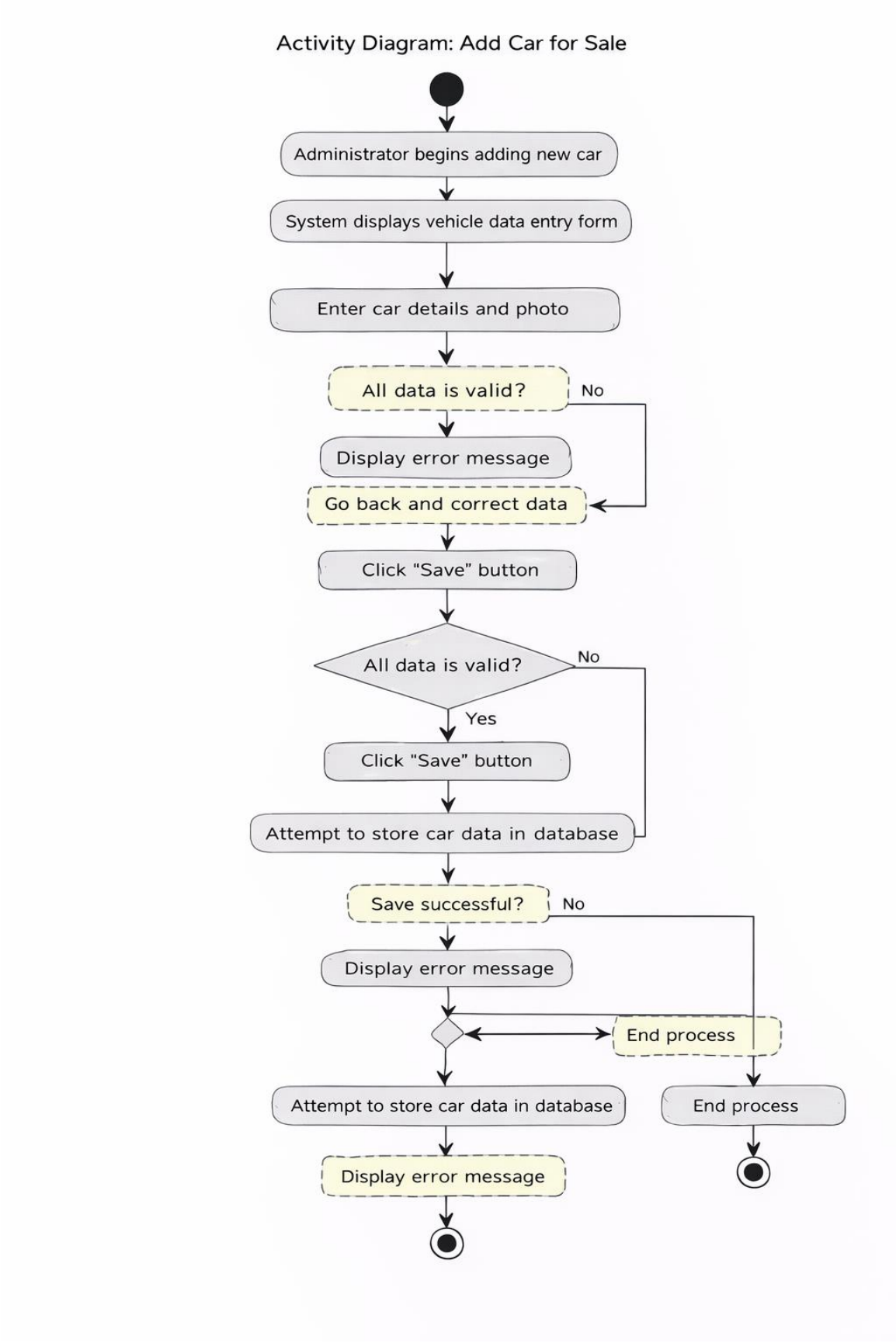

Activity Diagram: Delete Customer Account
Display account confirmation
User submits confirmation info
Valid data?
Yes
No
Request final confirmation
Try again
Show error message
Request final confirmation
Confirm deletion?
Delete user account
Show success message


Customer
Account Page
System
User DB
Click "Delete Account"
Display account deletion form
Enter username, email, password
Click "Confirm"
Send deletion request
Verify account information
alt
[Data valid]
Account verified
Delete account
Deletion confirmation
Show "Account deleted successfully"
Display success message
[Data invalid]
Verification failed
Show error message
Request correct data
Customer
Account Page
System
User DB

6.8 Add car for sale

item	Description
Use Case Name	Add car for sale
Actors	Administrator
Goal	Add new car to the system of offering it for sale
Preconditions	The administrator must be logged into the system
Main Flow	1. The administrator begins the process of adding a new car for sale. 2. The system displays the vehicle data entry model. 3. The administrator enters the required data: • Type (Type) • Description (Description) • Price (Price) • Model (Model) • Picture (Picture) • Year of manufacture 4. The administrator presses the “Save” button. 5. The system verifies the validity of the entered data. 6. The system stores the car data in the database. 7. The system displays a message confirming the success of the addition process. 8. The process ends
Postcondition	• Success of the operation: The car's data is stored in the database and becomes visible in the list of cars for sale. • Operation failure: No new record is created for the car
Alternative Flow 1	– Incomplete or incorrect data • . The system detects a lack of data or errors (such as an invalid price). • . The system displays an error message explaining the issue. • . The administrator returns to modify the data. • The flow returns to step 4 of the main flow.
Alternative Flow 2	– Image Loading Failure • The administrator tries to upload an image but the system does not accept it (unsupported extension or large size). • . The system displays an error message about the image issue. • . The administrator chooses another image. • The system returns to step 3 in the main flow.
Alternative Flow 3	– Failure to save data (system error) • . A database error occurs while trying to save. • . The system displays a message stating that the process failed. • . The car is not created. • The case ends unsuccessfully.

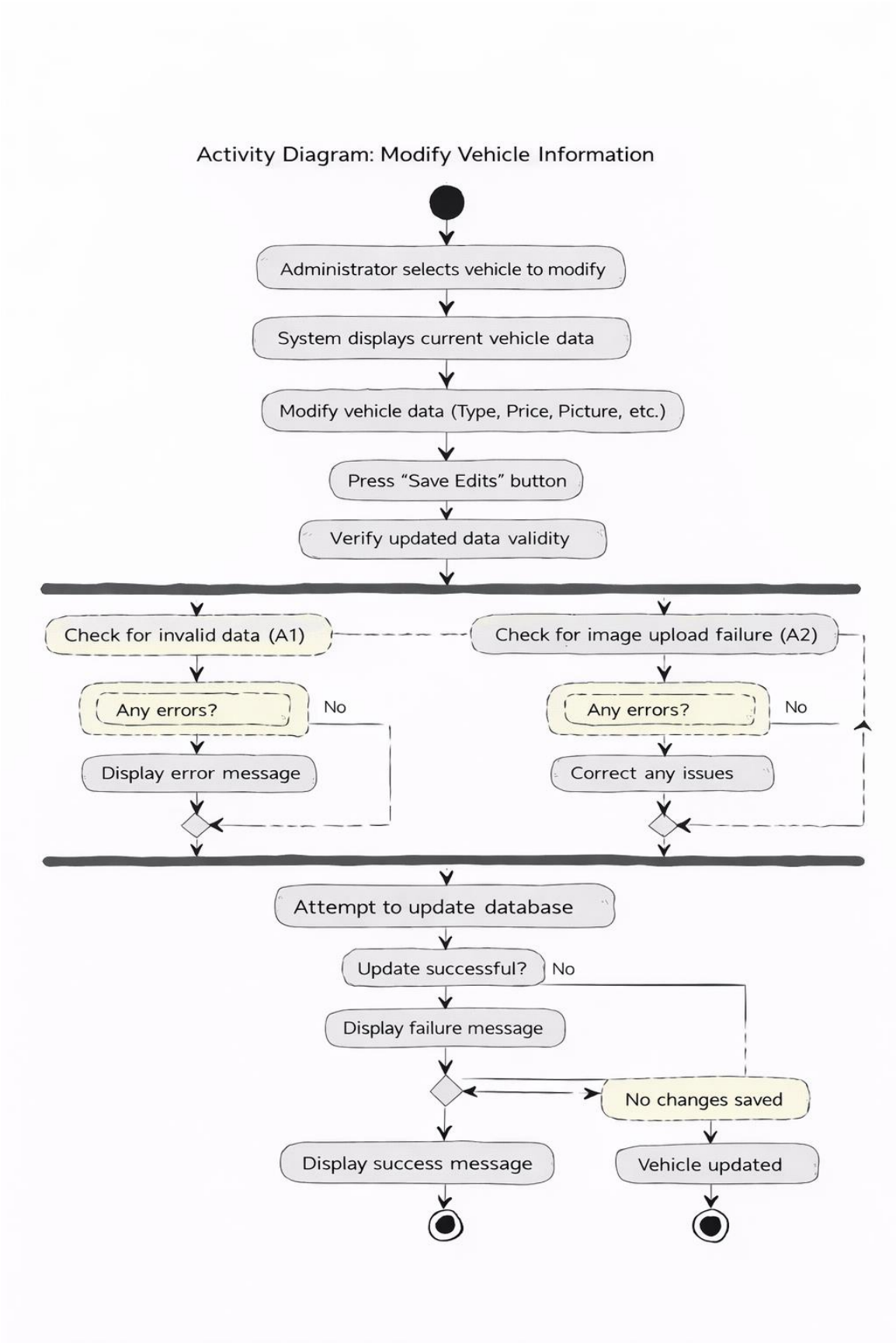

Sequence Diagram: Add Car for Sale
Administrator
User Interface (System)
Car Database
Start Process
Begins adding new car (Step 1)
Displays vehicle data entry form
Data Entry and Validation Loop
Enters required car data and picture
Presses “Save” button
Verifies data validity (Step 5)
alt
[Invalid or incomplete data]
Display error message
Returns to modify data
Data Storage
Store car data request
alt
[Database save successful]
Display success message
Save car data
End Process
alt
[Database save failure]
Database save successful
Display error message
Save car data
End process
Administrator

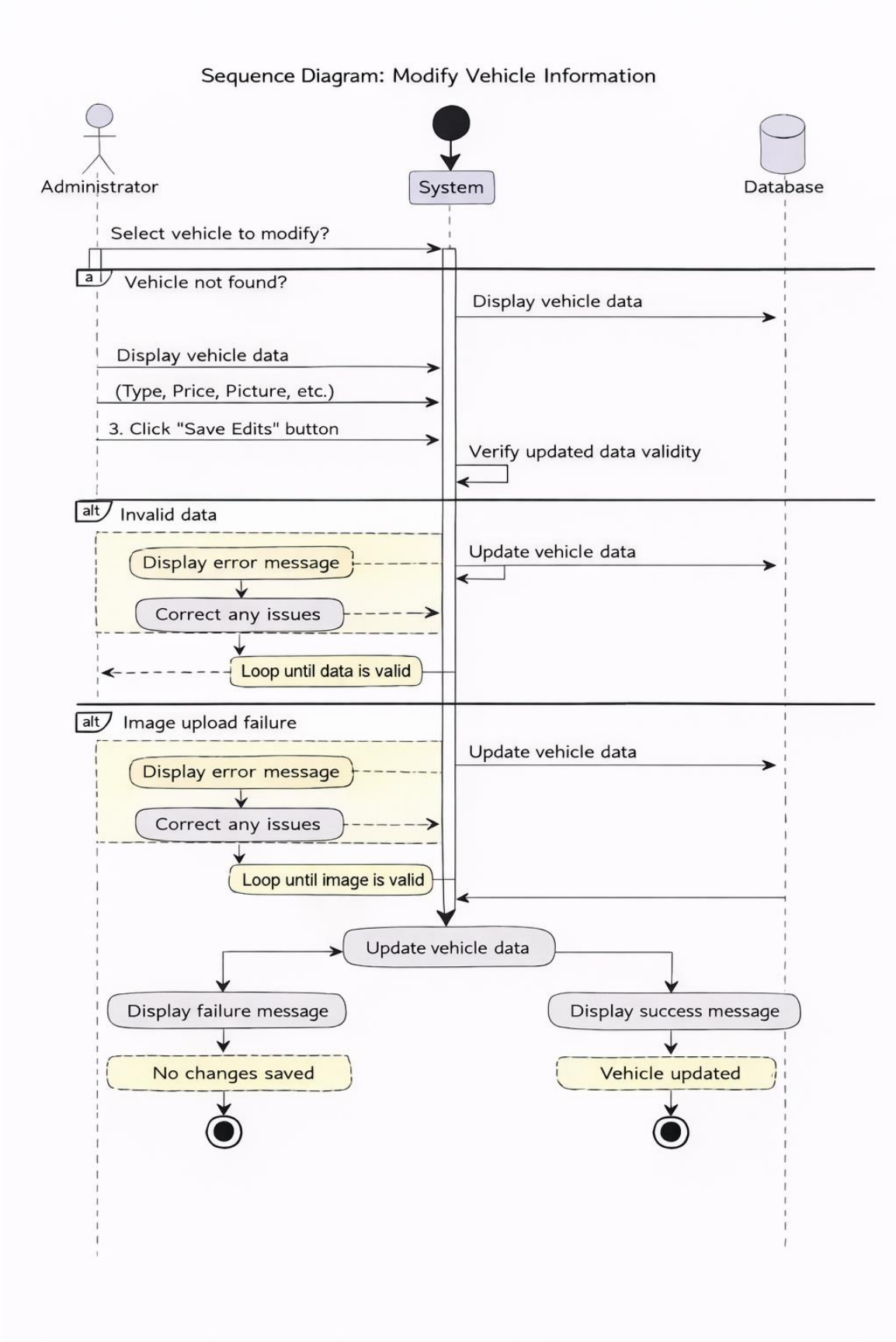

Activity Diagram: Add Car for Sale
Administrator begins adding new car
System displays vehicle data entry form
Enter car details and photo
All data is valid?
No
Display error message
Go back and correct data
Click "Save" button
All data is valid?
No
Yes
Click "Save" button
Attempt to store car data in database
Save successful?
No
Display error message
End process
Attempt to store car data in database
End process
Display error message

6.9 Modify vehicle information for sale

item	Description
Use Case Name	Modify vehicle information for sale
Actors	Administrator
Goal	This use case allows the administrator to modify pre-existing vehicle information in the system that is on sale.
Preconditions	The administrator must be logged into the system. The vehicle must have already been in the database
Main Flow	1. The administrator selects a specific vehicle to modify its data. 2. The system displays current vehicle data. 3. The administrator will modify one or more of the following fields: • Type (Type) • Description (Description) • Price (Price) • Model (Model) • Picture (Picture) • Year of manufacture 4. The administrator presses the “Save Edits” button. 5. The system verifies the validity of the updated data. 6. The system updates the vehicle data in the database. 7. The system displays a message confirming the success of the modification process. 8. The process ends successfully
Alternative Flow 1	– Entering invalid data The system detects that some of the data entered is invalid (such as a non-digital price or incorrect year). The system displays an error message explaining the issue. The administrator returns to correct the data. The system returns to step 4 in the main flow.
Alternative Flow 2	– Updated Image Upload Failed • The administrator tries to upload a new image but the system rejects it (unsupported extension or large size). • The system displays an error message about the image issue. • 3c. The administrator chooses another image. • 3d. The system returns to step 3 in the main flow.
Alternative Flow 3	– Vehicle does not exist • The administrator selects a vehicle that does not exist or has been deleted. • The system displays a message that the vehicle is not available. • The process ends without modification.

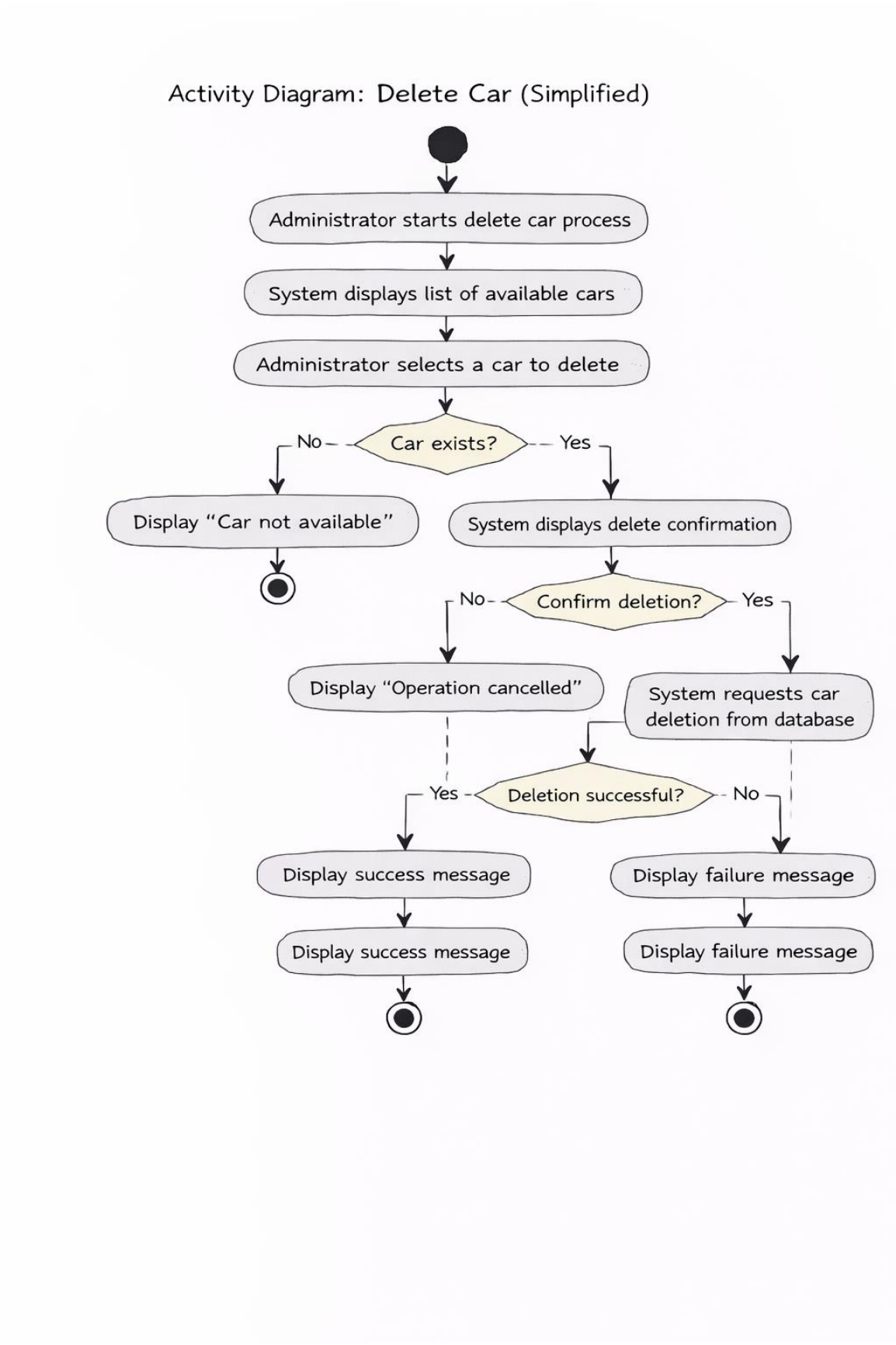
Alternative Flow 4	– Failure to save edits The system fails to update data due to a database glitch or internal error. • The system displays a message that the modification process failed. • No changes are saved. • The case ends unsuccessfully.
Postconditions	Success of the operation: Vehicle data is updated and stored in the database. • Operation failure: None of the vehicle data is changed

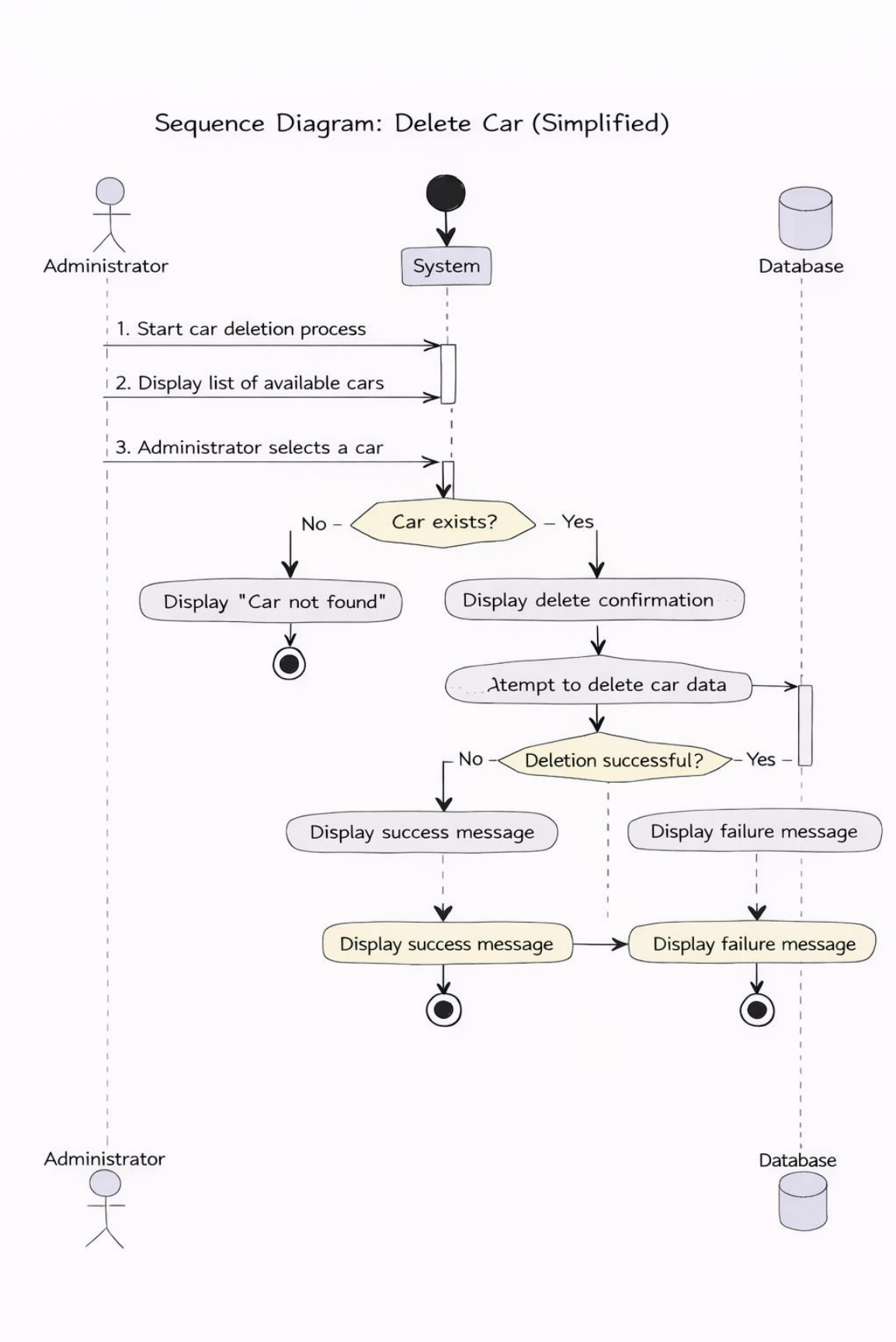

Activity Diagram: Modify Vehicle Information
Administrator selects vehicle to modify
System displays current vehicle data
Modify vehicle data (Type, Price, Picture, etc.)
Press “Save Edits” button
Verify updated data validity
Check for invalid data (A1)
Check for image upload failure (A2)
Any errors?
No
Display error message
Any errors?
No
Correct any issues
Attempt to update database
Update successful?
No
Display failure message
No changes saved
Display success message
Vehicle updated


Sequence Diagram: Modify Vehicle Information
Administrator
System
Database
Select vehicle to modify?
a
Vehicle not found?
Display vehicle data
Display vehicle data
(Type, Price, Picture, etc.)
3. Click "Save Edits" button
Verify updated data validity
alt
Invalid data
Display error message
Correct any issues
Loop until data is valid
Update vehicle data
alt
Image upload failure
Display error message
Correct any issues
Loop until image is valid
Update vehicle data
Update vehicle data
Display failure message
No changes saved
Display success message
Vehicle updated

6.10 Delete car

item	Description
Use Case Name	Delete car
Actors	Administrator
Goal	This case allows the administrator to delete an existing vehicle in the system
Preconditions	The administrator must be logged into the system. The car must be pre-existed in the database
Main Flow	1. The administrator begins the process of deleting a car. 2. The system displays a list of available cars. 3. The administrator specifies the vehicle to be deleted. 4. The system displays a confirmation message “Do you definitely want to delete the car?”. 5. The administrator confirms the deletion process. 6. The system deletes the car from the database. 7. The system displays a message indicating the success of the deletion process. 8. The process ends
Alternative Flow 1	– The car does not exist • The administrator selects a car that was previously deleted or did not exist. • The system displays a message stating that the car is not available. • The process ends without deleting.
Alternative Flow 2	– Cancel the deletion process • The administrator rejects the deletion process and clicks “Cancel”. • The system displays a message that the operation has been canceled. • The process ends without any change.
Alternative Flow 3	– Operation failure due to system error • The system tries to delete the car but an error occurs in the database. • The system displays a message with the failure of the deletion process. • The car stays the same. • The process ends unsuccessfully.
Postconditions	• Operation success: The vehicle is permanently deleted from the database. • Operation failure: the car remains unchanged

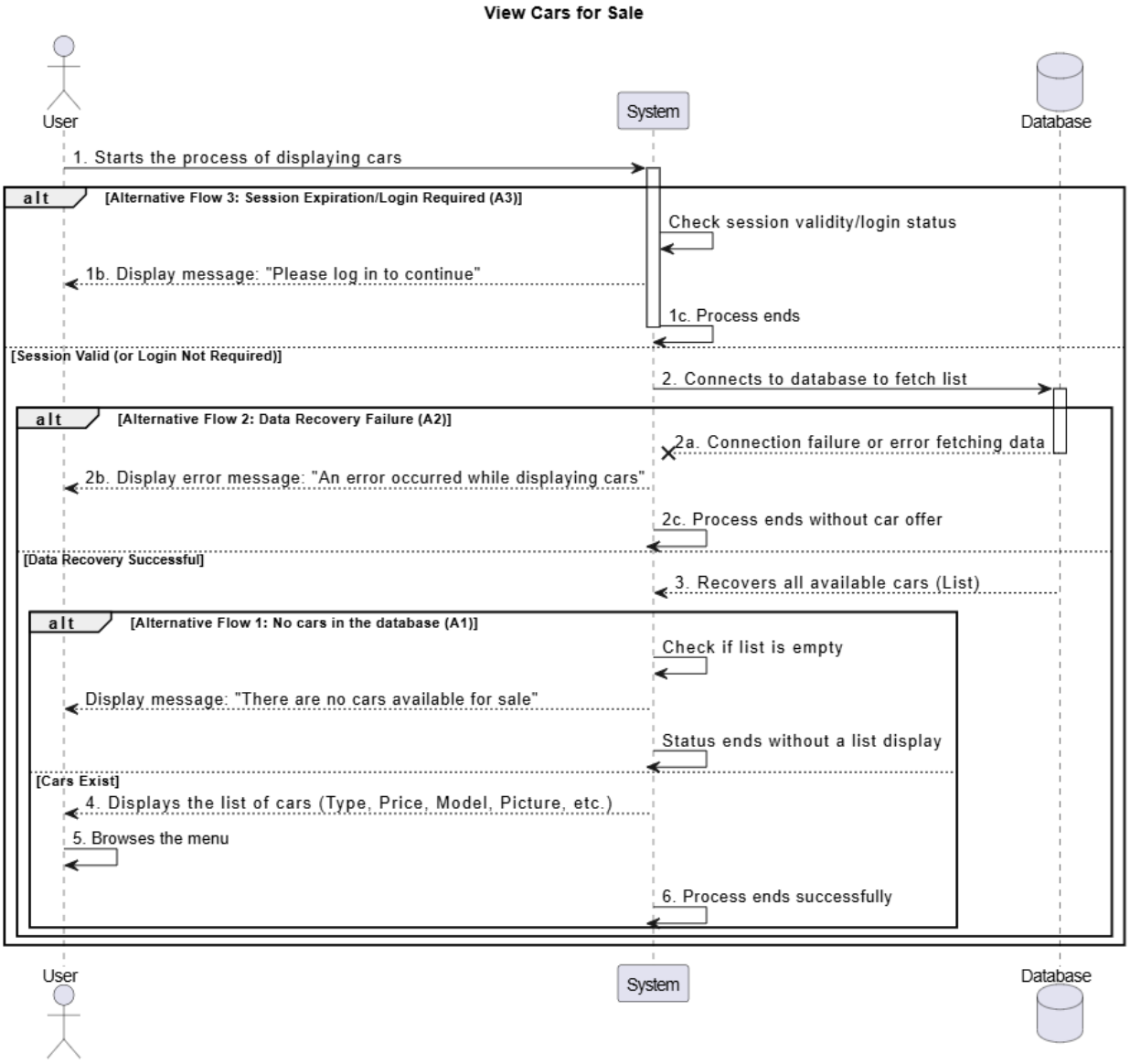

Activity Diagram: Delete Car (Simplified)
Administrator starts delete car process
System displays list of available cars
Administrator selects a car to delete
Car exists?
No
Yes
Display "Car not available"
System displays delete confirmation
Confirm deletion?
No
Yes
Display "Operation cancelled"
System requests car deletion from database
Deletion successful?
Yes
No
Display success message
Display success message
Display failure message
Display failure message

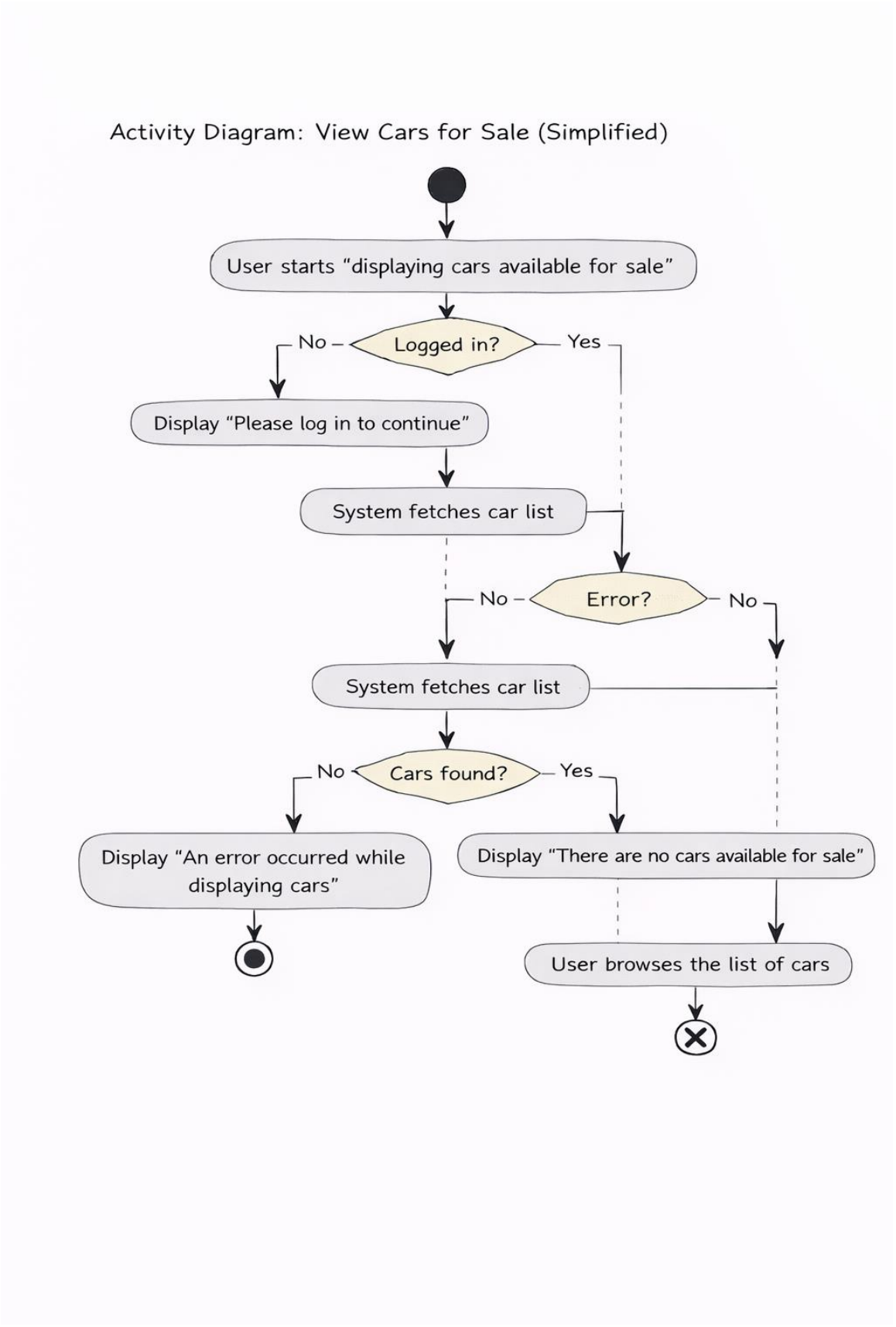

Sequence Diagram: Delete Car (Simplified)
Administrator
System
Database
1. Start car deletion process
2. Display list of available cars
3. Administrator selects a car
Car exists?
No
Yes
Display "Car not found"
Display delete confirmation
Attempt to delete car data
Deletion successful?
No
Yes
Display success message
Display failure message
Display success message
Display failure message
Administrator
Database

6.11 View cars for sale

item	Description
Use Case Name	View cars for sale
Actors	Customer , administrator
Goal	This case allows the user to view the list of cars available for sale within the system
Preconditions	The user must be able to access the system (log in if the system requires it). There must be pre-existing cars in the database (optionally, but reflected on the result)
Main Flow	1. The user starts the process of “displaying cars available for sale”. 2. The system connects to the database to bring the list of cars displayed. 3. The database recovers all available cars. 4. The system displays the list of cars, including the following information for each car: • Type (Type) • Description (Description) • Price (Price) • Model (Model) • Picture (Picture) • Year of manufacture 5. The user browses the menu. 6. The process ends successfully
Alternative Flow 1	– No cars in the database • The database does not contain any cars. • The system displays a message to the user: “There are no cars available for sale”. • The status ends without a list display.
Alternative Flow 2	– Data Recovery Failure (System Error) • The system fails to connect to the database or an error occurs while fetching the data. • The system displays an error message to the user: “An error occurred while displaying cars”. • The process ends without a car offer.
Alternative Flow 3	– Session expiration or lack of validity (optional if the system requires a login) • The user tries to access the feature but has no validity or the session has expired. • The system displays a message: “Please log in to continue”. • . The process ends.
Postconditions	Success of the operation: The list of cars available for sale is displayed to the user.

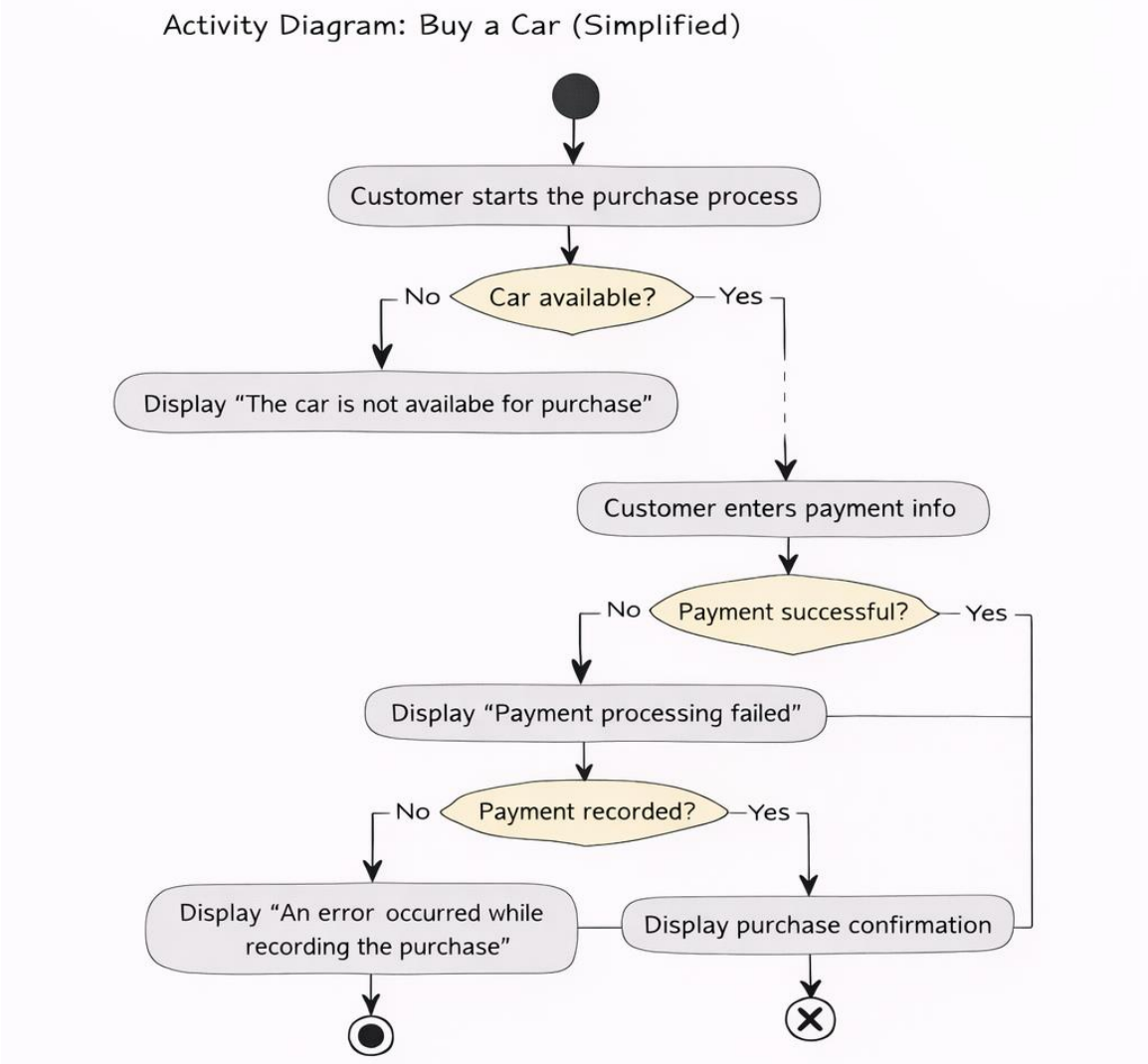
	• In the absence of cars: The system displays a message stating that there are no cars available. • Action failure: No data is displayed and the user shows a notification of a system issue

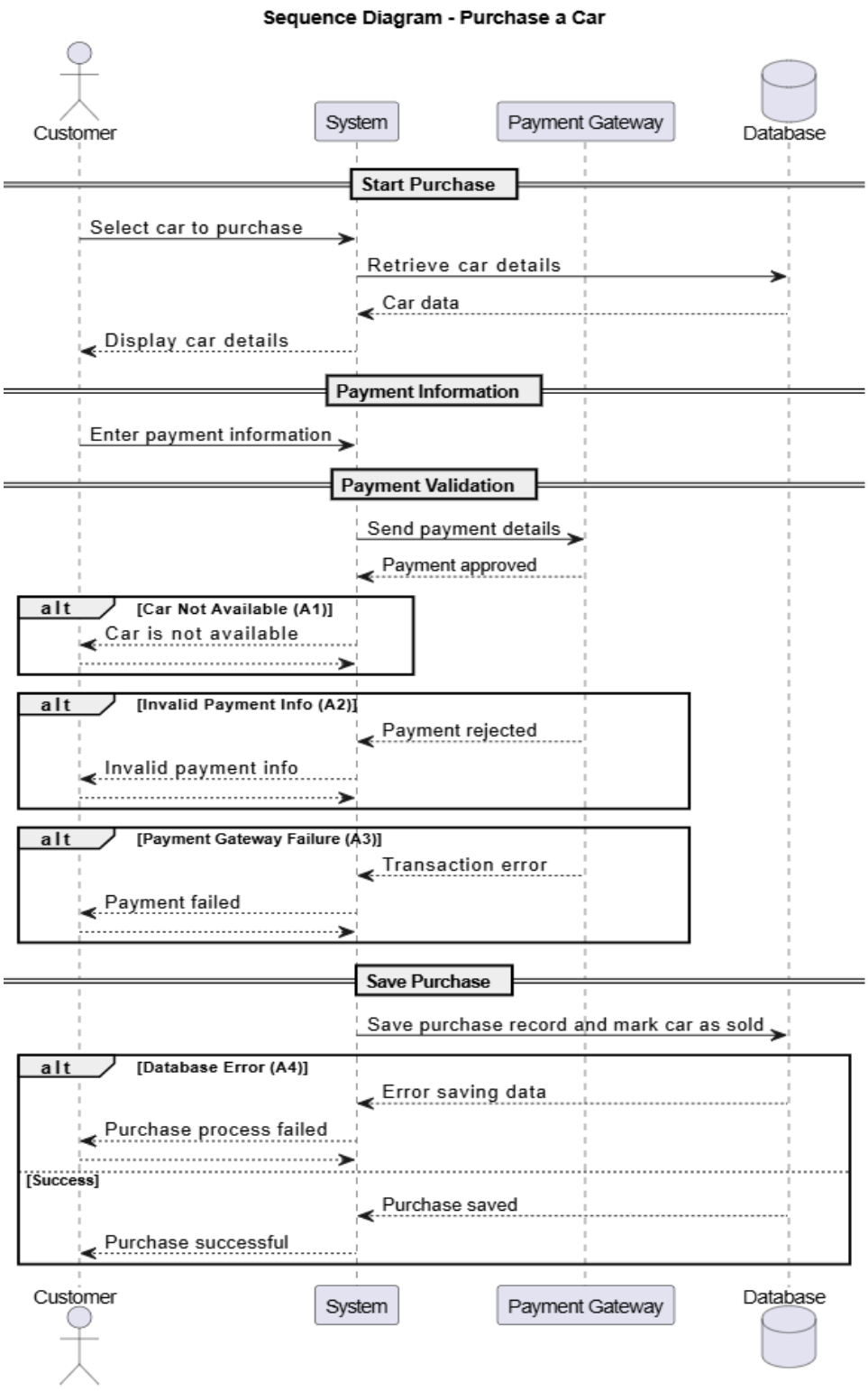



Activity Diagram: View Cars for Sale (Simplified)
User starts "displaying cars available for sale"
Logged in?
No
Yes
Display "Please log in to continue"
System fetches car list
Error?
No
No
System fetches car list
Cars found?
No
Yes
Display "An error occurred while displaying cars"
Display "There are no cars available for sale"
User browses the list of cars

Item	Description
Use Case Name	Buy a car
Actors	customer
Goal	This case allows the customer to buy a car for sale within the system.
Preconditions	• The customer must be logged in if the system requires it. • The car must be available for sale and not sold before. • The customer must have a valid payment method
Main Flow	The customer begins the process of buying a car. 1. The system displays the selected vehicle data. 2. The customer clicks on “Continue to purchase”. 3. The customer enters the payment data (example: bank card). 4. The system sends the payment data to the payment gateway for verification. 5. The payment gateway confirms the success of the payment. 6. The system records the purchase in the database. 7. The system updates the status of the car to “sold”. 8. The system displays a successful purchase confirmation message. 9. The process ends successfully
Alternative Flow 1	The car is not available / pre-sold • The system detects that the car is no longer available. • The system displays a message: “The car is not available for purchase”. • The process ends without purchase.
Alternative Flow 2	– Incorrect payment data entry • The payment gateway rejects the card data or finds an error. • The system displays an error message: “Payment information is incorrect”. • The customer returns to correct the payment information. • The system returns to step 4.
Alternative Flow 3	– Payment Gateway Failure • The payment gateway fails to perform the operation (e.g. bank refusal, technical error). • The system displays a message: “Payment processing failed”. • The process ends without purchase.
Alternative Flow 4	– Save Purchase Failed • The system tries to save the process but an error occurs in the database. • The system displays a message: “An error occurred while recording the process”. • The condition of the car is not changed. • The process ends.
Alternative Flow 5	• The customer cancels the purchase • The customer clicks on “Cancel”. • The system displays a message: “The operation has been canceled”.

	The process ends without any purchase.
Postconditions	Success of the operation: • The purchase is recorded in the system. • The condition of the car is changed to “sold”. • A confirmation message is displayed to the customer. • Operation failure: • No purchase is registered. • The car remains in its original condition (unsold)

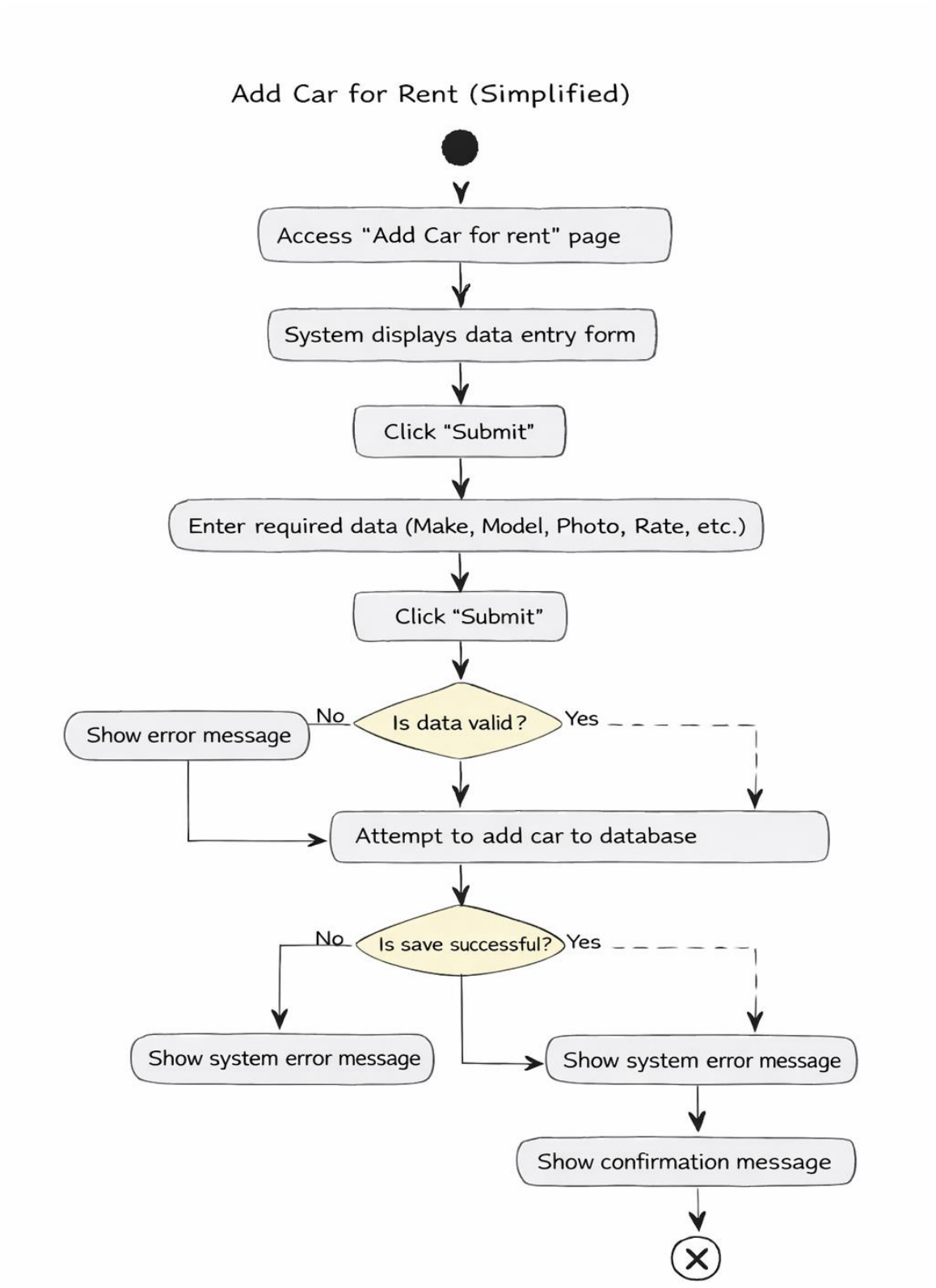

Activity Diagram: Buy a Car (Simplified)
Customer starts the purchase process
Car available?
No
Yes
Display "The car is not availabe for purchase"
Customer enters payment info
Payment successful?
No
Yes
Display "Payment processing failed"
Payment recorded?
No
Yes
Display "An error occurred while recording the purchase"
Display purchase confirmation

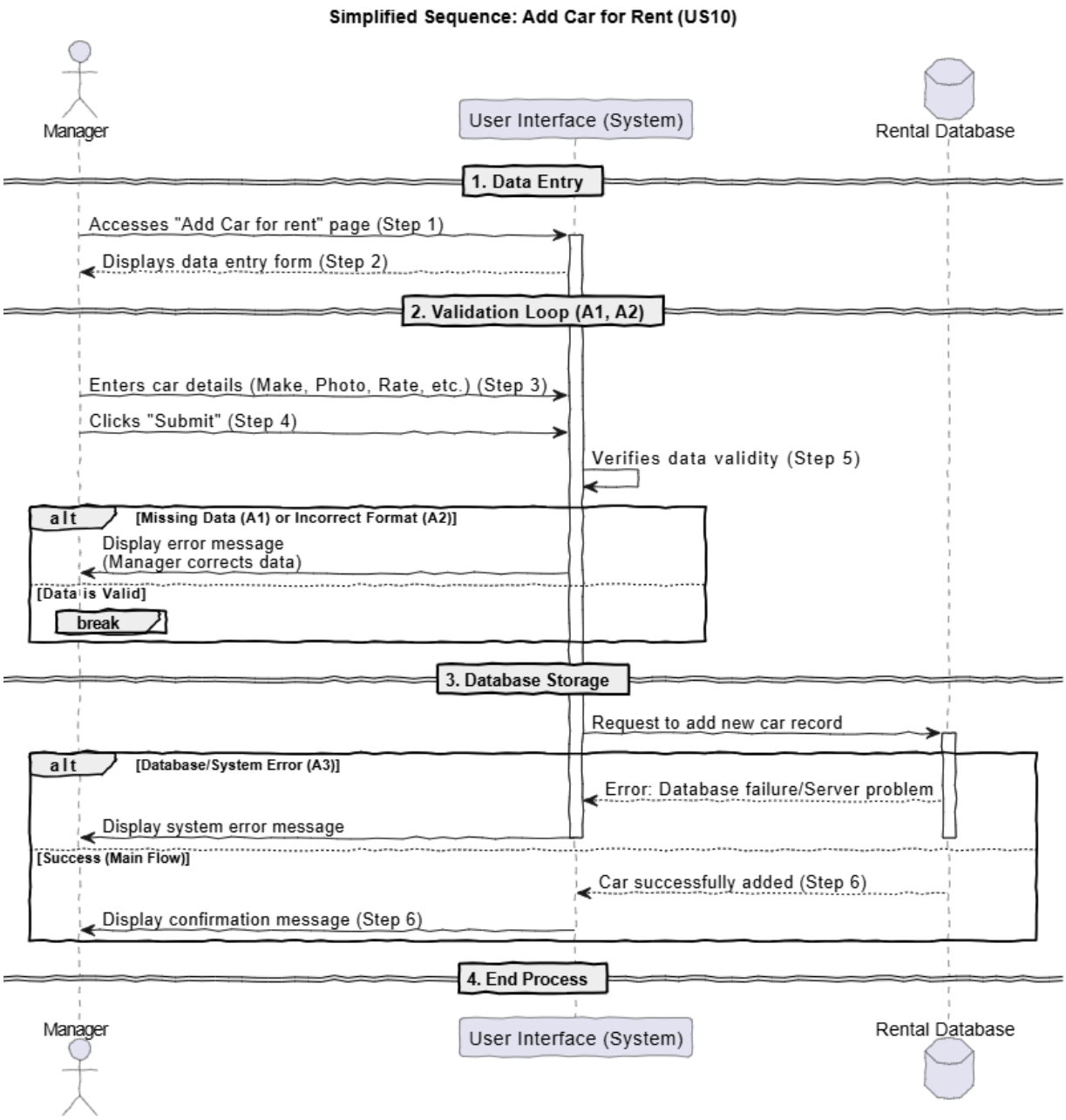

Sequence Diagram - Purchase a Car
Customer
System
Payment Gateway
Database
Start Purchase
Select car to purchase
Retrieve car details
Car data
Display car details
Payment Information
Enter payment information
Payment Validation
Send payment details
Payment approved
alt
[Car Not Available (A1)]
Car is not available
alt
[Invalid Payment Info (A2)]
Payment rejected
Invalid payment info
alt
[Payment Gateway Failure (A3)]
Transaction error
Payment failed
Save Purchase
Save purchase record and mark car as sold
alt
[Database Error (A4)]
Error saving data
Purchase process failed
[Success]
Purchase saved
Purchase successful
Customer
System
Payment Gateway
Database

6.12 Add the car for rent

Item	Description
Use Case Name	Add the car for rent
Actors	Admin
Goal	This status allows the manager to add a new car to the system's rental database.
Preconditions	The manager must have successfully logged into the system.
Main Flow	1. The manager accesses the "Add Car for rent " page in the system. 2. The system displays a form with fields for entering car details such as: make, description, model, photo, year, and daily rate. 3. The manager enters all the required data correctly. 4. The manager clicks the "Submit" button to send the data. 5. The system verifies the entered data (ensure all fields are filled out and formatted correctly). 6. If the data is correct, the system adds the car to the database and displays a confirmation message to the manager. 7. The car is now available for rent and appears in the list of available cars.
Alternative Flow 1	Missing Data If the manager enters incomplete data (such as missing the daily price or model), the system displays an error message indicating the missing fields. The manager enters the missing data and resubmits the form. The system verifies the entered data and adds the vehicle if the data is complete
Alternative Flow 2	Incorrect Data Format (such as entering a non-numeric price) If the manager enters incorrect data (such as entering a non-numeric value in the daily price field), the system displays an error message. The manager corrects the incorrect data and resubmits the form. The system verifies the entered data and adds the vehicle if the data is correct.

Alternative Flow 3	System Error If a system error occurs (such as a database failure or a server problem), the system displays an error message
Postconditions	The car is added to the rental database. The car becomes available for rent by customers. A confirmation message is displayed to the manager indicating that the car has been successfully added.

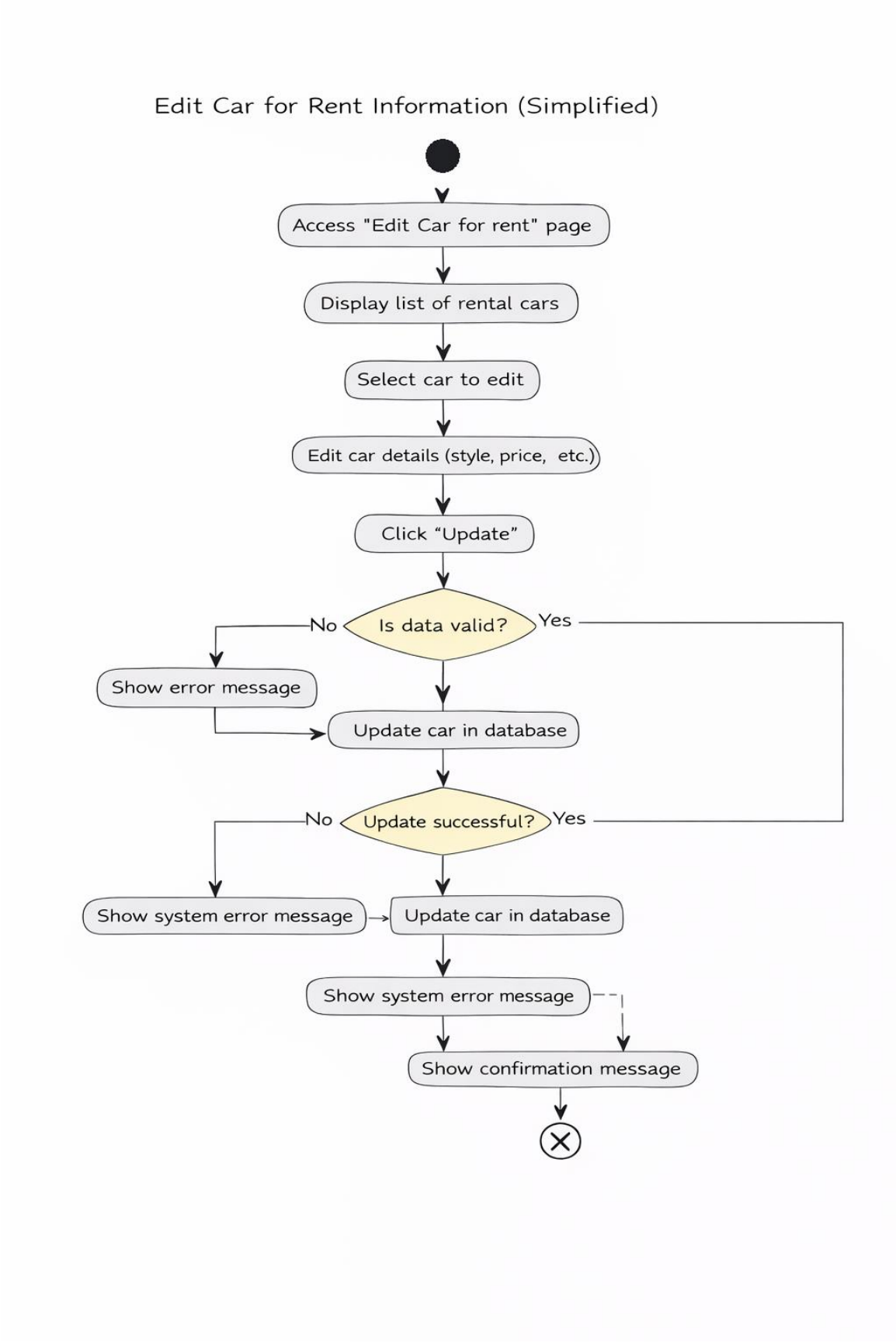

Add Car for Rent (Simplified)
Access "Add Car for rent" page
System displays data entry form
Click "Submit"
Enter required data (Make, Model, Photo, Rate, etc.)
Click "Submit"
Is data valid?
No
Yes
Show error message
Attempt to add car to database
Is save successful?
No
Yes
Show system error message
Show system error message
Show confirmation message

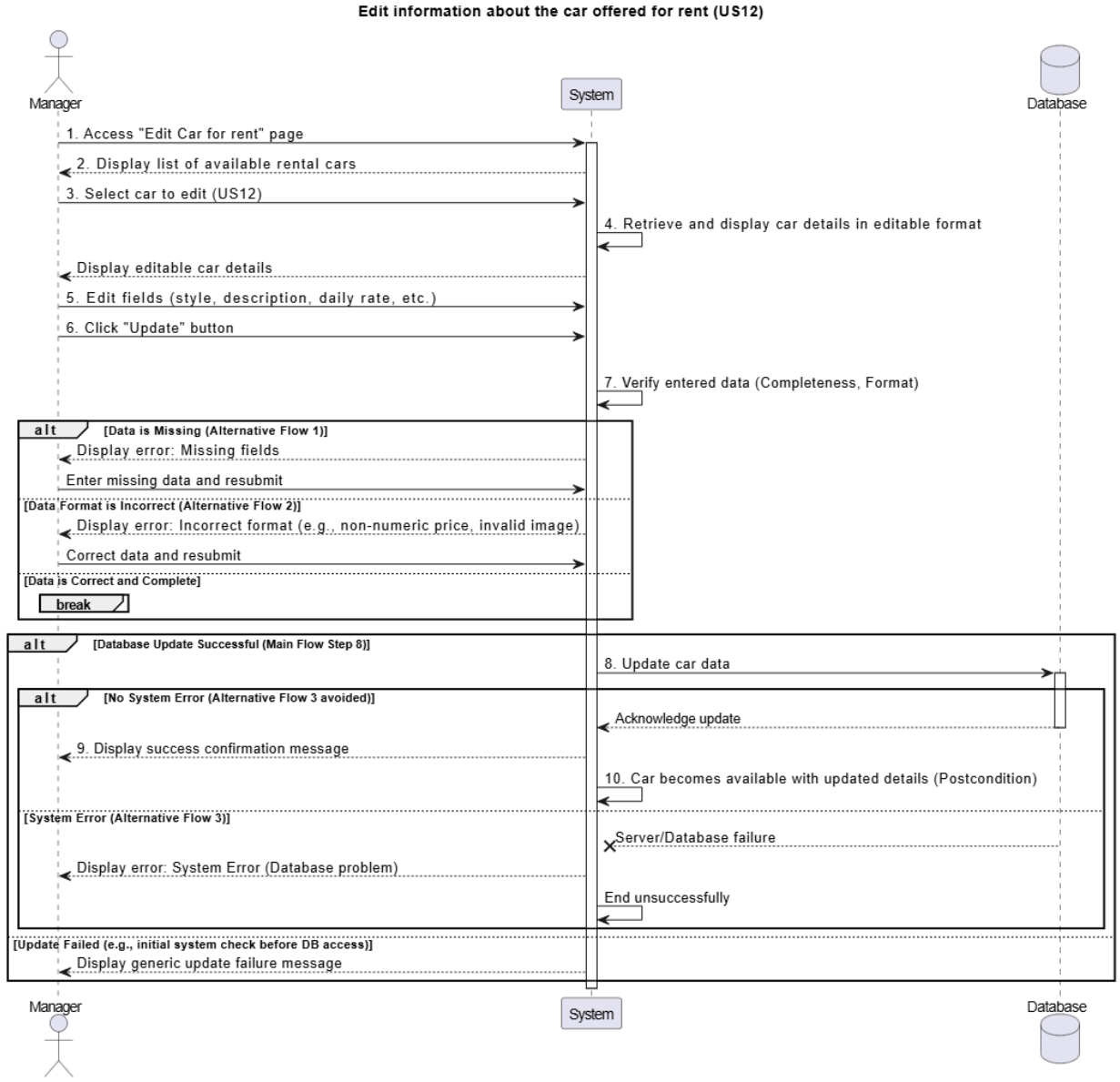

Simplified Sequence: Add Car for Rent (US10)
Manager
User Interface (System)
Rental Database
1. Data Entry
Accesses "Add Car for rent" page (Step 1)
Displays data entry form (Step 2)
2. Validation Loop (A1, A2)
Enters car details (Make, Photo, Rate, etc.) (Step 3)
Clicks "Submit" (Step 4)
Verifies data validity (Step 5)
alt
[Missing Data (A1) or Incorrect Format (A2)]
Display error message
(Manager corrects data)
[Data is Valid]
break
3. Database Storage
Request to add new car record
alt
[Database/System Error (A3)]
Error: Database failure/Server problem
Display system error message
[Success (Main Flow)]
Car successfully added (Step 6)
Display confirmation message (Step 6)
4. End Process
Manager
User Interface (System)
Rental Database

6.13 Edit information about the car offered for rent

item	Description
Use Case Name	Edit information about the car offered for rent
Actors	admin
Goal	This feature allows the manager to modify previously added vehicle rental information.
Preconditions	The manager must be registered in the system. The manager must have previously added the vehicle to the system. The vehicle must already exist in the system to be edited. All data available for editing must be accurate and valid.
Main Flow	1. The manager accesses the "Edit Car for rent" page in the system. 2. The system displays a list of available rental cars. 3. The manager selects the car they want to edit from the list. 4. The system displays the selected car's details (style, description, model, photo, year, daily rate) in an editable format. 5. The manager edits any of the listed fields. 6. After completing the edits, the manager clicks the "Update" button. 7. The system verifies the entered data (such as ensuring the daily rate is a correct numerical value and that the photo format is correct). 8. If the data is correct, the system updates the car in the database. 9. The system displays a confirmation message to the manager indicating that the edits were successful. 10. The car becomes available for rent with the updated details.
Alternative Flow 1	Missing Data If the manager enters incomplete data (such as missing the daily price or model), the system displays an error message indicating the missing fields. The manager enters the missing data and resubmits the form. The system verifies the entered data and updates the vehicle if the data is complete.
Alternative Flow 2	Incorrect Data Format (such as entering a non-numeric price or an invalid image)

	If the manager enters incorrect data (such as entering a non-numeric value in the daily price field or an image in an unsupported format), the system displays an error message. The manager corrects the incorrect data (e.g., entering the price correctly or uploading an image in the correct format) and resubmits the form. The system verifies the entered data and updates the vehicle if the data is correct
Alternative Flow 3	System Error If a system error occurs (such as a database problem or server malfunction), the system displays an error message. The manager can try again after resolving the issue or contact technical support if necessary.
Postconditions	The vehicle's data is updated in the system database. A confirmation message is displayed to the manager indicating that the modifications were successful. The vehicle remains available for rental with the updated data.

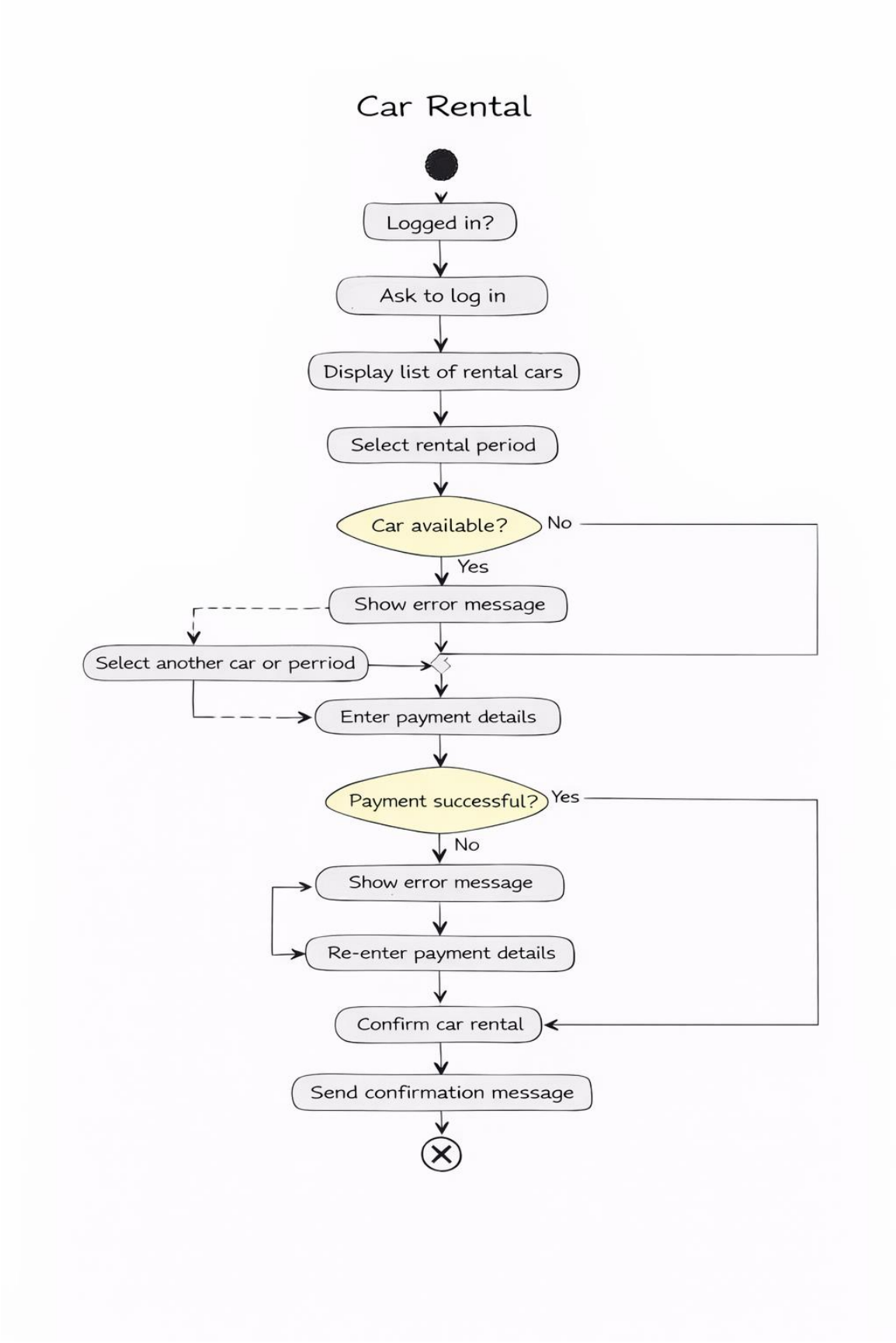

Edit Car for Rent Information (Simplified)
Access "Edit Car for rent" page
Display list of rental cars
Select car to edit
Edit car details (style, price, etc.)
Click "Update"
Is data valid?
No
Yes
Show error message
Update car in database
Update successful?
No
Yes
Show system error message
Update car in database
Show system error message
Show confirmation message

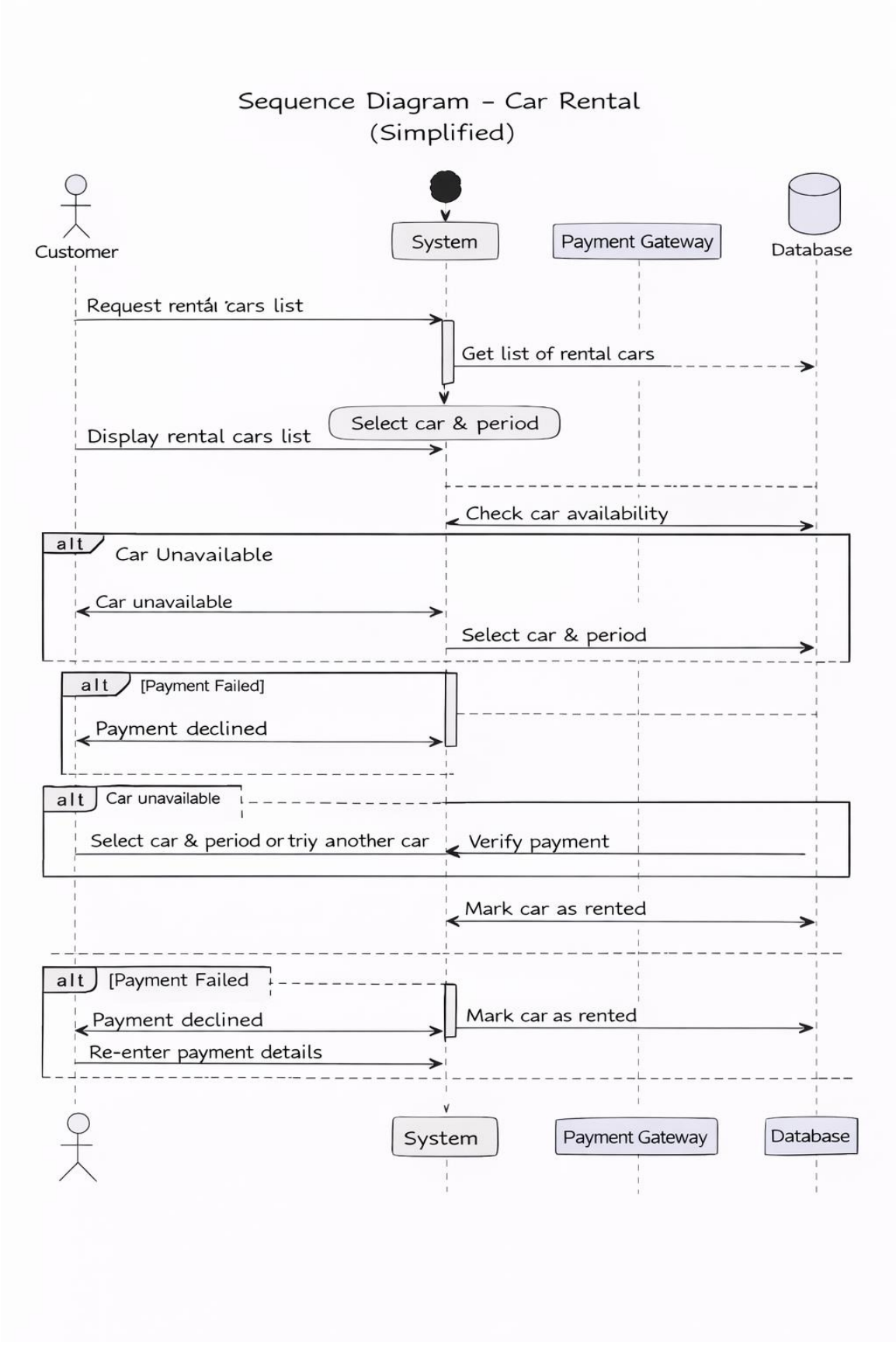

Edit information about the car offered for rent (US12)
Manager
System
Database
1. Access "Edit Car for rent" page
2. Display list of available rental cars
3. Select car to edit (US12)
4. Retrieve and display car details in editable format
Display editable car details
5. Edit fields (style, description, daily rate, etc.)
6. Click "Update" button
7. Verify entered data (Completeness, Format)
alt
[Data is Missing (Alternative Flow 1)]
Display error: Missing fields
Enter missing data and resubmit
[Data Format is Incorrect (Alternative Flow 2)]
Display error: Incorrect format (e.g., non-numeric price, invalid image)
Correct data and resubmit
[Data is Correct and Complete]
break
alt
[Database Update Successful (Main Flow Step 8)]
8. Update car data
alt
[No System Error (Alternative Flow 3 avoided)]
Acknowledge update
9. Display success confirmation message
10. Car becomes available with updated details (Postcondition)
[System Error (Alternative Flow 3)]
Server/Database failure
Display error: System Error (Database problem)
End unsuccessfully
[Update Failed (e.g., initial system check before DB access)]
Display generic update failure message
Manager
System
Database

6.14 Car rental

Item	Description
Use Case Name	Car rental
Actors	customer
Goal	This status allows the customer to rent a car from the system.
Preconditions	The customer must be registered in the system and have an active account. The customer must be logged into their account in the system. The requested vehicle must be available for rent at the specified time. The rental period must be predetermined (start and end dates). The customer must have a valid payment method (such as a credit card, digital wallet, etc.).
Main Flow	1. The work requires the system to display the cars available for rent 2. The system displays a list of available rental cars. 3. The customer selects the car they wish to rent. 4. The system displays car details (make, model, daily rate, etc.) and rental period options (start and end dates). 5. The customer selects the rental period and agrees to the rental terms. 6. The customer clicks the "Rent Now" button. 7. 7.The system checks car availability for the selected rental period. 8. If the car is available, the system displays payment information (such as credit card details or other payment methods). 9. The customer enters their payment details. 10. The system verifies the payment method and deducts the amount due. 11. If the transaction is successful, the system confirms the car rental. 12. The car's status in the system is updated to "Rented" for the selected period. 13. A confirmation message is sent to the customer containing the rental details (start and end dates, car type, and amount paid).
Alternative Flow 1	Car unavailable during the requested rental period If the car is unavailable during the specified rental period (for example, it has been booked by another customer), the system displays an unavailability message.

	The customer can select another available car or change the rental period. The customer reselects the car and rental period. The system checks the availability of the new car and displays payment options.
Alternative Flow 2	Payment failed (e.g., card declined or insufficient funds) If the payment fails (e.g., card declined or insufficient funds), the system displays an error message. The customer can try using another payment method. The customer re-enters the payment details. If the payment is successfully accepted, the system confirms the rental.
Alternative Flow 3	Incorrect Data Entry (e.g., Illogical Dates or Incorrect Credit Card Information) If the customer enters incorrect data (e.g., a rental start date before today's date or incorrect credit card information), the system displays an error message explaining the problem. The customer corrects the data and tries again. After verifying the data, the system confirms the rental.
Alternative Flow 4	Login Failure or Inactive Account If the customer has not logged in or their account is inactive, the system displays a message asking the customer to log in or activate their account. The customer logs in or activates their account and tries again.
Postconditions	The car is reserved for the customer during the specified rental period. The car's status in the system is updated to "Rented." A rental confirmation is sent to the customer via email or an in-system notification. The payment due is deducted from the customer's chosen payment method.

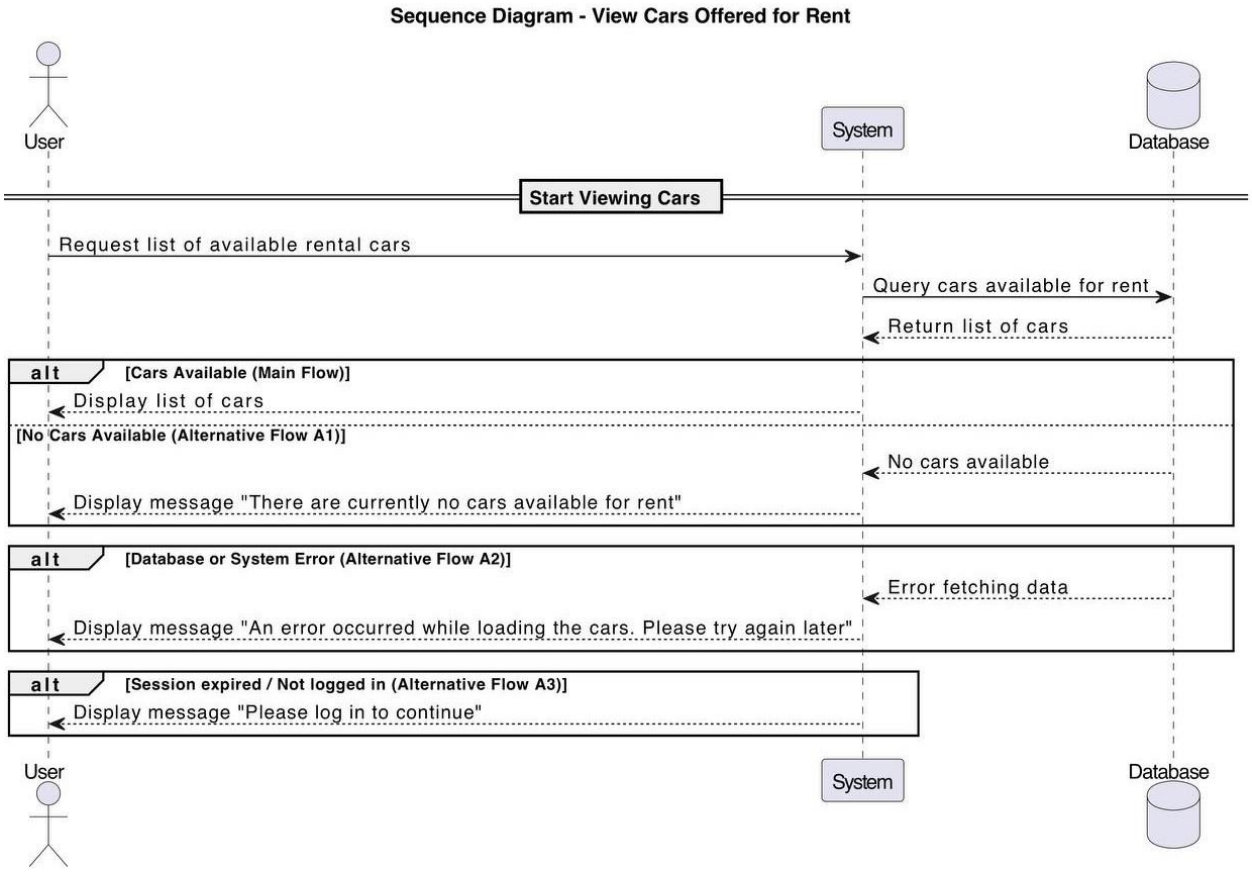

Car Rental
Logged in?
Ask to log in
Display list of rental cars
Select rental period
Car available?
No
Yes
Show error message
Select another car or perriod
Enter payment details
Payment successful?
Yes
No
Show error message
Re-enter payment details
Confirm car rental
Send confirmation message

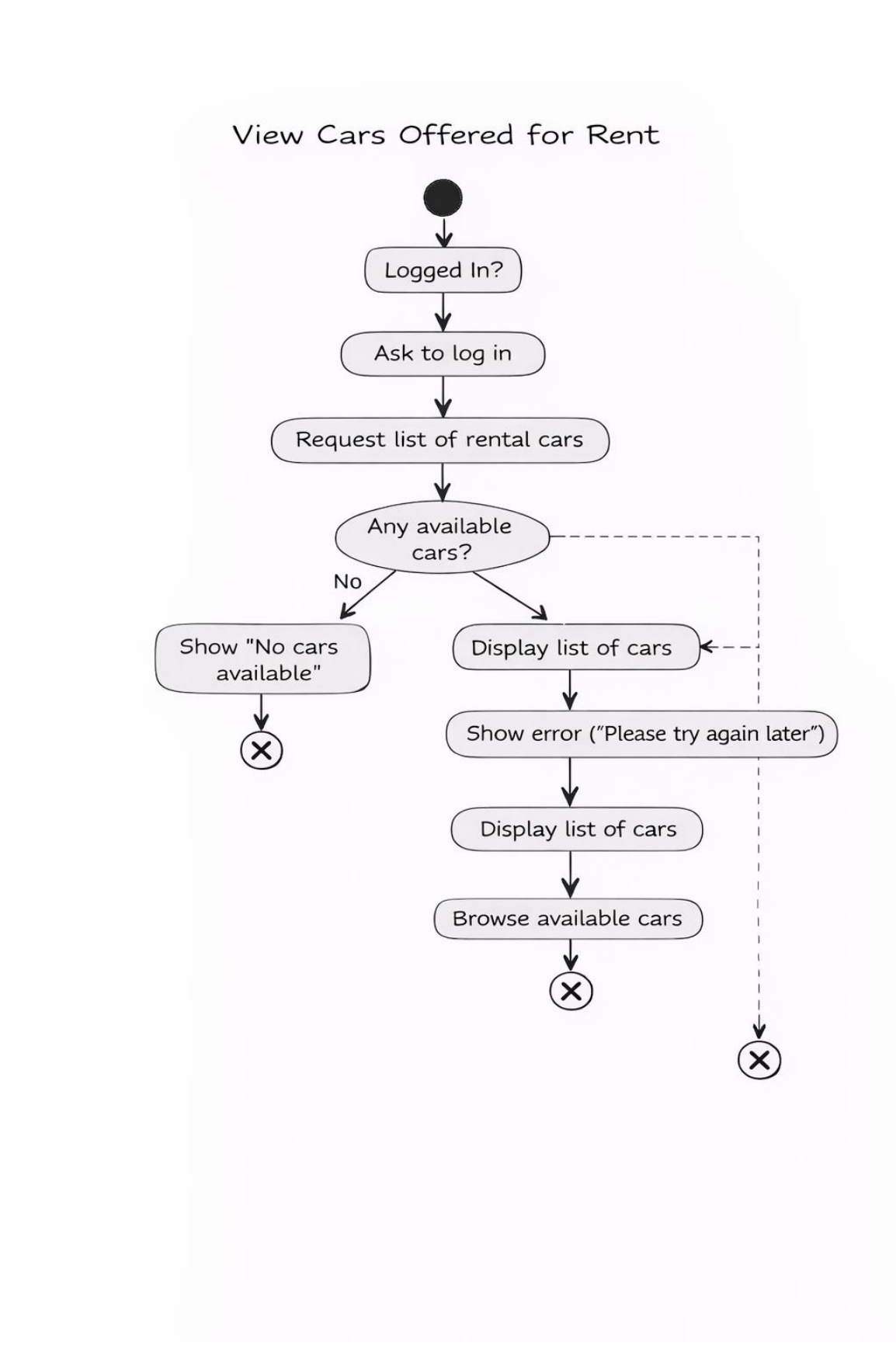

Sequence Diagram – Car Rental
(Simplified)
Customer
System
Payment Gateway
Database
Request rental cars list
Get list of rental cars
Select car & period
Display rental cars list
Check car availability
alt
Car Unavailable
Car unavailable
Select car & period
alt
[Payment Failed]
Payment declined
alt
Car unavailable
Select car & period or triy another car
Verify payment
Mark car as rented
alt
[Payment Failed
Payment declined
Mark car as rented
Re-enter payment details
System
Payment Gateway
Database

6.15 View cars offered for rent

item	Description
Use Case Name	View cars offered for rent
Actors	Customer , administrator
Goal	This case allows the user to view the list of cars available for rent within the system.
Preconditions	• The user must be able to access the system (log in if the system requires it). • The database must contain cars for rent (optional but affects the result)
Main Flow	1. The user starts the process of “Viewing Cars Available for Rent”. 2. The system sends a request to the database to bring the available cars. 3. The database returns a list of all cars available for rent. 4. The system displays the list of cars including the following details: • Type (Type) • Description (Description) • Daily Rental Price (Rental Price) • Model (Model) • Picture (Picture) • Year of manufacture 5. The user browses the available cars. 6. The process ends successfully
Alternative Flow 1	– No cars available for rent • The database returns without available cars. • The system displays a message to the user: • “There are currently no cars available for rent.” • The process ends without displaying a list of cars.
Alternative Flow 2	– Failure to retrieve data (system or database error) • The system fails to connect to the database or malfunction occurs during fetching. • The system displays an error message to the user: • “An error occurred while loading the cars. Please try again later.” • The process ends without success.
Alternative Flow 3	– Session expiration or lack of validity (if the system requires a login) • The user tries to access the feature without logging in or after the session ends. • The system displays a message: • “Please log in to continue.” • The process ends.

Postconditions	Success of the operation: Show the user the list of cars available for rent. • In the absence of cars: The system displays a message stating that there are no cars available for rent. • If the action fails: An error message is displayed to the user and the list does not appear

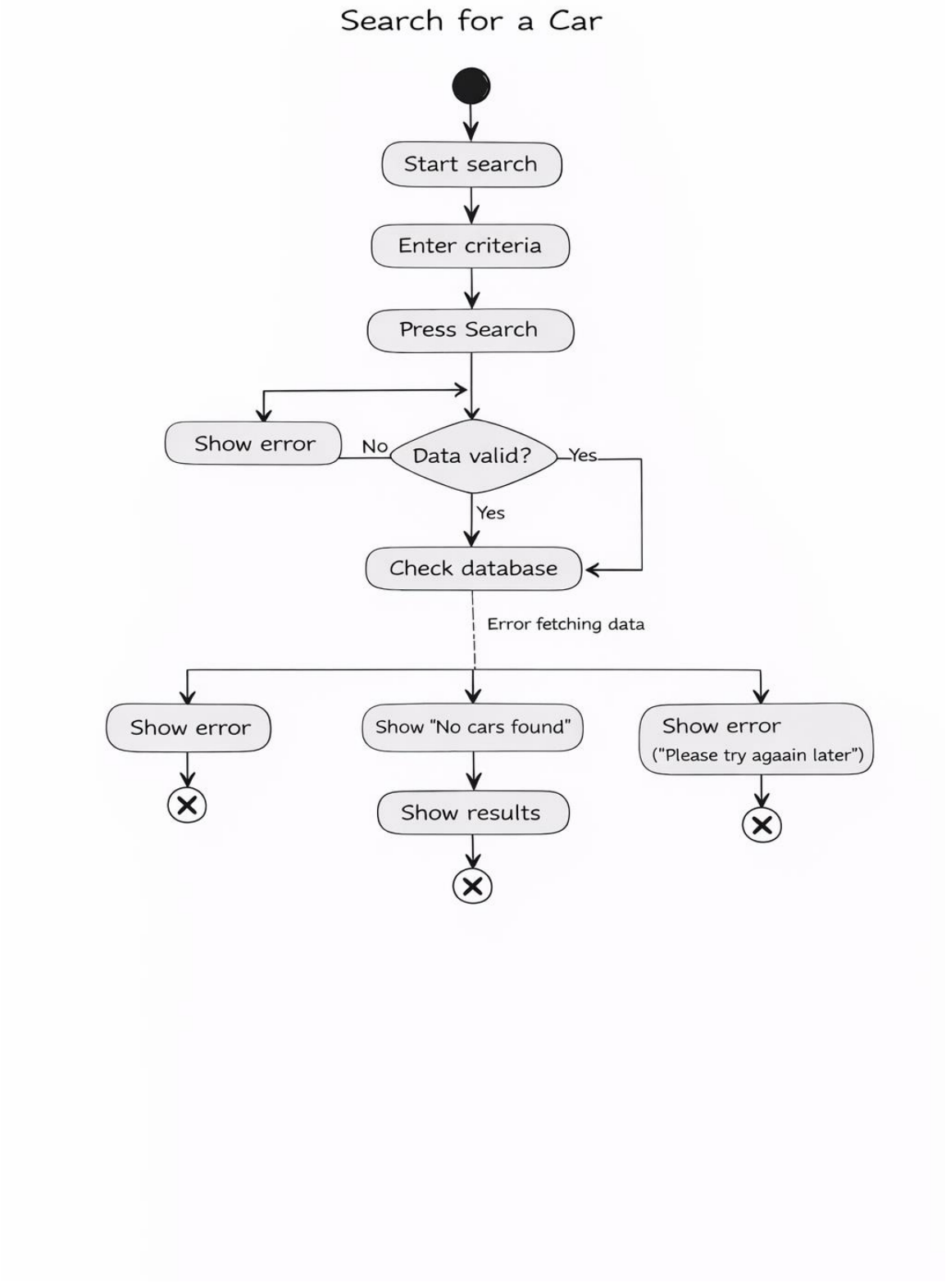


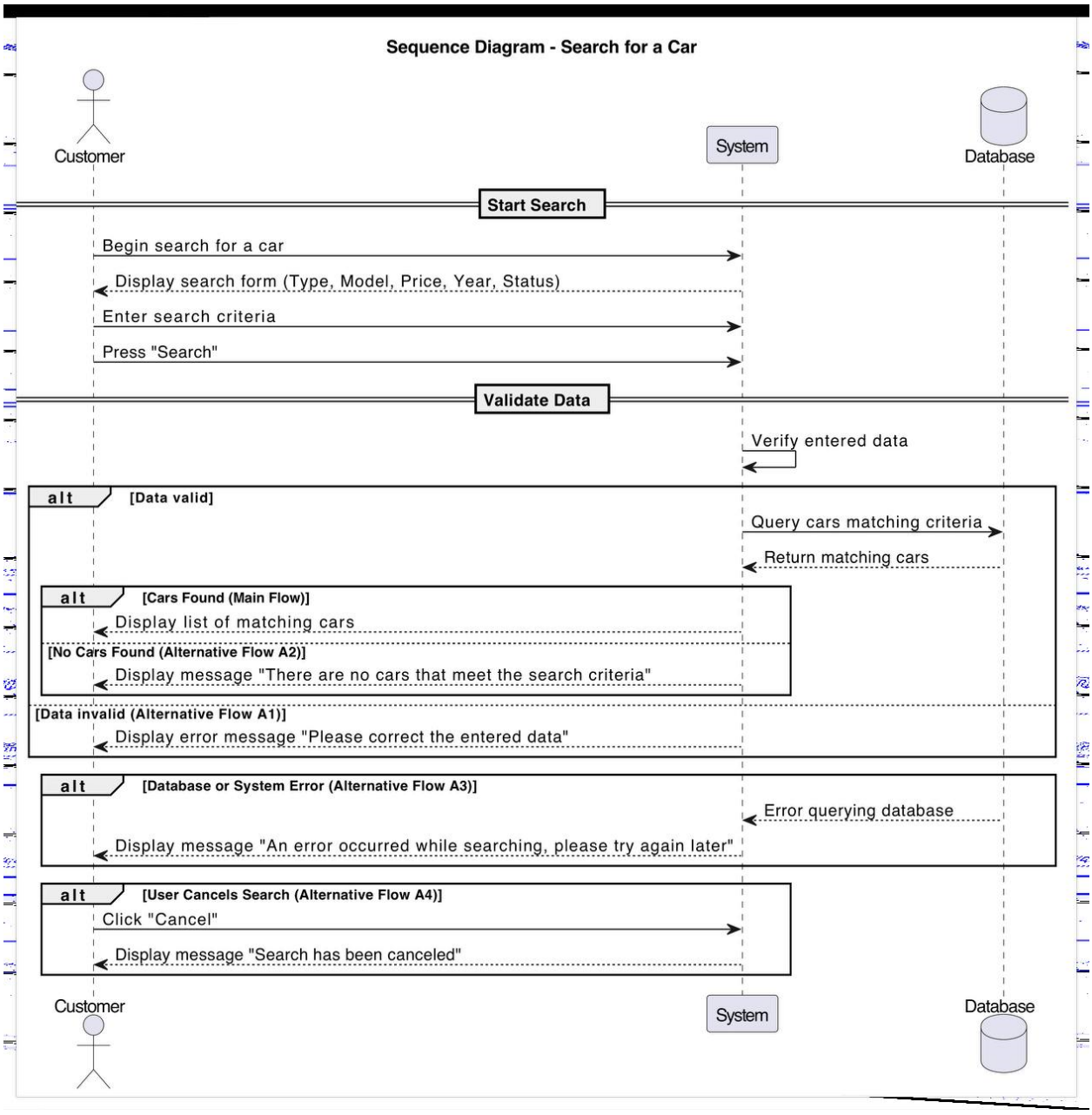

View Cars Offered for Rent
Logged In?
Ask to log in
Request list of rental cars
Any available cars?
No
Show "No cars available"
Display list of cars
Show error ("Please try again later")
Display list of cars
Browse available cars

6.16 Search for a car

Item	Description
Use Case Name	Search for a car
Actors	Customer
Goal	This case enables the user (Customer) to search for specific cars in the system, whether for purchase or rent.
Preconditions	The user must be able to access the system (log in if the system requires it). • The database must contain cars to search for
Main Flow	1. The user begins the process of searching for a car. 2. The system displays the search model and requires the entry of search criteria, such as: • Type (Type) • Model (Model) • Price (Price) • Year of manufacture • Status (For Sale/Rent) 3. The user enters the required criteria. 4. The user presses the “Search” button. 5. The system verifies the validity of the entered data (for example, making sure that the price is digital). 6. The system informs the database about matching cars. 7. The results are presented to the user in the form of a list of cars with information: • Type, Model, Price, Year of Manufacture, Picture, Car Description 8. The process ends successfully
Alternative Flow 1	Entering incorrect search criteria • The system detects a data error (such as entering text in the price field). • The system displays an error message: “Please correct the entered data”. • The user returns to modify the data. • The system returns to step 4 in the main flow.
Alternative Flow 2	– No cars matching the search • The database returns without results. • The system displays a message: “There are no cars that meet the search criteria”. • The process ends without displaying a list of cars.

Alternative Flow 3	– Database connection failure or system error • An error occurs while a database query. • The system displays a message: “An error occurred while searching, please try again later”. • The process ends without displaying results.
Alternative Flow 4	– User cancels search • The user clicks on “Cancel”. • The system displays the message: “Search has been canceled”. • The process ends without displaying results.
Postconditions	Success of the process: The list of cars that meet the search criteria is displayed. • No matching results: The system displays a message stating that there are no matching cars. • Operation failure: The user shows an error message without displaying any results

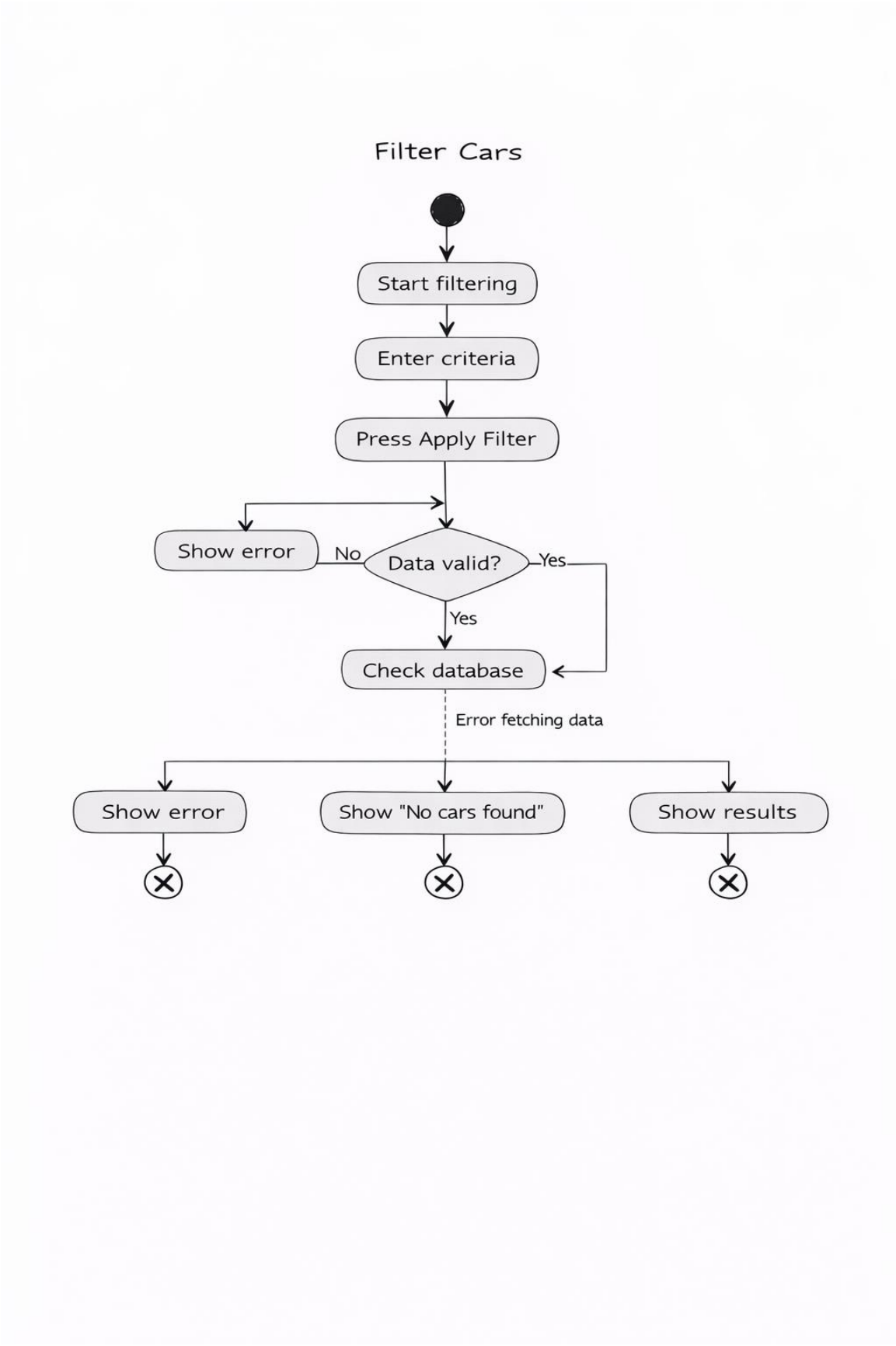

Search for a Car
Start search
Enter criteria
Press Search
Show error
No
Data valid?
Yes
Yes
Check database
Error fetching data
Show error
Show "No cars found"
Show error
("Please try agaain later")
Show results

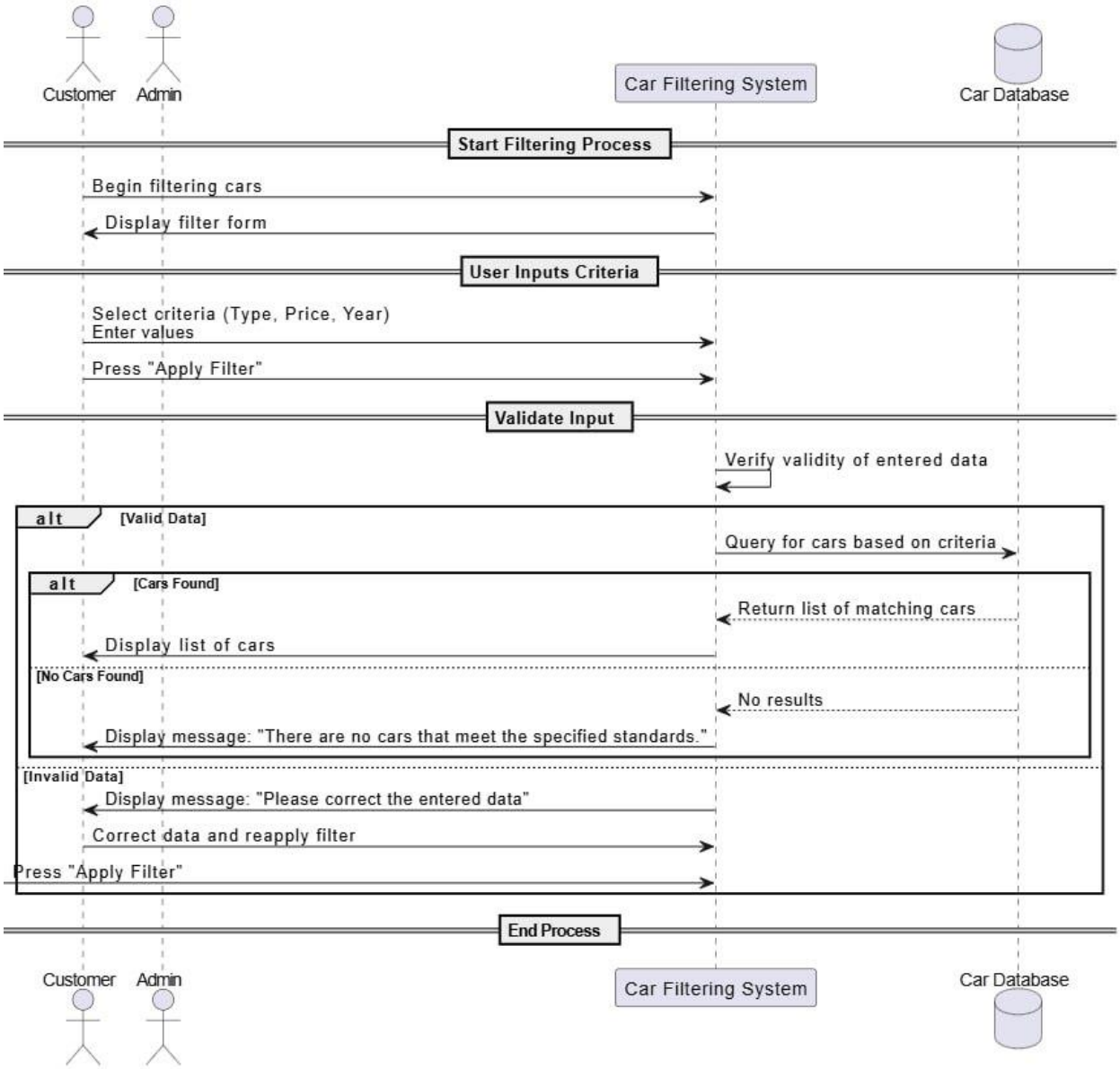

Sequence Diagram - Search for a Car
Customer
System
Database
Start Search
Begin search for a car
Display search form (Type, Model, Price, Year, Status)
Enter search criteria
Press "Search"
Validate Data
Verify entered data
alt
[Data valid]
Query cars matching criteria
Return matching cars
alt
[Cars Found (Main Flow)]
Display list of matching cars
[No Cars Found (Alternative Flow A2)]
Display message "There are no cars that meet the search criteria"
[Data invalid (Alternative Flow A1)]
Display error message "Please correct the entered data"
alt
[Database or System Error (Alternative Flow A3)]
Error querying database
Display message "An error occurred while searching, please try again later"
alt
[User Cancels Search (Alternative Flow A4)]
Click "Cancel"
Display message "Search has been canceled"
Customer
System
Database

6.17 Filters cars

Item	Description
Use Case Name	Filters cars
Actors	Customer , admin
Goal	This condition allows the user to filter the cars displayed in the system based on specific criteria.
Preconditions	The user must be able to access the system (log in if the system requires it). • The database must contain cars to filter the data based on it
Main Flow	1. The user begins the process of filtering cars. 2. The system displays the filter form and asks the user to specify which of the following criteria: • Type (Type) • Price (Price) • Year of manufacture 3. The user selects the required criteria and enters the appropriate values. 4. The user presses the “Apply Filter” button. 5. The system verifies the validity of the entered data (for example, the price is digital and the year is correct). 6. The system informs the database to return the vehicles that meet the selected criteria. 7. The system displays the results to the user in the form of a car list, including the basic information for each car: type, price, model, image, and year of manufacture. 9. The process ends successfully
Alternative Flow 1	– Entering invalid data • The system detects an error in the input values (such as text in the price field or incorrect year). • The system displays an error message: “Please correct the entered data”. • The user returns to correct the data. • The system returns to step 4 in the main flow.
Alternative Flow 2	– No standard-compliant cars • The database returns without results consistent with the selected criteria. • The system displays a message: “There are no cars that meet the specified standards.” • The process ends without displaying a list of cars.
Alternative Flow 3	– Data Recovery Failure/System Error • An error occurs while a database query.

	• The system displays a message: “An error occurred while filtering, please try again later.” • The process ends without showing the results.
Postconditions	Operation success: The user shows the list of cars that meet the filter criteria. • No matching results: The system displays a message stating that there are no matching cars. • Operation failure: The user shows an error message without displaying any results


Filter Cars
Start filtering
Enter criteria
Press Apply Filter
Show error
No
Data valid?
Yes
Yes
Check database
Error fetching data
Show error
Show "No cars found"
Show results


Customer
Admin
Car Filtering System
Car Database
Start Filtering Process
Begin filtering cars
Display filter form
User Inputs Criteria
Select criteria (Type, Price, Year)
Enter values
Press "Apply Filter"
Validate Input
Verify validity of entered data
alt
[Valid Data]
Query for cars based on criteria
alt
[Cars Found]
Return list of matching cars
Display list of cars
[No Cars Found]
No results
Display message: "There are no cars that meet the specified standards."
[Invalid Data]
Display message: "Please correct the entered data"
Correct data and reapply filter
Press "Apply Filter"
End Process
Customer
Admin
Car Filtering System
Car Database

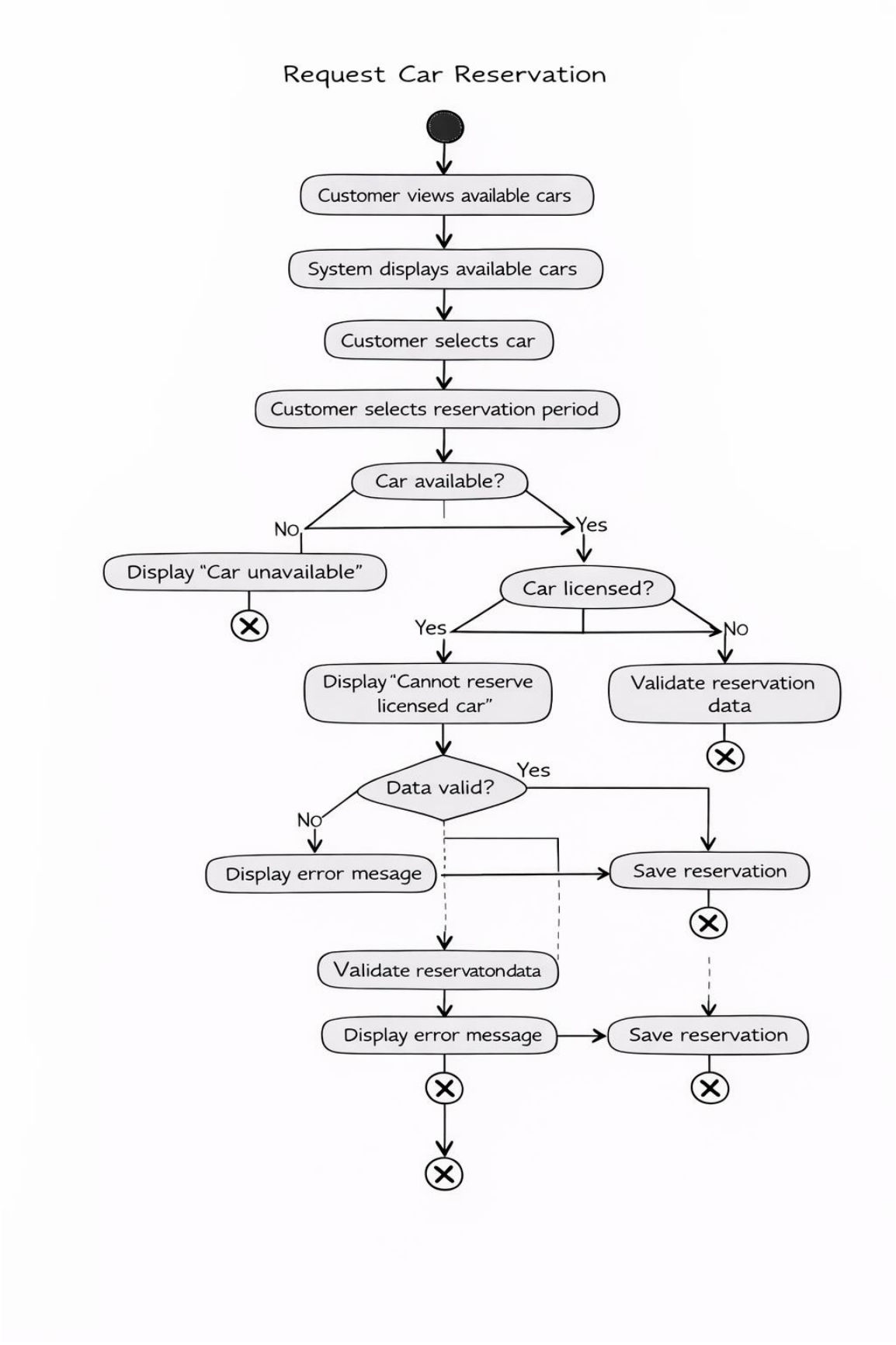
6.18 Request car reservation

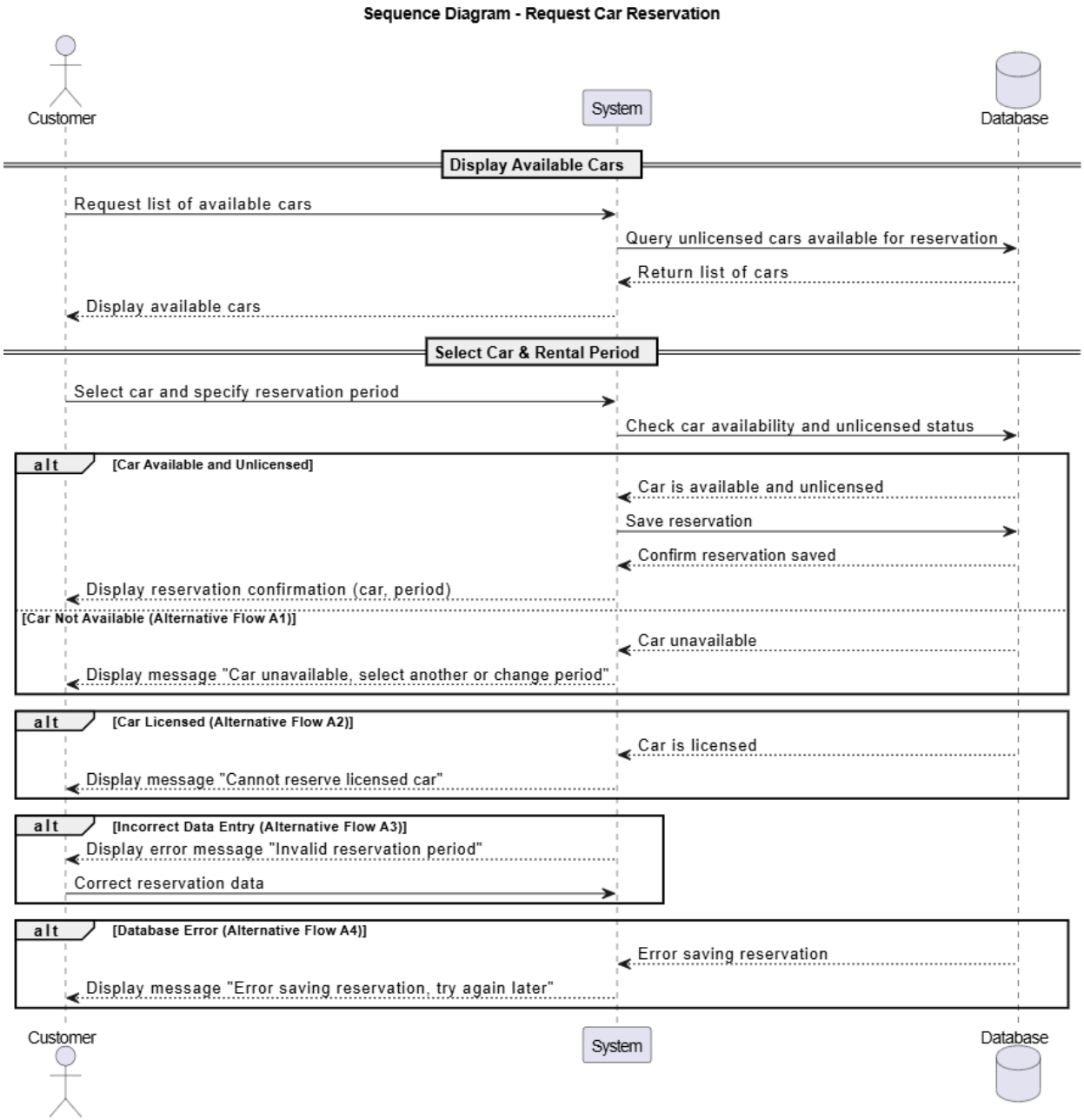
item	Description
Use Case Name	Request car reservation
Actors	Customer
Goal	This case enables the user to request a reservation of a car available in the system
Preconditions	• The user must be able to access the system (log in if the system requires it). • The car must be available for reservation. • The vehicle must be unlicensed. • The reservation period must be determined by the user (Start Date and End Date)

Main Flow	1. The user initiates a car reservation request.
	2. The system displays the list of cars available for booking.
	3. The user chooses the car to be booked.
	4. The system displays the details of the car (Type, Model, Year, Status).
	5. The user determines the reservation period (Start Date and End Date).
	6. The user clicks on the “Book Now” button.
	7. The system checks the booking conditions:
	  • The car is available for reservation.
	  • The car is not licensed.
	8. If the conditions are met, the system records the reservation in the database.
	9. The condition of the vehicle is updated to “Booked” during the booking period.
	10. The system displays a booking confirmation message to the user with the details of the car and the booking period.
	11. The process ends successfully

Alternative Flow 1	– The car is not available for booking • The system detects that the car is pre-reserved or unavailable. • The system displays a message: “The car is not available for booking in the specified period”. • The user can choose another car or modify the booking period. • The system returns to step 3 in the main flow
Alternative Flow 2	Licensed car (Licensed) • The system detects that the car is licensed and not eligible for reservation. • The system displays a message: “This car cannot be reserved because it is licensed.” • . The user can choose another unlicensed car.

	• The system returns to step 3.
Alternative Flow 3	– Incorrect data entry (e.g. illogical dates) • The user enters incorrect booking dates (Start Date after End Date or Preday Date). • The system displays an error message: “Please enter a valid booking period”. • He user returns to modify the data. • The system returns to step 5.
Alternative Flow 3	– Failed to save the reservation in the database • An error occurs while registering a reservation in the database. • The system displays a message: “An error occurred while registering the reservation, please try again later.” • The status of the vehicle is not updated. • The process ends without success.
Postconditions	Success of the operation: • The reservation is recorded in the database and the status of the vehicle is updated to “Booked”. • Operation failure: No reservation is registered and the vehicle remains available

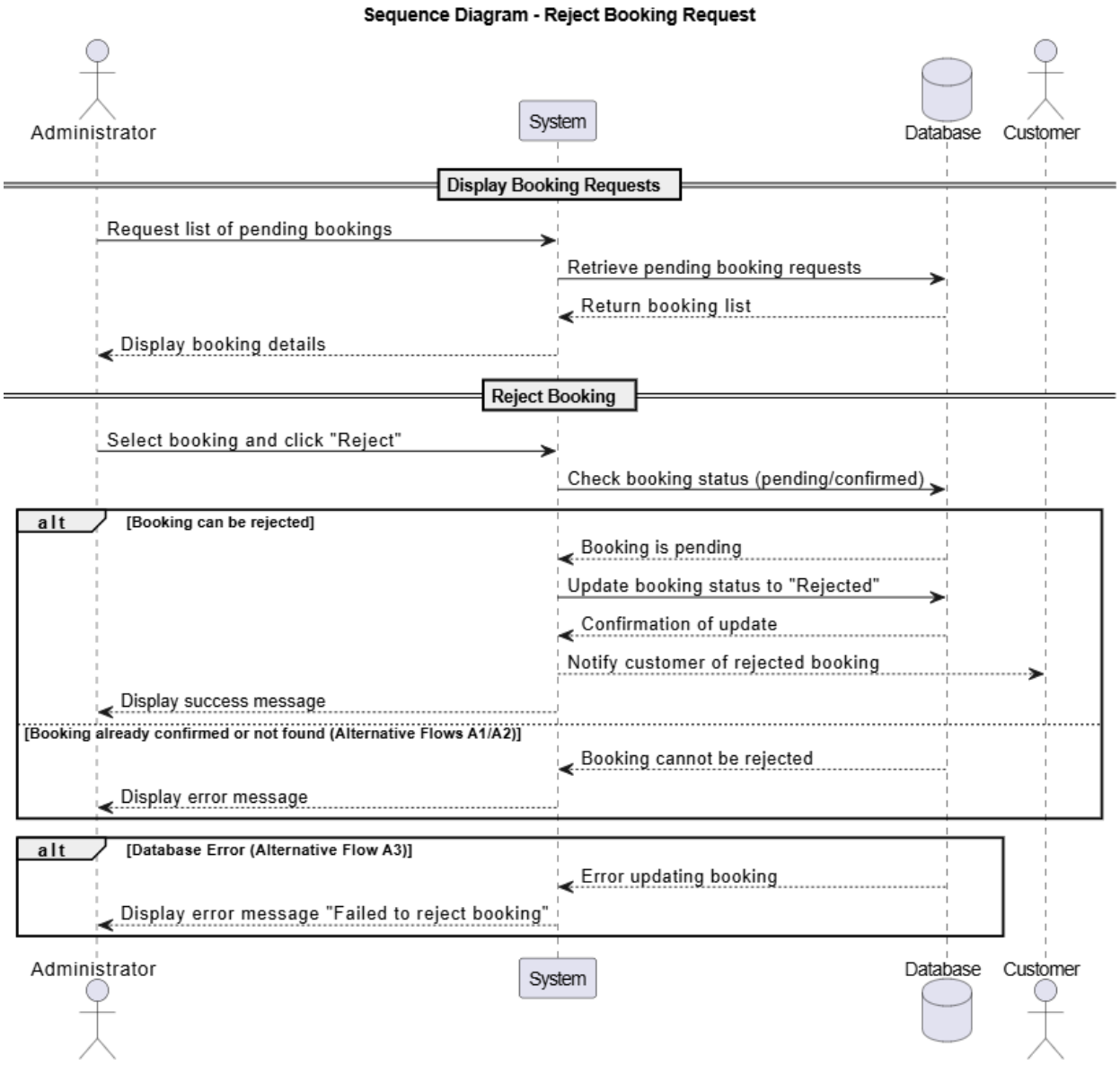

Request Car Reservation
Customer views available cars
System displays available cars
Customer selects car
Customer selects reservation period
Car available?
No
Yes
Display "Car unavailable"
Car licensed?
Yes
No
Display "Cannot reserve licensed car"
Validate reservation data
Data valid?
Yes
No
Display error mesage
Save reservation
Validate reservatondata
Display error message
Save reservation

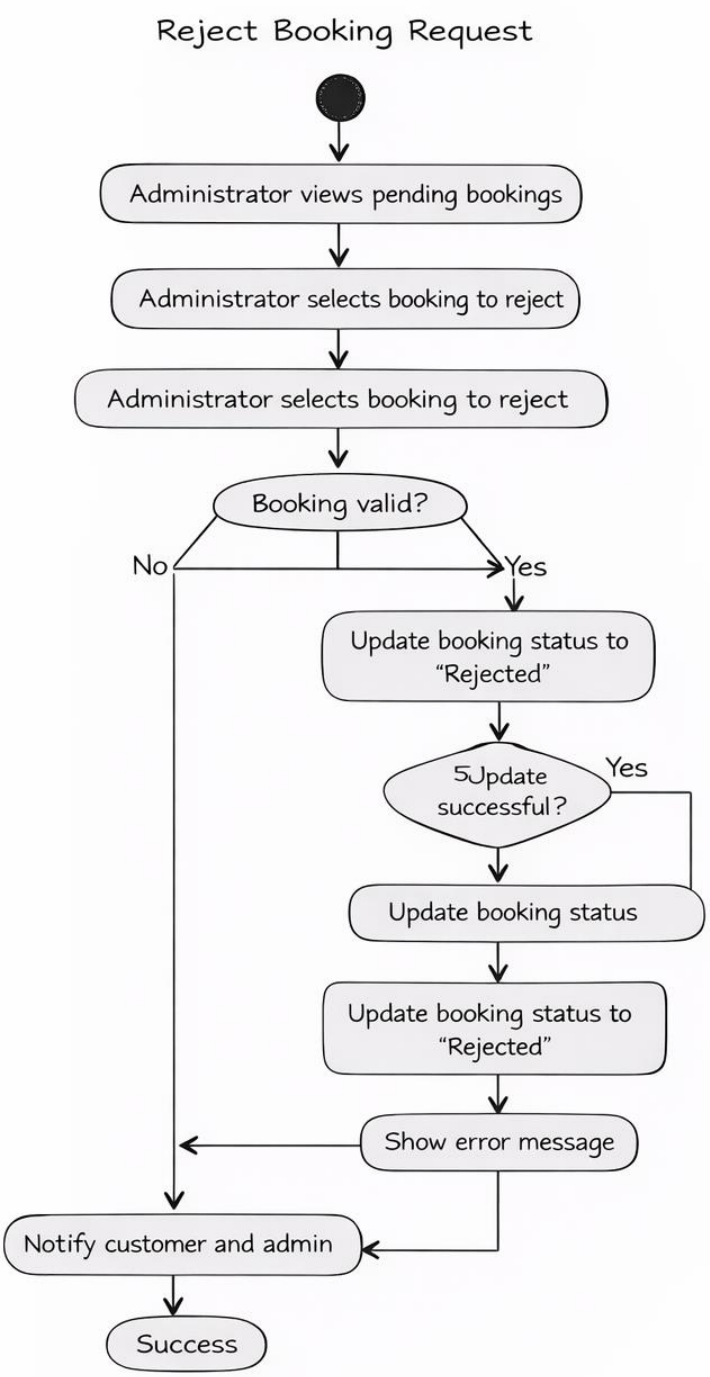

Sequence Diagram - Request Car Reservation
Customer
System
Database
Display Available Cars
Request list of available cars
Query unlicensed cars available for reservation
Return list of cars
Display available cars
Select Car & Rental Period
Select car and specify reservation period
Check car availability and unlicensed status
alt
[Car Available and Unlicensed]
Car is available and unlicensed
Save reservation
Confirm reservation saved
Display reservation confirmation (car, period)
[Car Not Available (Alternative Flow A1)]
Car unavailable
Display message "Car unavailable, select another or change period"
alt
[Car Licensed (Alternative Flow A2)]
Car is licensed
Display message "Cannot reserve licensed car"
alt
[Incorrect Data Entry (Alternative Flow A3)]
Display error message "Invalid reservation period"
Correct reservation data
alt
[Database Error (Alternative Flow A4)]
Error saving reservation
Display message "Error saving reservation, try again later"
Customer
System
Database

6.19 Reject booking request

item	Description
Use Case Name	Reject booking request
Actors	Admin
Goal	This case enables the system administrator to reject a car reservation request submitted by the user.
Preconditions	The administrator must be logged into the system and the owner of the booking management powers. • There must be a reservation request in the system to review the status. • The reservation has not been confirmed or accepted in advance
Main Flow	1. The administrator displays the list of unprocessed booking requests. 2. The administrator chooses the reservation request to be rejected. 3. The system displays the details of the request (car, booking period, customer, current status). 4. The administrator presses the “Decle request” button. 5. The system verifies that the request is subject to rejection (not accepted or confirmed in advance). 6. The system updates the status of the reservation in the database to “Rejected”. 7. The system sends a notice to the customer stating that the reservation has been rejected. 8. The system displays a message to the administrator confirming the rejection of the request successfully. 9. The process ends successfully
Alternative Flow 1	– The request does not exist or has been deleted • The administrator selects a request that does not exist or has been previously deleted. • The system displays a message: “Booking request does not exist”. • The process ends without any modification.
Alternative Flow 2	– The request has been accepted or confirmed in advance • The system detects that the request has been accepted or confirmed. • The system displays a message: “This request cannot be rejected because it has already been accepted.” • The process ends without modifying the status.
Alternative Flow 3	– Database Update Failed • The system fails to update the booking status due to a database glitch or an internal error. • The system displays a message: “Refusal of reservation failed, please try later”. • The process ends without changing the status.

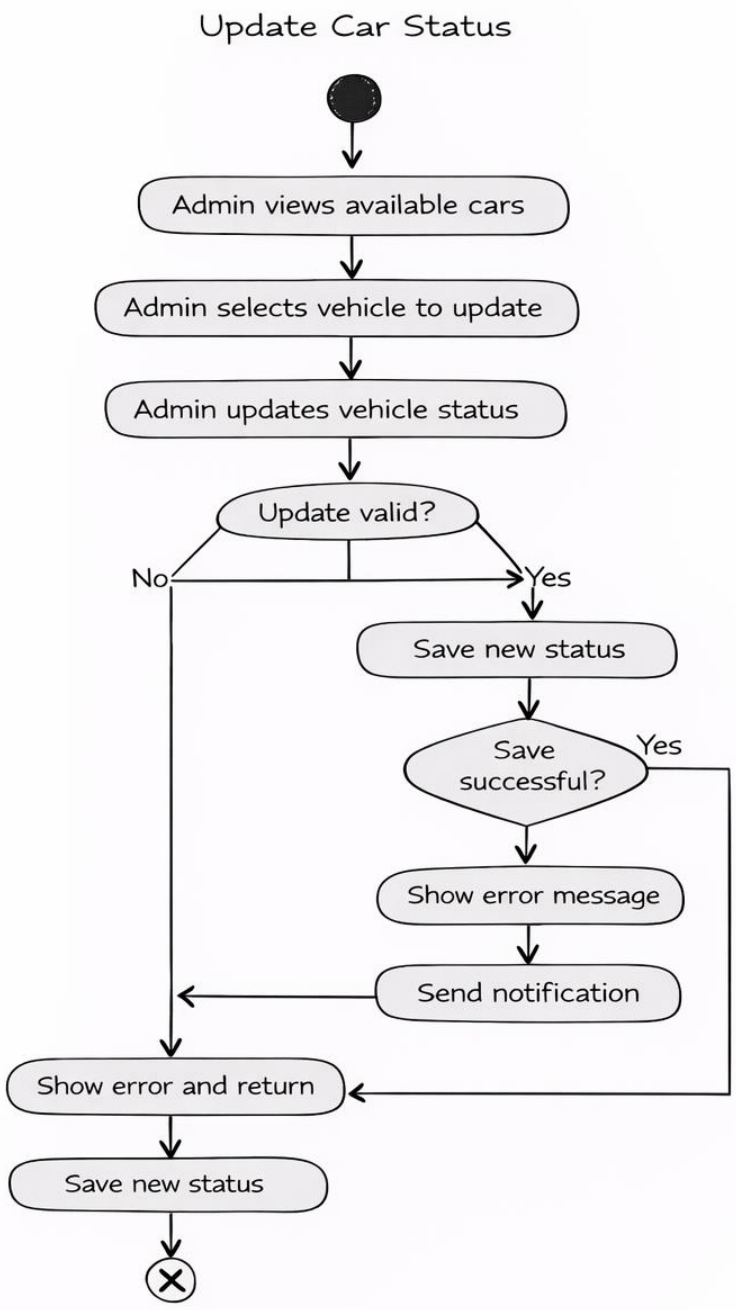
Postconditions	Success of the process: • The status of the reservation is updated to “Rejected”, and the customer is notified of the booking rejection message. • Process failure: The booking status is not changed, and the request remains the same

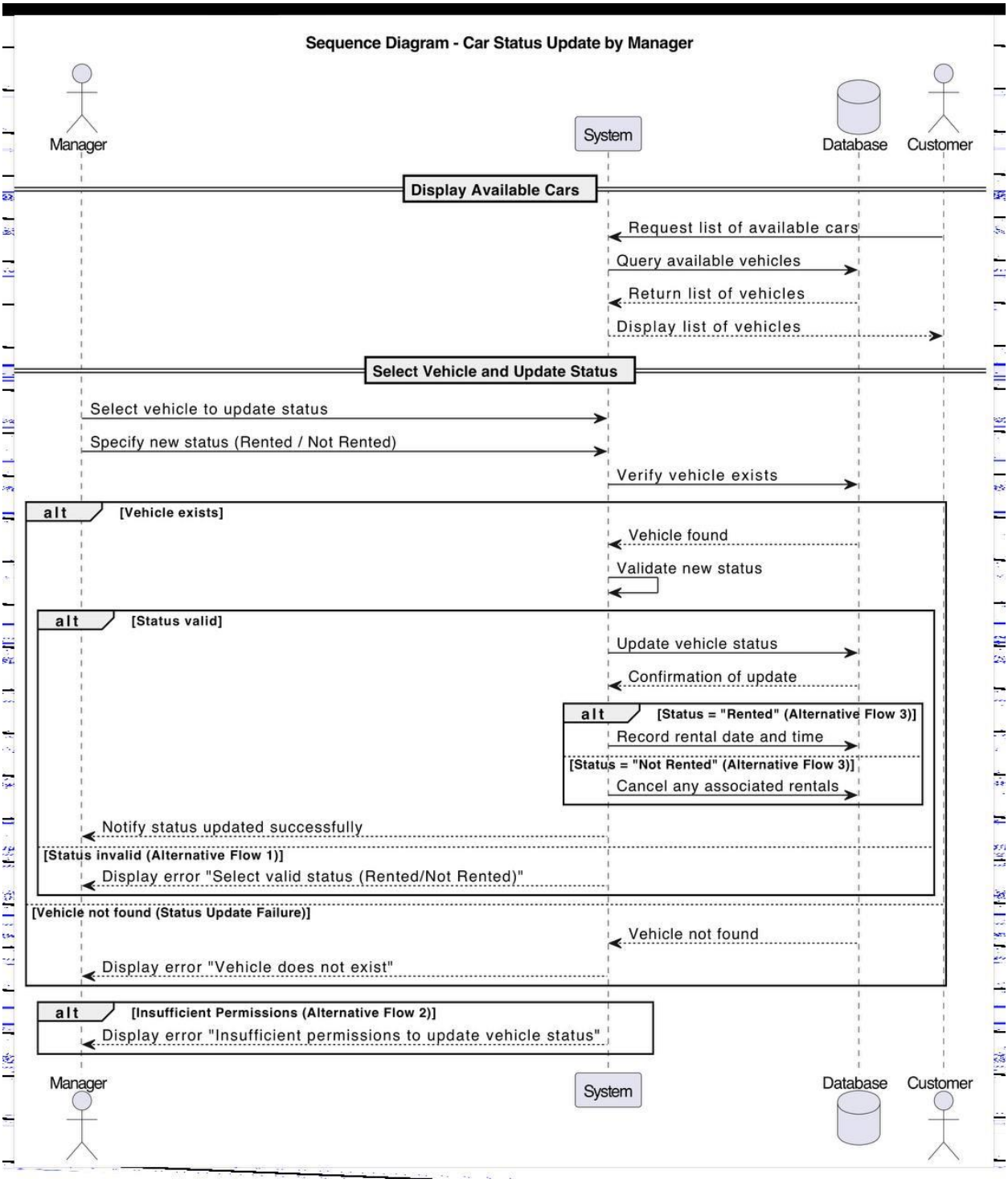



Reject Booking Request
Administrator views pending bookings
Administrator selects booking to reject
Administrator selects booking to reject
Booking valid?
No
Yes
Update booking status to "Rejected"
5Update successful?
Yes
Update booking status
Update booking status to "Rejected"
Show error message
Notify customer and admin
Success

6.20 Car status update by manager

Item	Description
Use Case Name	Car status update by manager
Actors	Admin
goal	The manager can update the car's status in the system to determine whether the car is rented or not.
Preconditions	The manager must be logged into the system. The vehicle must be listed in the database.
Main Flow	1. The customer asks the system to display the available cars. 2. The system displays a list of all available vehicles. 3. The manager selects the vehicle whose status they want to update. 4. The manager specifies the vehicle's status (rented/not rented) via the system interface. 5. The system updates the vehicle's status in the database. 6. The system sends a notification of the updated vehicle status. 7. The manager concludes the update process.
Alternative Flow 1	Status Update Failure: If the vehicle is not in the database or cannot be found, the system displays an error message stating that the vehicle does not exist. The manager tries again or selects a different vehicle.
Alternative Flow 2	Vehicle Status Selection Failure: If the manager cannot correctly select the vehicle status (for example, selects an unrecognized status), the system displays an error message requesting that the correct status (rented/not rented) be selected.
Alternative Flow 3	Insufficient Permissions: If the manager does not have the permissions to update the vehicle status, the system displays a message stating that there are insufficient permissions. The manager reviews the permissions or contacts the responsible person.
Postconditions	The vehicle's status is updated in the system (Leased/Unleased). If the vehicle's status is updated to "Leased," the date and time of the rental are recorded. If the vehicle's status is updated to "Unleased," any rentals associated with the vehicle are canceled.

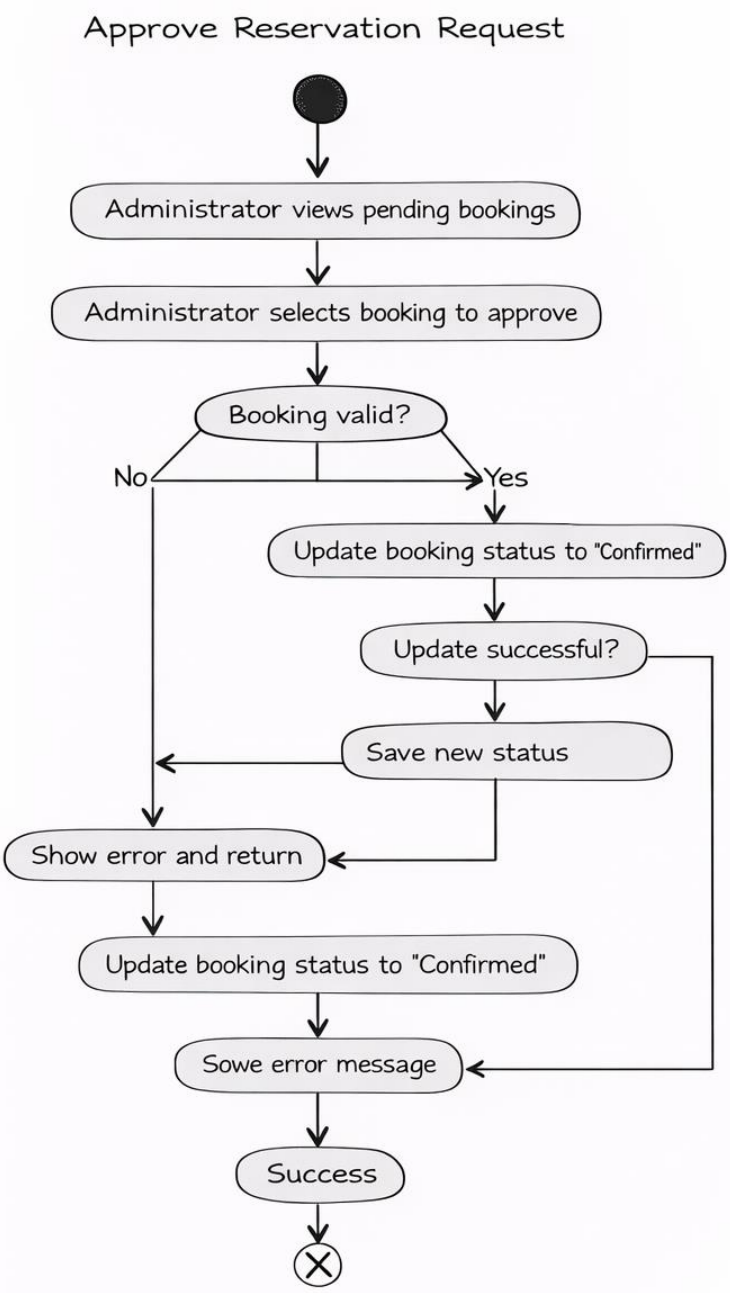

Update Car Status
Admin views available cars
Admin selects vehicle to update
Admin updates vehicle status
Update valid?
No
Yes
Save new status
Save successful?
Yes
Show error message
Send notification
Show error and return
Save new status

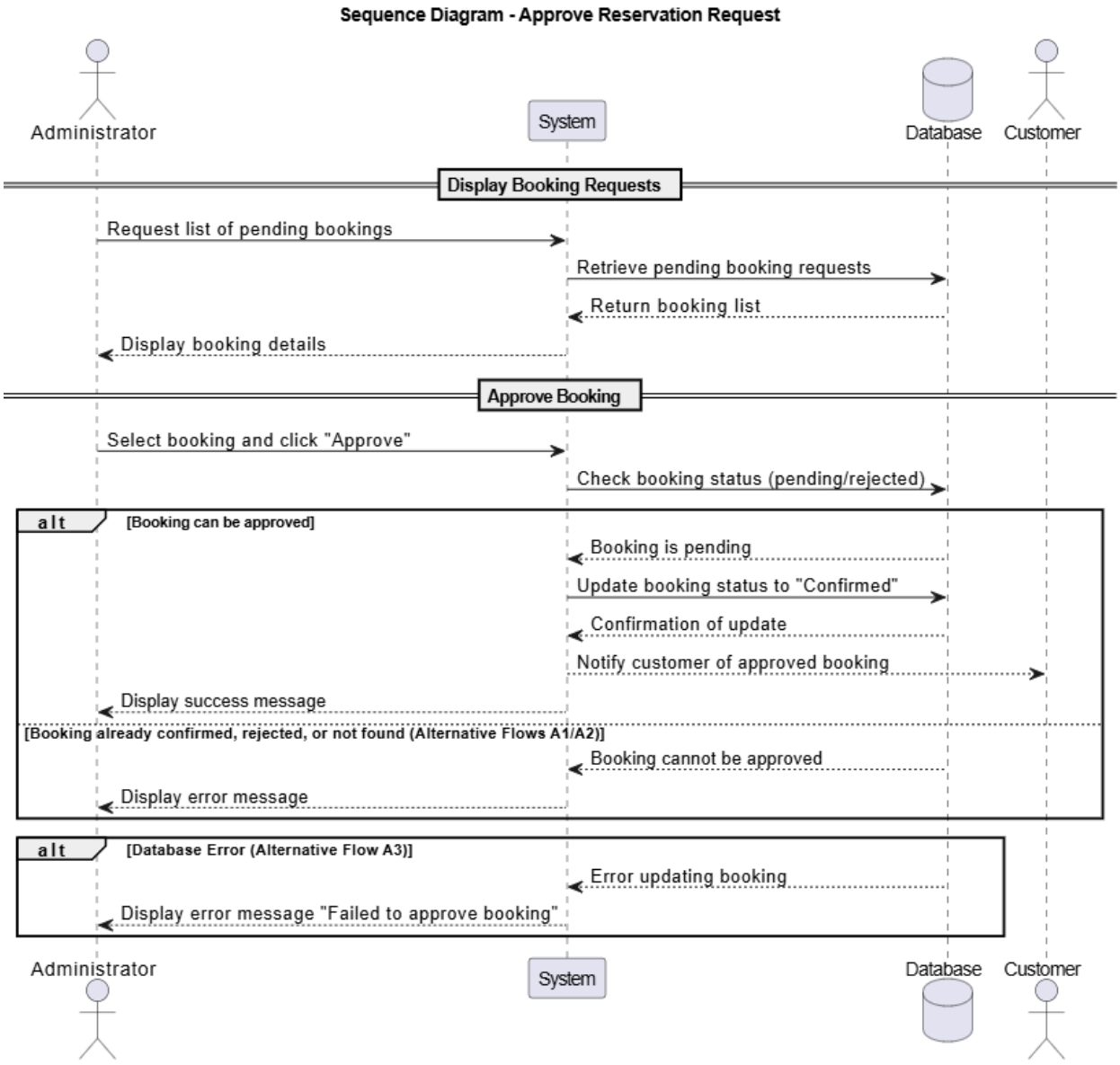

Sequence Diagram - Car Status Update by Manager
Manager
System
Database
Customer
Display Available Cars
Request list of available cars
Query available vehicles
Return list of vehicles
Display list of vehicles
Select Vehicle and Update Status
Select vehicle to update status
Specify new status (Rented / Not Rented)
Verify vehicle exists
alt
[Vehicle exists]
Vehicle found
Validate new status
alt
[Status valid]
Update vehicle status
Confirmation of update
alt
[Status = "Rented" (Alternative Flow 3)]
Record rental date and time
[Status = "Not Rented" (Alternative Flow 3)]
Cancel any associated rentals
Notify status updated successfully
[Status invalid (Alternative Flow 1)]
Display error "Select valid status (Rented/Not Rented)"
[Vehicle not found (Status Update Failure)]
Vehicle not found
Display error "Vehicle does not exist"
alt
[Insufficient Permissions (Alternative Flow 2)]
Display error "Insufficient permissions to update vehicle status"
Manager
System
Database
Customer

6.21 Approve Reservation Request

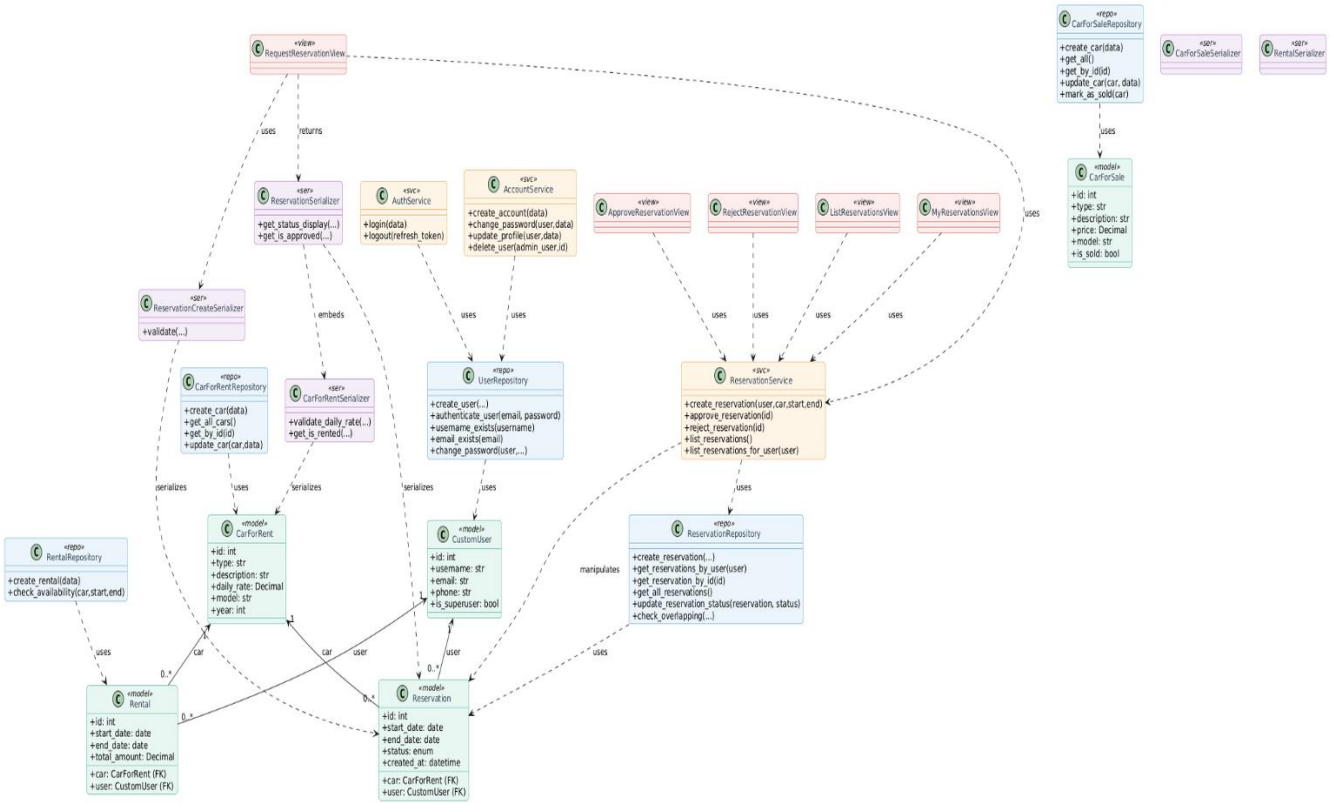
Item	Description
Use Case Name	Approve Reservation Request
Actors	Admin
goal	This case enables the system administrator to approve a car reservation request submitted by the user.
Preconditions	• The administrator must be logged into the system and the owner of the booking management powers. • There must be a reservation request in the system for review. • The reservation has not been rejected or confirmed in advance
Main Flow	1. The administrator displays the list of unprocessed booking requests. 2. The administrator selects the reservation request to be approved. 3. The system displays the details of the request (car, booking period, customer, current status). 4. The administrator presses the “Agree” button. 5. The system verifies that the request is subject to approval (not rejected or confirmed in advance). 6. The system updates the booking status in the database to “confirmed”. 7. The system sends a notification to the customer stating the approval of the reservation. 8. The system displays a message to the administrator confirming the success of the approval process. 9. The process ends successfully
Alternative Flow 1	– The request does not exist or has been deleted • The administrator selects a request that does not exist or has been deleted. • The system displays a message: “Booking request does not exist”. • The process ends without any modification.
Alternative Flow 2	– Application rejected or pre-confirmed • The system detects that the request has been rejected or confirmed in advance.

	• The system displays a message: “This request cannot be approved because it has been rejected or confirmed in advance.” • The process ends without modifying the status.
Alternative Flow 3	-Database Update Failed • The system fails to update the booking status due to a database glitch or an internal error. • The system displays a message: “The approval of the reservation failed, please try later”. • The process ends without changing the status.
Postconditions	Success of the process: • The status of the reservation is updated to "Confirmed", and the customer is notified by the booking confirmation message. • Operation failure: The booking status is not changed and the request remains the same

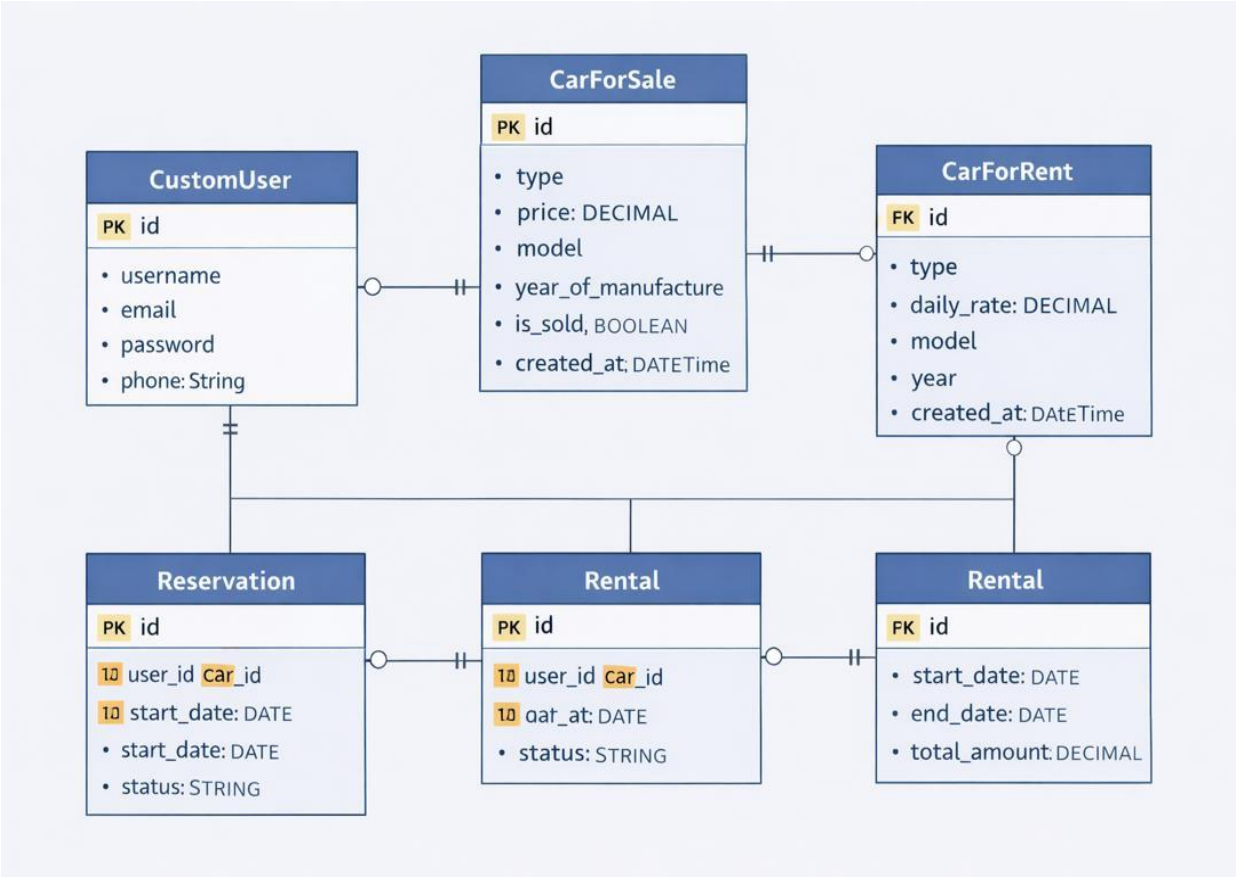

Approve Reservation Request
Administrator views pending bookings
Administrator selects booking to approve
Booking valid?
No
Yes
Update booking status to "Confirmed"
Update successful?
Save new status
Show error and return
Update booking status to "Confirmed"
Sowe error message
Success


Sequence Diagram - Approve Reservation Request
Administrator
System
Database
Customer
Display Booking Requests
Request list of pending bookings
Retrieve pending booking requests
Return booking list
Display booking details
Approve Booking
Select booking and click "Approve"
Check booking status (pending/rejected)
alt
[Booking can be approved]
Booking is pending
Update booking status to "Confirmed"
Confirmation of update
Notify customer of approved booking
Display success message
[Booking already confirmed, rejected, or not found (Alternative Flows A1/A2)]
Booking cannot be approved
Display error message
alt
[Database Error (Alternative Flow A3)]
Error updating booking
Display error message "Failed to approve booking"
Administrator
System
Database
Customer

6.22 Analysis Class Diagram:



6.23 ERD Diagram:



7. Initial Test Cases

Table 23 Test cases for User Management by Admin

Test Case Scenario: User Management by Admin

Test case id	Test case title	Test steps	Expected result
TC-01	Display users dashboard	1. Launch the website. 2. Login as Admin. 3. Open Users dashboard.	Users dashboard displayed with users list.
TC-02	Search for a user by email	1. Open Users dashboard. 2. Enter email in search box. 3. Click Search.	Matching user(s) displayed.
TC-03	Filter users by role	1. Open Users dashboard. 2. Choose role filter (Client/Admin). 3. Apply filter.	Users list filtered by role.
TC-04	Sort users list	1. Open Users dashboard. 2. Click sort by (Name/Date).	Users list sorted correctly.
TC-05	Add new user (valid)	1. Open Users dashboard. 2. Click Add User. 3. Insert valid user info. 4. Save.	User added successfully.
TC-06	Add new user but user already exists	1. Add User. 2. Insert existing email. 3. Save.	Error message: user already exist with this email.
TC-07	Add new user with missing fields	1. Add User. 2. Leave required fields empty. 3. Save.	Error message: fields required.
TC-08	Add new user with invalid information	1. Add User. 2. Insert invalid email/phone. 3. Save.	Error message: invalid information.

TC-09	Update user info (valid)	1. Users dashboard. 2. Click Edit. 3. Update user info. 4. Save.	User edited successfully.
TC-10	Update user info with invalid information	1. Edit user. 2. Insert invalid data. 3. Save.	Error message: invalid information.
TC-11	Deactivate user account	1. Users dashboard. 2. Click Deactivate. 3. Confirm action.	User status becomes inactive and cannot login.
TC-12	Reactivate user account	1. Users dashboard. 2. Click Activate. 3. Confirm action.	User status becomes active and can login.
TC-13	Reset user password (admin)	1. Users dashboard. 2. Select user. 3. Click Reset Password. 4. Confirm.	Password reset completed; user receives reset instruction (or temporary password).
TC-14	Delete user	1. Users dashboard. 2. Click Delete. 3. Confirm delete.	User deleted successfully.
TC-15	Prevent deleting admin account (rule)	1. Login as Admin. 2. Try to delete main Admin user.	System rejects action with proper message.

Table 24 Test cases for Account Info management

Test Case Scenario: Account Info management

Test case id	Test case title	Test steps	Expected result
TC-16	Display profile	1. Launch website. 2. Login. 3. Open profile page.	Profile page displayed with current user info.
TC-17	Update account info (valid)	1. Profile. 2. Click Edit My profile. 3. Update info. 4. Save.	User info updated successfully.
TC-18	Update account info with invalid information	1. Profile edit. 2. Insert invalid email/phone. 3. Save.	Error message: invalid information.
TC-19	Update account info without any changes	1. Profile edit. 2. Click Save without changes.	Message: No changes made.
TC-20	Change password (valid)	1. Profile. 2. Change password. 3. Enter current + new strong password. 4. Save.	Password updated successfully and user stays logged in (or re-login required).
TC-21	Change password with wrong current password	1. Change password. 2. Enter wrong current password. 3. Save.	Error message: current password incorrect.
TC-22	Change password with weak password	1. Change password. 2. Enter weak new password. 3. Save.	Error message: password too weak.
TC-23	Logout	1. Click Logout.	User logged out and redirected to login page.

Table 25 Test cases for Car For Sale management by Admin

Test Case Scenario: Car For Sale management by Admin

Test case id	Test case title	Test steps	Expected result
TC-24	Display cars-for-sale dashboard	1. Admin login. 2. Open Cars For Sale dashboard.	Cars-for-sale list displayed.
TC-25	Add car for sale (valid)	1. Add Car. 2. Insert valid data (type, description, price, images). 3. Save.	Car for sale added successfully.
TC-26	Add car for sale missing fields	1. Add Car. 2. Leave required fields empty. 3. Save.	Error message: fields required.
TC-27	Add car for sale invalid price	1. Add Car. 2. Enter negative/invalid price. 3. Save.	Error message: invalid information.
TC-28	Upload car images (valid)	1. Add/Edit Car. 2. Upload supported image type. 3. Save.	Images uploaded and displayed correctly.
TC-29	Upload car images unsupported type	1. Upload .exe/.txt file.	Error message: unsupported file type.
TC-30	Update car for sale info	1. Edit car. 2. Modify details. 3. Save.	Car updated successfully.
TC-31	Delete car for sale	1. Delete car. 2. Confirm.	Car deleted successfully.
TC-32	Search car for sale by model/type	1. Dashboard. 2. Search by keyword/type filter.	Filtered results displayed correctly.
TC-33	Mark car as sold	1. Open car details. 2. Click Mark as Sold.	Car status becomes Sold and not shown in available listings.

Table 26 Test cases for Car For Rent management by Admin

Test Case Scenario: Car For Rent management by Admin

Test case id	Test case title	Test steps	Expected result
TC-34	Display cars-for-rent dashboard	1. Admin login. 2. Open Cars For Rent dashboard.	Cars-for-rent list displayed.
TC-35	Add car for rent (valid)	1. Add Car For Rent. 2. Enter valid model, price_per_day, availability. 3. Save.	Car for rent added successfully.
TC-36	Add car for rent missing fields	1. Add Car For Rent. 2. Leave required fields empty. 3. Save.	Error message: fields required.
TC-37	Add car for rent invalid price_per_day	1. Add Car For Rent. 2. Enter invalid/negative price. 3. Save.	Error message: invalid information.
TC-38	Update car availability	1. Edit car. 2. Change availability status. 3. Save.	Availability updated successfully.
TC-39	Delete car for rent	1. Delete car. 2. Confirm.	Car for rent deleted successfully.
TC-40	Prevent deleting car with active reservation	1. Select car with confirmed reservation. 2. Try delete.	System rejects or requests handling reservations first.

Table 27 Test cases for Reservation management

Test Case Scenario: Reservation management (Client + Admin)

Test case id	Test case title	Test steps	Expected result
TC-41	Create reservation (valid)	1. Client login. 2. Open car details. 3. Choose valid start/end dates. 4. Click Reserve.	Reservation created with status Pending.
TC-42	Create reservation with end date before start date	1. Choose invalid date range. 2. Reserve.	Error message: invalid date range.
TC-43	Create reservation overlapping existing confirmed reservation	1. Choose dates that overlap confirmed reservation. 2. Reserve.	System rejects reservation or shows unavailable message.
TC-44	View my reservations list	1. Client dashboard. 2. Open My Reservations.	Reservations list displayed.
TC-45	Cancel pending reservation	1. My Reservations. 2. Cancel reservation. 3. Confirm.	Reservation canceled successfully.
TC-46	Admin confirms reservation	1. Admin opens Reservations dashboard. 2. Select pending reservation. 3. Confirm.	Reservation status becomes Confirmed.
TC-47	Admin rejects reservation	1. Admin selects pending reservation. 2. Reject.	Reservation status becomes Rejected.
TC-48	Reservation audit trail	1. Open reservation details. 2. Verify created_at and status history (if supported).	Correct timestamps shown; status current is accurate.

Table 28 Test cases for Rental management

Test Case Scenario: Rental management (Admin + System)

Test case id	Test case title	Test steps	Expected result
TC-49	Create rental from confirmed reservation	1. Admin opens confirmed reservation. 2. Create rental.	Rental created successfully.
TC-50	Calculate total amount correctly	1. Create rental. 2. Set dates and car price_per_day.	Total amount = number_of_days × price_per_day calculated correctly.
TC-51	Prevent rental creation for pending reservation	1. Select pending reservation. 2. Try create rental.	System rejects this operation.
TC-52	End rental (return car)	1. Admin opens active rental. 2. Click End Rental. 3. Confirm.	Rental closed; car availability updated.
TC-53	Rental history for client	1. Client dashboard. 2. Open My Rentals.	Rental history displayed.
TC-54	Generate rental receipt/invoice (if supported)	1. Open rental details. 2. Click Generate receipt.	Receipt generated successfully.

Table 29 Test cases for Sign in Functionality

Test Case Scenario: Check Sign in Functionality

Test case id	Test case title	Test steps	Expected result
TC-55	Login with valid credentials	1. Open login page. 2. Enter valid email and password. 3. Click Sign in.	Login should be successful.
TC-56	Login with invalid credentials	1. Enter invalid email or password. 2. Click Sign in.	Error message “invalid email or password.”
TC-57	Login with empty fields	1. Leave email or password empty. 2. Click Sign in.	Error message “a field is missing”
TC-58	Remember me functionality	1. Login with Remember me enabled. 2. Close and reopen browser. 3. Open website.	User remains logged in (if feature exists).
TC-59	Session timeout	1. Login. 2. Stay inactive until session expires. 3. Navigate to protected page.	User redirected to login page.
TC-60	Unauthorized access to admin pages	1. Login as client. 2. Open admin URL/dashboard.	Access denied message or redirect.

Table 30 Test cases for Register Functionality

Test Case Scenario: Check Register Functionality

Test case id	Test case title	Test steps	Expected result
TC-61	Register with valid information	1. Open register page. 2. Complete form correctly. 3. Click Register.	The Register should be successful.
TC-62	Register with existing email	1. Enter existing email. 2. Register.	Error message “the email is already existed”.
TC-63	Register with weak password	1. Enter password < 8 characters. 2. Register.	Error message “Must contain at least 8 characters”.
TC-64	Register with missing fields	1. Leave required fields empty. 2. Register.	Error message “a field is missing”.
TC-65	Register with invalid email format	1. Enter invalid email. 2. Register.	Error message: invalid information.
TC-66	Register and verify account activation (if supported)	1. Register. 2. Open activation link/email.	Account activated successfully.

Table 31 Test cases for Spare Parts Store

Test Case Scenario: Spare Parts browsing and purchase

Test case id	Test case title	Test steps	Expected result
TC-67	View spare parts catalog	1. Open Spare Parts section.	Catalog displayed.
TC-68	Search spare part by name/part number	1. Enter keyword. 2. Search.	Matching parts displayed.
TC-69	Filter spare parts by category	1. Select category filter. 2. Apply.	Results filtered correctly.
TC-70	View spare part details	1. Open part details.	Part details displayed (price, stock, description).
TC-71	Add spare part to cart	1. From part details. 2. Click Add to cart.	Part added to cart successfully.
TC-72	Add out-of-stock part to cart	1. Select out-of-stock part. 2. Add to cart.	System rejects and shows out-of-stock message.
TC-73	Update cart quantities	1. Cart. 2. Increase/decrease quantity.	Cart updates totals correctly.
TC-74	Remove item from cart	1. Cart. 2. Remove item.	Item removed successfully.
TC-75	Checkout (cash/online)	1. Cart. 2. Proceed to checkout. 3. Enter shipping/contact. 4. Confirm.	Order created successfully.

Table 32 Test cases for Workshop / Maintenance Services

Test Case Scenario: Workshop booking and service management

Test case id	Test case title	Test steps	Expected result
TC-76	View workshop services list	1. Open Workshop section.	Services list displayed.
TC-77	Book maintenance appointment (valid)	1. Choose service. 2. Select date/time. 3. Submit booking.	Appointment created successfully.
TC-78	Book appointment missing fields	1. Leave contact/service details empty. 2. Submit.	Error message: fields required.
TC-79	Book appointment in past date	1. Select past date. 2. Submit.	Error message: invalid date.
TC-80	Admin view appointments dashboard	1. Admin login. 2. Open appointments dashboard.	Appointments displayed.
TC-81	Admin update appointment status	1. Select appointment. 2. Change status (Scheduled/Done/Cancelled).	Status updated successfully.
TC-82	Client cancel appointment	1. My appointments. 2. Cancel. 3. Confirm.	Appointment cancelled successfully.

Table 33 Test cases for Cars Browsing (Client)

Test Case Scenario: Cars browsing and details view

Test case id	Test case title	Test steps	Expected result
TC-83	Browse available cars for sale	1. Open Cars for Sale page.	Cars list displayed.
TC-84	Browse available cars for rent	1. Open Cars for Rent page.	Cars list displayed.
TC-85	Filter cars by type	1. Choose type filter. 2. Apply.	Filtered cars displayed.
TC-86	Sort cars by price	1. Sort by price low-to-high/high-to-low.	Cars sorted correctly.
TC-87	View car details page	1. Click a car card.	Car details displayed with images and specs.
TC-88	Handle car details missing image	1. Open car without image.	Default image/placeholder shown without errors.

8. Initial Requirement Trackability Matrix (RTM)

Req_ID	Title	SRS Section	System Design	Analysis (Use Cases)	Detail Design	Coding	Test Cases	Changed Requests No
RE-FR-UM-1.1	The system shall allow the Admin to create a new user profile by entering necessary information	User Management	User Module	FR(Add_User) Seq1.1				
RE-FR-UM-1.2	The system shall allow the Admin to edit user information	User Management	User Module	FR(Edit_User) Seq1.2				
RE-FR-UM-1.3	The system shall allow the Admin to delete a user	User Management	User Module	FR(Delete_User) Seq1.3				
RE-FR-UM-1.4	The system shall automatically assign a default role to a newly created user based on the actor	User Management	Authentication Module	FR(Assign_Role) Seq1.4				

	performing the creation							
RE-FR-UM-1.4.1	When the Admin creates a new user, the system shall assign the role Admin by default	User Management	Authentication Module	FR(Assign_Role) Seq1.4.1				
RE-FR-UM-1.4.2	When a user registers through the system, the system shall assign the role Client by default	User Management	Authentication Module	FR(Register) Seq1.4.2				
RE-FR-CSM-2.1	The system shall allow the Admin to add a car for sale	Car Management	Car Module	FR(Add_Car_For_Sale) Seq2.1				
RE-FR-CSM-2.2	The system shall allow the Admin to edit car for sale details	Car Management	Car Module	FR(Edit_Car_For_Sale) Seq2.2				

RE-FR-CSM-2.3	The system shall allow the Admin to delete a car for sale	Car Management	Car Module	FR(Delete_Car_For_Sale) Seq2.3				
RE-FR-CRM-3.1	The system shall allow the Admin to add a car for rent	Car Rental	Car Module	FR(Add_Car_For_Rent) Seq3.1				
RE-FR-CRM-3.2	The system shall allow the Admin to update car availability	Car Rental	Car Module	FR(Update_Availability) Seq3.2				
RE-FR-RES-4.1	The system shall allow Clients to create a reservation by selecting rental period	Reservation Management	Reservation Module	FR(Create_Reservation) Seq4.1				
RE-FR-RES-4.2	The system shall allow the Admin to confirm or reject reservations	Reservation Management	Reservation Module	FR(Confirm_Reservation) Seq4.2				
RE-FR-REN-5.1	The system shall allow the Admin	Rental Management	Rental Module	FR(Create_Rental) Seq5.1				

	to create a rental from confirmed reservation							
RE-FR-REN-5.2	The system shall calculate total rental amount automatically	Rental Management	Rental Module	FR(Calculate_Rental) Seq5.2				
RE-FR-AIM-6.1	The system shall allow users to update their account information	Account Management	Account Module	FR(Update_Account) Seq6.1				
RE-FR-AIM-6.2	The system shall allow users to delete their accounts	Account Management	Account Module	FR(Delete_Account) Seq6.2				
RE-FR-AUTH-7.1	The system shall provide a login page for all users	Authentication	Auth Module	FR(Login) Seq7.1				
RE-FR-AUTH-7.2	The system shall validate user credentials	Authentication	Auth Module	FR(Login) Seq7.2				

	and display error messages							
RE-FR-AUTH-7.3	The system shall provide a logout option for users	Authentication	Auth Module	FR(Logout) Seq7.3				
RE-NF-SEC-8.1	The system shall encrypt all sensitive data	Security	Security Layer	—				
RE-NF-SCAL-8.2	The system shall be scalable to support increasing users and cars	Scalability	Architecture	—				

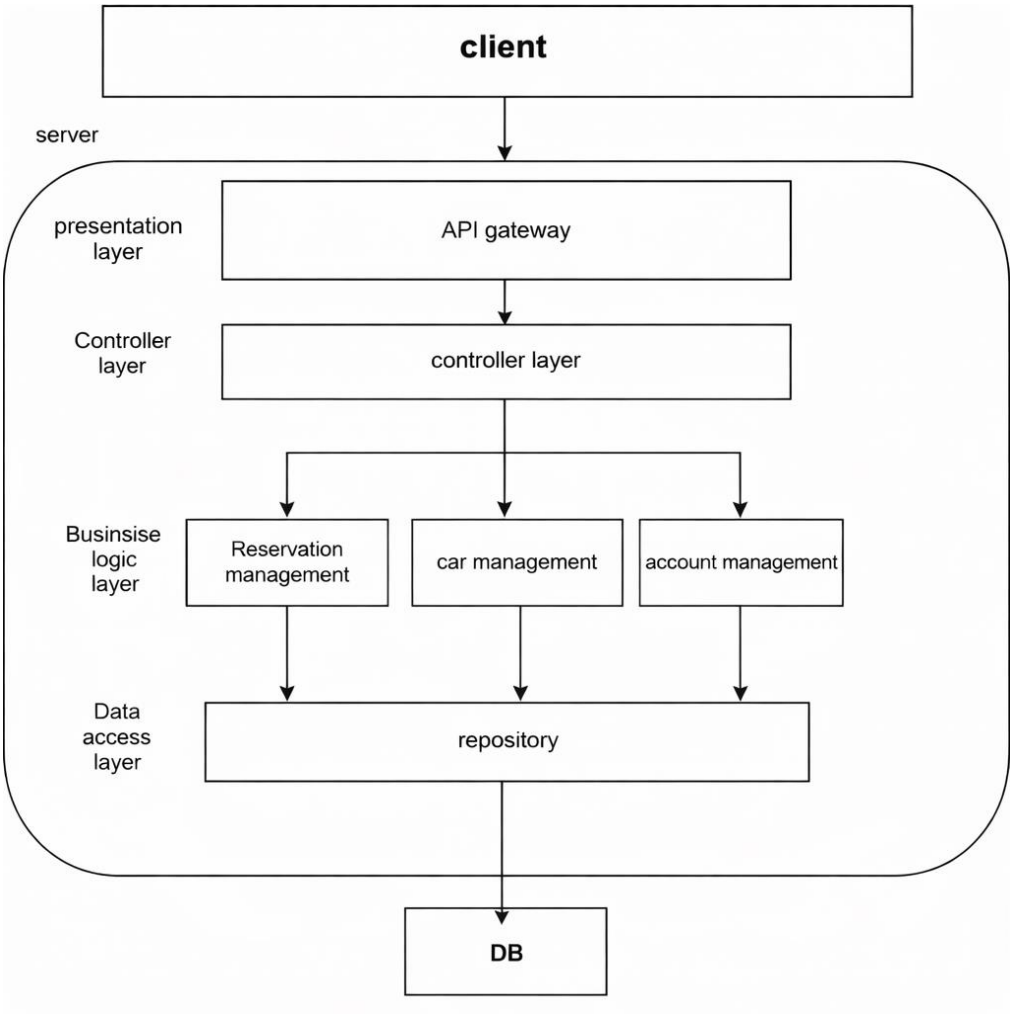
Chapter 5 System Design

1. Introduction

This chapter presents an overview of the design phase of the Integrated Car Center Management System. It outlines the main design activities, including system architecture definition and detailed design and modeling. These steps are essential for building a reliable and well-structured system that supports efficient management of users, vehicles, and system operations, while ensuring scalability and maintainability.

2. System Architecture:

This section discusses the **system architecture** of the Integrated Car Center Management System. System architecture defines the overall structure of the software and describes how system components interact and integrate with each other. This phase establishes a clear blueprint that ensures efficient data flow, smooth interaction between modules, and reliable system integration.



3. Component Functionalities:

1. API Layer (Views):

This component is responsible for handling requests coming from the client side. It routes requests to the appropriate logic, processes inputs, and returns responses to the user interface, ensuring secure and efficient communication between the client and the server.

2. Controllers (Django Views):

Acts as the entry point for system operations. It receives user requests, validates input data, invokes the appropriate business logic, and returns the results to the client.

3. User Management:

Handles all user-related functionalities, including user registration, authentication, role assignment (Admin, Client), profile management, and account control.

4. Car Management:

Manages all functionalities related to cars, including adding, editing, deleting, and viewing cars for sale and cars for rent, as well as managing car availability.

5. Reservation Management:

Responsible for handling car reservations, including creating reservations, managing reservation dates, and allowing administrators to confirm or reject reservations.

6. Rental Management:

Handles rental operations such as creating rentals from confirmed reservations, calculating rental costs, and managing rental periods.

7. Repository (Data Access Layer):

Acts as the data access layer of the system. It interacts directly with the database through Django models, providing functionality to create, retrieve, update, and delete system data.

8. Database (DB):

Stores all persistent system data, including user accounts, car details, reservations, rentals, and related system information.

4. System Architecture:

We used layered architecture with client-server architecture.

4.1 Client Side:

This is the topmost layer, where the user interacts with the system through a user interface.

4.2 Server-Side Layers:

- **Presentation Layer:** The API Gateway represents the entry point for client requests. This layer handles communication between the client and the server
- **Business Logic Layer:**This layer contains the core functionality of the system. It includes controllers and components for Ticket Management, Team Management, User Management, Reporting, and AI Services .
- **Data Access Layer:** This layer provides an abstraction to interact with the database. It includes the Repository pattern, which is responsible for data access operations.

4.3 Database Layer:

The database stores persistent data and is accessed via the repository in the Data Access Layer.

4.4 Detailed design for system component:

In this section we will dive into the detailed design for each component of the system, also mentions the design principles and patterns used to build a strong software system that fulfills its requirements.

5. Design Class Diagram – User Authentication:

This class diagram illustrates the structure, responsibilities, and relationships between the main classes involved in the **user authentication and account management** process in the *Integrated Car Center Management System*. The diagram reflects the actual implementation found in the project source files.

The primary classes included in this diagram are:

1. **CustomUser (User Model):**
   Represents the user entity in the system. It extends Django's built-in user model and stores user-related information such as username, email, password, and phone number.

2. **UserRepository (Model Layer):**
   Responsible for handling database operations related to user management. It interacts directly with the CustomUser model to create, retrieve, update, and delete user data.

3. **AuthService (Authentication Logic):**
   Provides authentication and authorization services. It handles operations such as user registration, login validation, role assignment, and logout by utilizing the repository layer.

4. **BaseAuthView:**
   A base view class that contains shared authentication logic. It provides common functionality required by all authentication-related views.

5. **RegisterView, LoginView, and LogoutView:**
   These classes represent the authentication endpoints of the system.

   - **RegisterView** handles new user registration.
   - **LoginView** manages user login and credential validation.
   - **LogoutView** handles secure user logout and session termination.

6. Relationships :

6.1 Dependency

- **AuthService** depends on **UserRepository** to perform all user-related database operations.
- **BaseAuthView** depends on **AuthService** to access authentication and authorization functionalities.

6.2 Inheritance

- **RegisterView**, **LoginView**, and **LogoutView** inherit from **BaseAuthView**, allowing them to reuse common authentication logic and maintain consistency across authentication endpoints.

7. Sprint 1:

User Authentication and User Management System

System Description :

This class diagram represents the architecture of the User Authentication and User Management System. It illustrates the relationships between components responsible for user registration, login, logout, profile management, and account control. The architecture follows a modular design that separates data access, business logic, and API endpoints, ensuring maintainability and alignment with the Django framework.

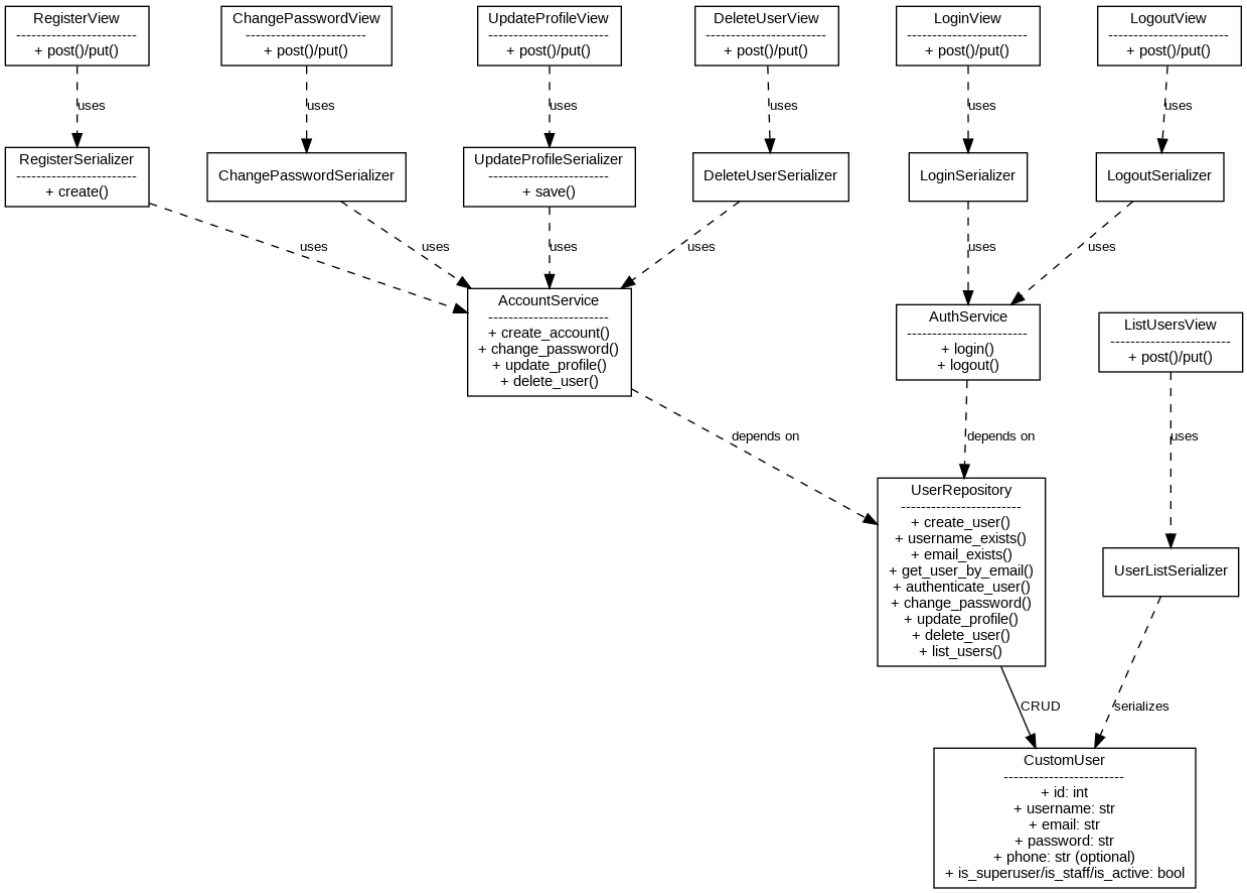
1. Relationships:

1.1 Dependency

- AuthService and AccountService depend on UserRepository for all database operations related to user accounts.
- BaseAuthView uses AuthService and AccountService to process authentication and account management requests.
- Serializer classes (e.g., RegisterSerializer, LoginSerializer) rely on authentication services to validate input data and execute business logic.

1.2 Inheritance

- RegisterView, LoginView, LogoutView, UpdateProfileView, and DeleteUserView inherit from BaseAuthView, allowing them to reuse common authentication logic and enforce consistent behavior across authentication endpoints.


RegisterView
+ post()/put()
ChangePasswordView
+ post()/put()
UpdateProfileView
+ post()/put()
DeleteUserView
+ post()/put()
LoginView
+ post()/put()
LogoutView
+ post()/put()
uses
RegisterSerializer
+ create()
ChangePasswordSerializer
UpdateProfileSerializer
+ save()
DeleteUserSerializer
LoginSerializer
LogoutSerializer
AccountService
+ create_account()
+ change_password()
+ update_profile()
+ delete_user()
AuthService
+ login()
+ logout()
ListUsersView
+ post()/put()
depends on
UserRepository
+ create_user()
+ username_exists()
+ email_exists()
+ get_user_by_email()
+ authenticate_user()
+ change_password()
+ update_profile()
+ delete_user()
+ list_users()
UserListSerializer
CRUD
serializes
CustomUser
+ id: int
+ username: str
+ email: str
+ password: str
+ phone: str (optional)
+ is_superuser/is_staff/is_active: bool

2. Sprint 2:

Car Management and Rental System

System Description :

This class diagram represents the architecture of the **Car Management and Rental System**. It demonstrates the relationships between components responsible for managing cars for sale and rent, processing rental operations, and calculating rental costs. The system adopts a layered architecture consisting of repositories, services, serializers, and views to ensure clear separation of responsibilities.

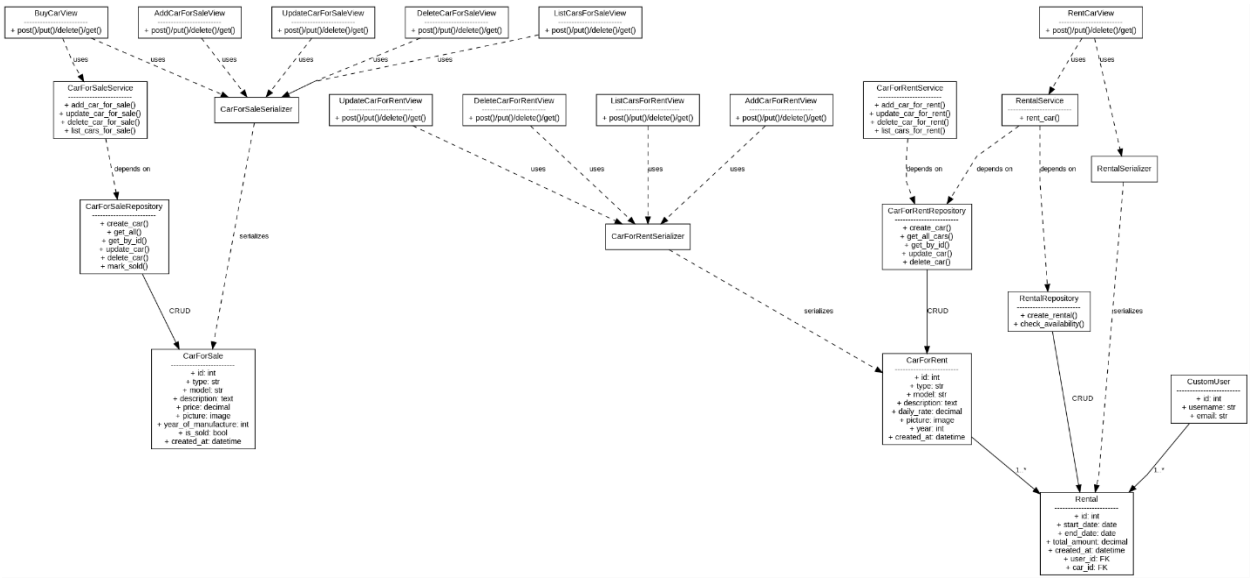
1. Relationships:

1.1 Dependency

- **CarForSaleService** depends on **CarForSaleRepository** for database operations related to cars for sale.
- **CarForRentService** depends on **CarForRentRepository** to manage cars available for rent.
- **RentalService** depends on **RentalRepository** and **CarForRentRepository** to validate availability and create rental records.
- API views depend on serializers and services to process and respond to client requests.

1.2 Association

- **CarForRent** is associated with **Rental** in a one-to-many relationship, where a single car can have multiple rental records.
- **CustomUser** is associated with **Rental**, allowing a user to perform multiple rental transactions.


BuyCarView
+ post()/put()/delete()/get()
AddCarForSaleView
+ post()/put()/delete()/get()
UpdateCarForSaleView
+ post()/put()/delete()/get()
DeleteCarForSaleView
+ post()/put()/delete()/get()
ListCarsForSaleView
+ post()/put()/delete()/get()
RentCarView
+ post()/put()/delete()/get()
uses
CarForSaleService
+ add_car_for_sale()
+ update_car_for_sale()
+ delete_car_for_sale()
+ list_cars_for_sale()
CarForSaleSerializer
UpdateCarForRentView
+ post()/put()/delete()/get()
DeleteCarForRentView
+ post()/put()/delete()/get()
ListCarsForRentView
+ post()/put()/delete()/get()
AddCarForRentView
+ post()/put()/delete()/get()
CarForRentService
+ add_car_for_rent()
+ update_car_for_rent()
+ delete_car_for_rent()
+ list_cars_for_rent()
RentalService
+ rent_car()
depends on
RentalSerializer
CarForSaleRepository
+ create_car()
+ get_all()
+ get_by_id()
+ update_car()
+ delete_car()
+ mark_sold()
serializes
CarForRentSerializer
CarForRentRepository
+ create_car()
+ get_all_cars()
+ get_by_id()
+ update_car()
+ delete_car()
CRUD
RentalRepository
+ create_rental()
+ check_availability()
CarForSale
+ id: int
+ type: str
+ model: str
+ description: text
+ price: decimal
+ picture: image
+ year_of_manufacture: int
+ is_sold: bool
+ created_at: datetime
CarForRent
+ id: int
+ type: str
+ model: str
+ description: text
+ daily_rate: decimal
+ picture: image
+ year: int
+ created_at: datetime
CustomUser
+ id: int
+ username: str
+ email: str
1 *
Rental
+ id: int
+ start_date: date
+ end_date: date
+ total_amount: decimal
+ created_at: datetime
+ user_id: FK
+ car_id: FK

2. Sprint 3:

Reservation Management System

System Description

This class diagram represents the architecture of the **Reservation Management System**. It illustrates the structure and relationships between components responsible for handling car reservation requests, managing reservation statuses, and displaying reservation information. The design ensures consistency between users, cars, and reservation records within the system.

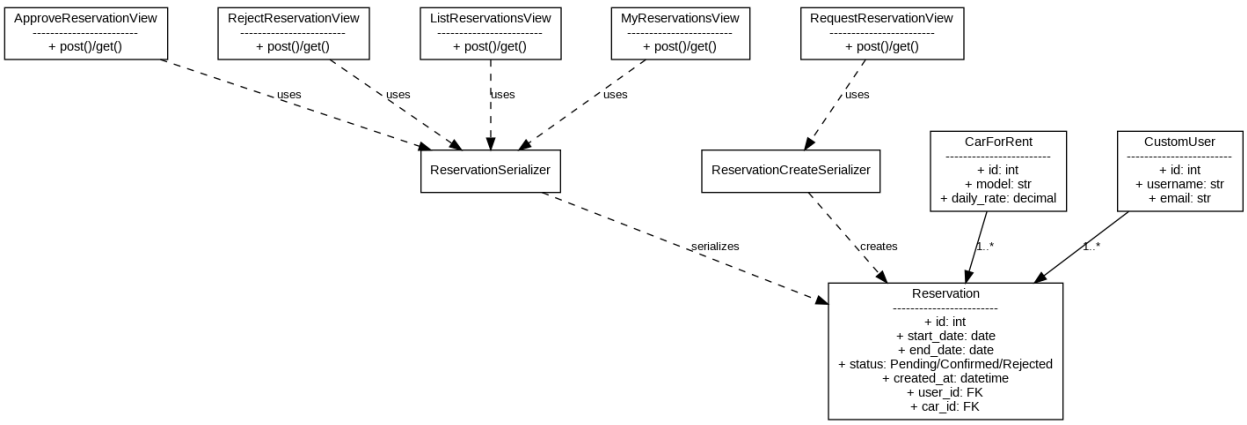
1. Relationships :

1.1 Dependency

- RequestReservationView, ApproveReservationView, and RejectReservationView depend on ReservationSerializer and ReservationCreateSerializer to process reservation data.
- Serializers interact directly with the Reservation model to create and update reservation records.

1.2 Association

- **CustomUser** is associated with **Reservation** in a one-to-many relationship, allowing users to create multiple reservations.
- **CarForRent** is associated with **Reservation**, enabling a single car to be reserved multiple times across different periods.



Chapter 6 Practical implementation

1. Introduction

In this chapter, we present the **implementation phase** of the *Integrated Car Center Management System*. This chapter describes the technologies, frameworks, and tools used to develop the system. It also presents the main system interfaces and functionalities. Finally, test cases are executed to validate the system behavior and verify that all functional requirements operate correctly under different scenarios.

2. Used tools :

2.1 Django:

is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. especially we use DRF: Django REST Framework is a widely used, full-featured API framework designed for building RESTful APIs with Django. At its core, DRF integrates with Django's core features "models, views, and URLs" making it simple and seamless to create a RESTful API.

2.2 React:

React is a popular JavaScript library for building user interfaces .It was created by Facebook and is widely used in web development. React allows developers to build reusable UI components that can efficiently update and render changes to the user interface when the underlying data changes. Reacts primary focus is on building user interfaces, and it excels in creating interactive and dynamic web applications.

2.3 My SQL Database:

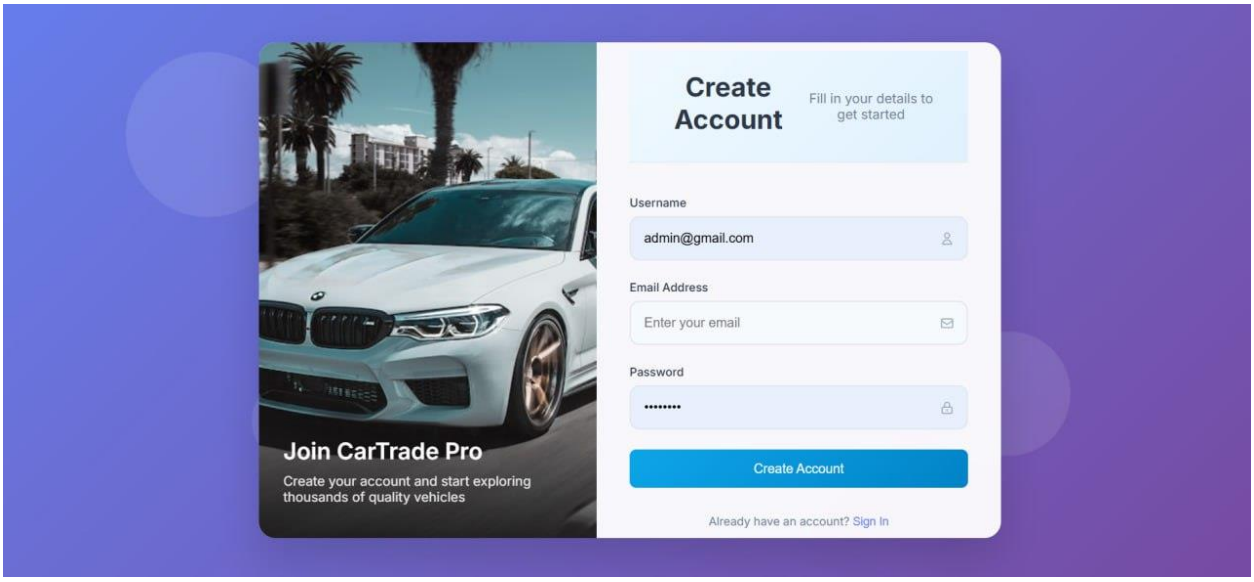
MySQL is an open-source relational database management system (RDBMS) that is widely used for storing, managing, and retrieving data. It is one of the most popular and widely adopted databases in the world, known for its reliability, scalability, and ease of use.

2.4 Visual studio code (VS code):

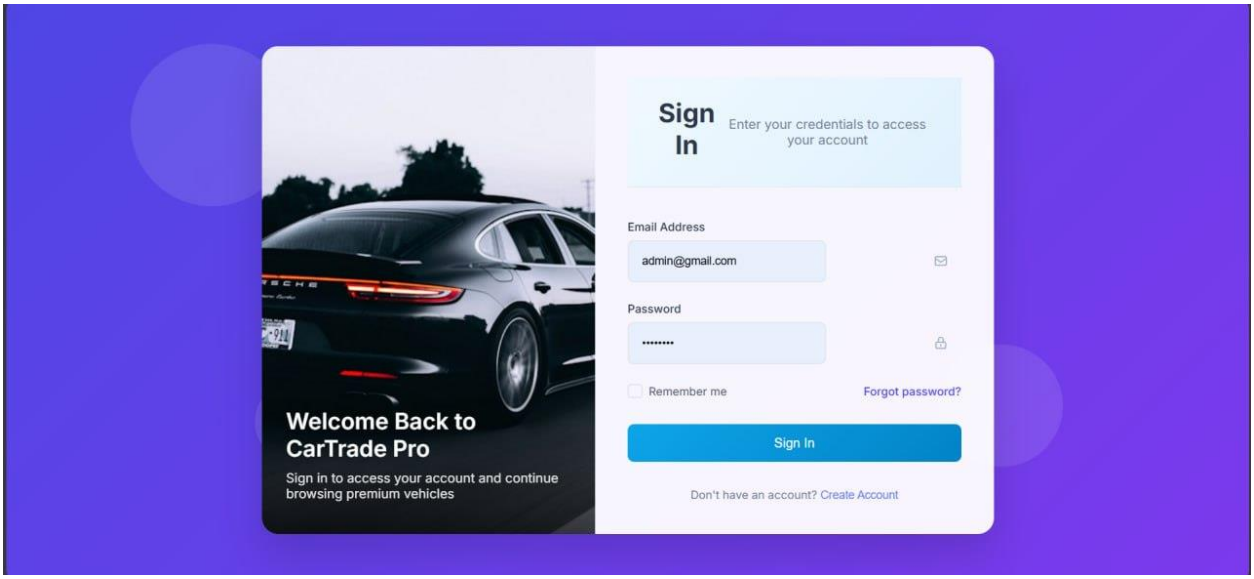
Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging . We use it to develop the whole project (frontend, backend).

3. System interfaces :

3.1 Register

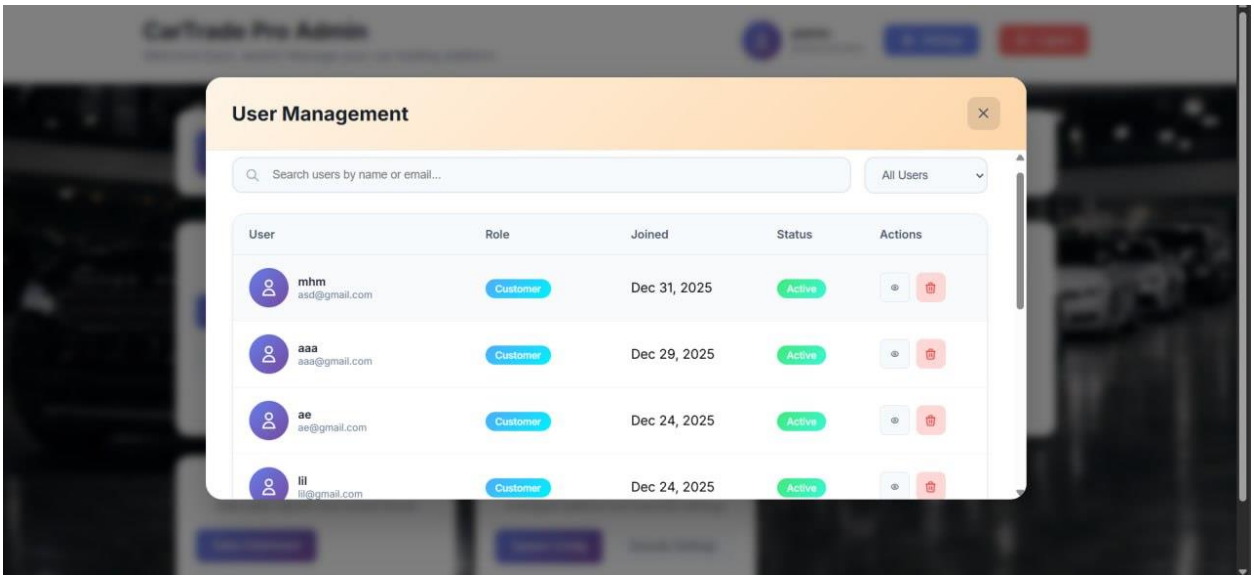


3.2 Log in:

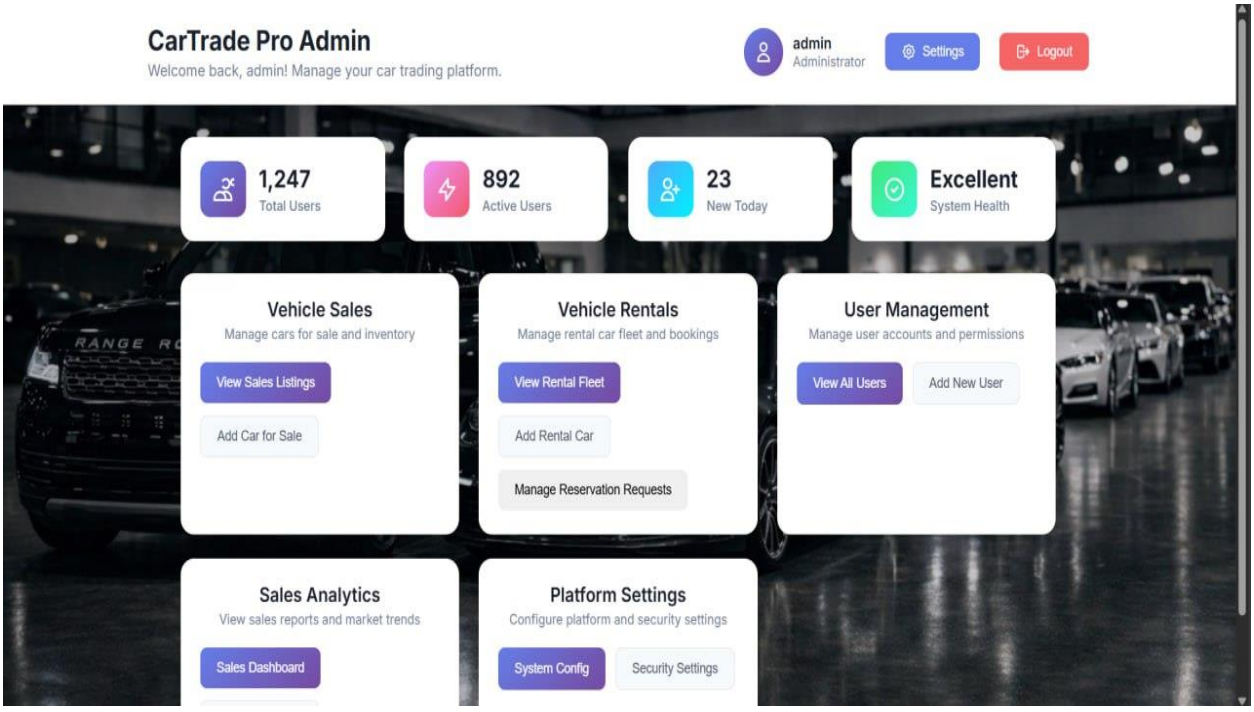


4. Admin's interfaces:

4.1 Users' management:

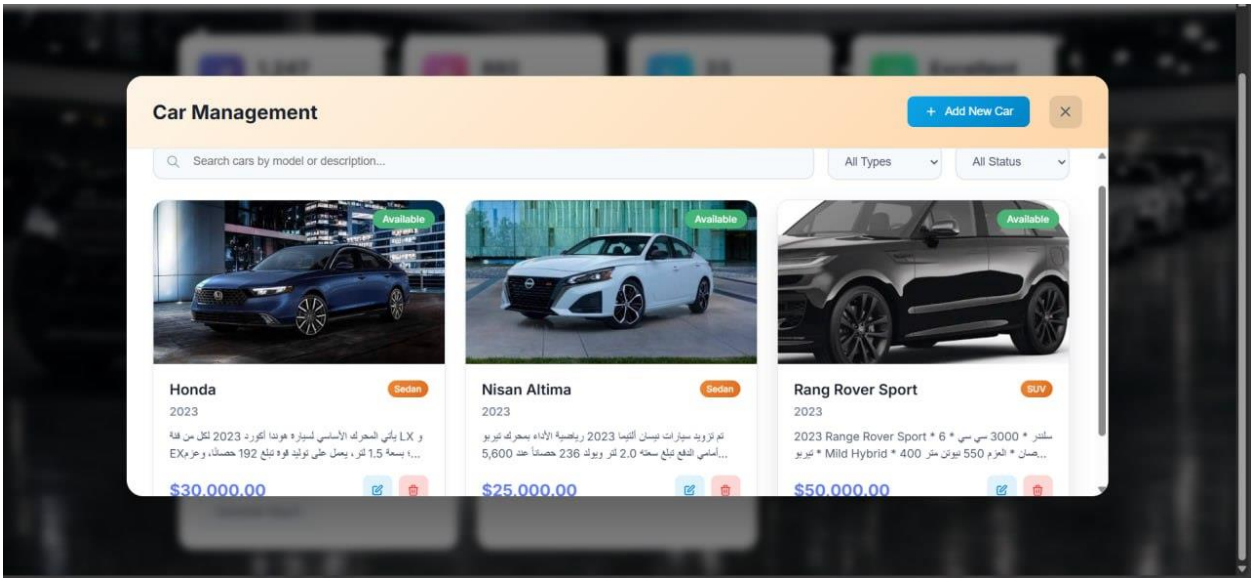


4.2 Admin dashboard:

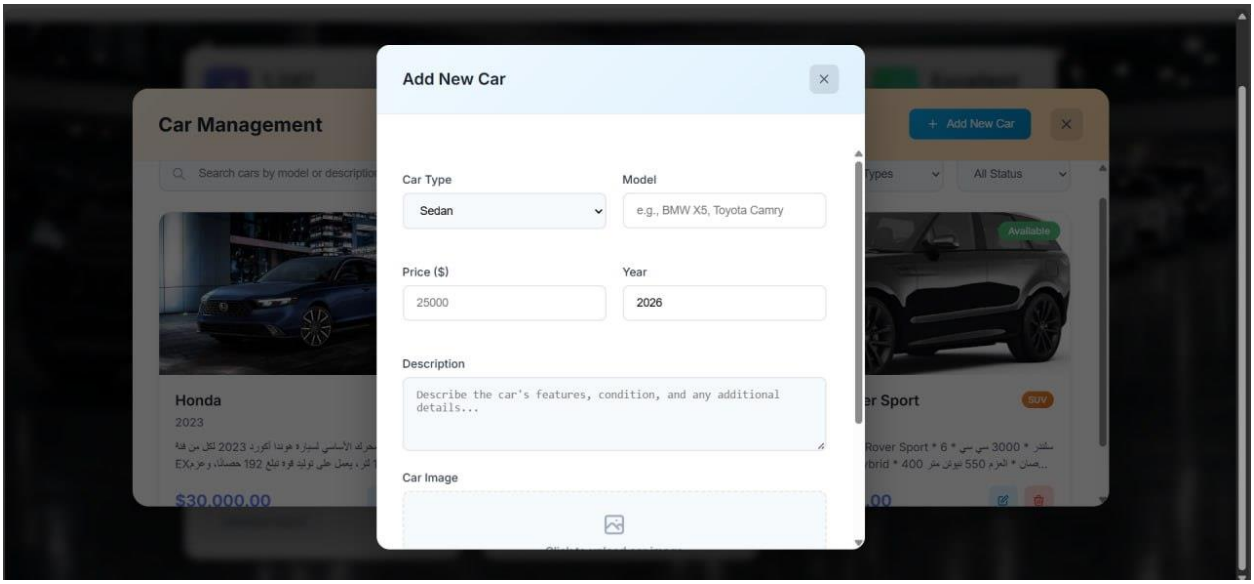


5. Car management :

5.1 View:

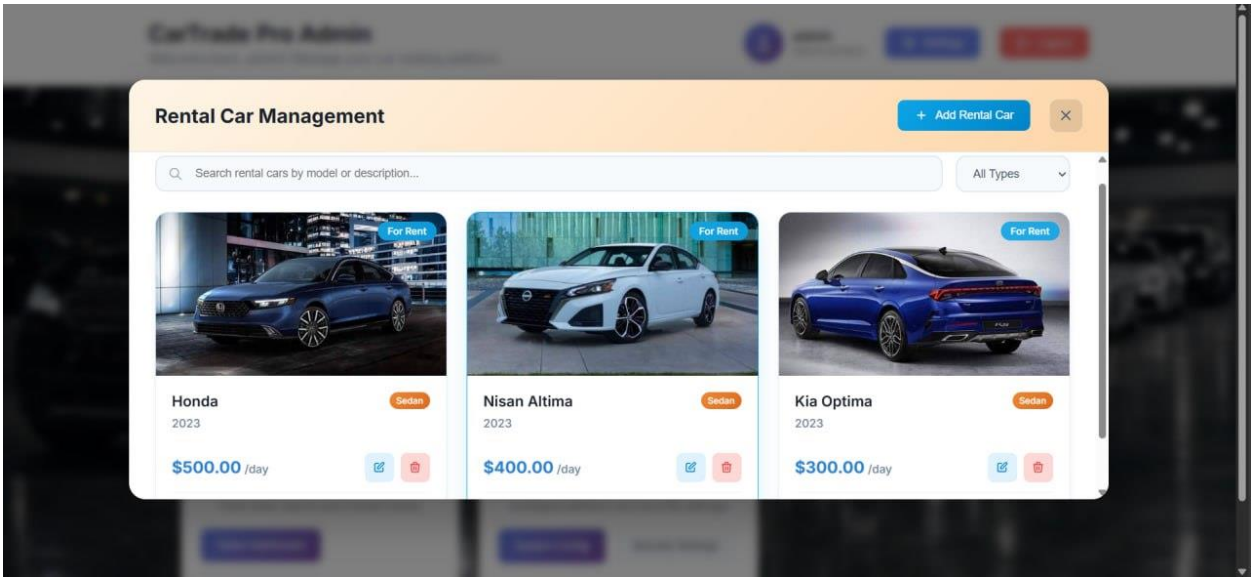


5.2 Add new car:

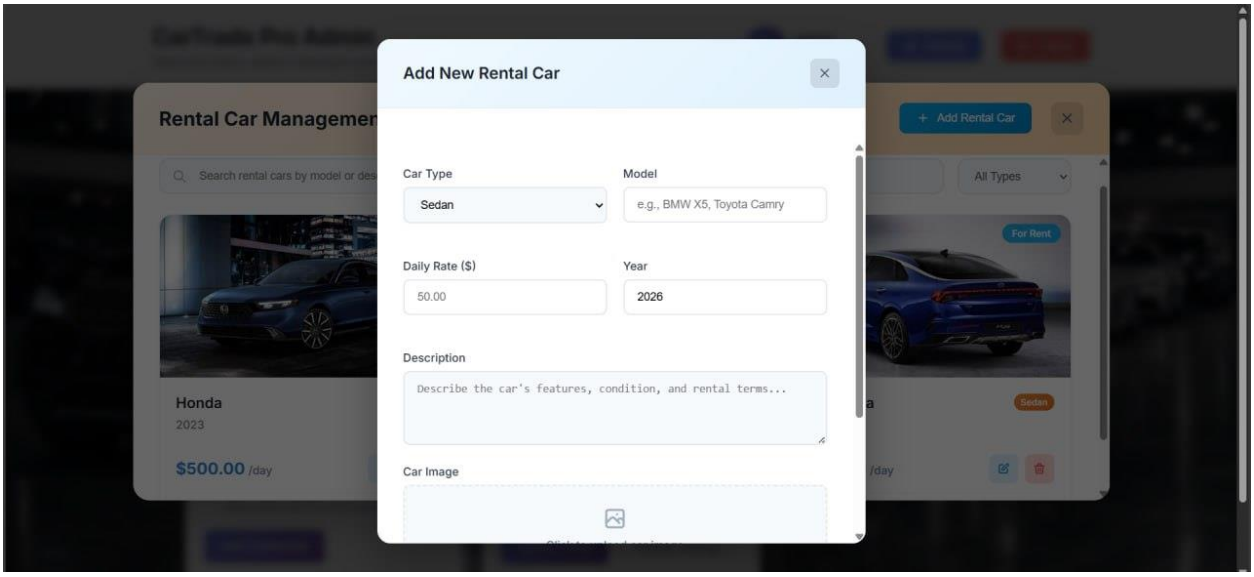


6. Rental car management:

6.1 View car rental:

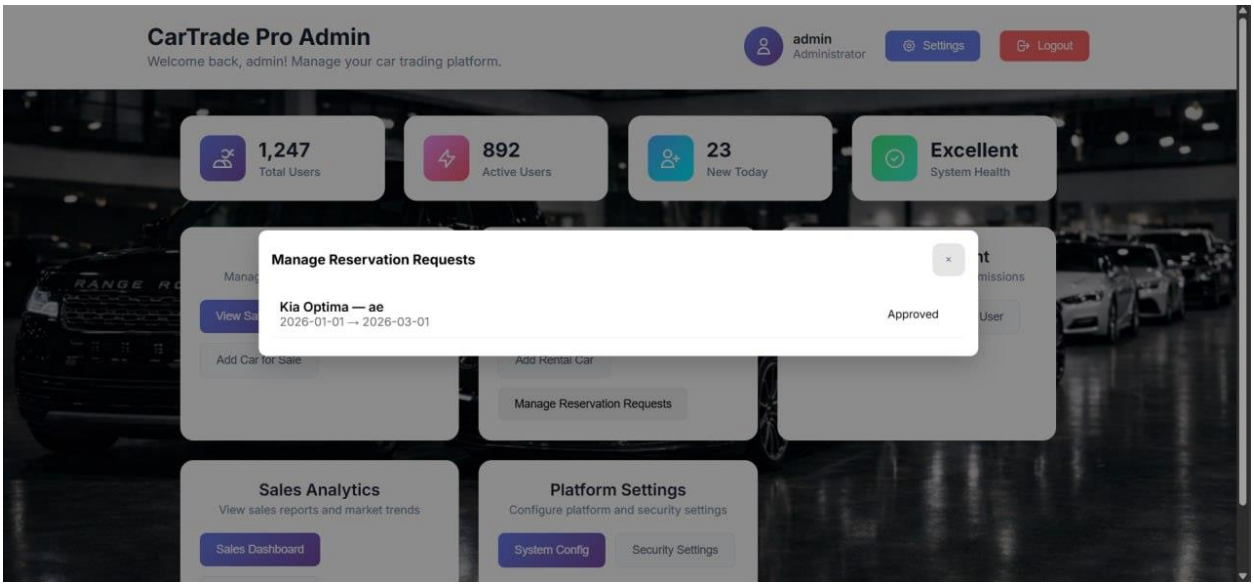


6.2 Add new car rental:



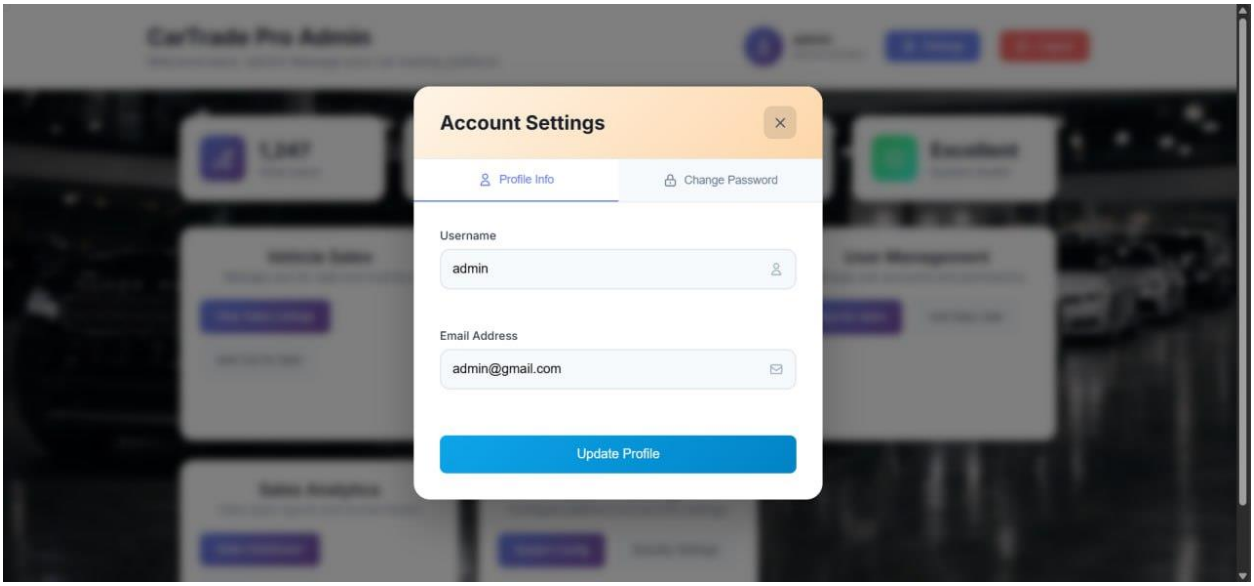
\

6.3 Manage reservation request:



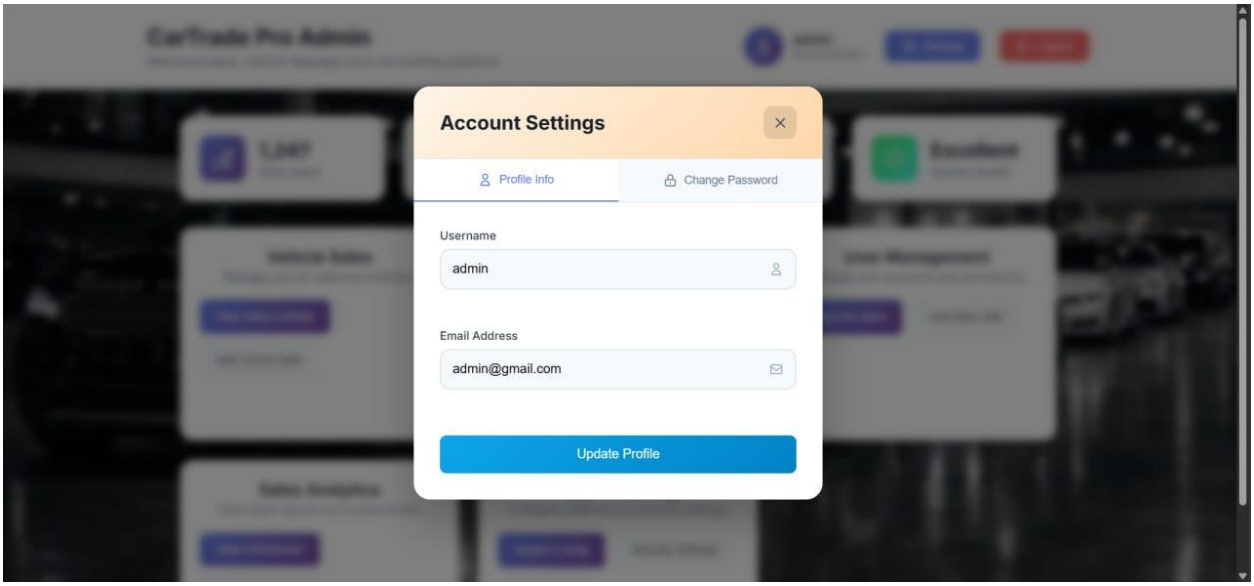
7. Account settings:

7.1 Profile info:

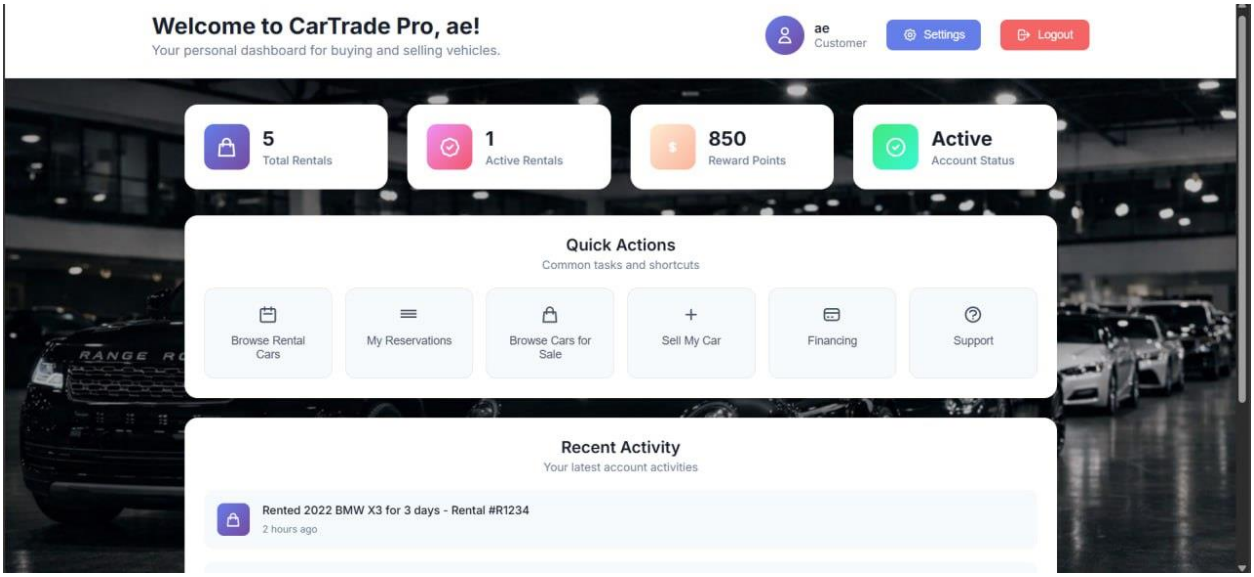


8. Account settings:

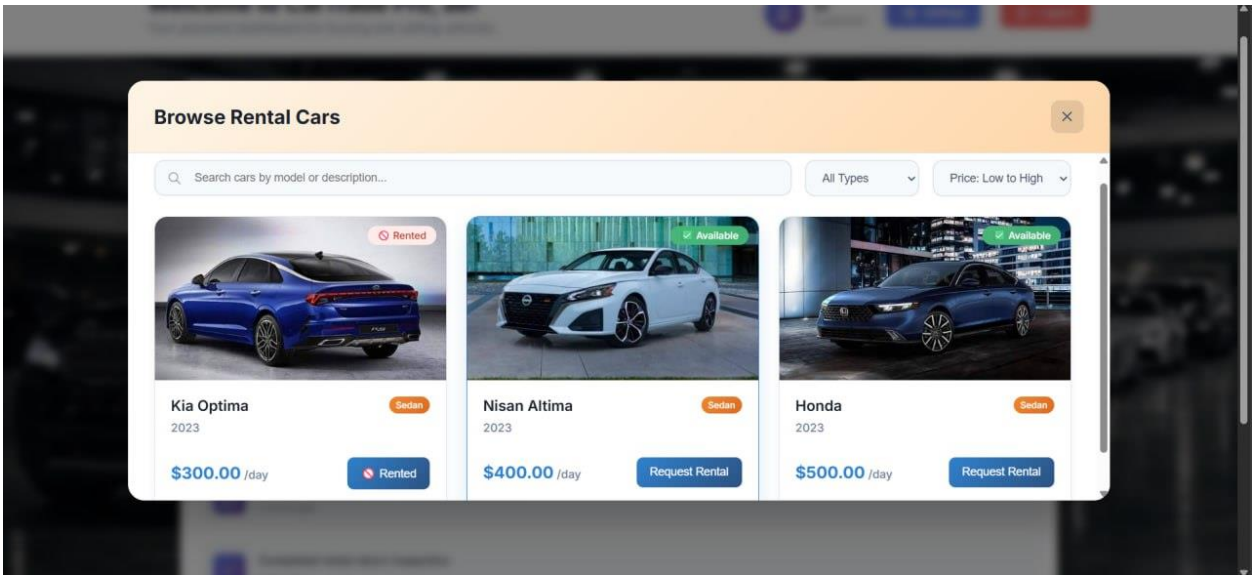
8.1 Change password:



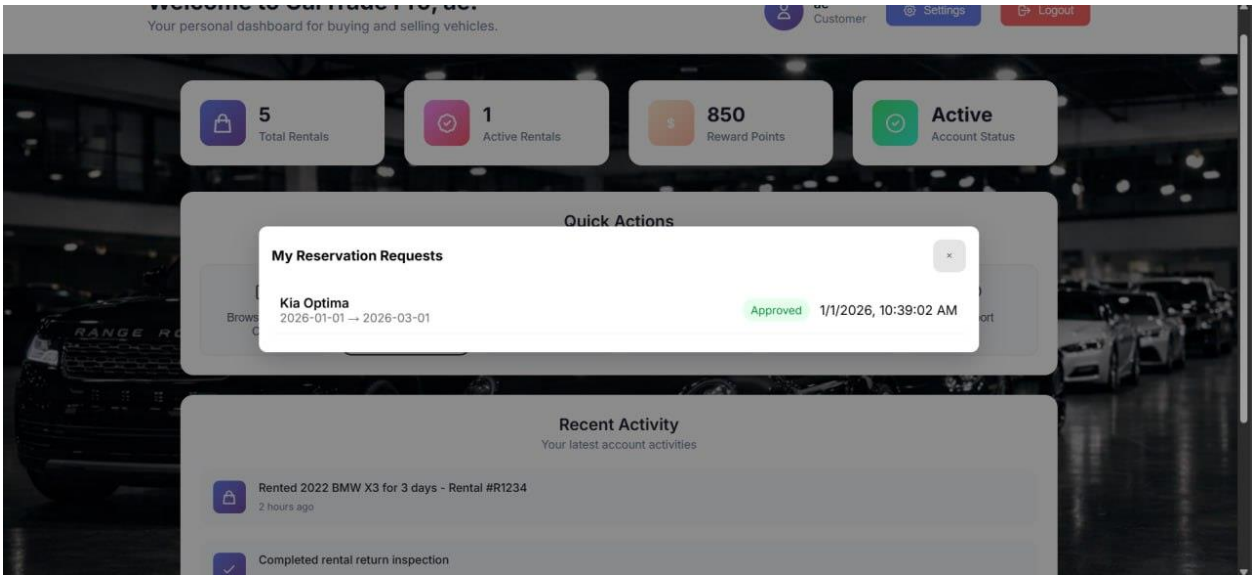
8.2 Customer interface:



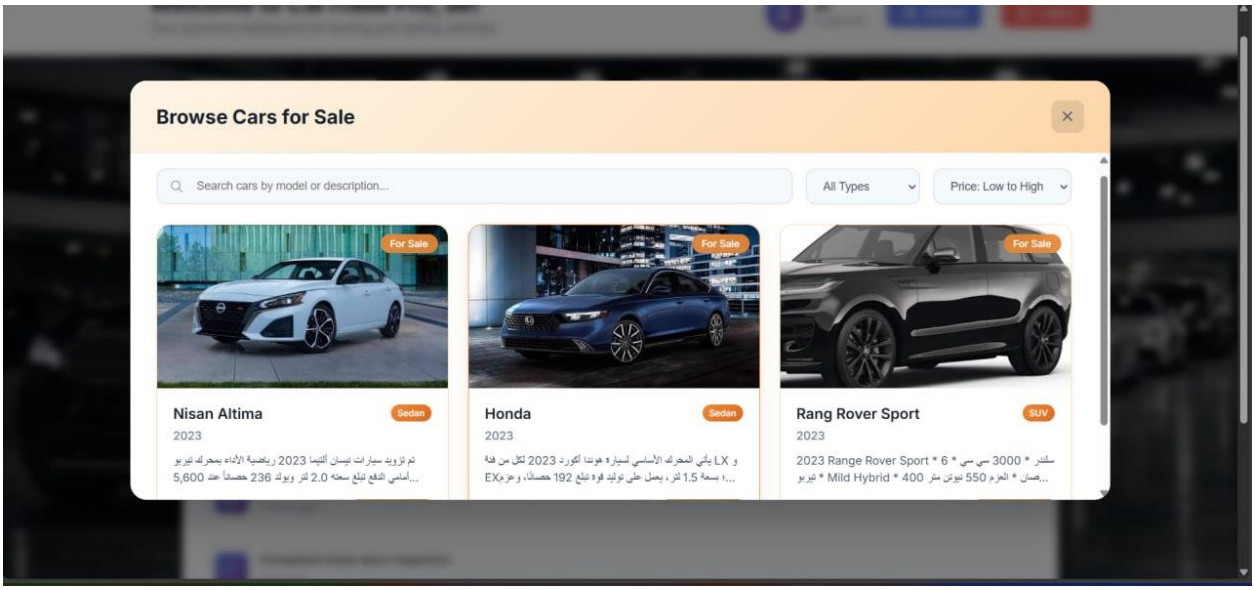
8.3 View rent car:



8.4 My reservations requests:

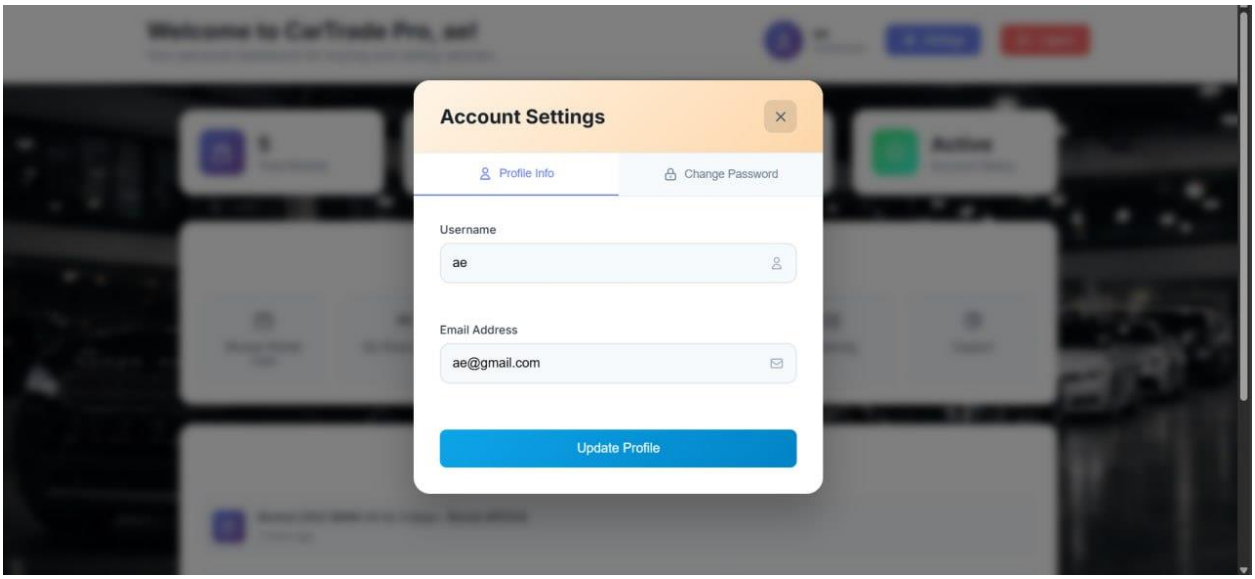


8.5 View car for sales:



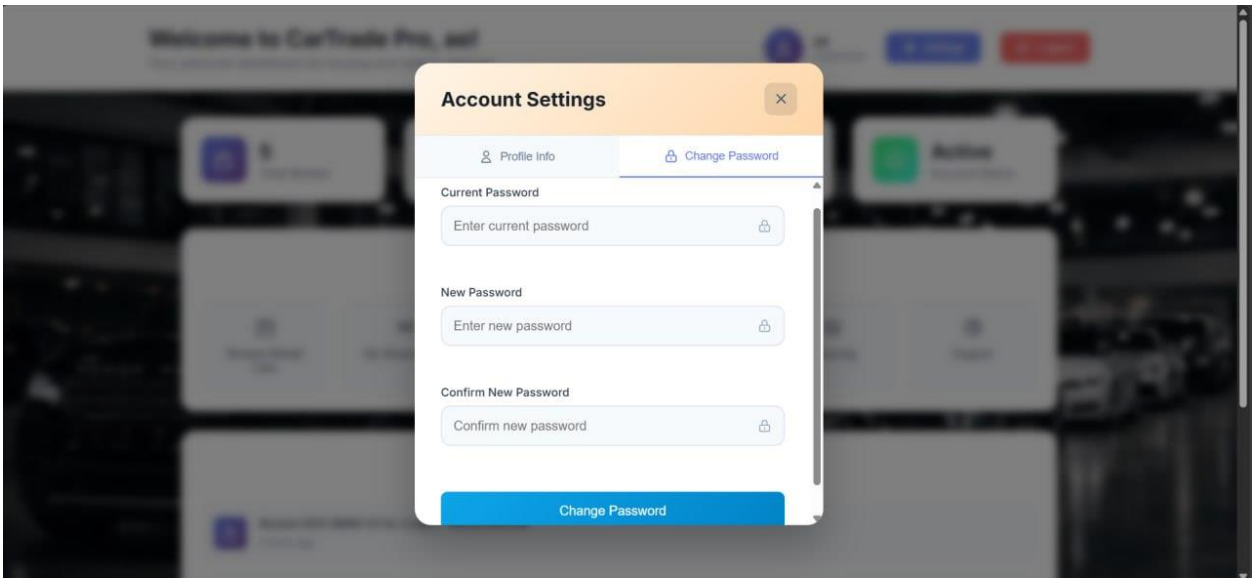
9. Account settings:

9.1 Profile info:

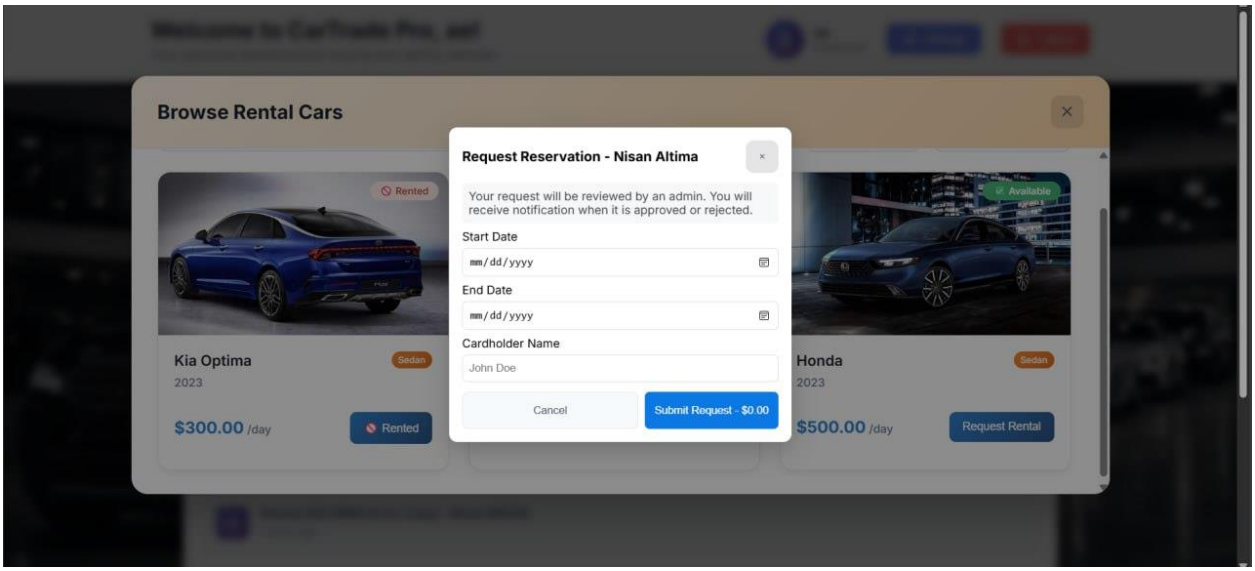


10.Account settings:

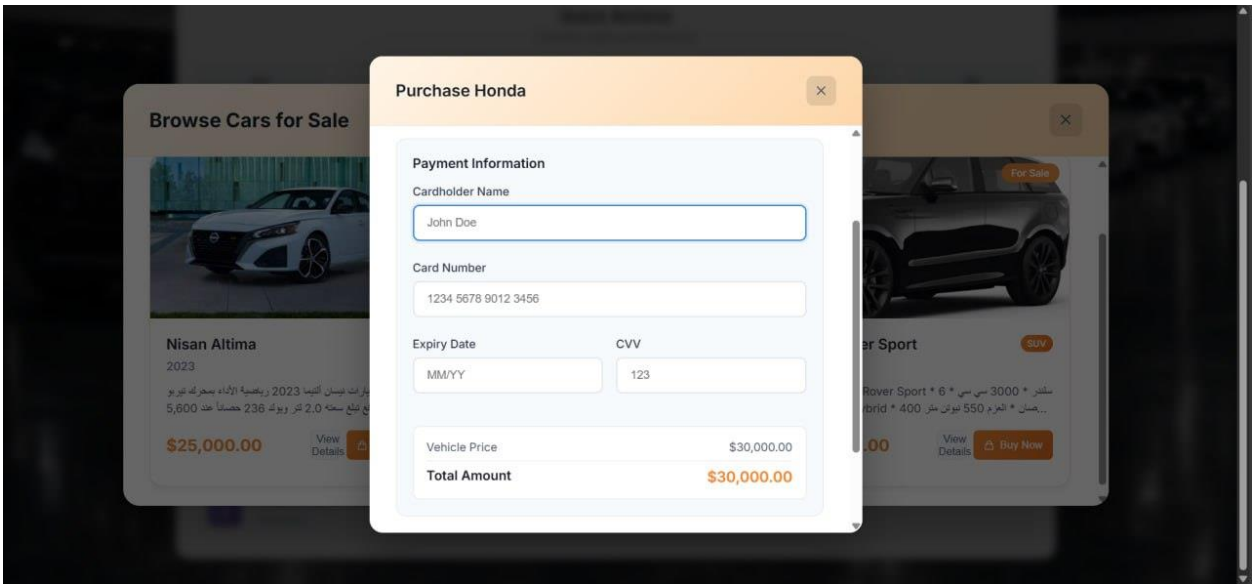
10.1 Change password:



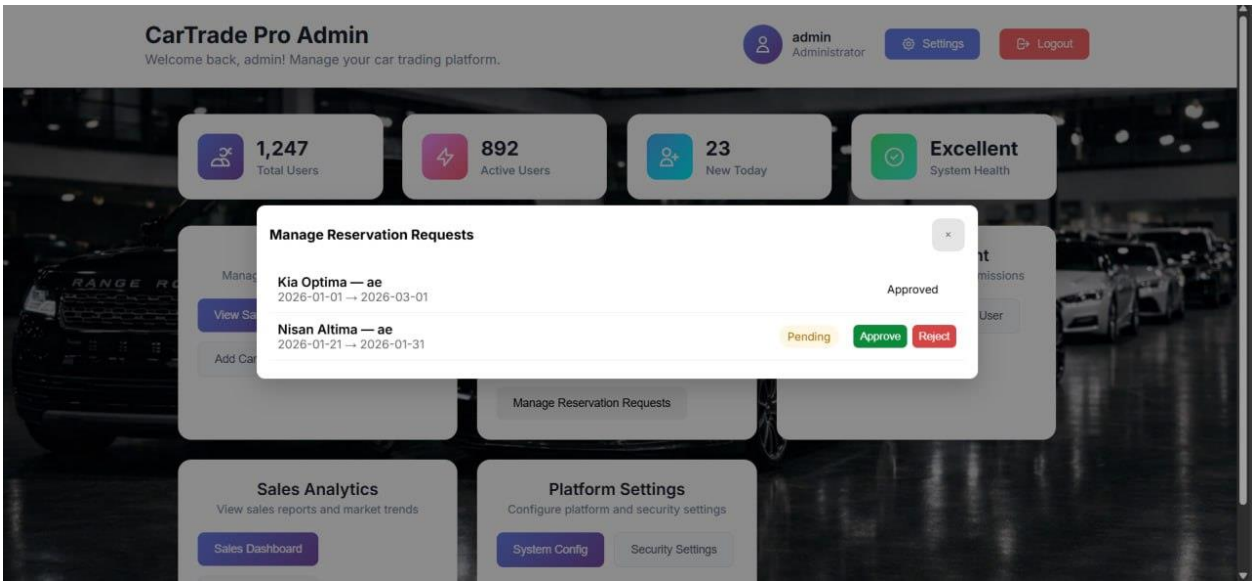
10.2 Rent car and reqest:



10.3 Buy car:



10.4 Manage reservation request:



11.Table 32 Test Cases Execution :

TC ID	Test Case Title	Tested Data	Expected Result	Actual Result	Pass/Fail
TC-01	Display user dashboard	—	Dashboard displayed successfully	Dashboard displayed successfully	Pass
TC-02	Register new client	Valid client data	Client registered successfully	Client registered successfully	Pass
TC-03	Register client with existing email	Duplicate email	Error message displayed	Error message displayed	Pass
TC-04	Register client with missing fields	Missing phone	Error: fields required	Error shown	Pass
TC-05	Login with valid credentials	Valid email/password	Login successful	Login successful	Pass
TC-06	Login with invalid credentials	Wrong password	Error message	Error message	Pass
TC-07	Add new car for sale	Valid car data	Car added successfully	Car added successfully	Pass
TC-08	Add new car for rent	Valid car data	Car added successfully	Car added successfully	Pass
TC-09	Add car with missing fields	Missing price	Error message	Error message	Pass
TC-10	Update car information	Valid update data	Car updated successfully	Car updated successfully	Pass
TC-11	Delete car	—	Car deleted successfully	Car deleted successfully	Pass
TC-12	View all cars	—	Cars list displayed	Cars list displayed	Pass
TC-13	Create reservation	Valid reservation dates	Reservation created	Reservation created	Pass
TC-14	Create reservation with invalid dates	End date < start date	Error message	Error message	Pass

TC-15	Approve reservation	Valid reservation	Reservation approved	Reservation approved	Pass
TC-16	Reject reservation	Valid reservation	Reservation rejected	Reservation rejected	Pass
TC-17	Create rental from reservation	Approved reservation	Rental created	Rental created	Pass
TC-18	Calculate rental cost	Rental days & price	Cost calculated correctly	Cost calculated correctly	Pass
TC-19	View user reservations	—	Reservations displayed	Reservations displayed	Pass
TC-20	View user rentals	—	Rentals displayed	Rentals displayed	Pass
TC-21	Update user profile	Valid data	Profile updated	Profile updated	Pass
TC-22	Update profile invalid	Invalid email	Error message	Error message	Pass
TC-23	Logout	—	Logout successful	Logout successful	Pass

12. Table 33 Final Requirement Traceability Matrix (RTM)

Car Showroom Management System

Req_ID	Requirement Description	SRS Section	System Design	Analysis (Use Case)	Detailed Design (Class)	Coding (Module/File)	Test Cases	Changed Requests No
RE-FR-US-1.1	Admin can create a new user account	SRS 3.1.1	User UI	UC-01	User	accounts/views.py	TC-01–TC-05	—
RE-FR-US-1.2	Admin can edit user information	SRS 3.1.2	User UI	UC-02	User	accounts/views.py	TC-06–TC-07	—
RE-FR-US-1.3	Admin can delete a user	SRS 3.1.3	User UI	UC-03	User	accounts/views.py	TC-08	—

RE-FR-US-1.4	System assigns default roles automatically	SRS 3.1.4	RBAC	UC-04	User	accounts/models.py	TC-02, TC-50	—
RE-FR-AUTH-2.1	User registration	SRS 3.2.1	Auth UI	UC-05	AuthService	accounts/views.py	TC-50–TC-54	—
RE-FR-AUTH-2.2	User login	SRS 3.2.2	Auth UI	UC-06	AuthService	accounts/views.py	TC-47–TC-49	—
RE-FR-AUTH-2.3	User logout	SRS 3.2.3	Auth UI	UC-07	AuthService	accounts/views.py	TC-23	—
RE-FR-CAR-3.1	Admin can add car for sale	SRS 3.3.1	Car UI	UC-08	CarForSale	cars/views.py	TC-07	—
RE-FR-CAR-3.2	Admin can add car for rent	SRS 3.3.2	Car UI	UC-09	CarForRent	cars/views.py	TC-08	—
RE-FR-CAR-3.3	Admin can update car info	SRS 3.3.3	Car UI	UC-10	Car	cars/views.py	TC-10	—
RE-FR-CAR-3.4	Admin can delete car	SRS 3.3.4	Car UI	UC-11	Car	cars/views.py	TC-11	—
RE-FR-CAR-3.5	Users can view cars list	SRS 3.3.5	Car UI	UC-12	Car	cars/views.py	TC-12	—
RE-FR-RES-4.1	Client can create reservation	SRS 3.4.1	Reservation UI	UC-13	Reservation	reservations/views.py	TC-13	—

RE-FR-RES-4.2	Admin can approve reservation	SRS 3.4.2	Reservation UI	UC-14	Reservation	reservations/views.py	TC-15	—
RE-FR-RES-4.3	Admin can reject reservation	SRS 3.4.3	Reservation UI	UC-15	Reservation	reservations/views.py	TC-16	—
RE-FR-RES-4.4	Users can view their reservations	SRS 3.4.4	Reservation UI	UC-16	Reservation	reservations/views.py	TC-19	—
RE-FR-REN-5.1	Create rental from reservation	SRS 3.5.1	Rental UI	UC-17	Rental	rentals/views.py	TC-17	—
RE-FR-REN-5.2	Calculate rental cost	SRS 3.5.2	Rental Logic	UC-18	Rental	rentals/services.py	TC-18	—
RE-FR-REN-5.3	Users can view rentals	SRS 3.5.3	Rental UI	UC-19	Rental	rentals/views.py	TC-20	—
RE-NFR-SC-6.1	System scalability	SRS 4.1.1	Architecture	—	Deployment	—	Performance Test	—
RE-NFR-AV-6.2	System availability	SRS 4.1.2	Architecture	—	Server Setup	—	Availability Test	—
RE-NFR-SEC-6.3	Secure authentication	SRS 4.2.1	Security	—	Auth Module	accounts/auth.py	Security Test	—

RE-NFR-SEC-6.4	Password encryption	SRS 4.2.2	Security	—	Hashing	accounts/utils.py	Security Test	—
RE-NFR-US-6.5	Usability	SRS 4.3.1	UI Design	—	Frontend	—	Usability Test	—
RE-NFR-EX-6.6	Extensibility	SRS 4.4.1	Modular Design	—	Service Layer	—	Integration Test	—

Chapter 7 Report Overview

1. Introduction

This chapter provides an overview of the report structure and explains the purpose of each chapter. It aims to guide the reader through the logical flow of the report, ensuring a clear understanding of how the different sections contribute to the overall study and the development of the proposed system.

2. Report Structure and Purposes

- **Chapter 1: Introduction**

  This chapter establishes the foundation of the report by introducing the Car Showroom Management System project. It presents the problem statement, defines the project objectives, provides an overview of the proposed system, and explains how the report is structured.

- **Chapter 2: Fundamental Concepts and Literature Review**

  This chapter presents the theoretical background and fundamental concepts related to the Car Showroom Management System. It also includes a literature review of existing and similar systems, technologies, and methodologies that influenced the design and development of the project.

- **Chapter 3: Project Management**

  This chapter outlines the project management aspects of the Car Showroom Management System, including the project charter, Scope of Work (SOW), Gantt chart for project scheduling, and risk management strategies. It ensures that the project is well planned, structured, and efficiently managed.

- **Chapter 4: System Analysis**

  This chapter focuses on the system analysis phase, starting with the project timeline, followed by the Software Requirements Specification (SRS). It also includes requirements modeling, initial test cases, and the Initial Requirement Traceability Matrix (RTM), ensuring a comprehensive analysis of the system requirements.

- **Chapter 5: System Design**

  This chapter provides a detailed explanation of the system design, including the overall system architecture and the detailed design of system components. It serves as a blueprint for implementing the Car Showroom Management System.

- **Chapter 6: Practical Implementation**

  This chapter describes the practical implementation of the system, including the tools and technologies used, system interfaces, functional components, and test case execution. It also presents the Final Requirement Traceability Matrix (RTM), demonstrating the alignment between system requirements and the actual implementation.

3. Summary

This chapter summarizes the structure of the report and clarifies how each section contributes to the development and documentation of the Integrated Car Center Management System.

The organization of the report ensures a logical progression from identifying the project problem and requirements to system design, implementation, and testing. Through this structured approach, the report provides a comprehensive and coherent presentation of the system, highlighting both the technical and practical aspects of the project.