

## **PROJECT REPORT**

ON

**Rider's Rent**

Submitted in partial fulfillment for the degree of  
**Bachelor of Science in Information Technology**



**University of Mumbai**

Submitted By

**Gaurav Arun Joshi**

**Examination Seat No. - \_\_\_\_\_**

**Amit Pappu Kannaujiya**

**Examination Seat No. - \_\_\_\_\_**

Under the Guidance of

**Asst. Prof. Sumeet M. Rathod**

**Academic Year 2024-25**



**THAKUR RAMNARAYAN COLLEGE OF ARTS AND COMMERCE**

**Dahisar - (East), Mumbai.**



*Thakur Educational Trust's (Regd.)*

**THAKUR RAMNARAYAN  
COLLEGE OF ARTS & COMMERCE**

NAAC Accredited & ISO 21001:2018 Certified



Thakur Ramnarayan Educational Campus, S.V. Road, Dahisar (East), Mumbai - 400 068  
Tel: 022 2828 1200 | Fax: 022 2828 1300 | [www.trcac.org.in](http://www.trcac.org.in)

# CERTIFICATE

This is to certify that Mr./Ms. \_

Class \_\_\_\_\_ Roll No. \_\_\_\_\_ Examination Seat No. \_\_\_\_\_

has satisfactorily completed the project implementation for

\_\_\_\_\_

in the department of \_\_\_\_\_ during the academic

year \_\_\_\_\_.

**Internal Examiner**

**Principal**

**Extern Examiner**

**Date - \_**

Rider's Rent

T.Y.B.Sc.(I.T)

### **Acknowledgement**

Success is not just about achieving your goals, but about embracing the journey that led you there.

I would like to express my sincere gratitude to list of individuals or organizations for their invaluable support and contributions to the completion of this project. Their guidance, expertise, and encouragement have been instrumental in its success.

We take this opportunity to express our profound gratitude to the management of **THAKUR RAMNARAYAN COLLEGE OF ARTS & COMMERCE** for giving us the opportunity to accomplish this project work.

We are very thankful to **Dr. Sumati Rajkumar**, Principal for His kind cooperation in the completion of my project.

We are also grateful to my Project **Guide Asst. Prof. Vishal Sharma, B.Sc. (I.T.) Programme Coordinator** for being very much resourceful, kind and helpful. Their positive attitude, forceful personality, unassailable optimism and unwavering faith in us assured that we come out of the woods whenever we come out of difficulties.

It is our earnest endeavours to express our sincere thanks to the faculty for their kind co-operation, help and unending support.

We would also like to acknowledge the gratitude and the support extended towards us by Thakur Ramnarayan College of Arts & Commerce for giving us this opportunity to work on this project and for timely guidance and infrastructure.

Finally, we wish to thank all our friends and the entire I.T. department who directly or indirectly helped us in the completion of the project and to our family without whose support, motivation and encouragement this would not have been possible.

**CONTENTS**

Topic No	Topic	Page No
1	<b>1.Introduction</b>	
1.1	Existing System	
1.2	Proposed System	
1.3	Advantage of Proposed System	
1.4	Problem Definition	
1.5	Scope of Project	
1.6	Theoretical Background	
2	<b>2.Analysis</b>	
2.1	Methodology Adopted	
2.2	Pert Chart	
2.3	Gantt Chart	
2.4	User Requirements	
2.5	Feasibility Study	
2.6	Fact Finding Techniques	
2.7	Spiral Model	
3	<b>3.Design</b>	
3.1	Entity Relationship Diagram	
3.2	Data Flow Diagram	
3.3	System Flow Chart	
3.4	Use Cases	
3.5	Sequence Diagram	
3.6	Event Table	
3,7	Data Dictionary	
3.8	Menu Tree	
3.9	Screen Shot	
4	<b>4. Coding</b>	
4.1	Pseudo Code	

5	<b>5.Test Procedure and Implementation</b>	
5.1	Testing	
5.2	Test Code	
5.3	Black Box Testing	
5.4	Implementation	
5.5	Hardware & Software Requirement	
6	<b>6. Conclusion</b>	
7	<b>7. Appendix</b>	
7.1	System Control	
7.2	Operational Manual	
7.3	User Manual	
7.4	Future Enhancement	
7.5	List of Tables & Forms	
7.6	Variable Naming Conventions	
	<b>8. Bibliography</b>	

# **INTRODUCTION**

- In today's fast-paced world, the demand for rental cars has seen a significant rise, driven by both individuals and businesses . Car rental companies are tasked with variety of operations like vehicle bookings, inventory tracking, customer management, and financial transactions. Traditionally, these tasks were handled manually or with outdated systems, leading to inefficiencies, errors, and a lack of real-time updates.
- Manual data entry processes increase the chances of errors in recording customer details, booking information, and payment transactions. These inaccuracies can lead to disputes or customer dissatisfaction.
- In many traditional car rental businesses, client must visit the rental office in person or make reservations over the phone, leading to delays and inconveniences.
- Without an online platform Car agencies often rely on physical documents to store customer information, rental agreements, and vehicle records, making it difficult to manage and prone to human error.
- In response to these challenges, many companies are adopting automated systems to streamline their car rental operations. A well-designed Car Rental Management System that can integrate various aspects of rental management—inventory, booking, billing, and reporting—into one cohesive solution.



## **1.1 Existing System**

The current car rental management systems used by many agencies are either manual or rely on fragmented, outdated software tools. These systems often struggle to meet the growing demands of modern customers and operational complexities.

Following are the challenges

### **1. Manual Record-Keeping**

Many small or mid-sized car rental agencies still use paper forms or spreadsheets to log vehicle details, rentals, and customer information. This approach is time-consuming and prone to human error.

### **2. Lack of Automation**

Manual processes require significant effort and time to track vehicle availability, rental bookings, and customer details.

### **3. Data Inaccuracy:**

Human errors in data entry can lead to inconsistencies, such as double bookings or incorrect billing.

### **4. Inadequate Reporting and Analytics**

Existing systems often fail to provide insights into fleet utilization, revenue trends or customer preferences, which are crucial for decision-making.

### **5. Inefficient Maintenance Tracking**

Vehicle maintenance schedules are poorly tracked, leading to increased wear and tears, unplanned downtime, and higher maintenance costs.

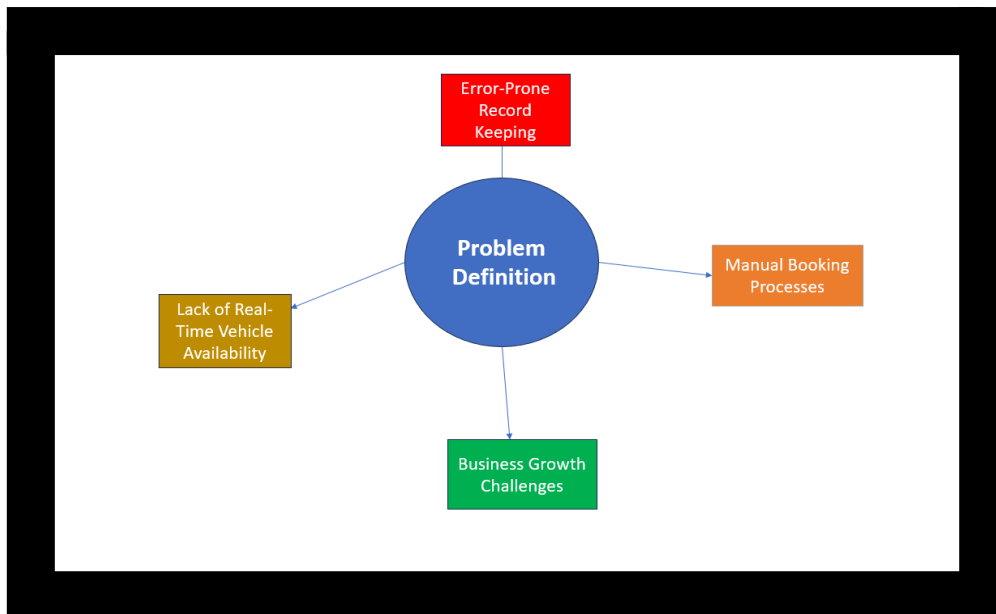
## **1.2 Proposed System**

- The System includes User Login & Authentication for verification
- Provide an easy- to- use online platform where clients can browse available vehicle view prices and make reservations anytime.
- Instead of paper- based records, the Proposed System will store all the client data in a secure digital database.
- With the implementation of an online platform, clients can access the system 24/7 , allowing them to make reservations, check availability of car & view rent of it.
- Client's will be able to create accounts, where they can save personal details, view past rentals, check upcoming bookings, and receive special offers.
- The proposed system will offer multi-channel customer support, including chat, email, and phone. Customers will be able to resolve issues and ask questions directly through the platform, improving communication and service quality
- Simple UX/UI easily accessible to any customer.

### **1.3 Advantage of Proposed System**

- No paperwork. All data entry is computerized.
- Automated booking, payments, and inventory management reduces manual work.
- The system provides real-time tracking of vehicle availability, preventing double bookings and optimizing fleet utilization.
- Data will be secured and protected.
- Calculation of Data is Automatically counted.
- The system provides real-time tracking of vehicle availability, preventing double bookings.

### 1.4 Problem Definition



The problem with existing car rental systems is their reliance on manual processes, outdated technology, and lack of customer convenience. These limitations result in inefficient operations, high error rates, poor fleet management, and customer dissatisfaction. To address these issues, a modern, integrated car rental system is needed, offering real-time vehicle tracking, automated bookings, secure payment options, and improved customer accessibility, ultimately driving business growth and enhancing operational efficiency.

## **1.5 Scope of Project**

### **The objectives of scope of project are as follows :-**

- User Registration and Account Management
- A user-friendly interface for customers to browse available vehicles, check pricing, and make reservations online
- Integrated support channels, including chat, email, and phone support, to assist customers with inquiries or issues.
- The Proposed system should be designed to handle an increasing number of users and vehicles as the business grows.
- A user-friendly interface that is easy to navigate for both customers and administrators, ensuring a positive experience.
- Implementing loyalty programs or referral systems to encourage repeat business and customer engagement

## **Theoretical Background**



- C# developed by Microsoft, it is widely used in enterprise applications, offering a powerful combination of object-oriented programming features and a strong type system.
- It integrates well with the .NET ecosystem and provides comprehensive libraries for various functionalities.
- It follows an OOP paradigm, which helps in developing a well-structured and maintainable codebase, crucial for large systems like a Car Rental Management System.
- Using .NET Core, the system can be deployed across multiple platforms, including Windows, Linux, and macOS, providing flexibility in deployment.
- ASP.NET Core is the web framework chosen for this project. It is a free, open-source framework for building modern web applications.
- ASP.NET Core is known for its high performance and low memory usage, making it ideal for high-traffic systems.



- For Database Management , Microsoft SQL Server was selected.
- Microsoft SQL integrates well with C# and ASP.NET Core, allowing for efficient database management.
- It offers excellent performance for transactional systems, which is crucial for managing vehicle inventory, customer details, and booking records.
- Entity Framework Core (EF Core) is used as the Object-Relational Mapping (ORM) tool for database access in this project. EF Core simplifies data manipulation by allowing developers to work with C# objects instead of raw SQL queries.
- EF Core ensures that database interactions are type-safe, reducing runtime errors.

# **ANALYSIS**



## **2.1 Methodology Adopted**

Methodology: Modules and their Description.

### **1. Planning**

We started by gathering and analysing the business requirements for the car rental system. The goal was to design an application that supports key features like car reservations, customer management, payment processing, and reporting.

### **2. Technology Selection**

- Frontend: ASP.NET for creating a dynamic and user-friendly web interface.
- Backend: C# for business logic and server-side operations.
- Database: SQL Server for efficient and secure data storage and management.
- IDE: Visual Studio for seamless development and debugging.

### **3. System Design**

The system was designed based on a layered architecture to ensure maintainability and scalability:

- Presentation Layer: Handles the user interface (UI).
- Business Logic Layer: Manages application logic and rules using C#.
- Data Access Layer: Handles interactions with the SQL Server database.

### **4. Database Design**

We created a relational database in SQL Server with tables such as:

- Customers: Stores customer information.
- Cars: Maintains car details (make, model, availability).
- Reservations: Tracks car booking information.
- Payments: Stores payment records.

### **5. Development Process**

- Backend Development:
  - ◆ Created methods in C# to handle (Create, Read, Update, Delete).
  - ◆ Implemented validation logic for data integrity.

- Frontend Development:
  - ◆ Built forms and pages using ASP.NET Web Forms or MVC pattern for a user-friendly experience.
- Database Integration:
  - ◆ Established connections to SQL Server using ADO.NET and Entity Framework.

## **6. Security Implementation**

- Encryption for sensitive data like passwords.

## **7. Testing**

We performed:

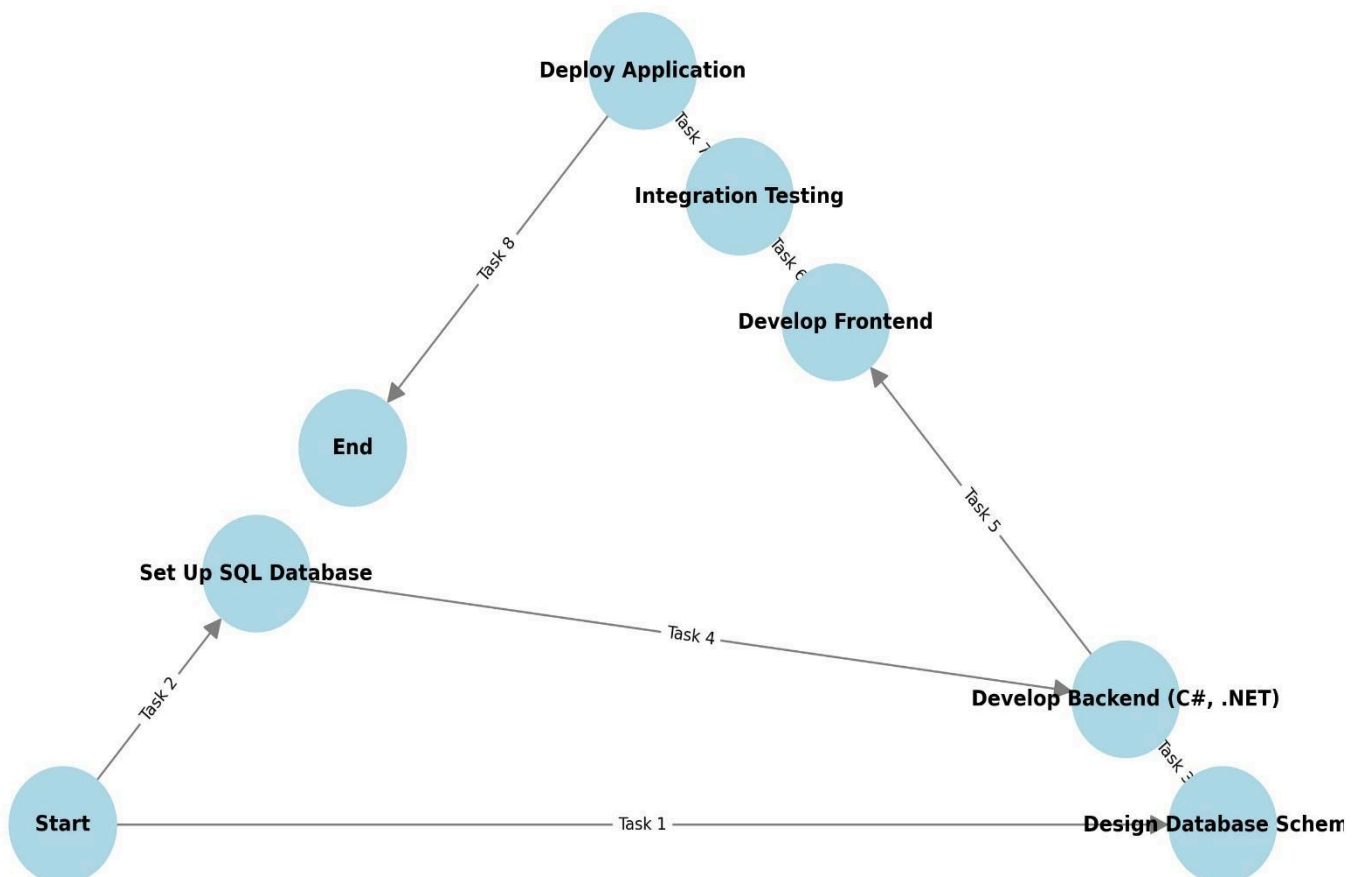
## **8. Deployment**

The application was deployed on a web server (IIS) for users to access the system via browsers.

## **9. Maintenance and Future Enhancements**

Regular monitoring and updates are planned for bug fixes and adding new features like a mobile app interface or advanced reporting options.

- Program Evaluation and Review Technique) is a project management tool used to plan, schedule, and control tasks within a project.\
- It visually represents the flow of activities and helps to identify the critical path—the sequence of tasks that directly affects the project's completion time.



### **2.3 Gantt Chart**

- A Gantt chart is a visual project management tool used to plan, track, and manage tasks or activities over time.
- It displays the timeline of a project and shows when each task starts and ends, as well as their dependencies.

Task	Start Date	End Date
1. Requirement Gathering		
2. System Design		
3. Development		
4. Testing		
5. Development & Maintenance		

## **2.4 USER REQUIREMENT**

The Car Rental Management System is designed to address the needs of various user groups, including customers, administrators, and general. The following are the key requirements.

### **1. Customer Requirements**

- **Vehicle Browsing:**

Customers should be able to view available vehicles with details like make, model, price, and specifications.

- **Real-Time Availability:**

Users require real-time updates on vehicle availability to avoid booking conflicts.

- **Online Booking and Payment:**

The system should allow customers to book vehicles online and make secure payments through multiple payment methods.

- 

- **Booking History:**

Customers should be able to view their previous bookings and payment history.

### **2. Admin Requirement**

- **Booking Management:**

A centralized dashboard to monitor and manage all customer bookings, cancellations, and payments.

- **User Management:**

Features to manage customer accounts, Editing and managing vehicles.

### **3. General System Requirements**

- **Data Security:**

Secure handling of sensitive data, including customer information and payment details, is critical.

- **Customer Support Features:**

The ability to manage customer inquiries, complaints, and feedback efficiently.

## **2.5 FEASIBILITY STUDY**

A feasibility study helps evaluate whether developing a car rental system is practical and worthwhile, considering technical, economic, and operational factors.

### **1) Technical Feasibility**

The combination of C#,Microsoft SQL Server is well-suited for building secure, scalable, and efficient web applications.

provides robust tools for creating dynamic user interfaces.

Microsoft SQL Server ensures efficient data storage and retrieval for managing large amounts of information (customers, cars, bookings)

Visual Studio as the primary IDE for development.

Entity Framework for smooth database integration.

### **2) Cost Analysis**

Savings on licensing costs if using free versions (Community Editions) of Visual Studio and Microsoft SQL Server.

Reduced maintenance costs due to built-in features in the .NET ecosystem.

Hosting on IIS (Internet Information Services) minimizes additional expenses.

Scalable architecture reduces future upgrade costs.

Revenue generation through efficient car rentals and streamlined operations.

Improved customer experience can lead to increased bookings.

### **3) Operational Feasibility**

Web-based system accessible on multiple devices.

Simple and intuitive reservation and payment process.

Automating bookings and payment processing reduces administrative overhead.

Real-time data for better fleet management.

Ability to integrate with external payment gateways and third-party services for additional functionality.

#### **4) Schedule Feasibility**

The development timeline for the project is realistic.

A phased rollout approach can be adopted:

- Phase 1: Basic functionalities (login, registration, about us etc).
- Phase 2: Vehicle registration, Customer Registration
- Phase 3 : Payment and others staff
- Phase 4 : generating invoice and reports.

#### **5) Legal and Security Feasibility**

The platform adheres to data protection laws by securely handling customer data.

Role-based access control ensures secure operations.

## **2.6 FACT FINDING TECHNIQUES**

Fact-finding techniques are essential for gathering detailed information and requirements for your car rental project.

### **1. Interviews**

- **What it is:** Talking to people involved in the project, like clients, users, and stakeholders.
- **How to do it:** Prepare a list of questions, conduct face-to-face or virtual interviews, and take notes of the responses.

### **2. Surveys and Questionnaires**

- **What it is:** Distributing forms with questions to a large group of people.
- **How to do it:** Create online surveys, ask relevant questions about car rentals, and collect responses to analyze it later.

### **3. Observation**

- **What it is:** Watching how the current car rental process works in real-time.
- **How to do it:** Visit car rental locations, observe the workflow, and note down any important details about the process and apply it to project work.

### **4. Document Analysis**

- **What it is:** Reviewing existing documents related to car rentals.
- **How to do it:** Gather contracts, rental forms, and policies, and study them to understand the current system and requirements.

### **5. Workshops**

- **What it is:** Group sessions where stakeholders discuss the project requirements.
- **How to do it:** Organize workshops, facilitate discussions, and ensure all important aspects of car rental are covered.

### **6. Prototyping**

- **What it is:** Creating a simple version of the car rental system to get feedback.
- **How to do it:** Develop a basic prototype, test it, add required features and deploy it.



## **2.7 COCOMO MODEL**

The COCOMO (CONstructive COst MOdel) is a software cost estimation model developed by Barry Boehm in the early 1980s. The model provides a framework to estimate the effort, time, and cost involved in developing a software project based on its size and complexity. COCOMO is particularly useful for project managers and software engineers as it helps in making predictions about resource requirements and delivery schedules for software development projects.

The model is based on empirical data from numerous software projects and applies mathematical formulas to estimate project costs. It also considers factors like the complexity of the software, the experience of the development team, and the project environment.

This provides transparency to the model which allows software managers to understand why the model gives the estimates it does. Moreover, the baseline COCOMO originally underlies a waterfall model lifecycle. The developments are done on multiple COCOMO models in parallel for cost estimates that cover a broader scope that exceeds the boundaries of traditional software development are discussed. The COCOMO model has basically two parameters like effort calculation and development time to define the quality of any software product:

**Efforts calculation:** Efforts can be calculated by the number of persons required to complete the task successfully. It is calculated in the unit person-month.

**Development time:** the time that is required to complete the task. It is calculated in the unit of time like months, weeks, and days. It depends on the effort calculation, if the number of persons is greater then definitely the development time is low.

There are various types of models of COCOMO that have been proposed to check the correctness of the software products and to calculate the cost estimations at the different levels. These levels also depend on the strategies to develop accurate software products. These strategies are as follows:

Software projects under COCOMO model strategies are classified into 3 categories, organic, semi-detached, and embedded.

**Organic:-** A software project is said to be an organic type if-

- Project is small and simple.
- Project team is small with prior experience.
- The problem is well understood and has been solved in the past.

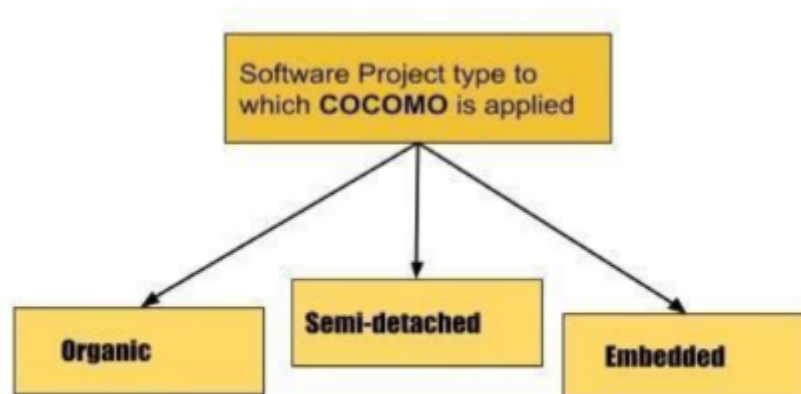
- Requirements of projects are not rigid, such a mode example is payroll processing system.

**Semi-Detached Mode:-** A software project is said to be a Semi-Detached type if-

- Project has complexity.
- Project team requires more experience, better guidance and creativity.
- The project has an intermediate size and has mixed rigid requirements such a mode example is a transaction processing system which has fixed requirements.
- It also includes the elements of organic mode and embedded mode.
- Few such projects are- Database Management System (DBMS), new unknown operating system, difficult inventory management system.

**Embedded Mode:-** A software project is said to be an Embedded mode type if-

- A software project has fixed requirements of resources.
- Product is developed within very tight constraints.
- A software project requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such mode software requires a larger team size than the other two models.



## **2.8 SPIRAL MODEL**

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the incremental model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

1. Identification
2. Design
3. Construct/ build
4. Evaluation/ Risk Analysis

### **Identification:**

This phase starts with gathering the business requirements in the baseline(first/innermost) spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

### **Design:**

The Design phase starts with the conceptual design in the first spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

### **Construct or Build:**

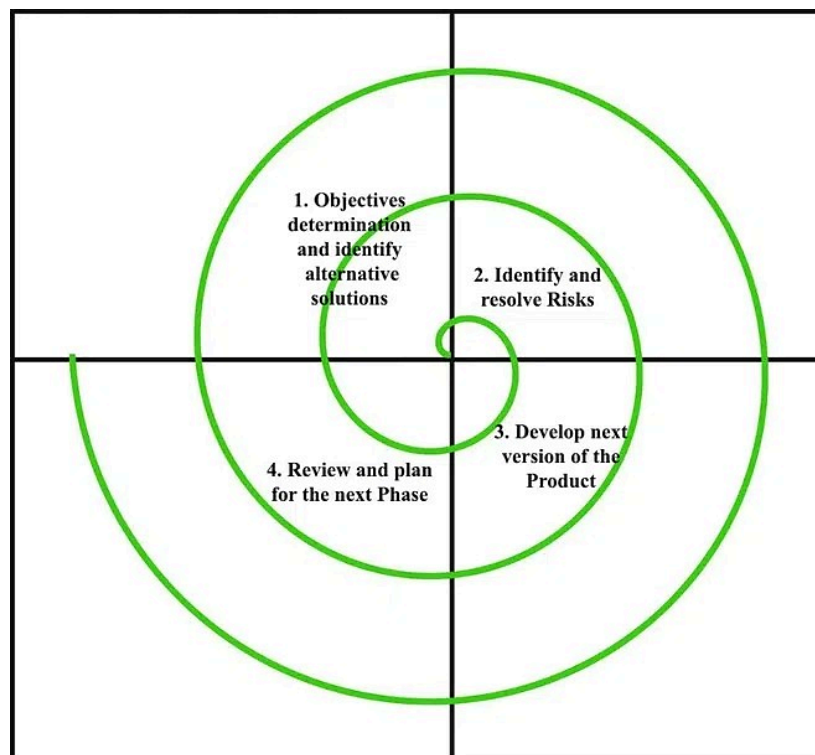
The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

### **Evaluation and Risk Analysis:**

Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

Based on the customer evaluation, the software development process enters the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.



**Applications:**

The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product, i.e. learning with maturity which involves minimum risk for the customer as well as the development firms.

The following pointers explain the typical uses of a Spiral Model –

- When there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

**Advantage:**

- The advantages of the Spiral SDLC Model are as follows –
- Changing requirements can be accommodated.
- Allows extensive use of prototypes.
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

**Disadvantage:**

- The disadvantages of the Spiral SDLC Model are as follows –
- Management is more complex.
- End of the project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex.
- Spiral may go on indefinitely.

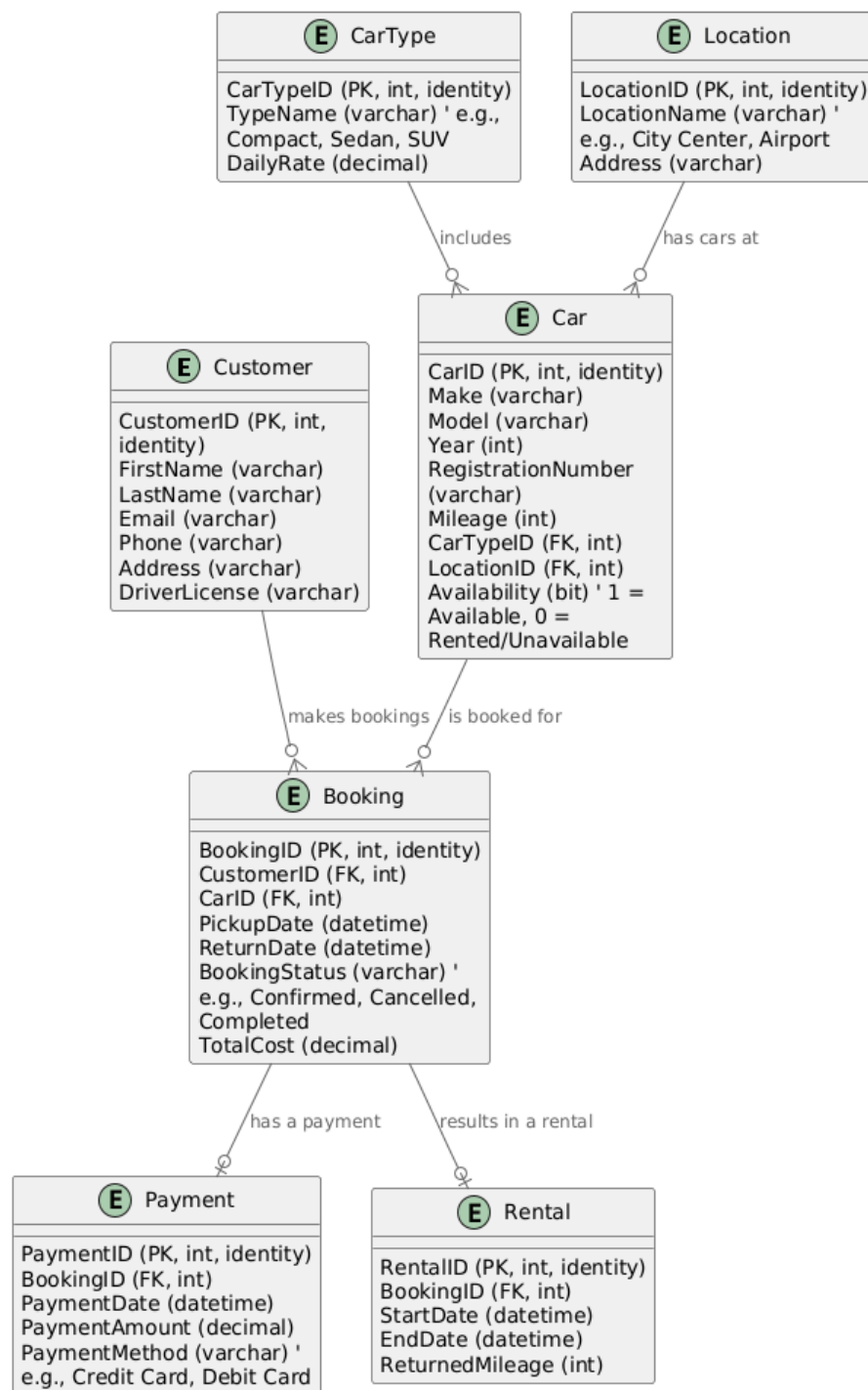
# **DESIGNS**

### **3.1 ENTITY RELATIONSHIP DIAGRAM**

A graphical model of the data needed by the system, including thinking about which information is stored and the relationship among them, produced in structured analysis and information engineering.

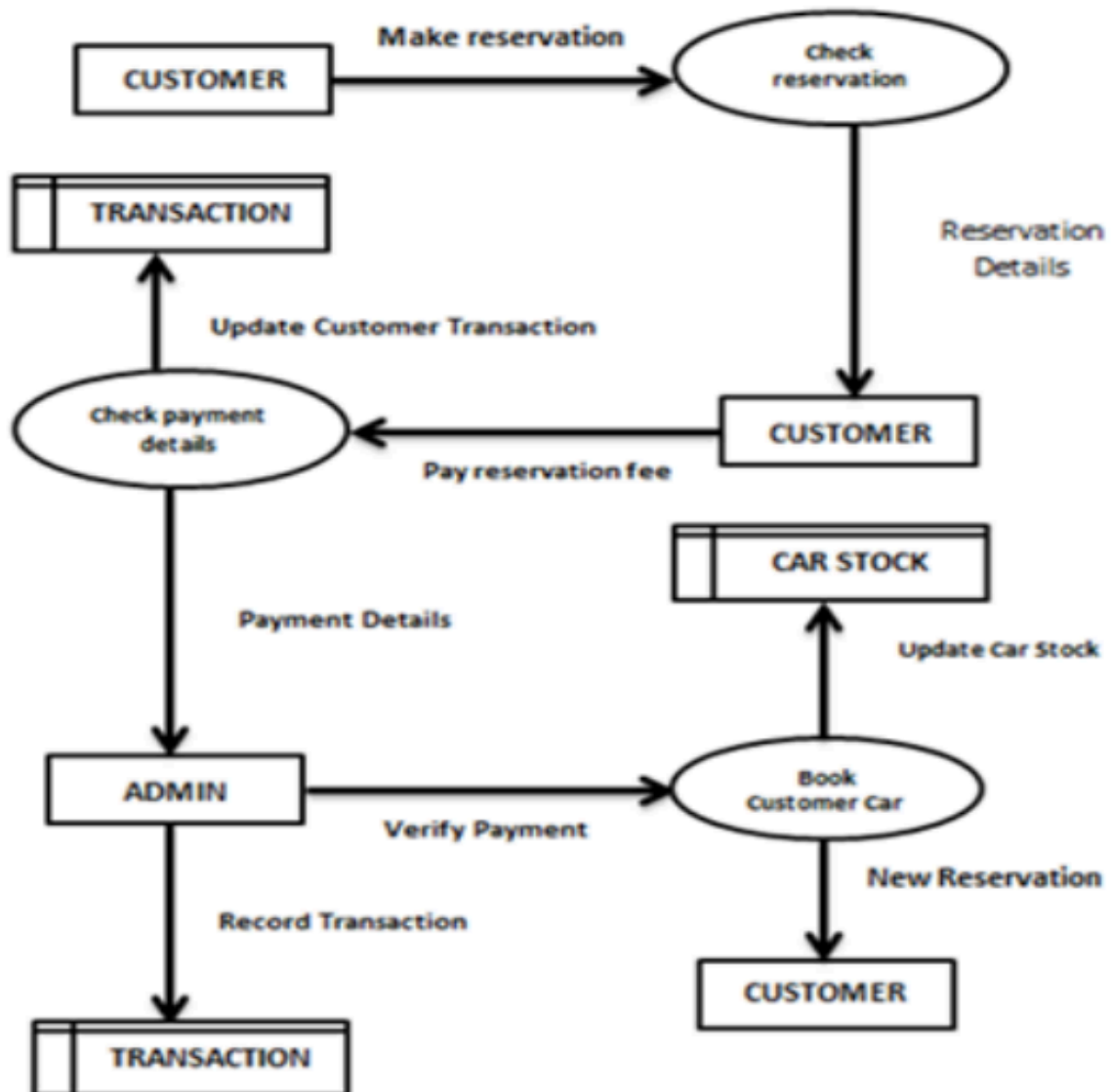
The relational approaches to system development places a great deal of emphasis on data storage requirements including the data entities, their attributes and the relationship among the data entities. The model used to define the data storage requirements is called the Entity Relationship Diagram.

On the Entity Relationship Diagram, a rectangle represents data entities, and lines connecting the rectangle show the relationship among data entities.

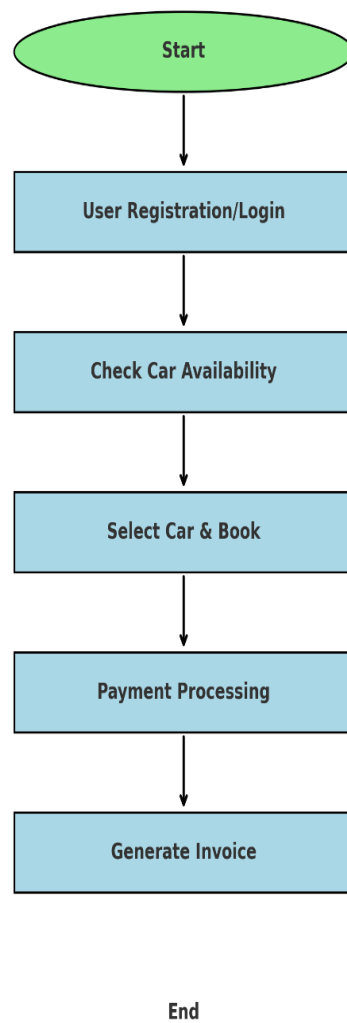




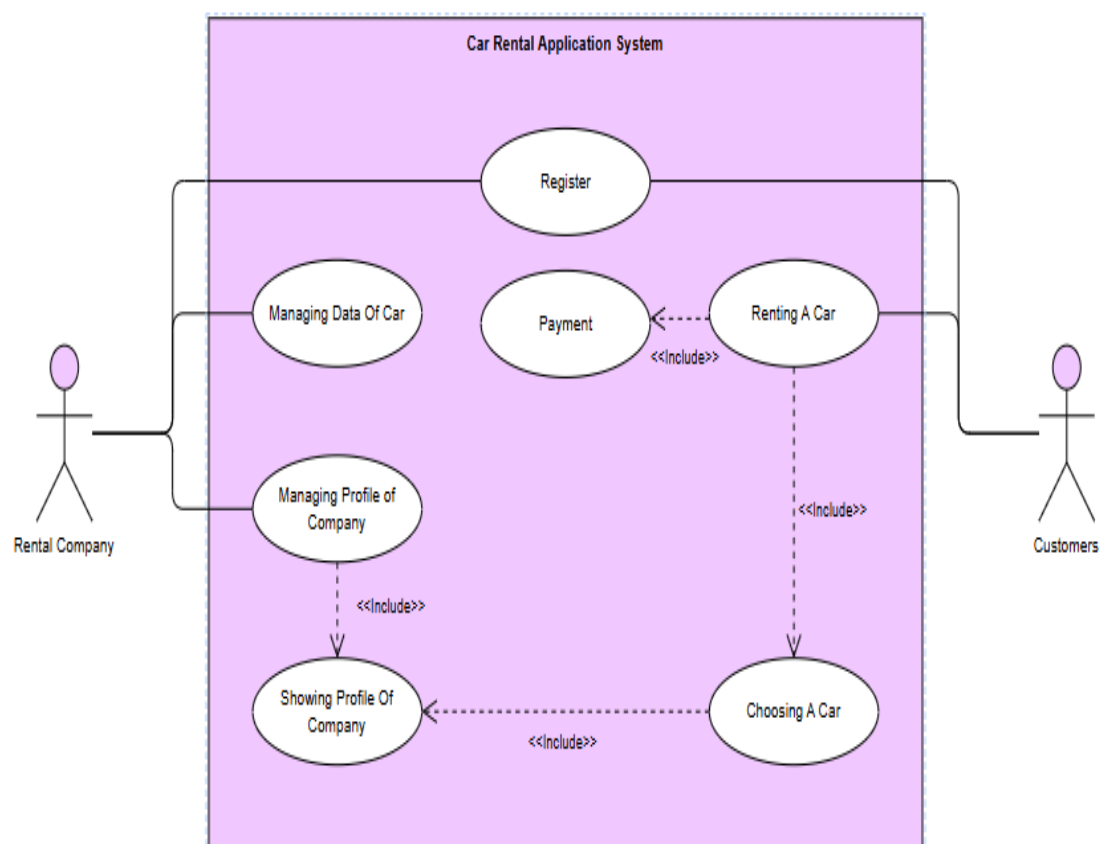
### 3.2 DATA FLOW DIAGRAM



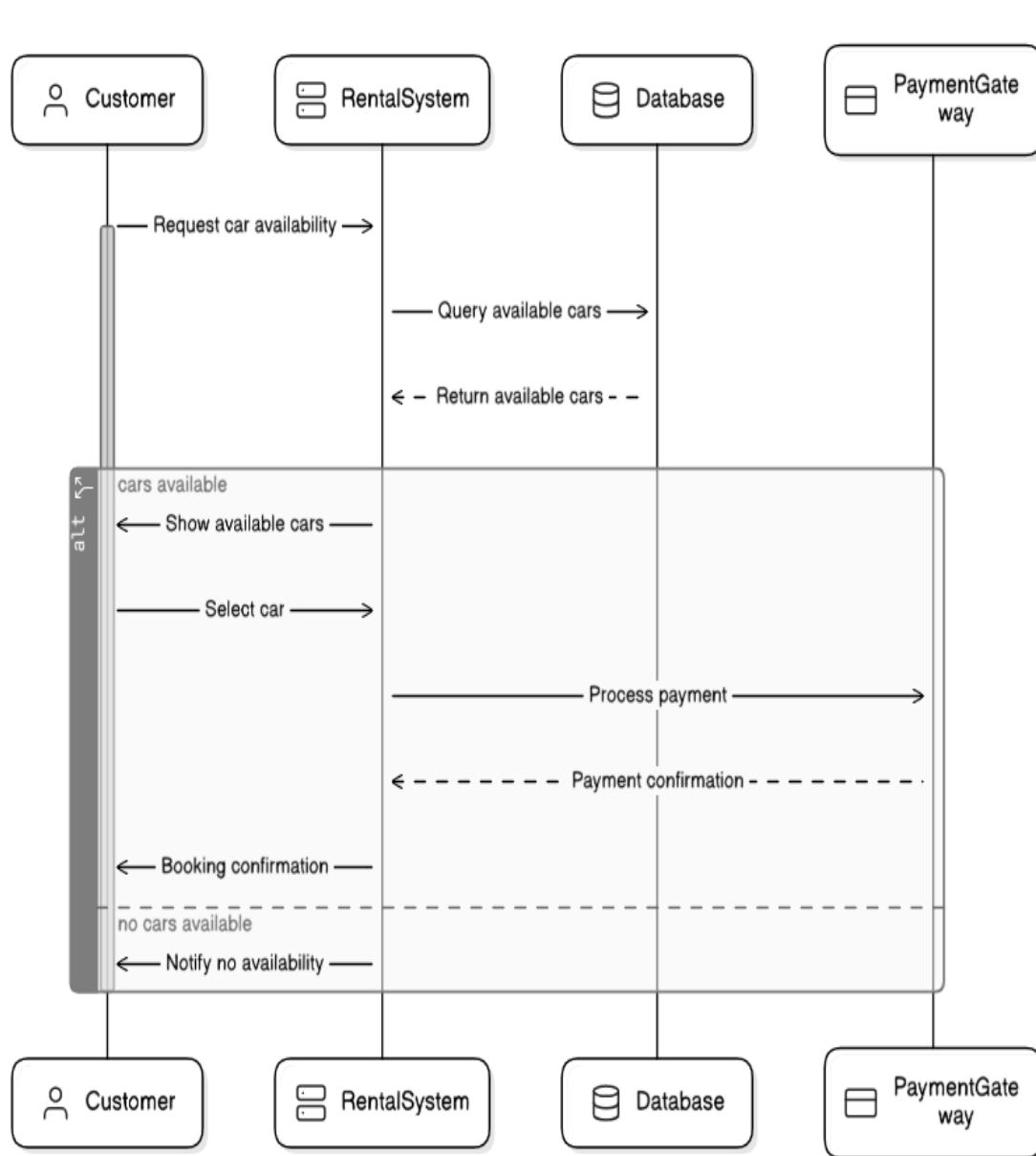
### 3.3 SYSTEM FLOWCHART



### 3.4 USECASE DIAGRAM



### 3.5 SEQUENCE DIAGRAM



### 3.6 Event Table

Event Name	Trigger	Activity	Response/Output
<b>User Registration</b>	User submits registration form	Validate and store user details in the database	Success message or validation errors
<b>User Login</b>	User enters credentials	Authenticate user via business logic	Redirect to dashboard or error message
<b>Search Available Cars</b>	User specifies dates and location	Query database for available cars	Display available car options
<b>Book Car Reservation</b>	User selects a car and enters booking details	Validate booking and update reservation records	Booking confirmation or errors
<b>Cancel Reservation</b>	User initiates cancellation	Update reservation status in database	Cancellation confirmation
<b>View Rental History</b>	User requests rental history	Fetch booking history from the database	Display list of past and current rentals
<b>Add Car to Inventory</b>	Admin adds new car details	Insert car details into the database	Car addition confirmation
<b>Update Car Details</b>	Admin updates car information	Modify car details in the database	Car update confirmation
<b>Delete Car from Inventory</b>	Admin removes car from inventory	Delete car record from the database	Car removal confirmation
<b>Generate Reports</b>	Admin requests usage report	Fetch and aggregate data from the database	Display or export report
<b>Logout</b>	User clicks logout	Clear session and redirect to login page	Redirect to login page

### **3.7 DATA DICTIONARY**

The Data Dictionary is a central repository for all the information required by a specific system. The objective of creating a data dictionary is to centralize all information so that it becomes easy for both the analyst and to understand the system and the information contained in it.

A conventional Data Dictionary describes all the entities in the system along with all its attributes, the various processes, methods and modules in which these attributes are used and their aliases.

#### **1. Table: Users**

<b>Field Name</b>	<b>Data Type</b>	<b>Description</b>
<b>UserID</b>	INT (PK)	Unique identifier for the user
<b>Username</b>	VARCHAR(50)	User's login name
<b>Password</b>	VARCHAR(50)	Hashed password
<b>Email</b>	VARCHAR(50)	User's email address

#### **2. Table: Cars**

<b>Field Name</b>	<b>Data Type</b>	<b>Description</b>
<b>CarID</b>	INT (PK)	Unique identifier for the car
<b>Company</b>	VARCHAR(50)	Car manufacturer
<b>Model</b>	VARCHAR(50)	Car model name
<b>Year</b>	INT	Year of manufacture
<b>License Plate</b>	VARCHAR(50)	Car license plate
<b>Status</b>	ENUM	Availability status
<b>Daily Rate</b>	DECIMAL	Rental price per day

### 3. Customer Register

Field Name	Data Type	Description
Customer_id	INT (PK)	Unique identifier for the customer
Customer_name	NCHAR(50)	identifier for the customer
Customer_Address	NCHAR(50)	identifier for the customer home address
Customer_Mobno	NVARCHAR(50)	identifier for the customer to contact
Customer_Email	NCHAR(50)	identifier for the customer to contact, and remind

### 4. Table: Reservations

Field Name	Data Type	Description
Reservation id	INT (PK)	Unique reservation identifier
UserID	INT (FK)	Reference to the user
CarID	INT (FK)	Reference to the car
StartDate	DATE	Rental start date
EndDate	DATE	Rental end date

<b>TotalCost</b>	DECIMAL(10.2)	Total cost of the reservation
<b>DailyRate</b>	ENUM	Status of the reservation

### 5. Table: Payments

Field Name	Data Type	Description
PaymentID	INT (PK)	Unique payment identifier
ReservationID	INT (FK)	Reference to the reservation
PaymentDate	DATETIME	Date and time of the payment
Amount	DECIMAL(10,2)	Payment amount
PaymentStatus	ENUM('Completed', 'Failed', 'Pending')	Payment status

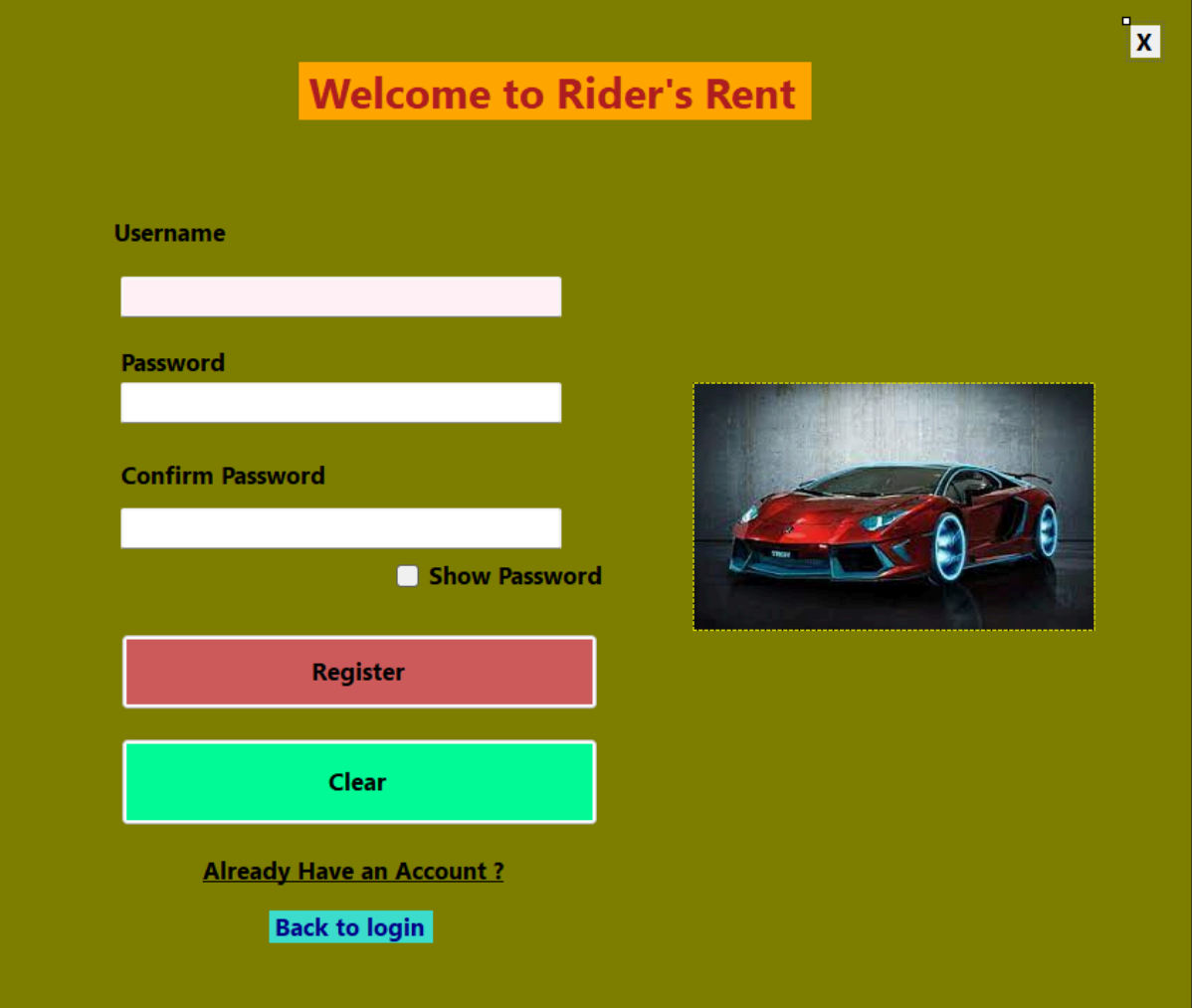
### 6. Table: Report

Field Name	Data Type	Description
<b>ReportID</b>	INT (PK)	Unique report identifier
<b>GenerateDate</b>	DATETIME	Date and time the report was generated
<b>ReportType</b>	VARCHAR(500)	Type of report (e.g., revenue, usage)
<b>ReportData</b>	TEXT	JSON or text data for the report



### **3.8 Menu Tree**

### 3.9 Screen Shot

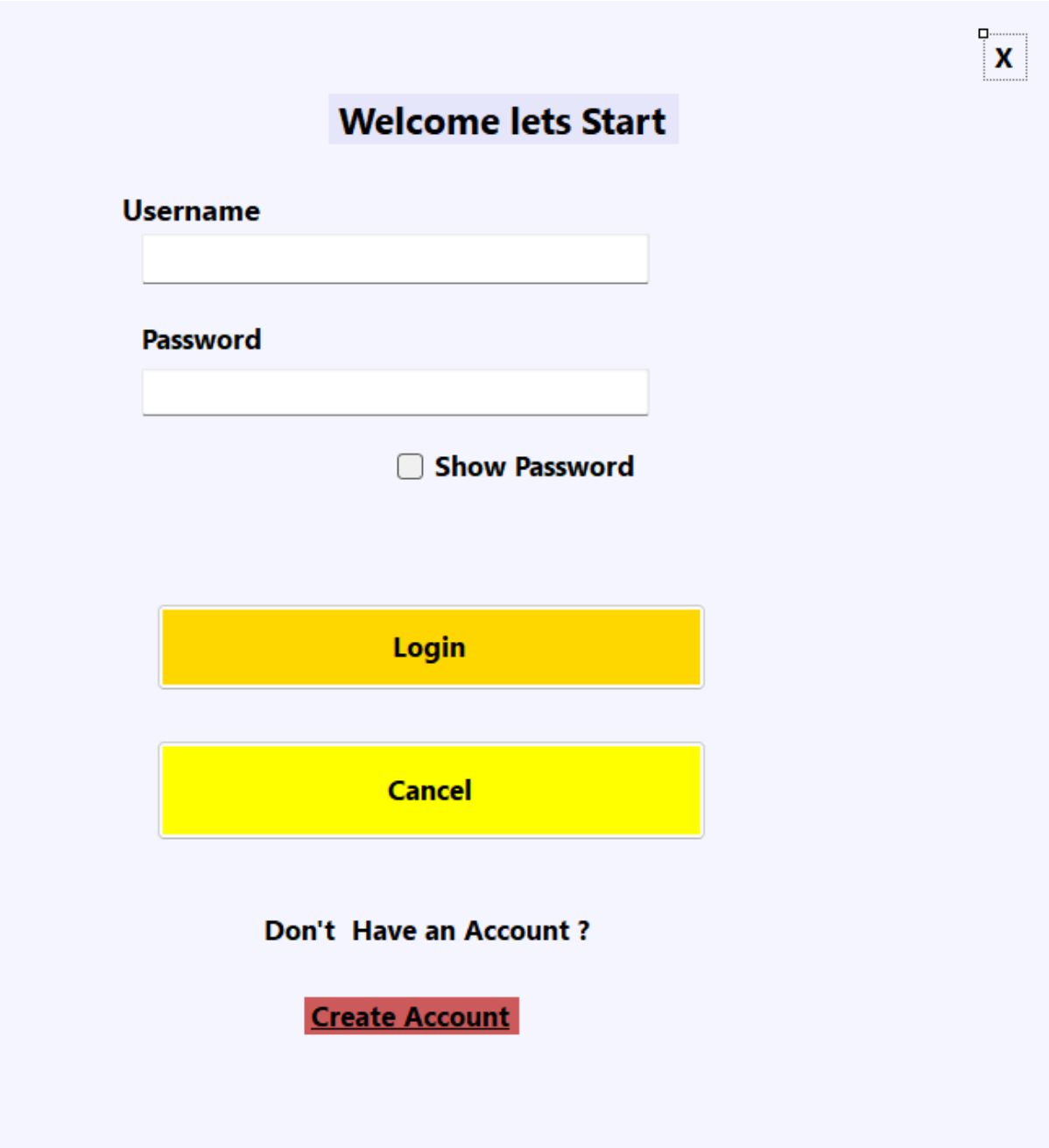


The screenshot shows a web application interface for 'Rider's Rent' with a dark olive green background. At the top center, a yellow banner displays 'Welcome to Rider's Rent' in red text. In the top right corner, there is a small white square button with a black 'X'. The registration form is located on the left side and includes the following elements:

- Username:** A label followed by a light pink rectangular input field.
- Password:** A label followed by a white rectangular input field.
- Confirm Password:** A label followed by a white rectangular input field.
- Show Password:** A small white square checkbox followed by the text 'Show Password'.
- Register:** A red rectangular button with white text.
- Clear:** A green rectangular button with white text.
- Already Have an Account ?** A text label with a blue underline.
- Back to login:** A blue rectangular button with white text.

To the right of the form, there is a rectangular image of a red sports car with glowing blue wheels, set against a dark, textured background. The image is enclosed in a thin yellow dashed border.

Figure.1



The form is titled "Welcome lets Start" in a light blue box. It includes input fields for "Username" and "Password", a "Show Password" checkbox, and "Login" and "Cancel" buttons. At the bottom, it asks "Don't Have an Account ?" and features a "Create Account" button. A close button (X) is in the top right corner.

X

Welcome lets Start

Username

Password

☐ Show Password

Login

Cancel

Don't Have an Account ?

Create Account

Figure.2

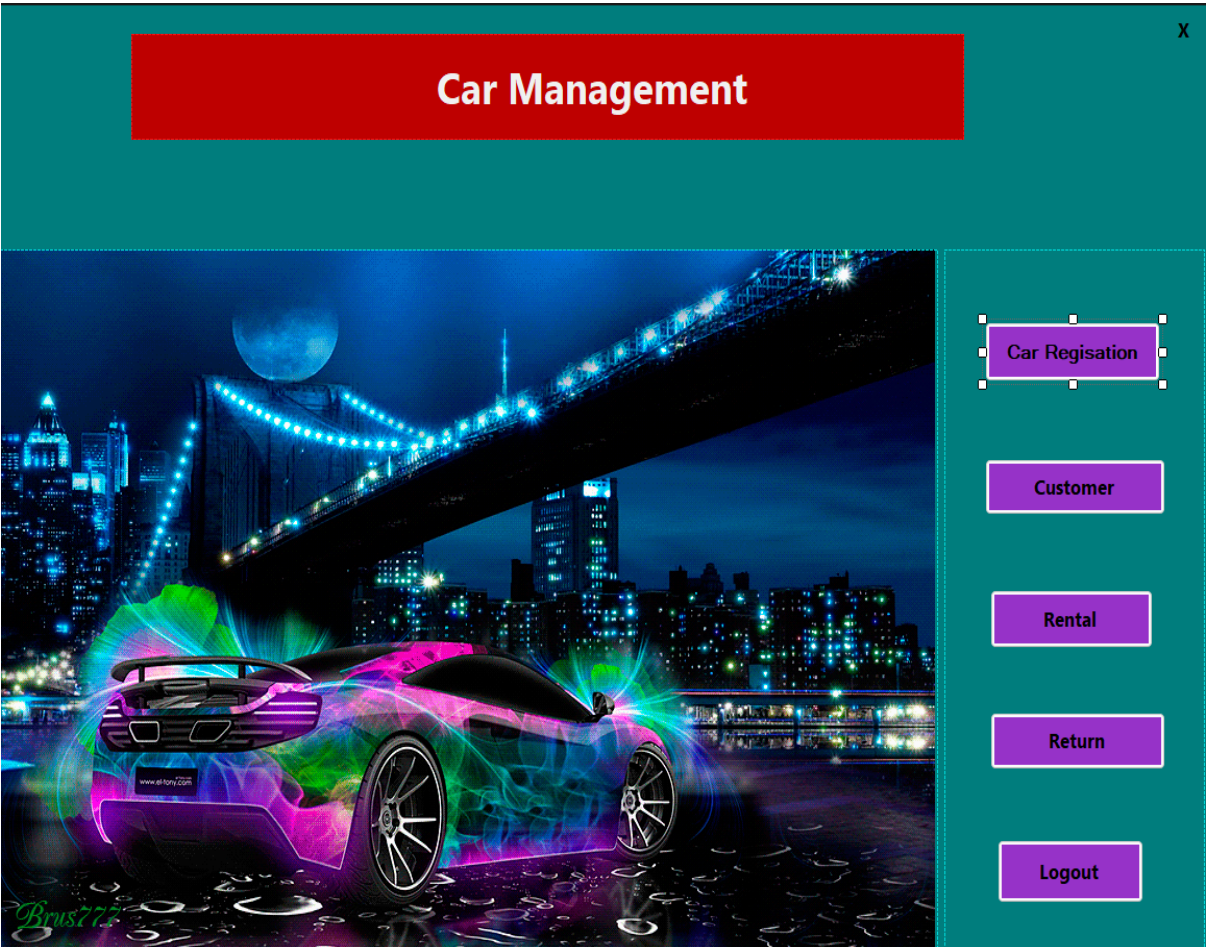


Figure.3

### Car Registration

Car Register

Car Reg no:-

Company

Model

Available

Regno	Company	Model	Aval	Edit	delete
-------	---------	-------	------	------	--------

Add

Cancel

Back to Customer Registration

Refresh

Customer Registration

X

Customer

Customer Id

Customer Name

Address

Mob No

Email Id

Add

Cancel

Customer Id	Customer Name	Address	Mob No	Email	Edit
-------------	---------------	---------	--------	-------	------

Back to Car Registration

Refresh

## 4.1 Pseudo Code

### For Login Page

```
1 START
2 Display Login Form
3 Accept username and password from user
4 If username and password are valid Then
    Redirect user to Homepage
    Display "Login successful" message
Or
    Display "Invalid username or password " message
5 END
```

### For Register Page

```
1 START
2 Display registration form
3 Accept username, password, email, and other details from user
4 If all required fields are filled and data is valid Then
    Save user details in the system
    Redirect user to login page
    Display "Registration successful" message
Or
    Display "Please fill all required fields correctly" message
5 END
```

### For Logout Page

```
1 START
2 If user is logged in Then
    Clear session data
    Redirect user to login page
    Display "Logout successful" message
Or
    Display "You are not logged in" message
3 END
```

### For Car Registration

```
Fill the following details
1.Fill the Car Regno
2. Add Company, model Name
3. After Adding the detail click on "Add" Button
4. An popup will occur, stating "Record added successful"
5. if by chance there is an mistake in typing the entry
6. click on edit button near Grid view table to edit the entry
7. after editing the entry click on refresh button
To see your entry on the grid view
```

### **For Customer Registration**

Fill the following details

- 1.Fill the Customer id
2. customer name, email, address, mobile
3. After Adding the detail click on “Add” Button
4. An popup will occur, stating ”Record added successful”
5. if by chance there is an mistake in typing the entry
6. click on edit button near Grid view table to edit the entry
7. after editing the entry click on refresh button

To see your entry on the grid view



# **Testing procedures and implementation**

## **5.1 Testing**

### **1. Introduction**

Testing is a critical phase in the development lifecycle of the Car Rental System. It ensures the software meets functional and performance requirements. This section details the types of testing performed, test cases, and results.

### **2. Testing Methodologies**

The following testing methodologies were used:

- Unit Testing -Testing individual components (services, repositories).
- Integration Testing -Verifying interaction with the MySQL database.
- Functional Testing - Ensuring the system behaves as expected.
- Performance Testing - Checking the system's response under load.

### **3. Unit Testing**

Unit tests were written using xUnit and Moq framework to test the core functionalities of the service layer.

- Test Name: GetAllCars\_ReturnsCarList
- Objective: Ensure that the system retrieves all available cars from the repository.
- Test Scenario:
- Given: The repository contains multiple car records.
- When: The GetAllCars() method is called.
- Then: The method should return a list of all available cars.

## 4. Integration Testing

Integration tests verify the interaction between the application and the MySQL database.

- Test Name: Verify Database Connection
- Objective: Ensure that the application can connect to MySQL and retrieve data.
- Test Scenario:
  - Given: A MySQL database is set up with test data.
  - When: The application queries for available cars.
  - Then: The application should return accurate results.
- Test Steps:
  1. Start the MySQL database.
  2. Seed test data into the database.
  3. Call the API endpoint /api/cars.
  4. Verify the response matches expected results.

## 5. Test Case Summary

Test Case ID	Description	Input	Expected Output	Status
TC001	Get All Cars	API Call /api/cars	List of cars	Passed
TC002	Book a Car	Car ID	Confirmation Message	Passed
TC003	Check Database Connection	None	Successful Connection	Passed

## 5.2 Test Cases

### 1. Introduction

The Car Rental System Black Book defines the expected behaviour and test cases for the system. It serves as a comprehensive guide for testing, ensuring all functionalities meet the requirements. This document is intended for testers, developers, and stakeholders involved in the project.

### 2. Scope

This Black Book covers the functional testing of the Car Rental System, including:

- Customer Management (registration, login, profile management)
- Car Management (adding, updating, searching, availability)
- Rental Management (booking, returning, cancellation, payment)
- Reporting (generating reports on rentals, revenue, etc.)
- Administrative Functions (managing users, car types, locations)

### 3. Test Cases

The following table provides a structured format for defining test cases.

Test Case ID	Feature	Description	Input Data	Output	Pre-Conditions	Post-Conditions
001	Customer Registration	Verify successful customer registration	Valid customer data (name, address, email, etc.)	Customer account created, confirmation message displayed	None	Customer record exists in the database
002	Customer Registration	Verify registration with invalid email format	Invalid email address	Error message displayed indicating invalid email format	None	No new customer record created
003	Customer Login	Verify successful customer login	Valid username and password	User logged in, access granted to	Customer account exists	User session active

		er login		customer featur		
004	Customer Login	Verify login with incorrect password	Valid username, incorrect password	Error message displayed indicating incorrect	Customer account exists	User session Remains inactive
CAR-001	Add Car	Verify adding a new ca	Valid car details (make, model, year, registration, etc.)	Car record created, confirmation message displayed	Admin user logged in	Car record exists in the database
CAR-002	Search Car	Verify searching cars by make	Car make	List of cars matching the make	Car records exist	Search results displayed
RENT-001	Book Car	Verify booking a car	Customer details, car ID, rental dates	Booking confirmed, booking details displayed	Customer logged in, car available	Booking record created in the database, car availability updated
RENT-002	Book Car	Verify booking an unavailable car	Customer details, car ID, rental dates (car already booked)	Error message displayed indicating car unavailability	Customer logged in	No new booking created
RENT-003	Return Car	Verify car return	Booking ID, return date, mileage	Rental record updated, car availability updated, payment calculated	Car booked	Rental record updated, car available
REPORT-001	Generate Rental Report	Verify generating a rental report	Date range	Report containing rental details for the specified period	Admin user logged in	Report generated and displayed/downloaded

ADMIN-001	Manage Users	Verify adding a new admin user	Valid user details	Admin user created	Admin user logged in	User record created in the database
-----------	--------------	--------------------------------	--------------------	--------------------	----------------------	-------------------------------------

#### 4. Test Data

- **Valid Data:** Using realistic, but valid data for testing positive scenarios. Consider edge cases (e.g., maximum length strings).
- **Invalid Data:** Using data that violates the system's rules (e.g., incorrect formats, missing fields, exceeding limits). This is crucial for negative testing.
- **Boundary Values:** Testing at the limits of input fields (e.g., minimum and maximum lengths, valid date ranges).
- **Equivalence Partitioning:** Divide input data into ranges that are expected to be treated similarly by the system and test representative values from each partition.

#### 5. SQL Database Considerations

- **Data Integrity:** Test database constraints (e.g., foreign keys, unique indexes) to ensure data integrity.
- **Data Validation:** Verify data stored in the database is correct and consistent with the application's logic. Directly query the database for this.
- **Performance Testing:** If performance is critical, include tests that measure the database's response times for various operations.

#### 6. C#/.NET Specific Considerations

- **Unit Testing:** While this Black Book is for functional testing, unit tests are essential for testing individual components of the C# code.
- **Integration Testing:** Verify the interactions between different parts of the .NET application.
- **Exception Handling:** Test how the application handles exceptions and errors.

### **5.3 Black Box Testing**

Black Box Testing is a software testing technique where the functionality of the application is tested without any knowledge of its internal implementation, code, or architecture. Testers treat the software as a "black box," focusing solely on the inputs and outputs. They don't need to know *how* the system works, just what it should do.

#### **Why:-**

- **Focus on User Perspective:** Black box tests simulate how real users will interact with the system. This helps identify usability issues and ensures the system meets user requirements.
- **Independent Testing:** Black box tests can be performed by testers who don't have programming skills, providing an independent perspective.
- **Early Bug Detection:** Black box testing can start early in the development lifecycle, even before the code is complete, by testing against requirements and specifications.
- **Comprehensive Coverage:** By focusing on inputs and outputs, black box testing can help ensure that all functionalities are tested.

#### **Black Box Testing Techniques for a Car Rental System:-**

- **Equivalence Partitioning:** Divide the input data into ranges (partitions) that are expected to be treated similarly by the system. Test one valid value from each partition. For example, for car types, you might have partitions for "Compact," "Sedan," "SUV," etc. You test one car from each type rather than testing every single compact car model.
- **Boundary Value Analysis:** Test the values at the edges of the input partitions. These are the most likely places for errors to occur. For example, test the minimum and maximum rental durations allowed, the minimum and maximum ages for a driver, etc.
- **Decision Table Testing:** Use a decision table to represent complex business rules and conditions. This is useful for testing scenarios with multiple inputs and outputs. For example, a decision table could cover the rules for calculating rental costs based on car type, rental duration, discounts, etc.

- **State Transition Testing:** Model the system's behaviour as a set of states and transitions between those states. Test the transitions to ensure they are handled correctly. For example, the state of a car could be "Available," "Rented," or "Maintenance." Test the transitions between these states (e.g., from "Available" to "Rented" when a car is booked).
- **Use Case Testing:** Test the system by executing specific use cases, which represent typical user interactions. Examples of use cases for a car rental system include "Rent a Car," "Return a Car," "Search for Available Cars," "Manage User Profile," etc.
- **Check the edges:** We also need to test the limits. What's the shortest rental period allowed? What's the longest? What happens if someone tries to book a car for just one minute? These edge cases are often where bugs hide. This is "Boundary Value Analysis."
- **Consider all the possibilities:** Sometimes, things get complicated. Maybe the rental price depends on the car type, the rental duration, and whether there are any discounts. We can use something called a "Decision Table" to make sure we test all the different combinations of these factors.

#### **Black Box Test Cases:-**

- **Valid Customer Registration:** Input: Valid customer data (name, address, email, phone number, driver's license). Expected Output: Customer account created successfully, confirmation message displayed.
- **Invalid Email Format:** Input: Customer registration with an invalid email address. Expected Output: Error message displayed indicating invalid email format.
- **Book Car (Available Car):** Input: Customer login, car ID, valid rental dates. Expected Output: Booking confirmed, booking details displayed, car status updated to "Rented."
- **Book Car (Unavailable Car):** Input: Customer login, car ID (already rented), valid rental dates. Expected Output: Error message displayed indicating car is not available.
- **Return Car (On Time):** Input: Booking ID, return date (within the rental period). Expected Output: Rental record updated, car status updated to "Available," payment calculated.



- **Return Car (Late):** Input: Booking ID, return date (after the rental period). Expected Output: Rental record updated, car status updated to "Available," late fee calculated and added to the payment.
- **Search Cars (By Make):** Input: Car makes (e.g., "Toyota"). Expected Output: List of available cars matching the specified make.
- **Generate Report (Rental Report):** Input: Date range. Expected Output: Report containing rental details for the specified period.

### Testing Process:

1. **Define Test Cases:** Based on the requirements and specifications, create a comprehensive set of black box test cases covering all functionalities.
2. **Prepare Test Data:** Create the necessary test data for each test case.
3. **Execute Test Cases:** Run the test cases and record the actual results.
4. **Compare Results:** Compare the actual results with the expected results.
5. **Report Bugs:** If there are any discrepancies, report them as bugs.
6. **Retest:** After the bugs are fixed, retest the affected functionalities.

## **5.4 Implementation**

The implementation of the Car Rental Management System involves the practical development and integration of the system's components to ensure smooth functionality, efficient data management, and a user-friendly experience. This section details the technologies used, the development environment, and the step-by-step approach followed during the project.

**1. Technology :-** The Car Rental Management System is developed entirely using C# for both front-end and back-end functionalities. The following technologies were selected for their reliability, scalability, and ease of integration:

- Front-end & Back-end: C# (for WinForms UI and business logic).
- Database: Microsoft SQL Server (for storing car details, user information, bookings, and payments).
- IDE : Visual Studio 2022

**2. Development Environment :-** the following are required material

- WinForms Application –
- Designed using C# in Visual Studio.
- Contains forms for car management, customer, and payment
- Implements payment gateways for secure transactions.

**3. Database Design :-** was designed to support the core functionalities of the car rental process. Following are the tables for it till now

- Users- stores their personal data
- Car Register – stores info of car regno, company, model.
- Rental- tracks rental period of customer, customer-Id and so on
- Payment – tracks the record for payment status.

**4. Implementation Process :-** it followed a modular and iterative approach, ensuring that each component was developed, tested, and integrated systematically.

**1. Requirement Gathering and Planning:**

- Identified key functionalities such as car listing, booking, and payment options.
- Created a project timeline and allocated tasks accordingly.

**2. Design**

- Developed wireframes for the WinForms application interface.
- Designed the database schema to support relational data operations.

**3: development :**

- Built the WinForms application, focusing on operations for cars and bookings.

**4. Testing :**

- Conducted testing for each module to ensure smoothness of functionality

**5. Deployment :-** The deployment of the Car Rental Management System involved creating a standalone executable and setting up the necessary production environment.

**a. Building the Executable:**

- Compiled the C# WinForms project in Visual Studio to generate a .exe file.
- Ensured that all required assemblies (DLLs) were included in the build output folder.

**b. Database Configuration:**

- Deployed the SQL Server database to a production environment.
- Created a database backup from the development server and restored it to the production server.
- Updated the application's connection string to point to the production database.

**c. Packaging the Application:**

- Created a setup file using Visual Studio Installer or a third-party packaging tool.
- Included prerequisites such as .NET Framework runtime and SQL Server client tools.

**d. Installation:**

- Tested the installer on multiple machines to ensure a smooth setup process.
- Verified the application's functionality with the production database.

**e. Distribution:**

- Provided the setup file to end-users for installation.
- Offered a user manual to guide installation and initial configuration.

## **5.5 Hardware and Software Requirement**

### **1) Hardware Requirement :-**

- I. Processor:** Intel core i3 or higher
- II. RAM :** Minimum 8GB(Recommend 16BGB)
- III. Storage:** 500 GB or higher (SSD preferred for better performance)
- IV. Network:** High-speed internet connection

### **2) Software Requirement :-**

- I. Visual Studio 2022**
- II. Microsoft SQL Server 2022**

## **6. Conclusion**

- The Car Rental Management System project successfully demonstrates the practical application of software development principles, offering an efficient and streamlined solution for managing car rentals. By leveraging C# and SQL Server, the system integrates core functionalities such as car management, booking, and secure payment processing into a single cohesive platform.
- This project not only enhances the rental experience for customers but also simplifies administrative tasks for rental agencies. The modular development approach, combined with continuous testing, ensured a reliable and user-friendly application. Moreover, deploying the system as a standalone executable makes it accessible and easy to install, catering to real-world business needs.
- Overall, the Car Rental Management System stands as a testament to efficient coding practices, thoughtful database design, and a seamless user experience, marking a successful achievement in software engineering.

# **APPENDIX**

## **7.1 System Control**

### **1. Access Control and Security:**

- **Authentication:** Verifying the identity of users. This typically involves usernames and passwords. Consider secure password storage (hashing and salting) and potentially multi-factor authentication for added security. .NET provides built-in mechanisms for authentication and authorization.
- **Authorization:** Controlling what authenticated users are allowed to do. Different user roles (e.g., customer, admin, staff) should have different levels of access. For example, only admins should be able to add new cars to the system. .NET's role-based access control can be used here.
- **Data Encryption:** Protecting sensitive data, both in transit (e.g., using HTTPS) and at rest (e.g., encrypting data in the database). This is crucial for protecting customer information and payment details. SQL Server offers encryption features, and .NET provides classes for encrypting data.
- **Input Validation:** Preventing malicious or incorrect data from entering the system. This involves checking all user inputs to ensure they are in the correct format, within acceptable ranges, and don't contain harmful code (e.g., SQL injection). .NET provides validation controls and techniques for this.
- **Database Constraints:** Using database features like primary keys, foreign keys, unique indexes, and check constraints to enforce data integrity rules. For example, a foreign key constraint can ensure that a rental record refers to a valid car and customer. This is handled in your SQL database.
- **Transactions:** Ensuring that database operations are performed atomically. Either all changes within a transaction are committed, or none are. This prevents data



corruption in case of errors. .NET provides support for transactions in database interactions.

- **Data Validation (Server-Side):** Even with client-side validation, it's essential to perform server-side validation to ensure data integrity. This is because client-side validation can be bypassed. .NET code should handle this.
- **Concurrency Control:** Managing simultaneous access to the database by multiple users. This prevents data conflicts and ensures data consistency. SQL Server provides concurrency control mechanisms.

## 2. Error Handling and Logging:

- **Exception Handling:** Gracefully handling errors and exceptions that occur during program execution. This prevents the application from crashing and provides informative error messages to users. .NET's try-catch blocks are used for exception handling.
- **Logging:** Recording important events and errors that occur in the system. Logs are essential for debugging, monitoring, and auditing. Consider using a logging framework like log4net.
- **Error Reporting:** Providing a way to report errors to developers. This could involve automatic email notifications or integration with a bug tracking system.

## 3. System Monitoring and Maintenance:

- **Performance Monitoring:** Tracking system performance metrics (e.g., response times, resource usage) to identify bottlenecks and optimize performance. Tools like Performance Monitor can be used.
- **Security Auditing:** Regularly reviewing system logs to detect any suspicious activity.

- **Database Backup and Restore:** Implementing a robust backup and restore strategy to protect against data loss. SQL Server provides backup and restore capabilities.
- **System Updates and Patching:** Keeping the system software (operating system, .NET framework, database server) up to date with the latest security patches and bug fixes.

#### 4. Configuration Management:

- **Configuration Files:** Using configuration files (e.g., web.config or app.config) to store system settings. This makes it easier to change settings without recompiling the code. .NET uses configuration files.
- **Environment-Specific Configurations:** Managing different configurations for different environments (e.g., development, testing, production).

## **6.2 Operational Manual**

By Operational Manual well-maintained, that can ensure your Car Rental System is used effectively and efficiently, minimizing downtime and maximizing its value. It empowers users to solve problems themselves and reduces the burden on your support staff.

### **Introduction:**

### **System Access and Login:**

### **Daily Operations:-**

This is the core of the manual, detailing how to perform common tasks within the system. Organize it by functional area:

#### **1. Customer Management:**

- Adding new customers: Step-by-step instructions, including required fields and any validation rules.
- Updating customer information: How to modify existing customer details.
- Searching for customers: Explaining search criteria and filtering options.

#### **2. Car Management:**

- Adding new cars: Including details like make, model, year, registration, and images.
- Updating car information: How to modify car details, including maintenance records.
- Searching for cars: Explaining search criteria (e.g., by make, model, availability).
- Managing car availability: How to mark cars as available, rented, or under maintenance.

#### **3. Rental Management:**

- Creating a rental agreement: Step-by-step guide, including selecting a car, customer, rental dates, and calculating the rental cost.
- Processing payments: Explaining different payment methods and how to record payments in the system.
- Returning a car: Instructions for processing car returns, including mileage updates and final payment calculations.
- Managing cancellations: How to cancel bookings and handle refunds.

#### **4. Reporting:**

- Generating reports: Explaining the different types of reports available (e.g., daily rentals, revenue reports) and how to generate them.
- Exporting reports: How to export reports in various formats (e.g., PDF,)

#### **5. Administrative Functions:**

- User management: Adding, modifying, and deleting user accounts.
- Managing car types and categories.
- Setting up rental locations.
- Configuring system settings.

#### **6. Contact Information**

- **Support Contact:** Provide contact information for IT support or system administrators.
- **Vendor Contact:** If applicable, include contact information for the software vendor.

#### **7. Security**

- **Security Best Practices:** Outline security best practices for using the system, including password management, data protection, and access control.
- **Data Security:** Explain how sensitive data is handled and protected within the system.

## **7.3 User Manual**

### **1. Introduction:**

Welcome to the Car Rental System! This manual provides a comprehensive guide to using the system for both customers and administrators. This system is designed to streamline the car rental process, making it easy to book, manage, and return vehicles.

### **2. Getting Started:-**

**Accessing the System:** The system can be accessed through a web browser at [System URL]. You will need a valid username and password to log in.

#### **Logging In:-**

- Enter your username in the "Username" field.
- Enter your password in the "Password" field.
- Click the "Login" button.
- If you have forgotten your password, click the "Forgot Password" link and follow the instructions.

### **3. Customer Features:-**

#### **Registering a New Account:**

- Click the "Register" button on the login page.
- Fill out the registration form with your personal details (name, address, email, phone number, driver's license number, etc.).
- Click the "Register" button. A confirmation email will be sent to your registered email address.

#### **Managing Your Profile:**

- After logging in, click on your name or profile icon.
- You can view and update your personal information, change your password, and view your rental history.

#### **Booking a Car:**

- Select the desired rental dates and location.
- Browse available cars based on various criteria (car type, make, model, price range). \* Select the car you wish to rent. \* Review the booking details and confirm your reservation.

**Viewing/Modifying Bookings:**

- Access your booking history through your profile.
- You can view details of your past and upcoming bookings.
- Depending on the system's policy, you may be able to modify or cancel upcoming bookings.

**4. Administrator Features:-****Managing Cars:**

- Add new cars to the system, including details like make, model, year, registration number, car type, and images.
- Update car information (e.g., availability, mileage).
- Remove cars from the system.

**Managing Customers:**

- View a list of all registered customers.
- View customer details.
- Potentially, manage customer accounts (e.g., reset passwords).

**Managing Bookings:**

- View all current and past bookings.
- Manage booking status (e.g., confirm, cancel).
- Generate reports on bookings.

**Generating Reports:**

- Generate reports on various aspects of the business, such as revenue, rentals by car type, customer activity, etc. Specify the date range for the report.

**5. Troubleshooting:-**

- **Login Issues:** If you are having trouble logging in, ensure that you are using the correct username and password. If you have forgotten your password, use the "Forgot Password" link.
- **Booking Issues:** If you are unable to book a car, check the car's availability and ensure that you have entered all the required information correctly.
- **Other Issues:** If you encounter any other issues, please contact the system administrator at [Admin Email Address] or [Admin Phone Number].

**6. Contact Information:-**

For any questions or support, please contact us at:

- Email: [Support Email Address]
- Phone: [Support Phone Number]

## **7.4 Future Enhancement**

### **1. Mobile Application Development**

- **Description:** A mobile application will be developed to complement the web application. This app will allow users to book and manage rentals on-the-go. Using Xamarin or MAUI, a cross-platform mobile development framework, we can create a seamless experience for both Android and iOS users.
- **Benefits:** This enhancement will improve user convenience and accessibility, allowing users to book cars anytime, anywhere.

### **2. Advanced Search and Filtering**

- **Description:** Implementing advanced search and filtering options will enable users to find cars based on various criteria, such as location, price range, car type, and features. This will enhance the current C# and .NET codebase to provide a more refined search experience.
- **Benefits:** Users will enjoy a better experience and quicker car selection, leading to higher satisfaction and potentially increased bookings.

### **3. Integration with Payment Gateways**

- **Description:** Integrate multiple payment gateways. This will involve using third-party APIs and libraries in .NET to provide various payment options.
- **Benefits:** Offering multiple payment options will increase flexibility for users and build trust in the system, leading to higher user retention and satisfaction.

### **4. Enhanced Security Features**

- **Description:** Implement advanced security measures like two-factor authentication (2FA), encryption, and secure payment processing using ASP.NET Core Identity. This will protect user data and transactions.
- **Benefits:** Improved security will enhance user confidence and trust in the system.



## 5. Customer Reviews and Ratings

- **Description:** Add features for customers to leave reviews and ratings for cars and rental services. This will involve creating new tables in the SQL Database and updating the .NET backend to handle reviews and ratings.
- **Benefits:** Enabling reviews and ratings will provide valuable feedback for continuous improvement and increase user engagement.

## 6. Dynamic Pricing Algorithm

- **Description:** Implement a dynamic pricing algorithm that adjusts rental rates based on demand, location, and time. Using ML.NET, a machine learning library for .NET, we can optimize pricing strategies.
- **Benefits:** This will help optimize revenue and offer competitive pricing, attracting more customers during peak times.

## 7. Real-time GPS Tracking

- **Description:** Integrate real-time GPS tracking for rented cars using third-party GPS tracking APIs. This will allow monitoring of car locations for additional security and fleet management.
- **Benefits:** Enhanced security and better fleet management through real-time tracking of vehicles.

## 8. Loyalty Program

- **Description:** Introduce a loyalty program to reward frequent customers with discounts and special offers. This will involve updating the SQL Database to manage loyalty points and implementing corresponding logic in the .NET backend.
- **Benefits:** A loyalty program will increase customer retention and satisfaction, encouraging repeat business.

## 9. Automated Email and SMS Notifications

- **Description:** Implement automated email and SMS notifications for booking confirmations, reminders, and promotions. Using services like SendGrid for emails and Twilio for SMS, we can keep users informed.
- **Benefits:** Improved communication and customer satisfaction through timely notifications.

## 10. Multilingual Support

- **Description:** Add support for multiple languages to cater to a diverse user base. we can make the system accessible to non-English speaking users.
- **Benefits:** Expanded user reach and inclusivity by offering the service in multiple languages.

### **7.5 Lists of Table and Forms**

Following are the list of tables that are use in database to showcase the project implementation

#### **1. Login Table**

Column Name	Data Type
Username (PK)	Nchar(50)
Password	Nchar(50)
Confirm Password	Nchar(50)

#### **2. Car Registration**

Column Name	Data Type
Reg No [PK]	int
Company	Varchar(50)
Model	Varchar(50)
Available	Varchar(50)

#### **3. Customer Registration**

Column Name	Data Type
Customer id [PK]	Int
Customer Name	Nchar(50)
Customer Address	Nchar(50)
Customer Mob	Nvarchar(50)
Customer_email	Nchar(50)

## 4. Rental Table

Coloumn Name	Data Type
Id [PK]	Int
Car_id	Int
Customer_id	Int
Customer name	Varchar(50)
Fee	int
Date	Varchar(50)
Due date	Varchar(50)

## 5. Return Table

Coloumn Name	Data Type
Id [PK]	Int
Car_id	Int
Customer_id	Int
Date	Varchar(50)
Elp	Varchar(50)
Fine	Varchar(50)

## 7.6 Variable naming conversation

Following are the conversation for the variable to the respective pages

### 1. Register Page

SrNo	objects	prefix
1.	Form 1	Frm 1
2.	Label	Label 1
3.	TextBox	Txt 1
4.	TextBox	Txt 2
5.	TextBox	Txt 3
6.	CheckBox	CheckBox 1
7.	Label	Label 2
8.	Label	Label 3
9.	Button	Btn 1
10.	Button	Btn 2

### 2. Login page

SrNo	objects	Name of variable
1.	Form 2	Form 2
2.	Label	Label 1
3.	TextBox	Txt 1
4.	TextBox	Txt 2
5.	CheckBox	CheckBox 1
6.	Label	Label 2
7.	Label	Label 3
8.	Button	Btn 1
9.	Button	Btn 2

### 3. Main Form

SrNo	objects	Name of variable
1.	Form 3	Form 3
2.	Label	Label 1
3.	Button	Btn 1
4.	Button	Btn 2
5.	Button	Btn 3
6.	Button	Btn 4
7.	Button	Btn 5
8.	Label	Label 2
9.	Panel	Panel 1

### 4. Car Registration

SrNo	objects	Name of variable
1.	Form 4	Form 3
2.	Label	Label 1
3.	Button	Btn 1
4.	Button	Btn 2
5.	Button	Btn 3
6.	Button	Btn 4
7.	Label	Label 2
8.	Panel	Panel 1
9	Label	Label 3
10	Label	Label 4
11	Label	Label 5
12.	DataGrid	DataGrid 1
13.	GroupBox	grpBox1

**5. Customer Registration**

SrNo	objects	Name of variable
1.	Form 5	Form 3
2.	Label	Label 1
3.	Button	Btn 1
4.	Button	Btn 2
5.	Button	Btn 3
6.	Button	Btn 4
7.	Label	Label 2
8.	Panel	Panel 1
9	Label	Label 3
10	Label	Label 4
11	Label	Label 5
12.	DataGrid	DataGrid 1
13.	Label	Label 6
14.	GroupBox	GrpBox 1
15.	DataGrid	DataGrid 1

**6. Rental page**

SrNo	objects	Name of variable
1.	Form 6	Form 3
2.	Label	Label 1
3.	Button	Btn 1
4.	Label	Label 2
5.	Panel	Panel 1
6	Label	Label 3
7	Label	Label 4
8	Label	Label 5
9.	DataGrid	DataGrid 1
10.	Label	Label 6
11.	GroupBox	GrpBox 1
12.	DatePicker	Datepic 1
13.	DateTimePicker	Datepic2



**7. Return Page**

SrNo	objects	Name of variable
1.	Form 7	Form 3
2.	Label	Label 1
3.	Button	Btn 1
7.	Label	Label 2
8.	Panel	Panel 1
9	Label	Label 3
10	Label	Label 4
11	Label	Label 5
12.	DataGrid	DataGrid 1
13.	Label	Label 6
14.	GroupBox	GrpBox 1
15.	TextBox	Txt1
16.	TextBox	Txt2
17.	TextBox	Txt1