**ASSIGNMENT # 1**

**GROUP MEMBERS :**

* SHARIF ALI (P20-0130)
* ABUBAKER ATTIQUE (P20-0560)

**SECTION BCS 6-A**

**CONTENTS**

* **Requirement Document Template**

1. **Introduction**
2. **Functional Requirements**
3. **Non-Functional Requirements**
4. **System Architecture**
5. **Data Requirements**
6. **Constraints**
7. **Glossary**

* **UML Diagrams**

1. **Use Case Diagram**
2. **Class Diagram**
3. **Sequence Diagram**
4. **Activity Diagram**

* **PRODUCT SCOPE**

1. **User Management**
2. **Car Inventory Management**
3. **Reservation Management**
4. **Payment Management**
5. **Check-in and Check-out Management**
6. **Reporting and Analytics**

* **Design and Implementation Constraints**

1. **Technical Constraints**
2. **Operational Constraints**
3. **Regulatory Constraints**
4. **Time Constraints**
5. **Budget Constraints**

* **USE CASE SCENARIOS**

**Requirement Document Template**

1. **Introduction**
   1. **Purpose:** This document aims to define the requirements for developing a car rental system that allows users to reserve and return a vehicle from different locations , keeping the record of system in database so that it could be accessed any time. It should perform a variety of functions when chosen from the given category.
   2. **Scope:** The car rental system will be a web-based application that can be accessed from any device with an internet connection.
   3. **Objectives:** This system aims to simplify the process of renting a car for short-term periods, keep track of client information, and provide a convenient and user-friendly platform for both customers and rental company employees with this it also aims to provide different functions at a time in which user can make a choice.
2. **Functional Requirements**

2.1 **User Management**

2.1.1 The system shall allow users to create an account with their name, address, and contact number.

2.1.2 The system shall allow users to log in using their email address and password.

2.1.3 The system shall allow users to reset their password.

2.1.4 System will allow a user to select any one functionality from the given options.

2.2 **Reservation Management**

2.2.1 The system shall allow users to make a reservation by supplying the pickup and drop-off locations, the kind of vehicle, and the day and time of the reservation.

2.2.2 The system shall allow users to modify or cancel their reservations

.

2.2.3 The system shall prevent users from making overlapping reservations.

2.2.4 Online support system will be available to guide users while making bookings.

2.3 **Rental Management**

2.3.1 The system shall allow users to rent a car for a certain number of days.

2.3.2 The system shall allow users to return the car they rented.

2.3.3 The system shall calculate the rental fee based on the rental period and the type of vehicle rented.

3. **Non-Functional Requirements**

**3.1** **Usability**

3.1.1 The system shall have a user-friendly interface.

3.1.2 The system shall provide clear and concise instructions to users, helpful error messages, and support various languages.

**3.2 Performance**

3.2.1 The system shall respond to user requests within 3 seconds.

3.2.2 The system shall be able to handle 1000 concurrent users.

3.2.3 The system should have scalability mechanism.

**3.3** **Security**

3.3.1 The system shall store user passwords in a hashed format.

3.3.2 The system shall use HTTPS to encrypt data transmission between the client and server.

3.3.3 The system shall have a backup and recovery plan in place to ensure that user data is not lost in case of a system failure.

* 1. **Compliance**:

3.3.1 The system should comply with relevant legal and regulatory requirements such as data protection laws, accessibility standards, and industry-specific regulations.

1. **System Architecture**

4.1 The car rental system will be a web-based application developed using the following technologies:

4.1.1 Front-end: HTML, CSS, JavaScript

4.1.2 Back-end: Node.js, Express.js

4.1.3 Database: MongoDB

4.2 The system will use a REST API to communicate between the client and server.

4.3 The system will use JWT (JSON Web Token) for authentication and authorization.

1. **Data Requirements**

5.1 The system will store the following data for each user:

5.1.1 Name

5.1.2 Email address

5.1.3 Password (hashed)

5.1.4 Address

5.1.5 Contact number

5.2 The system will store the following data for each car rental:

5.2.1 User ID

5.2.2 Car ID

5.2.3 Pickup location

5.2.4 Pickup date and time

5.2.5 Drop-off location

5.2.6 Drop-off date and time

5.2.7 Rental fee

6. **Constraints**

6.1 The system must comply with data protection laws and regulations, such as GDPR.

6.2 The system must be compatible with all major web browsers.

6.3 The system must be developed within a budget of $100,000.

* 1. The system must be developed and deployed within a timeframe of 6 months.

1. **Glossary** 
   1. Car rental system: The web-based application that manages the renting of automobiles for short periods of time.
   2. User: A person who uses the car rental system to rent a car.
   3. Reservation: The act of reserving a car for a specific date and time.
   4. Rental: The act of renting a car for a specific period.

7.5 Car: An automobile that can be rented through the car rental system.

**UML Diagrams:**

**Use Case Diagram:**

1. A use case diagram is a graphical representation of the interactions between actors and the system. In the car rental system, there are two main actors: the user and the system.
2. The user can create an account, log in, make a reservation, rent a car, and return a car. The system can manage user accounts, reservations, rentals, and returns.

**Class Diagram:**

1.

A class diagram is a graphical representation of the classes, attributes, and methods in the system.

In the car rental system, there are several classes such as User, Car, Reservation, and Rental. Each class has its own attributes and methods.

1. **Sequence Diagram:**

A sequence diagram is a graphical representation of the interactions between objects in a particular scenario.

In the car rental system, a user can make a reservation, rent a car, and return a car.

The sequence diagram below shows the interactions between the user, the reservation object, the rental object, and the car object.

1. **Activity Diagram:**

An activity diagram is a graphical representation of the activities and flows in a particular scenario.

In the car rental system, a user can make a reservation, rent a car, and return a car. The activity diagram below shows the activities and flows involved in the process of renting a car.

**PRODUCT SCOPE:**

The product scope for a web-based car rental system defines the boundaries of the system and outlines what the system will and will not do. The product scope for this application could include the following:

**User Management**:

The system will allow users to register and sign up providing personal and contact information, driving license details, and payment information.

**Car Inventory Management:**

The system will allow rental agencies to manage their car inventory, adding and removing cars from the system, and updating car information such as make model, year, and mileage.

**Reservation Management**:

The system will allow users to search for available cars, select pick-up and drop-off locations, and reserve cars for a specific period.

The system will also allow users to modify or cancel their reservations.

**Payment Management:**

The system will provide a secure payment gateway that accepts various payment methods such as credit cards, debit cards or by cash.

The system will also generate invoices and receipts for complete transaction .

**Check-in and Check-out Management**:

The system will allow rental agencies to check-in and check-out cars, inspecting them for damages and collecting security deposits.

The system will also calculate rental fees and generate receipts for completed rentals.

**Reporting and Analytics:**

The system will provide reporting and analytics features that allow rental agencies to track rental activity, analyze revenue, and identify trends and patterns in rental demand.

Overall, the product scope for this application focuses on managing the car rental functions from start to end, providing users with a unified and convenient way to reserve and rent cars.

**Design and Implementation Constraints**

Design and implementation constraints refer to limitations that impact the development and design of a web-based car rental system. These constraints may be imposed by technical, operational, or regulatory factors. Some examples of design and implementation constraints for this application include:

**Technical Constraints:**

Technical constraints refer to limitations related to the development environment, hardware, software, and infrastructure used to build and operate the system.

For example, the system may need to be compatible with various web browsers and operating systems, and the development team may have limited access to hardware and software resources.

**Operational Constraints:**

Operational constraints refer to limitations related to the business operations of the car rental company. For example, the system may need to be integrated with existing inventory and rental management systems, and the development team may need to coordinate with rental agency staff to ensure a smooth transition to the new system.

**Regulatory Constraints:**

Regulatory constraints refer to limitations related to legal and regulatory requirements that impact the design and implementation of the system.

For example, the system may need to comply with data privacy laws and regulations, accessibility standards, and other industry-specific regulations.

**Time Constraints:**

Time constraints refer to limitations related to the project timeline and deadlines.

For example, the development team may have a limited timeframe to complete the project, and they may need to prioritize certain features and functionalities to meet the project's deadline.

**Budget Constraints:**

Budget constraints refer to limitations related to the available budget for the project. For example, the development team may have a limited budget for software development, hardware and software purchases, and system maintenance.

Thus ,

* Overall, these design and implementation constraints can impact the design and development of the web-based car rental system, requiring the development team to carefully consider these factors when designing and implementing the system.
* The team may need to prioritize certain features and functionalities, work with rental agency staff to ensure a smooth transition to the new system, and comply with relevant legal and regulatory requirements.
* The team will also need to consider the project timeline and budget to ensure that the system is delivered on time and within budget.

**Assumptions and Dependencies**

Assumptions and dependencies are issues that are not within the direct control of the development team but may impact the design and implementation of a web-based car rental system . Some examples of assumptions and dependencies for this application includes :

**Assumption :**

The system will be accessed using a web browser and an internet connection. The development team assumes that users will have access to these resources.

**Assumption:**

Users will provide accurate and up-to-date personal information, driving license details, and payment information during the registration process.

**Assumption:**

Rental agencies will provide accurate and up-to-date car inventory and rental agreement information that can be integrated into the system.

**Dependency:**

The system will need to integrate with existing inventory and rental management systems used by rental agencies.

**Dependency:**

The system will need to comply with relevant legal and regulatory requirements such as data protection laws, accessibility standards, and industry-specific regulations.

## **Dependency:**

## The development team will rely on third-party payment gateway providers to process payments securely.

## **Assumption:**

## The system will be tested thoroughly to ensure that it functions correctly, is user-friendly, and meets all requirements and specifications.

## Overall, these assumptions and dependencies can impact the design and implementation of the web-based car rental system, requiring the development team to consider these factors when designing and implementing the system.

## The team will need to work closely with rental agencies and other stakeholders to ensure that accurate and up-to-date information is integrated into the system.

**USE CASE SCENARIOS:**

|  |  |
| --- | --- |
| Name | Generating Receipts |
| Description | This function will calculate the total payment for the car rented. |
| Actors | Cashier |
| Pre-Condition | The customer should be registered first. |
| Main flows of Events | 1. The user will select any category. 2. The system will note the time consumed. 3. The system will generate the bill. |

|  |  |
| --- | --- |
| Name | Registration of customer |
| Description | A name needs to fill form related to his personal information e.g address, name, etc. |
| Actors | customer |
| Main flows of Events | For renting a car it’s compulsory for a user to enter the purpose, enter details and select the category. |

|  |  |
| --- | --- |
| Name | User |
| Description | Responsible for taking care of a returned vehicle and updating the vehicle log. |
| Actors | Worker |
| Pre-Condition | - |
| Main flows of Events | A user will let a customer to take vehicle only after completing the registration process. |
| Post conditions | Workers must be user-friendly. |

|  |  |
| --- | --- |
| Name | Registered Member |
| Description | A registered member is one who is their permanent client for long. |
| Actors | Members |
| Pre-Condition | He should be using this service from long. |
| Main flows of Events | A registered can easily rent a car at any time. |

|  |  |
| --- | --- |
| Name | System |
| Description | A system is responsible to run all the functions according to the requirements. |
| Actors | Developer |
| Pre-Condition | Requirements should be explained fully . |