EXPERIMENT NO. 1

Aim: Create a feasibility report of the online cab booking system

Theory:

A feasibility report is a testimony that attempts to create some sort of action. It is created to persuade the decision-maker to choose between available options. It also determines whether or not investigated task can be done with the amounts of resources available.

Features of a feasibility report:

1. Introduction: Needs to persuade the decision-maker.
2. Criterion: Need to map out the criteria of what ideal outcomes are'
3. Method: It is important to present facts that are accurate and relevant.
4. Overview: Key features must be underlined.
5. Evaluation: Major part of the report, add graphs, charts etc.
6. Conclusion: Need to state the conclusion.
7. Recommendation: Need to use your experience and knowledge in order to state which option should be adopted.

Feasibility Report for Online Cab Booking System

1. **Objective**
   To draft feasibility report of Online Cab Booking System (OCBS).

2. **Description**
   The online cab booking system (OCBS) is a social initiative which is designed to help the customers to
   reach the destination in time and with convenience. The model aims at providing
   door-to-door cab service for people who want a comfortable, economical and safe ride.

3. **The problem in the existing system**
   Existing application is full of bugs and does not have a user-friendly interface.

4. **Software Requirement**
   4.1 software compatibility with the application

4.2 synced location services
4.3 web browser like Google, Firefox etc.

5. Hardware Requirement

5.1 Network Installation
5.2 Database Required
5.3 Hosting of the server required
5.4 Monthly fees for database maintenance.
5.5 Memory space

6. Development of the Software

6.1 Frontend Developer
    6.1.1 interactive user interface
6.2 Backend Developer
    6.2.1 Database Maintenance
    6.2.2 SSL protection
    6.2.3 Payment Gateway

7. Cost Analysis

7.1 Software
    - 20,000 to 40,000 one-time cost and 5,000 to 10,000 maintenance cost
7.2 Hardware
    - 8,000 to 10,000 INR per year

8. Future Scope

In future, this model will make travelling hassle-free and economical. It will be a bountiful ride for longer distances.

9. Limitations

9.1 Fake accounts
9.2 Fake Payments

10. References

10.1 https://www.uber.com/in/en/
10.2 https://www.olacabs.com/
10.3 https://ride.shuttl.com/
10.4 https://www.zoomcar.com/

EXPERIMENT NO- 2

SOFTWARE SPECIFICATION REQUIREMENT (SRS)

ONLINE CAB BOOKING SYSTEM (OCBS)

1. Introduction

Purpose:

This document is meant to delineate the features of OCBS, so as to serve as a guide to the developers on one hand and a software validation document for the prospective client on the other. The Online Cab Booking System (OCBS), for example OLA, UBER, SHUTTL, LYFT etc. are intended to provide hassle-free solutions for booking cabs at the click of a finger. It enables the customers to find themselves rides without having to physically wait for one which saves a lot of time. The driver module helps the drivers to find them potential riders at the nearby locations.

1.1 Scope:

This system allows the user to book cabs by logging into the system and entering their current and drop location. Based upon that, the system then finds the nearby cab.

1.2 Definitions, Acronyms and Abbreviations:

- **OCBS** – Online Cab Booking System
- **SRS** – Software Requirement Specification
- **GUI** – Graphical User Interface
- **Customer-** the person who books the cab through the customer application module
- **Driver-** the person who registers himself as the driver of the cab through the driver application module

1.3 References:

The document only covers the requirement specification for the online platform. This document does not provide any references to the other components of the online platform. All the external interfaces and the dependencies are also identified in this document.

1.4 Overview:

The system provides an effective method to book cabs according to the needs of the customers. The proposed system consists of two modules namely- Driver and

Customer. Driver module notifies them about the nearby potential customers. Customer module is slightly different as it provides an option of choosing a pick-up and a drop location, selecting the car type and then indicates the fare and ride time.

2. Overall Description

The online cab booking system (OCBS) is a social initiative which is designed to help the customers to reach at the destination in time and with convenience. It comes with an option of pool ride for economic and fuel-saving purposes.

The proposed system will also help in generating large-scale employment in different sectors like information technology (IT), administrative work, car rental and overall.

2.1 Product Perspective:

The model aims at providing door-to-door cab service for people who want a comfortable, economical and safe ride. The cab services are available 24x7 and very convenient for those who want to travel late nights or early mornings. It is a bountiful ride for longer distances.

2.2 Product Functions:

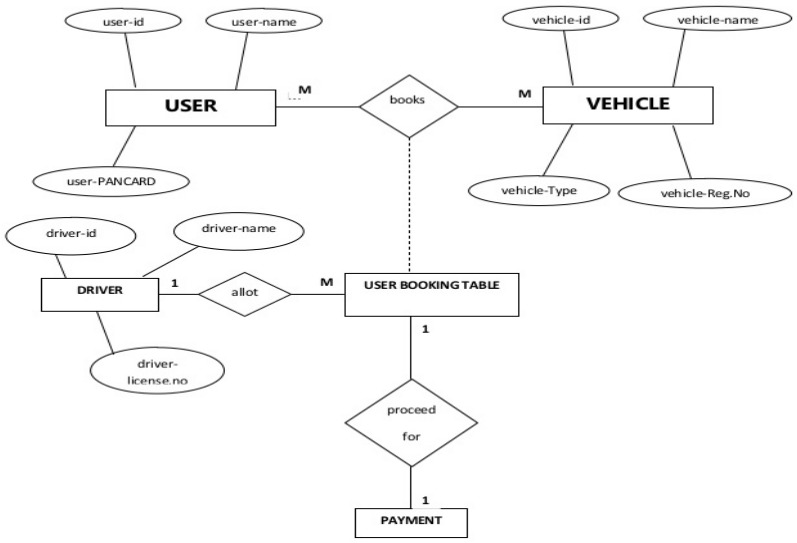
This document gives the procedural approach to how a user will be able to book a cab service at any time and any place.

The data represented in the online application will perform the following major functions:

I. For the person/user booking a cab
- Register themselves on the app via email or phone number. They are assigned a unique user id.
- Enter the pick-up and drop destinations
- Select car type
- Select payment mode
- Provide feedback for the ride and the driver

II. For drivers
- Register themselves in the driver module app. They are assigned a unique driver id.
- Register the vehicle along with the vehicle name, vehicle type, vehicle registration number

- Pick the customer from the nearby location
- Receive payment from the customer



2.3 User Classes and Characteristics:

Users have to download the application, register themselves using phone number or e-mail ID and enable location services. Users enjoy the facility to choose from various payment modes.

2.4 General Constraints:

A fully working internet connection is required to operate the application and track live location.

2.5 Assumptions and Dependencies:

Functionality of this application requires internet connection.

3. System Requirements

3.1 External Interfaces

3.1.1 User Interfaces

This software is designed in such a way that the total appearance of the application looks more user friendly. The customer as well as the driver have a login id and password to log into the app.

3.1.2 Hardware interfaces

The system requirements of the mobile devices determine the performance of the app. The app can support any number of users provided the database is large enough. The app can take space ranging from a few Kilobytes (KB) to Megabytes (MB)The app needs to be updated at all times to enjoy the benefit the newest features.

3.1.3 Software Interfaces

The customer and the driver interact with the system using their respective applications. A customer should be able to register on the app using his mail id or phone number in order to log in to his account and manage account information. A driver module also operates in the similar fashion for registering the driver and the vehicles with the corresponding details.

3.2 Communication Interfaces

The communication between the different modules of the system is important since they are interdependent. The administrator controls the communication between the user as well as the driver modules.

3.3 Design and Implementation Constraints

This will help the customers to access their ride details in the past and any future rides whenever necessary. They can also calculate the fare of the ride depending on the distance and the type of car selected. This project will help to smoother the process of the online cab bookings.

3.4 System Features

This project is to maintain the users' details, cab booking details and calculate the fare of the ride. The application consists of the following modules:

- **Administration module:** handles the backend and database of the applications. Moderates the customer and driver modules.
- **Customer Module:** the user who books the cab
- **Driver Module:** the user who own the vehicle and drives to the location of the customer

3.5 Software System Attributes

3.5.1 Availability

The software is available as soon as it is developed and the users are free to use all its features.

3.5.2 Maintainability

The system will be designed in maintainable manner and it will be easy to incorporate any new requirements in the individual modules.

3.5.3 Portability

The application is easily portable on the android as well as an iOS System.

3.5.4 Security

The application will be password protected. The users will have to enter the correct username or phone number along with the password in order to access the application. Also, their mobile numbers are protected using two-factor authentication.

4. Change Management Process

Department will call expertise to add new modules and update them using an existing interface.

5. Document Approved

Approved by the designated college authorities and IT cell.

EXPERIMENT NO. 3

AIM: To create a Class Diagram for Online Cab Booking System.

THEORY:

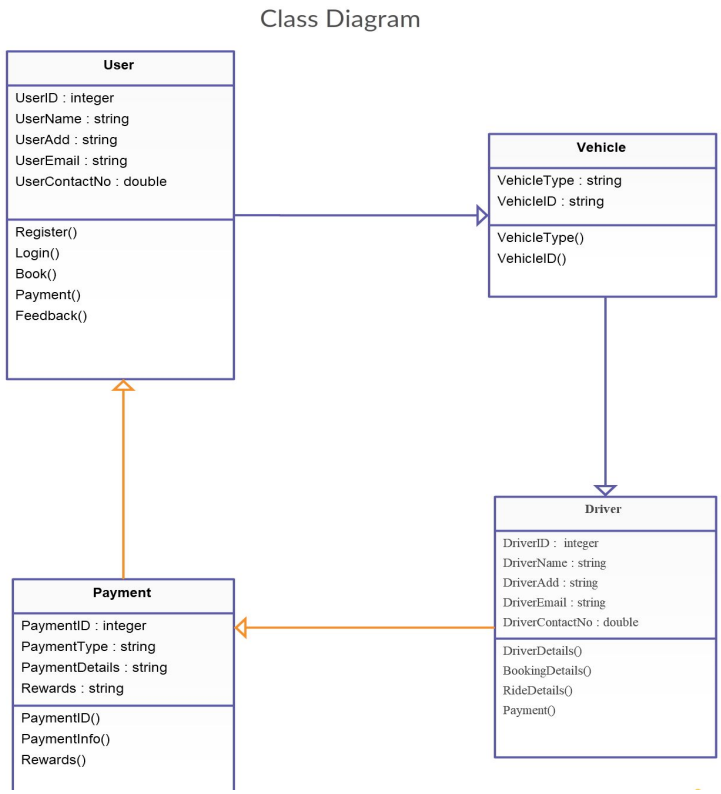
In software engineering, a class diagram in the Unified Modeling Language (UML) is **a type of static structure diagram** that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Purpose of Class Diagrams:

1. Shows the static structure of classifiers in a system
2. Diagram provides a basic notation for other structure diagrams prescribed by UML
3. Helpful for developers and other team members too
4. Business Analysts can use class diagrams to model systems from a business perspective

A UML class diagram is made up of:

- A set of classes and
- A set of relationships between classes

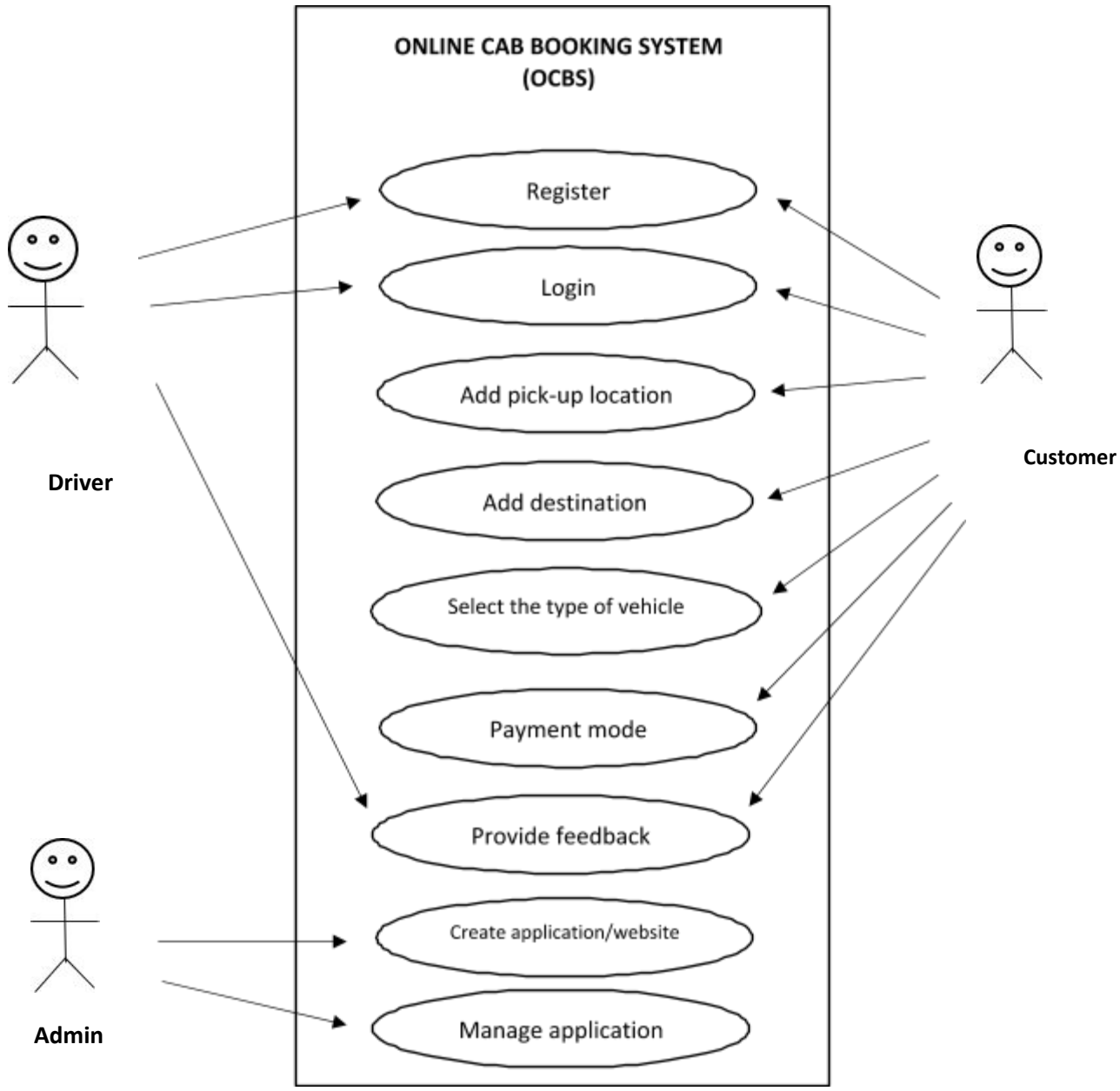


EXPERIMENT NO. 4

AIM: To create a Use Case Diagram for Online Cab Booking System

THEORY:

Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

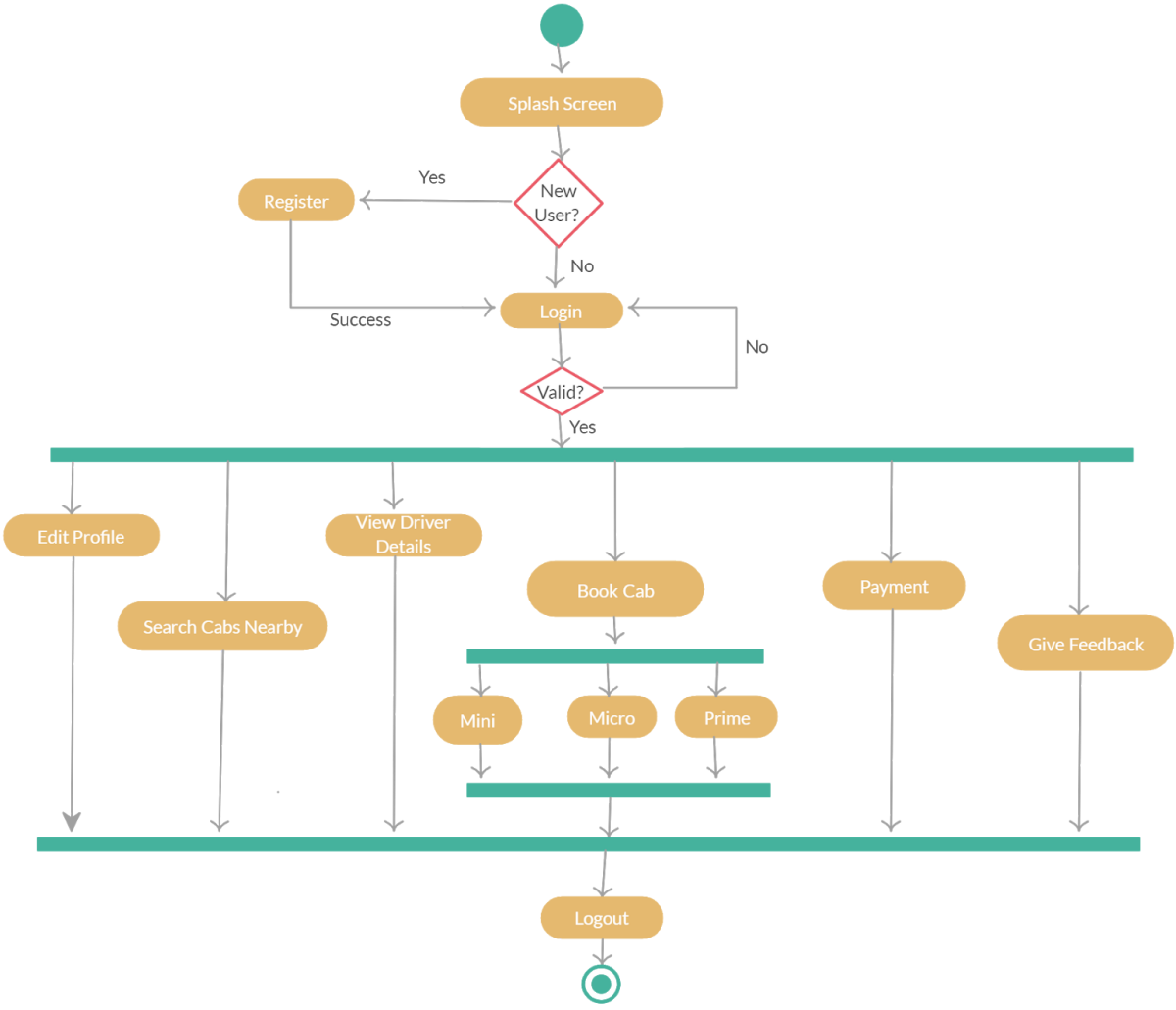


EXPERIMENT NO. 5

Aim – Draw and implement Activity Diagram on Online Cab Booking System.

Theory

Activity diagram is another important behavioural diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of the flow chart that modelling the flow from one activity to another activity. Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction.

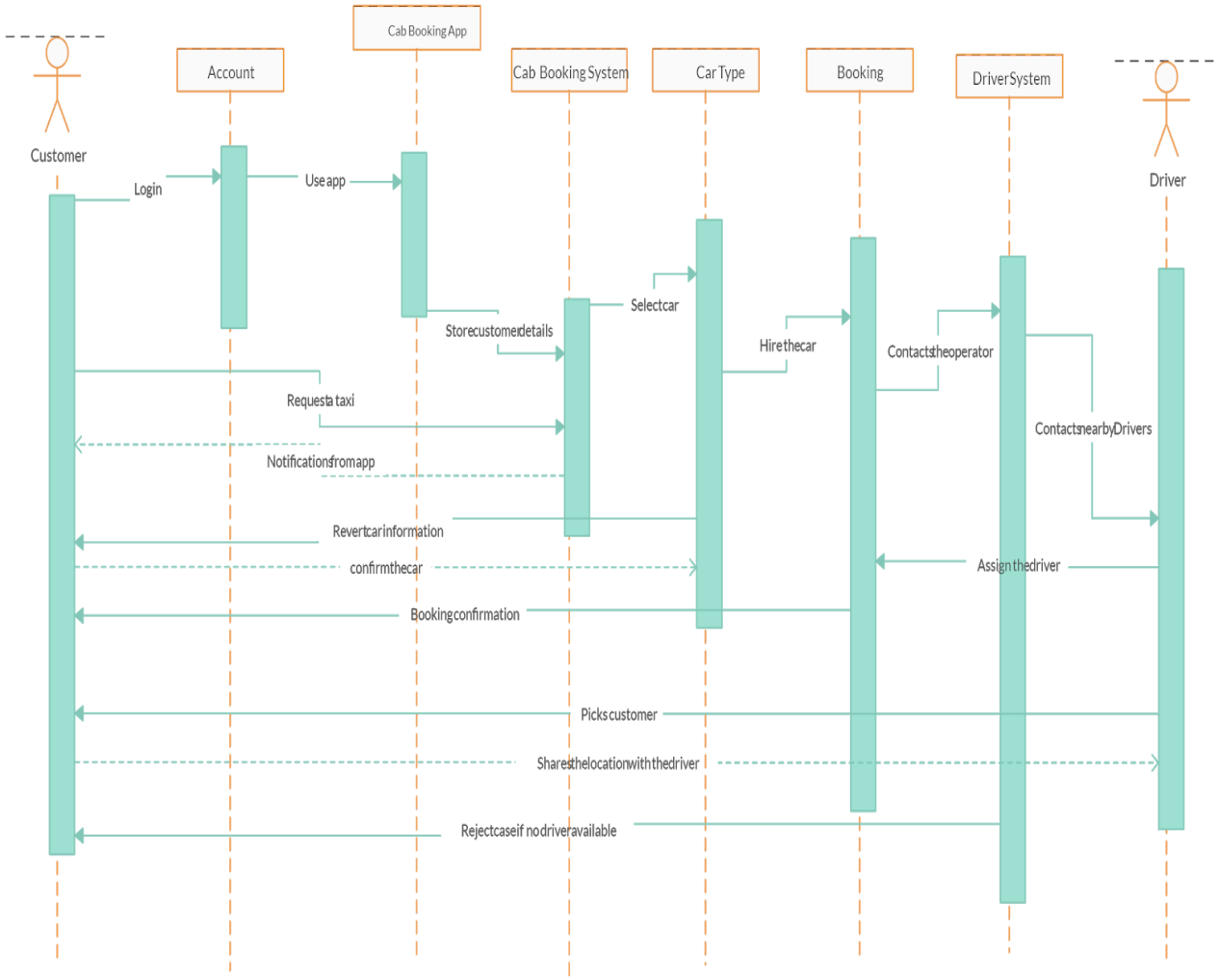


EXPERIMENT NO. 6

Aim – Draw and Implement Sequence Diagram on Online Cab Booking System.

Theory

A sequence diagram simply depicts an interaction between objects in sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

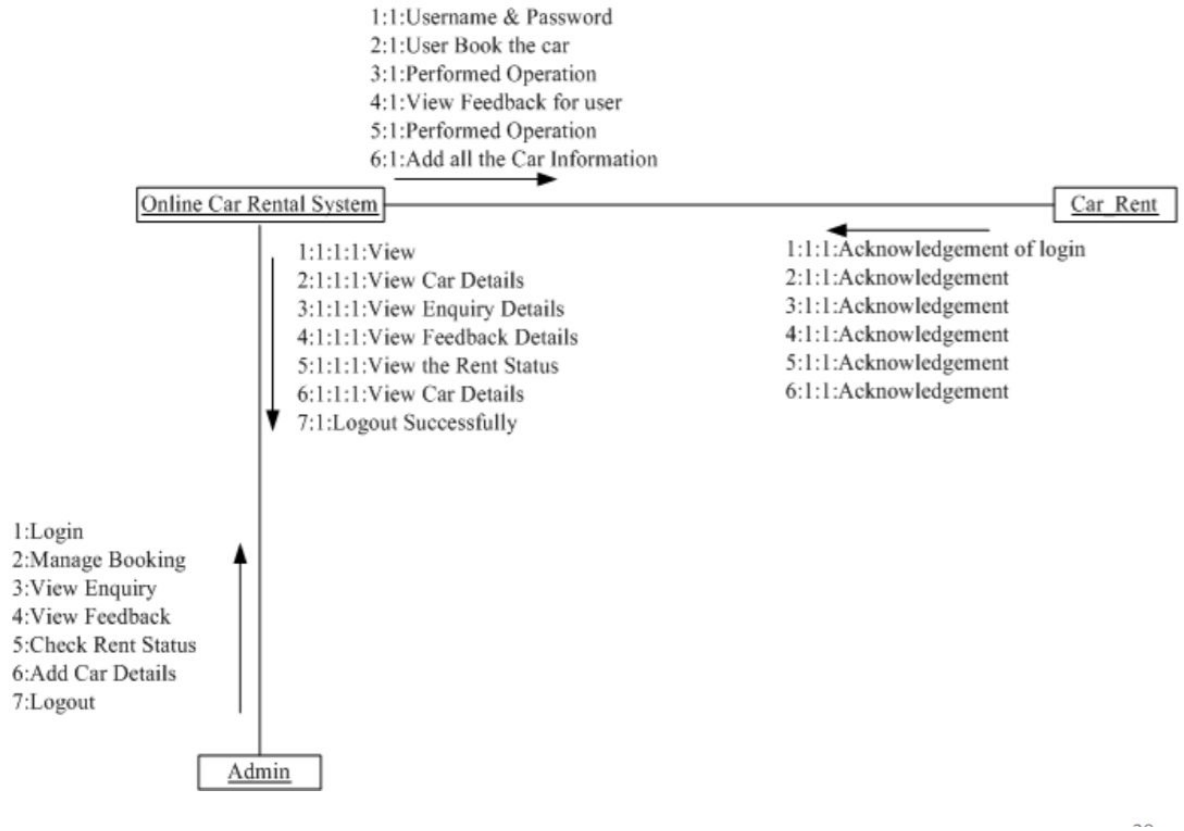


EXPERIMENT NO. 7

Aim – Draw and Implement a Collaboration Diagram on Online Cab Booking System.

Theory

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object. Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction. A model is then built using the relationships between those elements. Several vendors offer software for creating and editing collaboration diagrams.

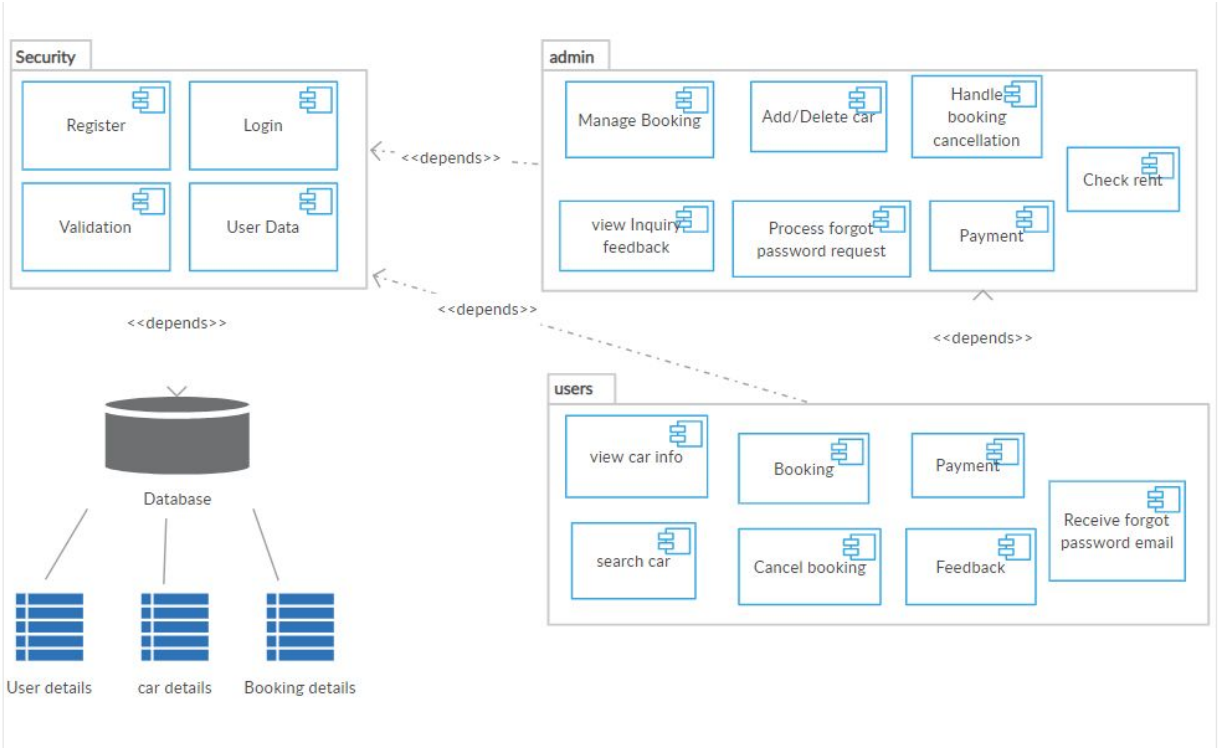


EXPERIMENT NO. 8

Aim – Draw and Implement Component Diagram on Online Cab Booking System.

Theory

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development. In the first version of UML, components included in these diagrams were physical: documents, database table, files, and executables, all physical elements with a location. In the world of UML 2, these components are less physical and more conceptual stand-alone design elements such as a business process that provides or requires interfaces to interact with other constructs in the system. The physical elements described in UML 1, like files and documents, are now referred to as artefacts.



EXPERIMENT NO. 9

Aim: Draw and Implement Deployment Diagram on Online Cab Booking System.

Theory:

A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes.To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI). The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have subnodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

There are two types of Nodes:

1. Device Node
2. Execution Environment Node

Device nodes are physical computing resources with processing memory and services to execute software, such as typical computers or mobile phones. An execution environment node (EEN) is a software computing resource that runs within an outer node and which itself provides a service to host and execute other executable software elements.