PROJECT REPORT

on

<Project Name>

from

<Company Name>

Towards partial fulfillment of the requirements

for the award of degree of

Master of Computer Applications

from

Babu Banarasi Das University

Lucknow

Academic Session 2023 - 24
School of Computer Applications

I Floor, EC-Block, BBDU, BBD City, Faizabad Road, Lucknow (U. P.) INDIA 226028
PHONE: HEAD: 0522-3911127, 3911321 **Dept. Adm. & Exam Cell:** 0522-3911326 **Dept. T&P Cell:** 0522-3911128; **E-Mail:** head.sca@gmail.com
w w w . b b d u . a c . i n

PROJECT REPORT

on

<Project Name>

from

<Organization Name>

Towards partial fulfillment of the requirements for the award of degree of

Master of Computer Applications

from

Babu Banarasi Das University Lucknow

Developed and Submitted by
<Student Name>

Under Guidance of
<Guide name>

Academic Session 2023 - 24

School of Computer Applications

UNDER TAKING

This is to certify that Project Report entitled

<Project Name>

being submitted by

Towards the partial fulfillment of the requirements
for the award of the degree of

Master of Computer Applications

from

Babu Banarasi Das University
Lucknow

Academic Year 2023-24

is a record of the student's own work carried out at

and to the best of our knowledge the work reported herein does not form a part of any other thesis or work on the basis of which degree or award was conferred on an earlier occasion to this or any other candidate.

Authorized Signatory
Organization Name

Students Signature

DECLARATION

Acknowledgement

INDEX

1. INTRODUCTION

1.1 Background:

Nowadays, there are online vehicle reservations that give much benefit to users. A rental service is a service in which customers arrive to request the hire of a rental unit. It is more convenient than carrying the cost of owning and maintaining the unit. This project is designed to be used by a Vehicle Rental Company specializing in renting vehicles to customers. It is an online system through which customers can view available vehicles, register, view profiles and book vehicles. A vehicle rental or vehicle hire agency is a company that rents automobiles for a short period for a fee whether in a few hours a few days or a week. It is an extended form of a rental shop, often organized with numerous local branches (which allow a user to return a vehicle to a different location), and primarily located near airports or busy city areas and often complemented by a website allowing online reservations. Vehicle rental agencies primarily serve people who have vehicles that are temporarily out of reach or out of service, for example, travelers who are out of town or owners of damaged or destroyed vehicles who are awaiting repair or insurance compensation. Because of the variety of sizes of the vehicles, our company serves the self-moving industry needs, by renting vans or trucks, and in certain markets other types of vehicles such as motorcycles or scooters are also offered. In short, it is a system designed especially for large, premium and small vehicle rental businesses. The vehicle rental system provides complete functionality for listing and booking vehicles.

1.2 Objective:

Today's world is a computer world because most of the work is done with the help of computer. Dependency on computer is behind the few reasons. We cannot easily manage to store large number of data or information single handle. If we will be needing some information or data in urgency then we cannot manage in manually these works are very difficult if we cannot use computer.

This software is updating the manual chemist Inventory system to an automated Inventory system. So that organization can manage their record in efficiently and organize them.

The main objective is to automate non-computer environment:

- To save manpower.
- It will speed the processing of data and transaction.
- It will provide best security features such as provisions of passwords
- To transform the manual process of hiring vehicle to a computerize system.
- To validate the Rental vehicle system using user satisfaction test.

- To produce the documentation such as Software Requirement Specification (SRS), Software Design Description (SDD) as system development reference.
- To produce a web-based system that allow customer to register and reserve vehicle online and for the company to effectively manage their vehicle rental business.
- To ease customer's task whenever they need to rent a vehicle.

System Objective: Today's world is computer world because most of work is doing with the help of computer. Dependency on computer is behind the few reasons. We cannot easily manage to store large number of data or information single handle. If we will be needing some information or data in urgency then we cannot manage in manually these works are very difficult if we cannot use computer.

System Context: This section clearly depicts the environment and boundaries of the Vehicle Rental System and the entities with which it interacts. It helps us see how the system fits into the existing scheme of things. What the system will do by itself.

Functional Requirement: This Software must request Username and Password for access to data, only after authentication will allow access to the system. The Software must allow input of products data from administrator and secured access. Requirement analysis is a software engineering technique that is composed of the various tasks that determine the needs or conditions that are to be met for a new or altered product, taking into consideration the possible conflicting requirements of the various users. Functional requirements are those requirements that are used to illustrate the internal working nature of the system, the description of the system, and explanation of each subsystem. It consists of what task the system should perform, the processes involved, which data should the system holds and the interfaces with the user.

The functional requirements identified are:

a) ***Customer's registration:*** The system should allow new users to register online and generate membership card.

b) ***Online reservation of vehicles:*** Customers should be able to use the system to make booking and online reservation.

c) ***Automatic update to database once reservation is made or new customer registered:*** Whenever there's new reservation or new registration, the system should be able update the database without any additional efforts from the admin.

d) ***Feedbacks to customers:*** It should provide means for customers to leave feedback.

Non-Functional Requirement: In this Software Input error will be returned in red with appropriate message box. System should automatically update after every transaction. It describes aspects of the system that are concerned with how the system provides the functional requirements. They are:

a) ***Security:*** The subsystem should provide a high level of security and integrity of the data held by the system, only authorized personnel of the company can gain access To the Company's secured page on the system; and only users with valid password and username can login to view user's page.
b) ***Performance and response time:*** The system should have high performance rate when executing user's input and should be able to provide feedback or response within a short time span usually 50 seconds for highly complicated task and 20 to 25seconds for less complicated task.
c) ***Error handling:*** Error should be considerably minimized and an appropriate error message that guides the user to recover from an error should be provided. Validation of user's input is highly essential. Also, the standard time taken to recover from an error should be 15 to 20 seconds.
d) ***Availability:*** This system should always be available for access at 24 hours, 7 days awake. Also, in the occurrence of any major system malfunctioning, the system should be available in 1 to 2 working days, so that the business process is not severely affected.
e) ***Ease of use:*** Considered the level of knowledge possessed by the users of this system, a simple but quality user interface should be developed to make it easy to understand and required less training.

1.3 Purpose and Scope

Purpose: The purpose of this document is to specify requirements and to give guidelines for the development of above said project. In particular it gives guidelines on how to prepare the above said project. The advancement in Information Technology and internet penetration has greatly enhanced various business processes and communication between companies (services provider) and their customers of which car rental industry is not left out.

This Vehicle Rental System is developed to provide the following services:

a) ***Enhance business processes:*** To be able to use internet technology to project the rental company to the global world instead of limiting their services to their local domain alone, thus increase their return on investment (ROI).

b) ***Online vehicle reservation:*** Tools through which customers can reserve available vehicles online prior to their expected pick-up date or time.

c) ***Customer's registration:*** A registration portal to hold customer's details, monitor their transaction and used same to offer better and improve services to them.

d) ***Group bookings:*** Allows the customer to book space for a group in the case of weddings or corporate meetings (Event management).

Scope: This project traverses a lot of areas ranging from business concept to computing field and required to perform several researches to be able to achieve the project objectives.

The area covers include:

- This includes study on how the car rental business is being done, process involved and opportunity that exist for improvement.
- Java Technology used for the development of the application.
- General customers as well as the company's staff will be able to use the system effectively.
- Web-platform means that the system will be available for access 24/7 except when there is a temporary server issue which is expected to be minimal
- The monitoring of the vehicle activity and the overall business becomes easy and includes the least of paper work.
- The software acts as an office that is open 24/7. It increases the efficiency of the management at offering quality services to the customers.
- It provides custom features development and support with the software.

2. SURVEY OF TECHNOLOGY

2.1 Java:

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

Java is a powerful and versatile programming language that has been widely used since its introduction in 1995 by Sun Microsystems. Known for its simplicity, portability, and robustness, Java has become one of the most popular programming languages in the world, powering everything from enterprise applications to mobile apps, web development, and embedded systems. Java's design philosophy revolves around the concept of "write once, run anywhere" (WORA), meaning that Java programs can be developed on one platform and executed on any other platform without modification, thanks to its platform-independent nature. One of the key features of Java is its object-oriented programming (OOP) paradigm, which promotes the concept of objects and classes as fundamental building blocks of software development. In Java, everything is treated as an object, which encapsulates data and behavior within a single entity. Classes serve as blueprints for creating objects, defining their attributes (fields) and methods (functions) that operate on those attributes. This modular and reusable approach to programming makes Java code more organized, maintainable, and scalable, facilitating the development of large-scale software systems. Another hallmark of Java is its strong emphasis on platform independence, achieved through the use of the Java Virtual Machine (JVM). Java source code is compiled into an intermediate bytecode format that is executed by the JVM, rather than directly by the underlying hardware or operating system. This abstraction layer shields Java programs from platform-specific details, allowing them to run on any system that has a compatible JVM installed, whether it's a Windows PC, macOS, Linux machine, or even

embedded devices like smartphones or IoT devices. This cross-platform compatibility has made Java an ideal choice for developing software that needs to run on diverse environments.

Java's standard library, known as the Java Development Kit (JDK), provides a rich set of APIs (Application Programming Interfaces) that simplify common programming tasks and enable developers to build robust and feature-rich applications more efficiently. The JDK includes packages for essential functionalities such as input/output operations, networking, multithreading, database connectivity, GUI (Graphical User Interface) development, and more. Additionally, Java's extensive ecosystem boasts a wide range of third-party libraries, frameworks, and tools contributed by the vibrant Java community, further enhancing the language's capabilities and versatility. While Google has introduced Kotlin as an alternative programming language for Android development, Java remains a cornerstone of the Android platform and continues to be widely used in the Android development community. Furthermore, Java's versatility extends to web development, where frameworks like Spring, JavaServer Faces (JSF), and Apache Struts provide powerful tools for building scalable and maintainable web applications. These frameworks leverage Java's enterprise features, such as dependency injection, aspect-oriented programming, and transaction management, to streamline the development process and ensure the reliability and performance of web-based systems. Additionally, Java's support for server-side programming makes it a popular choice for backend development in conjunction with technologies like Java EE (Enterprise Edition), Servlets, and JSP (JavaServer Pages).

2.2 Versions:

Java versions encompass the various iterations of the Java programming language and its accompanying platform, which include the Java Development Kit (JDK) and the Java Runtime Environment (JRE). Each release typically introduces new functionalities, enhancements, and bug fixes. The latest version of Java is Java 22 or JDK 22 released on March, 19th 2024. JDK 22 is a regular update of Java SE platform. And JDK 21 is currently the latest long-term support release (LTS), replacing JDK 17 which is the previous LTS of the Java SE platform.

From the first version released in 1996 to the latest version JDK 22 available to the public since March 2024, the Java platform has been actively being developed for more than 28 years. Many changes and improvements have been made to the technology over the years. The following table summarizes all versions of Java SE from its early days to the latest.

Version	Release date	End of Public Updates (Free)	End of Extended Support (Paid)
JDK 1.0	23rd January 1996	May 1996	—
JDK 1.1	18th February 1997	October 2002	—
J2SE 1.2	4th December 1998	November 2003	—
J2SE 1.3	8th May 2000	March 2006	—
J2SE 1.4	13th February 2002	October 2008	—
J2SE 5.0	30th September 2004	October 2009	—
Java SE 6	11th December 2006	April 2013	October 2018 for Oracle
Java SE 7	28th July 2011	July 2015	July 2022 for Oracle
Java SE 8 (LTS)	18th March 2014	April 2019 for Oracle	December 2030 for Oracle
Java SE 9	21st September 2017	March 2018	—
Java SE 10	20th March 2018	September 2018	—
Java SE 11 (LTS)	25th September 2018	April 2019 for Oracle	January 2032 for Oracle
Java SE 12	19th March 2019	September 2019	—
Java SE 13	17th September 2019	March 2020	—
Java SE 14	17th March 2020	September 2020	—

Java SE 15	16th September 2020	March 2021	—
Java SE 16	16th March 2021	September 2021	—
Java SE 17 (LTS)	14th September 2021	September 2024 for Oracle	September 2029 for Oracle
Java SE 18	22nd March 2022	September 2022	—
Java SE 19	20th September 2022	March 2023	—
Java SE 20	21st March 2023	September 2023	—
Java SE 21 (LTS)	19th September 2023	September 2026 for Oracle	September 2031 for Oracle
Java SE 22	19th March 2024	September 2024	—

2.3 MySQL:

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

MySQL is extremely popular for:

Ecommerce: Many of the world's largest ecommerce applications (for example, Shopify, Uber, and Booking.com) run their transactional systems on MySQL. It's a popular choice for managing user profiles, credentials, user content, financial data including payments, and fraud detection.

Social platforms: Facebook, Twitter, and LinkedIn are among the world's largest social networks that rely on MySQL.

Content management: Unlike single-purpose document databases, MySQL enables both SQL and NoSQL with a single database. The MySQL Document Store enables CRUD operations and the power of SQL to query data from JSON documents for reporting and analytics.

SaaS and ISVs: More than 2,000 ISVs, OEMs, and VARs, including Ericsson, F5, and IBM, rely on MySQL as their embedded database to make their applications, hardware, and appliances more competitive, bring them to market faster, and lower their cost of goods sold. MySQL is also the database behind popular SaaS applications, including Zendesk and HubSpot.

On-premises applications with MySQL Enterprise Edition: MySQL Enterprise Edition includes the most comprehensive set of advanced features, management tools, and technical support to achieve the highest levels of MySQL scalability, security, reliability, and uptime.

Release History:

Release	General availability	Latest minor version	Latest release	End of support
5.1 LTS	14 November 2008; 15 years ago	5.1.73	2013-12-03	Dec 2013
5.5 LTS	3 December 2010; 13 years ago	5.5.62	2018-10-22	Dec 2018
5.6 LTS	5 February 2013; 11 years ago	5.6.51	2021-01-20	Feb 2021
5.7 LTS	21 October 2015; 8 years ago	5.7.44	2023-10-25	Oct 2023
8.0 LTS	19 April 2018; 6 years ago	8.0.37[1]	2024-04-30	Apr 2026
8.1 IR	18 July 2023; 10 months ago	8.1.26	2023-07-18	Oct 2023
8.2 IR	25 October 2023; 6 months ago	8.2.0	2023-10-25	Jan 2024
8.3 IR	16 January 2024; 4 months ago	8.3.0	2024-01-16	Apr 2024
8.4 LTS	30 April 2024; 20 days ago	8.4.0	2024-04-30	Apr 2032

3. REQUIREMENT ANALYSIS

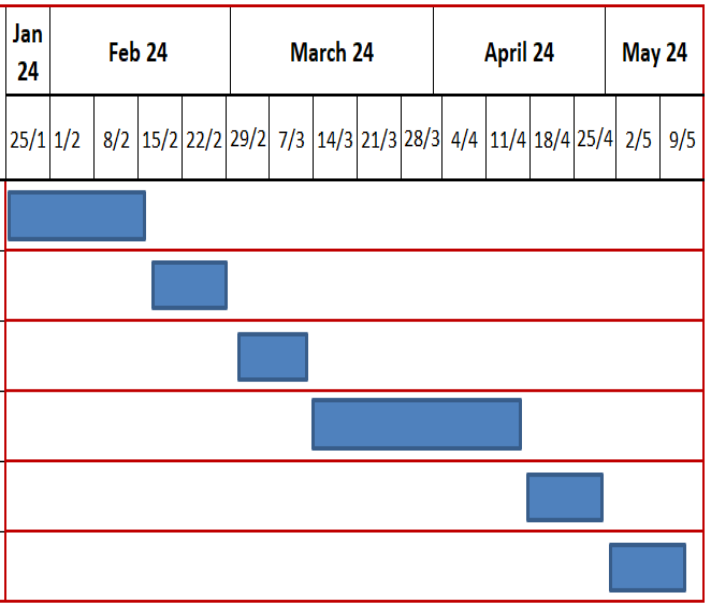
3.1 Problem Definition:

In a vehicle rental service, a vehicle that can be used temporarily by paying a fee during a specified period. Getting a rental vehicle helps people get around despite the fact they do not have access to their own personal vehicle or don't own a vehicle at all. The individual who needs a car must contact a rental vehicle company and contract out for a vehicle. This system increases customer retention and simplify vehicle and staff management.

3.2 Planning and Scheduling:

Pert Chart: A project plan needs to be created to ensure the timely completion of the project. As part of project analysis, we break the project down to a number of stages and use a Gantt chart and PERT chart to describe specific tasks and status. The Work Breakdown Structure of our proposed system "Vehicle Rental System" is shown below.

SN	Task Name	Start	Finish	Duration
1	Technology Learning	25-01-2024	17-02-2024	20d
2	Requirement Gathering & Analysis	18-02-2024	29-02-2024	10d
3	Designing	01-03-2024	12-03-2024	10d
4	Coding	13-03-2024	18-04-2024	30d
5	Testing	19-04-2024	30-04-2024	10d
6	Implementation	01-05-2024	10-05-2024	10d



3.3 Requirements Specification:

Software Requirements

- Java/JDK
- Net Beans
- MySQL
- SQL YOG

Hardware Requirements

- Pentium IV Processor
- 512 MB RAM
- 40 GB HDD
- Color Monitor
- Keyboard, Mouse

3.4 Preliminary Product Description:

It would be a multi user account system in key features. There will three types of main modules in the system.

- Admin
- Customer
- Invoice Management

Admin: Like every other management system, the vehicle rental management system will have the admin. The admin will the entity that will monitor the activities and the records of whole system. Following are some main facts related to the admin of the system. There will be only one admin in the system. Admin can view other user's profile. The admin will have the power to delete any other users form the records or update the data of any other users. Any vehicle or the payment deal will be approved by the admin.

Customers: Customers are the reason why I feel to introduce the vehicle rental management system, to make their journey wonderful, to get them fit for the environment they are traveling into.

a) ***View the vehicles:*** You, as a customer, can observe the lists of vehicles available in the inventory. The user can filter the records of the vehicle based on:

b) ***Price:*** The budget is an important factor. It will be easier to choose a vehicle rather than wondering what if I choose this vehicle and the price is higher. No tension at all, you can analyze the vehicle record and choose your best vehicle.

c) ***Popular vehicles:*** If you want to take the vehicle which is popular in the system rather than thinking about the fact how this vehicle would perform, you better look into it. The already registered customers have given the feedback of their vehicle driving experience.

d) ***Vehicle brand:*** If you are into brand, you can view the vehicles of your favorite brand. I have taken that too into the account. The vehicle brand can be BMW, Mercedes, Aston Martin, Honda, Mahindra etc. Just pick your pick.

e) ***Rent a vehicle:*** After you have selected your favorite vehicle, you can fill the vehicle rental form which is available online. You just have to fill some details like for how many days you want to rent the vehicle, or if you want to rent on hourly basis, the vehicle details of the vehicle model you want to rent. After completing the rental form, you can pay suitable amount using net banking, your credit / debit vehicle.

f) ***Return a rented vehicle:*** The customer can return a vehicle and if all the payments are cleared and the parts of the vehicles are not damaged, a number will be provided to the customer so that the customer can enter that number into the return vehicle section and the record is cleared from the rent a vehicle system and is moved to rental vehicle history.

g) ***View rental history:*** You as a user can view the history of the vehicle you have rented in the vehicle rental management system. You can keep track of the amount you have spent, the vehicle you have driven, the number of vehicles you have rented etc.

h) ***Feedback:*** You as a user can share your experience with the vehicle rental management system. How much you loved it, or hated it. You can give the stars and provide some comments so that the dealer can assist the customers to their best capability they can.

Invoice Management: After the amount is paid by the customers, the invoice department will generate the bill of the vehicle used and will reflect into the customers' account. This department will also keep the receipts of newly vehicle is brought to the system so that it can further be used for analysis purpose.

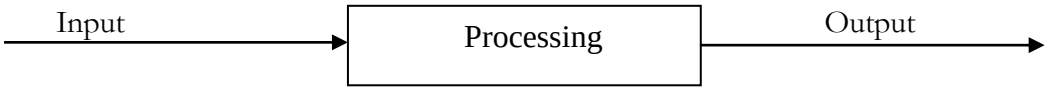
3.5 Conceptual Models System Architecture Design:

Defining A System: Defining a system is a critical initial phase that sets the stage for the entire development process. This phase involves conceptualizing, designing, and describing the software solution that will address the identified problem or need. It serves as a roadmap for the development team and stakeholders, guiding the subsequent stages of the SDLC. Defining a system encompasses several key aspects, each crucial for the success of the project. First and foremost, defining a system involves understanding and articulating the problem or need that the software aims to address. This requires collaboration among stakeholders, including clients, end-users, project sponsors, and domain experts. Through discussions, interviews, surveys, and other techniques, stakeholders identify the pain points, challenges, and opportunities that the software solution should target. This process of problem definition provides the context and motivation for the development effort, ensuring that the resulting system meets the real-world needs of its users. Once the problem is defined, the next step is to establish the scope of the system. The scope defines the boundaries of what the software will and will not do, outlining the

functionalities, features, and components that will be included in the final product. Stakeholders prioritize requirements based on their importance, urgency, and feasibility, helping to define a clear scope for the project. This ensures that the development team stays focused on delivering value to the users while managing expectations and avoiding scope creep.

With the scope defined, attention turns to the architecture of the system. System architecture outlines the high-level structure and design of the software solution, including the arrangement of components, modules, and subsystems, as well as the interactions and interfaces between them. It provides a blueprint for developers to follow during the implementation phase, guiding decisions about technology stack, frameworks, and design patterns. A well-designed architecture lays the foundation for a scalable, maintainable, and extensible system, capable of evolving with changing requirements and technologies. In parallel with defining the architecture, the system's data model is designed. The data model describes the organization and structure of the data that the system will manage, including data entities, attributes, relationships, and constraints. Designing a data model ensures that the system can effectively store, retrieve, and manipulate data to meet the requirements of the users. It lays the groundwork for database design, data storage, and data access mechanisms, ensuring data integrity, consistency, and security. Another crucial aspect of defining a system is designing the user interface (UI). The UI design focuses on creating intuitive and visually appealing interfaces that enable users to interact with the system. This involves designing screens, forms, menus, and other elements to facilitate user input and feedback.

Lastly, defining a system involves specifying both functional and non-functional requirements. Functional requirements describe the specific actions and behaviors that the system must perform to fulfill its intended purpose. These requirements define the functionalities and features that users expect from the software, such as data processing, calculations, reporting, and integration with other systems. Non-functional requirements address the quality attributes of the system, such as performance, reliability, security, scalability, and maintainability. These requirements define the criteria for evaluating the system's overall effectiveness and usability beyond its functional capabilities.



System Development Life Cycle: The System development life cycle (SDLC), or Software development processing systems engineering, information systems and software engineering, is a process of creating or altering information systems, and the models and methodologies that

people use to develop these systems. In software engineering, the SDLC concept underpins many kinds of software development methodologies. It provides a structured framework for managing the entire software development process, from the initial conception of an idea to the final delivery of a functional product. SDLC consists of several phases, each with its specific objectives, activities, and deliverables, ensuring that the software meets quality standards, user requirements, and business goals.

These methodologies form the framework for planning and controlling the creation of an information system the process. Broadly, following are the different activities to be considered while defining the system development life cycle for the said project:

- Problem Definition
- System Analysis
- Study of existing system
- Drawback of the existing system
- Proposed system
- System Requirement study
- Data flow analysis
- Feasibility study
- System design
- Input Design (Database & Forms)
- Updating
- Query /Report design
- Administration
- Testing
- Implementation
- Maintenance

System Analysis: System analysis is a critical phase in the software development life cycle (SDLC) that involves studying, understanding, and defining the requirements of a proposed software system or application. It is a systematic approach to investigating, identifying, and documenting the needs, objectives, and constraints of the system to be developed. System analysis serves as the foundation for the subsequent phases of software development, providing the necessary insights and information for designing, implementing, and testing the system effectively. The process of system analysis begins with gathering information from stakeholders,

including end-users, clients, managers, and domain experts, to gain a comprehensive understanding of the problem domain and the goals of the proposed system. This often involves conducting interviews, surveys, workshops, and other techniques to elicit requirements and capture stakeholders' perspectives on what the system should accomplish. Requirements gathering is a collaborative effort that aims to identify both functional and non-functional requirements, as well as any constraints or dependencies that may impact the design and implementation of the system. Once the requirements have been collected, system analysts analyze and prioritize them to determine their significance and feasibility. This involves evaluating the scope of the project, assessing the potential risks and benefits, and identifying any conflicting or ambiguous requirements that may need further clarification. Requirements analysis requires careful attention to detail and a deep understanding of the domain in which the system will operate, as well as consideration of factors such as cost, schedule, and available resources.

During the analysis phase, system analysts also work to develop a conceptual model of the system that captures its key components, processes, and interactions. This may include creating diagrams, such as data flow diagrams, use case diagrams, or entity-relationship diagrams, to represent the structure and behavior of the system from a high-level perspective. The conceptual model serves as a visual aid for stakeholders to understand the proposed system and provides a basis for further refinement and elaboration during the design phase. Throughout the analysis phase, system analysts collaborate closely with stakeholders to validate requirements, gather feedback, and ensure that the proposed solution aligns with the needs and expectations of its users. This may involve conducting reviews, demonstrations, or prototypes to solicit input and verify that the system will meet its intended objectives. Effective communication and collaboration are essential for building consensus and ensuring that all stakeholders have a shared understanding of the system requirements and design.

In summary, system analysis is a systematic and collaborative process of studying, understanding, and defining the requirements of a software system or application. It involves gathering information from stakeholders, analyzing and prioritizing requirements, developing a conceptual model of the system, evaluating alternative solutions, and making recommendations for the best course of action. By providing a clear understanding of the problem domain and the goals of the proposed system, system analysis lays the groundwork for the successful development and implementation of software solutions that meet the needs of their users.

System Design: System design is a comprehensive process that encompasses the creation of a blueprint for the development and implementation of a software system or application. It involves translating requirements gathered during the initial stages of software development into

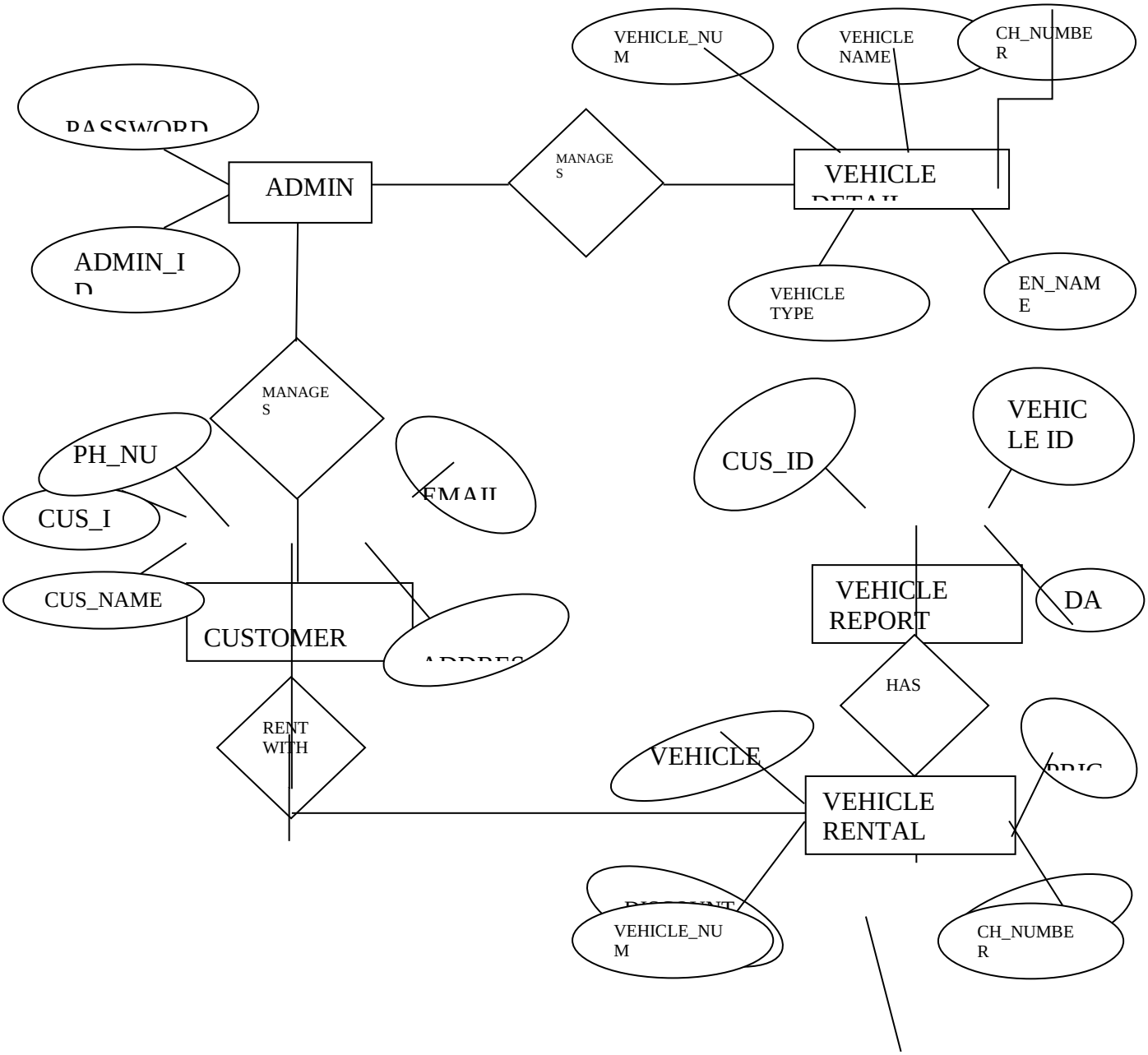
a detailed plan that outlines how the system will function, how its components will interact, and how it will be built and deployed. System design is a crucial phase in the software development life cycle (SDLC) as it sets the foundation for the construction and operation of the final product. At its core, system design involves making a series of decisions about the architecture, components, modules, interfaces, and data structures of the system. These decisions are guided by the functional and non-functional requirements of the system, as well as considerations such as scalability, reliability, performance, security, and maintainability. System designers must strike a balance between meeting these requirements while also ensuring that the system is feasible to implement within the constraints of time, budget, and available technology. The first step in system design is to analyze and understand the requirements of the system. This involves gathering information from stakeholders, including end-users, clients, and domain experts, to identify the goals, objectives, and constraints of the system. Requirements analysis helps to define the scope of the system and establish a clear understanding of what needs to be accomplished. Once the requirements are understood, system designers begin the process of conceptualizing the system architecture. This involves defining the overall structure of the system, including its high-level components, modules, and their interactions. Architecture design focuses on identifying the key subsystems and their responsibilities, as well as defining the interfaces between them. At this stage, designers may use techniques such as architectural patterns, such as client-server, layered, or microservices architecture, to organize and structure the system effectively. With the architecture in place, system designers move on to detailed design, where they flesh out the internal workings of each component and module. This involves specifying the algorithms, data structures, and logic necessary to implement the system's functionality. Design decisions at this level are guided by principles of modularity, encapsulation, and information hiding, which aim to promote reusability, maintainability, and flexibility. Once the design is complete, system designers document their decisions and create detailed design specifications that serve as a guide for developers during the implementation phase. These specifications may include architectural diagrams, data models, interface definitions, algorithms, and other design artifacts. Clear and comprehensive documentation is essential for ensuring that the system is implemented correctly and that all stakeholders have a shared understanding of its design and functionality.

Finally, system designers may also conduct design reviews and evaluations to validate the design against the requirements and identify any potential issues or improvements. Iterative refinement of the design may be necessary as new information becomes available or as the project

progresses. Ultimately, the goal of system design is to create a robust, scalable, and maintainable solution that meets the needs of its users and stakeholders.

Entity Relation Diagram: The Entity Relation Model or Entity Relation Diagram (ERD) is a data model or diagram for high-level description of conceptual data model, and it provides a graphical notation for representing such data models in the form of entity relationship diagrams. Such models are typically used in the first stage of Management information system design; they are used for example, to describe information needs and/ or the type of information that is to be stored in the Database during the requirement analysis. The data modeling technique, however, can be used to describe any ontology (i.e. an overview and classification of used term and their relationships) for a certain universe of discourse (i.e. area of interest). At the heart of an ER diagram are entities, which are objects or concepts with independent existence and properties that are relevant to the database. Entities can represent real-world objects like people, places, or things, or they can be abstract concepts like accounts or transactions in a banking system. Each entity is depicted as a rectangle in the ER diagram, labeled with its name.

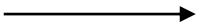
Attributes are the properties or characteristics of entities that are stored in the database. These attributes describe the features of an entity and are represented as ovals connected to their respective entities by lines. For example, in a database for a university, a "Student" entity may have attributes such as "Student ID," "Name," "Date of Birth," and "Major." Relationships define how entities interact with each other within the database. They represent the associations between entities and are crucial for understanding the connections between different parts of the system. Relationships are illustrated as lines connecting entities, typically labeled with verbs or phrases that describe the nature of the association. For instance, in a library database, a "Borrower" entity may have a relationship with a "Book" entity labeled as "Borrows," indicating that a borrower can borrow multiple books. To represent a many-to-many relationship in an ER diagram, it is common to introduce a junction entity, also known as an associative entity or a linking table. This junction entity resolves the many-to-many relationship into two one-to-many relationships, simplifying the structure of the database. Keys are crucial for maintaining data integrity and ensuring efficient database operations, while cardinality constraints help define the nature of the relationships between entities.Overall, ER diagrams provide a powerful visual tool for database designers to analyze, design, and communicate the structure of a database system. By representing entities, attributes, and relationships in a clear and concise manner, ER diagrams facilitate collaboration between stakeholders and ensure that the database meets the requirements of its users.


VEHICLE_NUM
VEHICLE NAME
CH_NUMBER
PASSWORD
ADMIN
MANAGES
VEHICLE DETAIL
ADMIN_ID
VEHICLE TYPE
EN_NAME
MANAGES
PH_NU
CUS_I
EMAIL
CUS_ID
VEHICLE ID
CUS_NAME
CUSTOMER
VEHICLE REPORT
DA
HAS
RENT WITH
VEHICLE
VEHICLE RENTAL
VEHICLE_NUM
CH_NUMBER

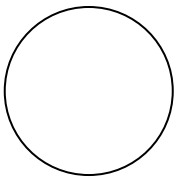
Data Flow Diagram: The data flow diagram shows the flow of data within any system. It is an important tool for designing phase of software engineering. Larry Constantine first developed it. It represents graphical view of flow of data. It's also known as BUBBLE CHART. The purpose of DFD is major transformation that will become in system design symbols used in DFD.
In the DFD, four symbols are used and they are as follows.

- A square defines a source (originator) or destination of system data.

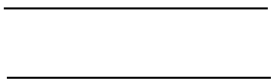
- An arrow identifies data flow-data in motion. It is 2a pipeline through which information flows.



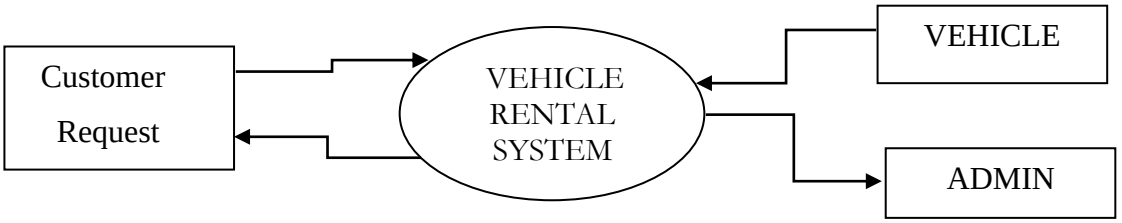
- A circle or a "bubble "(Some people use an oval bubble) represents a process that transfers informing data flows into outgoing data flows.



- An open rectangle is a data store-data at rest, or a temporary repository of data.

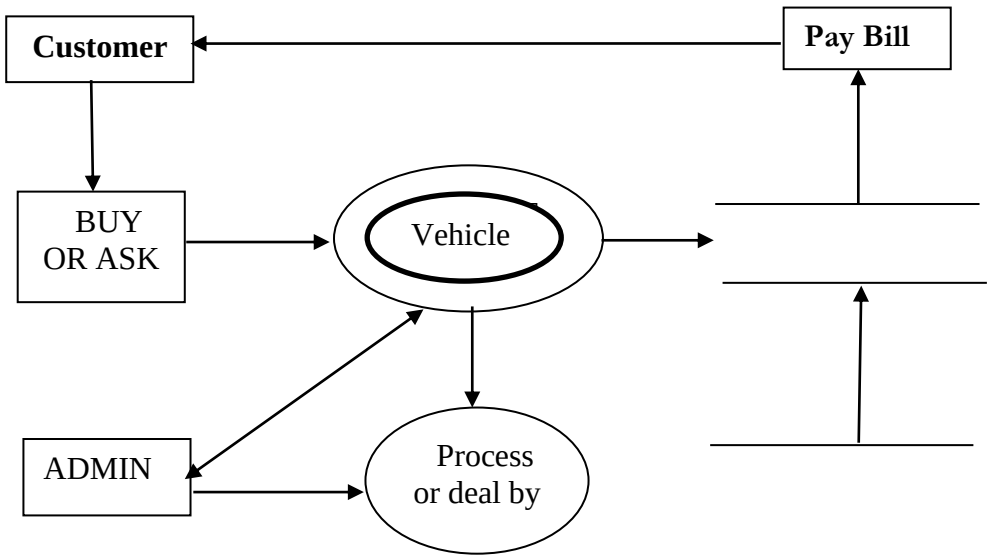


Context Level Data Flow Diagram: This level shows the overall context of the system and its operating environment and shows the whole system as just one process. Online book store is shown as one process in the context diagram; which is also known as zero level DFD, shown below.



Zero Level Data Flow Diagram: The context diagram plays important role in understanding the system and determining the boundaries. The main process can be broken into sub-processes and system can be studied with more detail; this is where 1st level DFD comes into play.

First Level DFD: This level (level 1) shows all processes at the first level of numbering, data stores, external entities and the data flows between them. The purpose of this level is to show the major high-level processes of the system and their interrelation. A process model will have one, and only one, level-1 diagram. A level-1 diagram must be balanced with its parent context level diagram, i.e. there must be the same external entities and the same data flows, these can be broken down to more detail in the level 1.



Data Tables:

a) ***Login Table***

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
user	varchar	20		☑	☑	☐	☐	☐	
password	varchar	20		☐	☐	☐	☐	☐	
				☐	☐	☐	☐	☐	

b) ***Add Vehicle Table***

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
Model	varchar	40		☐	☐	☐	☐	☐	
Company	varchar	20		☐	☐	☐	☐	☐	
Car_number	varchar	20		☑	☑	☐	☐	☐	
Make_Year	year	4		☐	☐	☐	☐	☐	
Use	varchar	30		☐	☐	☐	☐	☐	
Seating_capacity	varchar	30		☐	☐	☐	☐	☐	
Type	varchar	20		☐	☐	☐	☐	☐	
Varient	varchar	30		☐	☐	☐	☐	☐	
Engine_capacity	varchar	20		☐	☐	☐	☐	☐	
Rent	varchar	30		☐	☐	☐	☐	☐	
Avaliability	tinyint	1		☐	☐	☐	☐	☐	
				☐	☐	☐	☐	☐	

c) ***Add Customer Table***

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
Customer_ID	varchar	20		☑	☑	☐	☐	☐	
Name	varchar	30		☐	☐	☐	☐	☐	
Gender	varchar	10		☐	☐	☐	☐	☐	
Address	varchar	50		☐	☐	☐	☐	☐	
Use_As	varchar	40		☐	☐	☐	☐	☐	
Book_Date	varchar	20		☐	☐	☐	☐	☐	
Rent_Date	varchar	30		☐	☐	☐	☐	☐	
Mobile	decimal	10,0		☐	☐	☐	☐	☐	
Company	varchar	40		☐	☐	☐	☐	☐	
Seating_Capacity	varchar	20		☐	☐	☐	☐	☐	
Car_number	varchar	40		☐	☐	☐	☐	☐	
Driver_ID	varchar	20		☑	☑	☐	☐	☐	
Rent_Days	varchar	20		☐	☐	☐	☐	☐	
Total_rent	varchar	40		☐	☐	☐	☐	☐	
				☐	☐	☐	☐	☐	

d) ***Add Company Table***

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
Company_name	varchar	40		☑	☑	☐	☐	☐	
				☐	☐	☐	☐	☐	

e) Add Driver Table

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
Name	varchar	40		☐	☐	☐	☐	☐	
Gender	varchar	10		☐	☐	☐	☐	☐	
Address	varchar	60		☐	☐	☐	☐	☐	
Experience	varchar	50		☐	☐	☐	☐	☐	
DOB	varchar	30		☐	☐	☐	☐	☐	
Mobile	decimal	10,0		☐	☐	☐	☐	☐	
Driver_ID	varchar	30		☐	☐	☐	☐	☐	
Seating_Capacity	varchar	40		☐	☐	☐	☐	☐	
Company	varchar	50		☐	☐	☐	☐	☐	
Car_number	varchar	60		☐	☐	☐	☐	☐	
Identification_Number	varchar	15		☐	☐	☐	☐	☐	
				☐	☐	☐	☐	☐	

f) Add Employee Table

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
Name	varchar	40		☐	☐	☐	☐	☐	
Gender	varchar	6		☐	☐	☐	☐	☐	
Address	varchar	50		☐	☐	☐	☐	☐	
DOB	varchar	20		☐	☐	☐	☐	☐	
Qualification	varchar	20		☐	☐	☐	☐	☐	
Mobile	decimal	10,0		☐	☐	☐	☐	☐	
user	varchar	40		☐	☐	☐	☐	☐	
password	varchar	40		☐	☐	☐	☐	☐	
Date_of_Joining	varchar	20		☐	☐	☐	☐	☐	
				☐	☐	☐	☐	☐	

g) Return Car Table

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
Customer_ID	varchar	50		☑	☑	☐	☐	☐	
Name	varchar	50		☐	☐	☐	☐	☐	
Car_number	varchar	50		☑	☑	☐	☐	☐	
Model	varchar	40		☐	☐	☐	☐	☐	
Company	varchar	50		☐	☐	☐	☐	☐	
Booking_Date	varchar	50		☐	☐	☐	☐	☐	
Return_Date	varchar	50		☐	☐	☐	☐	☐	
				☐	☐	☐	☐	☐	

4. CODING

4.1 Database Connectivity:

Coding:

```
package vehicle_renting_system;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
public class DB {
   public Connection con;
   public PreparedStatement pstmt;
   public Statement stmt;
   public ResultSet rst;
   public DB()
   {
      try
      {
      Class.forName("com.mysql.jdbc.Driver");

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/renting","root","singh@323")
;
      }
      catch(Exception e)
      {
         e.printStackTrace();
      }
   }
}
```

4.2 Animation:

Coding:

```
package vehicle_renting_system;
public class Vehicle_Renting_System
```

```
{
  public static void main(String[] args)
  {
    splash sp=new splash();
    sp.setVisible(true);
    login l=new login();
    try
    {
       for(int z=0;z<=100;z++)
       {
          Thread.sleep(30);
          sp.jProgressBar1.setValue(z);
          if(z==100)
          {
             sp.setVisible(false);
             l.setVisible(true);
          }
       }
    }
    catch(Exception e)
    {
       e.printStactTrace();
    }
  }
}
```

4.3 Login:

Coding On Login as Employee Button:

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{
      Driver.setVisible(true);
}
```

Coding On Admin Button Coding:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{
```

```
        Employee.setVisible(true);}
```

Coding On Admin Login Button:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
        try
        {
           String user;
           user = t1.getText();
           String password;
           password = p1.getText();
           DB dbc=new DB();
           dbc.pstmt=dbc.con.prepareStatement("select * from admin_login  where user=? And
password=?");
           dbc.pstmt.setString(1,user);
           dbc.pstmt.setString(2,password);
           dbc.rst=dbc.pstmt.executeQuery();
           if(dbc.rst.next())
           {
              new login().setVisible(false);
              Driver.setVisible(false);
              new Driver().setVisible(true);
           }
           else
           {
              JOptionPane.showMessageDialog(null,"Incorrect Username Or Password");
           }
        }
        catch(Exception e)
        {
           e.printStackTrace();
        }
    }
```

4.4 Add Driver:

Event Performed on Menu Item:

Action Event performed(actionPerformed)

Coding:

```
private void jMenuItem8ActionPerformed(java.awt.event.ActionEvent evt)
{
      Add_Driver.setVisible(true);
      try
      {
         DB db=new DB();
         db.pstmt=db.con.prepareStatement("select Company_name from add_company");
         db.rst=db.pstmt.executeQuery();
         while(db.rst.next())
         {
           jComboBox14.addItem(db.rst.getString(1));
         }
      }
      catch(Exception e)
      {
         e.printStackTrace();
      }
   }
```

4.5 Modify Driver:

Event Performed on Menu Item:

Action Event performed(actionPerformed)

Coding: -

```
private void jMenuItem9ActionPerformed(java.awt.event.ActionEvent evt)
{
      Modify_Driver.setVisible(true);
}
```

Coding On Search Button: -

```
private void jButton15ActionPerformed(java.awt.event.ActionEvent evt)
{
```

```
        String user2=jTextField24.getText();
        try {
        DB dbc=new DB();
        dbc.pstmt=dbc.con.prepareStatement("select * from add_Driver where
Driver_ID='"+user2+"'");
        dbc.rst=dbc.pstmt.executeQuery();
        if(dbc.rst.next())
        {
           jTextField25.setText(dbc.rst.getString(1));
           String s=dbc.rst.getString(2);
           if(s.equals("Male"))
           {
           jRadioButton7.setSelected(true);
           jRadioButton8.setSelected(false);
           }
           if(s.equals("Female"))
           {
           jRadioButton8.setSelected(true);
           jRadioButton7.setSelected(false);
           }
           jTextArea4.setText(dbc.rst.getString(3));
           String a=dbc.rst.getString(4);
            for(int h=0;h<jComboBox16.getItemCount();h++)
           {
              if(a.equals(jComboBox16.getItemAt(h)))
              {
                 jComboBox16.setSelectedIndex(h);
              }
           }
           jTextField26.setText(dbc.rst.getString(5));
           String q=dbc.rst.getString(6);
           for(int h=0;h<jComboBox17.getItemCount();h++)
           {
              if(q.equals(jComboBox17.getItemAt(h)))
```

```
            {
               jComboBox17.setSelectedIndex(h);
            }
          }
         jTextField27.setText(dbc.rst.getString(7));

       }
       } catch (Exception e) {
       e.printStackTrace();
       }
   }
```

Coding On Delete Button:

```
private void jButton17ActionPerformed(java.awt.event.ActionEvent evt) {
     try
     {
      String user2=jTextField24.getText();
      DB db=new DB();
      db.pstmt=db.con.prepareStatement("delete from add_Driver where
Driver_ID='"+user2+"' ");
      int i=db.pstmt.executeUpdate();
       if(i>0)
      {
            JOptionPane.showMessageDialog(this, "Driver Record Deleted Successfully");
            jTextField24.setText(null);
            jTextField25.setText(null);
            jTextArea4.setText(null);
            jComboBox17.setSelectedItem("Select");
            jTextField26.setText(null);
            jTextField27.setText(null);
            jComboBox16.setSelectedItem("Select");
            jRadioButton7.setSelected(false);
            jRadioButton8.setSelected(false);

      }
```

```
        else
        {
              JOptionPane.showMessageDialog(this, "Error!!!");
        }
    }
     catch (Exception e)
     {
      e.printStackTrace();
    }
    }
```

4.6 List of Customer:

Event Performed On Menu Item:

Action Event performed(actionPerformed)

Coding:

```
private void jMenuItem12ActionPerformed(java.awt.event.ActionEvent evt)
{
       Customer_List.setVisible(true);
       try
       {
          Vector<Vector<String>>data=new Vector<Vector<String>>();
          data.removeAllElements();
          Vector<String>header=new Vector<String>();
          header.removeAllElements();
          header.add("Customer ID");
          header.add("Name");
          header.add("Gender");
          header.add("Address");
          header.add("Use As");
          header.add("Book Date");
          header.add("Renting Date");
          header.add("Mobile");
          header.add("Company");
          header.add("Seating Capacity");
          header.add("Car Number");
```

```
        header.add("Driver ID");
        header.add("Renting Days");
        header.add("Total Renting Amount");
        DB db=new DB();
        db.stmt=db.con.createStatement();
        db.rst=db.stmt.executeQuery("select * from add_customer");
        while(db.rst.next())
        {
           Vector<String>data1=new Vector<String>();
           data1.add(db.rst.getString(1));
           data1.add(db.rst.getString(2));
           data1.add(db.rst.getString(3));
           data1.add(db.rst.getString(4));
           data1.add(db.rst.getString(5));
           data1.add(db.rst.getString(6));
           data1.add(db.rst.getString(7));
           data1.add(db.rst.getString(8));
           data1.add(db.rst.getString(9));
           data1.add(db.rst.getString(10));
           data1.add(db.rst.getString(11));
           data1.add(db.rst.getString(12));
           data1.add(db.rst.getString(13));
           data1.add(db.rst.getString(14));
           data.add(data1);
           jTable6.setModel(new javax.swing.table.DefaultTableModel(data,header));
        }
      }
      catch(Exception e)
      {          e.printStackTrace();}}
```

4.7 Add Customer:

Event Performed On Menu Item:

Action Event performed(actionPerformed)

Coding:

```
private void jMenuItem10ActionPerformed(java.awt.event.ActionEvent evt)
```

```
{
      Add_Customer.setVisible(true);
      try
      {
         DB db=new DB();
         db.pstmt=db.con.prepareStatement("select Company_name from add_company");
         db.rst=db.pstmt.executeQuery();
         while(db.rst.next())
         {
           jComboBox19.addItem(db.rst.getString(1));
         }
      }
      catch(Exception e)
      {
         e.printStackTrace();
      }
 }
```

Coding On Calculate Button:

```
 private void jButton19ActionPerformed(java.awt.event.ActionEvent evt)
{
      String days=jTextField32.getText();
      String car_number=jTextField30.getText();
      int total;
      try{
                String rent = null;
                int temp;
                DB dbc=new DB();
      dbc.pstmt=dbc.con.prepareStatement("select rent from add_vehicle where
Car_number='"+car_number+"'");
      dbc.rst=dbc.pstmt.executeQuery();
      if(dbc.rst.next())
      {
         rent=dbc.rst.getString(1);
      }
```

```
        temp=Integer.parseInt(rent);
        total=Integer.parseInt(days)*temp;
        jTextField33.setText(Integer.toString(total));
           }
              catch(Exception e)
              {
               e.printStackTrace();
              }
   }
```

Coding On Cancel Button:

```
private void jButton21ActionPerformed(java.awt.event.ActionEvent evt)
{
        Add_Customer.setVisible(false);
}
```

4.8 Modify Customer:

Event Performed on Menu Item:

Action Event performed(actionPerformed)

Coding:

```
 private void jMenuItem11ActionPerformed(java.awt.event.ActionEvent evt)
{
        Modify_Customer.setVisible(true);
}
```

Coding On Search by ID Button:

```
private void jButton25ActionPerformed(java.awt.event.ActionEvent evt)
{
      String user2=jTextField40.getText();
        try
        {
        DB dbc=new DB();
        dbc.pstmt=dbc.con.prepareStatement("select * from add_customer where
Customer_ID='"+user2+"'");
        dbc.rst=dbc.pstmt.executeQuery();
        if(dbc.rst.next())
        {
```

```
            jTextField39.setText(dbc.rst.getString(2));
            String s=dbc.rst.getString(3);
            if(s.equals("Male"))
            {
            jRadioButton11.setSelected(true);
            jRadioButton12.setSelected(false);
            }
            if(s.equals("Female"))
            {
            jRadioButton12.setSelected(true);
            jRadioButton11.setSelected(false);
            }
            jTextArea6.setText(dbc.rst.getString(4));
            jTextField37.setText(dbc.rst.getString(8));
             }
        }
        catch (Exception e)
        {
        e.printStackTrace();
        }
    }
```

Coding on Delete Button:

```
private void jButton24ActionPerformed(java.awt.event.ActionEvent evt) {
      try
      {
       String user2=jTextField40.getText();
       DB db=new DB();
       db.pstmt=db.con.prepareStatement("delete from add_Customer where
Customer_ID='"+user2+"' ");
       int i=db.pstmt.executeUpdate();
       if(i>0)
       {
              JOptionPane.showMessageDialog(this, "Driver Record Deleted Successfully");
               jTextField40.setText(null);
```

```
            jTextField39.setText(null);
            jTextArea6.setText(null);
            jTextField37.setText(null);
            jRadioButton11.setSelected(false);
            jRadioButton12.setSelected(false);
        }
        else
        {
            JOptionPane.showMessageDialog(this, "Error!!!");
        }
    }
     catch (Exception e)
     {
      e.printStackTrace();
    }
    }
```

4.9 Return Car:

Event Performed on Menu Item:

Action Event performed(actionPerformed)

Coding:

```
private void jMenuItem14ActionPerformed(java.awt.event.ActionEvent evt)
{
      Return_Car.setVisible(true);
}
```

Coding On Search by ID Button:

```
private void jButton26ActionPerformed(java.awt.event.ActionEvent evt)
{
      String user2=jTextField38.getText();
      try
      {
         DB dbc=new DB();
         dbc.pstmt=dbc.con.prepareStatement("select * from add_customer where
Customer_ID='"+user2+"'");
```

```
            dbc.rst=dbc.pstmt.executeQuery();
            if(dbc.rst.next())
            {
               jTextField41.setText(dbc.rst.getString(2));
               jTextField42.setText(dbc.rst.getString(11));
               jTextField44.setText(dbc.rst.getString(9));
               jTextField47.setText(dbc.rst.getString(6));
               jTextField45.setText(dbc.rst.getString(7));
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        String car_number=jTextField42.getText();
        try
        {
            DB dbc=new DB();
            dbc.pstmt=dbc.con.prepareStatement("select Model from add_vehicle where
Car_number='"+car_number+"'");
            dbc.rst=dbc.pstmt.executeQuery();
            if(dbc.rst.next())
            {
                jTextField43.setText(dbc.rst.getString(1));
             }
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
```

Coding on Returned Button:

```
  private void jButton28ActionPerformed(java.awt.event.ActionEvent evt) {
        String customer_id=jTextField38.getText();
```

```
String name=jTextField41.getText();
String car_number=jTextField42.getText();
String model=jTextField43.getText();
String company=jTextField44.getText();
String Booking_Date=jTextField47.getText();
String Return_Date=jTextField45.getText();
try
{
     DB db=new DB();
    db.pstmt=db.con.prepareStatement("insert into return_car values(?,?,?,?,?,?,?)");
    db.pstmt.setString(1, customer_id);
    db.pstmt.setString(2, name);
    db.pstmt.setString(3, car_number);
    db.pstmt.setString(4,model);
    db.pstmt.setString(5, company);
    db.pstmt.setString(6, Booking_Date);
    db.pstmt.setString(7, Return_Date);
    int i=db.pstmt.executeUpdate();
    if(i>0)
    {
       JOptionPane.showMessageDialog(this, "Vehicle Returned Successfully");
       jTextField38.setText(null);
       jTextField41.setText(null);
       jTextField42.setText(null);
       jTextField43.setText(null);
       jTextField44.setText(null);
       jTextField45.setText(null);
       jTextField47.setText(null);
    }
   else
    {
      JOptionPane.showMessageDialog(this, "ERROR!!!!!!!!!");
    }
}
```

```
        catch(Exception e)
        {
           e.printStackTrace();
        }
            try
        {
          DB db=new DB();
          db.pstmt=db.con.prepareStatement("Update Avaliability from add_vehicle where
Car_number='"+car_number+"' ");
          db.pstmt.setString(1,"1");
          db.pstmt.executeUpdate();
        }
        catch(Exception e)
        {
         e.printStackTrace();
        }
    }
```

Coding On Cancel Button:

```
private void jButton29ActionPerformed(java.awt.event.ActionEvent evt)
{
      Return_Car.setVisible(false);
}
```

4.10 View Cars:

Event Performed on Menu Item:

Action Event performed(actionPerformed)

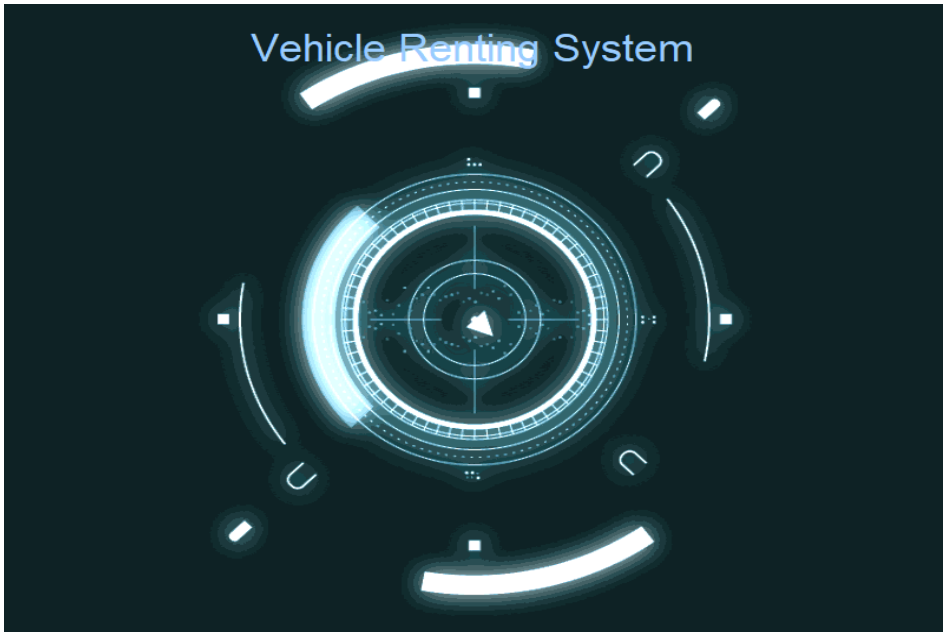
Coding:

```
private void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt)
{
      View_Cars.setVisible(true);
      try
      {
         DB db=new DB();
         db.pstmt=db.con.prepareStatement("select Company_name from add_company");
         db.rst=db.pstmt.executeQuery();
```

```
        while(db.rst.next())
        {
            jComboBox9.addItem(db.rst.getString(1));
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

5. DESIGN SCREENSHOTS

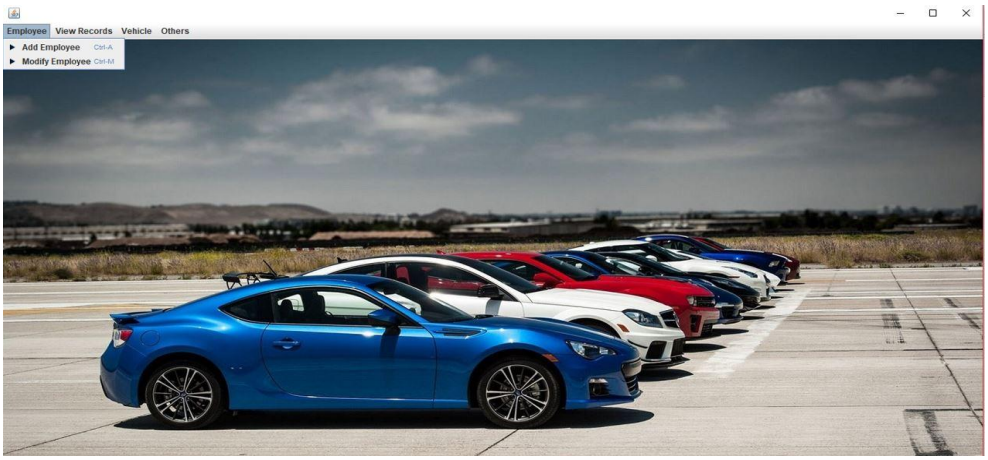
Splash Screen:

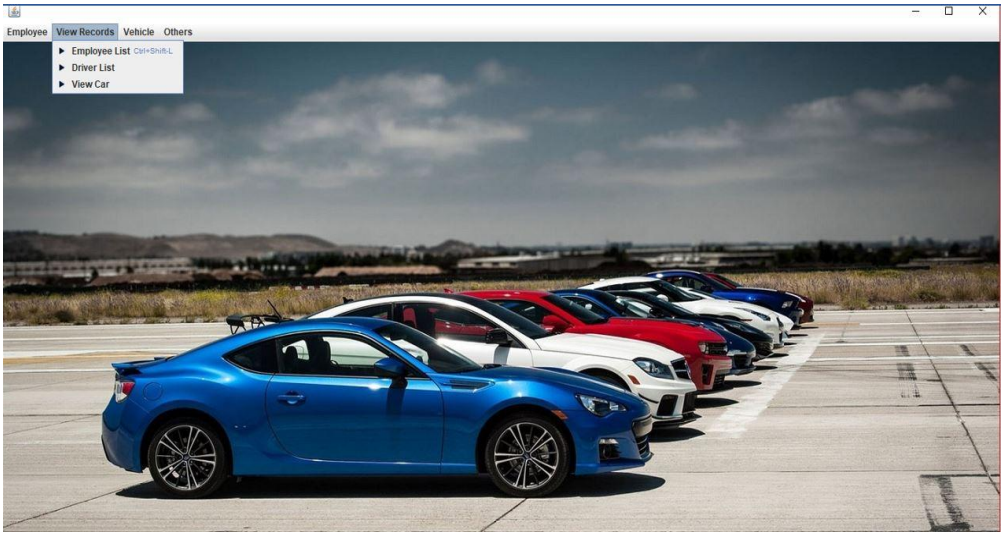


Login Window:

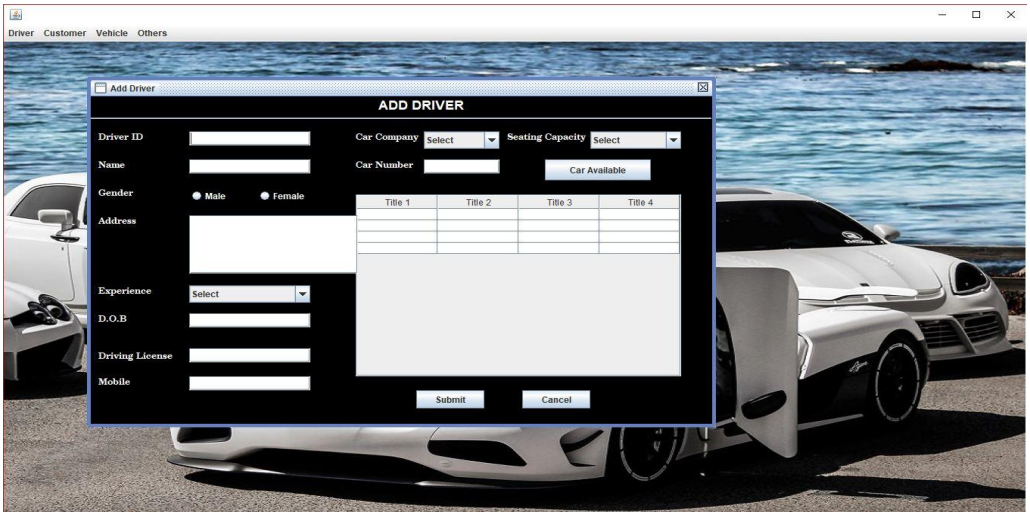
Employee Window:

Admin Window:

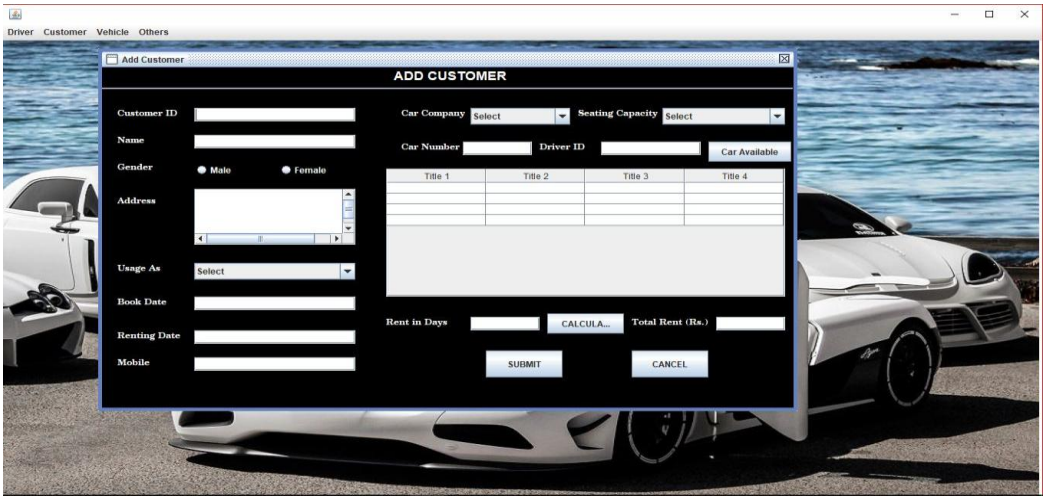




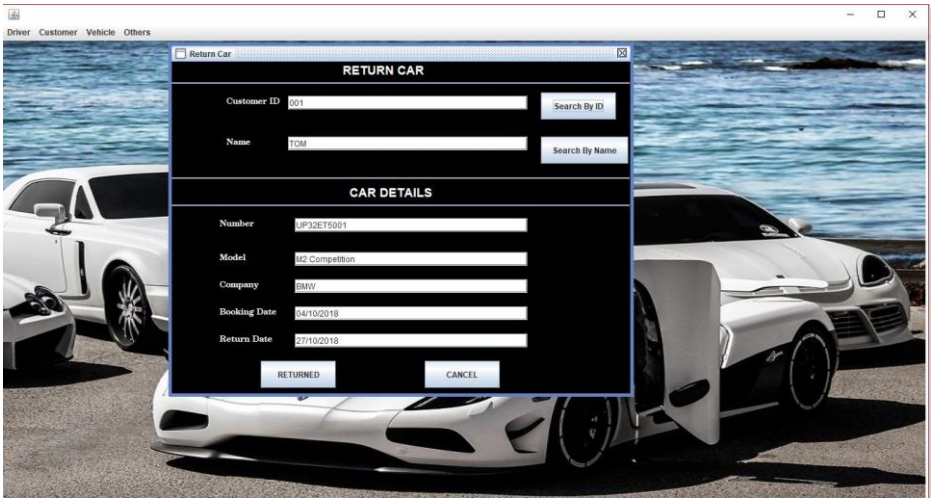
Add Driver:



Add Customer:



Return Car:



6. TESTING

6.1 System Testing:

Black box testing method was used for system testing. The black box testing usually demonstrates that software functions are operational; that the input is properly accepted and the output is correctly produced; and that integrity of external information (databases) is maintained.

Why testing is done?

- Testing is the process of running a system with the intention of finding errors.
- Testing enhances the integrity of a system by detecting deviations in design and errors in the system.
- Testing aims at detecting error-prone areas. This helps in the prevention of errors in a system.
- Testing also add value to the product by confirming to the user requirements.

Causes of Errors

The most common causes of errors in a software system are:

a) ***Communication gap between the developer and the business decisionmaker: A*** communication gap between the developer and the business decision maker is normally due to subtle differences between them. The differences can be classified into five broad areas: Thought process, Background and Experience, Interest, Priorities, Language

b) ***Time provided to a developer to complete the project:*** A common source of errors in projects comes from time constraints in delivering a product. To keep to the schedule, features can be cut. To keep the features, the schedule can be slipped. Failing to adjust the feature set or schedule when problems are discovered can lead to rushed work and flawed systems.

c) ***Over Commitment by the developer:*** High enthusiasm can lead to over commitment by the developer. In these situations, developers are usually unable to adhere to deadlines or quality due to lack of resources or required skills on the team.

d) ***Insufficient testing and quality control:*** Insufficient testing is also a major source of breakdown of e-commerce systems during operations, as testing must be done during all phases of development.

e) ***Inadequate requirements gathering:*** A short time to market results in developers starting work on the Web site development without truly understanding the business and technical requirements. Also, developers may create client-side scripts using language that may not work on some client browsers.

f) ***Keeping pace with the fast-changing technology:*** New technologies are constantly introduced. There may not be adequate time to develop expertise in the new technologies. This is a problem for two reasons. First, the technology may not be properly implemented. Second, the technology may not integrate well with the existing environment.

6.2 Testing Principles:

- To discover as yet undiscovered errors.
- All tests should be traceable to customer's requirement.
- Tests should be planned long before the testing actually begins.
- Testing should begin "in the small" & progress towards "testing in the large".
- Exhaustive Testing is not possible.
- To be most effective training should be conducted by an Independent Third Party

6.3 Testing Objectives:

- Testing is a process of executing a program with the intent of finding errors.
- A good test case is one that has a high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.

6.4 Kinds of Testing:

- ***Black Box Testing-*** **Not** based on any knowledge of internal designs or code. Tests are based on requirements and functionality. This method treats the software as a "black box," focusing solely on the inputs provided to the system and the outputs it produces. The primary objective is to ensure that the software behaves as expected according to the specified requirements and performs all its intended functions correctly. In black box testing, testers create test cases based on software requirements and specifications. They provide various inputs to the system and verify if the outputs are as anticipated. This type of testing is crucial for validating the overall functionality and user interface of the software. It is also effective in identifying discrepancies between the actual behavior and the expected behavior of the software, such as incorrect or missing functions, interface errors, and performance issues.
- ***White Box Testing-*** Based on the knowledge of the internal logic of an application's code. Tests are based on coverage of code statements, branches, paths and statements.

This approach focuses on verifying the internal logic, control flow, and data flow within the application, ensuring that each part of the code performs as intended. In white box testing, testers create test cases based on the internal structure of the software. They examine the code to identify paths, branches, conditions, loops, and statements that need to be tested. The goal is to ensure that all code paths are executed at least once, and potential errors are detected early in the development process. White box testing provides a thorough examination of the code, enabling the detection of hidden errors, optimization of code, and improvement of overall software quality. By understanding the internal logic, testers can create more effective and targeted test cases, leading to a robust and reliable application.

- ***Unit Testing-*** The most 'micro' scale of testing; to test particular functions and code modules. Typically done by the programmer and not by the testers, as it requires detailed knowledge of the internal program design and code. The goal is to catch bugs early in the development process, making it easier to fix them. Unit tests are usually automated, allowing them to be run frequently and efficiently. Tools commonly used for unit testing include JUnit for Java, NUnit for .NET, and PyTest for Python.
- ***Integration Testing-*** Testing of combined parts of an application to determine if they function together correctly. The 'parts' can be code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/ server and distributed systems. There are different approaches to integration testing, such as top-down, bottom-up, and sandwich (hybrid) testing. This phase is critical for detecting problems that may arise when combining modules, even if each module works correctly in isolation. Tools like JUnit, TestNG, and Postman (for API testing) are frequently used in integration testing.
- ***Functional Testing-*** Black-box type testing geared to functional requirements of an application; testers should do this type of testing. This doesn't mean that the programmers shouldn't check that their code works before releasing it. The purpose is to ensure that the software behaves as expected and that all features function correctly. This type of testing primarily focuses on what the system does, rather than how it does it, and involves testing the user interface, APIs, databases, security, client/server applications, and functionality of the software. Functional testing is driven by the business requirements and specifications. Testers create test cases based on these requirements to ensure that each function of the software application operates in conformance with the requirement specification. The testing process mimics the actions of an end-user. It

checks if the software application behaves as expected under various conditions, ensuring a good user experience. This testing verifies that the software performs its intended functions correctly. It includes validation of input data, processing, and output results.

- ***Regression Testing-*** Re-testing after fixes or modifications of the software or its environment. It is difficult to determine how much re testing is needed, especially near the end of the development cycle. Automated testing tools can be especially useful for this type of testing. It involves re-running previously executed test cases to verify that the software continues to perform as expected after modifications such as bug fixes, enhancements, or new features. Regression testing can be conducted at various levels, including unit, integration, and system testing, and can range from retesting a selective subset of affected test cases to comprehensive re-execution of all test cases, depending on the scope of changes. The process typically includes identifying relevant test cases, setting up a test environment that closely mimics the production environment, executing test cases, logging and tracking defects, and maintaining test cases to keep them updated with the latest changes. Automation plays a significant role in regression testing, as it helps efficiently manage repetitive and extensive testing processes, thereby ensuring that the software remains stable and reliable over time.
- ***Acceptance Testing-*** Final testing based on the specifications of the end user or customer or based on use by end-users/ customers over some limited period of time. This testing phase involves real-world scenarios and end-users to validate that the software functions as intended and meets their needs. It typically encompasses User Acceptance Testing (UAT), where end-users verify the system's usability and functionality, and Operational Acceptance Testing (OAT), which focuses on the operational aspects such as backup, recovery, and maintenance procedures. The primary goal of acceptance testing is to ensure that the software is fit for purpose, providing a final check before the software is released into the production environment. This phase helps identify any issues from the end-user perspective, ensuring the software delivers a satisfactory user experience and meets all specified requirements.
- ***User Acceptance Testing-*** Determining if software is satisfactory to an end user customer. During UAT, real users interact with the software to validate its usability, functionality, and overall user experience in real-world scenarios. This testing phase allows stakeholders to assess whether the software meets the agreed-upon requirements and whether it aligns with the organization's business goals. Any discrepancies or issues identified during UAT are documented and addressed before the software is released to

ensure a high level of quality and user satisfaction. UAT provides valuable feedback to developers and project managers, helping to refine the software and ensure it meets the needs of its intended users effectively.

6.5 Testing Technique Used:

We will continuously test our project to ensure that it is fully functional. In order to perform testing test cases are designed with the intent of finding the errors in the project and help in removing those errors. Testing begins at the module level and is conducted systematically. It is generally conducted by independent test groups or third party.

Testing is done in our project Vehicle Renting System with help of black box testing that exercise all the functional requirement of the project test cases are designed using this approach by providing set of input conditions to get the expected output.

7. CONCLUSION

The Vehicle Renting System project represents a comprehensive solution designed to revolutionize the way vehicle rental services are managed and accessed. Throughout our presentation, we have highlighted the key features and functionalities of our system, including modules such as Admin, Customer, and Invoice Management. The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

- A description of the background and context of the project and its relation to work already done in the area.
- Made statement of the aims and objectives of the project.
- The description of Purpose, Scope, and applicability.
- We define the problem on which we are working in the project.
- We describe the requirement Specifications of the system and the actions that can be done on these things.
- We understand the problem domain and produce a model of the system, which describes operations that can be performed on the system.
- We included features and operations in detail, including screen layouts.
- We designed user interface and security issues related to system.

Finally, the system is implemented and tested according to test cases.

8. FUTURE SCOPE

This project traverses a lot of areas ranging from business concept to computing field and required to perform several researches to be able to achieve the project objectives.

The area covers include:

- ***Car rental industry:*** This includes study on how the car rental business is being done, process involved and opportunity that exist for improvement.
- ***Java Technology:*** It is used for the development of the application.
- ***Eco-friendly:*** The monitoring of the vehicle activity and the overall business becomes easy and includes the least of paper work.

The software acts as an office that is open 24/7. It increases the efficiency of the management at offering quality services to the customers. It provides custom features development and support with the software.

9. BIBLIOGRAPHY

References:

The following books were referred during the analysis and execution phase of the project

- **Common Language Runtime**

  By Steven Pratschner

- **Software Engineering**

  By Roger S. Pressman

- **Unified Modeling Language**

  By Gradi Booch, Ivar Jacobson, James Rambaugh

- **Complete Reference .Net**

  By David S Platt

- **MSDN 2003**

  By Microsoft

- **Images**

  Google Search

  Msn Search

  Bing Search

- **Html Publishing Bibl**

  By Alan Simpson.

- **C# 2008**

  Andrew Troelson

Websites:

- www.google.com
- www.bing.com
- www.msn.com
- www.wikipedia.com
- www.imagegallery.com
- www.sql.com'
- www.learnjava.com