

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**  
JNANASANGAMA, BELAGAVI - 590018



**DBMS LABORATORY WITH MINI-PROJECT REPORT**

**ON**

**“CAR RENTAL MANAGEMENT SYSTEM”**

*Submitted in partial fulfillment for the award of degree of*

**Bachelor of Engineering  
in  
COMPUTER SCIENCE AND ENGINEERING**

Submitted by

**HARSHA M S 1BG20CS043**

Under the Guidance of

**Smt. Jayashree**

**Assistant Professor**

**Department of CSE, BNMIT**



*Vidyaya Amrutham Ashnatho*

**B.N.M. Institute of Technology**

**An Autonomous Institute under VTU**

Approved by AICTE, Accredited as grade A Institution by NAAC. All eligible branches – CSE, ECE, EEE, ISE & Mech. Engg. are Accredited by NBA for academic years 2018-19 to 2024-25 & valid upto 30.06.2025

URL: [www.bnmit.org](http://www.bnmit.org)

**Department of Computer Science and Engineering**

**2022– 2023**

# *B.N.M. Institute of Technology*

**An Autonomous Institution under VTU**

Approved by AICTE, Accredited as grade A Institution by NAAC. All eligible branches – CSE, ECE, EEE, ISE & Mech. Engg. are Accredited by NBA for academic years 2018-2019 to 2024-25 & valid upto 30.06.2025

URL: [www.bnmit.org](http://www.bnmit.org)

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



## **CERTIFICATE**

Certified that the project work entitled **“Car rental management system”** carried out by **Mr. Harsha M S (1BG20CS043)**, are bonafide students of V Semester, BNM Institute of Technology in partial fulfillment for the award of Bachelor of Engineering in COMPUTER SCIENCE AND ENGINEERING of Visvesvaraya Technological University, Belagavi during the year 2022-23. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

**Smt. Jayashree**  
**Assistant Professor**  
**Department of CSE**  
**BNMIT, Bengaluru**

**Dr. Chayadevi M L**  
**Professor and HOD**  
**BNMIT, Bengaluru**

**Dr. Krishnamurthy G N**  
**Principal**  
**BNMIT, Bengaluru**

**Examiner 1:**

**Examiner 2:**

## **ABSTRACT**

The paper developed an automated system that is used to manage car information and its administration. This was with a view to eliminate the problem of inappropriate data keeping, inaccurate reports, time wastage in storing, processing and reserving information encountered by the traditional car rental system in order to improve the overall efficiency of the organization. The design provides excellent car rental management service and improved information structure.

# ACKNOWLEDGEMENT

The completion of this project brings with a sense of satisfaction, but it is never complete without thanking the persons responsible for its successful completion.

I take this opportunity to express our profound gratitude to **Shri. Narayan Rao R Maanay, Secretary, BNMIT, Bengaluru** for his constant support and encouragement.

I would like to express my special thanks to **Prof. T. J. Rama Murthy, director, BNMIT, Bengaluru** and **S Y Kulkarni, director, BNMIT, Bengaluru** for their constant guidance towards our goals and professions.

I extend my deep sense of sincere gratitude to **Dr. Krishnamurthy G.N, Principal, BNMIT, Bengaluru**, for providing us the facilities required for the project.

I would also like to thank **Prof Eishwar Maanay, Dean Administration, BNMIT, Bengaluru**, for providing us useful suggestions required for the project.

I express my in-depth, heartfelt, sincere gratitude to **Dr. Chayadevi M L, Professor and H.O.D, Department of Computer Science and Engineering, BNMIT, Bengaluru**, for her valuable suggestions and support.

I extend my heartfelt, sincere gratitude to **Prof. Jayashree, Assistant Professor, Department of Computer Science and Engineering, BNMIT, Bengaluru**, for completion of the project.

Finally, I would like to thank all the faculty members of Department of Computer Science and Engineering, BNMIT, Bengaluru, for their support. I would like to thank our family and friends for their unfailing moral support and encouragement.

**Harsha M S-1BG20CS043**

# TABLE OF CONTENTS

<u>Contents</u>	<u>Page No.</u>
<b>Abstract</b>	
<b>Acknowledgement</b>	
<b>1. Introduction</b>	1
<b>2. Literature Review</b>	2
2.1 Database Management System	2
2.2 Structured Query Language	3
2.3 MySQL	3
2.3.1 MySQL Workbench	4
2.4 Entity Relationship Diagram	4
2.5 Relational Schema	7
2.6 Normalization	7
2.7 Uses of DBMS	8
2.8 Application of Database	8
<b>3. Requirement Specification</b>	9
3.1 Hardware Requirements	9
3.2 Software Requirements	9
<b>4. System Design</b>	10
4.1 E-R Diagram	11
4.2 Relational Schema	12
<b>5. Coding</b>	16
5.1 Table Creation and Insertion	16
5.2 Queries	21
5.3 Front-end Code and Snapshots	25
<b>6. Conclusion</b>	30
<b>7. Future Enhancements</b>	31
<b>8. References</b>	

## CHAPTER 1

### INTRODUCTION

A Car Rental Management System is looking to develop a state-of-the-art car rental portfolio management system which is able to track their car rental booking history and billing and car details. This system facilitates the owner of the Car Rental Company to retrieve, update and track and delete the bookings and cars efficiently. At the same time, can utilize this system to monitor their financial management. Currently, different locations of the car rental company have their own separated systems leading to lack of communication and inefficient data sharing. For example, the car rental company located in Mumbai uses simple Microsoft Excel to keep the record of their customers, cars and insurance of their cars which is inconvenient to retrieve and update the cars and customer's information. In the car rental company located at Dharwad, maintain a book-based ledger to keep a record of their customers and cars and the insurance for the same. The main branch of the company located in Bengaluru has to keep the customer and car details of all their branch offices on their own computer system. While each statement serves a distinctive process, there is no coordination, assimilation and representation of data. The systems may have duplicate data which leads to waste of space. The different systems also may have different application programs which cause incompatible files.

Due to these disadvantages of the current system, a car rental management system is proposed. Car Rental Management System is a database management system (DBMS) which is based on computer networks, using the advance database technology to construct, maintain and update various kinds of data in data base system. The DBMS can track and update all the information of the cars available and customers during a particular time span. The major advantages of the DBMS are easy to retrieve and update information, efficient data sharing and communication and reliable backup and security. Information about cars is done by just writing the car name, car model and insurance details. Whenever a new car is added to the company its information is stored freshly. Bills are generated by recording the usage of the car by the customer. Details of the customer are updated on a written sheet and at last, they all summed up. Car usage details are recorded on the document, which contains the customer information. It is destroyed after sometime time period to decrease the paper load in the office. The admin themselves have to track the car details and customer details which is a tedious job.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 DATABASE MANAGEMENT SYSTEM

Data can be defined as a representation of facts, concepts, or instructions in a formalized manner, which should be suitable for communication, interpretation, or processing by human or electronic machine.

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system. Let us discuss few examples. An online telephone directory uses a database to store data of people, phone numbers, and other contact details. Your electricity service provider uses a database to manage billing, client-related issues, handle fault data, etc. Let us also consider Facebook. It needs to store, manipulate, and present data related to members, their friends, member activities, messages, advertisements, and a lot more. We can provide a countless number of examples for the usage of databases.

Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data. DBMS allows users to create their own databases as per their requirement. The term “DBMS” includes the user of the database and other application programs. It provides an interface between the data and the software application. There are 4 major types of DBMS. Let's look into them in detail.

- **HIERARCHICAL** -In a Hierarchical database, model data is organized in a tree-like structure. Data is Stored Hierarchically (top down or bottom up) format. Data is represented using a parent- child relationship. In Hierarchical DBMS parent may have many children, but children have only one parent.
- **NETWORK DBMS** -The network database model allows each child to have multiple parents. It helps you to address the need to model more complex relationships like as the orders/parts many- to-many relationship. In this model, entities are organized in a graph which can be accessed through several paths.
- **RELATIONAL MODEL**-Relational DBMS is the most widely used DBMS model because it is one of the easiest. This model is based on normalizing data in the rows and columns of the tables. Relational model stored in fixed structures and manipulated using SQL.
- **OBJECT-ORIENTED MODEL**-In Object-oriented Model data stored in the form of objects. The structure which is called classes which display data within it. It defines a

database as a collection of objects, which stores both data members values and operations.

## 2.2 SQL – STRUCTURED QUERY LANGUAGE

SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system.

## 2.2 MySQL

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open- source and free software under the GNU license. It is supported by Oracle Company.

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications. It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in C programming language and C++ programming language. Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

MySQL is a Relational Database Management System (RDBMS) software that provides many things, which are as follows:

- It allows us to implement database operations on tables, rows, columns, and indexes.
- It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.
- It provides the Referential Integrity between rows or columns of various tables.
- It allows us to updates the table indexes automatically.



- It uses many SQL queries and combines useful information from multiple tables for the end-users.

### 2.3 MySQL Workbench

MySQL Workbench is a unified visual database designing or graphical user interface tool used for working with database architects, developers, and Database Administrators. It is developed and maintained by Oracle. It provides SQL development, data modeling, data migration, and comprehensive administration tools for server configuration, user administration, backup, and many more. We can use this Server Administration for creating new physical data models, E-R diagrams, and for SQL development (run queries, etc.). It is available for all major operating systems like Mac OS, Windows, and Linux.

With MySQL Workbench, you use an intuitive, browser-based interface, to:

- Administer the database
- Create tables, views, and other database objects
- Import, export, and view table data
- Run queries and SQL scripts
- Generate reports.

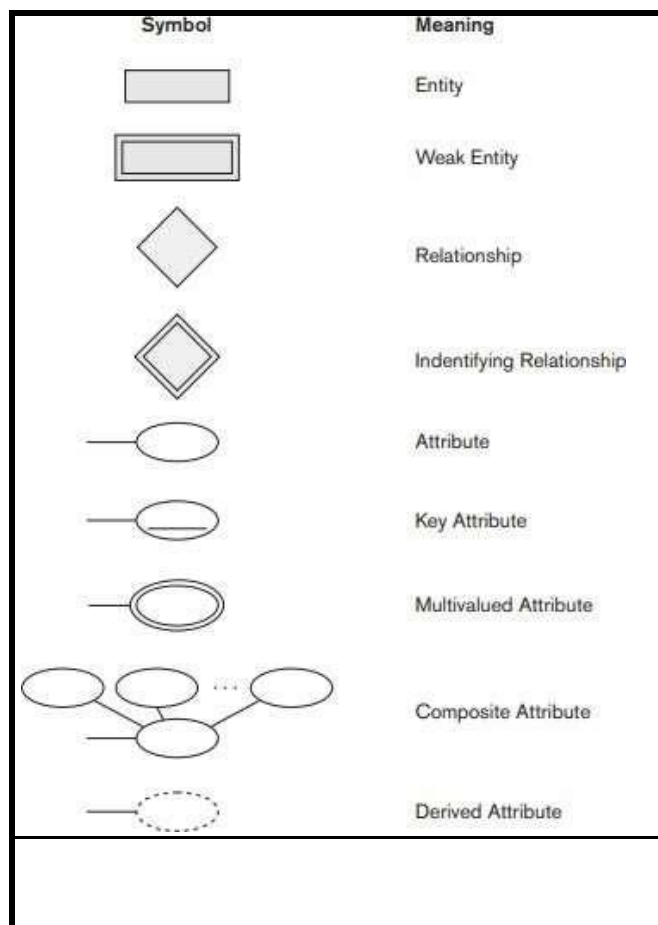
### 2.4 ENTITY RELATIONSHIP DIAGRAM

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

- The properties that are used to describe an entity are known as Attributes; for example, an Employee entity may have a Name, Gender, Date of Birth of his/her attributes.
- If book is regarded as an entity then Author's name, Price, published by, etc. are its various attributes.

A specific entity will have a value for each of its attributes. Thus, an entity has a value for each attribute. A diagram representing entities and relationships among them is known as entity relationship diagram. The major elements used in ER diagram are entities, attributes, identifiers and relationships that express a reality for which database is designed.

Fig. 2.4.1 - ER Diagram



### ENTITY TYPE:

It symbolizes anything in the real world that has multiple existence.

- **WEAK ENTITY TYPE:** The weak entity in DBMS do not have a primary key and are dependent on the parent entity. It mainly depends on other entities. □
- **RELATIONSHIP TYPE:** A diamond box is used to represent the relationship between two entities. Relationships can be one-to-one, one-to-many or many-to-many. □
- **IDENTIFYING RELATIONSHIP TYPE:** The relationship type that is used to relate a weak entity type to its owner is shown by double lined diamond shaped box.

### ATTRIBUTE:

Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes. There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

- **KEY ATTRIBUTE:** A key attribute is one for which each entity has a unique value. It is represented by an oval shape with the attribute name underlined.
- **MULTI VALUED ATTRIBUTE:** An entity that has multiple values for that attribute is called multivalued attribute.
- **DERIEVED ATTRIBUTE:** As discussed earlier, an attribute whose value depends upon the value of the stored attribute. It is represented using a dashed oval shape.

In a database system, we deal with various types of keys as follows:

- **CANDIDATE KEY:** Minimal set of attributes that uniquely identifies each occurrence of an entity type.
- **PRIMARY KEY:** Candidate key selected to uniquely identify each occurrence of an entity type. **UNIQUE KEY:** Can accept unique of null values.
- **COMPOSITE KEY:** A key that consists of two or more attributes and removal of even one of them would result in loss of intended information.

## 2.5 RELATIONAL SCHEMA

The relational schema is the primary element of the relational database. These databases are managed using language and structure that is consistent with first-order logic. This allows for database management based on entity relationships, making them easy to organize according to volume. Relational schema refers to the meta-data that describes the structure of data within a certain domain. It is the blueprint of a database that outlines the way its structure organizes data into tables. There are two steps to creating a relational database schema: creating the logical schema and creating the physical schema. The logical schema depicts the structure of the database, showing the tables, columns and relationships with other tables in the database and can be created with modeling tools or spreadsheet and drawing software. The physical schema is created by actually generating the tables, columns and relationships in the relational database management software (RDBMS).

## 2.6 NORMALIZATION

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Here are the most commonly used normal forms.

- First Normal Form
- Second Normal Form
- Third Normal Form
- Boyce & Codd Normal Form.

### First Normal Form (1NF)

For a table to be in the First Normal Form, it should follow the following 4 rules:

- It should only have single(atomic) valued attributes/columns.
- Values stored in a column should be of the same domain
- All the columns in a table should have unique names.
- The order in which data is stored, does not matter.

### Second Normal Form (2NF)

For a table to be in the Second Normal Form, it should follow the following 4 rules:

- It should be in the First Normal form.
- It should not have Partial Dependency.

### Third Normal Form (3NF)

A table is said to be in the Third Normal Form when,

- It is in the Second Normal form.
- It doesn't have Transitive Dependency.

### Boyce and Codd Normal Form (BCNF)

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

- R must be in 3rd Normal Form
- For each functional dependency ( $X \rightarrow Y$ ), X should be a super key.

## 2.7 USES OF DBMS

Data that is well organized and integrated is very useful in decision making. We can infer some of the following uses of DBMS.

- Effective and efficient management of data.
- Query processing and management.
- Easy to understand and user friendly.
- Security and integrity of data.
- Better decision making.
- Data sharing and storage.
- Better access to accurate data.
- Ensures error free information.

## 2.8 APPLICATIONS OF DATABASE

Databases are used to support internal operations of organizations and to underpin online interactions with customers and suppliers' databases are used to hold administrative information and more specialized data, such as engineering data or economic models.

Databases touch all aspects of our lives. Some of the major areas of application are as follows:

- Airlines
- Universities
- E- Commerce
- Human Resources

## CHAPTER 3

### REQUIREMENT SPECIFICATION

The hardware and software components of a computer system that are required to install and use software efficiently are specified in the SRS. The minimum system requirements need to be met for the programs to run at all times on the system.

#### 3.1 HARDWARE REQUIREMENTS

The hardware requirements specify the necessary hardware which provides us the platform to implement our programs.

- 2.2 GHz processor (Pentium).
- GB RAM (System Memory).
- 20 GB of hard-drive space.
- VGA capable of 1024 x 768 screen resolution.
- Necessary computer peripherals such as keyboard etc.

#### 3.2 SOFTWARE REQUIREMENTS

The software requirement specifies the pre-installed software needed to run the code being implemented in this project.

- Windows Operating System
- MySQL Workbench
- MySQL Shell
- MySQL Server
- Connector/J
- IDE – Apache NetBeans

## CHAPTER 4

### DESCRIPTION

A Car Rental Management System is looking to develop a state-of-the-art car rental portfolio management system which is able to track their car rental booking history and billing and car details. This system facilitates the owner of the Car Rental Company to retrieve, update and track and delete the bookings and cars efficiently. At the same time, can utilize this system to monitor their financial management.

Currently, different locations of the car rental company have their own separated systems leading to lack of communication and inefficient data sharing. For example, the car rental company located in Basveshwarnagar uses simple Microsoft Excel to keep the record of their customers, cars and insurance of their cars which is inconvenient to retrieve and update the cars and customer's information. In the car rental company located at Whitefield, maintain a book-based ledger to keep a record of their customers and cars and the insurance for the same. The main branch of the company located in Rajajinagar has to keep the customer and car details of all their branch offices on their own computer system. While each statement serves a distinctive process, there is no coordination, assimilation and representation of data. The systems may have duplicate data which leads to waste of space. The different systems also may have different application programs which cause incompatible files.

Due to these disadvantages of the current system, a car rental management system is proposed. Car Rental Management System is a database management system (DBMS) which is based on computer networks, using the advance database technology to construct, maintain and update various kinds of data in data base system. The DBMS can track and update all the information of the cars available and customers during a particular time span. The major advantages of the DBMS are easy to retrieve and update information, efficient data sharing and communication and reliable backup and security.

## 4.1 E-R DIAGRAM

Entity	Attributes
Login	Password , Username
Car_reg	<u>Id</u> , car_reg , make , model , available
Customer	<u>Id</u> , cust_id , name , address , mobile
Rental	<u>Id</u> , car_id , cust_id, fee , date , due
Returncar	<u>Id</u> , car_id , cust_id , return_date , elp , fine

Table 4.1 - E-R Diagram

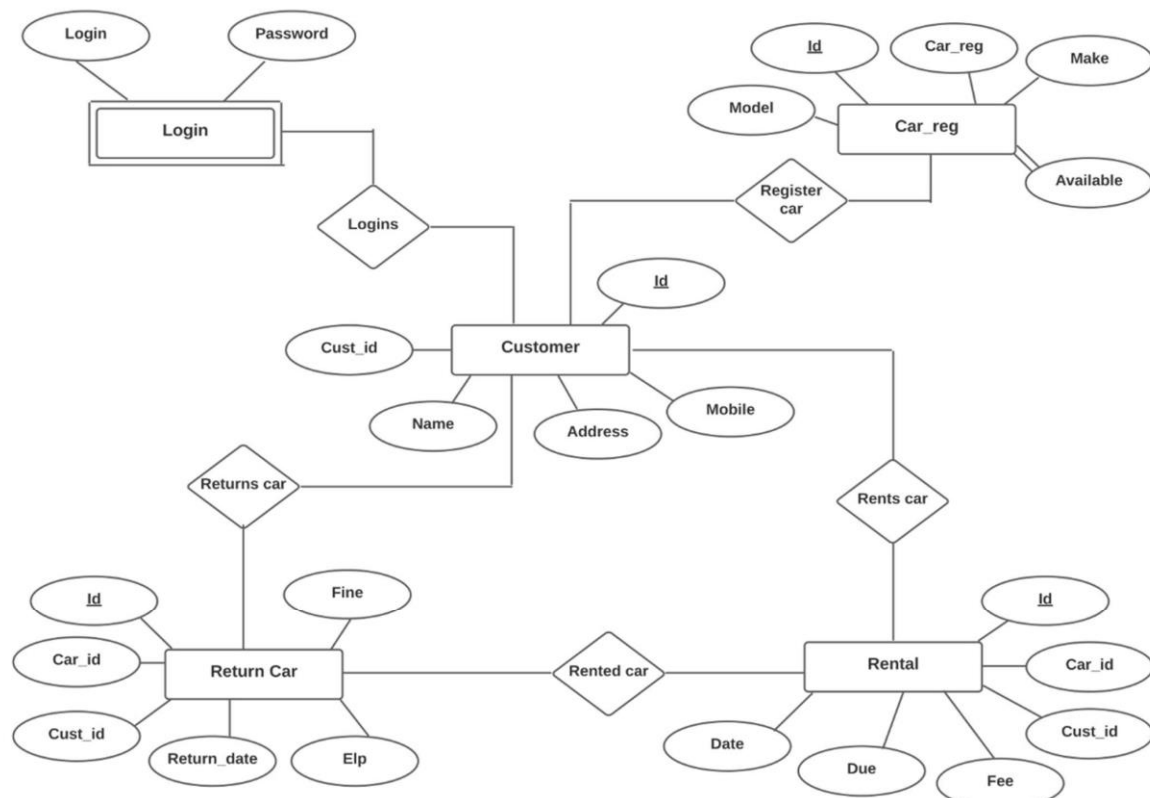


Fig. 4.1 - E-R diagram for “CAR RENTAL MANAGEMENT SYSTEM”



## 4.2 RELATIONAL SCHEMA

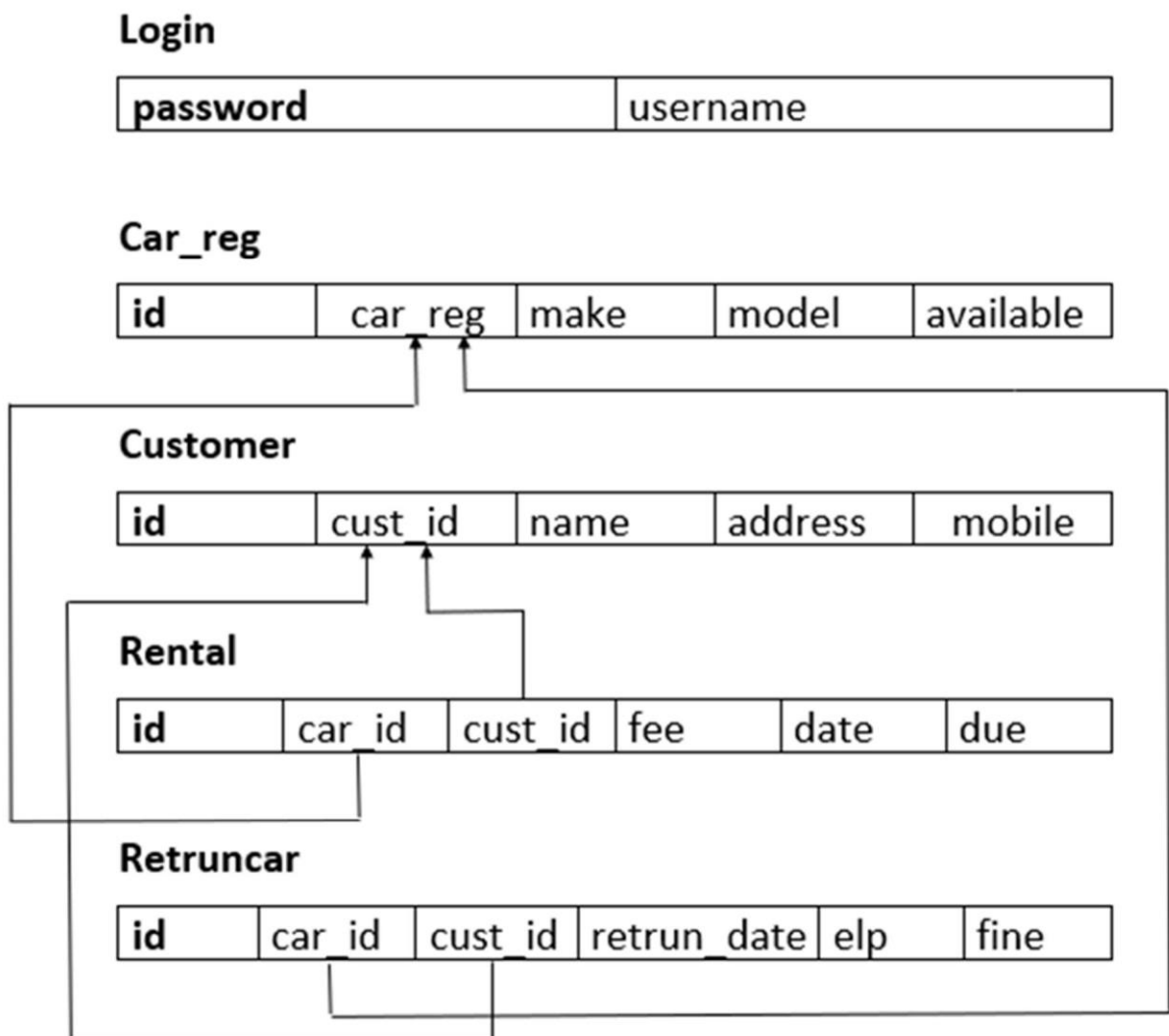


Fig. 4.2.1 - Relation Schema for “Car rental Management System”

### LOGIN TABLE:

#### LOGIN

Password	Username
----------	----------

#### Attributes:

- Password: Password is of type int, and it is used in login page to enter the main page.
- Username: Is used to enter the login page is of type varchar.

### CAR REGISTRATION TABLE:

#### Car\_reg

<u>Id</u>	Car_reg	Make	Model	Available
-----------	---------	------	-------	-----------

#### Attributes:

- Id: This is used to uniquely identify a row in the table. It is of the type int.
- Car\_reg: Every car has a car registration number which is used to identify that individual car uniquely. It is of the type varchar.
- Make: This gives the information of the manufacturer. It is of the type varchar.
- Model: Every car is different from one another because the model is different. It is of the type varchar.
- Available: This gives the field whether the car is available or not. It is of the type varchar

## CUSTOMER TABLE:

Customer

<u>Id</u>	Cust_id	Name	Address	Mobile
-----------	---------	------	---------	--------

### Attributes:

- Id: Every customer row is unique and it should be uniquely identified. It is of the type int.
- Cust\_id: Every customer has an id. It is of the type varchar.
- Name: Every customer has name to be identified differently. It is of the type varchar.
- Address: Every customer has an address on where they stay. It is of the type text .
- Mobile: Every customer has a phone no. through which they can be contacted. It is of the type int.

## CAR RENTAL TABLE:

Rental

<u>Id</u>	<u>Car_id</u>	<u>Cust_id</u>	Fee	Date	Due
-----------	---------------	----------------	-----	------	-----

### Attributes:

- Id: Every Rental row is unique and it should be uniquely identified. It is of the type int.
- Car\_id: If we are renting a car the id should be considered for the rented car and it is the foreign key of the table. It is of the type int.
- Cust\_id: Customer id play a major role on who has rented the car and it is also a foreign key of the table. It is of the type varchar.
- Fee: Each rented car after returning it will be charged a fee based on the number of days used . It is of the type int.
- Date: It is the staring date of renting a car. It is of the type int.
- Due:It is the last date of returning a car It is of the type int.

## RETURNING THR CAR TABLE:

Returncar

Id	<u>Car_id</u>	<u>Cust_id</u>	Return_date	Elp	Fine
----	---------------	----------------	-------------	-----	------

### Attributes:

- Id: Every Rental row is unique and it should be uniquely identified. It is of the type int.
- Car\_id: If we are renting a car the id should be considered for the rented car and it is the foreign key of the table. It is of the type int.
- Cust\_id: Customer id play a major role on who has rented he car and it is also a foreign key of the table. It is of the type varchar.
- Return\_date: □ It is the last date of returning a car It is of the type varchar
- Elp: It is the time elapsed on after the return date of the car was passed. It is of the type int.
- Fine: Is the amount of money that has been deducted as a fine. It is of the type int.

## CHAPTER 5

## CODING

## 5.1 Table Creation and Insertion

Login Table:

```
CREATE TABLE `login` (  
  `password` varchar(10) NOT NULL,  
  `username` varchar(10) NOT NULL  
);
```

Field	Type	Null	Key	Default	Extra
username	varchar(10)	YES		NULL	
password	varchar(10)	YES		NULL	

Fig 5.1 Table creation for login

Inserting:

```
INSERT INTO `login` (`username`, `password`) VALUES('harsha', 1234);  
INSERT INTO `login` (`username`, `password`) VALUES('balaji', 5678);
```

password	username
1234	harsha
5678	balaji

Fig 5.2 Insertion of tuples for login table

Car\_reg Table:

```
CREATE TABLE `car_reg` (  
  `id` int(11) NOT NULL,  
  `car_reg` varchar(255) NOT NULL,  
  `make` varchar(255) NOT NULL,  
  `model` varchar(255) NOT NULL,  
  `available` varchar(255) NOT NULL  
);
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
car_reg	varchar(255)	NO		NULL	
make	varchar(255)	NO		NULL	
model	varchar(255)	NO		NULL	
available	varchar(255)	NO		NULL	

Fig 5.3 Table creation for car\_reg

Inserting:

```
INSERT INTO `car_reg` (`id`,`car_reg`,`make`,`model`,`available`) VALUES(1,
'KA52TZ5289', 'Maruti Suzuki', 'Alto 800', 'No');
```

```
INSERT INTO `car_reg` (`id`,`car_reg`,`make`,`model`,`available`) VALUES(2,
'KA01MN7456', 'Hyundai Motors', 'Hyundai i10', 'Yes');
```

```
INSERT INTO `car_reg` (`id`,`car_reg`,`make`,`model`,`available`) VALUES(3,
'KA02QP1023', 'Hyundai Motors', 'Hyundai i20', 'Yes');
```

```
INSERT INTO `car_reg` (`id`,`car_reg`,`make`,`model`,`available`) VALUES(4,
'KA53XY1151', 'Tata',
'Tiago', 'Yes');
```

```
INSERT INTO `car_reg` (`id`,`car_reg`,`make`,`model`,`available`) VALUES(5,
'KA48PN3289', 'Tata',
'Tigor', 'No');
```

```
INSERT INTO `car_reg` (`id`,`car_reg`,`make`,`model`,`available`) VALUES(6,
'KA01AD7690', 'Toyato', 'Camry', 'Yes');
```

```
INSERT INTO `car_reg` (`id`,`car_reg`,`make`,`model`,`available`) VALUES(7,
'KA51ML8281', 'Honda', 'City', 'Yes');
```

id	car_reg	make	model	available
1	KA52TZ5289	Maruti Suzuki	Alto 800	No
2	KA01MN7456	Hyundai Motors	Hyundai i10	Yes
3	KA02QP1023	Hyundai Motors	Hyundai i20	Yes
4	KA53XY1151	Tata	Tiago	Yes
5	KA48PN3289	Tata	Tigor	No
6	KA01AD7690	Toyato	Camry	Yes
7	KA51ML8281	Honda	City	Yes

Fig 5.4 Insertion of tuples for car\_reg table

Customer Table:

```
CREATE TABLE `customer` (  
  `id` int(11) NOT NULL,  
  `cust_id` varchar(255) NOT NULL,  
  `name` varchar(255) NOT NULL,  
  `address` text NOT NULL,  
  `mobile` int(11) NOT NULL  
);
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
cust_id	varchar(255)	NO		NULL	
name	varchar(255)	NO		NULL	
address	text	NO		NULL	
mobile	int	NO		NULL	

Fig 5.5 Table creation for customer

Inserting:

```
INSERT INTO `customer` (`id`, `cust_id`, `name`, `address`, `mobile`) VALUES(1, 'C0001',  
'Anurag', 'Bellary Karnataka', 94523436);  
INSERT INTO `customer` (`id`, `cust_id`, `name`, `address`, `mobile`) VALUES(2, 'C0002',  
'Tejaswi', 'Raichur Karnataka', 86636811);  
INSERT INTO `customer` (`id`, `cust_id`, `name`, `address`, `mobile`) VALUES(3, 'C0003',  
'Thoyaj', 'Mysore Karnataka', 91326742);  
INSERT INTO `customer` (`id`, `cust_id`, `name`, `address`, `mobile`) VALUES(4, 'C0004',  
'Vallabh', 'Manglore Karantaka', 65457601);  
INSERT INTO `customer` (`id`, `cust_id`, `name`, `address`, `mobile`) VALUES(5, 'C0005',  
'Likith', 'Banglore Rural Karantaka', 65973898);  
INSERT INTO `customer` (`id`, `cust_id`, `name`, `address`, `mobile`) VALUES(6, 'C0006',  
'Varun', 'Banglore Karnataka', 64251931);
```

id	cust_id	name	address	mobile
1	C0001	Anurag	Bellary Karnataka	94523436
2	C0002	Tejaswi	Raichur Karnataka	86636811
3	C0003	Thoyaj	Mysore Karnataka	91326742
4	C0004	Vallabh	Manglore Karantaka	65457601
5	C0005	Likith	Banglore Rural Karantaka	65973898
6	C0006	Varun	Banglore Karnataka	64251931

Fig 5.6 Insertion of tuples for customer table

Rental Table:

```
CREATE TABLE `rental` (  
  `id` int(11) NOT NULL,  
  `car_id` varchar(255) NOT NULL,  
  `cust_id` varchar(255) NOT NULL,  
  `fee` int(11) NOT NULL,  
  `date` varchar(255) NOT NULL,  
  `due` varchar(255) NOT NULL  
);
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
car_id	varchar(255)	NO		NULL	
cust_id	varchar(255)	NO		NULL	
fee	int	NO		NULL	
date	varchar(255)	NO		NULL	
due	varchar(255)	NO		NULL	

Fig 5.7 Table creation for rental table

Inserting:

```
INSERT INTO `rental` (`id`,`car_id`,`cust_id`,`fee`,`date`,`due`) VALUES(1, 'KA01MN7456',  
'C0002', 5000, '2019-02-16', '2019-02-20');
```

```
INSERT INTO `rental` (`id`,`car_id`,`cust_id`,`fee`,`date`,`due`) VALUES(2, 'KA02QP1023',  
'C0001', 4000, '2019-02-17', '2019-02-22');
```

```
INSERT INTO `rental` (`id`,`car_id`,`cust_id`,`fee`,`date`,`due`) VALUES(3, 'KA53XY1151',  
'C0004', 5000, '2019-02-17', '2019-02-18');
```

```
INSERT INTO `rental` (`id`,`car_id`,`cust_id`,`fee`,`date`,`due`) VALUES(4, 'KA01AD7690',  
'C0001', 3000, '2019-10-14', '2019-10-15');
```

```
INSERT INTO `rental` (`id`,`car_id`,`cust_id`,`fee`,`date`,`due`) VALUES(5, 'KA01MN7456',  
'C0006', 1000, '2020-06-17', '2020-06-20');
```

id	car_id	cust_id	fee	date	due
1	KA01MN7456	C0002	5000	2019-02-16	2019-02-20
2	KA02QP1023	C0001	4000	2019-02-17	2019-02-22
3	KA53XY1151	C0004	5000	2019-02-17	2019-02-18
4	KA01AD7690	C0001	3000	2019-10-14	2019-10-15
5	KA01MN7456	C0006	1000	2020-06-17	2020-06-20

Fig 5.8 Insertion of tuples for rental table



Returncar Table:

```
CREATE TABLE `returncar` (  
  `id` int(11) NOT NULL,  
  `car_id` varchar(255) NOT NULL,  
  `cust_id` varchar(255) NOT NULL,  
  `return_date` varchar(255) NOT NULL,  
  `elp` int(11) NOT NULL,  
  `fine` int(11) NOT NULL  
);
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
car_id	varchar(255)	NO		NULL	
cust_id	varchar(255)	NO		NULL	
return_date	varchar(255)	NO		NULL	
elp	int	NO		NULL	
fine	int	NO		NULL	

Fig 5.7 Table creation for return car table

Inserting:

```
INSERT INTO `returncar` (`id`, `car_id`, `cust_id`, `return_date`, `elp`, `fine`) VALUES(1,  
'KA01MN7456', 'C0001', '2019-02-04', 12, 1200);  
INSERT INTO `returncar` (`id`, `car_id`, `cust_id`, `return_date`, `elp`, `fine`) VALUES(2,  
'KA02QP1023', 'C0002', '2019-02-21', 0, 0);  
INSERT INTO `returncar` (`id`, `car_id`, `cust_id`, `return_date`, `elp`, `fine`) VALUES(3,  
'KA53XY1151', 'C0005', '2019-02-19', 0, 0);  
INSERT INTO `returncar` (`id`, `car_id`, `cust_id`, `return_date`, `elp`, `fine`) VALUES(4,  
'KA01AD7690', 'C0001', '2019-02-06', 11, 1100);  
INSERT INTO `returncar` (`id`, `car_id`, `cust_id`, `return_date`, `elp`, `fine`) VALUES(5,  
'KA01MN7456', 'C0006', '2019-10-15', 0, 0);
```

id	car_id	cust_id	return_date	elp	fine
1	KA01MN7456	C0001	2019-02-04	12	1200
2	KA02QP1023	C0002	2019-02-21	0	0
3	KA53XY1151	C0005	2019-02-19	0	0
4	KA01AD7690	C0001	2019-02-06	11	1100
5	KA01MN7456	C0006	2019-10-15	0	0

Fig 5.8 Insertion of tuples for return car table

## 5.2 QUERIES

The most common operation in SQL, the query, makes use of the declarative. SELECT statement. SELECT retrieves data from one or more tables, or expressions. Standard SELECT statements have no persistent effects on the database. Some non-standard implementations of SELECT can have persistent effects, such as the SELECT INTO Syntax provided in some databases.

Queries allow the user to describe desired data, leaving the database management system (DBMS) to carry out planning, optimizing, and performing the physical operations necessary to produce that result, normally immediately following the SELECT keyword. An asterisk (“\*”) can be used to specify that the query should return all columns of the queried tables. Select is the most complex statement in SQL, with optional keywords and clauses that include:

- The FROM clause, which indicates the table(s) to retrieve data from. The FROM clause can include optional JOIN subclauses to specify the rules for joining tables.
- The WHERE clause includes a comparison predicate, which restricts the rows returned by the query. The WHERE clause eliminates all rows from the result set where the comparison predicate does not evaluate to True.
- The GROUP BY clause projects rows having common values into a smaller set of rows. GROUP BY is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set. The WHERE clause is applied before the GROUP BY clause.
- The HAVING clause includes a predicate used to filter rows resulting from the GROUP BY clause. Because it acts on the results of the GROUP BY clause, aggregation functions can be used in the HAVING clause predicate.
- The ORDER BY clause identifies which column[s] to use to sort the resulting data, and in which direction to sort them (ascending or descending). Without an ORDER BY clause, the order of rows returned by an SQL query is undefined.
- The DISTINCT keyword eliminates duplicate data.

## QUERY 1

- i) Display the details of the cars available from car\_reg.
- ii) Display the details of the cars available in Hyundai motors.

```
mysql> select * from car_reg;
+----+-----+-----+-----+-----+
| id | car_reg | make       | model    | available |
+----+-----+-----+-----+-----+
| 1  | KA52TZ5289 | Maruti Suzuki | Alto 800 | No        |
| 2  | KA01MN7456 | Hyundai Motors | Hyundai i10 | No        |
| 3  | KA02QP1023 | Hyundai Motors | Hyundai i20 | Yes       |
| 4  | KA53XY1151 | Tata         | Tiago    | Yes       |
| 5  | KA48PN3289 | Tata         | Tigor    | No        |
| 6  | KA01AD7690 | Toyato      | Camry    | Yes       |
| 7  | KA51ML8281 | Honda       | City     | Yes       |
+----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

mysql> select * from car_reg where available='Yes';
+----+-----+-----+-----+-----+
| id | car_reg | make       | model    | available |
+----+-----+-----+-----+-----+
| 3  | KA02QP1023 | Hyundai Motors | Hyundai i20 | Yes       |
| 4  | KA53XY1151 | Tata         | Tiago    | Yes       |
| 6  | KA01AD7690 | Toyato      | Camry    | Yes       |
| 7  | KA51ML8281 | Honda       | City     | Yes       |
+----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> select * from car_reg where make='Hyundai Motors' and available='Yes';
+----+-----+-----+-----+-----+
| id | car_reg | make       | model    | available |
+----+-----+-----+-----+-----+
| 3  | KA02QP1023 | Hyundai Motors | Hyundai i20 | Yes       |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Fig 5.2.1 Displaying all tuples of car\_reg table

## QUERY 2

Update mobile number of a customer where id is 3.

```
mysql> select * from customer;
+----+-----+-----+-----+-----+
| id | cust_id | name   | address           | mobile |
+----+-----+-----+-----+-----+
| 1  | C0001  | Anurag | Bellary Karnataka | 94523436 |
| 2  | C0002  | Tejaswi | Raichur Karnataka | 86636811 |
| 3  | C0003  | Thoyaj | Mysore Karnataka  | 91326742 |
| 4  | C0004  | Vallabh | Manglore Karantaka | 65457601 |
| 5  | C0005  | Likith | Banglore Rural Karantaka | 65973898 |
| 6  | C0006  | Varun  | Banglore Karnataka | 64251931 |
+----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> update customer set mobile='78792728' where id='3';
Query OK, 1 row affected (0.13 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from customer;
+----+-----+-----+-----+-----+
| id | cust_id | name   | address           | mobile |
+----+-----+-----+-----+-----+
| 1  | C0001  | Anurag | Bellary Karnataka | 94523436 |
| 2  | C0002  | Tejaswi | Raichur Karnataka | 86636811 |
| 3  | C0003  | Thoyaj | Mysore Karnataka  | 78792728 |
| 4  | C0004  | Vallabh | Manglore Karantaka | 65457601 |
| 5  | C0005  | Likith | Banglore Rural Karantaka | 65973898 |
| 6  | C0006  | Varun  | Banglore Karnataka | 64251931 |
+----+-----+-----+-----+-----+
6 rows in set (0.03 sec)
```

Fig 5.2.1 Displaying all tuples of customer table

## QUERY 3:

Get information of customers where the customer has paid fee more than 4000.

```
mysql> select * from customer;
```

id	cust_id	name	address	mobile
1	C0001	Anurag	Bellary Karnataka	94523436
2	C0002	Tejaswi	Raichur Karnataka	86636811
3	C0003	Thoyaj	Mysore Karnataka	78792728
4	C0004	Vallabh	Manglore Karantaka	65457601
5	C0005	Likith	Banglore Rural Karantaka	65973898
6	C0006	Varun	Banglore Karnataka	64251931

6 rows in set (0.00 sec)

```
mysql> select * from rental;
```

id	car_id	cust_id	fee	date	due
1	KA01MN7456	C0002	5000	2019-02-16	2019-02-20
2	KA02QP1023	C0001	4000	2019-02-17	2019-02-22
3	KA53XY1151	C0004	5000	2019-02-17	2019-02-18
4	KA01AD7690	C0001	3000	2019-10-14	2019-10-15
5	KA01MN7456	C0006	1000	2020-06-17	2020-06-20
23	KA01MN7456	C0007	12000	2022-01-02	2022-01-11

6 rows in set (0.00 sec)

```
mysql> select name,address,mobile from customer where cust_id in (select cust_id from rental where fee>4000 );
```

name	address	mobile
Tejaswi	Raichur Karnataka	86636811
Vallabh	Manglore Karantaka	65457601

2 rows in set (0.00 sec)

Fig 5.2.1 Displaying all tuples of rental table

## QUERY 4

Retrieve information of customers where the customer has paid fine.

```
mysql> select * from customer;
```

id	cust_id	name	address	mobile
1	C0001	Anurag	Bellary Karnataka	94523436
2	C0002	Tejaswi	Raichur Karnataka	86636811
3	C0003	Thoyaj	Mysore Karnataka	78792728
4	C0004	Vallabh	Manglore Karantaka	65457601
5	C0005	Likith	Banglore Rural Karantaka	65973898
6	C0006	Varun	Banglore Karnataka	64251931

6 rows in set (0.00 sec)

```
mysql> select * from rental;
```

id	car_id	cust_id	fee	date	due
1	KA01MN7456	C0002	5000	2019-02-16	2019-02-20
2	KA02QP1023	C0001	4000	2019-02-17	2019-02-22
3	KA53XY1151	C0004	5000	2019-02-17	2019-02-18
4	KA01AD7690	C0001	3000	2019-10-14	2019-10-15
5	KA01MN7456	C0006	1000	2020-06-17	2020-06-20
23	KA01MN7456	C0007	12000	2022-01-02	2022-01-11

6 rows in set (0.00 sec)

```
mysql> select * from returncar;
```

id	car_id	cust_id	return_date	elp	fine
1	KA01MN7456	C0001	2019-02-04	12	1200
2	KA02QP1023	C0002	2019-02-21	9	900
3	KA53XY1151	C0005	2019-02-19	0	0
4	KA01AD7690	C0001	2019-02-06	11	1100
5	KA01MN7456	C0006	2019-10-15	5	500

5 rows in set (0.00 sec)

```
mysql> select * from customer where cust_id in (select cust_id from rental where car_id in (select car_id from returncar where fine!=0));
```

id	cust_id	name	address	mobile
1	C0001	Anurag	Bellary Karnataka	94523436
2	C0002	Tejaswi	Raichur Karnataka	86636811
6	C0006	Varun	Banglore Karnataka	64251931

3 rows in set (0.00 sec)

Fig 5.2.1 Displaying all tuples of rental table

### QUERY 5

Create a view of customers who have rented more than one car and retrieve the data of those customers.

```
mysql> create view rentedcar as select cust_id,count(cust_id) as count from rental group by cust_id order by count(cust_id)>1;
Query OK, 0 rows affected (0.13 sec)

mysql> select * from rentedcar;
+-----+-----+
| cust_id | count |
+-----+-----+
| C0002   | 1     |
| C0004   | 1     |
| C0006   | 1     |
| C0007   | 1     |
| C0001   | 2     |
+-----+-----+
5 rows in set (0.00 sec)

mysql> select * from customer where cust_id in (select cust_id from rentedcar where count>1);
+----+-----+-----+-----+-----+
| id | cust_id | name  | address          | mobile |
+----+-----+-----+-----+-----+
| 1  | C0001   | Anurag | Bellary Karnataka | 94523436 |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Fig 5.2.1 Displaying all tuples of returncar table



## 5.3 Front end Code and Snapshots

We have 6 java classes with their source code and Design that are under the package car that is being imported in starting of every code.

Login.java (Which uses user name and password to login into the Main java source and design file)

Main.java (Which has links to other four java classes)

Carreg.java (Where we can add cars to be registered for rental agreement)

Customer.java (Where we have info about the customers and which car they have rented)

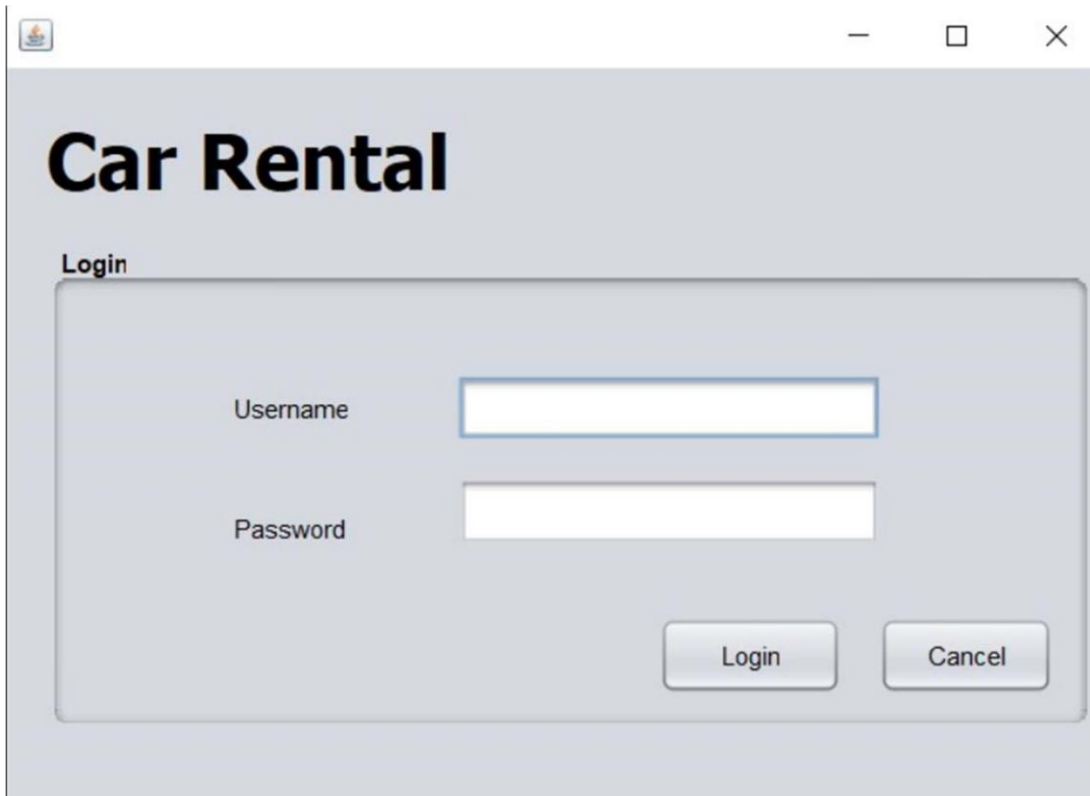
Rental.java (Where we have info about the car that are rented and for how many days)

Duepay.java (Where we have info about the car returned date and time elapsed and fine that is collected)

So that we can access every code from anywhere. In Main.java we have created a function which on clicking opens the individual design and runs the code of that particular java class.

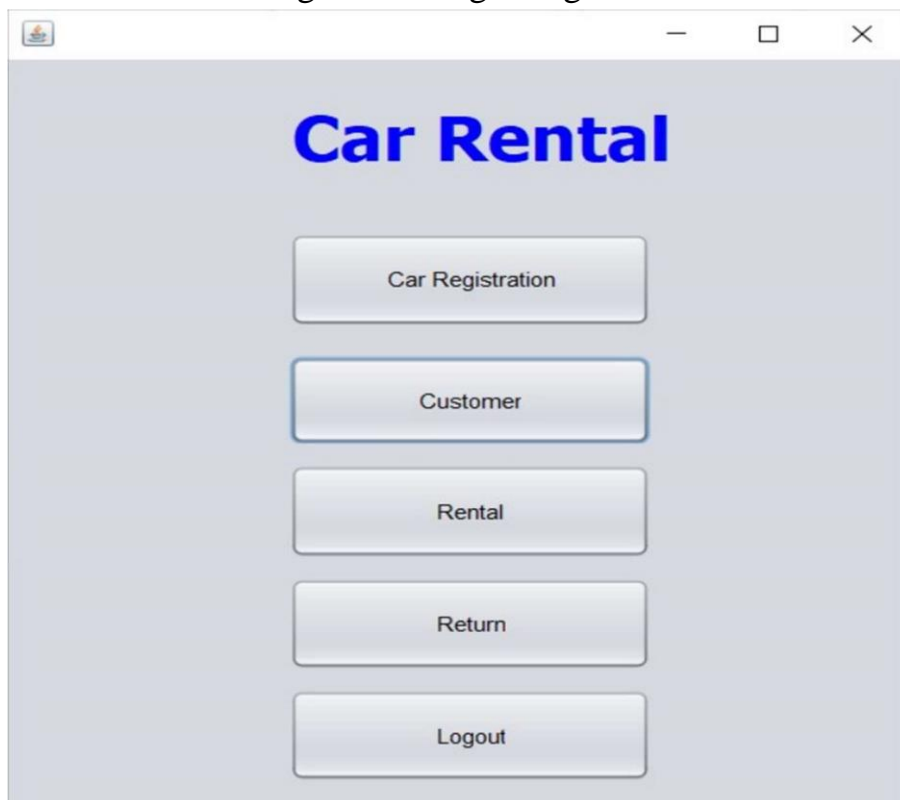
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    carreg add = new carreg();  
    add.setVisible(true);  
}  
  
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    customer add = new customer();  
    add.setVisible(true);  
}  
  
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    rental add = new rental();  
    add.setVisible(true);  
}  
  
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    Duepay add = new Duepay();  
    add.setVisible(true);  
}
```

## Front end Snapshots




A screenshot of a web application window titled "Car Rental". The window has a light gray background and standard window controls (minimize, maximize, close) in the top right corner. Below the title, the word "Login" is displayed in a small, bold font. The main content area contains two input fields: "Username" and "Password". The "Username" field is a white rectangle with a blue border, and the "Password" field is a white rectangle with a gray border. Below these fields are two buttons: "Login" and "Cancel", both with a light gray background and a thin border.

Fig. 5.3.1 Login Page



A screenshot of a web application window titled "Car Rental". The window has a light gray background and standard window controls (minimize, maximize, close) in the top right corner. The title "Car Rental" is displayed in a large, bold, blue font. Below the title, there are five buttons stacked vertically: "Car Registration", "Customer", "Rental", "Return", and "Logout". Each button has a light gray background and a thin border. The "Customer" button is highlighted with a blue border.

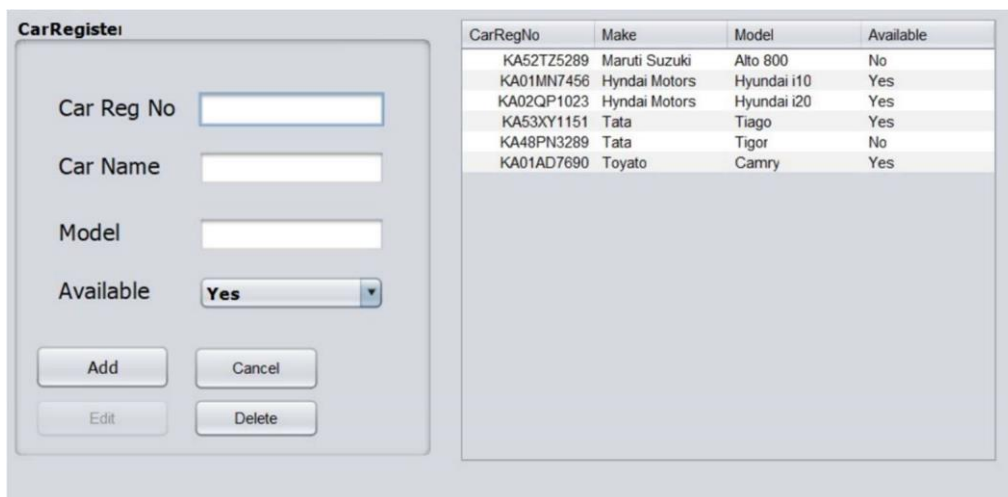
Fig. 5.3.2 Main Page



The screenshot shows the 'CarRegister' form with the following fields: Car Reg No (KA52TZ5289), Car Name (Maruti Suzuki), Model (Alto 800), and Available (No). A 'Message' dialog box is displayed in the center, stating 'Successfully Saved' with an 'OK' button. Below the form are buttons for 'Add', 'Cancel', 'Edit', and 'Delete'.

CarRegNo	Make	Model	Available
----------	------	-------	-----------

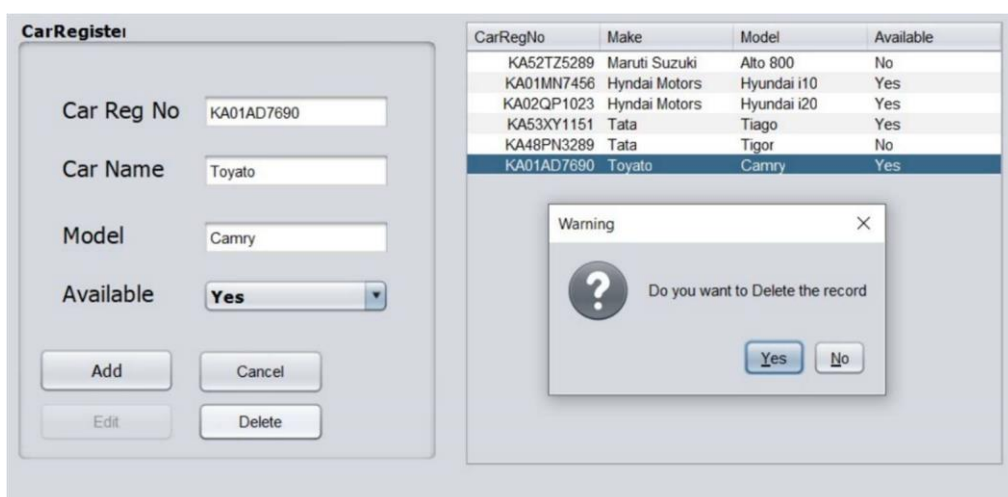
Fig. 5.3.3.1 Car Registration - Adding Information



The screenshot shows the 'CarRegister' form with empty fields for Car Reg No, Car Name, and Model, and 'Available' set to 'Yes'. A table on the right displays the registered cars:

CarRegNo	Make	Model	Available
KA52TZ5289	Maruti Suzuki	Alto 800	No
KA01MN7456	Hyundai Motors	Hyundai i10	Yes
KA02QP1023	Hyundai Motors	Hyundai i20	Yes
KA53XY1151	Tata	Tiago	Yes
KA48PN3289	Tata	Tigor	No
KA01AD7690	Toyato	Camry	Yes

Fig. 5.3.3.2 Car registration - Information added Successfully



The screenshot shows the 'CarRegister' form with fields filled with 'KA01AD7690', 'Toyato', and 'Camry', and 'Available' set to 'Yes'. A 'Warning' dialog box is displayed, asking 'Do you want to Delete the record' with 'Yes' and 'No' buttons. The table on the right shows the registered cars, with the row for 'KA01AD7690' highlighted in blue.

CarRegNo	Make	Model	Available
KA52TZ5289	Maruti Suzuki	Alto 800	No
KA01MN7456	Hyundai Motors	Hyundai i10	Yes
KA02QP1023	Hyundai Motors	Hyundai i20	Yes
KA53XY1151	Tata	Tiago	Yes
KA48PN3289	Tata	Tigor	No
KA01AD7690	Toyato	Camry	Yes

Fig. 5.3.3.3 Car registration - Information deletion of registered car



**Customer**

Customer ID:

Customer Name:

Address:

Mobile:

CustomerID	CustomerName	Address	Mobile
C0001	Anurag	Bellary Karnataka	94523436
C0002	Tejaswi	Raichur Karnataka	86636811
C0003	Thoyaj	Mysore Karnataka	78792728
C0004	Vallabh	Manglore Karantaka	65457601
C0005	Likith	Banglore Rural Kar...	65973898
C0006	Varun	Banglore Karnataka	64251931
C0007	Shreyas	RV Clg Banglore	99999999
C0008	Kruthik	8th Mile Banglore	21212121
C0009	Rakshitha	Baglugunte 8th mile	22446688
C0010	Yasaswini	Mumbai India	998877

Fig. 5.3.4 Customer – Customer Registration

**Renta**

Car ID:

Customer ID:

Customer Name:

Rental fee:

Date:

Due Date:

**Available**

Fig. 5.3.5.1 Rental – Car Available

**Renta**

Car ID:

Customer ID:

Customer Name:

Rental fee:

Date:

Due Date:

**Available**

Fig. 5.3.5.2 Rental – Car not Available

### Return Car

Car ...

Customer ID

Date

Days Elapsed

Fine

CustID	CarID	ReturnDate	Elapsed	Fine
KA01MN7456	C0001	2019-02-04	12	1200
KA02QP1023	C0002	2019-02-21	9	900
KA53XY1151	C0005	2019-02-19	0	0
KA01AD7690	C0001	2019-02-06	11	1100
KA01MN7456	C0006	2019-10-15	5	500
KA01AD7690	C007	2019-11-16	2	200
KA01AD7690	C007	2019-11-16	2	200

Fig. 5.3.6.1 Return car – Car returning procedure

### Return Car

Car ...

Customer ID

Date

Days Elapsed

Fine

CustID	CarID	ReturnDate	Elapsed	Fine
KA01MN7456	C0001	2019-02-04	12	1200
KA02QP1023	C0002	2019-02-21	9	900
KA53XY1151	C0005	2019-02-19	0	0
KA01AD7690	C0001	2019-02-06	11	1100
KA01MN7456	C0006	2019-10-15	5	500
KA01AD7690	C007	2019-11-16	2	200
KA01AD7690	C007	2019-11-16	2	200
KA53XY1151	C0001	2022-02-03	3	300

Fig. 5.3.6.2 Return car – Returning car table updated

## **CONCLUSION**

The project Car Rental Management System (CRMS) is for computerizing the working in a car rental company. The software takes care of all the requirements of an average car rental company and is capable to provide easy and effective storage of information related to cars and customers of the car rental company.

It generates bills, provides car details including car insurance and service details. It also provides customer details such as driving license number, membership id, name etc. The system also provides the facility of backup as per requirement.

## **FUTURE ENHANCEMENTS**

The system is designed in such a way that provisions can be given for further enhanced without affecting the system presently developed. The enhancements that can be incorporated are:

- Building app with more features and developing the design.
- We can work on providing end to end customer service.

## REFERENCES

1. Fundamental of Database System by Elmasri and Navathe ,5<sup>th</sup> Edition, Addison-Wesley,2007.
2. Database System Concepts by Avi Silberschatz, Henry F Korth, and S. Sudharshan,1996.
3. Concepts of Database Management by Philip J. Pratt,2008.
4. Modern Database Management by Jeffery A Hoffer,2010.
5. <https://www.w3schools.com> .
6. <https://www.youtube.com>
7. <https://www.google.co.in>
8. <https://www.wikipedia.org>