Documentation BIKE Rental final

Bachelors Of Computer Application (Sharnbasva University)

1. INTRODUCTION

1.1 Project Description

This project is designed so as to be used by Bike Rental Company specializing in renting bikes to customers. It is an online system through which customers can view available bikes, register, view profile and book bike.

1.2 Objective

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 bike rental industry is not left out. This E-Bike Rental system is developed to provide the following services

•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

2. SYSTEM ANALYSIS

2.1 Existing System & Proposed System

2.1 Existing System

Bike Rental System service will help users to book a bike for some fee specified. Till now there was no clear web based UI to help the users to rent the vehicle. They had to manually rent the vehicle through their offices. It was a difficult task to manage rental vehicles. Keeping track of all the rental bikes was a problem.

2.1 Proposed System

This Bike Rental System project will enable the user to rent a vehicle. The user shall login to the system and check for availability of bikes. The user specifies a type of bike and the journey date and time. The Bike Rental System shall check for the availability of the bike and rent the bike to the customer. The user can make payment online. The tool is designed using PHP. All the data regarding the rental bikes are stored in MySQL database. The user has to enter his name, address, phone details and check for the bikes available for rent. The UI is very simple and the connectivity to back end is robust. The main advantage is that the user shall be able to choose a bike depending on his budget.

2.2 Feasibility Study

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. The objective of a feasibility study is to find out if an information system project can be done and to suggest possible alternative solutions.

There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

2.2.1 Technical

It refers to whether the software that is available in the market fully supports the present application. It studies the pros and cons of using particular software for the development and its feasibility. It also studies the additional training needed to be given to the people to make the application work. The technical requirements are then compared to the technical capability of the organization. The systems project is considered technically feasible if the internal technical capability is sufficient to support the project requirements. The analyst must find out whether current technical resources can be upgraded or added to in a manner that fulfils the request under consideration.

2.2.2 Operational

It refers to the feasibility of the product to be operational. Some products may work very well at design and implementation but may fail in the real time environment. It includes the study of additional human resource required and their Technical expertise. It dependent on human resources available for the project and involves projecting whether the system will be used if it is developed and implemented. It measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the analysis phase of system development. It reviews the willingness of the organization to support the proposed system.

2.2.3 Economical

It refers to the benefits or outcomes we are deriving from the product as compared to the total cost we are spending for developing the project. It the more or less same as the older system, then it is not feasible to develop the product. Economic analysis could also be referred to as cost/benefit analysis. It is the most frequently used method for evaluating the effectiveness of a new system. In economic analysis the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action.

2.3 TOOLS AND TECHNOLOGIES USED

A graphical user interface (GUI) is an interface for the user to communicate with a computer application using graphical symbols rather than typing the instructions in. The GUI of the proposed BIKE Rental system will be developed using HTML5, CSS and PHP (PHP Hypertext Processor).

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects, such as interactive forms, may be embedded into the rendered page. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags,written using angle brackets. The BIKE Rental system uses HTML as the building blocks for creating UI elements.

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. CSS is used to format the pages to make it appealing to the user. CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colours and fonts. The BIKE Rental system application uses Bootstrap 4.1 a boilerplate designed with CSS to reduce development times on the GUI design.

PHP (PHP Hypertext Processor) is a server-side scripting language used to dynamically create webpages. PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications. The BIKE Rental system uses PHP for interacting with the database using MySQL and to display dynamic content on the webpage based on the users queries.

2.4 HARDWARE AND SOFTWARE REQUIREMENTS

2.4.1 Hardware Requirements

- Processor : Pentium Duel Core or above
- RAM : 4GB
- Hard Disk : 100 GB hard disk or above
- Monitor : 15 inch Color Monitor
- Keyboard : 102/104 Keys
- Mouse : Optical Mouse

2.4.2Software Requirements

- Operating System :Windows 7/8/10 or Linux distribution
- Browser :Chrome or Firefox or any browser
- Front-end :HTML/CSS/PHP
- Backend :PHP

3. SOFTWARE REQUIREMENTS SPECIFICATION

3.1 FUNCTIONAL REQUIREMENTS

- Usability: The interface should use terms and concepts, which are drawn from the experience of the people who will make most of the system.
- Efficiency: The system must provide easy and fast access without consuming more cost
- Reliability: User should never be surprised by the behaviour of the system and it's easy to use to stored data and easy to used transfer voice data (only way files)

3.2 NON – FUNCTIONAL REQUIREMENTS

It describes aspects of the system that are concerned with how the system provides the functional requirements. They are:

- **Security**: The subsystem should provide a high level of security and integrity of the data held by the system, only authorized persons of the company can gain access to the company's.
- **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 25 seconds for less complicated task.
- **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.
- **Availability**: This system should always be available for access at 24 hours,7days a week. Also in the occurrence of any major system malfunctioning, the system should be available in 1 to 2 working days.

4. SYSTEM DESIGN

System designs the process of defining the architecture, modules, interfaces and data for a system to satisfy specific requirements. System design could be seen as the application of system theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

4.1 Module Description

Bikes:

This is one of the most important modules. This module helps the customers to take any bike on rent from any bike seller. The status of bikes can be updated as soon as it gets free or it gets booked so that no other customer tries to book the same bike. The information that is added to a bike is its model no, vehicle no, and owner name.

Login:

After registration, one can log in to the system either as the event manager or the customer. The option to register either an employee or customer is given when the user is registering. The interface of this system depends upon the registration. If the user has registered as an employee, then it will have options like arranging all the things as asked by a customer for an event.

For the customer, the interface includes the option to check all the available event managers and can select anyone for this event.

Admin:

This module is present only for one account. That is, no one can register as an admin after one account is created. The admin account has all the privileges, to check about any particular bike seller or customer. To block any account, to calculate the salary of the employees after deducting their leaves, to update the status about any event, to calculate the payment, to make changes into accounts of users, etc.

User:

As explained in the login section, the user can be of two types and both the users will have different interfaces and after the user has registered and login then the features provided can be used by the user. A user who has registered as the seller can update about the bikes that are free card can be given on rent. If the user is a customer, then he can see all the available bikes that he can take on rent.

About us:

WE ARE THE BIKE RENTAL. The only 100% dedicated bike rental booking website. The first Bike Rental shop was our own bike shop. Ever Since it has been our aim to make bike rental easier for everyone, everywhere.We focus on making bike rentals easier for you.Your rental business has a unique set of challenges.

4.2. CONTEXT DIAGRAM

Data flow diagram is graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data. The DFD does not mention anything about how data flows through the system.

There is a prominent difference between DFD and Flowchart. The flowchart depicts flow of control in program modules. DFDs depict flow of data in the system at various levels. DFD does not contain any control or branch elements.

Types of DFD

Data Flow Diagrams are either Logical or Physical.

- **Logical DFD** - This type of DFD concentrates on the system process and flow of data in the system. For example in a Banking software system, how data is moved between different entities.
- **Physical DFD** - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

DFD Components

DFD can represent Source, destination, storage and flow of data using the following set of components -

- **Entities** - Entities are source and destination of information data. Entities are represented by rectangles with their respective names.
- **Process** - Activities and action taken on the data are represented by Circle or Round-edged rectangles.
- **Data Storage** - There are two variants of data storage - it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.

- **Data Flow** - Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

Levels of DFD

- **Level 0** - Highest abstraction level DFD is known as Level 0 DFD, which depicts the entire information system as one diagram concealing all the underlying details. Level 0 DFDs are also known as context level DFDs

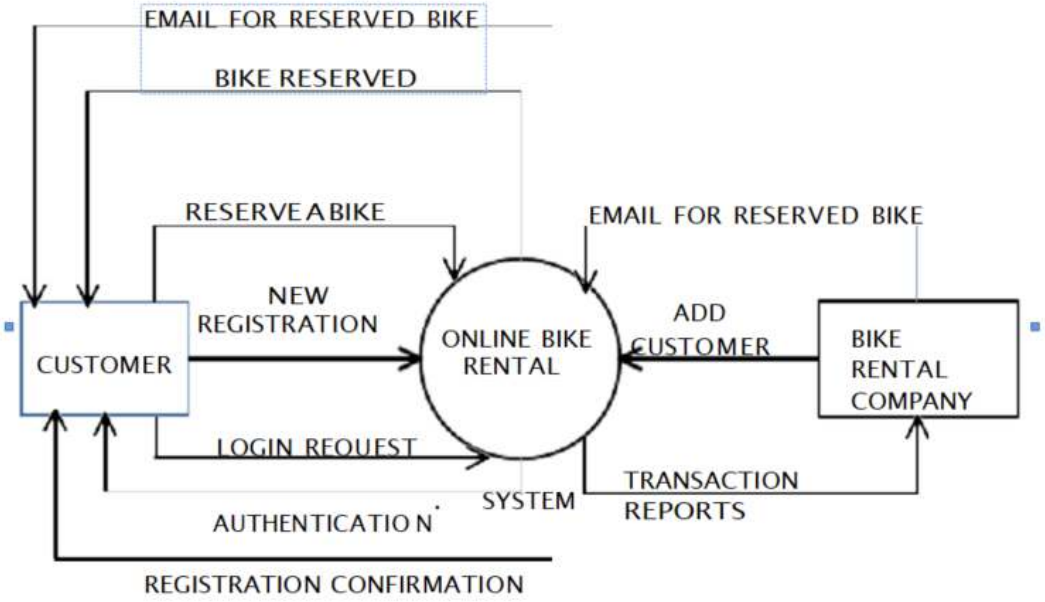


Fig: level 0

- **Level 1** - The Level 0 DFD is broken down into more specific, Level 1 DFD. Level 1 DFD depicts basic modules in the system and flow of data among various modules. Level 1 DFD also mentions basic processes and sources of information.

  Change

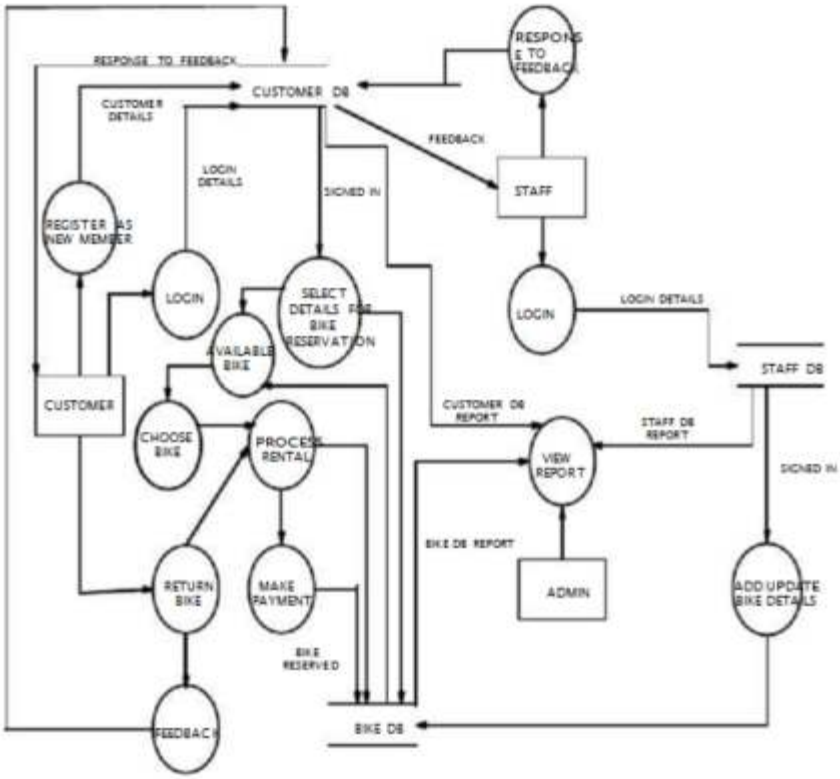


Fig: level 01 DFD

5. DETAILED DESIGN

5.1 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Purpose of Class Diagrams

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.UML diagrams like activity diagram; sequence diagram can only give the sequence flow of the application; however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as −

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.

How to Draw a Class Diagram?

Class diagrams are the most popular UML diagrams used for construction of software applications. It is very important to learn the drawing procedure of class diagram.Class diagrams have a lot of properties to consider while drawing but here the diagram will be considered from a top level view.

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represent the whole system.

The following points should be remembered while drawing a class diagram −

- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified
- For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. At the end of the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

The following diagram is an example of an Order System of an application. It describes a particular aspect of the entire application.

- First of all, Order and Customer are identified as the two elements of the system. They have a one-to-many relationship because a customer can have multiple orders.
- Order class is an abstract class and it has two concrete classes (inheritance relationship) Special Order and Normal Order.
- The two inherited classes have all the properties as the Order class. In addition, they have additional functions like dispatch () and receive ().

The following class diagram has been drawn considering all the points mentioned above.

Where to Use Class Diagrams?

Class diagram is a static diagram and it is used to model the static view of a system. The static view describes the vocabulary of the system.

Class diagram is also considered as the foundation for component and deployment diagrams. Class diagrams are not only used to visualize the static view of the system but they are also used to construct the executable code for forward and reverse engineering of any system.

Generally, UML diagrams are not directly mapped with any object-oriented programming languages but the class diagram is an exception.

Class diagram clearly shows the mapping with object-oriented languages such as Java, C++, etc. From practical experience, class diagram is generally used for construction purpose.

In a nutshell it can be said, class diagrams are used for −

- Describing the static view of the system.
- Showing the collaboration among the elements of the static view.
- Describing the functionalities performed by the system.

5.2 Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behaviour. Dynamic behaviour means the behaviour of the system when it is running/operating.

Only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them.

Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Hence to model the entire system, a number of use case diagrams are used.

Purpose of Use Case Diagrams

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams

(activity, sequence, collaboration, and State chart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams. when a system is analysed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, use case diagrams are modelled to present the outside view.

In brief, the purposes of use case diagrams can be said to be as follows –

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements is actors.

How to Draw a Use Case Diagram?

Use case diagrams are considered for high level requirement analysis of a system. When the requirements of a system are analysed, the functionalities are captured in use cases.

We can say that use cases are nothing but the system functionalities written in an organized manner. The second thing which is relevant to use cases is the actors. Actors can be defined as something that interacts with the system.

- Functionalities to be represented as use case
- Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram

- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.

- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.

Following is a sample use case diagram representing the order management system. Hence, if we look into the diagram then we will find three use cases **(Order, Special Order, and Normal Order)** and one actor which is the customer.

The Special Order and Normal Order use cases are extended from *Order* use case. Hence, they have extended relationship. Another important point is to identify the system boundary, which is shown in the picture. The actor Customer lies outside the system as it is an external user of the system.

Where to Use a Use Case Diagram?

As we have already discussed there are five diagrams in UML to model the dynamic view of a system. Now each and every model has some specific purpose to use. Actually these specific purposes are different angles of a running system.

To understand the dynamics of a system, we need to use different types of diagrams. Use case diagram is one of them and its specific purpose is to gather system requirements and actors.

Use case diagrams specify the events of a system and their flows. But use case diagram never describes how they are implemented. Use case diagram can be imagined as a black box where only the input, output, and the function of the black box are known.

These diagrams are used at a very high level of design. This high level design is refined again and again to get a complete and practical picture of the system. A well-structured use case also describes the pre-condition, post condition, and exceptions. These extra elements are used to make test cases when performing the testing.

Although use case is not a good candidate for forward and reverse engineering, still they are used in a slightly different way to make forward and reverse engineering. The same is true for reverse engineering. Use case diagram is used differently to make it suitable for reverse engineering.

In forward engineering, use case diagrams are used to make test cases and in reverse engineering use cases are used to prepare the requirement details from the existing application.

Use case diagrams can be used for −

- Requirement analysis and high level design.
- Model the context of a system.
- Reverse engineering.
- Forward engineering.

5.1.1 Use case

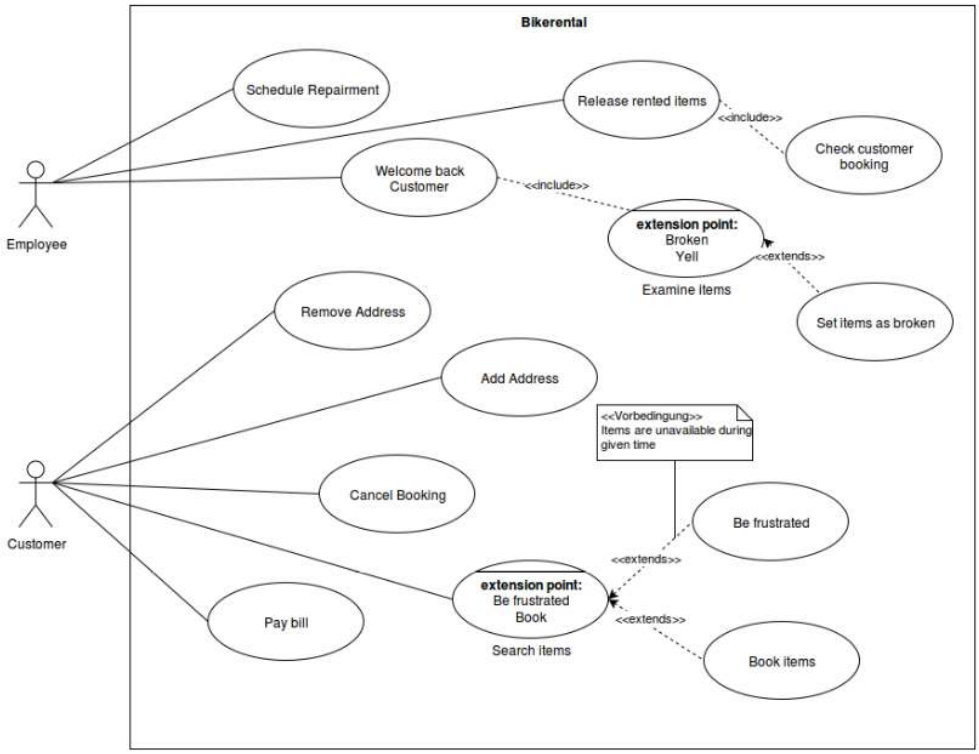


Fig: Use case diagram

5.3 Entity Relationship Diagram

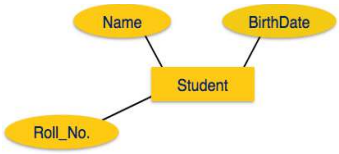
The ER Model is represented by means of an ER diagram. Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an ER diagram.

Entity

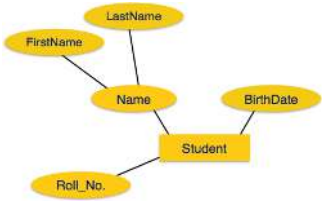
Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

Attributes

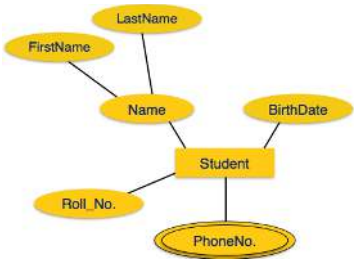
Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



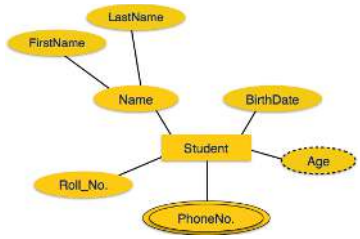
If the attributes are **composite**, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.



Multivalued attributes are depicted by double ellipse.



Derived attributes are depicted by dashed ellipse.



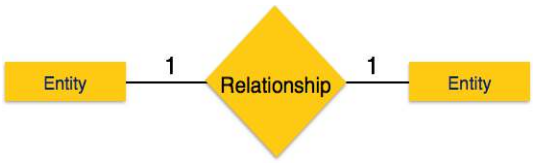
Relationship

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

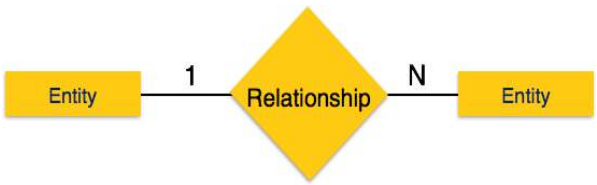
Binary Relationship and Bikedinality

A relationship where two entities are participating is called a **binary relationship**. Bikedinality is the number of instance of an entity from a relation that can be associated with the relation.

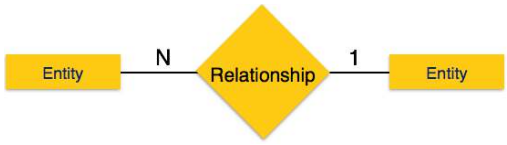
- **One-to-one** − when only one instance of an entity is associated with the relationship, it is marked as '1:1'. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship.



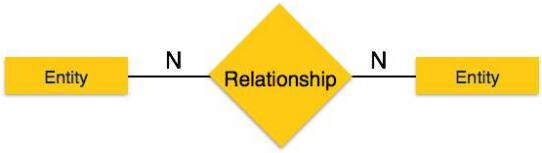
- **One-to-many** − When more than one instance of an entity is associated with a relationship, it is marked as '1: N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many relationship.



- **Many-to-one** − when more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one relationship.

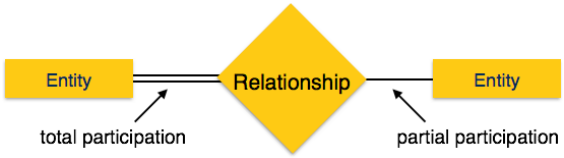


- **Many-to-many** − The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.



Participation Constraints

- **Total Participation** − each entity is involved in the relationship. Total participation is represented by double lines.
- **Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.



5.3.1 ER-Diagram

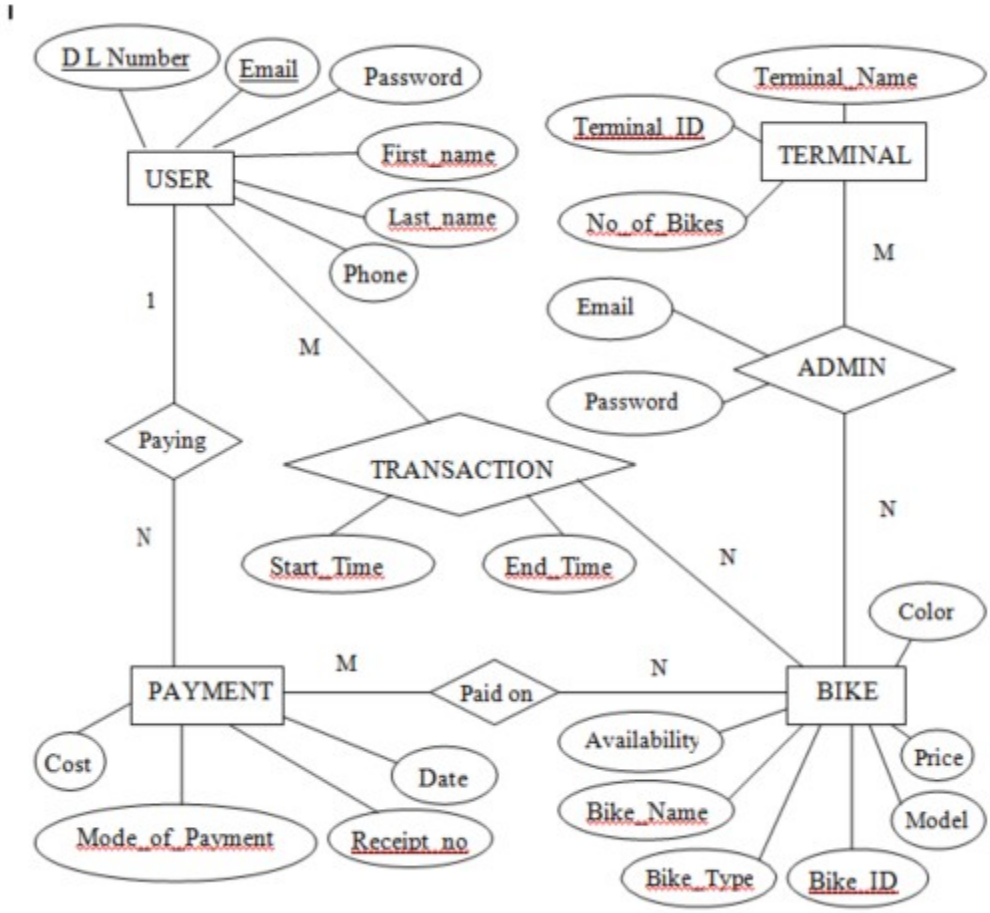


Fig:ER-Diagr

5.4 Database Design

ADMIN

CREATE TABLE ADMIN (EMAIL VARCHAR2(20),

PASSWORD VARCHAR2(15));

USER

CREATE TABLE USER (FNAME VARCHAR2(15),

LNAME VARCHAR2(15),

EMAIL VARCHAR2(20) PRIMARY KEY, PASSWORD VARCHAR2(15),

PHONE BIGINT(12),

DLNO VARCHAR2(15) UNIQUE);

TERMINAL

CREATE TABLE TERMINAL (TERM_ID INT(5) PRIMARY KEY,

TERM_NAME VARCHAR2(15), NO_OF_BIKES INT(2));

BIKE

CREATE TABLE BIKE (BIKE_ID INT(5) PRIMARY KEY,

BIKE_NAME VARCHAR2(10), MODEL YEAR,

COLOR VARCHAR2(10), BIKE_TYPE VARCHAR2(8), PRICE INT(4),

TERM_ID REFERENCES
TERMINAL(TERM_ID) ON
DELETE CASCADE,

AVAIL INT (1));

TRANSACTION

CREATE TABLE TRANSACTION (EMAIL REFERENCES USER(EMAIL) ON DELETE

ON DELETE CASCADE,

BIKE_ID REFERENCES BIKE(BIKE_ID) ON DELETE CASCADE,

START_TIME DATETIME,

END_TIME DATETIME);

5.5 Description of Tables

ADMIN

DESC ADMIN;

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	email	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
2	password	varchar(20)	latin1_swedish_ci		No	None			Change Drop More

Figure 5.5.1 ADMIN

USER

DESC USER;

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	fname	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
2	lname	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
3	email	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
4	phone	bigint(13)			No	None			Change Drop More
5	dlno	bigint(20)			No	None			Change Drop More
6	password	varchar(20)	latin1_swedish_ci		No	None			Change Drop More

Figure 5.5.2 USER

TERMINAL

DESC TERMINAL;

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	term_id	int(5)			No	None			Change Drop More
2	term_name	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
3	no_of_bikes	int(2)			No	None			Change Drop More

Figure 4.3 TERMINAL

4.4 SQL Triggers and Stored Procedure

4.4.1 Triggers

Triggers are stored programs, which are automatically executed or fired when some event occur. Triggers are, in fact, written to be executed in response to any of the following events:

•A Database manipulation (DML) statement (DELETE, INSERT, or UPDATE)

•A Database definition(DDL) statement (CREATE, ALTER, or DROP)

•A Database operation(SERVERERROR,LOGON,LOGOFF,STARTUP,SHUTDOWN)

Triggers can be defined on the table, view, schema, or database with which the event is associated. The trigger used in this application is used to increment the value of no_of_BIKEs in Terminal when BIKE is added to that terminal. Another Trigger is used to decrement the no_of_BIKEs when a BIKE in that terminal is deleted. By knowing the value of no_of _BIKEs, it is easier to get count of BIKEs in particular terminal.

The Trigger is

CREATE TRIGGER `BIKEadd` AFTER INSERT ON `BIKE` FOR EACH ROW UPDATE terminal

set no_of_BIKEs = no_of_BIKEs + 1 WHERE term_id = new.term_id;

CREATE TRIGGER `BIKErem` AFTER DELETE ON `BIKE` FOR EACH ROW UPDATE terminal

set no_of_BIKEs = no_of_BIKEs - 1 WHERE term_id = old.term_id;

Show Triggers:

	term_id	term_name	no_of_bikes
Edit Copy Delete	1020	BSK	2
Edit Copy Delete	1021	J P Nagar	2
Edit Copy Delete	1022	Hebbal	3
Edit Copy Delete	1023	Kormangala	2

Figure 4.13 Triggers

4.4.2 Stored Procedure

A stored procedure is a prepared SQL code that can be saved and can be reused over and over again. So if a query has to be written over and over again, instead of having to write that query each time, it can be saved as a stored procedure and can be executed just by calling the procedure. In addition, parameters can also be passed to the stored procedure. So depending on the need, the stored procedure can act accordingly.

Stored procedures are useful in the following circumstances:

- If a database program is needed by several applications, it can be stored at the server and invoked by any of the application programs. This reduces duplication of effort and improves software modularity.

- Executing a program at the server can reduce data transfer and communication cost between the client and server in certain situations.

• These procedures can enhance the modelling power provided by views by allowing, more complex types of derived data to be made available to the database users via the stored procedures. Additionally, they can be used to check for complex constraints that are beyond the specification power of assertions and triggers.

The Stored procedure used in this application is to calculate the cost by accepting 2 parameters. When calling this Stored Procedure, 2 parameters needs to be passed with a call.

Stored Procedure is:

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `spcost`(IN `hour` INT(3), IN `id` INT(5))

BEGIN

DECLARE price int;

DECLARE BIKEid CURSOR FOR SELECT price FROM BIKE WHERE BIKE_id=id; OPEN BIKEid;

FETCH FROM BIKEid INTO price; UPDATE payment

SET cost = (hour * price1)

WHERE BIKE_id =id and date is NULL; CLOSE BIKEid;

END$$ DELIMITER ;

Show Procedure:

Figure 4.14 Stored Procedure

←T→		email	bike_id	cost	mode	receipt_no	date
Edit Copy Delete		vinay@gmail.com	102	512.00	card	22	2019-11-12 13:15:00
Edit Copy Delete		pavan@gmail.com	103	640.00	paypal	25	2019-11-13 13:13:00
Edit Copy Delete		pavan@gmail.com	105	614.00	paypal	28	2019-11-03 11:17:02

4.5 Database Connectivity

The front end can easily be connected to the back end/database (i.e., MySql) by adding a few instructions in PHP. The following instructions are to be added.

```
<?php

//initializing variables

// function OpenCon() {

$servername = "localhost";

$username = "root";

$password = "";

$db ="BIKErental";

//connect to server

$conn = mysqli_connect ($servername , $username , $password,$db) or die("unable to connect to host");

//return $conn;

//}

?>
```

6. IMPLEMENTATION

6.1 PSEUDO CODE

Index page Code

```
<?php
session_start();
include('includes/config.php');
error_reporting(0);?>
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,initial-scale=1">
<meta name="keywords" content="">
<meta name="description" content="">
<title>Bike Rental Portal</title>
<!--Bootstrap -->
<link rel="stylesheet" href="assets/css/bootstrap.min.css" type="text/css">
<link rel="stylesheet" href="assets/css/style.css" type="text/css">
<link rel="stylesheet" href="assets/css/owl.bikeousel.css" type="text/css">
<link rel="stylesheet" href="assets/css/owl.transitions.css" type="text/css">
<link href="assets/css/slick.css" rel="stylesheet">
<link href="assets/css/bootstrap-slider.min.css" rel="stylesheet">
<link href="assets/css/font-awesome.min.css" rel="stylesheet">
<linkrel="stylesheetid="switcher-css"type="text/css"
href="assets/switcher/css/switcher.css" media="all" />
```

```
<link rel="alternate stylesheet" type="text/css" href="assets/switcher/css/red.css" title="red" media="all" data-default-color="true" />

<linkrel="alternatestylesheet" type="text/css" href="assets/switcher/css/orange.css" title="orange" media="all" />

        <link rel="alternate stylesheet" type="text/css" href="assets/switcher/css/blue.css" title="blue" media="all" />

        <link rel="alternate stylesheet" type="text/css" href="assets/switcher/css/pink.css" title="pink" media="all" />

        <link rel="alternate stylesheet" type="text/css" href="assets/switcher/css/green.css" title="green" media="all" />

        <link rel="alternate stylesheet" type="text/css" href="assets/switcher/css/purple.css" title="purple" media="all" />

<link rel="apple-touch-icon-precomposed" sizes="144x144" href="assets/images/favicon-icon/apple-touch-icon-144-precomposed.png">

<link rel="apple-touch-icon-precomposed" sizes="114x114" href="assets/images/favicon-icon/apple-touch-icon-114-precomposed.html">

<link rel="apple-touch-icon-precomposed" sizes="72x72" href="assets/images/favicon-icon/apple-touch-icon-72-precomposed.png">

<link rel="apple-touch-icon-precomposed" href="assets/images/favicon-icon/apple-touch-icon-57-precomposed.png">

<link rel="shortcut icon" href="assets/images/favicon-icon/favicon.png">

<link href="https://fonts.googleapis.com/css?family=Lato:300,400,700,900" rel="stylesheet">

</head>

<body>

<!-- Start Switcher -->
```

```
<?php include('includes/colorswitcher.php');?>

<!-- /Switcher -->


<!--Header-->

<?php include('includes/header.php');?>

<!-- /Header -->

<!-- Banners -->

<section id="banner" class="banner-section">

  <div class="container">

    <div class="div_zindex">

      <div class="row">

        <div class="col-md-5 col-md-push-7">

          <div class="banner_content">

            <h1>Find the right bike for you.</h1>

            <p>We have more than a thousand bikes for you to choose. </p>

                  <a href="#" class="btn">Read More <span class="angle_arrow"><i class="fa fa-angle-right" aria-hidden="true"></i></span></a> </div>

        </div>

      </div>

    </div>

  </div>

</section>

<!-- /Banners -->
```

```
<!-- Resent Cat-->

<section class="section-padding gray-bg">

 <div class="container">

  <div class="section-header text-center">

   <h2>Find the Best <span>BikeForYou</span></h2>

    <p>There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn't anything embarrassing hidden in the middle of text.</p>

  </div>

  <div class="row">


   <!-- Nav tabs -->

   <div class="recent-tab">

    <ul class="nav nav-tabs" role="tablist">

     <li role="presentation" class="active"><a href="#resentnewbike" role="tab" data-toggle="tab">New Bike</a></li>

    </ul>

   </div>

   <!-- Recently Listed New Bikes -->

   <div class="tab-content">

    <div role="tabpanel" class="tab-pane active" id="resentnewbike">

<?php$sql="SELECT tblvehicles.VehiclesTitle,tblbrands.BrandName,tblvehicles.PricePerDay,tblvehicles.F
```

```
uelType,tblvehicles.ModelYear,tblvehicles.id,tblvehicles.SeatingCapacity,tblvehicles.
VehiclesOverview,tblvehicles.Vimage1 from tblvehicles join tblbrands on
tblbrands.id=tblvehicles.VehiclesBrand";

$query = $dbh -> prepare($sql);

$query->execute();

$results=$query->fetchAll(PDO::FETCH_OBJ);

$cnt=1;

if($query->rowCount() > 0)

{

foreach($results as $result)

{

?>

<div class="col-list-3">

<div class="recent-bike-list">

<div class="bike-info-box"> <a href="vehical-details.php?vhid=<?php echo
htmlentities($result->id);?>"><img src="admin/img/vehicleimages/<?php echo
htmlentities($result->Vimage1);?>" class="img-responsive" alt="image"></a>

<ul>

<li><i class="fa fa-bike" aria-hidden="true"></i><?php echo htmlentities($result-
>FuelType);?></li>

<li><i class="fa fa-calendar" aria-hidden="true"></i><?php echo
htmlentities($result->ModelYear);?> Model</li>

<li><i class="fa fa-user" aria-hidden="true"></i><?php echo htmlentities($result-
>SeatingCapacity);?> seats</li>

</ul>
```

```
</div>

<div class="bike-title-m">

<h6><a href="vehical-details.php?vhid=<?php echo htmlentities($result->id);?>"><?php echo htmlentities($result->BrandName);?> , <?php echo htmlentities($result->VehiclesTitle);?></a></h6>

<span class="price">$<?php echo htmlentities($result->PricePerDay);?> /Day</span>

</div>

<div class="inventory_info_m">

<p><?php echo substr($result->VehiclesOverview,0,70);?></p>

</div>

</div>

</div>

<?php }}?>


    </div>

   </div>

  </div>

</section>

<!-- /Resent Cat -->

<!-- Fun Facts-->

<section class="fun-facts-section">

  <div class="container div_zindex">

    <div class="row">
```

```
<div class="col-lg-3 col-xs-6 col-sm-3">
 <div class="fun-facts-m">
  <div class="cell">
   <h2><i class="fa fa-calendar" aria-hidden="true"></i>40+</h2>
   <p>Years In Business</p>
  </div>
 </div>
</div>
<div class="col-lg-3 col-xs-6 col-sm-3">
 <div class="fun-facts-m">
  <div class="cell">
   <h2><i class="fa fa-bike" aria-hidden="true"></i>1200+</h2>
   <p>New Bikes For Sale</p>
  </div>
 </div>
</div>
<div class="col-lg-3 col-xs-6 col-sm-3">
 <div class="fun-facts-m">
  <div class="cell">
   <h2><i class="fa fa-bike" aria-hidden="true"></i>1000+</h2>
   <p>Used Bikes For Sale</p>
  </div>
 </div>
```

```
    </div>
    <div class="col-lg-3 col-xs-6 col-sm-3">
     <div class="fun-facts-m">
      <div class="cell">
       <h2><i class="fa fa-user-circle-o" aria-hidden="true"></i>600+</h2>
       <p>Satisfied Customers</p>
      </div>
     </div>
    </div>
   </div>
  </div>
  <!-- Dark Overlay-->
  <div class="dark-overlay"></div>
</section>
<!-- /Fun Facts-->
<!--Testimonial -->
<section class="section-padding testimonial-section parallex-bg">
  <div class="container div_zindex">
   <div class="section-header white-text text-center">
    <h2>Our Satisfied <span>Customers</span></h2>
   </div>
   <div class="row">
    <div id="testimonial-slider">
```

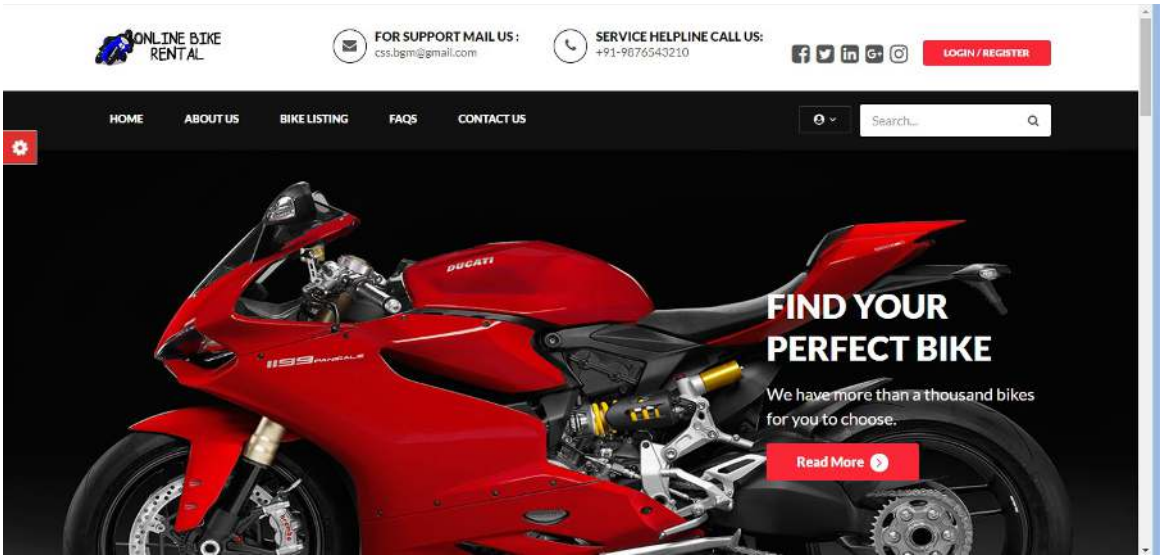
```
<?php

$tid=1;

$sql = "SELECT tbltestimonial.Testimonial,tblusers.FullName from tbltestimonial join tblusers on tbltestimonial.UserEmail=tblusers.EmailId where tbltestimonial.status=:tid";

$query = $dbh -> prepare($sql);

$query->bindParam(':tid',$tid, PDO::PARAM_STR);

$query->execute();

$results=$query->fetchAll(PDO::FETCH_OBJ);

$cnt=1;

if($query->rowCount() > 0)

{

foreach($results as $result)

{  ?>

      <div class="testimonial-m">

            <div class="testimonial-img"> <img src="assets/images/cat-profile.png" alt="" /> </div>

        <div class="testimonial-content">

          <div class="testimonial-heading">

            <h5><?php echo htmlentities($result->FullName);?></h5>

          <p><?php echo htmlentities($result->Testimonial);?></p>

        </div>

      </div>

      </div>
```

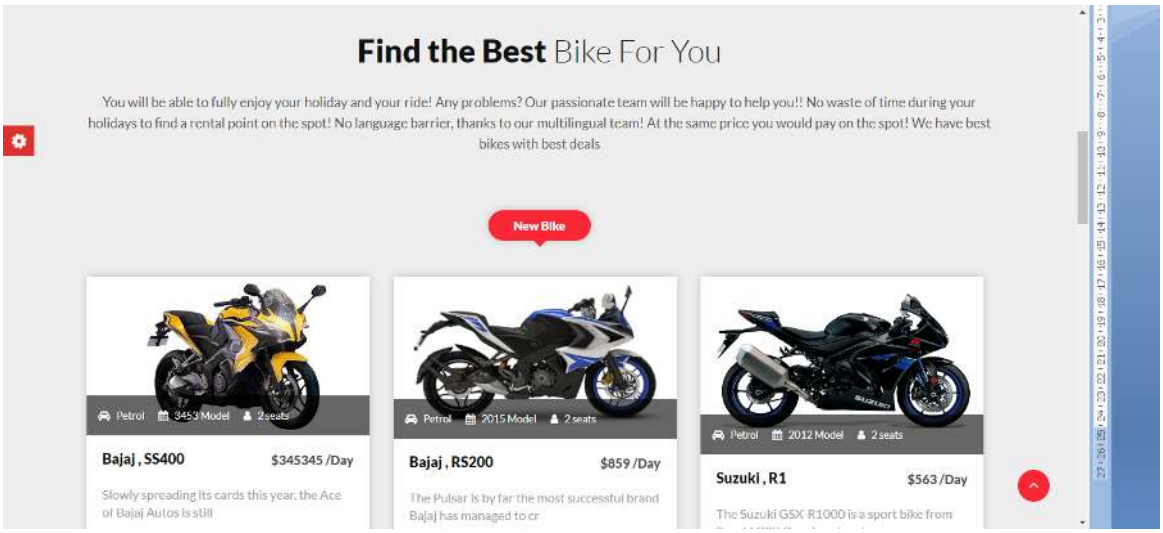
```
      <?php }} ?>


    </div>
   </div>
  </div>
  <!-- Dark Overlay-->
  <div class="dark-overlay"></div>
</section>
<!-- /Testimonial-->
<!--Footer -->
<?php include('includes/footer.php');?>
<!-- /Footer-->
<!--Back to top-->
<div id="back-top" class="back-top"> <a href="#top"><i class="fa fa-angle-up" aria-hidden="true"></i> </a> </div>
<!--/Back to top-->
<!--Login-Form -->
<?php include('includes/login.php');?>
<!--/Login-Form -->
<!--Register-Form -->
<?php include('includes/registration.php');?>
<!--/Register-Form -->
<!--Forgot-password-Form -->
```

```
<?php include('includes/forgotpassword.php');?>

<!--/Forgot-password-Form -->

<!-- Scripts -->

<script src="assets/js/jquery.min.js"></script>

<script src="assets/js/bootstrap.min.js"></script>

<script src="assets/js/interface.js"></script>

<!--Switcher-->

<script src="assets/switcher/js/switcher.js"></script>

<!--bootstrap-slider-JS-->

<script src="assets/js/bootstrap-slider.min.js"></script>

<!--Slider-JS-->

<script src="assets/js/slick.min.js"></script>

<scriptsrc="assets/js/owl.bikeousel.min.js"></script>

</body>

<!-- Mirrored from themes.webmasterdriver.net/bikeforyou/demo/index.html by HTTrack Website Copier/3.x [XR&CO'2014], Fri, 16 Jun 2017 07:22:11 GMT -->

</html>
```

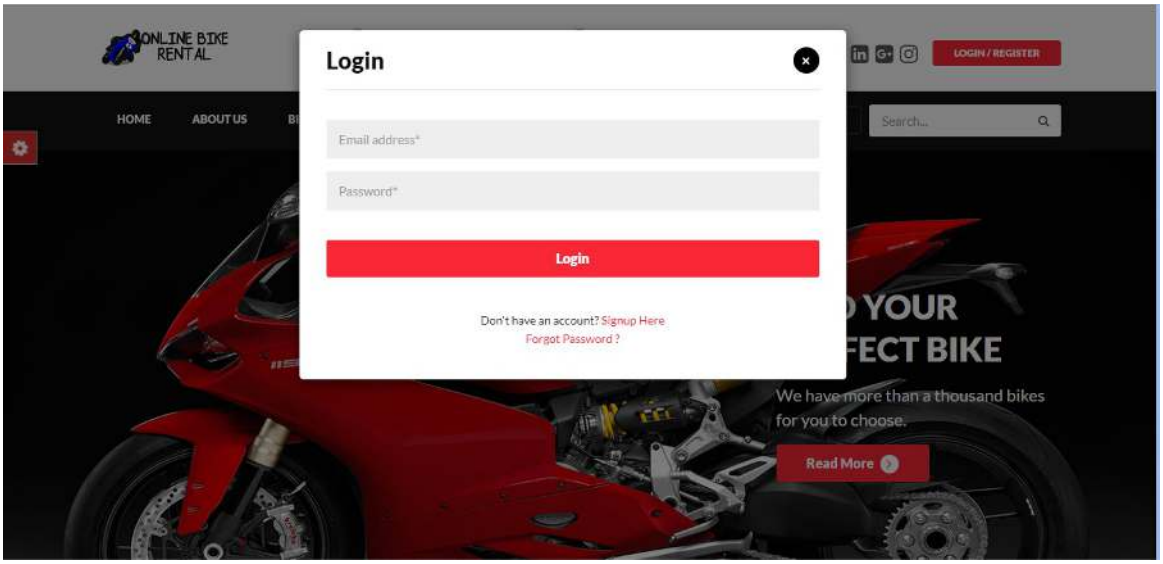
6.2 SCREENSHOTS

6.2.1 Homepage

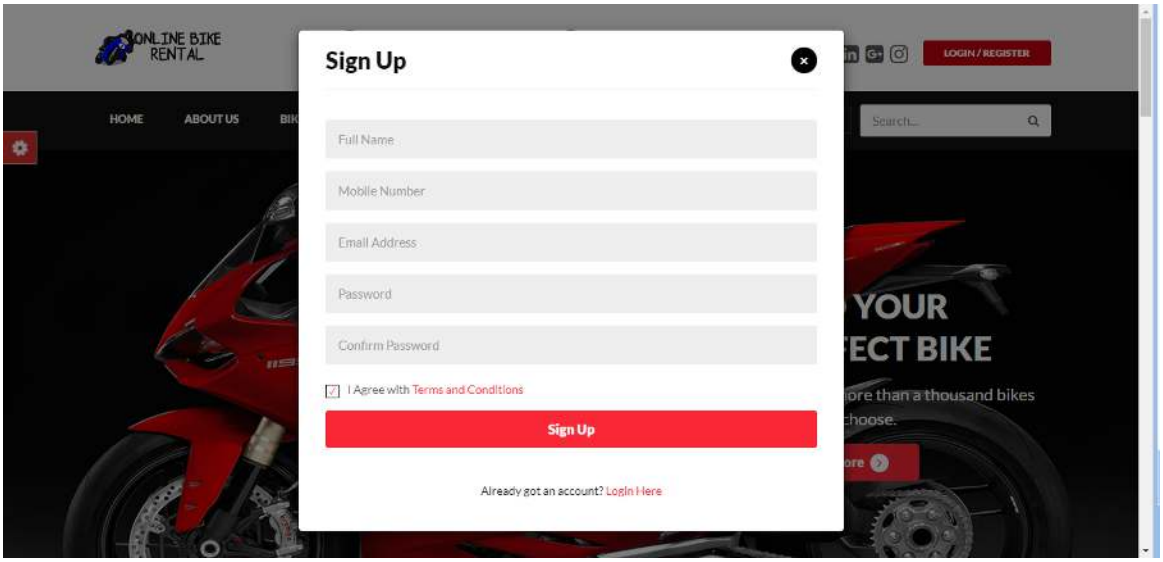




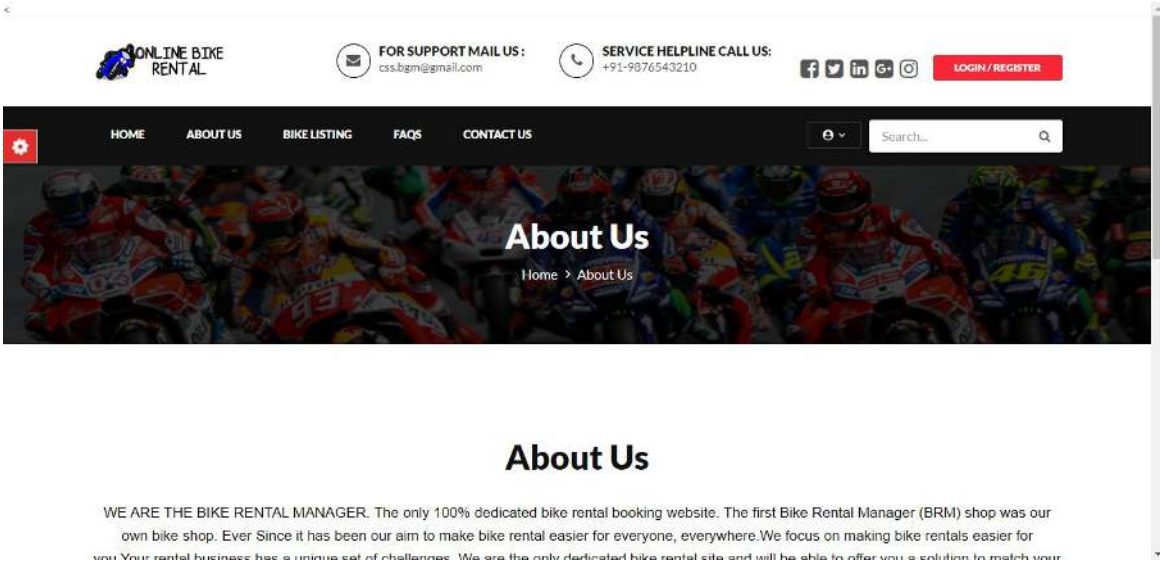
6.2.2 Login



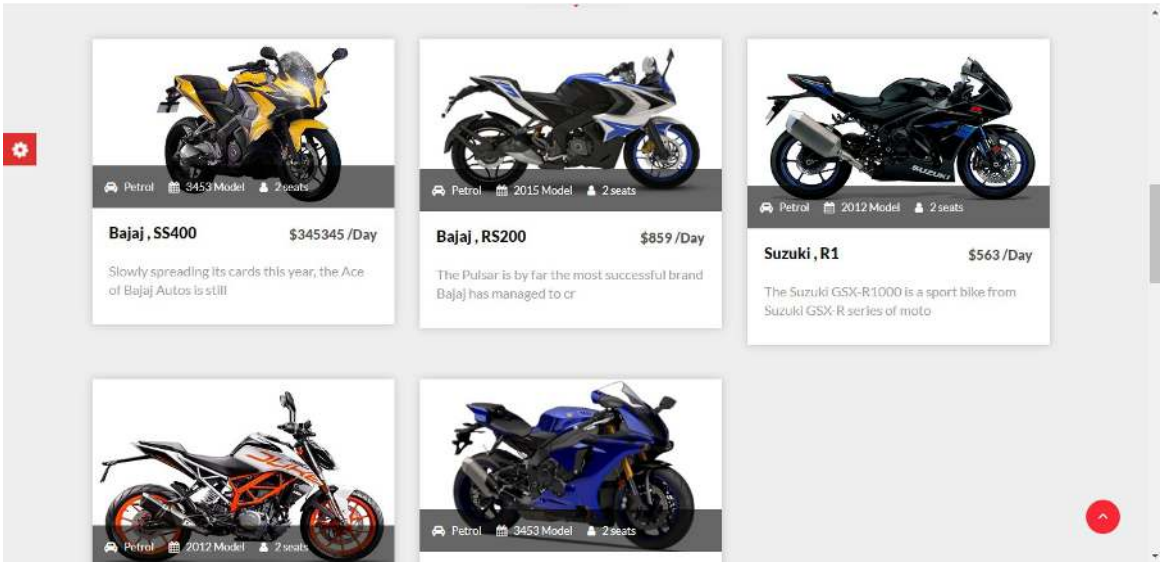
6.2.4 Registration



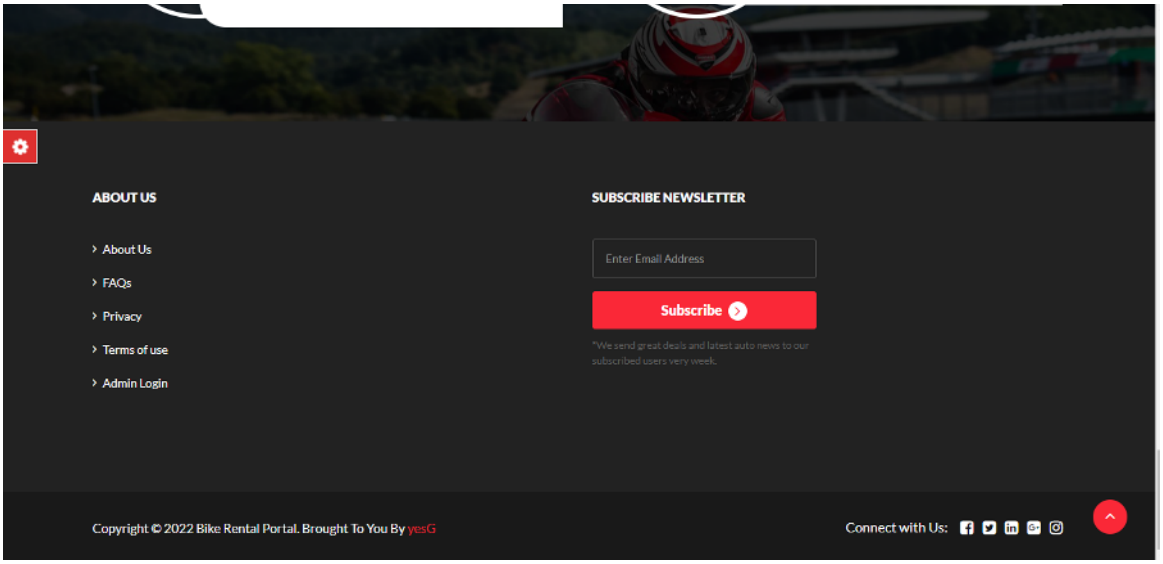
6.2.6 About



6.2.7 Listing Bike



6.1.8 Footer



7. TESTING AND IMPLEMENTATION

7.1 Introduction:

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, designing and coding.

Testing is the process of executing a program with the intent of finding errors. During testing, the program to be tested is executed with a set of test cases, and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected.

Testing Objectives:

1. Testing is process of executing a program with the intent of finding an error.
2. A good test case design is one that has a probability of finding an as yet undiscovered error.
3. A successful test is one that uncovers an as yet undiscovered error.
   These above objectives imply a dramatic change in view port. Testing cannot show the absence of defects, it can only show that software errors are present.

Testing methodologies:

Unit Testing:

Unit testing focuses verification effort on the smallest unit of software design that is the module. This test focuses on each module individually ensuring that it properly as a unit. Hence the naming is unit testing so that each module is tested individually.

Integration Testing:

It is a systematic technique for constructing different program module into an integrated software structure. This test uncovers the errors during the entire module and validated.

Output Testing:

Output testing is done to verify whether the given output is right or wrong.

Validation Testing:

After the integration testing software is ready as per the specification. But it has to be validated as per the specification and uncover the unexpected future errors and to improve its reliability.

Software Testing Strategies:

A software testing strategy provides a road map for the software developer. Testing is a set of activities that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be defined for software engineering process. Any software testing strategy should have the following characteristics:

1. Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
2. Different testing techniques are appropriate at different points in time.
3. The developer of the software and an independent test group conducts testing.
4. Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

8. CONCLUSION

BIKE Rental System is a web application and it is restricted to only limited type of users. In this application, Admin have been given access rights and are restricted up to certain functionalities, so that the data is maintained securely and redundant data is prevented. As the Data is stored electronically, it is necessary to have a Computer and Network connection to access the Application. It is a software which helps the user to rent BIKE base on their need. This software reduces the amount of manual data entry and gives greater efficiency. The User Interface of it is very friendly and can be easily used by anyone. It also decreases the amount of time taken to write details and other modules. At the end, this software can perform all the tasks accurately and can do the work for which it is made.

Any Suggestions are Welcome.

9. FUTURE ENHANCEMENT

Once the final Online car rental , is built, users or clients can book the bike already exists in the web application. In future, users or clients can add bikes to the rent section and also we like to add slide bar of trending bikes in web Page. Also we like to add a Chat Bot which helps user or clients to enquire their doubts related to the application

Further enhancement, the use of search engine can be customisable using the filter option according to the user or a reader.