PARKING MANAGEMENT SYSTEM

Hibernate Based Application

A Major Software Project
Submitted in partial fulfillment of the requirements for the
Award of degree of BCA

(2017-2020)

Submitted by:

Rahul Buswala

Under Guidance of:

Mr. Ajay Kumar

BHARATI VIDYAPEETH UNIVERSITY

Academic Study Center – BVUSDE, New Delhi

An ISO 9001:2008 Certified Institute

"A" Grade Accreditation by NAAC

Student Undertaking

I Rahul Buswala have completed the Major Software Project titled "Parking Management System" in BVIMR, New Delhi under the guidance of Mr. Ajay Kumar in the partial fulfillment of the requirement for the award of degree of BCA of **BVUSDE, Academic Study Center BVIMR, New Delhi**. This is an original piece of work & I have neither copied and nor submitted it earlier elsewhere.

Students Signature

Certificate

This is to certify that the Major Software Project titled "Parking Management System" is an academic work done by " Rahul Buswala " submitted in the partial fulfillment of the requirement for the award of the Degree of "Bachelor of Computer Application" from "**BVUSDE, Academic Study Center BVIMR, New Delhi"** under my guidance & direction. To the best of my knowledge and belief the data & information presented by him/her in the project has not been submitted earlier.

Mr. Ajay Kumar
Faculty Guide

Certificate

This is to certify that the Major Software Project titled "Parking Management System" is an academic work done by "Rahul Buswala" submitted in the partial fulfillment of the requirement for the award of the Degree of "Bachelor of Computer Application" from "**BVUSDE, Academic Study Center BVIMR, New Delhi**".

(Director)

ACKNOWLEDGEMENT

As with any project, the effort is widely collaborative, so I have many people to thank. This project itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals.

This project would never have seen the light of day without help and guidance that I have received. I express my sincere thanks to Mr.Ajay Kumar, for his eminent support.

I would like to thank my project guide Mr. Ajay Kumar, who helped me in great way by giving his precious time. The project has been thoroughly edited and tested so I believe this project likely to be error-free.

Table of Contents

CHAPTER 1- INTRODUCTION

1.1 PROJECT INTRODUCTION

As an important component of traffic system, parking management system is playing an important role and affecting people's daily life. By detecting and processing the information from parking lots, smart parking system allows drivers to obtain real-time parking information and alleviates parking contentions. The use of intelligent systems have become the most prevalent among the world contribute to the implementation of daily business in a more efficient and flexible reveal. In this research project, we study state-of-the-art parking policies in smart parking systems, and show that the smart parking system needs to be "smarter". Our design goals of the smart parking systems include: simplify the operations of parking systems, improve drivers' satisfaction, increase parking revenue, and alleviate traffic congestion. Objectives is to alleviate the parking contention, balance the benefits between parking service providers and drivers, coordinate among service providers, differentiate the needs of individual drivers, and reduce the amount of traffic searching for parking as well. To achieve the design goals, a powerful tool to model the behavior of both service providers and drivers is required. Meanwhile, we need to design control signals to guide the parking selection of large scale, autonomous drivers. Through analysis, design, implementation and testing phases, we introduce the proposed parking management system prototype that has the potential to achieve the above goals.

1.2 OBJECTIVES & SCOPE OF PROPOSED SOFTWARE

OBJECTIVE: -

Objectives are one of the characteristics of a system. Objectives may be real or stated. The user must know the central objectives of the computer application in the analysis for a successful design and conversion. The analyst must work around such obstacles to Identify the real objectives of the proposed change. Following are the objectives of the proposed system: -

1. Provide a automate system to easily keep record of the parked vehicle.
2. There is uniqueness provided to customers so that there is no inaccuracy

done on their behalf. This is provided through Reg.no.

3. Confirmation status can be efficiently known just by clicking record status submenu on the form which acts as front end.
4. To handle the addition of cars in a computerized manner.
5. To handle the removal of cars in a computerized way.
6. To answer the user query automatically.
7. Computerized version removes some of the basic limitations of the existing i.e. manual system.

SCOPE: -

1. This software has been built in two highly popular software on Eclipse and MySQL. This software provides GUI (Graphical User Interface) and DBMS.
2. This software can be used by any organization or for general use in infrastructure who need automated system to store parking entries in database.
3. This software will have its scope in efficient database handling as there will be no redundancy of data.
4. Error free database can result in better analysis of the system output.
5. Computerized version will reduce the labor cost.

CHAPTER2- SYSTEM ANALYSIS

2.1 FEASIBILITY STUDY

Depending on the results of the initial investigation, the survey is expended to a more detailed study. A feasibility study is a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of Resources. It focuses on three major questions:

1. What are the user's demonstrable needs and how does a Candidate system meet them?
2. What resources are available for given candidate system? Is the problem worth solving?
3. What are the likely impacts of the candidate's system on the organization? How well does it fit within the organization's Master MIS plan?

Each of these questions must be answered carefully. They resolve around investigation and evaluation of the problem, identification and description of candidate systems, specification of performance and the cost of each system, and final selection of the best system.

The objective of a feasibility study is not to solve the problem but to acquire a sense of its scope of its scope. During the study, the problem definition is crystallized and aspects of the problem to be included in the system are determined. Consequently, costs and benefits are estimated with greater accuracy at this stage.

The result of the feasibility study is a formal proposal. This is simply a report-a formal document detailing the nature and scope of the proposed solution. The proposal summarizes what is known and what is going to be done. It consists of the following sections: -

1. *Statement of the problem*- a carefully worded statement of the problem that led analysis.

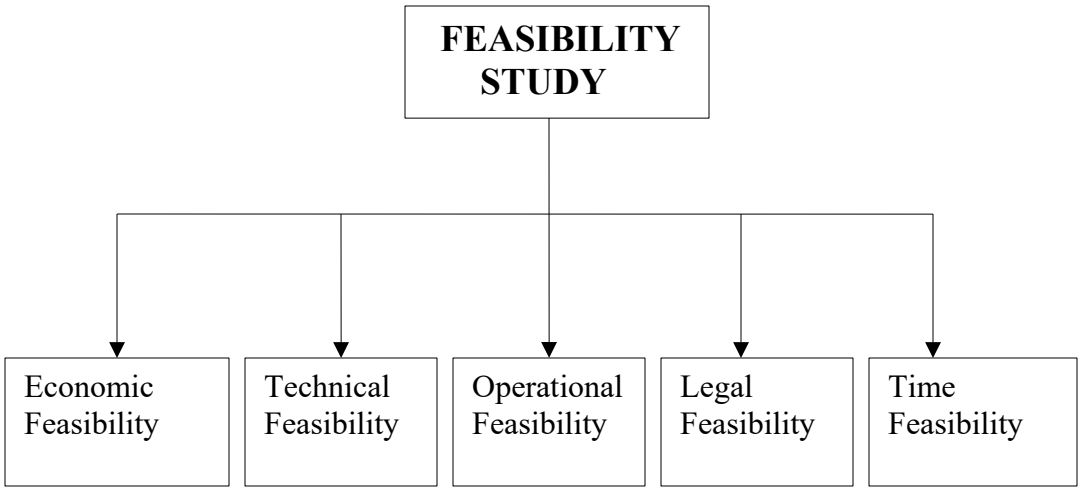
2. *Summary of findings and recommendations*- a list of the major findings and recommendations of the study. It is ideal for the user who requires quick access to the results of the analysis of the system under study. Conclusions are stated, followed by a list of the recommendations and a justification for them.

3. *Details of findings*- an outline of the methods and procedures undertaken by the existing system followed by coverage of the objectives and procedures of the candidate system. Included are also discussions of output reports, file structures, and costs and benefits of the candidate system.

4. *Recommendations and conclusions*- specific recommendation regarding the candidate system, including personnel assignments, costs, project schedules and target dates.

After management reviews the proposal, it becomes a formal agreement that paves the way for actual design and implementation. This is a crucial decision point in the life cycle. Many projects die here, whereas the more promising ones continue through implementation. Changes in the proposal are made in writing. depending on the complexity, size and cost of the project. It is simply common sense to verify changes before committing the project to design.

2.1.1 Types of feasibility study:



1) Economic feasibility: - is the major of cost of effectiveness of Nothing but judging whether the possible benefit of solving Problem is worthwhile or not.

2) Technical feasibility: - is concerned with the availability of hardware and software requirement for the development of the system. These three issues addressed during this study:

Q. Is the proposed technology proven and practical?

Ans. At this stage analyst has to see or identify the proposed technology its ability or scope of solving the problem.

Q. Does the firm possess the necessary technology its needs?

Ans. We have insured that the require technology is practical Available.

Q. Is the software and hardware being available with the availability of technical expertise?

Ans. It may be difficult to find the skill men power.

3.) Operational feasibility: - is all about problems that may arise During operations there are other issues related with Operational feasibility.

I) Information: - The system needs to provide adequate, timely, Accurate and useful information.

ii) Response time: - It needs to study the response time of the system in terms of throughput. It's should be fast enough to give the require output to the users.

iii) Accuracy: - A software system must operate accurately it means that it should provide value to its users.

4). Legal feasibility: - the issues which are considered in legal Feasibility might include copy write low, labor low, foreign Trade etc.

5.) Time feasibility: the main issues arise is that software System is possible to develop with in the time limit specified by the users.

2.2 ANALYSIS

Analysis is a detailed study of the various operation performed by a system and their relationships within and outside of the system. A key question is: what must be done to solve the problem? One aspect of analysis is defining the boundaries of the system and determining whether or not a candidate system should consider other related systems. During analysis, data are collected on the available files, decision points, and transactions handled by the present system.

Data flow diagrams, interviews, on-site observations and questionnaires are examples. The interview is a commonly used tool in analysis. It requires special skills and sensitivity to the subjects being interviewed. Bias in data collection and interpretation can be a problem. Training, experience, and common sense are required for collection of the information needed to do the analysis.

Once analysis is completed, the analyst has a firm understanding or what is to be done. The next step is to decide how the problem might be solved. Thus, in systems design, we move from the logical to the physical aspect of the life cycle.

Requirement analysis

In this, requirements of the to be developed software are established. These are usually the services it will provide, its constraints and the goal of the software.

SOFTWARE REQUIREMENTS SPECIFICATION (SRS):

. This document is to be two types of system requirements: - System specification is a contract between the client and the software programmer. Software Requirement Specification is a set of complete and precisely stated property along with the constraints. A well design SRS establishes boundaries and solution of the system to develop a useful software

- Hardware
- Software

Rules for specifying software requirements: -

- Apply and use an industry standard to ensure that standard Formats are used to describe the requirement completeness and consistency must be maintained.
- Use standard models to specify the functional relationship and data between the system and data structure to express complete requirement.
- Limit the structure of each requirement and try to increase the ability to check for completeness.
- Each requirement should be verifiable and stated.

Characteristics of a Software Requirement Specification: -

- The entire requirement must be stated unambiguously. Every requirement stated has only one interpretation.
- It should be complete which means all the function and
  Constraints should be included intended by the system user.
- The requirements should be realistic and achievable with the current technology. It must be verifiable and consistent.

2.3 Hardware & Software Requirement

HARDWARE REQUIREMENTS: -

1) PROCESSOR: Intel(R) Core(TM)
2) RAM: 2 GB
3) HARD DISK: 80 GB

SOFTWARE REQUIREMENTS: -

1) ECLIPSE
2) WINDOWS 7(OPERATING SYSTEM)
3) HIBERNATE
4). MySQL
5). Java 8

CHAPTER 3 - SYSTEM DESIGN

3.1 DESIGN METHODOLOGY

Every software development methodology approach acts as a basis for applying specific frameworks to develop and maintain software. Several software developments approaches have been used since the origin of information technology.

Broadly these are:

- Software development life cycle methodology
- Agile methodology

There are many models under these methodologies:

➢ Software development life cycle:

- Waterfall: a linear framework
- Spiral: a combined linear-iterative framework
- Incremental: a combined linear-iterative framework or V Model
- Prototyping: an iterative framework
- Rapid application development (RAD): an iterative framework

➢ Agile methodology:

- Scrum
- Extreme programming
- Adaptive software development.
- Dynamic system development method (DSDM)
- Waterfall development

The waterfall model is a sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance. The first formal

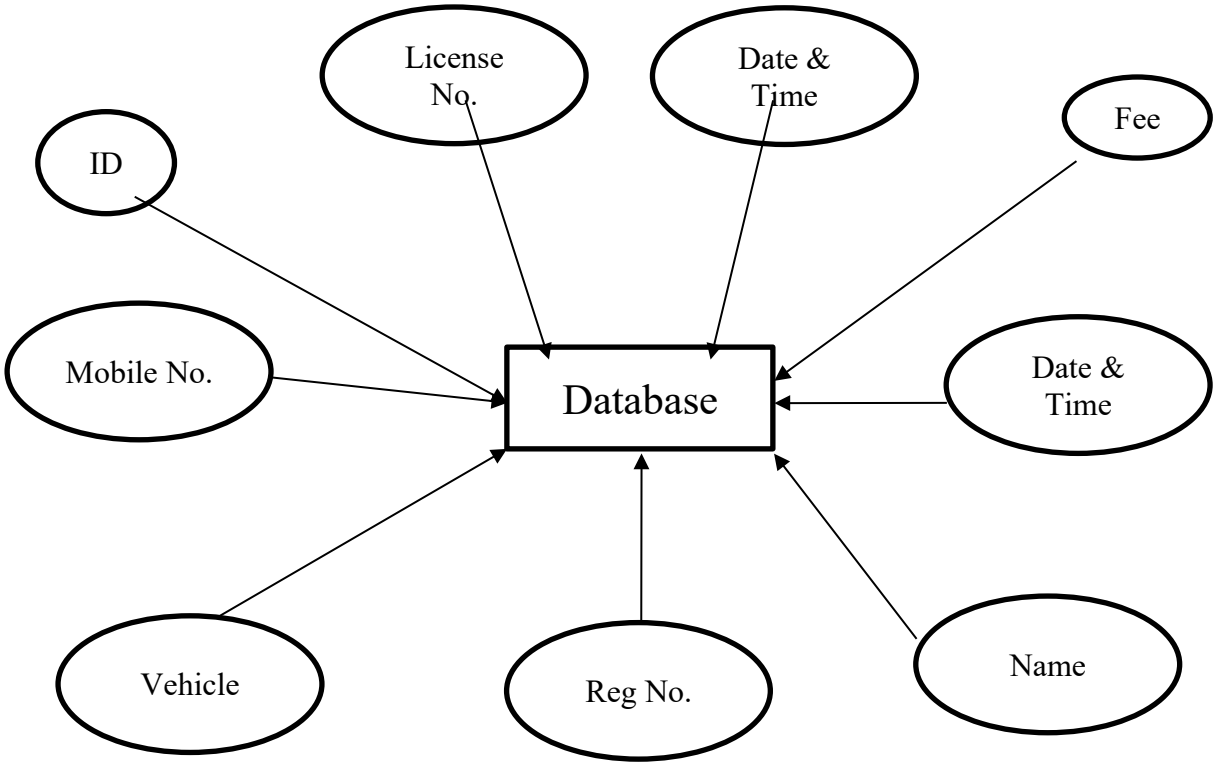
description of the method is often cited as an article published by Winston W. Royce [3] in 1970 although Royce did not use the term "waterfall" in this article.

The basic principles are:

- Project is divided into sequential phases, with some overlap and splash back
  acceptable between phases.
- Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.

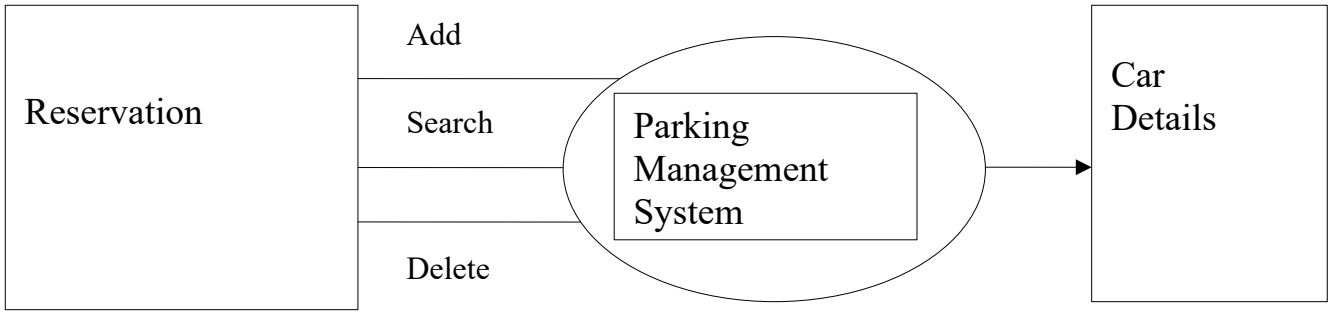
The Waterfall model is a traditional engineering approach applied to software engineering. It has been widely blamed for several large-scale government projects running over budget, over time and sometimes failing to deliver on requirements due to the Big Design Up Front approach. Except when contractually required, the Waterfall model has been largely superseded by more flexible and versatile methodologies developed specifically for software development.

3.2 DATABASE DESIGN



DATA FLOW DIAGRAM

1.ZERO-LEVEL DFD:



3.3 CODE-SEGMENT

Screen.class

```
package com.bvp;
import javax.swing.*;
import java.awt.event.*;

public class Screen extends JFrame implements ActionListener
{
                JButton car;
                JButton bus;
                JButton riksha;
                JButton bike;
                JButton close;
                JButton records;
                JButton departure;
                JButton about;
                JButton reset;
                Thread t;
                JLabel select;
        Screen(String title)
        {
                super(title);
                setSize(800,600);
                setLayout(null);
                setLocationRelativeTo(null);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
                setUndecorated(true);
```

```
t = Thread.currentThread();

ImageIcon pcar = new ImageIcon("resources/car.png");
ImageIcon pbus = new ImageIcon("resources/bus.png");
ImageIcon priksha = new ImageIcon("resources/riksha.png");
ImageIcon pbike = new ImageIcon("resources/bike.png");
ImageIcon precords = new ImageIcon("resources/records.png");

ImageIcon icon = new ImageIcon("resources/icon.png");
setIconImage(icon.getImage());

JLabel lb;
ImageIcon img = new ImageIcon("resources/screen.jpg");
lb = new JLabel("",img,SwingConstants.CENTER);
lb.setBounds(0,0, 800, 600);

ImageIcon pclose = new ImageIcon("resources/close.png");
ImageIcon pdeparture = new ImageIcon("resources/departure.png");
ImageIcon pabout = new ImageIcon("resources/about.png");
ImageIcon preset = new ImageIcon("resources/reset.png");

car = new JButton("Car",pcar);
bus = new JButton("Bus",pbus);
bike = new JButton("Bike",pbike);
riksha = new JButton("Riksha",priksha);
close = new JButton("Close",pclose);
records = new JButton("Records",precords);
departure = new JButton("Departure",pdeparture);
select = new JLabel("Select vehicle: ");
about = new JButton("About",pabout);
reset = new JButton("Reset",preset);
```

```
car.setBounds  (50,100,110,100);
bike.setBounds          (50,210,110,100);
riksha.setBounds(50,320,110,100);
bus.setBounds (50,430,110,100);
close.setBounds         (740,5,50,50);
records.setBounds(430,410,83,100);
departure.setBounds(300,400,87,100);
about.setBounds(550,400,83,100);
reset.setBounds(670,400,83,100);
select.setBounds(30,20,200,100);

close.setBorder(BorderFactory.createEmptyBorder(4, 4, 4, 4));
close.setContentAreaFilled(false);

car.setBorder(BorderFactory.createEmptyBorder(4, 4, 4, 4));
car.setContentAreaFilled(false);

bike.setBorder(BorderFactory.createEmptyBorder(4, 4, 4, 4));
bike.setContentAreaFilled(false);

riksha.setBorder(BorderFactory.createEmptyBorder(4, 4, 4, 4));
riksha.setContentAreaFilled(false);

bus.setBorder(BorderFactory.createEmptyBorder(4, 4, 4, 4));
bus.setContentAreaFilled(false);

records.setBorder(BorderFactory.createEmptyBorder(4, 4, 4, 4));
records.setContentAreaFilled(false);
```

```
departure.setBorder(BorderFactory.createEmptyBorder(4, 4, 4, 4));
departure.setContentAreaFilled(false);

about.setBorder(BorderFactory.createEmptyBorder(4, 4, 4, 4));
about.setContentAreaFilled(false);

reset.setBorder(BorderFactory.createEmptyBorder(4, 4, 4, 4));
reset.setContentAreaFilled(false);

add(car);
car.addActionListener(this);
add(bus);
bus.addActionListener(this);
add(riksha);
riksha.addActionListener(this);
add(bike);
bike.addActionListener(this);
add(close);
close.addActionListener(this);
add(records);
records.addActionListener(this);
add(select);
add(departure);
departure.addActionListener(this);
add(about);
about.addActionListener(this);
add(reset);
reset.addActionListener(this);

add(lb);
```

```
                setVisible(true);
        }
@Override
        public void actionPerformed(ActionEvent a)
        {
                JButton obj = (JButton)a.getSource();
                String str = obj.getText();
                if(str.equals("Car"))
                {
                        this.setVisible(false);
                        new park(str);

                }
                else if(str.equals("Bike"))
                {
                        this.setVisible(false);
                        new park(str);
                }
                else if(str.equals("Riksha"))
                {
                        this.setVisible(false);
                        new park(str);
                }
                else if(str.equals("Bus"))
                {
                        this.setVisible(false);
                        new park(str);
                }
                else if(str.equals("Records"))
                {
                        new Records();
```

```
                }

                else if(str.equals("Departure"))
                {
                        new Departure("Departure");
                        this.setVisible(false);
                }
                else if(str.equals("Reset"))
                {
                        new Reset();
                        this.setVisible(false);
                        new Screen("Main");
                }
                else if(str.equals("About"))
                {
                        this.setVisible(false);
                        new About("About");
                }
                else if(str.equals("Close"))
                        System.exit(0);


        }
}
```

Parking Window:

```
package com.bvp;

import com.bvp.VehicleBean;
```

```
class park extends JFrame implements ActionListener{
	static JTextField txt1;
	static JTextField txt2;
	static JTextField txt3;
	static JTextField txt4;
	static JLabel lebel1;
	static JLabel lebel2;
	static JLabel lebel3;
	static JLabel lebel4;
	String name;
	String number;
	String licence_no;
	String mobile;
	static String vehicle;
	static int	fee=0;
	static JButton btn1;
	static JLabel veh;
	static JButton back;
	private VehicleBean vehicleBean;


	park(String title){

		super(title);
		park.vehicle = title;
		if (title.equals("Bike"))
		{
			ImageIcon pVeh = new ImageIcon("resources/bike.png");
			veh = new JLabel(pVeh);

			park.fee = 20;
```

```
}
else if (title.equals("Car"))
{
        ImageIcon pVeh = new ImageIcon("resources/car.png");
        veh = new JLabel(pVeh);

        park.fee = 50;
}
else if (title.equals("Riksha"))
{
        ImageIcon pVeh = new
        ImageIcon("resources/riksha.png");
        veh = new JLabel(pVeh);

        park.fee = 40;
}
else if (title.equals("Bus"))
{
        ImageIcon pVeh = new ImageIcon("resources/bus.png");
        veh = new JLabel(pVeh);

        park.fee = 80;
}
veh.setBounds(130,80,110,100);
add(veh);


ImageIcon icon = new ImageIcon("resources/icon.png");
setIconImage(icon.getImage());

setSize(800,600);
```

```
setLayout(null);
setUndecorated(true);
setLocationRelativeTo(null);
setDefaultCloseOperation(EXIT_ON_CLOSE);

lebel1 = new JLabel ("Enter Driver Name");
lebel1.setBounds(150,210,200,30);
lebel1.setForeground (Color.white);

txt1 = new JTextField();
txt1.setBounds(305,210,200,30);

lebel2 = new JLabel ("Enter Vehicle Reg.Number");
lebel2.setBounds(150,260,200,30);
lebel2.setForeground (Color.white);

txt2 = new JTextField();
txt2.setBounds(305,260,200,30);

lebel3 = new JLabel ("Enter licence_no");
lebel3.setBounds(150,310,200,30);
lebel3.setForeground (Color.white);

txt3 = new JTextField();
txt3.setBounds(305,310,200,30);

lebel4 = new JLabel ("Enter Mobile Number");
lebel4.setBounds(150,360,200,30);
lebel4.setForeground (Color.white);

txt4 = new JTextField();
```

```
txt4.setBounds(305,360,200,30);

JLabel background;

ImageIcon pback = new ImageIcon("resources/back.png");
back = new JButton("back",pback);
back.setBounds          (20,20,70,60);
back.setBorder(BorderFactory.createEmptyBorder(4, 4, 4, 4));
back.setContentAreaFilled(false);

ImageIcon img = new ImageIcon("resources/screen.jpg");
background = new JLabel("",img,SwingConstants.CENTER);
background.setBounds(0,0,800,600);
btn1 = new JButton("Park It");
btn1.setBounds(305,410,200,30);
btn1.setBorderPainted(false);
btn1.setContentAreaFilled(false);
btn1.setOpaque(true);

back.addActionListener(this);
btn1.addActionListener(this);


add(btn1);
add(txt1);
add(txt2);
add(txt3);
add(txt4);
add(lebel1);
add(lebel2);
add(lebel3);
```

```
        add(lebel4);
        add(back);
        add(background);
        setVisible(true);

}

@Override
public void actionPerformed(ActionEvent ae)
{
        name = park.txt1.getText();
        number = park.txt2.getText();
        licence_no = park.txt3.getText();
        mobile = park.txt4.getText();
        JButton obj = (JButton)ae.getSource();
        String im = obj.getText();
        if(im.equals("Park It")){
                if (txt1.getText().equals("") || txt2.getText().equals("") ||
                txt3.getText().equals("")|| txt4.getText().equals(""))
                {
                        JOptionPane.showMessageDialog(this,"All fields must be
                        filled");
                }
                else
                {
                        database();


                        JOptionPane.showMessageDialog(this,"Successfully
                        Parked");
                        this.setVisible(false);
```

```
                    new Screen("Main");
                }
            }
            else if(im.equals("back")){
                this.setVisible(false);
                new Screen("Main");
            }

        }
        public void database(){
            try{
                String date = new SimpleDateFormat("dd-MM-yyyy
                HH:mm:ss").format(Calendar.getInstance().getTime());
                vehicleBean = new VehicleBean(park.txt1.getText(), park.vehicle,
                park.txt4.getText(), park.txt3.getText(), park.txt2.getText(),
                ""+park.fee, date);
                VehicleDatabase.save(vehicleBean);
                }
                catch(Exception ae){
                    ae.printStackTrace();
                }
        }
}
```

Hibernate Coding:

```
package com.bvp;

import org.hibernate.*;
import org.hibernate.cfg.*;
```

```
import com.bvp.VehicleBean;

public class VehicleDatabase {

	static Session s;

	static {
		try {
			Configuration cfg = new Configuration().configure();

			s = cfg.buildSessionFactory().openSession();

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

	public static void save(VehicleBean vehicleBean) {

		s.beginTransaction();
		s.save(vehicleBean);
		s.getTransaction().commit();
	}
	public void reset() {

	}

	@Override
	protected void finalize() throws Throwable {
		s.close();
	}
```

}

Hibernate Configuration XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration SYSTEM
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd" PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN">
<hibernate-configuration><session-factory><property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property><propert
y name="hibernate.connection.url">jdbc:mysql://127.0.0.1:3306/demo</
property><property
name="hibernate.connection.username">root</property><property
name="hibernate.connection.password">root</property><property
name="hibernate.dialect">org.hibernate.dialect.MySQL5InnoDBDialect</
property><property name="hbm2ddl.auto">update</property><property
name="show_sql">true</property><mapping
class="com.bvp.VehicleBean"/></session-factory></hibernate-configuration>
```

3.3 SCREEN DESIGN

1). Welcome Screen:

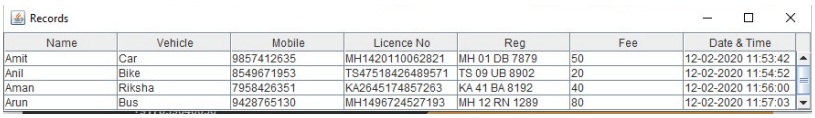
2). Login Screen:

3). Main Screen:

4). Parking Window:

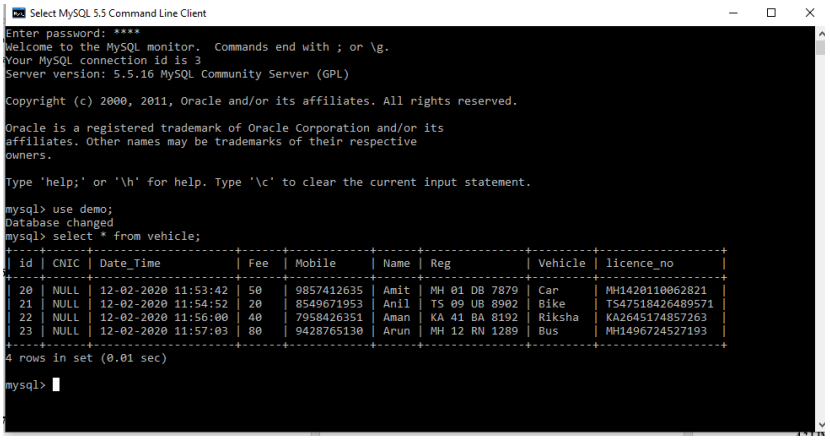
5). Departure Window:

6). Records:



7). About:

8). Database Table



CHAPTER 4:

SYSTEM TESTING & IMPLEMENTATION

4.1 TESTING

- All software intended for public consumption should receive some level of testing. Without testing, you have no assurance that software will behave as expected. The results in public environment can be truly embarrassing.
- Testing is a critical element of software quality assurance and represents the ultimate review of specification, designing, and coding. Testing is done throughout the system development at various stages. If this is not done, then the poorly tested system can fail after installation. Testing is a very important part of SDLC and takes approximately 50%of the time.
- The first step in testing is developing a test plan based on the product requirements. The test plan is usually a formal document that ensures that the product meets the following standards:

➢ Is thoroughly Tested- Untested code adds an unknown element to the product and increases the risk of product failure.

➢ Meets product requirements- To meet customer needs, the product must provide the features and behavior described in the product specification.

➢ Does not contain defects- Features must work within established quality standards and those standards should be clearly stated within the test plan.

Testing Techniques

4.1.1 Black box Testing

It aims to test a given program's behavior against its specification or component without making any reference to the internal structures of the program or the algorithms used. Therefore, the source code is not needed, and so even purchased modules can be tested. We study the system by

examining its inputs and related outputs. The key is to devise inputs that have a higher likelihood of causing outputs that reveal the presence of defects. We use experience and knowledge of the domain to identify such test cases. Failing this a systematic approach may be necessary. Equivalence partitioning is where the input to a program falls into a number of classes. e.g. positive numbers vs. negative numbers. Programs normally behave the same way for each member of a class. Partitions exist for both input and output. Partitions may be discrete or overlap. Invalid data (i.e. outside the normal partitions) is one for which partitions should be tested. Test cases are chosen to exercise each portion. Also test boundary cases (atypical, extreme, zero) should be considered since these frequently show up defects. For completeness, test all combinations of partitions. Black box testing is rarely exhaustive (because one doesn't test every value in an equivalence partition) and sometimes fails to reveal corruption defects caused by weird combination of inputs. Black box testing should not be used to try and reveal corruption defects caused, Example, by assigning a pointer to point to an object of the wrong type. Static inspection (or using a better programming language) is preferred.

4.1.2 White box Testing

It was used as an important primary testing approach. Code is tested using code scripts, drivers, stubs, etc. which are employed to directly interface with it and drive the code. The tester can analyze the code and use the knowledge about the structure of a component to derive test data. This testing is based on the knowledge of structure of component (e.g. by looking at source code). The advantage is that structure of code can be used to find out how many test cases needed to be performed. Knowledge of the algorithm (examination of the code) can be used to identify the equivalence partitions. Path testing is where the tester aims to exercise every independent execution path through the component. All conditional statements tested for both true and false cases. If a unit has no control

statements, there will be up to 2n possible paths through it. This demonstrates that it is much easier to test small program units than large ones. Flow graphs are a pictorial representation of the paths of control through a program (ignoring assignments, procedure calls and I/O statements). We use a flow graph to design test cases that execute each path. Static tools may be used to make this easier in programs that have a complex branching structure. Dynamic program analyzers instrument a program with additional code. Typically, this will count how many times each statement is executed. At end, print out report showing which statements have and have not been executed.

Possible methods:

- Usual method is to ensure that every line of code is executed at least once.
- Test capabilities rather than components (e.g. concentrate on tests for data loss over ones for screen layout).
- Test old in preference to new (users less affected by failure of new capabilities).
- Test typical cases rather than boundary ones (ensure normal operation works properly).

Debugging: Debugging is a cycle of detection, location, repair and test. Debugging is a hypothesis testing process. When a bug is detected, the tester must form a hypothesis about the cause and location of the bug. Further examination of the execution of the program (possible including many returns of it) will usually take place to confirm the hypothesis. If the hypothesis is demonstrated to be incorrect, a new hypothesis must be formed. Debugging tools that show the state of the program are useful for this, but inserting print statements is often the only approach. Experienced debuggers use their knowledge of common and/or obscure bugs to facilitate the hypothesis testing process. After fixing a bug, the system must be reset to ensure that the fix has worked and that no other bugs have

been introduced. In principle, all tests should be performed again but this is often too expensive to do.

DATABASE

Introduction to Hibernate

Introduction

A database is a collection of information that's related. Access allows you to manage your information in one database file.

Introduction to Hibernate

Hibernate ORM is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database.

• Tables store your data in your database
• Queries ask questions about information stored in your tables
• Forms allow you to view data stored in your tables
• Reports allow you to print data based on queries/tables that you have created

CHAPTER 5 –

CONCLUSION AND REFRENCES

5.1 CONCLUSION

The application has been implemented and is working successfully. This is user friendly software that can prove to be immense help in the automated process of parked cars records. This software provides fast working in the field of cars parked.
Computerization of parking facilitates the clients with following features:

1. Ease of work.

2. Search and delete duplicate records.

3. The car's information can be easily accessible.

4. Save lot of time and reduce paper work.

5.2 LIMITATION OF THE PROJECT

Every system has some limitation's, some challenges are faced by every software. It might be difficult for a naïve user (users Who are not aware of the rules and regulations of a particular Language) to understand and use it. For naïve users to use it, it is very important that someone should teach them first which is a very difficult task. So, the simplest way to solve this problem is that the developer should prepare a list of instructions which shows the stepwise procedure of how to use this system.

- Database Driver should be installed in the system to support different database.

- The charges for this application for various vehicles have been hard-coded.
- Pugin should be added for the printing of fee.
- As this program is write in java so to use it in embedded system it should be exported to that service.

5.3 FUTURE SCOPE OF THE PROJECT

- Our project will prove to be very efficient and economical in future.
- Simple to use.
- Easy to understand.
- May it will be adopted by the number of companies because of the above-mentioned features with uniqueness.
- We have kept a scope of up gradation that is to be done according to the changing environment technology and culture.
- Less time consuming and effective.
- User interface is so easy to understand, no job specialization, specification, special training and analysis is required. Anybody can handle it with no efforts.
- The project can be converted into an online system.
- The facility for changing the login Password can be added.

5.4 REFERENCES

- WWW.GOOGLE.COM
- WWW.WIKEPEDIA.COM
- The Complete Reference Java (By Hebert Schilt)
- WWW.UDEMY.COM