PROJECT REPORT

SUBMITTED TO

DEPARTMENT OF COMPUTER SCIENCE

VES COLLEGE OF ARTS, SCIENCE AND COMMERCE

SINDHI SOCIETY, CHEMBUR MUMBAI - 400 071.

VEHICLE SERVICE MANAGEMENT SYSTEM

For Partial Fulfillment for Degree of

Bachelor of Science (Computer Science)

2018-2019

HEAD OF DEPARTMENT

Mr. KAMLAKAR BHOPATKAR

PROJECT GUIDE

Mr. KAMLAKAR BHOPATKAR

Ms. MADHAVI VAIDYA

SUBMITTED BY

BHAVESH SADASHIV GADAG

VIVEKANAND EDUCATION SOCIETY'S COLLEGE OF ARTS, SCIENCE AND COMMERCE
Sindhi Society, Chembur Mumbai 400071.
Phones: 25227514/25227470.
NAAC Re-Accredited 'A' Grade

CERTIFICATE

This is to certify that **Mr. BHAVESH SADASHIV GADAG** of **T.Y.B.Sc. (Computer Science)** affiliated to University of Mumbai has successfully completed a project work entitled.

VEHICLE SERVICE MANAGEMENT SYSTEM

As partial fulfilment of the requirement for the degree of **B.Sc. (Computer Science)** for the academic year 2018-2019.

CO-ORDINATOR OF DEPARTMENT
Mr. Kamlakar Bhopatkar

Date:

PROJECT GUIDE
Mr. Kamlakar Bhopatkar
Ms. Madhavi Vaidya

Date:

EXAMINER

Date:

College Seal

ACKNOWLEDGMENT

I have great pleasure in presenting this project entitled "**Vehicle Service Management System**" and I grab this opportunity to convey my immense regards towards all the people who with their invaluable contributions made this project successful.

It gives me great pleasure in presenting this project report. Its justification will never complete if I don't express my vote of thanks to our **V.E.S. College** and Principal **Dr. Mrs. J.K. Phadnis**.

I sincerely thank and express my profound gratitude to our Project Guides Mr. **Kamlakar Bhopatkar** and Ms. **Madhavi Vaidya** for timely and prestigious guidance required for the project completion at each phase of the project development.

I also owe to my friends who have been a constant source of help to solve the problems that cropped up during the development of the project, positive criticism, suggestions, constant support, encouragement and the guidance force towards the successful completion of the project.

INDEX

DESCRIPTION OF CURRENT SYSTEM

The current system that is being used mostly consists of manual paperwork. This system is time consuming as the manager has to do the bills manually and store it in the traditional file system. It can be a tedious job to handle and which can be very time consuming and also lead to a lot of human errors while storing the data. But now-a-day's computerization has made it easy to work by replacing manual work. By replacing the existing system with a computerized system, one can:-

1. Reduce the burden of paperwork.
2. Save time of management for recording details of customers and cars.
3. Generate required reports easily.

LIMITATION OF CURRENT SYSTEM

- Time consuming

  As the records are to be manually maintained it consumes a lot of time.

- Store requirements:

  As traditional files and registers are used to store data, storage space requirement is increased.

- Paperwork:

  Lot of paperwork is involved as the records are maintained in the files and registers.

- Less reliable:

  Use of papers for storing data is not reliable.

- Accuracy:

  As the system is stored manually, there are many chances of human errors, which lead to errors in calculating and storing incorrect data.

PROPOSED SYSTEM

The proposed Car Service Center management system is developed for maintaining the service center activities like, car maintenance, customer bill generation, car service and spare parts sales, employee details, to improve accuracy and enhance the efficiency of service center to better serve the customer. This system helps the end-user to maintain record of car and it's servicing. This system helps to maintain detailed information of customers. Through this system we maintain the details of customers who bring their car in the service center. To keep track of previous services performed on the car by their service center, keep track of spare parts used in service, generate bill for customer. Store records of employees and assign a service-adviser to car. Search for available spare parts in the stockroom. This system helps the user to search for an entry which is related to customer and easily get data with description of model. Through this system we can easily maintain customer record and servicing details of all customer with time period. And we efficiently maintain servicing records. In this system we use Front-end as Python and back-end as MySQL.

Objectives:

- Data Management

  The management can store and maintain data easily.
- Data Retrieval:

  The data can be retrieved very easily through this application.
- Data Modification:

  The data of any customer can be modified when needed.
- Reducing work time:

  Data can be retrieved quickly, thus saving time.

ADVANTAGES OF PROPOSED SYSTEM

- Car Service Center Management system provides the searching facilities based on service, car, customer information and bill.
- Generates bill with error-free and accurate cost calculation.
- It will help in maintaining the information of every customer who visits the service center.
- It will be easy to retrieve and modify any customer and car data.
- The main purpose of Car Service Management system is to help any organization to maintain and manage cars data.
- To efficiently keep track of spare parts in stock and order the parts if not available.

TECHNOLOGIES USED

Python 3.6.6 :

Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development. Python is relatively simple, so it's easy to learn since it requires a unique syntax that focuses on readability. Developers can read and translate Python code much easier than other languages. In turn, this reduces the cost of program maintenance and development because it allows teams to work collaboratively without significant language and experience barriers.

PyQt :

PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in. PyQt is free software developed by the British firm Riverbank Computing. It is available under similar terms to Qt versions older than 4.5; this means a variety of licenses including GNU General Public License (GPL) and commercial license, but not the GNU Lesser General Public License (LGPL).

MySQL :

MySQL is an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL runs on virtually all platforms, including Linux, UNIX and Windows. Although it can be used in a wide range of applications. MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses.

Features:

- **Relational Database System**: Like almost all other database systems on the market, MySQL is a relational database system.
- **SQL compatibility**: MySQL supports as its database language -- as its name suggests – SQL (Structured Query Language). SQL is a standardized language for querying and updating data and for the administration of a database.
- **Allows roll-back**: MySQL allows transactions to be rolled back, commit and crash recovery.

- **Scalable**: MySQL can handle almost any amount of data, up to as much as 50 million rows or more.
- **High Performance**: MySQL is faster, more reliable and cheaper because of its unique storage engine architecture.

ReportLab

This is a software library that lets you directly create documents in Adobe's Portable Document Format (PDF) using the Python programming language. It also creates charts and data graphics in various bitmap and vector formats as well as PDF. The ReportLab library directly creates PDF based on your graphics commands. There are no intervening steps. Your applications can generate reports extremely fast – sometimes orders of magnitude faster than traditional report-writing tools.

BASIC SCENARIO

- Customer gives order
- Add customer details
- Update customer details
- Delete customer
- Add Vehicle details
- Update Vehicle details
- Delete Vehicle
- Add Part details
- Update Part details
- Delete Part
- Add Employee details
- Update Employee details
- Delete Employee
- User analyses the order
- Users checks the stock
- User fulfils the order
- Generate bill

USE-CASE DIAGRAM

A use case diagram depicts the various operations that a system performs. It contains use cases, actors, and their relationships. Use cases are a sequence of actions that form a single unit of work for an actor. An actor represents a user who is external to the system and interacts with the use case.

Elements of Use Case Diagram

Actors

An **actor** portrays any entity (or entities) that perform certain roles in a given system. The different roles the actor represents are the actual business roles of users in a given system. An actor in a use case diagram interacts with a use case. For example, for modelling a banking application, a customer entity represents an actor in the application. Similarly, the person who provides service at the counter is also an actor. But it is up to you to consider what actors make an impact on the functionality that you want to model. If an entity does not affect a certain piece of functionality that you are modelling, it makes no sense to represent it as an actor. An actor is shown as a stick figure in a use case diagram depicted "outside" the system boundary.

Use Cases

A use case in a use case diagram is a visual representation of distinct business functionality in a system. The key term here is "distinct business functionality." To choose a business process as a likely candidate for modelling as a use case, you need to ensure that the business process is discrete in nature. As the first step in identifying use cases, you should list the discrete business functions in your problem statement. Each of these business functions can be classified as a potential use case. Remember that identifying use cases is a discovery rather than a creation. As business functionality becomes clearer, the underlying use cases become more easily evident. A use case is shown as an ellipse in a use case diagram.

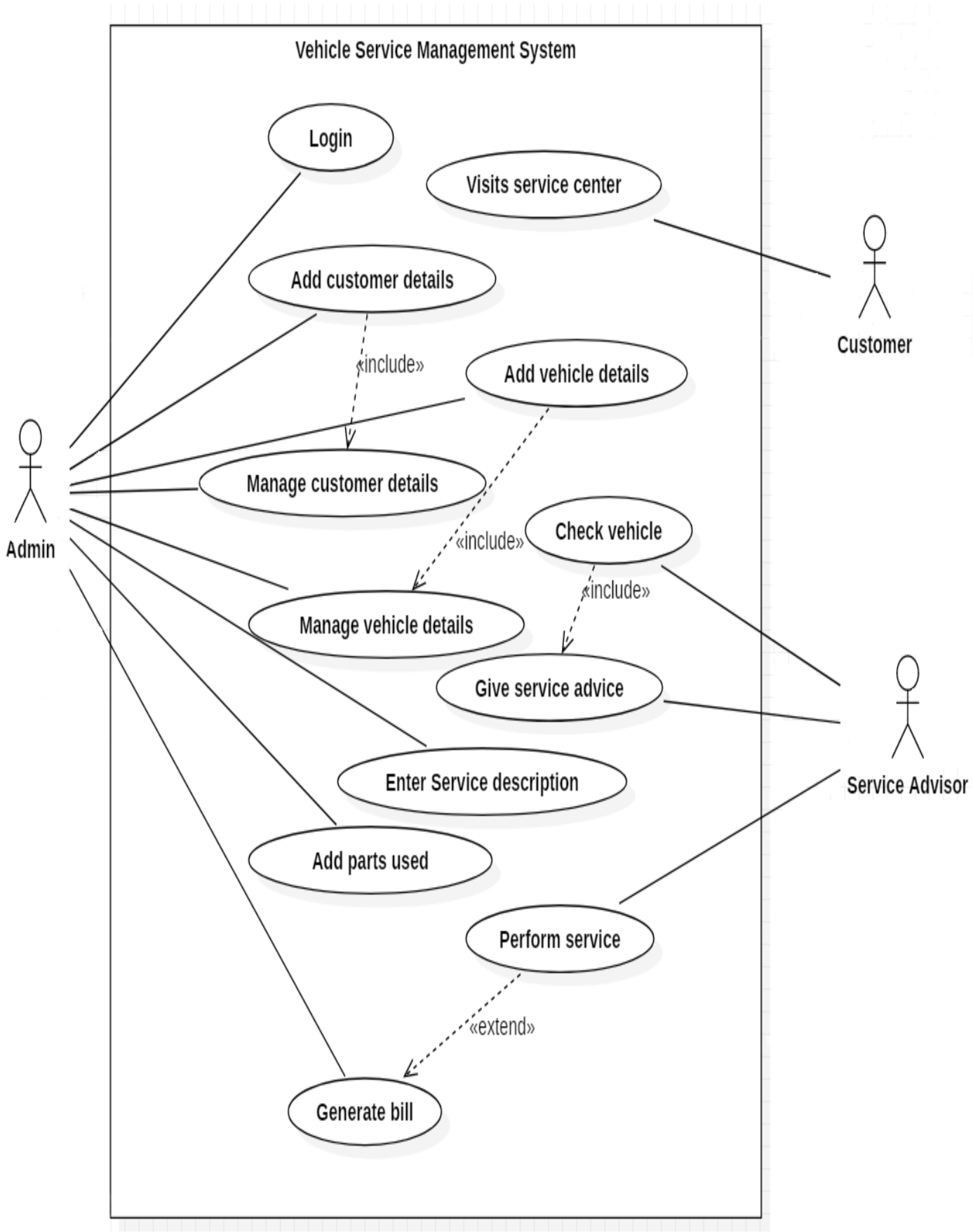
System Boundary

A system boundary defines the scope of what a system will be. A system cannot have infinite functionality. So, it follows that use cases also need to have definitive limits defined. A system boundary of a use case diagram defines the limits of the system. The system boundary is shown as a rectangle spanning all the use cases in the system.

Relationships: The following relationships can be established amongst use cases

- **Extends:** A use case may extend another. This relationship indicates that the behaviour of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label “«extend»".
- **Includes:** A use case may include another. Include is a Directed Relationship between two use cases, implying that the behaviour of the included use case is inserted into the behaviour of the including use case. The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviour from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»".

USE-CASE DIAGRAM



ENTITY-RELATIONSHIP DIAGRAM

Data models are tools used in analysis to describe the data requirement and assumptions in the system from a top-down perspective. They also set the stage for design of databases later on in the SDLC.

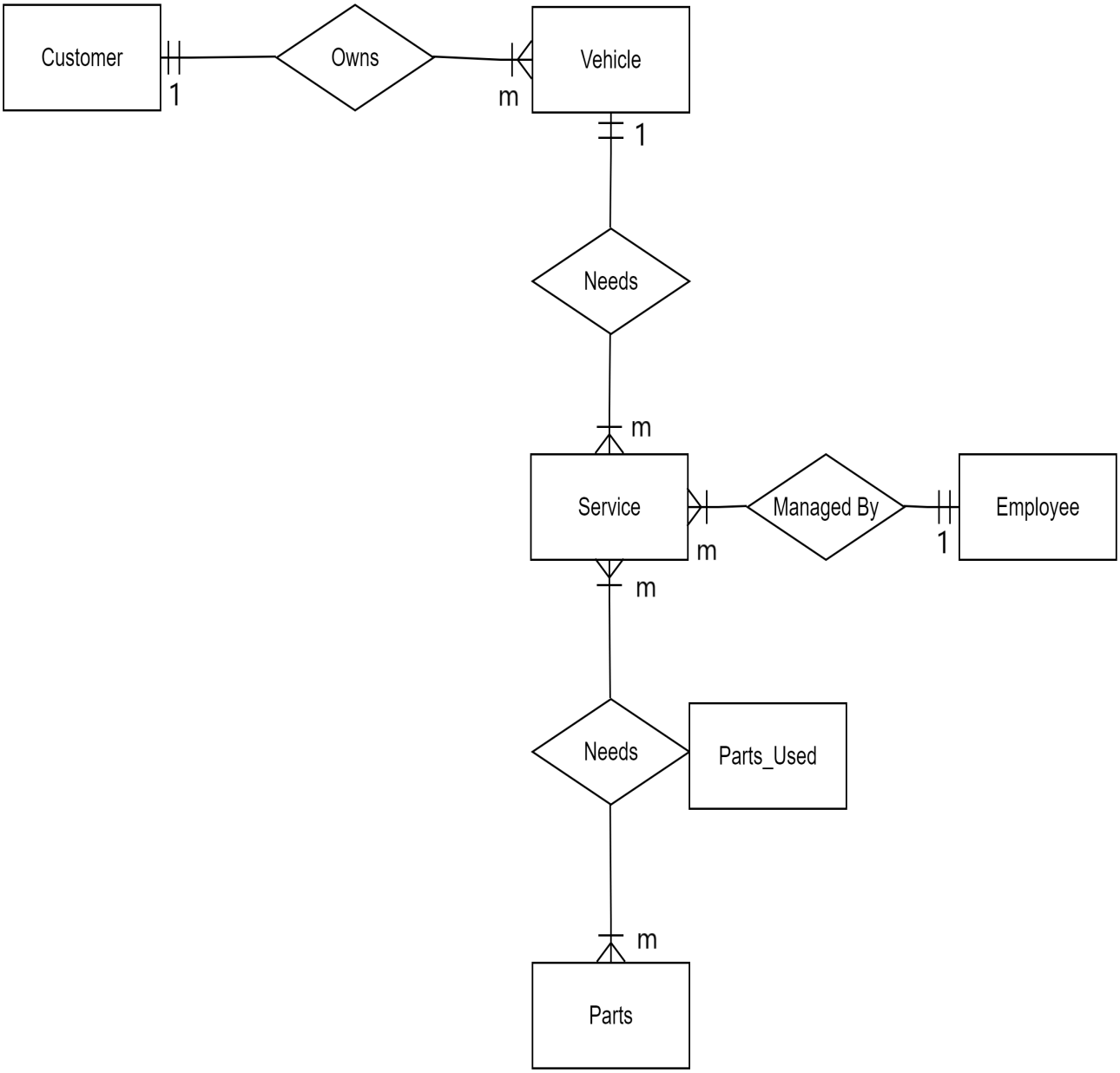
There are three basic elements in ER model:
Entities are the "things" about which we seek information.
Attributes are the data we collect about the entities.
Relationships provide the structure needed to draw information from multiple entities.
Entity: It represents a collection of objects o things in the real world whose individual members or instances have the following characteristics:

Each can be identified uniquely in some fashion.
Each plays a necessary role in the system we are building.
Each can be described by one or more data elements.
Attributes: They express the properties of entities. Attributes having unique values are called candidate keys (Primary key).
Relationships: They describe the association between entities. They are characterized by cardinality as follows:

- **One-to-Many** relationship means that one instance of the first entity is related to many instance of second entity, while an instance of second entity is associated with only instance of the first entity.

- **Many-to-Many** means that an instance of the first entity is related to many instances of the second entity and the same is true in the reverse direction also.

ENTITY-RELATIONSHIP DIAGRAM



LIST OF TABLES

Customer:

Column Name	Data Type	Size	Constraints
customer_id	INT		PRIMARY KEY
customer_name	VARCHAR	45	NOT NULL
address	VARCHAR	100	
contact_no	VARCHAR	10	UNIQUE
email	VARCHAR	80	UNIQUE

Vehicle:

Column Name	Data Type	Size	Constraints
vehicle_id	INT		PRIMARY KEY
registration_name	VARCHAR	10	UNIQUE
company	VARCHAR	45	NOT NULL
model	VARCHAR	45	
vehicle_type	VARCHAR	20	NOT NULL
transmission	ENUM		NOT NULL
customer_id	INT		FOREIGN KEY

Employee:

Column Name	Data Type	Size	Constraints
employee_id	INT		PRIMARY KEY
employee_name	VARCHAR	45	NOT NULL
employee_address	VARCHAR	150	
employee_contact_no	VARCHAR	10	UNIQUE
date_of_joining	DATE		NOT NULL
salary	INT		

Service:

Column Name	Data Type	Size	Constraints
service_id	INT		PRIMARY KEY
description	TINYTEXT		
service_date	DATE		NOT NULL
distance	INT	10	
damages	VARCHAR	45	
total_price	FLOAT		NOT NULL
vehicle_id	INT		FOREIGN KEY
employee_id	INT		FOREIGN KEY

Parts:

Column Name	Data Type	Size	Constraints
part_id	INT		PRIMARY KEY
part_name	VARCHAR	60	NOT NULL
brand	VARCHAR	45	
part_price	FLOAT		NOT NULL

Parts_Used:

Column Name	Data Type	Size	Constraints
Service_service_id	INT		FOREIGN KEY
Parts_part_id	INT		FOREIGN KEY
quantity	TINYINT	3	

ACTIVITY DIAGRAM

Activities are a representation of the various operations performed by a class. An activity diagram depicts the flow of control from one activity to another. An activity diagram uses activities to model objects, classes, interfaces, components and nodes. An activity represents a set of actions such as call to a method of class, send or receive a signal, create or destroy an object, and evaluate an expression.

The basic elements of activity diagram are

Action state: Represents the state of the system in terms of actions

Activity state: Represents an activity and therefore this state can further expand into another activity state or action state.

Transition: Represents the control flow that performs a particular operation

Decision: Represents the if-else or branch condition that decides the path of control flow

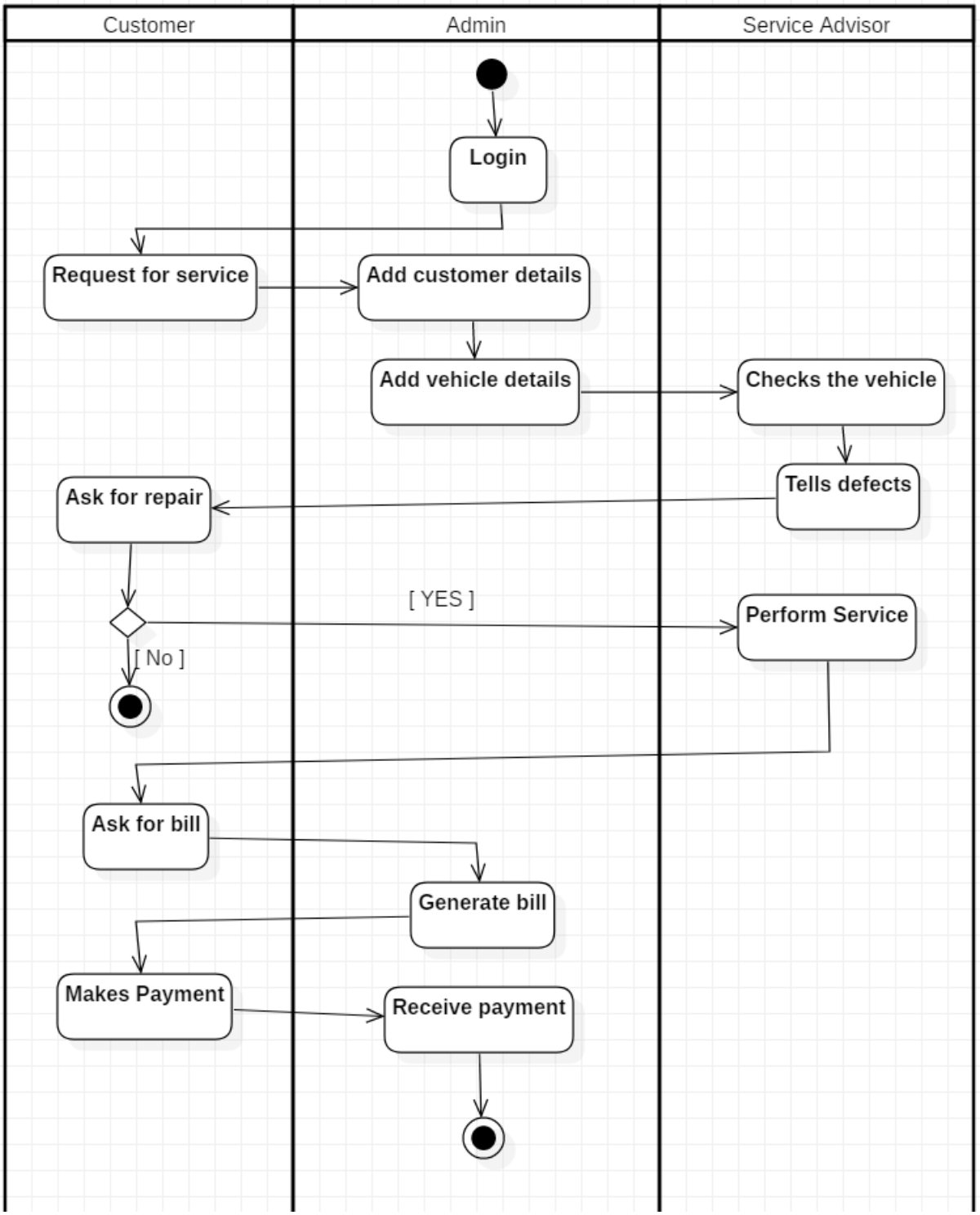
Initial State: A Filled circle followed by an arrow represents the initial action state.

Final State: An arrow pointing to a filled circle nested inside another circle represents the final action state.

Synchronization: A synchronization bar helps illustrate parallel transition. It is also called fork and joining.

Swim lanes: Swim lanes group related activities into one column.

ACTIVITY DIAGRAM



CLASS DIAGRAM

A class diagram consists of a group of classes and interfaces reflecting important entities of the system, and the relationships between these classes and interfaces. Classes in a class diagram are interconnected in a hierarchical fashion. Class Diagrams describe the static structure of a system, or how it is structured, rather than how it behaves. A class diagram is represented in 3 parts:

ELEMENTS OF CLASS DIAGRAM:

Class: A class represents an entity of a given system that provides certain functionality of a given entity. The properties of a class are called as attributes.

A class is represented by a rectangle, which is divided into compartments. These contain

- The upper part holds the name of the class.
- The middle part contains the attributes of the class.
- The bottom part gives the methods or operations the class can take or undertake

Association: Association represents the static relationship shared among the objects of two classes. Associations are normally represented as a line, with each end connected to a class.

Aggregation: Aggregation is a variant of the “has a” or association relationship; aggregation is more specific than association. It is an association that represents a part whole or part of relationship.

Composition: Composition is a stronger variant of the “owns a” or association relationship; composition is more specific than aggregation. It is represented with a solid diamond shape.

Generalization: The Generalization relationship indicates that one of the two related classes (the subtype) is considered to be a specialized form of the other and super type is considered as Generalization of subtype.

Multiplicity: Multiplicity notations indicate the number of instances of one class linked to one instance of the other class. They are placed near the end of an association.

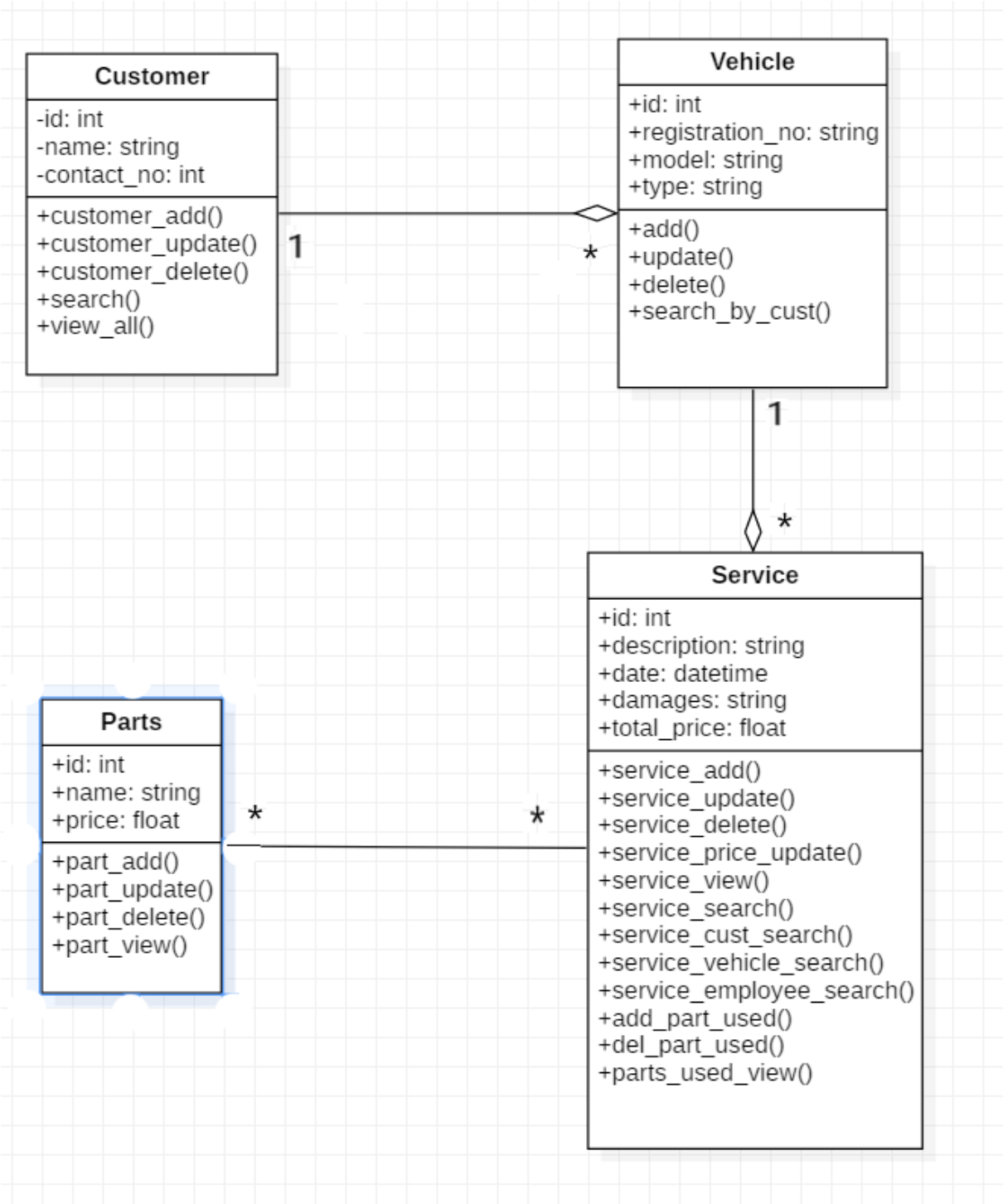
0..1 - zero or one instance

1 - exactly one instance

0..* or * - zero or more instances

1..* - one or many instances (at least one)

CLASS DIAGRAM



OBJECT DIAGRAM

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.

Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

Object diagrams are used to render a set of objects and their relationships as an instance.

The difference is that a class diagram represents an abstract model consists of classes and their relationships. But an object diagram represents an instance at a particular moment which is concrete in nature.

It means the object diagram is more close to the actual system behaviour. The purpose is to capture the static view of a system at a particular moment.

So the purpose of the object diagram can be summarized as:

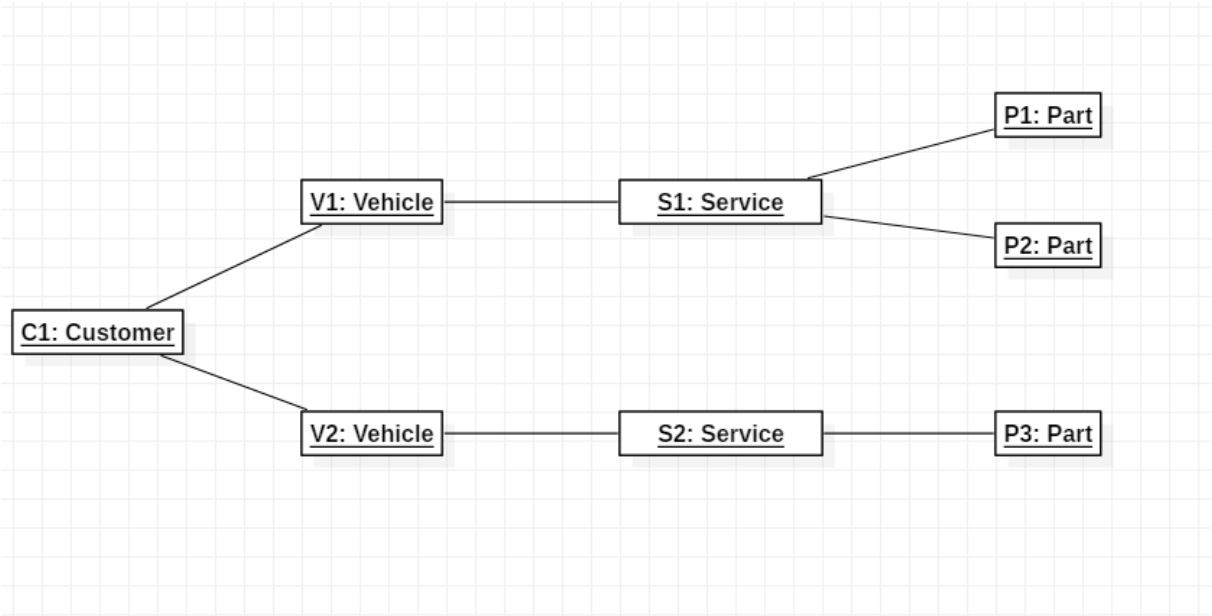
- Forward and reverse engineering.
- Object relationships of a system
- Static view of an interaction.
- Understand object behaviour and their relationship from practical perspective

ELEMENTS OF OBJECT DIAGRAM:

➢ Objects: Objects represent particular entities. These are instances of classes.

➢ Links: Links represent particular relationships between objects. These are instances of Associations. A link is shown as a solid line.

OBJECT DIAGRAM



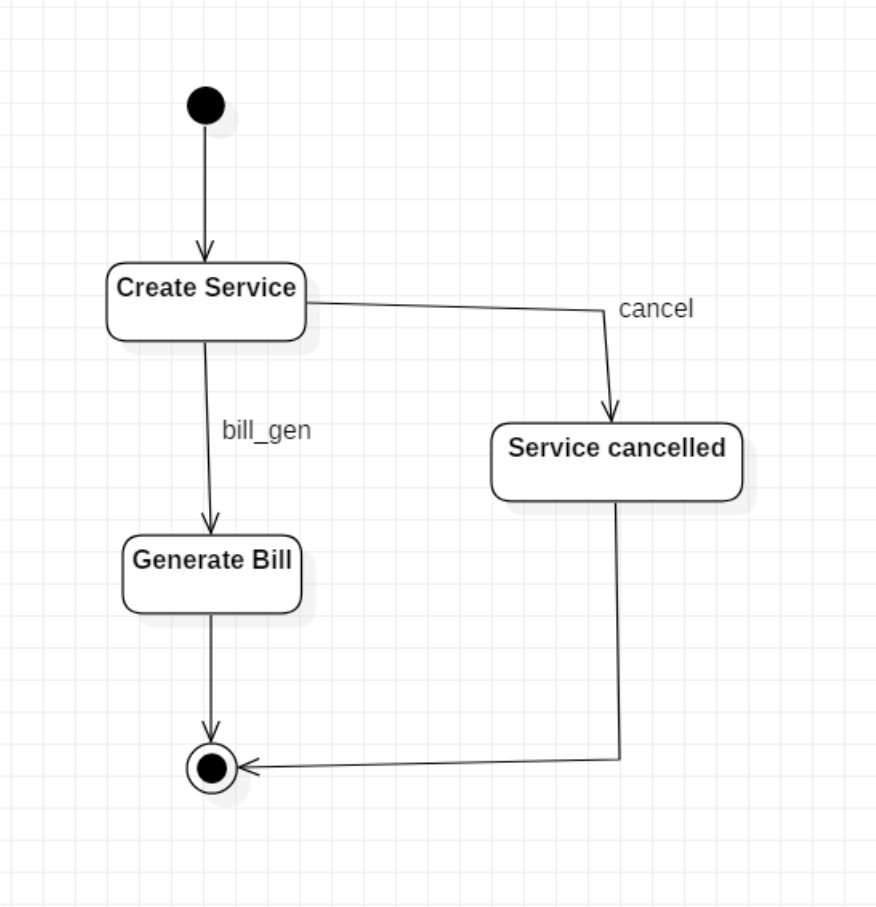
STATE DIAGRAM

A state chart diagram is a behaviour which specifies the sequence of states an object visits during its lifetime in response to events, together with its responses to those events.

ELEMENTS OF A STATE DIAGRAM:

- **Initial State**: This shows the starting point or a first activity of the flow, denoted by a solid circle. This is also called as a "pseudo state," where the state has no variables describing it further and no activities.
- **State**: Represents the state of object at an instant of time. In a state diagram, there will be multiple of such symbols, one for each state of the Object we are discussing. Denoted by a rectangle with rounded corners and compartments (such as a class with rounded corners to denote and Object). We will describe this symbol in detail a little later.
- **Event and Action**: A trigger that causes a transition to occur is called as an event or action. Every transition need not occur due to the occurrence of an event or action directly related to the state that transitioned from one state to another. As described above, an event/action is written above a transition that it causes.
- **Final State**: The end of the state diagram is shown by a bull‟s eye symbol, also called a final state. A final state is another example of a pseudo state because it does not have any variable or action described.

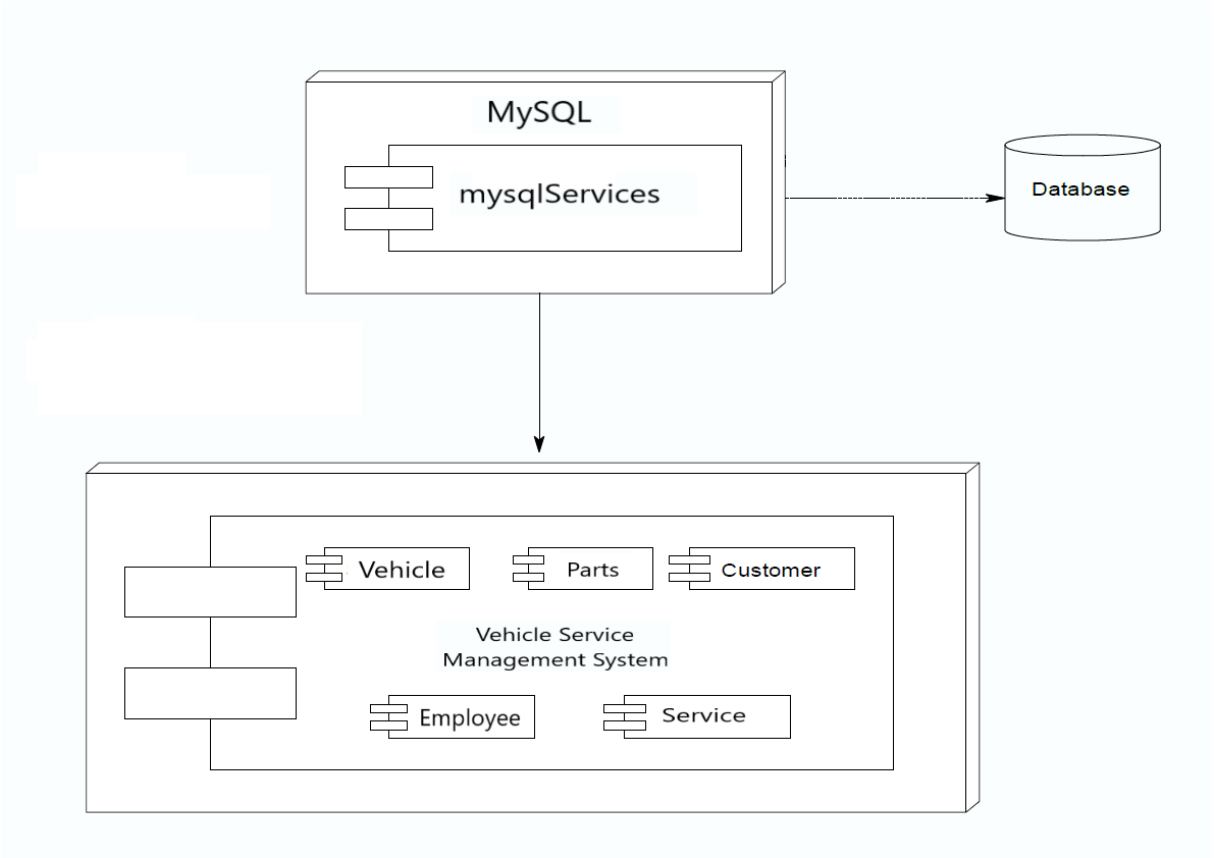
STATE DIAGRAM



COMPONENT DIAGRAM

- Component Diagram provides the realization of a set of interfaces and form the executable parts of a software system. Examples of components can be an executable file, a COM+ component, or an Enterprise Java Bean (EJB) or a .NET component. In UML we represent a component as a rectangle with tabs. Each Component has a name that identifies it.
- A Component is a standalone piece of software because it performs a complete function.
- Relationship among Components, Classes and Interfaces. Component realizes a set of interfaces in which each interface specifies the function offered by a class. A class represents a real world entity and contains the code for implementing behaviour that is specific to the entity that the class represents. Thus various classes contained in a component and their relationship using a component diagram.

COMPONENT DIAGRAM



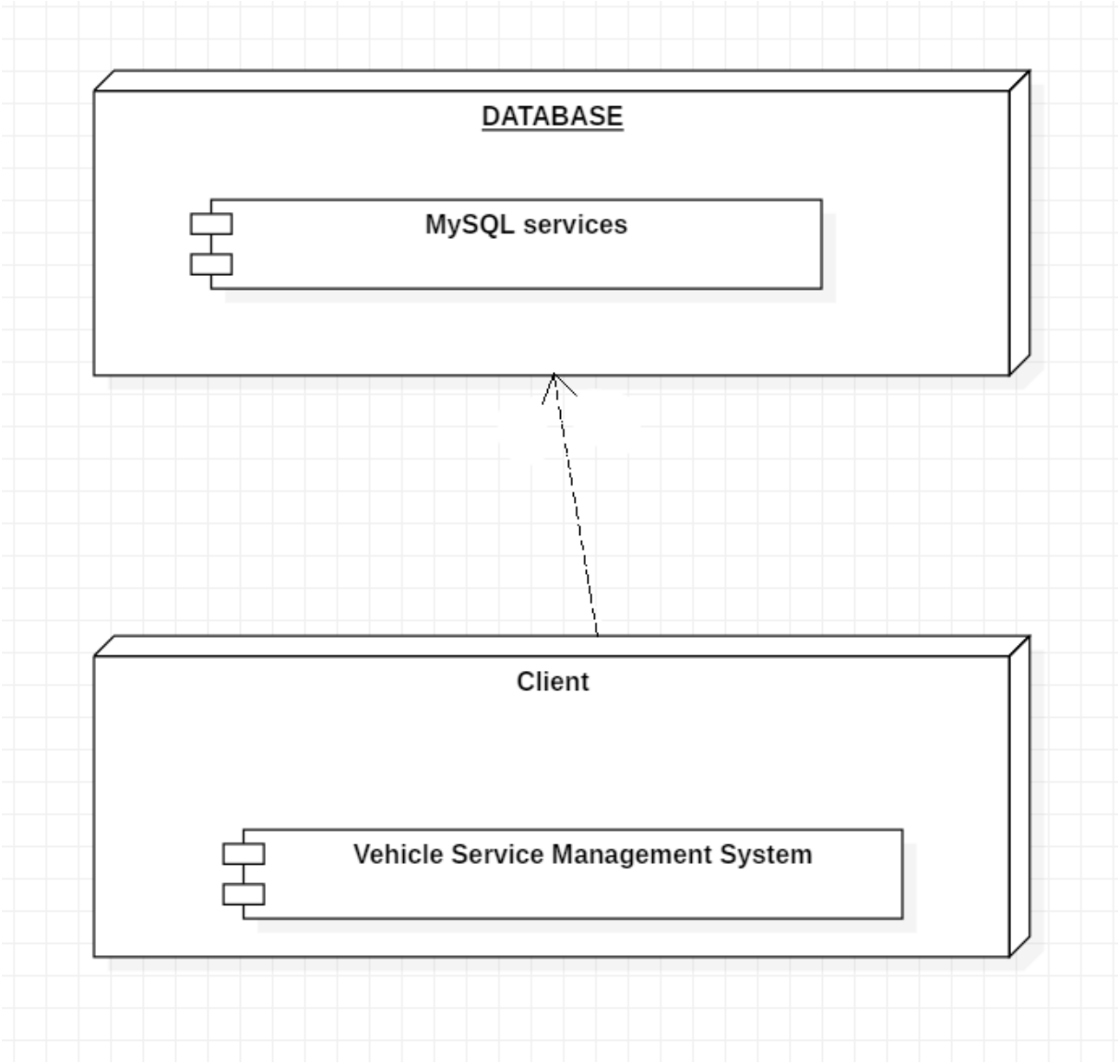
DEPLOYMENT DIAGRAM

➢ The Deployment diagram is drawn to visualize the hardware on which the software components need to be deployed. A deployment diagram is drawn immediately after identifying classes, interfaces and their relationships.

➢ The various computer systems or processing devices on which components are deployed are called nodes. We may have all the components on one node or on different nodes. In UML a node is represented as a 3-D rectangular box containing the components that execute within the node. A node may be located anywhere in network. Distinction among nodes is achieved by assigning each node a name and classifying the various nods into types depending on the type of components they execute. Nodes are associated with each other by a connection, which represents a communication channel. It depicts how the nodes are connected with each other.

➢ The dependency on node on components is depicted using dashed lines. This means that a node uses the services of the components that are executing on another node.

DEPLOYMENT DIAGRAM



SEQUENCE DIAGRAM

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

Sequence Diagrams captures:

- the interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)
- high-level interactions between user of the system and the system, between the system and other
  systems, or between subsystems (sometimes known as system sequence diagrams)

Actor

- a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data)
- external to the subject (i.e., in the sense that an instance of an actor is not a part of the instance of its corresponding subject).

Lifeline

- A lifeline represents an individual participant in the Interaction.

Activations

- A thin rectangle on a lifeline) represents the period during which an element is performing an operation.
- The top and the bottom of the of the rectangle are aligned with the initiation and the completion time respectively.

Call Message

- A message defines a particular communication between Lifelines of an Interaction.
- Call message is a kind of message that represents an invocation of operation of target lifeline.

Return Message

- A message defines a particular communication between Lifelines of an Interaction.
- Return message is a kind of message that represents the pass of information back to the caller of a corresponded former message.

Self Message

- A message defines a particular communication between Lifelines of an Interaction.
- Self message is a kind of message that represents the invocation of message of the same lifeline.

Recursive Message

- A message defines a particular communication between Lifelines of an Interaction.
- Recursive message is a kind of message that represents the invocation of message of the same lifeline.
- It's target points to an activation on top of the activation where the message was invoked from.

Create Message

- A message defines a particular communication between Lifelines of an Interaction.
- Create message is a kind of message that represents the instantiation of (target) lifeline.

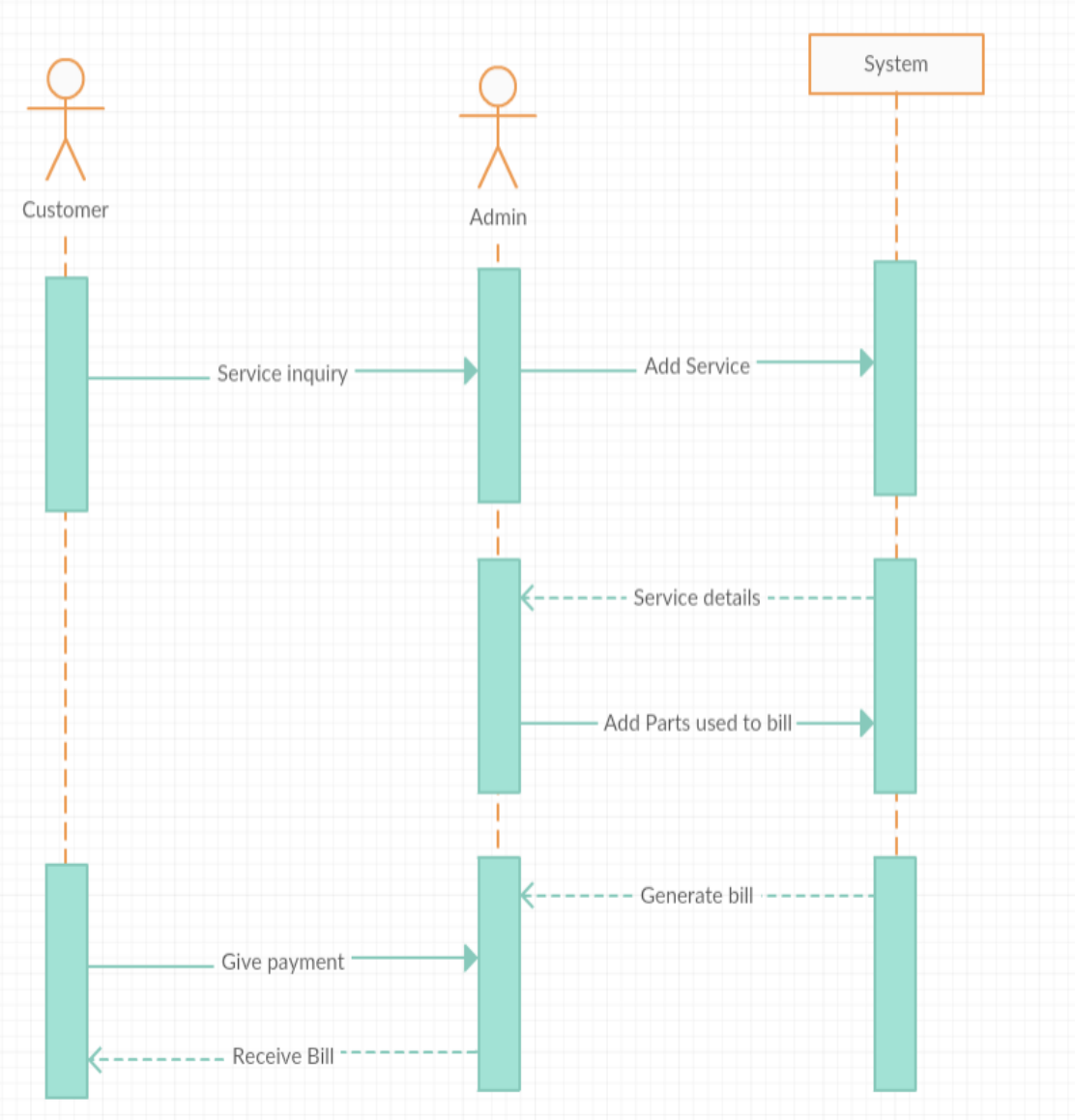
Destroy Message

- A message defines a particular communication between Lifelines of an Interaction.
- Destroy message is a kind of message that represents the request of destroying the lifecycle of target lifeline.

Duration Message

- A message defines a particular communication between Lifelines of an Interaction.
- Duration message shows the distance between two time instants for a message invocation.

SEQUENCE DIAGRAM



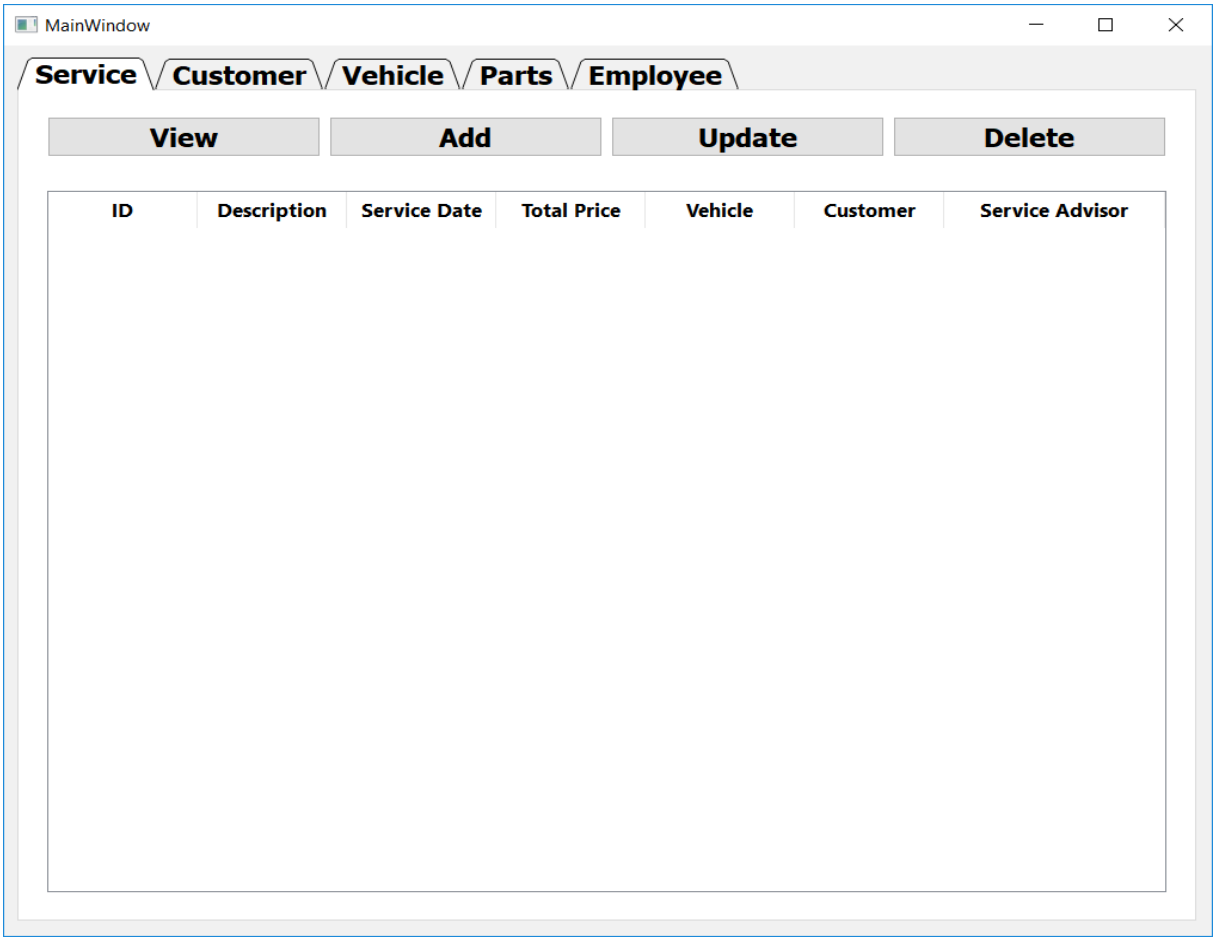
TEST CASES

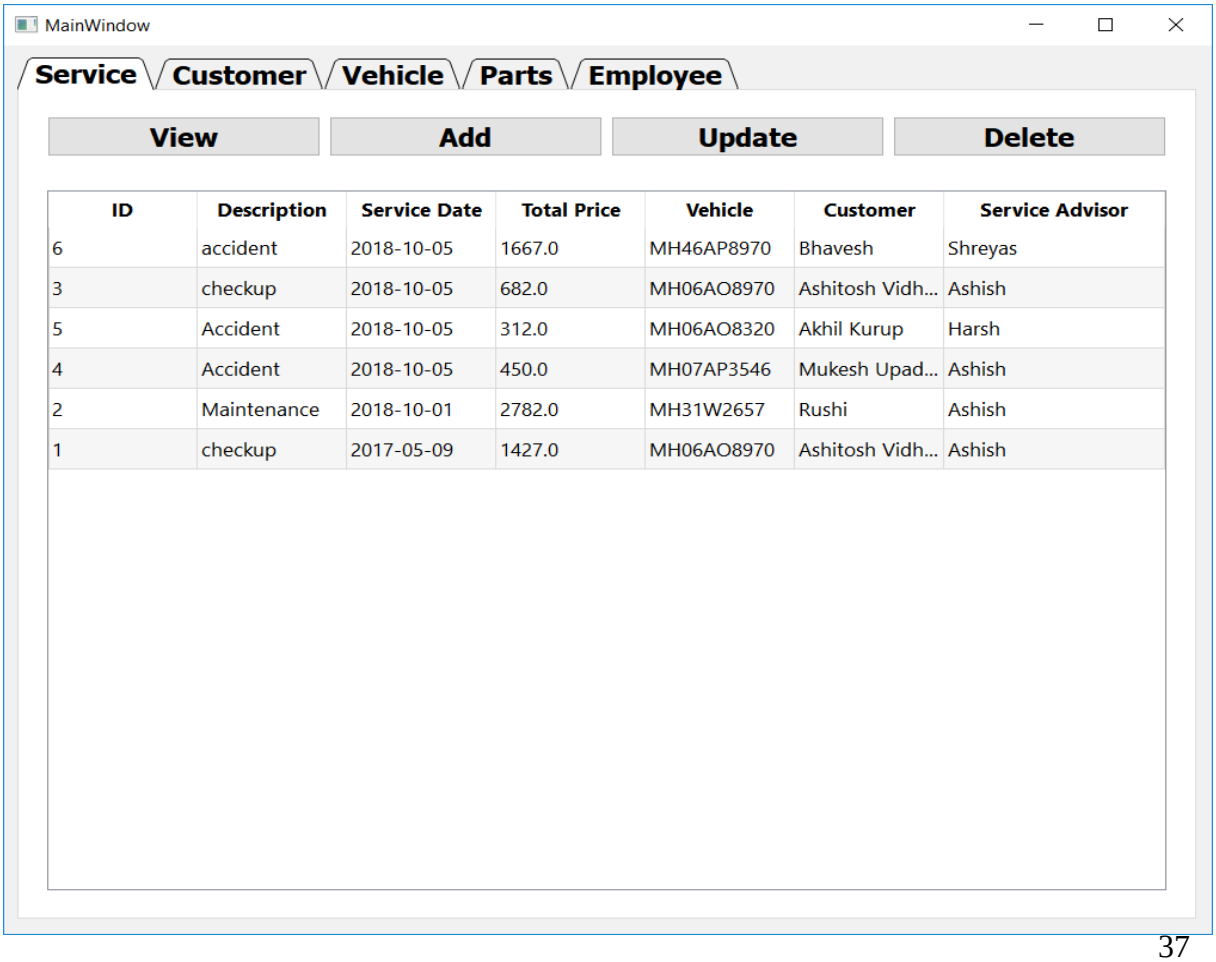
Test Case ID	Module/ Sub Module Name	Form Name	Test Condition	Pre-requisite	Steps	Input Test Data	Expected Output	Actual Output	Pass/ Fail
T01	Main window	Main window	To check main window	System loaded	Click on customer tab	Mouse click	Switch to Customer tab	Swith to Customer tab	Pass
T02	Main window	Main window	To check main window	System loaded	Click on vehicle tab	Mouse click	Switch to Vehicle tab	Switch to Vehicle tab	Pass
T03	Main window	Main window	To check main window	System loaded	Click on vehicle tab	Mouse click	Switch to Parts tab	Switch to Parts tab	Pass
T04	Main window	Main window	To check main window	System loaded	Click on vehicle tab	Mouse click	Switch to Employee tab	Switch to Employee tab	Pass
T05	Main window	Customer tab	To check Add dialog	Customer Tab should be open	Click on Add button	Mouse click	Add dialog Pop-up	Add dialog Pop-up	Pass
T06	Customer tab	Add dialog	To check validation	Add dialog should be open	To check valid name is entered	2	Should not be allowed to enter	Not allowed to enter	Pass
T07	Customer tab	Add dialog		Add dialog should be open	Name is not entered and add is clicked	Name	Message “Check the entered values” should be displayed	Message “Check the Entered Value”	Pass
T08	Customer tab	Update dialog		Customer tab should be open	Update is clicked without selecting record	Update button clicked	Message “No record selected” should be displayed	Message “No record is selected”	Pass
T09	Vehicle tab	Add dialog		Add dialog should be open	Reg.no is not entered and add is clicked	Reg. no	Message “Check the entered values” should be displayed	Message “Check the Entered Value”	Pass

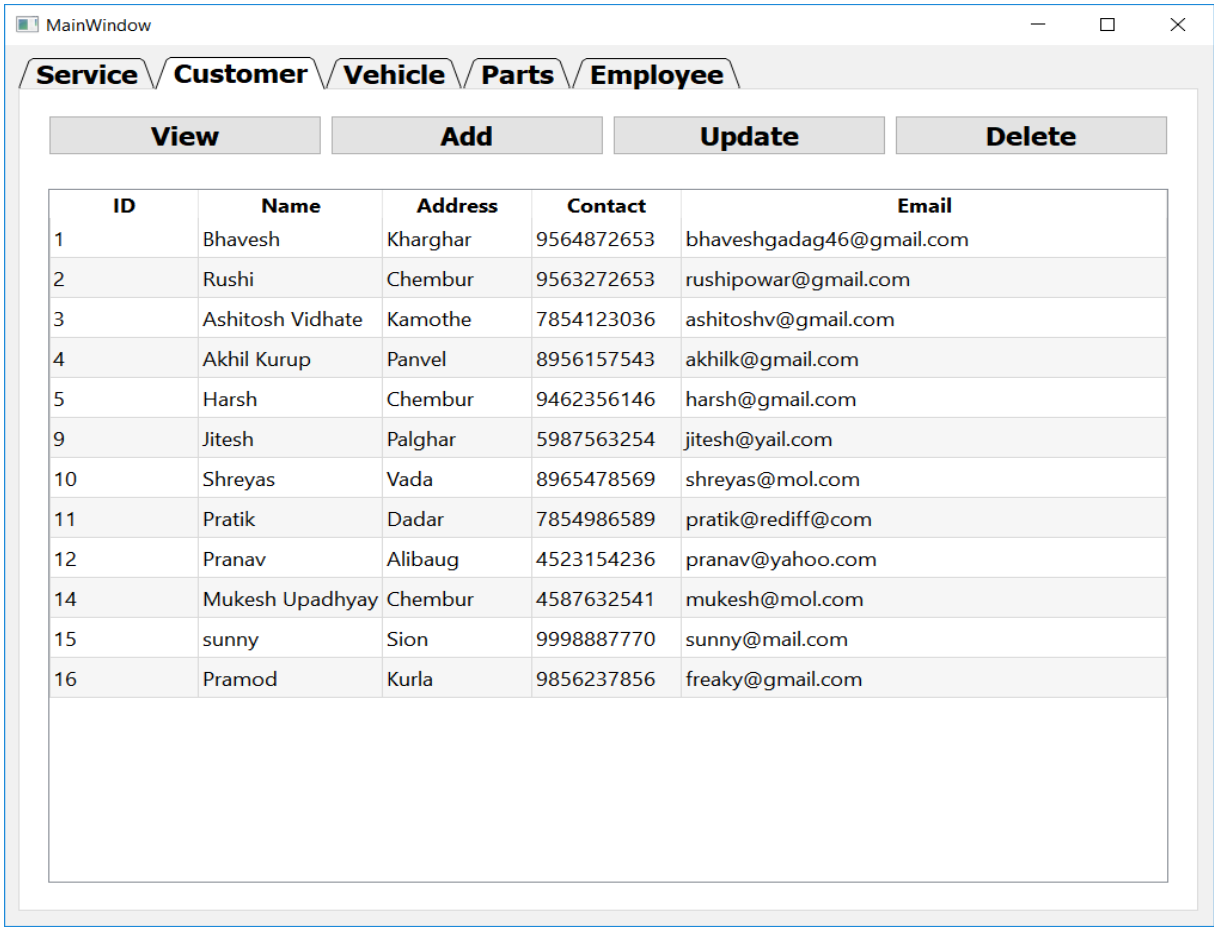
TEST CASES

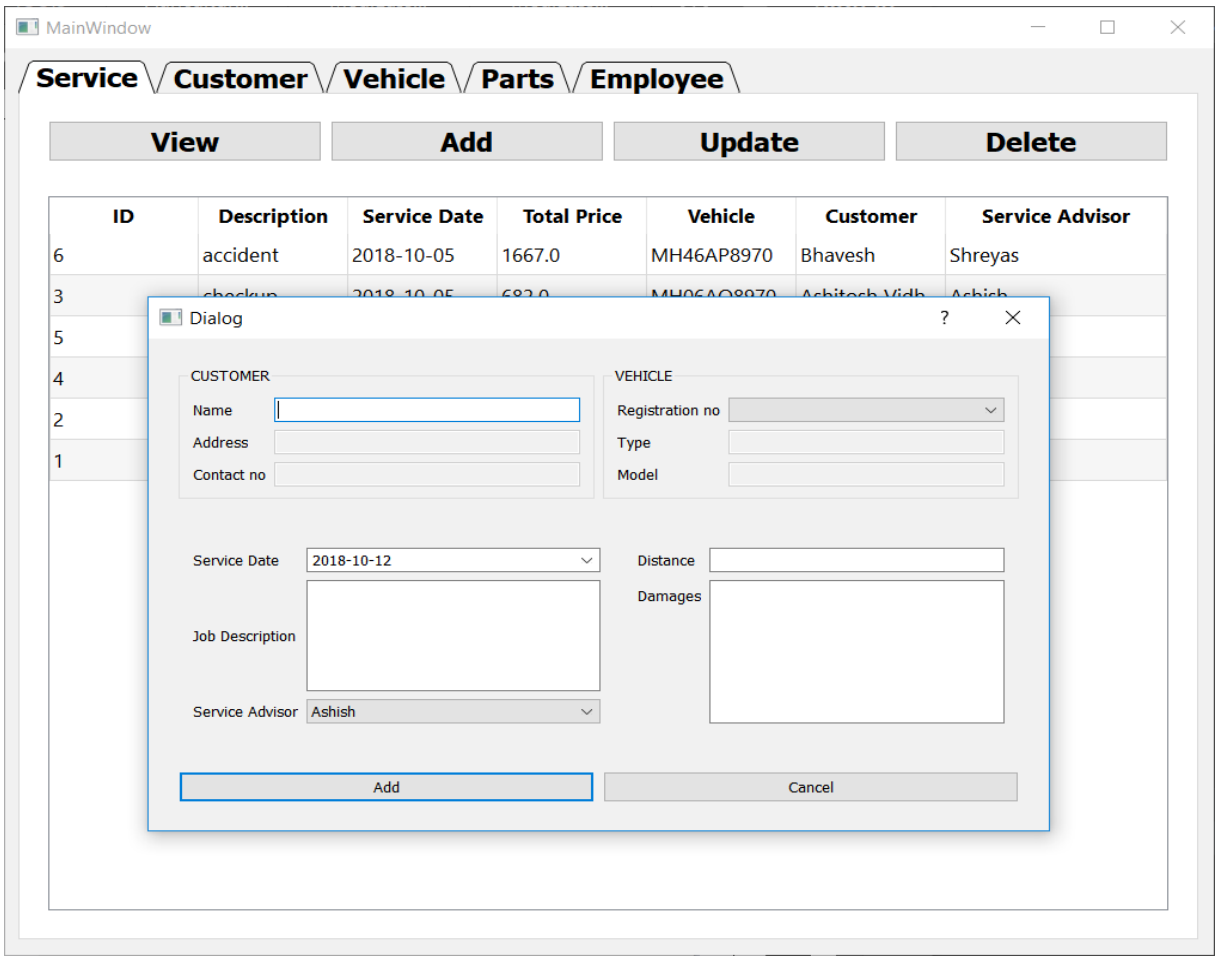
Test Case ID	Module / Sub Module Name	Form Name	Test Condition	Pre-requisite	Steps	Input Test Data	Expected Output	Actual Output	Pass/ Fail
T10	Vehicle tab	Update dialog		Vehicle tab should be open	Update is clicked without selecting record	Update button clicked	Message “No record selected” should be displayed	Message “No record is selected”	Pass
T11	Parts tab	Add dialog		Add dialog should be open	Name is not entered and add is clicked	Name	Message “Check the entered values” should be displayed	Message “Check the Entered Value”	Pass
T12	Parts tab	Update dialog		Parts tab should be open	Update is clicked without selecting record	Update button clicked	Message “No record selected” should be displayed	Message “No record is selected”	Pass
T13	Service tab	Add dialog	To check validation	Add dialog should be open	Valid date should be entered	Date	Message “Check the entered values” should be displayed	Message “Check the Entered Value”	Pass
T14	Service tab	Parts used		Service tab should be open	Double click on Service record	Mouse double click	Parts used dialog should Pop-up	Parts Used Dialog Pop-up	Pass

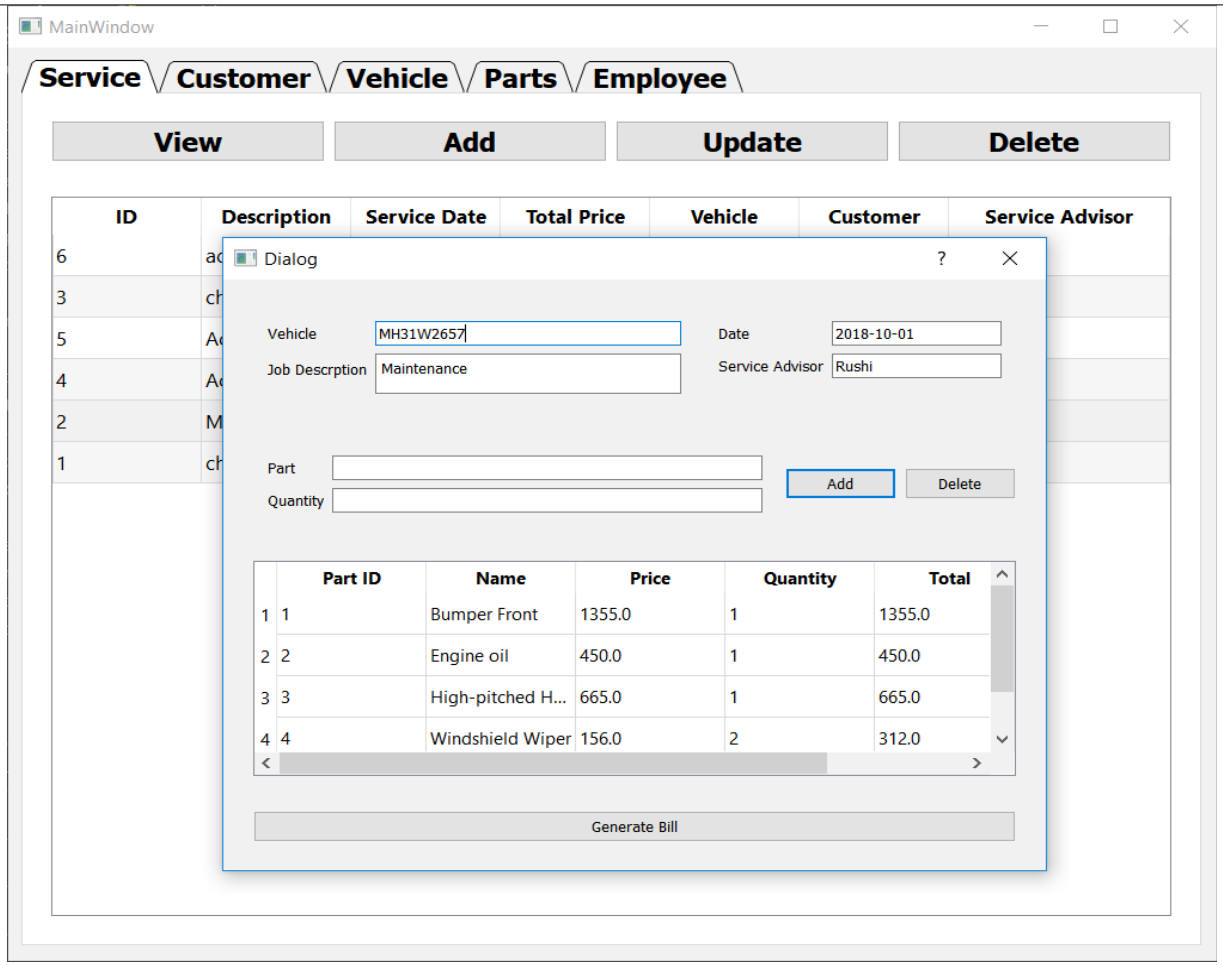
SCREEN LAYOUTS AND REPORT LAYOUTS

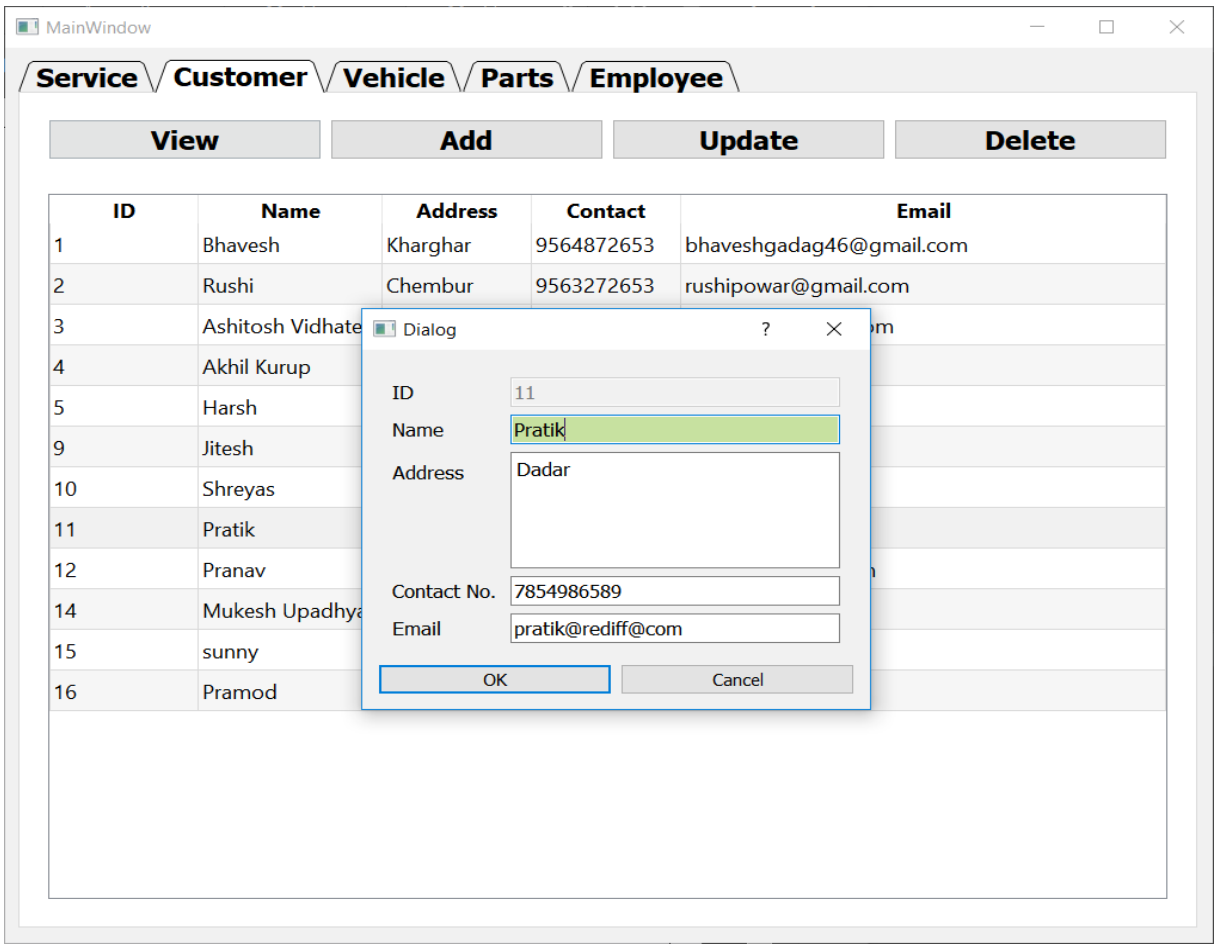





MainWindow
Service
Customer
Vehicle
Parts
Employee
View
Add
Update
Delete
ID Name Address Contact Email
1 Bhavesh Kharghar 9564872653 bhaveshgadag46@gmail.com
2 Rushi Chembur 9563272653 rushipowar@gmail.com
3 Ashitosh Vidhate Kamothe 7854123036 ashitoshv@gmail.com
4 Akhil Kurup Panvel 8956157543 akhilk@gmail.com
5 Harsh Chembur 9462356146 harsh@gmail.com
9 Jitesh Palghar 5987563254 jitesh@yail.com
10 Shreyas Vada 8965478569 shreyas@mol.com
11 Pratik Dadar 7854986589 pratik@rediff@com
12 Pranav Alibaug 4523154236 pranav@yahoo.com
14 Mukesh Upadhyay Chembur 4587632541 mukesh@mol.com
15 sunny Sion 9998887770 sunny@mail.com
16 Pramod Kurla 9856237856 freaky@gmail.com


MainWindow
Service
Customer
Vehicle
Parts
Employee
View
Add
Update
Delete
ID Description Service Date Total Price Vehicle Customer Service Advisor
6 accident 2018-10-05 1667.0 MH46AP8970 Bhavesh Shreyas
Dialog
CUSTOMER
Name
Address
Contact no
VEHICLE
Registration no
Type
Model
Service Date 2018-10-12
Distance
Damages
Job Description
Service Advisor Ashish
Add
Cancel


MainWindow
Service
Customer
Vehicle
Parts
Employee
View
Add
Update
Delete
ID
Description
Service Date
Total Price
Vehicle
Customer
Service Advisor
Dialog
Vehicle
MH31W2657
Date
2018-10-01
Job Descrption
Maintenance
Service Advisor
Rushi
Part
Quantity
Add
Delete
Part ID
Name
Price
Quantity
Total
1 1 Bumper Front 1355.0 1 1355.0
2 2 Engine oil 450.0 1 450.0
3 3 High-pitched H... 665.0 1 665.0
4 4 Windshield Wiper 156.0 2 312.0
Generate Bill


MainWindow
Service
Customer
Vehicle
Parts
Employee
View
Add
Update
Delete
ID
Name
Address
Contact
Email
1 Bhavesh Kharghar 9564872653 bhaveshgadag46@gmail.com
2 Rushi Chembur 9563272653 rushipowar@gmail.com
3 Ashitosh Vidhate
4 Akhil Kurup
5 Harsh
9 Jitesh
10 Shreyas
11 Pratik
12 Pranav
14 Mukesh Upadhya
15 sunny
16 Pramod
Dialog
ID
11
Name
Pratik
Address
Dadar
Contact No.
7854986589
Email
pratik@rediff@com
OK
Cancel

REPORT LAYOUT

Print Date: 2018-Oct-12 Timing: 10:27:10

Customer ID : 2
Name : Rushi
Address : Chembur
Mobile : 9563272653
Email : rushipowar@gmail.com
Vehicle ID : 5
Reg. no : MH31W2657
Company : Renalt
Model : Duster
Type : SUV

Service ID : 2
Job Desc. : Maintenance
Job Date : 2018-10-01
Distance : 6769

Service Advisor : Ashish
SA Contact no : 8615742365

Sr no.	Part ID	Part	Rate	Quantity	Amount
1	1	Bumper Front	1355.0	1	1355.0
2	2	Engine oil	450.0	1	450.0
3	3	High-pitched Horn	665.0	1	665.0
4	4	Windshield Wiper	156.0	2	312.0
				Total Amount	2782.0

(Customer Signature)

(Authorized Signature)

FUTURE ENHANCEMENTS

Future Enhancements are those features that can be incorporated in the application in its future versions and provide new features and better working of the present application.

Following are some of the few enhancements that can be incorporated in this application in future:

- Addition of stock module to generate stock orders.
- Addition of graphs to check profit/loss in revenue.

REFERENCES AND BIBLIOGRAPHY

Books Referred:

- PYTHON, Fundamental of Python Programming, by Rick Halterman.
- Murach's MySQL, Book by Joel Murach.

Sites Referred:

- www.youtube.com
- www.qtcentre.org
- www.stackoverflow.com
- https://www.reportlab.com/docs/reportlab-userguide.pdf