A

PROJECT REPORT ON

SALES AND SERVICE MANAGEMENT SYSTEM

FOR

YASH ELECTRICAL'S

SUBMITTED BY

AMAR UDAY PATIL

SUBMITTED TO

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN PARTIAL FULFILLMENT OF DEGREE

MASTER OF COMPUTER APPLICATION

UNDER THE GUIDANCE OF

Prof. Madhavi Shamkuwar

Through

Zeal Education Society's

Zeal Institute of Business Administration, Computer Application & Research (ZIBACAR)

2019 - 2022

ZEAL EDUCATION SOCIETY'S
ZEAL INSTITUTE OF BUSINESS ADMINISTRATION, COMPUTER APPLICATION AND RESEARCH (ZIBACAR)
NARHE | PUNE | INDIA
PUN CODE: IMMP013170 DTE CODE: 6152 AISHE CODE: C-41828

CERTIFICATE

This is to certify that the project report entitled, **"SALES AND SERVICE MANAGEMENT SYSTEM"** for **"YASH ELECTRICAL'S"** being submitted herewith for the internal work of the degree of **MASTER OF COMPUTER APPLICATION** to **SAVITRIVAI PHULE PUNE UNIVERSITY** is the result of the original project work completed by AMAR UDAY PATIL under my supervision and guidance and to the best of my knowledge and belief, the work embodies in this Project has not formed earlier the basis for the award of any Degree of similar title or any other University or examining body.

Date:

Place: Pune

Prof. Madhavi Shamkuwar	**Dr. B. J. Mohite**	**Dr. Rajesh Kashyap**
Project Guide	**Project Coordinator**	**Director**

Examiner 1 Sign Examiner 2 Sign

ZEAL EDUCATION SOCIETY'S
ZEAL INSTITUTE OF BUSINESS ADMINISTRATION, COMPUTER APPLICATION AND RESEARCH (ZIBACAR)
NARHE | PUNE | INDIA
PUN CODE: IMMP013170 DTE CODE: 6152 AISHE CODE: C-41828

DECLARATION BY STUDENT

To,

The Director,

ZIBACAR, Pune

I, undersigned hereby declare that this project titled, "Sales and Service Management System" written and submitted by me to SPPU, Pune, in partial fulfillment of the requirement of the award of the degree of **MASTER OF COMPUTER APPLICATION** under the guidance of **Prof. Madhvi Shamkuwar**, is my original work.

I further declare that to the best of my knowledge and belief, this project has not been submitted to this or any other University or Institution for the award of any Degree.

Place: Pune

Date:

AMAR UDAY PATIL

ACKNOWLEDGEMENT

I extend my sincere gratitude to Honorable **Shri Sambhajirao Katkar,** President, ZES Pune, **Dr. Sachin Chavan**, Head, ZES Management Programmes, **Dr. Rajesh Kashyap,** Director of ZIBACAR, Dr. Rishikesh Kakandikar and **Prof. Madhvi Shamkuwar** for allowing me to carry out the study and for his constant encouragement, valuable suggestions and guidance during the research work.

I extend my special thanks to **Prof. Madhavi Shamkuwar, Prof. Kirti Samrit, Prof. Madhavi Shamkuwar, Prof. Dharmendra Singh, Dr. Babasaheb Mohite, Dr. Rajesh Kashyap** for their kind co-operation and inspiration.

I extend my special gratitude to my dearest family members and friends who encouraged and motivated me to complete the project report.

Place: Pune

Date:

AMAR UDAY PATIL

YASH ELECTRICAL'S

SALES AND SERVICES

Main Road, A/P Kasegaon, Tal-Walwa, Dist – Sangli, Maharashtra 415404

COMPLETION LETTER

This is certify that **Mr. Amar Uday Patil** is undergoing his live project for **Yash Electrical's.** In partial fulfilment of the MCA (Master of Computer Application) course from **Zeal Institute of Business Administration, Computer Application & Research Narhe, Pune** under Savitribai Phule Pune University.

Details of Project –

Project Title	:	**SALES AND SERVICE MANAGEMENT SYSTEM**
Project Duration	:	July 2021 to July 2022
Technology	:	Angular and Spring Boot with Restful Web-services

Major part of project he has done till 30th July 2022

He has completed assigned work during tenure. He was sincere and found hardworking. He has completed design and development of system as per our company standards and requirements.

This certificate is issued to his academic purpose. We wish him all the best for his future.

Place: Kasegaon

For Yash Electrical's

Mr. Sanjay Lahigade.

INDEX

CHAPTER 1
INTRODUCTION

1.1 Client Profile

A marvelous journey that began with a small Electronics Showroom at Kasegaon in 2002, Offering guests the best in quality and service has been the primary principle at Yash Electricals since its inception by Mr. Sanjay Lahigade, a visionary with astute business acumen and foresight. Yash Electricals has since evolved and has become one of most trustable in the market. Along with a dedicated team of prolific managers, Mr. Sanjay stand tall on the promise of serving their guests with the absolute best in both product and after-sales service. Sir's guest-focused initiatives and intuitive understanding of emerging markets have helped Yash Electricals to become a household name and the first preference for many when it comes to quality electronics and consumer durables. Today, after more than two decades of successful operation, planning to have multiple showrooms across the upcoming years catering to customers of all budgets and all preferences.

We have always been a firm believer in servicing customer needs and efficient post-sales services. Our customer-focused policies together with the intuitive understanding of emerging markets has led Yash Electricals from strength to strength. Thanks to a very good relationship with suppliers of well-know and established brands, we can offer prompt and reliable after sales services to all customers.

- **MISSION**

Yash Electricals would give its customers best value, best products and the best available service in the industry. Honesty towards our customers, principals and our associates would be the pillar on which yash electricals would stand and grow.

Services/ Products Offered by Yash Electricals:

- ✓ **Your go-to electronics store**

At Yash Electricals, you'll find a range of electronic products and items. You can opt for devices with basic features, as well as ones with avant-garde features, depending on your level of comfort. Our gadgets are genuine, making them great gifts as well. You're sure to find exactly what you're looking for.

- ✓ **Washing machines**

Having a washing machine to take care of your laundry needs frees up a lot of your time. At our electronics shop, you can shop for washing machines in a range of sizes. Select by color, load type, dryer type and lots more. You can opt for fully or semi-automatic washing machines from Samsung, Whirlpool, LG, Godrej, and other top brands.

- ✓ **Home appliances**

Having the right gadgets at home can make your life comfortable and stress-free. Thanks to technology, there is a gadget for almost everything – keeping you cool, washing clothes, storing food. Our store offers the latest launches as well as tried and tested products.

1.2 Abstract

This Sales and Service Management System will help sales persons to look after their daily work, enquiries, visits, tickets they reached for week and monthly basis. Sales management systems are information systems used in CRM marketing and management that help automate some sales management functions.

1.3 Existing System and Need for System

1. **Existing System**

- The existing system is manual system. So it is not user friendly as like Graphical User Interface
- This manual system requires more time to collect details of customers.
- Visits details data is collected manually.
- Details of each customer/product are recorded manually. Hence data modification is not possible.
- Sales Enquiries and Service Ticket distribution is also done manually by responsible person of product category. For complete this process that person has to check current allocation ticket list.
- When enquiry/ticket get completed by employee then task record process in manually. Which can be leads to data misplacement.
- All customer details, enquiry/ticket details and employee details are saving manually.
- No report generation facility is available to take decision.
- In existing system searching of particular information or enquiry/ticket is very critical, it takes lot of time.

2. **Need for System**

- Time consuming.
- Consumes large volume of paper work.
- Needs manual calculations.
- No direct role for the higher officials.
- To avoid all these limitations and make the working more accurately the system needs to be computerized in a better way.
- To create web-based application for the organization.
- To reduce the time and complexity of maintaining the records.
- To save all visits/enquiry/tickets records as per customers.
- To manage enquiry allocation / ticket distribution to employee.
- To maintain all transaction records of visits, enquiries, tickets, customer information and employee task history.
- To trace the visits/enquiry/tickets done by employee and pending enquiry/tickets of employee.
- To manage service completion date for best service and customer satisfaction.
- To search particular information related to enquiry/ticket, employee and customers, products and master data without taking lot of time.
- To search visits/enquiry/ticket details between particular dates.

1.4 Scope of the System:

Scope of the project is limited to

1. **Visit details for every customer for post sales:**

   This system will be used to store all the visit details for each customers.

2. **Adding Enquiries for various products:**

   This system is mainly developed for manage enquiries and enquiries details. By using

this system users can able to change and track all the activities related to enquiries.

3. **Adding Service Tickets for customers for post sales services:**

   This system is also mainly developed for manage tickets as a part of post sales services. Using this system users can work on the various post sales services.

4. **Managing/Storing all Master Data**

   This system will be used to store and manage all the master data which is required to visit, enquiry and ticket modules.

1.5 Operating Environment – Hardware and Software

1. **Hardware Specification of machine used**

   A. Hardware at client side:

Processor	: Core i3 or above
RAM	: 2GB or above
Memory	: At least 20GB free memory on disk.

   B. Hardware at server side:

RAM	: 2GB or above
Memory	: At least 20GB free memory on disk.
Bandwidth	: 50GB or more.

2. **Software Specification of machine used**

   A. Software at client side:

OS	: Windows 10
Browser	: Google Chrome, Microsoft Edge, Internet Explorer, any

   B. Software at server side:

OS	: Windows 10
Browser	: Google Chrome, Microsoft Edge, Internet Explorer, any
DB Server	: MySQL
Tools	: Java JDK 1.8, Tomcat Server, Eclipse, Microsoft Visual Studio, Postman, Node.js

1.6 Brief Description of Technology Used

1) Angular 12
2) Spring Boot, Spring Data JPA
3) Restful Web-services

Angular

Angular is a very powerful JavaScript Framework. It is used in Single Page Application (SPA) projects. It extends HTML DOM with additional attributes and makes it more responsive to user actions. Angular is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache license version 2.0.

Core Features

Data-binding: It is the automatic synchronization of data between model and view components.

Scope: These are objects that refer to the model. They act as a glue between controller and view.

Controller: These are JavaScript functions bound to a particular scope.

Services: Angular comes with several built-in services such as $http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
Filters: These select a subset of items from an array and returns a new array.

Directives: Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. Angular has built-in directives such as ngBind, ngModel, etc.

Templates: These are the rendered view with information from the controller and model. These can be a single file (such as index.html) or multiple views in one page using partials.

Routing: It is concept of switching views.

Model View Whatever: MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. Angular does not

Implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-View Model). The Angular JS team refers it humorously as Model View Whatever.

Dependency Injection: Angular has a built-in dependency injection subsystem that helps the developer to create, understand, and test the applications easily.

Advantages of Angular

- It provides the capability to create Single Page Application in a very clean and maintainable way.
- It provides data binding capability to HTML. Thus, it gives user a rich and responsive experience.
- Angular code is unit testable.
- Angular uses dependency injection and make use of separation of concerns.
- Angular provides reusable components.
- With Angular, the developers can achieve more functionality with short code.
- In Angular, views are pure html pages, and controllers written in JavaScript do the business processing.

On the top of everything, Angular applications can run on all major browsers and smart phones, including Android and iOS based phones/tablets.

Angular – Directives

Angular directives are used to extend HTML. They are special attributes starting with ng-prefix. Let us discuss the following directives −

ng-app − This directive starts an Angular Application.

ng-init − This directive initializes application data.

Angular – Filters

Filters are used to modify the data. They can be clubbed in expression or directives using pipe (|) character. The following list shows the commonly used filters.

Sr. No	Name & Description
1	uppercase - converts a text to upper case text.
2	lowercase - converts a text to lower case text.
3	currency - formats text in a currency format.
4	filter - filter the array to a subset of it based on provided criteria.

Events
Angular provides multiple events associated with the HTML controls. For example, ng-click directive is generally associated with a button. Angular supports the following events −

- ng-click
- ng-mousedown
- ng-mouseup
- ng-mouseenter
- ng-mouseleave
- ng-mousemove
- ng-mouseover
- ng-keydown
- ng-keyup
- ng-keypress
- ng-change

Angular Dependency Injection

Dependency Injection in Angular is a software design pattern that implements inversion of control for resolving dependencies. It decides how components hold their dependencies. Itcan be used while defining the components or providing run and config blocks of the module. It enables you to make the components reusable, testable, and maintainable.

Inversion of Control: It means that objects do not create other objects on which they rely to do their work. Instead, they get these objects from an outside source. This forms the basis of Angular Dependency Injection wherein if one object is dependent on another; the primary object does not take the responsibility of creating the dependent object and then use its methods. Instead, an external source (which in Angular, is the Angular framework itself) creates the dependent object and gives it to the source object for further usage.

TypeScript:

TypeScript lets you write JavaScript the way you really want to. TypeScript is a typed superset of JavaScript that compiles to plain JavaScript. TypeScript is pure object oriented with classes, interfaces and statically typed like C# or Java. The popular JavaScript framework Angular is written in TypeScript. Mastering TypeScript can help programmers to write object-oriented programs and have them compiled to JavaScript, both on server side and client side.

JavaScript was introduced as a language for the client side. The development of Node.js has marked JavaScript as an emerging server-side technology too. However, as JavaScript code grows, it tends to get messier, making it difficult to maintain and reuse the code. Moreover, its failure to embrace the features of Object Orientation, strong type checking and compile-time error checks prevents JavaScript from succeeding at the enterprise level as a full-fledged server-side technology. TypeScript was presented to bridge this gap.

Spring Framework and Spring Boot with Spring Data JPA

Before moving to Spring Framework we need to understand some basic then only we can understand Spring Framework in detail. Spring Framework internally using all java technologies. Like

1) Core Java
2) Advanced Java – JDBC, Servlet, JSP

Core Java

Java is a MUST for become a great Software Engineer especially when they are working in Software Development Domain.

Spring Framework

Spring framework is an open source Java platform that provides comprehensive infrastructure support for developing robust Java applications very easily and very rapidly. Spring framework was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003. This tutorial has been written based on Spring Framework version 4.1.6 released in Mar 2015.

Why Spring?

- Spring is the most popular application development framework for enterprise Java. Millions of developers around the world use Spring Framework to create high performing, easily testable, and reusable code.

- Spring framework is an open source Java platform. It was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003.

- Spring is lightweight when it comes to size and transparency. The basic version of Spring framework is around 2MB.

- The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make J2EE development easier to use and promotes good programming practices by enabling a POJO-based programming model.

Applications of Spring
Following is the list of few of the great benefits of using Spring Framework −

- **POJO Based** - Spring enables developers to develop enterprise-class applications using POJOs. The benefit of using only POJOs is that you do not need an EJB container product such as an application server but you have the option of using only a robust servlet container such as Tomcat or some commercial product.

- **Modular** - Spring is organized in a modular fashion. Even though the number of

packages and classes are substantial, you have to worry only about the ones you need and ignore the rest.

- **Integration with existing frameworks** - Spring does not reinvent the wheel, instead it truly makes use of some of the existing technologies like several ORM frameworks, logging frameworks, JEE, Quartz and JDK timers, and other view technologies.

- **Testablity** - Testing an application written with Spring is simple because environment-dependent code is moved into this framework. Furthermore, by using Java Bean style POJOs, it becomes easier to use dependency injection for injecting test data.

- **Web MVC** - Spring's web framework is a well-designed web MVC framework, which provides a great alternative to web frameworks such as Struts or other over-engineered or less popular web frameworks.
- **Central Exception Handling** - Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO, for example) into consistent, unchecked exceptions.

- **Lightweight** - Lightweight IoC containers tend to be lightweight, especially when compared to EJB containers, for example. This is beneficial for developing and deploying applications on computers with limited memory and CPU resources.

- **Transaction management** - Spring provides a consistent transaction management interface that can scale down to a local transaction (using a single database, for example) and scale up to global transactions (using JTA, for example).

Spring - IoC Containers

The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete life cycle from creation till destruction. The Spring container uses DI to manage the components that make up an application. These objects are called Spring Beans, which we will discuss in the next chapter.

The container gets its instructions on what objects to instantiate, configure, and assemble by reading the configuration metadata provided. The configuration metadata can be represented either by XML, Java annotations, or Java code. The following diagram represents a high-level view of how Spring works. The Spring IoC container makes use of Java POJO classes and configuration metadata to produce a fully configured and executable system or application.

Spring provides the following two distinct types of containers.

Sr. No	Container and Description
1	**Spring BeanFactory Container:** This is the simplest container providing the basic support for DI and is defined by the *org.springframework.beans.factory.BeanFactory* interface. The *BeanFactory* and related interfaces, such as *BeanFactoryAware*,

	InitializingBean, *DisposableBean*, are still present in Spring for the purpose of backward compatibility with a large number of third-party frameworks that integrate with Spring.
2	**Spring ApplicationContext Container:** This container adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners. This container is defined by the *org.springframework.context.ApplicationContext* interface.

Spring - Bean Definition

The objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. These beans are created with the configuration metadata that you supply to the container. For example, in the form of XML <bean/> definitions which you have already seen in the previous chapters.

Bean definition contains the information called configuration metadata, which is needed for the container to know the following −

- How to create a bean
- Bean's lifecycle details
- Bean's dependencies

All the above configuration metadata translates into a set of the following properties that make up each bean definition.

Sr.No	Properties and Description
1	class: This attribute is mandatory and specifies the bean class to be used to create the bean.
2	name: This attribute specifies the bean identifier uniquely. In XMLbased configuration metadata, you use the id and/or name attributes to specify the bean identifier(s).
3	scope: This attribute specifies the scope of the objects created from a particular bean definition and it will be discussed in bean scopes chapter.
4	constructor-arg: This is used to inject the dependencies and will be discussed in subsequent chapters.
5	properties: This is used to inject the dependencies and will be discussed in subsequent chapters.
6	autowiring mode: This is used to inject the dependencies and will be discussed in subsequent chapters.
7	lazy-initialization mode: A lazy-initialized bean tells the IoC container to create a bean instance when it is first requested, rather than at the startup.

8	initialization method: A callback to be called just after all necessary properties on the bean have been set by the container. It will be discussed in bean life cycle chapter.
9	destruction method: A callback to be used when the container containing the bean is destroyed. It will be discussed in bean life cycle chapter.

Spring - Bean Life Cycle

The life cycle of a Spring bean is easy to understand. When a bean is instantiated, it may be required to perform some initialization to get it into a usable state. Similarly, when the bean is no longer required and is removed from the container, some cleanup may be required.

Though, there are lists of the activities that take place behind the scene between the time of bean Instantiation and its destruction, this chapter will discuss only two important bean life cycle callback methods, which are required at the time of bean initialization and its destruction.

To define setup and teardown for a bean, we simply declare the <bean> with initmethod and/or destroy-method parameters. The init-method attribute specifies a method that is to be called on the bean immediately upon instantiation. Similarly, destroymethod specifies a method that is called just before a bean is removed from the container.

Spring - Dependency Injection

Dependency Injection is a fundamental aspect of the Spring framework, through which the Spring container “injects” objects into other objects or “dependencies”.

Simply put, this allows for loose coupling of components and moves the responsibility of managing components onto the container.

Inversion of Control?

Inversion of Control is a principle in software engineering which transfers the control of objects or portions of a program to a container or framework. We most often use it in the context of object-oriented programming.

In contrast with traditional programming, in which our custom code makes calls to a library, IoC enables a framework to take control of the flow of a program and make calls to our custom code. To enable this, frameworks use abstractions with additional behavior built in. If we want to add our own behavior, we need to extend the classes of the framework or plugin our own classes.

The advantages of this architecture are:

- decoupling the execution of a task from its implementation
- making it easier to switch between different implementations
- greater modularity of a program
- greater ease in testing a program by isolating a component or mocking its dependencies, and allowing components to communicate through contracts

We can achieve Inversion of Control through various mechanisms such as: Strategy design pattern, Service Locator pattern, Factory pattern, and Dependency Injection (DI).

Dependency Injection?
Dependency injection is a pattern we can use to implement IoC, where the control being inverted is setting an object's dependencies.

Connecting objects with other objects, or “injecting” objects into other objects, is done by an assembler rather than by the objects themselves.

Spring Boot

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can **"just run".**

We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need minimal Spring configuration.

Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' dependencies to simplify your build configuration
- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks, and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

Spring Boot:

You can choose Spring Boot because of the features and benefits it offers as given here −

- It provides a flexible way to configure Java Beans, XML configurations, and Database Transactions.
- It provides a powerful batch processing and manages REST endpoints.
- In Spring Boot, everything is auto configured; no manual configurations are needed.
- It offers annotation-based spring application
- Eases dependency management

- It includes Embedded Servlet Container

How does spring boot works?

Spring Boot automatically configures your application based on the dependencies you have added to the project by using ***@EnableAutoConfiguration*** annotation. For example, if MySQL database is on your classpath, but you have not configured any database connection, then Spring Boot auto-configures an in-memory database.

The entry point of the spring boot application is the class contains ***@SpringBootApplication*** annotation and the main method.

Spring Boot automatically scans all the components included in the project by using ***@ComponentScan*** annotation.

Spring Boot Starters
Handling dependency management is a difficult task for big projects. Spring Boot resolves this problem by providing a set of dependencies for developers convenience.

For example, if you want to use Spring and JPA for database access, it is sufficient if you include spring-boot-starter-data-jpa dependency in your project.

Note that all Spring Boot starters follow the same naming pattern spring-boot-starter- *, where * indicates that it is a type of the application.

Spring Data JPA

Spring Data's mission is to provide a familiar and consistent, Spring-based programming model for data access while still retaining the special traits of the underlying data store.

It makes it easy to use data access technologies, relational and non-relational databases, map-reduce frameworks, and cloud-based data services. This is an umbrella project which contains many subprojects that are specific to a given database. The projects are developed by working together with many of the companies and developers that are behind these exciting technologies.

Features

- Sophisticated support to build repositories based on Spring and JPA
- Support for Querydsl predicates and thus type-safe JPA queries
- Transparent auditing of domain class
- Pagination support, dynamic query execution, ability to integrate custom data access code
- Validation of ***@Query*** annotated queries at bootstrap time

- Support for XML based entity mapping

- JavaConfig based repository configuration by introducing ***@EnableJpaRepositories.***

JPA Providers
JPA is an open source API, therefore various enterprise vendors such as Oracle, Redhat, Eclipse, etc. provide new products by adding the JPA persistence flavor in them. Some of these products include −

Hibernate, Eclipselink, Toplink, **Spring Data JPA**, etc.

Spring Based Restful Web-services

REST has quickly become the de-facto standard for building web services on the web because they're easy to build and easy to consume.

There's a much larger discussion to be had about how REST fits in the world of microservices, but — for this tutorial — let's just look at building RESTful services.

Why REST? REST embraces the precepts of the web, including its architecture, benefits, and everything else. This is no surprise given its author, Roy Fielding, was involved in probably a dozen specs which govern how the web operates.

What benefits? The web and its core protocol, HTTP, provide a stack of features:

- Suitable actions (GET, POST, PUT, DELETE)
- Caching
- Redirection and forwarding
- Security (encryption and authentication)

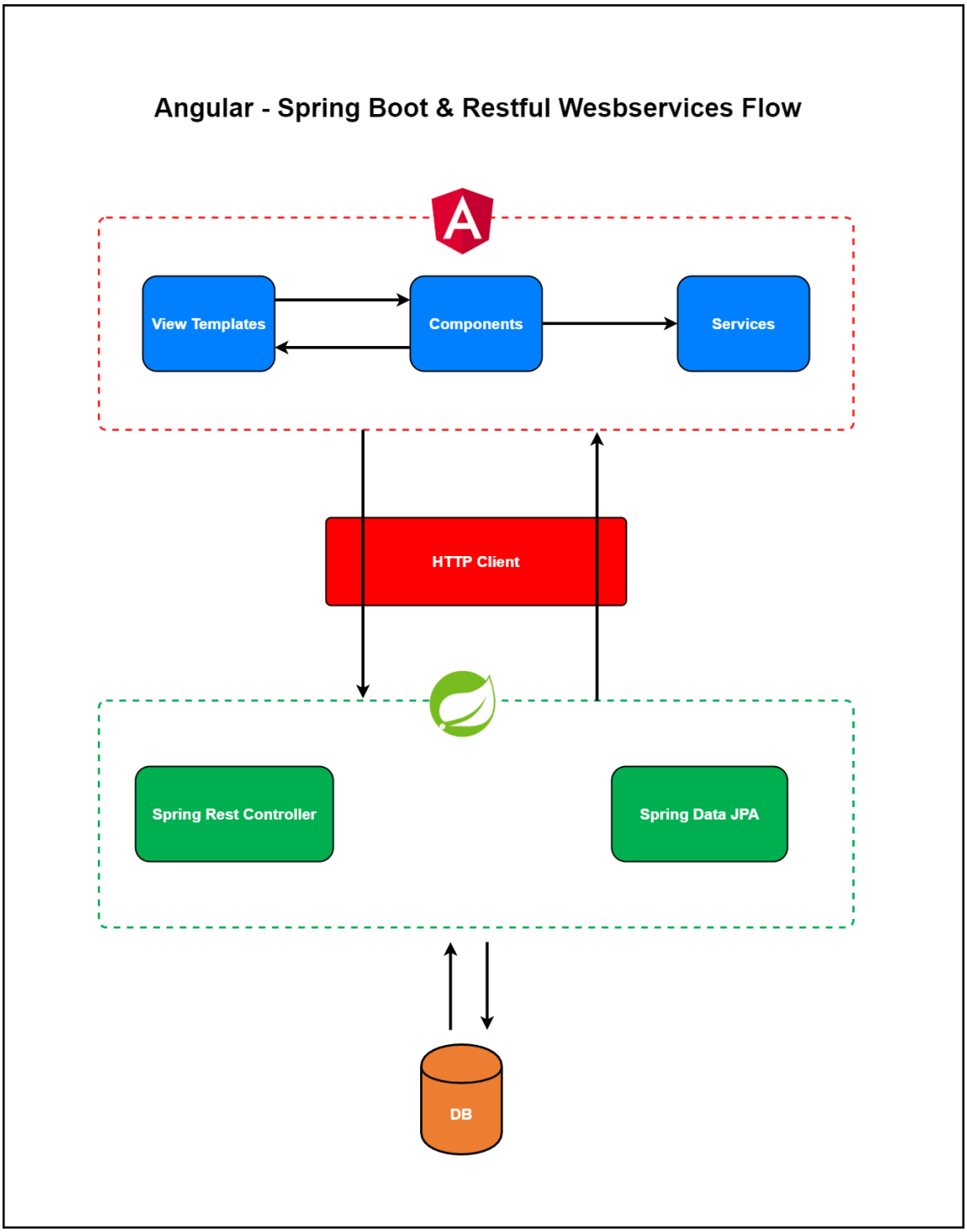
These are all critical factors on building resilient services. But that is not all. The web is built out of lots of tiny specs, hence it's been able to evolve easily, without getting bogged down in "standards wars".

Developers are able to draw upon 3rd party toolkits that implement these diverse specs and instantly have both client and server technology at their fingertips.

By building on top of HTTP, REST APIs provide the means to build:

- Backwards compatible APIs
- Evolvable APIs
- Scaleable services
- Securable services
- A spectrum of stateless to stateful services

What's important to realize is that REST, however ubiquitous, is not a standard, per se, but an approach, a style, a set of constraints on your architecture that can help you build web-scale systems. In this tutorial we will use the Spring portfolio to build a RESTful service while leveraging the stackless features of REST.


Angular - Spring Boot & Restful Wesbservices Flow
View Templates
Components
Services
HTTP Client
Spring Rest Controller
Spring Data JPA
DB

CHAPTER 2
PROPOSED SYSTEM

2.1 Feasibility Study

A feasibility study is an analysis that considers all of a project's relevant factors including economic, technical, legal, and scheduling considerations to ascertain the likelihood of completing the project successfully.

Whether a project is feasible or not can depend on several factors, including the project's cost and return on investment, meaning whether the project generated enough revenue or sales from consumers.

However, a feasibility study isn't only used for projects looking to measure and forecast financial gains. In other words, feasible can mean something different, depending on the industry and the project's goal. For example, a feasibility study could help determine whether a hospital can generate enough donations and investment dollars to expand and built a new cancer center.

Although feasibility studies can help project managers determine the risk and return of pursuing a plan of action, several steps and best practices should be considered before moving forward.

KEY TAKEAWAYS

- A feasibility study assesses the practicality of a proposed plan or project.
- A feasibility study considers many factors, including economic, technical, and scheduling to determine whether a project can succeed.
- Whether a project is feasible or not can depend on the project's cost and return on investment, which might include revenue from consumers.
- A company may conduct a feasibility study to consider launching a new business or adopting a new product line.
- It's a good idea to have a contingency plan in case of unforeseeable circumstances or if the original project is not feasible.

Understanding a Feasibility Study

A feasibility study is an assessment of the practicality of a proposed plan or project. A feasibility study analyzes the viability of a project to determine whether the project or venture is likely to succeed. The study is also designed to identify potential issues and problems that could arise from pursuing the project.

As part of the feasibility study, project managers must determine whether they have enough people, financial resources, and the appropriate technology. The study must also determine the return on investment, whether it's measured as a financial gain or a benefit to society, as in the case of a nonprofit.

In some cases, a feasibility study might include a significant change in how a business operates, such as an acquisition of a competitor. As a result, the feasibility study might include a cash flow analysis, measuring the level of cash generated from revenue versus the project's operating costs. A risk assessment must also be completed to determine whether the return is enough to offset the level of risk of undergoing the venture.

Important: When doing a feasibility study, it's always good to have a contingency plan that you also test to make sure it's a viable alternative in case the first plan fails.

Benefits of a Feasibility Study

There are several benefits to feasibility studies, including helping project managers discern the pros and cons of undertaking a project before investing a significant amount of time and capital into it. Feasibility studies can also provide a company's management team with crucial information that could prevent them from entering into a risky business venture.

Feasibility studies also help companies with new business development, including determining how it will operate, potential obstacles, competition, market analysis, and the amount and source of financing needed to grow the business. Feasibility studies aim for marketing strategies that could help convince investors and banks that investing in a particular project or business is a wise choice.

7 Steps to Do a Feasibility Study

1. Conduct a Preliminary Analysis

Begin by outlining your project plan. You should focus on an unserved need, a market where the demand is greater than the supply, and whether the product or service has a distinct advantage. Then you need to determine if the feasibility factors are too high to clear (i.e. too expensive, unable to effectively market, etc.).

2. Prepare a Projected Income Statement

This step requires you to work backward. Start with what you expect the income from the project to be and then what project funding is needed to achieve that goal. This is the foundation of an income statement. Things to take into account here include what services are required and how much they'll cost, any adjustments to revenues, such as reimbursements, etc.

3. Conduct a Market Survey, or Perform Market Research

This step is key to the success of your feasibility study, so make your market analysis as thorough as possible. It's so important that if your organization doesn't have the resources to do a proper one, then it is advantageous to hire an outside firm to do so.

The market research is going to give you the clearest picture of the revenues and return on investment you can realistically expect from the project. Some things to consider are the geographic influence on the market, demographics, analyzing competitors, the value of the market and what your share will be and if the market is open to expansion (that is, response to your offer).

4. Plan Business Organization and Operations

Once the groundwork of the previous steps has been laid, it's time to set up the organization and operations of the planned project to meet its technical, operational, economic and legal feasibility factors. This is not a superficial, broad-stroke endeavor. It should be thorough and include start-up costs, fixed investments and operating costs.

These costs address things such as equipment, merchandising methods, real estate, personnel, supply availability, overhead, etc.

5. Prepare an Opening Day Balance Sheet

This includes an estimate of the assets and liabilities, one that should be as accurate as possible. To do this, create a list that includes items, sources, costs and available financing. Liabilities to consider are such things as leasing or purchasing of land, buildings and equipment, financing for assets and accounts receivables.

6. Review and Analyze All Data

All these steps are important, but the review and analysis are especially important to make sure that everything is as it should be and nothing requires changing or tweaking. So, take a moment to look over your work one last time.

Reexamine your previous steps, such as the income statement, and compare it with your expenses and liabilities. Is it still realistic? This is also the time to think about risk, analyzing and managing, and come up with any contingency plans.

7. Make a Go/No-Go Decision

You're now at the point to make a decision about whether the project is feasible or not. That sounds simple, but all the previous steps lead to this decision-making moment. A couple of other things to consider before making that binary choice is whether the commitment is worth the time, effort and money and is it aligned with the organization's strategic goals and long-term aspirations.

Importance of Feasibility Study

The importance of a feasibility study is based on organizational desire to "get it right" before committing resources, time, or budget. A feasibility study might uncover new ideas that could completely change a project's scope. It's best to make these determinations in advance, rather than to jump in and to learn that the project won't work. Conducting a feasibility study is always beneficial to the project as it gives you and other stakeholders a clear picture of the proposed project.

Below are some key benefits of conducting a feasibility study:

- Improves project teams' focus.
- Identifies new opportunities.
- Provides valuable information for a "go/no-go" decision.

- Narrows the business alternatives.
- Identifies a valid reason to undertake the project.
- Enhances the success rate by evaluating multiple parameters.
- Aids decision-making on the project.
- Identifies reasons not to proceed.

The 4 elements of a feasibility analysis

There are four main elements that go into a feasibility study: technical feasibility, financial feasibility, schedule feasibility, and operational feasibility. You may also see these referred to as the four types of feasibility studies, though most feasibility studies actually include a review of all four elements.

- **Technical feasibility**

A technical feasibility study reviews the technical resources available for project. This study determines if you have the right equipment, enough equipment, and the right technical knowledge to complete your project objectives. For example, if your project plan proposes creating 50,000 products per month, but you can only produce 30,000 products per month in your factories, this project isn't technically feasible.

In Sales and Service Management system we are using Angular as Front-End technology and Spring-boot as Back-End technology and MySQL database to store the data. All three technologies are leading in their category. Angular, Spring-boot and MySQL are open source technologies and having huge community support if one stuck in a problem.

The system is self-explaining and does not need any training. A system has been built by concentrating on GUI concepts, the application can also be handled very easily. The overall time that a user needs to get trained is less than 10-15 minutes.

- **Financial feasibility**

Financial feasibility describes whether or not your project is fiscally viable. A financial feasibility report includes a cost/benefit analysis of the project. It also forecasts an expected return on investment (ROI), as well as outlines any financial risks. The goal at the end of the financial feasibility study is to understand the economic benefits the project will drive.

- **Schedule Feasibility: (Timeline estimations and optimizing resources):**

The timeline of the entire development process of the project is roughly determined to be 6 months with the adjustments and added features of the web application being able to be go live on owner's approval.

- **Operational feasibility**

An operational feasibility study evaluates whether or not your organization is able to complete this project. This includes staffing requirements, organizational structure, and any

applicable legal requirements. At the end of the operational feasibility study, your team will have a sense of whether or not you have the resources, skills, and competencies to complete this work

2.2 Objectives of Proposed System

- To provide an online catalog to manage all visits, enquiries, tickets, customers, users, product details for client.
- To provide online platform that will help to admin and managers to assign various task for employee.
- To provide informational system that will help to client for knowing about his complete all types task.
- To provide better way to manage enquiries and enquiry details.
- To provide better way to manage visits and visit details.
- To provide better way to manage tickets and ticket details.
- To provide unique space to employee, in which he will know about his pending/assigned activities.
- To provide an efficient way of managing the records.
- To generate different kind of reports as and when required by the management.

2.3 Users of the System

There are total 5 roles in Sales and Service Management System

1) Admin
2) Sales Manager
3) Sales Engineer
4) Service Manager
5) Service Engineer

CHAPTER 3
ANALYSIS AND DESIGN

3.1 System Requirements-Functional and Non-Functional requirements

Functional Requirements:

A Functional Requirement is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements are also called Functional Specification. In software engineering and systems engineering, a Functional Requirement can range from the high-level abstract statement of the sender's necessity to detailed mathematical functional requirement specifications. Functional software requirements help you to capture the intended behavior of the system

1) Admin

- The system allows to admin to managing all task and task activities done by all employees.
- The system would be able to show all task information.
- The system would be able to show due soon task and running late task details automatically.
- The system can show numeric information or total counts about all task as like, total count of enquiries, tickets, visits by their statues like Unassigned, Assigned, Prospect, In Progress, Completed, Cancelled, won and lost.
- The system gives permission to admin to allocating new ticket/enquiry for any employee.
- The customer, users and master data related details are manages by admin. Such as action types, brands, common replies, departments, enquiry sources, enquiry types, employees, ticket types, product categories, product types and product.
- The employee details are manages by admin, such as employee’s personal details.

2) Sales Manager

- The system allows to Sales Manager to managing visits, enquiries and enquiries activities done by all employees.
- The system gives permission to Sales Manager to add new enquiries.
- The system gives permission to Sales Manager to assign new enquiry for employee.
- The system allows to Sales Manager to add new customer as a prospect only.
- The customer details are manages by Sales Manager. Such as client contact info, email and address details.
- The employee details are manages by Sales Manager, such as employee's assigned enquiries, work done by employee and pending visits, enquiries of employee.
- User can manage and update his own profile.
- The system allows to Sales Manager to take actions on own and his assigned sales engineer under the product category assigned to Sales Manager.

3) Sales Engineer

- The system allows to Sales Engineer to add visits, enquiries and enquiries.
- The system give permission to Sales Engineer to add new enquiries.
- The system gives permission to Sales Engineer to do updates in his assigned enquiries.
- The system allows to Sales Engineer to see only permissions and authorities related to him only.
- The system allows to Sales Engineer to add new customer as a prospect only.
- User can manage and update his own profile

4) Service Manager

- The system allows to Service Manager to managing tickets and tickets activities done by all employees under him.
- The system gives permission to tickets Manager to add new tickets.

- The system gives permission to Service Manager to assign new tickets for employee.
- The employee details are manages by Service Manager, such as employee's assigned tickets, work done by employee and pending tickets.
- User can manage and update his own profile.
- The system allows to Service Manager to take actions on own and his assigned Service engineer under the product category assigned to Service Manager.

5) Service Engineer

- The system allows to Service Engineer to add tickets.
- The system give permission to Service Engineer to add new tickets.
- The system gives permission to Service Engineer to do updates in his assigned tickets.
- The system allows to Service Engineer to see only permissions and authorities related to him only.
- User can manage and update his own profile.

6) Reports

The system should be allows only admin to generate reports.
Admin can generate reports like:

- **Ticket Reports**
  - ✓ Manager-wise Ticket Report
  - ✓ Employee-wise Ticket Report
  - ✓ Customer-wise Ticket Report
  - ✓ Product-wise Ticket Report
  - ✓ Ticket Type-wise Ticket Report
- **Enquiry Reports**
  - ✓ Customer-wise Enquiry Report
  - ✓ Product-wise Enquiry Report
  - ✓ Enquiry Source Efficacy Report

❖ **Charts**
- ✓ Product Type-wise Enquiry Report
- ✓ Product Type-wise Ticket Report
- ✓ Ticket Type-wise Ticket Report

Non Functional Requirements:

A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load? A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

1) Security
- Each member is required to have an individual password.
- Administrators have the option of increasing the level of password security their members must use
- For maximum security, each member must protect their password.

2) Reliability
- System will prompt the user if any incorrect input is made.
- To handle data consistency, DBMS software is used.
- User can easily work through the different menus and buttons.

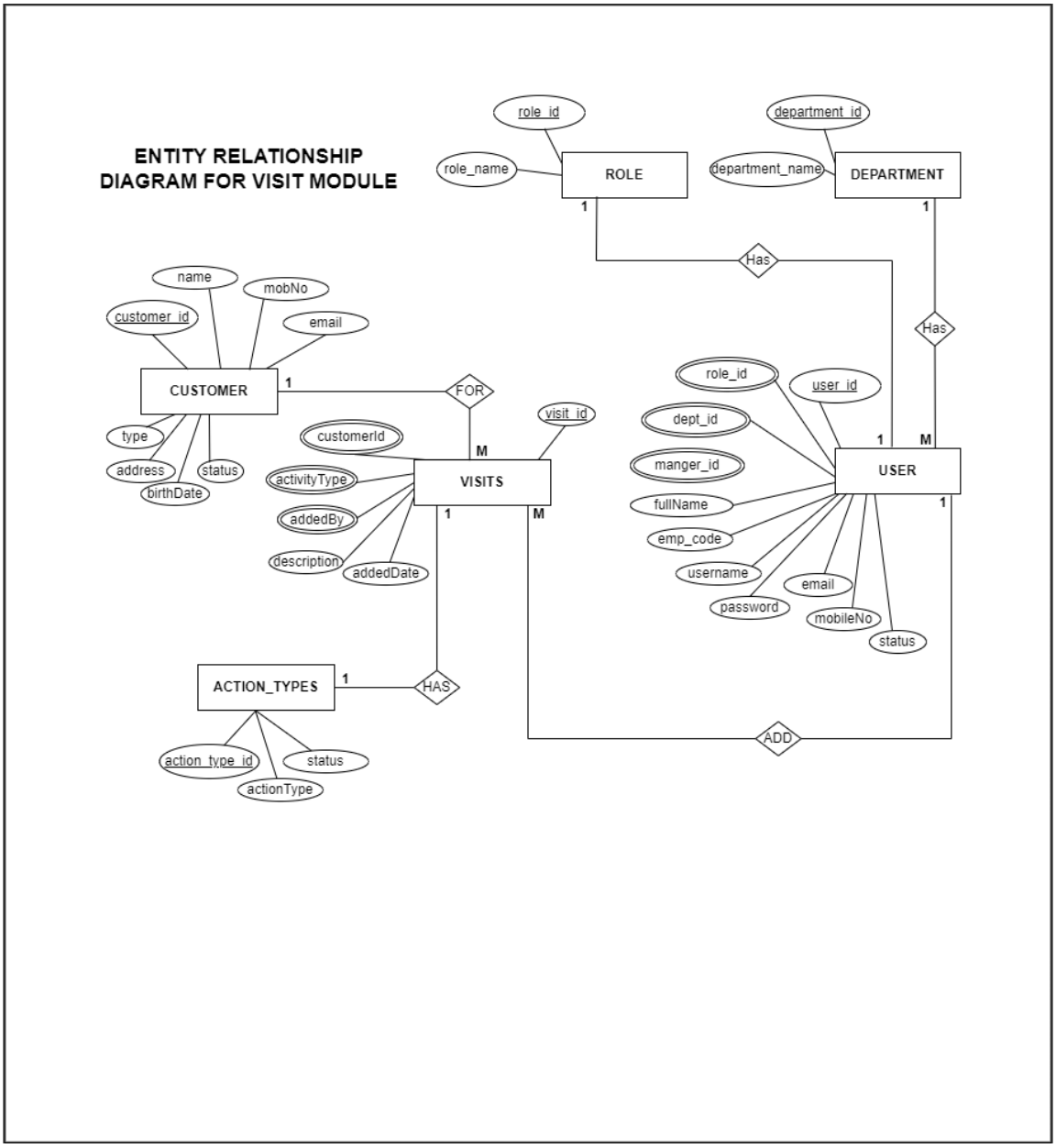
3) Portability
- System is independent of hardware specification.
- It can run on any operating system.
- System is independent of browser compatibility

4) Performance
- The performance of our product is at its best if stored locally, as the response time will be much faster.
- If the product accessed via Internet, the performance is limited by the connection speed.
- The only foreseen limitation is that of web server response.

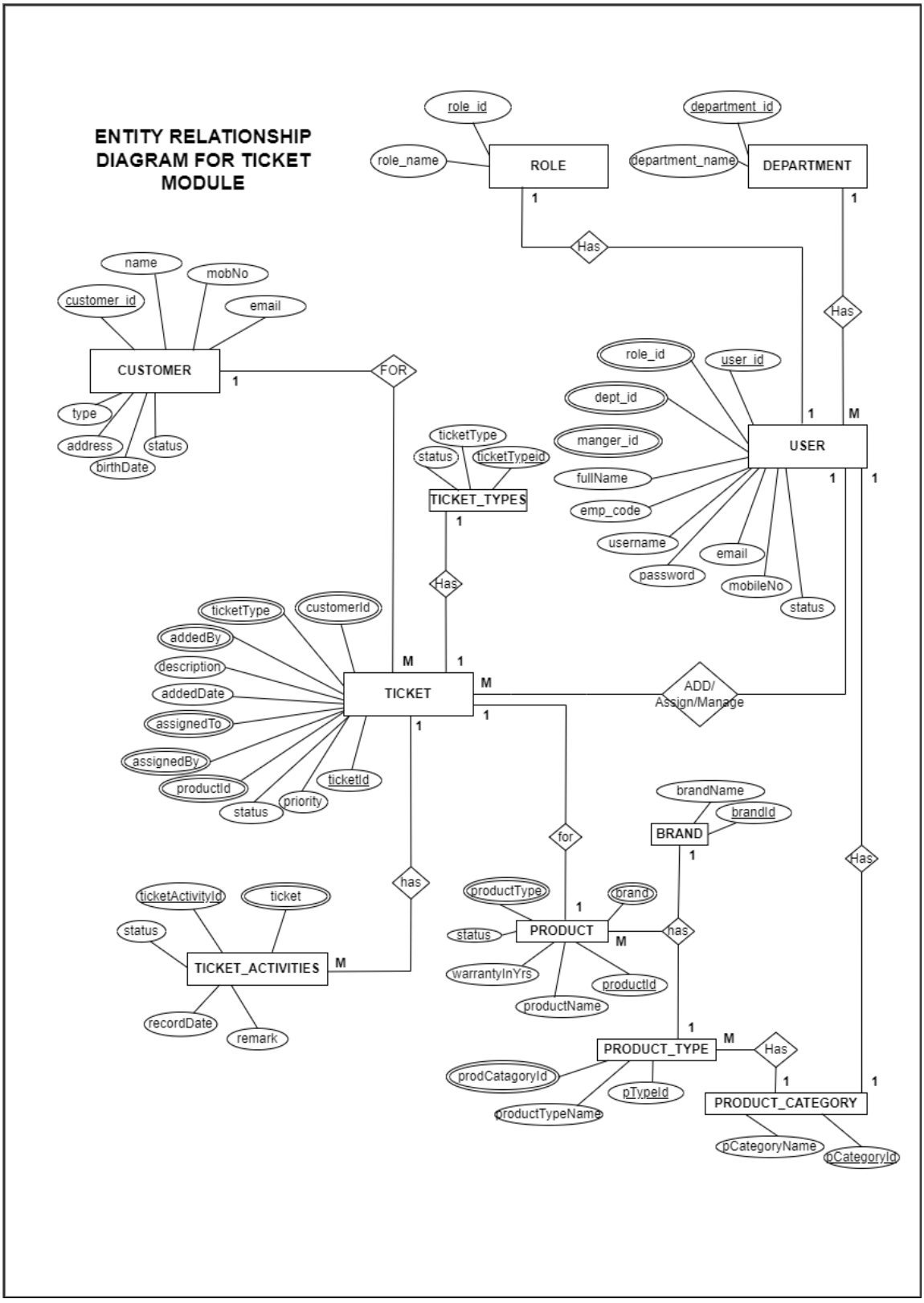
5) User Friendly
- Our system is very easy to use and user friendly. Its GUI is very attractive and understandable to the common users.

3.2 Entity Relationship Diagrams (ERD)

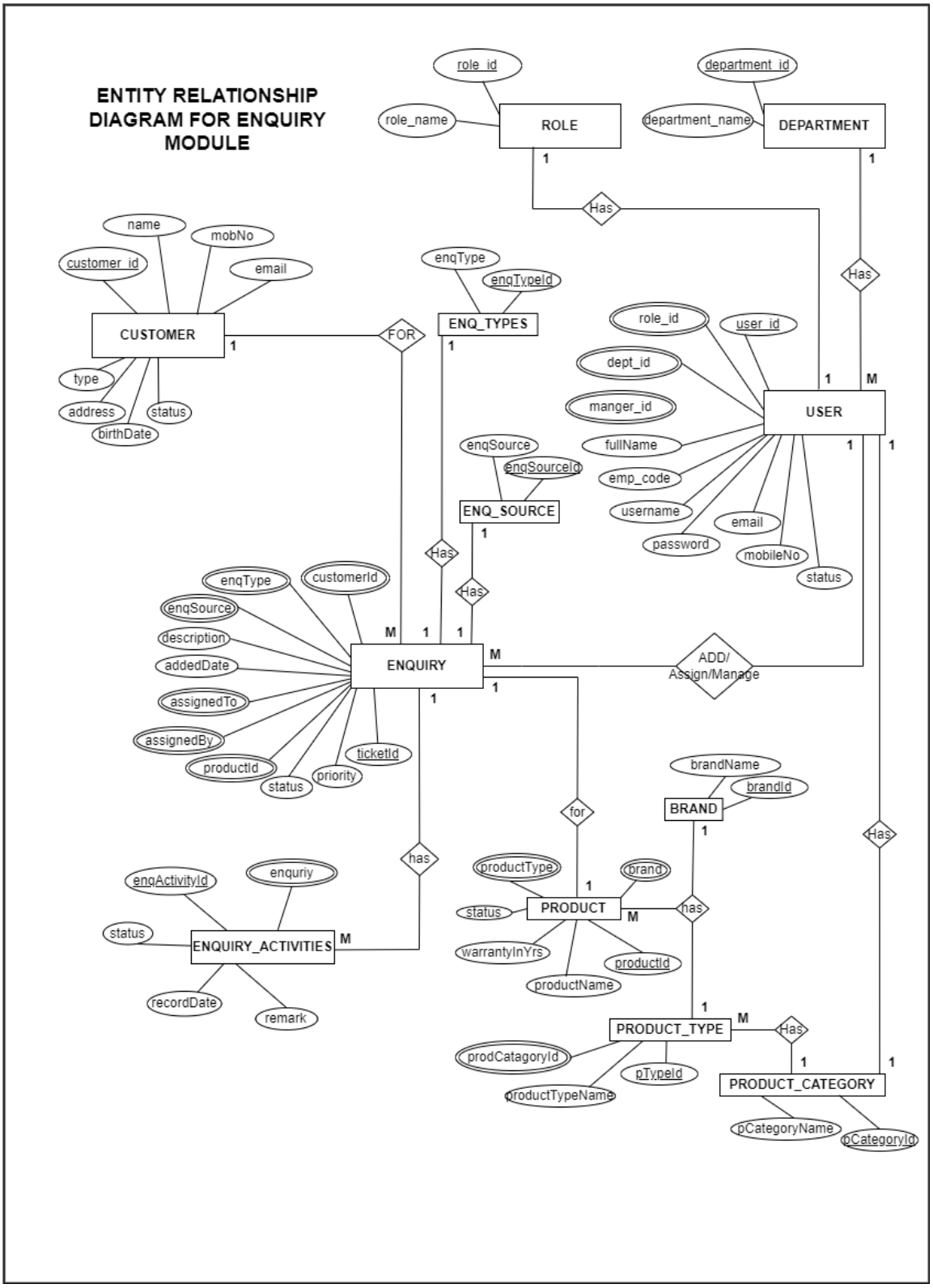
Visit Module:



Ticket Module:



Enquiry Module:



3.3 Table Structure

Abbreviations:

Sr. No	Abbreviation	Description
1	PK	Primary Key
2	FK	Foreign Key
3	AI	Auto Increment
4	NN	Not NULL
5	AN	Allow NULL

Data Dictionary:

Field Name	Data type	Size	Constraints	Description
action_type_id	int	10	NN AI	Unique identification of action type.
actionType	varchar	45	NN	Stores name of the action type.
added_by	int	10	NN	Stores the reference of the user who added the ticket
addedBy	int	10	NN	Stores the reference of user for the visit who added the visit.
addedDate	Timestamp	-	NN	Stores the creation time of the visit.
address	varchar	255	NN	Stores the address of customer.
birthDate	varchar	15	AN	Stores the birth date of the customer.
brand_id	int	10	NN AI	Unique identification of brand.
brandName	varchar	45	NN	Stores name of the brand.
common_reply	varchar	45	NN	Stores name of the brand.
common_reply_id	int	10	NN AI	Unique identification of common reply.
customer_email	varchar	10	NN	Stores the email of the customer
customer_id	int	10	NN AI	Unique identification of customer
customer_name	varchar	10	NN	Stores the name of the customer
customer_product_link_id	int	10	NN AI	Unique identification of Customer Product Link.
customer_type	varchar	45	NN	Stores the customer type.

date_of_purchase	Timestamp	-	NN	Store the date of purchase of product purchased by customer
department_id	int	10	NN AI	Unique identification of department
department_name	varchar	45	NN	Stores name of the department
description	varchar	255	NN	Stores the description of visit.
display_color	varchar	45	NN	Stores display color information for enquiry type.
email	varchar	45	AN	Stores the email of the user.
emp_code	varchar	45	NN	Stores the employee Code of the user.
enquiry_id	int	10	NN AI	Unique identification of enquiry.
enquiry_source	varchar	45	NN	Stores name of the enquiry source.
enquiry_source_id	int	10	NN AI	Unique identification of enquiry source.
enquiry_type	varchar	45	NN	Stores name of the enquiry type.
enquiry_type_id	int	10	NN AI	Unique identification of enquiry type.
full_name	varchar	45	NN	Stores the full name of user.
is_in_warranty	Boolean	1	AN	Stores whether product is in warranty or not
joining_date	Timestamp	-	AN	Stores the joining date of the user.
last_service_date	Timestamp	-	NN	Stores the product last service date
long_description	varchar	255	NN	Stores the long description of ticket
machine_serial_number	varchar	45	NN	Stores the products machine serial number
machine_serial_number	varchar	45	NN	Stores the machine serial number for the ticket.
manager_id	int	10	AN	Stores the manager information of the user.
mobile_number	varchar	15	NN	Stores the mobile no of the user.
notes	varchar	255	AN	Stores the notes.
password	varchar	255	NN	Stores the password of the user.

priority	varchar	45	NN	Stores the Ticket priority
product_category_id	int	10	NN AI	Unique identification of product category.
product_category_name	varchar	45	NN	Stores name of the enquiry source.
product_category_status	boolean	1	NN	Stores the product category status
product_id	int	10	NN AI	Unique identification of product.
product_name	varchar	255	NN	Stores the product name.
product_remark	varchar	50	AN	Stores the status of product.
product_type_name	varchar	45	NN	Stores the product type name
profile_pic	varchar	100	AN	Stores the profile picture path of the user.
quantity	int	10	NN	Stores the quantity of products.
recent_activity_date	Timestamp	-	NN	Stores the recent activity date of the enquiry.
record_date	Timestamp	-	NN	Stores the record date
remark	varchar	100	AN	Stores the remark for enquiry.
role_id	int	10	NN AI	Unique identification of role
role_name	varchar	45	NN	Stores name of the role
send_email	boolean	1	AN	Stores the flag for sending email
send_sms	boolean	1	AN	Stores the flag for sending sms
short_description	varchar	45	NN	Stores the short description about ticket
status	boolean	1	NN	Stores the status of the user.
ticket_id	int	10	NN AI	Unique identification of Ticket.
ticket_prefix	varchar	45	NN	Stores the ticket prefix for product type.
ticket_type	varchar	45	NN	Stores name of the ticket type.
ticket_type_id	int	10	NN AI	Unique identification of Ticket Type.
user_id	int	10	NN AI	Unique identification of user.
user_product_category_link_id	int	10	NN AI	Unique identification of User Product Category Link.
username	varchar	45	NN	Stores the username of the user.
visit_id	int	10	NN AI	Unique identification of visit.

warranty_date	Timestamp	-	AN	Stores the product warranty date
warranty_in_years	int	11	AN	Stores the warranty of products in years.
warranty_till	Timestamp	-	NN	Store the date of warranty till of product purchased by customer

1. Roles

Table Name	Roles			
Description	This table contains all the roles.			
Primary Key	role_id			
Field Name	**Data type**	**Size**	**Constraints**	**Description**
role_id	int	10	NN AI	Unique identification of role
role_name	varchar	45	NN	Stores name of the role

2. Department

Table Name	Department			
Description	This table contains all the departments.			
Primary Key	department_id			
Field Name	**Data type**	**Size**	**Constraints**	**Description**
department_id	int	10	NN AI	Unique identification of department
department_name	varchar	45	NN	Stores name of the department

3. Users

Table Name	Users			
Description	This table contains all the data of users.			
Primary Key	user_id			
Foreign Keys	manager_id, department_id, role_id			
Field Name	**Data type**	**Size**	**Constraints**	**Description**
user_id	int	10	NN AI	Unique identification of user.
manager_id	int	10	AN	Stores the manager information of the user.
department_id	int	10	NN	Stores the reference of the department for user.

role_id	int	10	NN	Stores the reference of the role for user.
full_name	varchar	45	NN	Stores the full name of user.
emp_code	varchar	45	NN	Stores the employee Code of the user.
username	varchar	45	NN	Stores the username of the user.
password	varchar	255	NN	Stores the password of the user.
email	varchar	45	AN	Stores the email of the user.
mobile_number	varchar	15	NN	Stores the mobile no of the user.
joining_date	Timestamp	-	AN	Stores the joining date of the user.
profile_pic	varchar	100	AN	Stores the profile picture path of the user.
status	boolean	1	NN	Stores the status of the user.

4. Customers

Table Name	Customers			
Description	This table contains all the data of customers.			
Primary Key	customer_id			
Field Name	**Data type**	**Size**	**Constraints**	**Description**
customer_id	int	10	NN AI	Unique identification of customer
customer_name	varchar	10	NN	Stores the name of the customer
customer_email	varchar	10	NN	Stores the email of the customer
mobile_number	varchar	15	NN	Stores the mobile number of customer.
address	varchar	255	NN	Stores the address of customer.
customer_type	varchar	45	NN	Stores the customer type.
send_sms	boolean	1	AN	Stores the flag for sending sms
send_email	boolean	1	AN	Stores the flag for sending email
birthDate	varchar	15	AN	Stores the birth date of the customer.
status	varchar	10	AN	Stores the status of the customer

5. Action Types

Table Name	action_types
Description	This table contains all the action types.
Primary Key	action_type_id

Field Name	**Data type**	**Size**	**Constraints**	**Description**
action_type_id	int	10	NN AI	Unique identification of action type.
actionType	varchar	45	NN	Stores name of the action type.
status	varchar	45	NN	Stores status of action type.

6. Visits

Table Name	visits
Description	This table contains all the data of customer visits.
Primary Key	visit_id
Foreign Keys	customer_id, action_type_id, addedBy

Field Name	**Data type**	**Size**	**Constraints**	**Description**
visit_id	int	10	NN AI	Unique identification of visit.
customer_id	int	10	NN	Stores the customer reference for the visit.
action_type_id	int	10	NN	Stores the action type reference for the visit
description	varchar	255	NN	Stores the description of visit.
addedBy	int	10	NN	Stores the reference of user for the visit who added the visit.
addedDate	Timestamp	-	NN	Stores the creation time of the visit.

7. Enquiry Source

Table Name	enquiry_sources
Description	This table contains all the enquiry sources.
Primary Key	enquiry_source_id

Field Name	**Data type**	**Size**	**Constraints**	**Description**
enquiry_source_id	int	10	NN AI	Unique identification of enquiry source.
enquiry_source	varchar	45	NN	Stores name of the enquiry source.

8. Enquiry Type

Table Name	enquiry_types
Description	This table contains all the enquiry types.

Primary Key	enquiry_type_id			
Field Name	**Data type**	**Size**	**Constraints**	**Description**
enquiry_type_id	int	10	NN AI	Unique identification of enquiry type.
enquiry_type	varchar	45	NN	Stores name of the enquiry type.
display_color	varchar	45	NN	Stores display color information for enquiry type.

9. Product Category

Table Name	product_category			
Description	This table contains all the product categories.			
Primary Key	product_category_id			
Field Name	**Data type**	**Size**	**Constraints**	**Description**
product_category_id	int	10	NN AI	Unique identification of product category.
product_category_name	varchar	45	NN	Stores name of the enquiry source.
product_category_status	boolean	1	NN	Stores the product category status

10. Product Types

Table Name	product_types			
Description	This table contains all the product types.			
Primary Key	product_type_id			
Foreign Keys	product_category_id			
Field Name	**Data type**	**Size**	**Constraints**	**Description**
product_type_id	int	10	NN AI	Unique identification of product types.
product_category_id	int	10	NN	Stores the reference of the product category.
product_type_name	varchar	45	NN	Stores the product type name
ticket_prefix	varchar	45	NN	Stores the ticket prefix for product type.
notes	varchar	255	AN	Stores the notes.

status	boolean	1	NN	Stores the status of product type.

11. Brands

Table Name	brands			
Description	This table contains all the brands.			
Primary Key	brand_id			
Field Name	**Data type**	**Size**	**Constraints**	**Description**
brand_id	int	10	NN AI	Unique identification of brand.
brandName	varchar	45	NN	Stores name of the brand.
status	varchar	45	NN	Stores status of brand.

12. Products

Table Name	products			
Description	This table contains all the products.			
Primary Key	product_id			
Foreign Keys	product_type_id, brand_id			
Field Name	**Data type**	**Size**	**Constraints**	**Description**
product_id	int	10	NN AI	Unique identification of product.
product_type_id	int	10	NN	Stores the reference of the product type.
brand_id	int	10	NN	Stores the reference of the brand.
product_name	varchar	255	NN	Stores the product name.
ticket_prefix	varchar	45	NN	Stores the ticket prefix for product type.
warranty_in_years	int	11	AN	Stores the warranty of products in years.
status	boolean	1	NN	Stores the status of product.

13. User Product Category Link

Table Name	user_product_category_link
Description	This table contains all the data of which user is mapped to the which product category.
Primary Key	user_product_category_link_id

Foreign Keys	product_category_id, user_id

Field Name	**Data type**	**Size**	**Constraints**	**Description**
user_product_category_link_id	int	10	NN AI	Unique identification of User Product Category Link.
product_category_id	int	10	NN	Stores the reference of the product category.
user_id	int	10	NN	Stores the reference of the user.
status	boolean	1	NN	Stores the status of User Product category Link.

14. Common Replies

Table Name	common_replies
Description	This table contains all the common replies.
Primary Key	common_reply_id

Field Name	**Data type**	**Size**	**Constraints**	**Description**
common_reply_id	int	10	NN AI	Unique identification of common reply.
common_reply	varchar	45	NN	Stores name of the brand.

15. Enquiries

Table Name	enquiries
Description	This table contains all the enquiries.
Primary Key	enquiry_id
Foreign Keys	customer_id, enquiry_source_id, enquiry_type_id, product_id, added_By, assigned_to, assigned_by

Field Name	**Data type**	**Size**	**Constraints**	**Description**
enquiry_id	int	10	NN AI	Unique identification of enquiry.
customer_id	int	10	NN	Stores the reference of the customer.

enquiry_source_id	int	10	NN	Stores the reference of the enquiry source.
enquiry_type_id	int	10	NN	Stores the reference of the enquiry type.
product_id	int	10	NN	Stores the reference of the product.
quantity	int	10	NN	Stores the quantity of products.
product_remark	varchar	50	AN	Stores the status of product.
added_by	int	10	NN	Stores the reference of the user who added the enquiry.
added_date	Timestamp	-	NN	Stores the added date of enquiry.
recent_activity_date	Timestamp	-	NN	Stores the recent activity date of the enquiry.
remark	varchar	100	AN	Stores the remark for enquiry.
status	varchar	45	NN	Stores the status of the enquiry.
assigned_to	int	10	AN	Stores the reference of the user to whom enquiry is assigned.
assigned_by	int	10	AN	Stores the reference of the user to whom the enquiry is assigned.

16. Customer Product Link

Table Name	customer_product_link
Description	This table contains all the data of customers and their purchased products.
Primary Key	customer_product_link_id
Foreign Keys	customer_id, product_category_id, product_type_id, product_id, brand_id

Field Name	**Data type**	**Size**	**Constraints**	**Description**
customer_product_link_id	int	10	NN AI	Unique identification of Customer Product Link.
product_category_id	int	10	NN	Stores the reference of the product category.
product_type_id	int	10	NN	Stores the reference of the product type.

brand_id	int	10	NN	Stores the reference of the brand
product_id	int	10	NN	Store the reference of the product purchased by customer
date_of_purchase	Timestamp	-	NN	Store the date of purchase of product purchased by customer
warranty_till	Timestamp	-	NN	Store the date of warranty till of product purchased by customer
quantity	int	10	NN	Store the quantity of product purchased by customer
machine_serial_number	varchar	45	NN	Stores the products machine serial number
record_date	Timestamp	-	NN	Stores the record date
remark	varchar	255	NN	Stores the remak

17. Ticket Types

Table Name	ticket_types			
Description	This table contains all the ticket types.			
Primary Key	ticket_type_id			
Field Name	**Data type**	**Size**	**Constraints**	**Description**
ticket_type_id	int	10	NN AI	Unique identification of Ticket Type.
ticket_type	varchar	45	NN	Stores name of the ticket type.
status	varchar	45	NN	Stores status of ticket types

18. Tickets

Table Name	tickets			
Description	This table contains all the tickes.			
Primary Key	ticket_id			
Foreign Key	ticket_type_id, customer_id, product_id, added_by, assigned_by, assigned_to			
Field Name	**Data type**	**Size**	**Constraints**	**Description**
ticket_id	int	10	NN AI	Unique identification of Ticket.

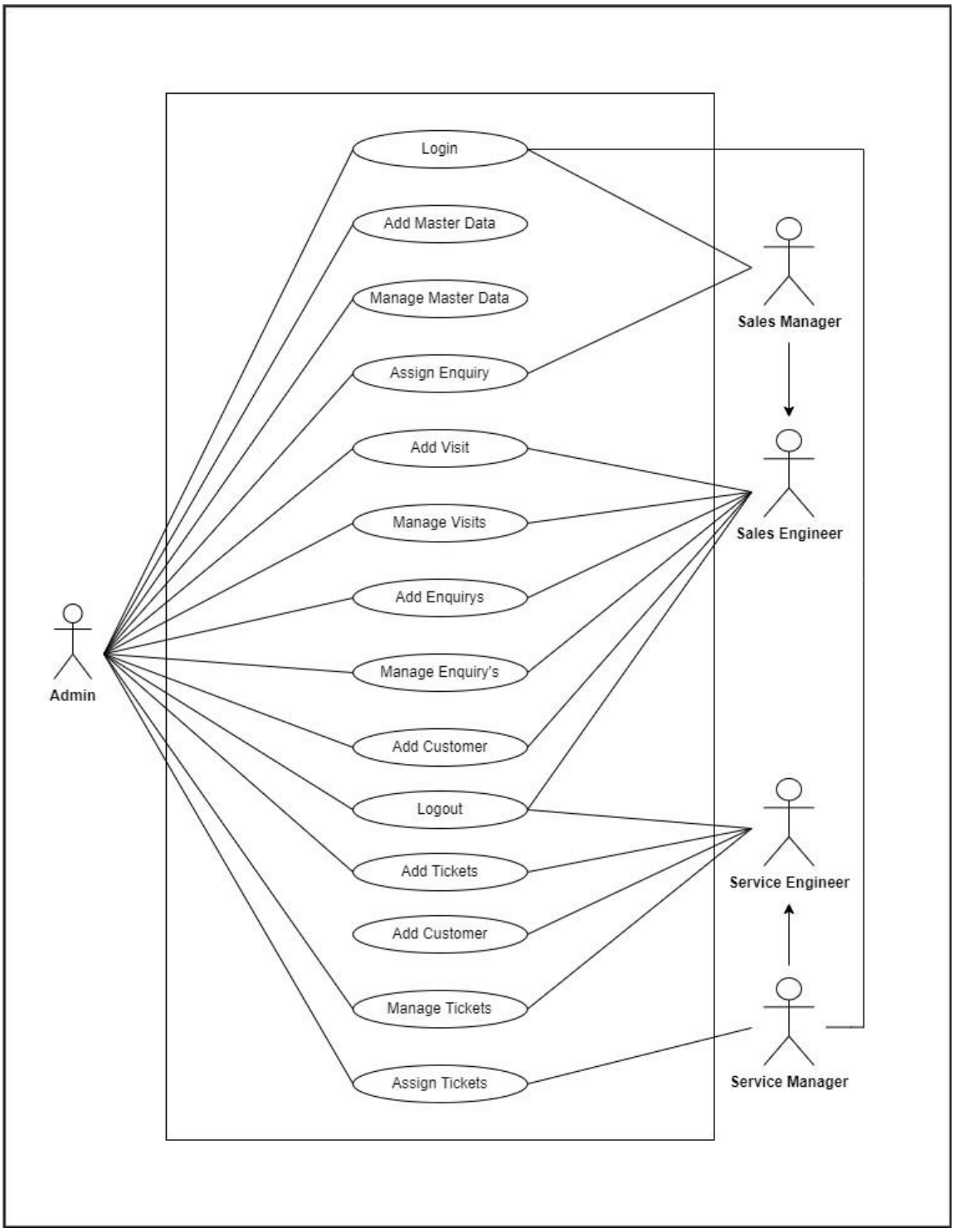
ticket_type_id	int	10	NN	Stores reference of the ticket type.
customer_id	int	10	NN	Stores reference of the customer
product_id	int	10	NN	Stores reference of the product
priority	varchar	45	NN	Stores the Ticket priority
last_service_date	Timestamp	-	NN	Stores the product last service date
warranty_date	Timestamp	-	AN	Stores the product warranty date
is_in_warranty	Boolean	1	AN	Stores whether product is in warranty or not
machine_serial_number	varchar	45	NN	Stores the machine serial number for the ticket.
short_description	varchar	45	NN	Stores the short description about ticket
long_description	varchar	255	NN	Stores the long description of ticket
added_date	Timestamp	-	NN	Stores the added date of the ticket
recent_activity_date	Timestamp	-	NN	Stores the recent activity done date for the ticket
status		45	NN	Stores the status of the ticket
added_by	int	10	NN	Stores the reference of the user who added the ticket
assigned_by	int	10	AN	Stores the reference of the user who assigned the ticket
assigned_to	int	10	AN	Stores the reference of the user by whom ticket is assigned.

19. Ticket Activities

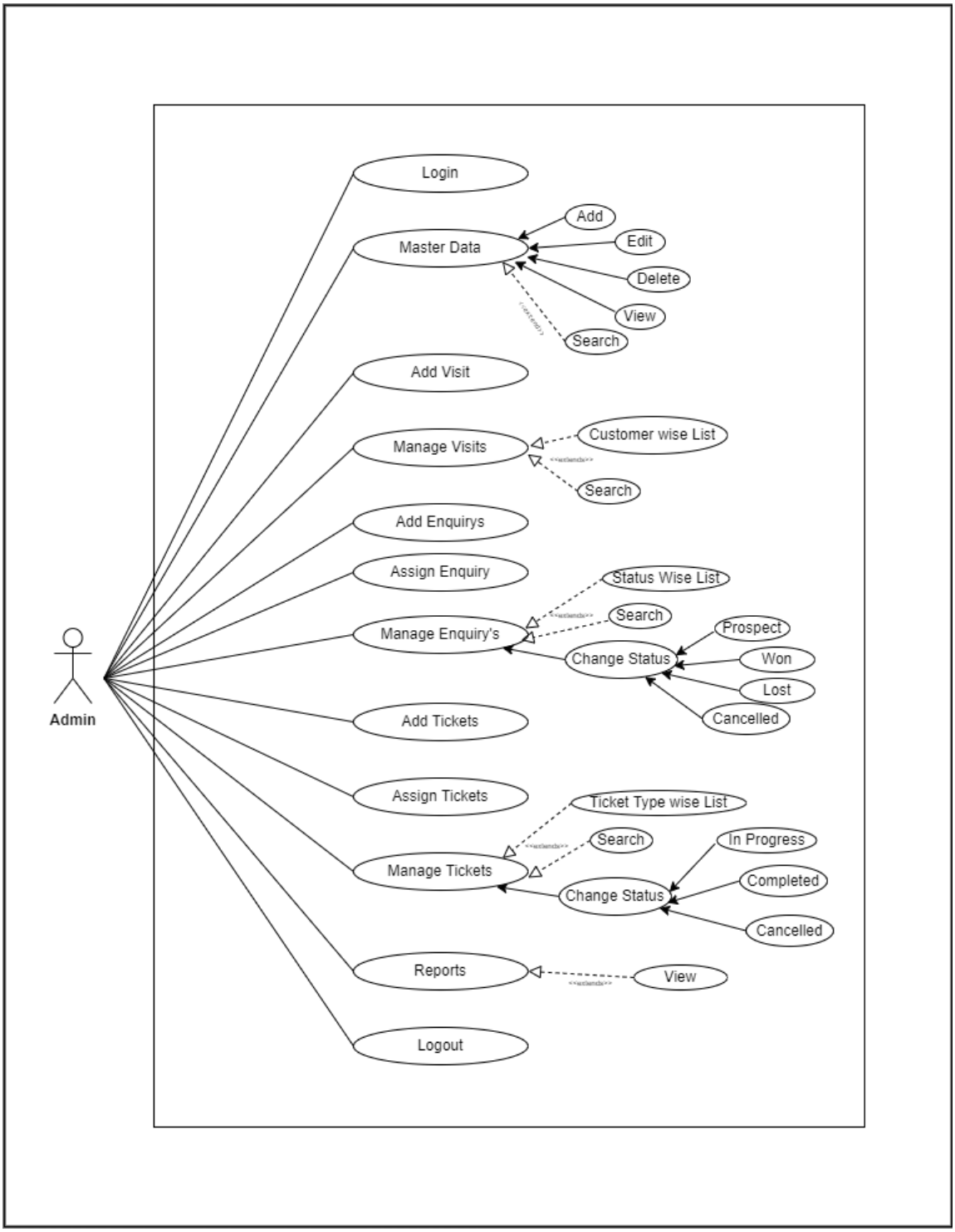
Table Name	ticket_activities			
Description	This table contains all the ticket activities.			
Primary Key	ticket_activity_id			
Foreign Key	ticket_id, user_id			
Field Name	**Data type**	**Size**	**Constraints**	**Description**
ticket_actvity_id	int	10	NN AI	Unique identification of Ticket Activity.
ticket_id	int	10	NN	Stores the reference of the ticket
user_id	int	10	NN	Stores the reference of the user who done this activity.
status	varchar	45	NN	Stores the status of the ticket
record_date	Timestamp	-	NN	Stores the record date
remark	varchar	100	NN	Stores the remark.

3.4 Use Case Diagrams

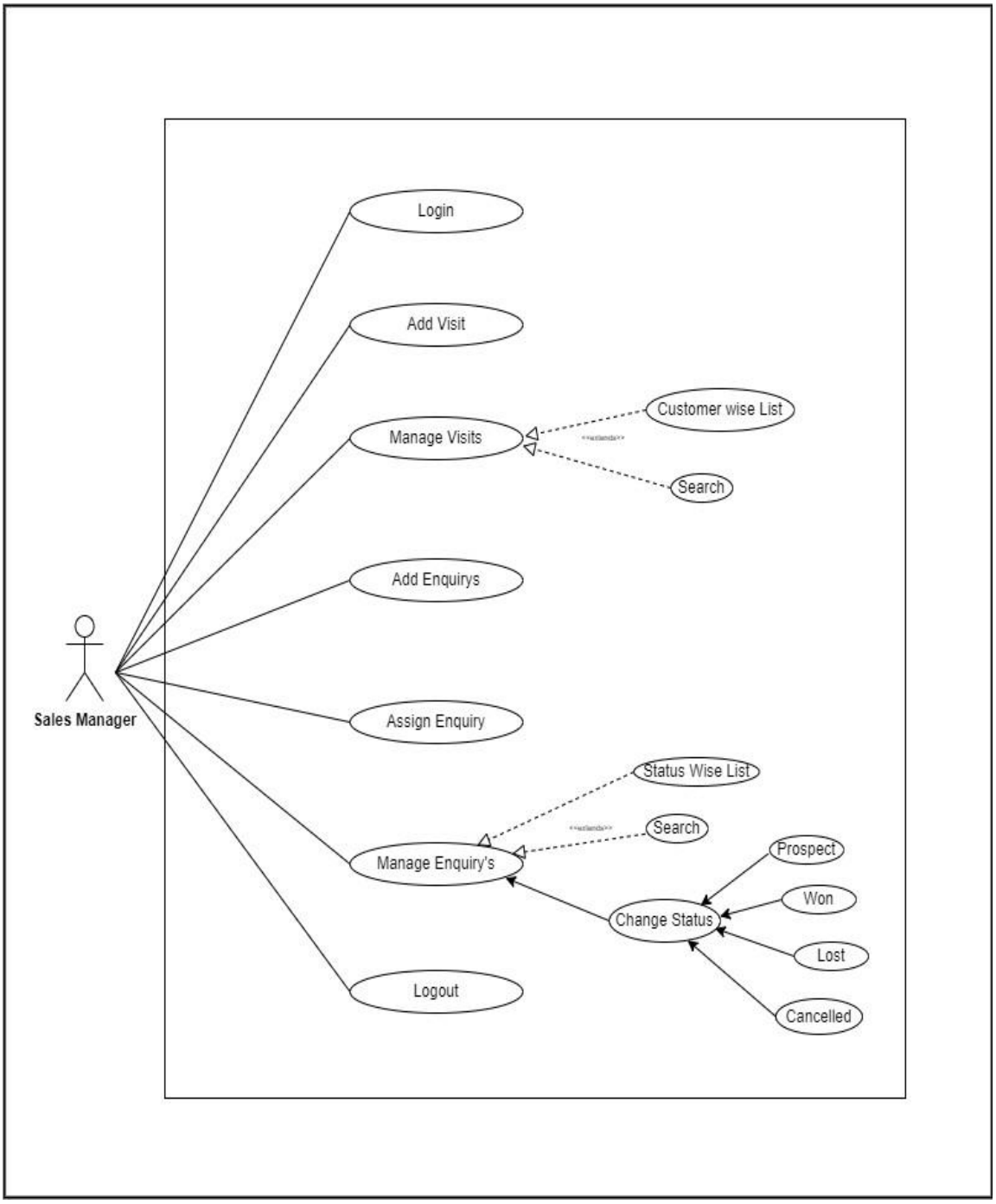
1. Global Use Case



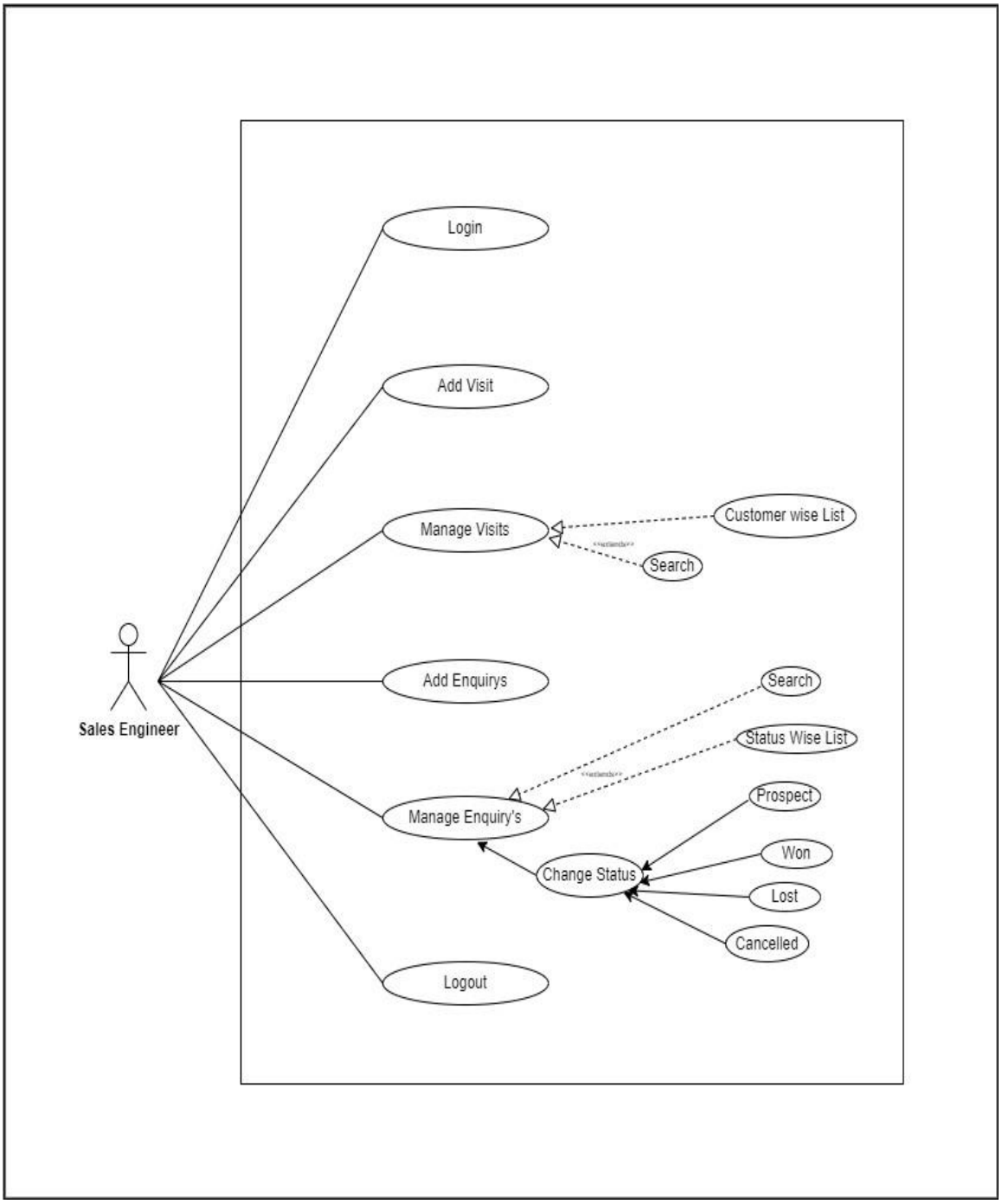
2. Use Case Diagram – Admin Module



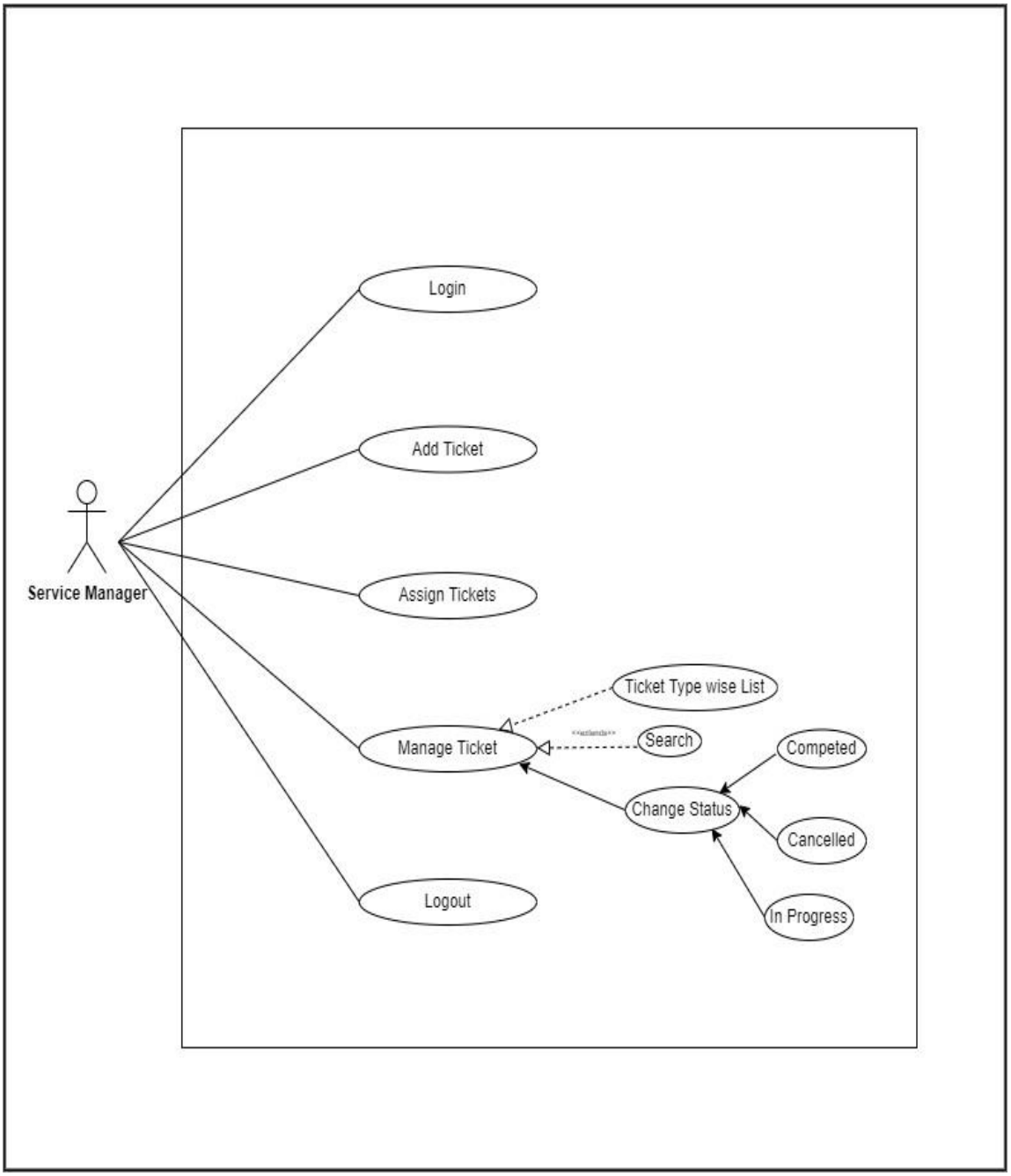
3. Use Case Diagram – Sales Manager



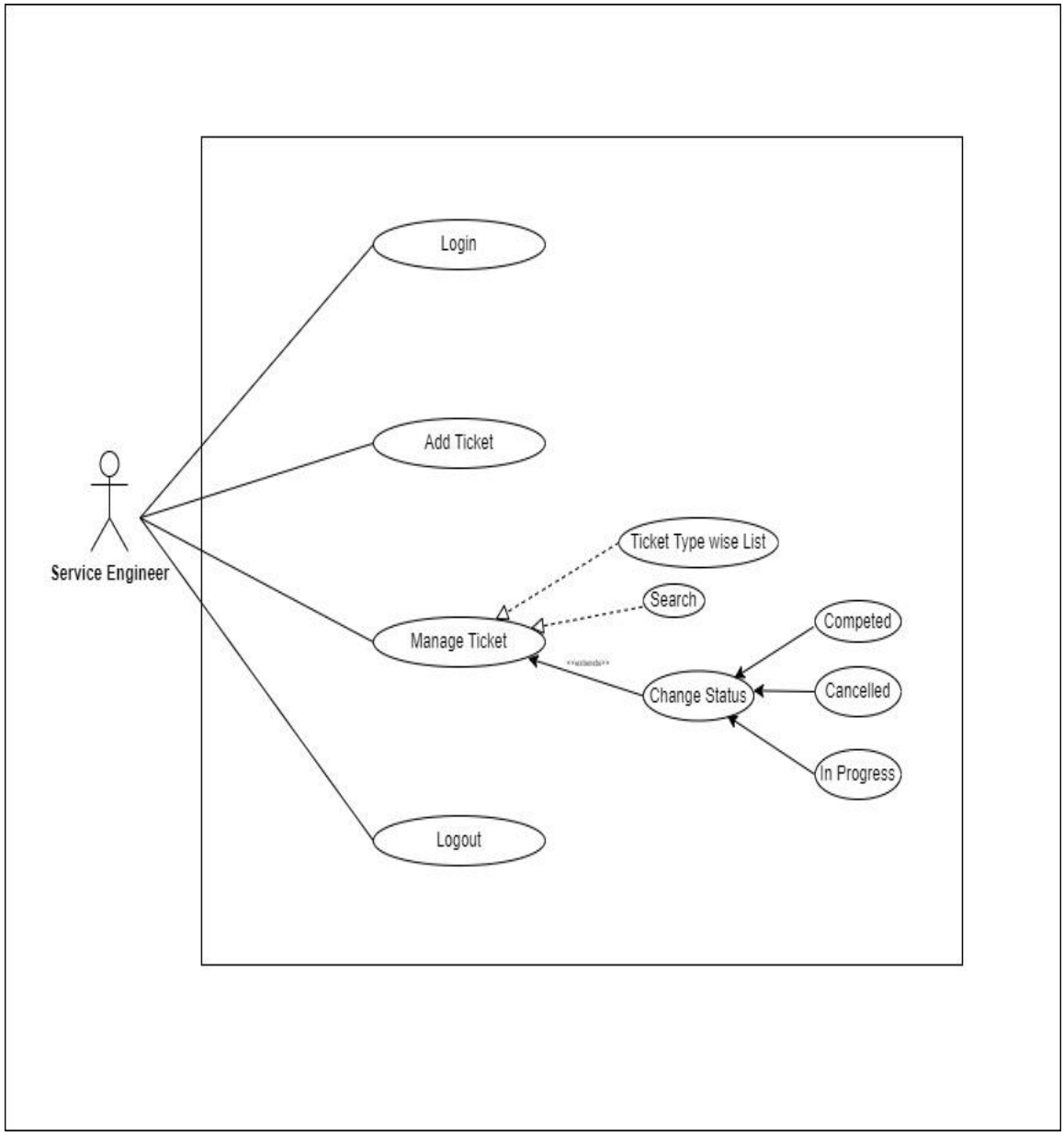
4. Use case diagram – Sales Engineer



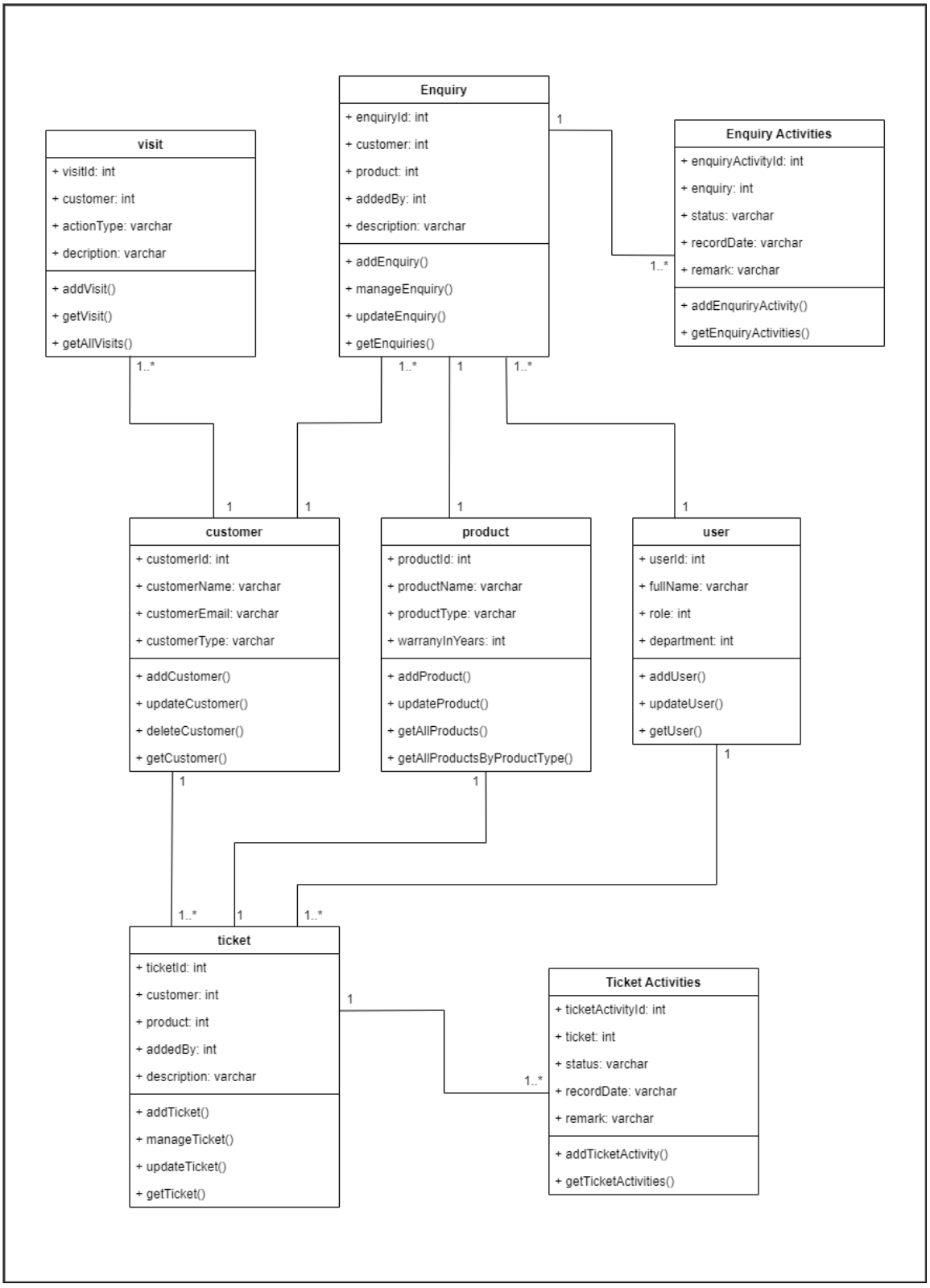
5. Use case diagram – Service Manager



6. Use Case Diagram – Service Engineer

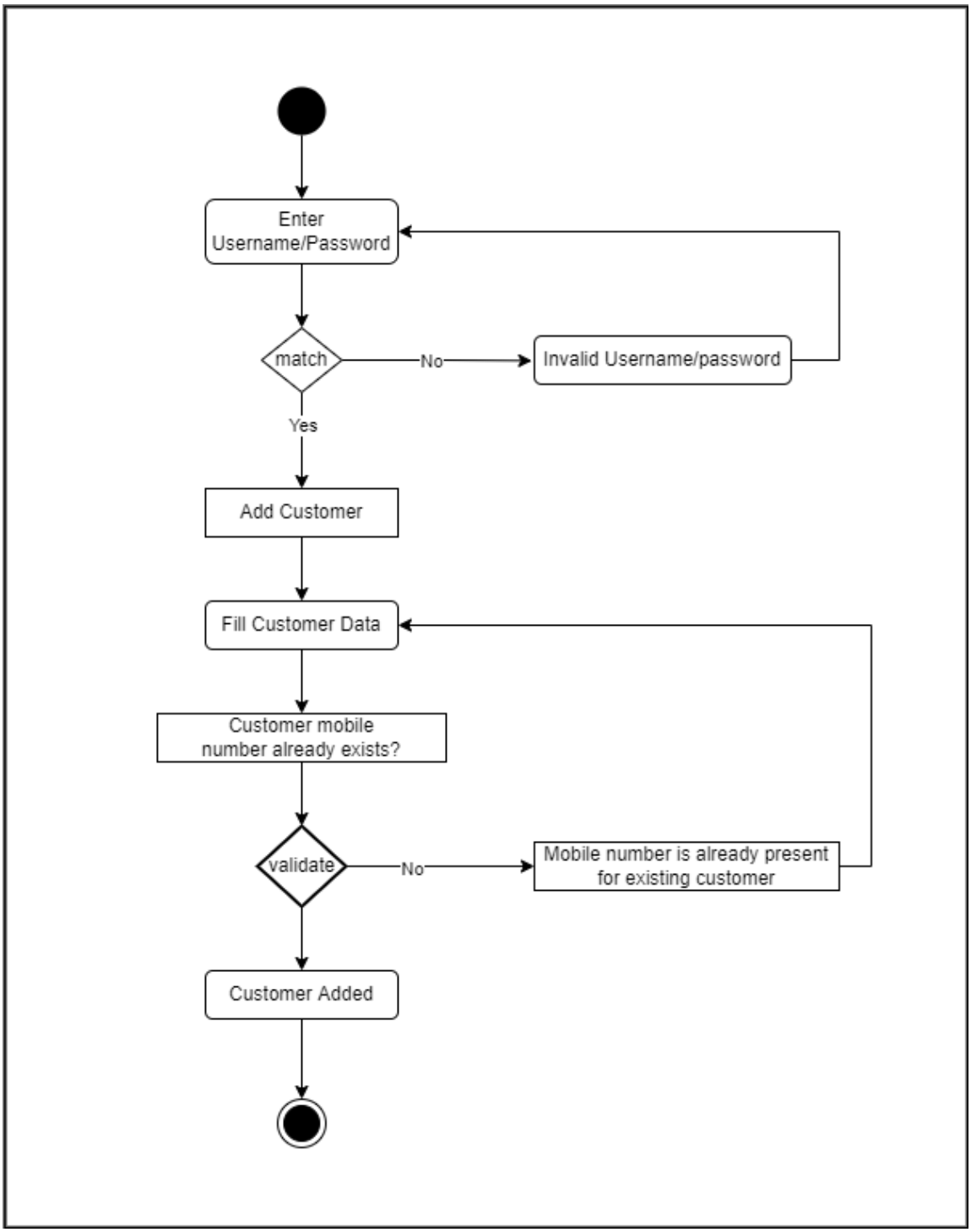


3.5 Class Diagram

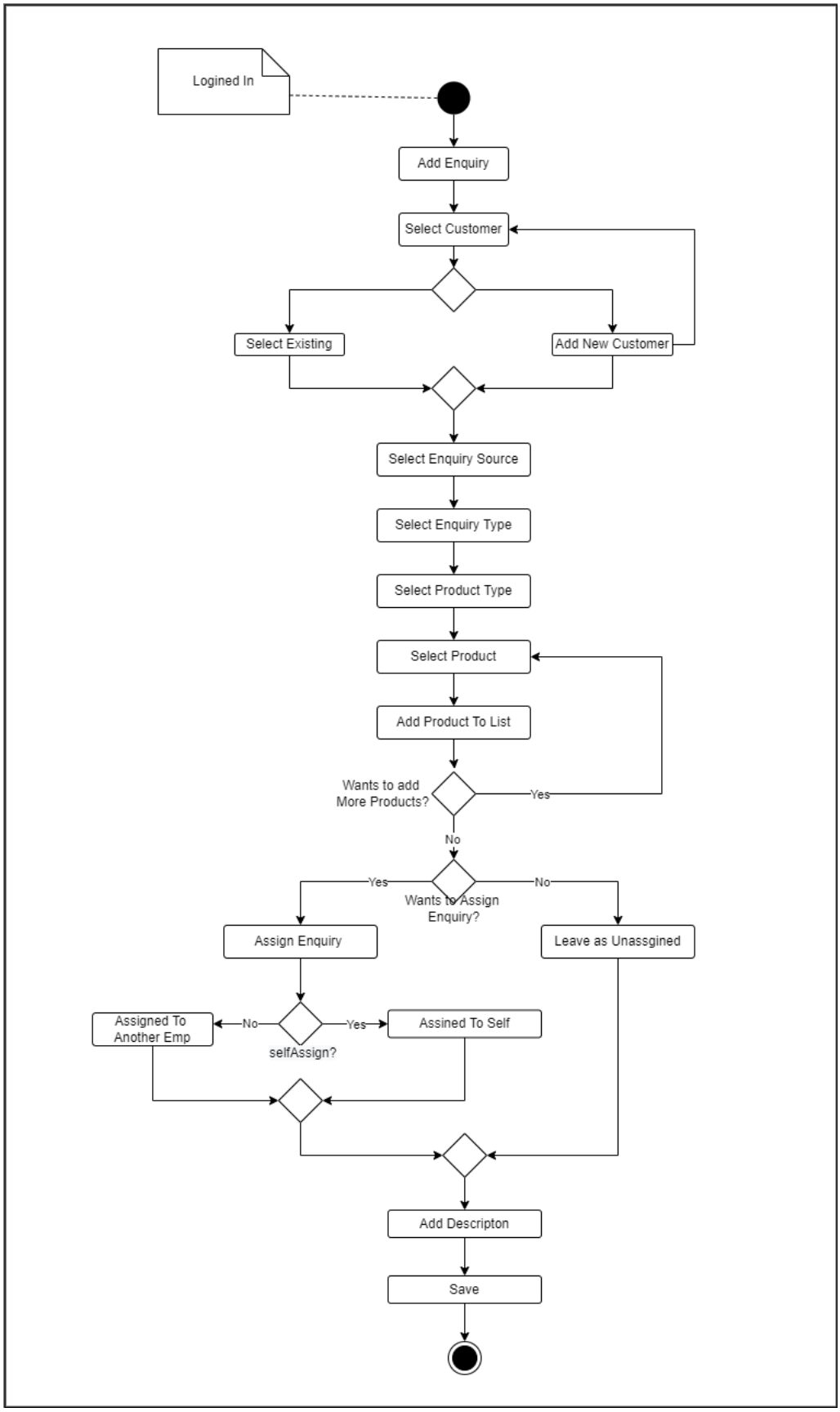


3.6 Activity Diagram

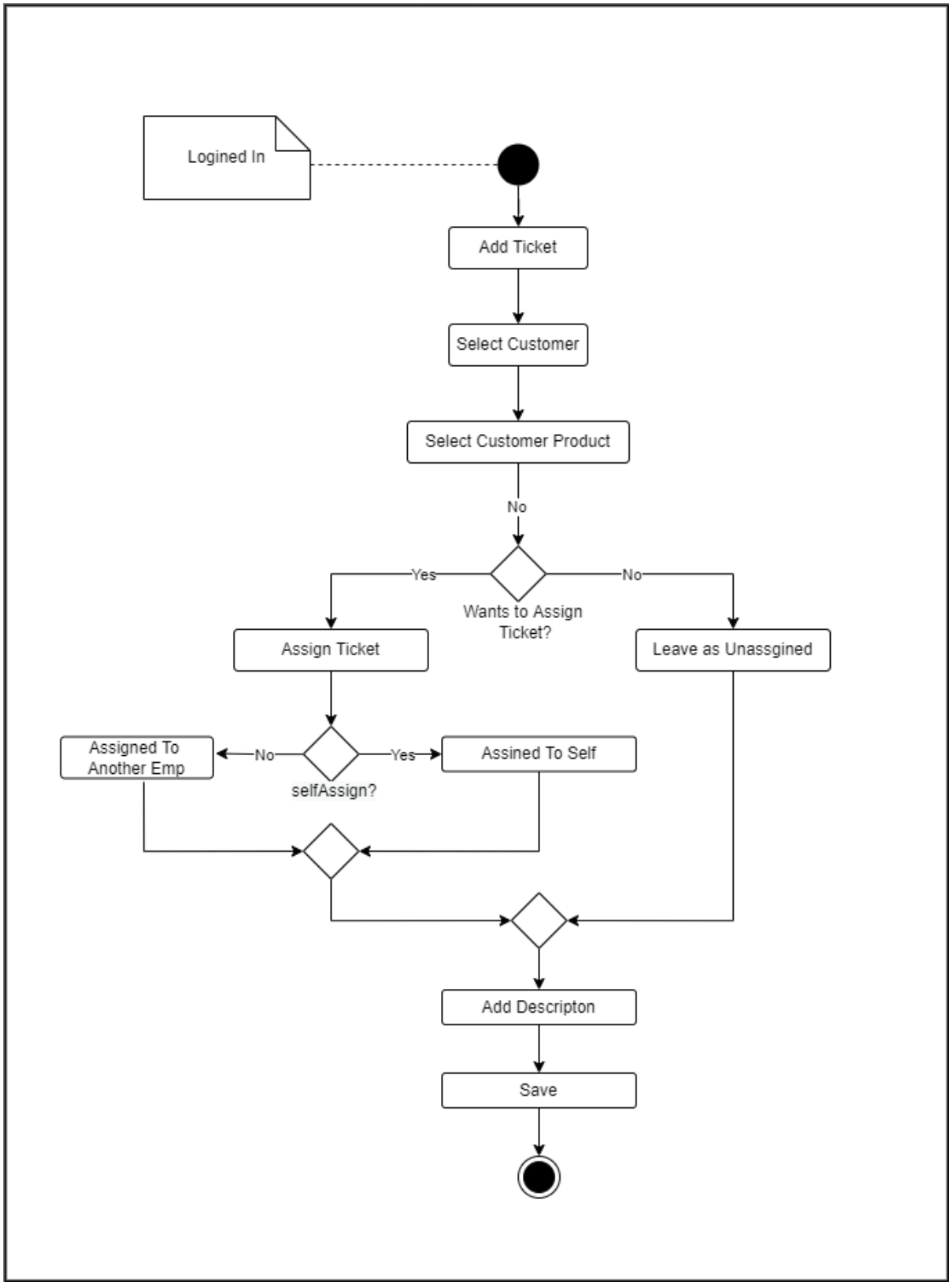
1. Activity diagram for Add Customer



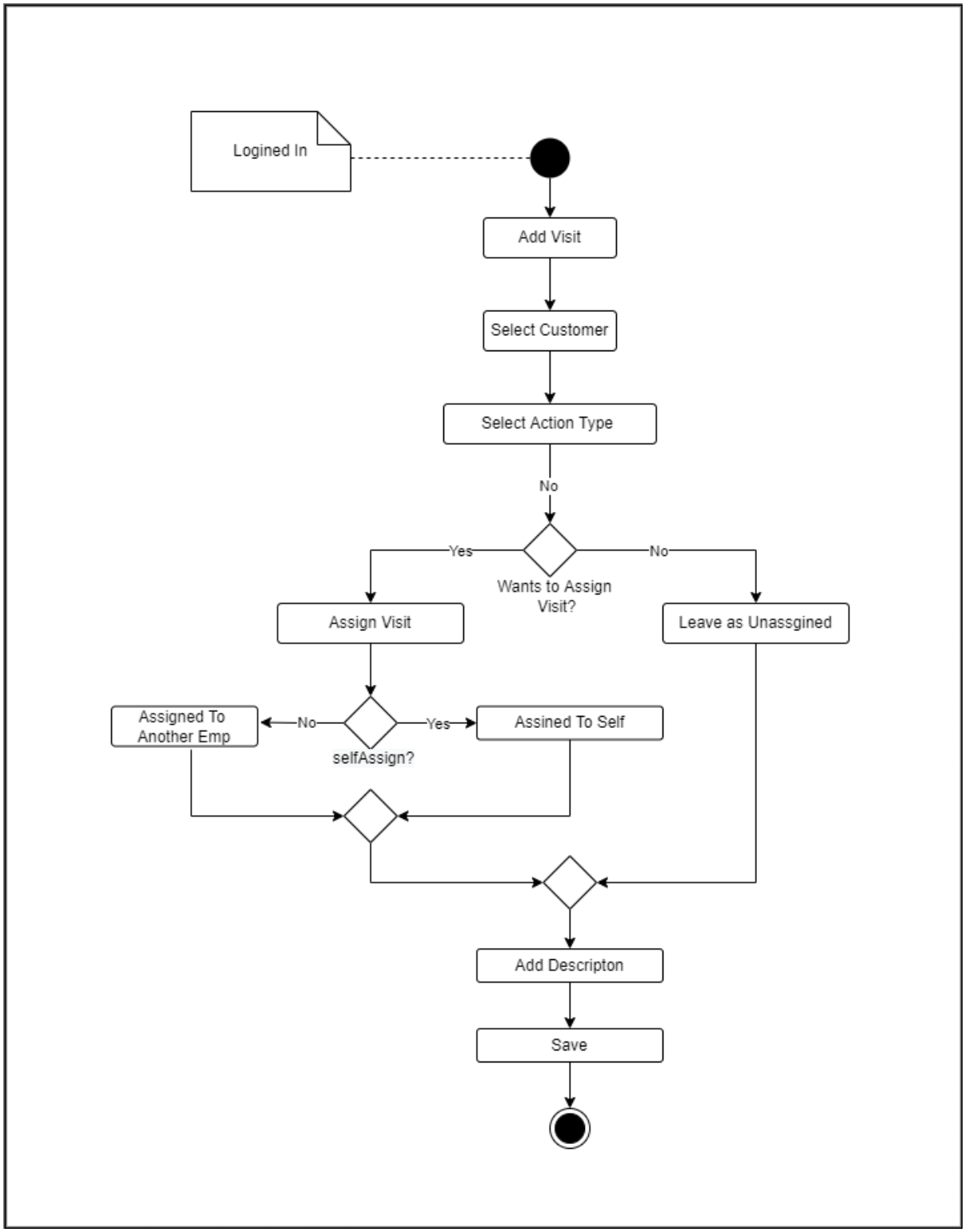
Add Enquiry – Activity Diagram



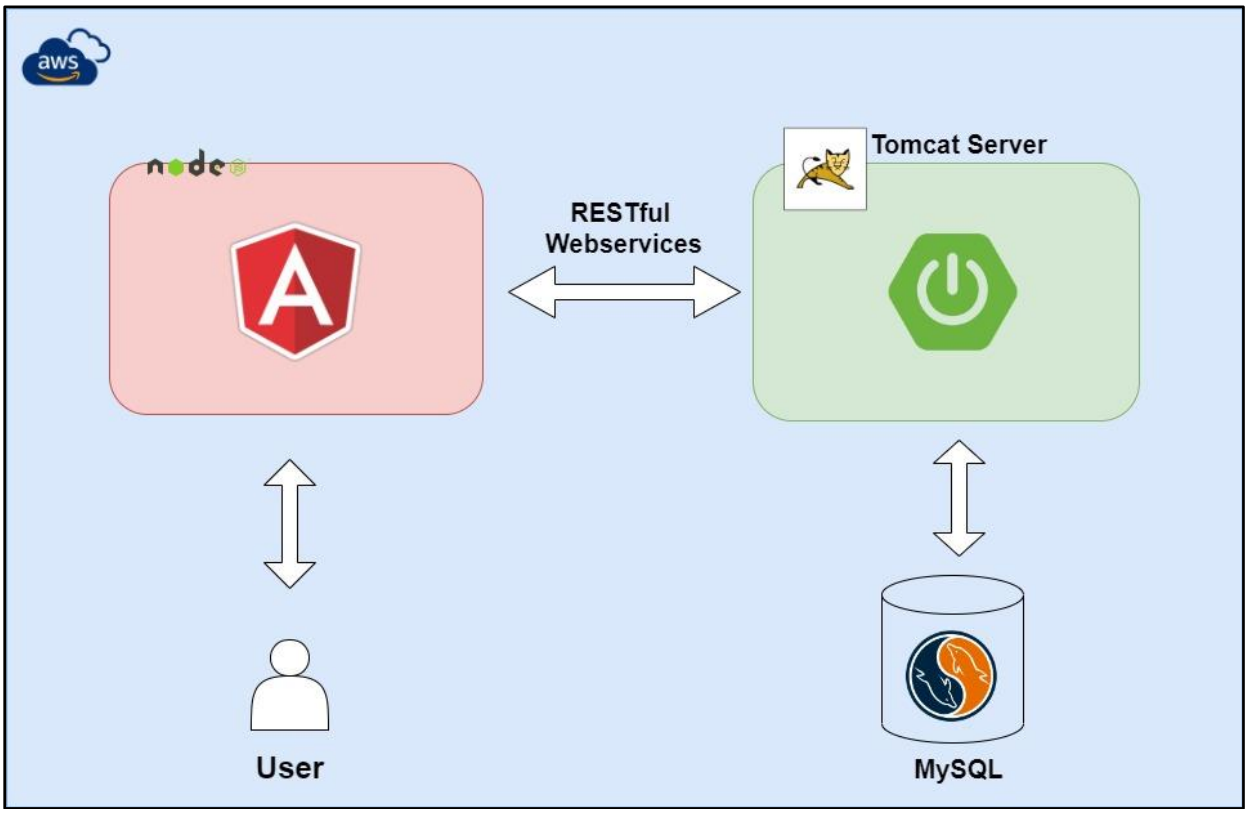
Add Ticket – Activity Diagram



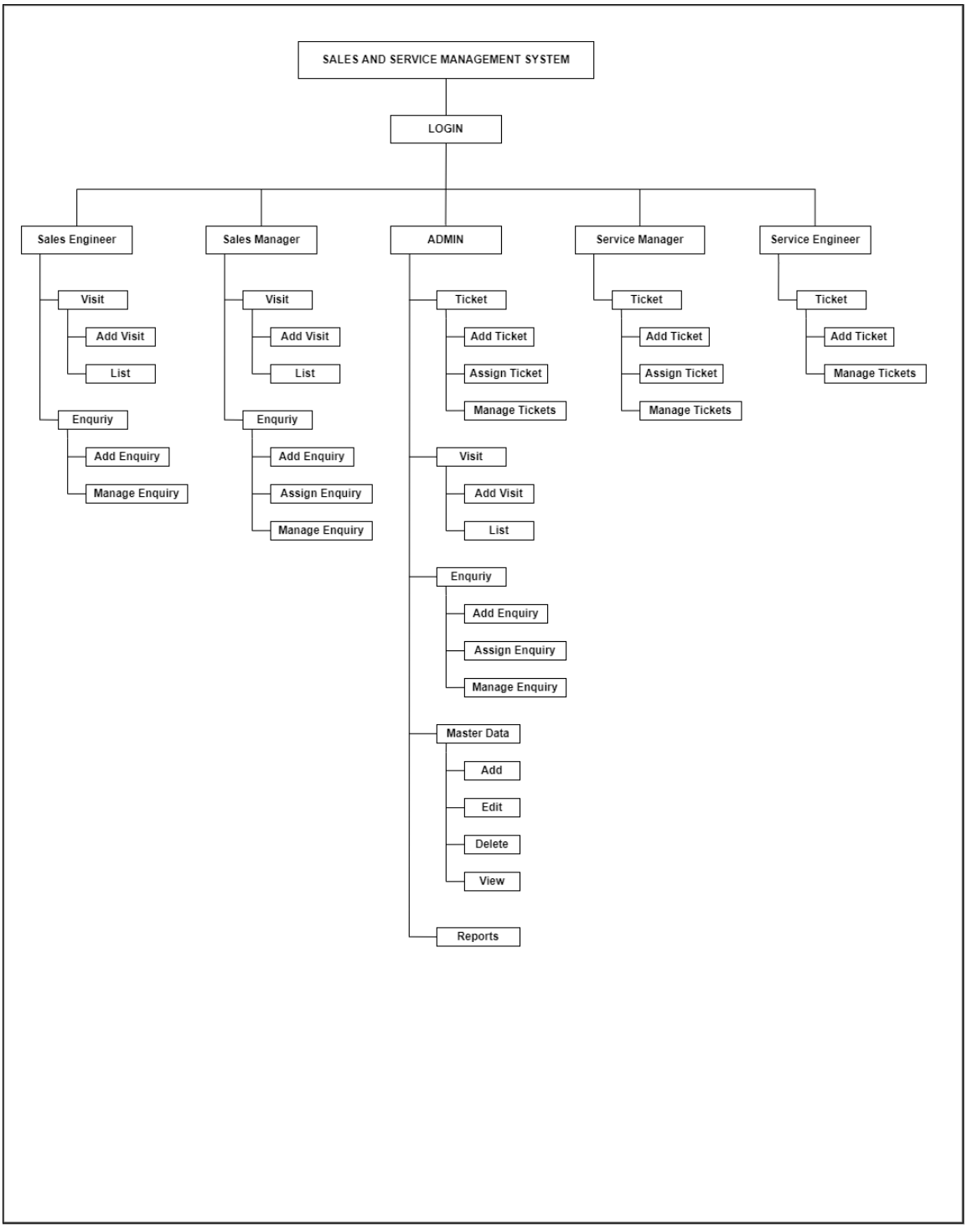
Add Visit - Activity Diagram



3.7 Deployment Diagram



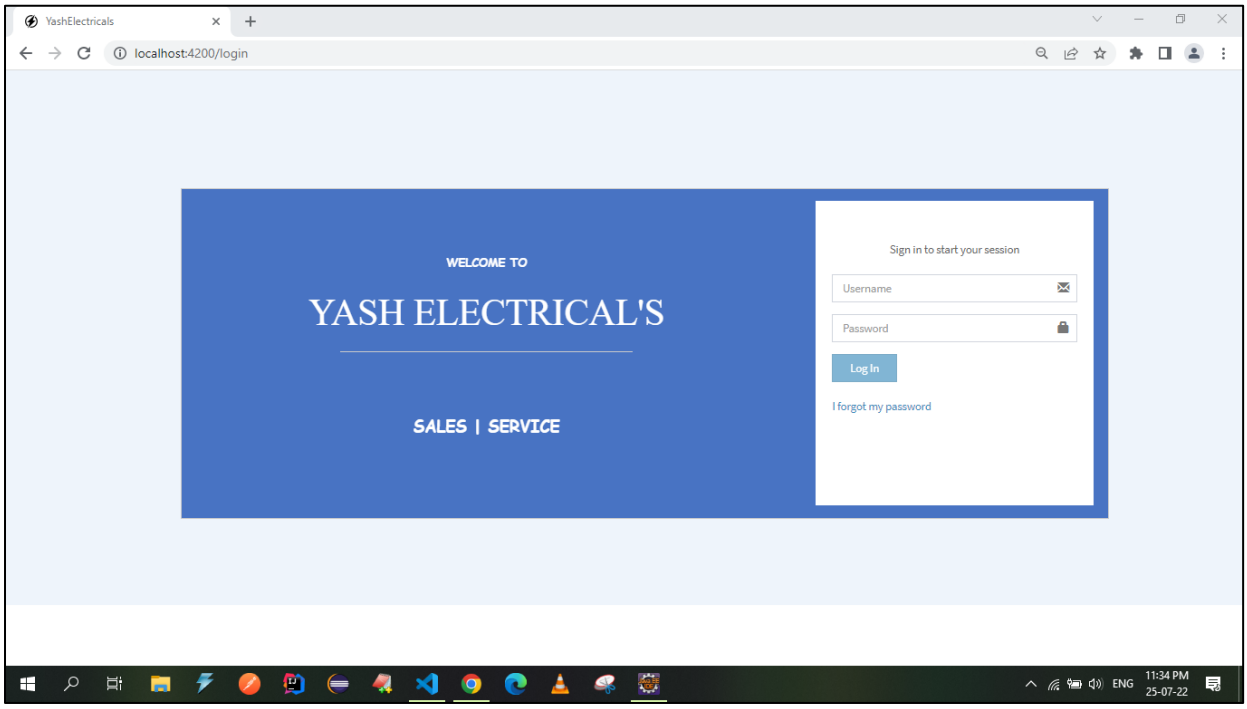
3.8 Module Hierarchy Diagram



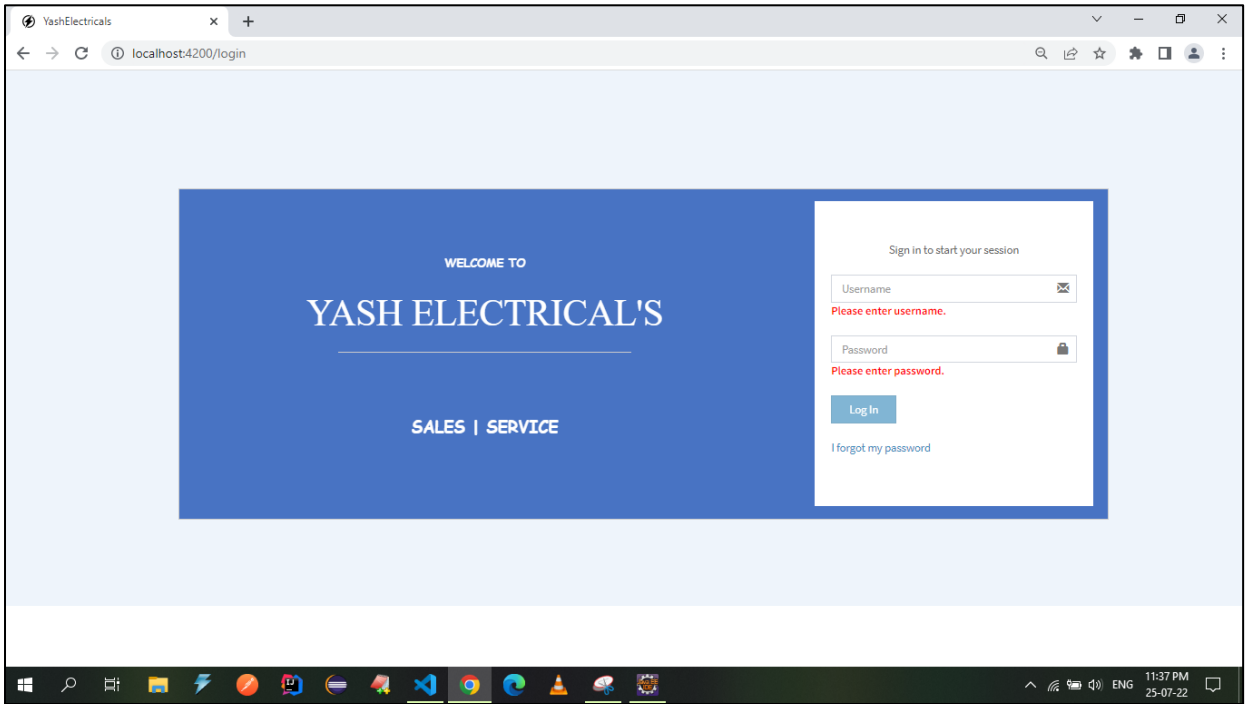
3.9 Sample Input and Output Screens

1. User Login:

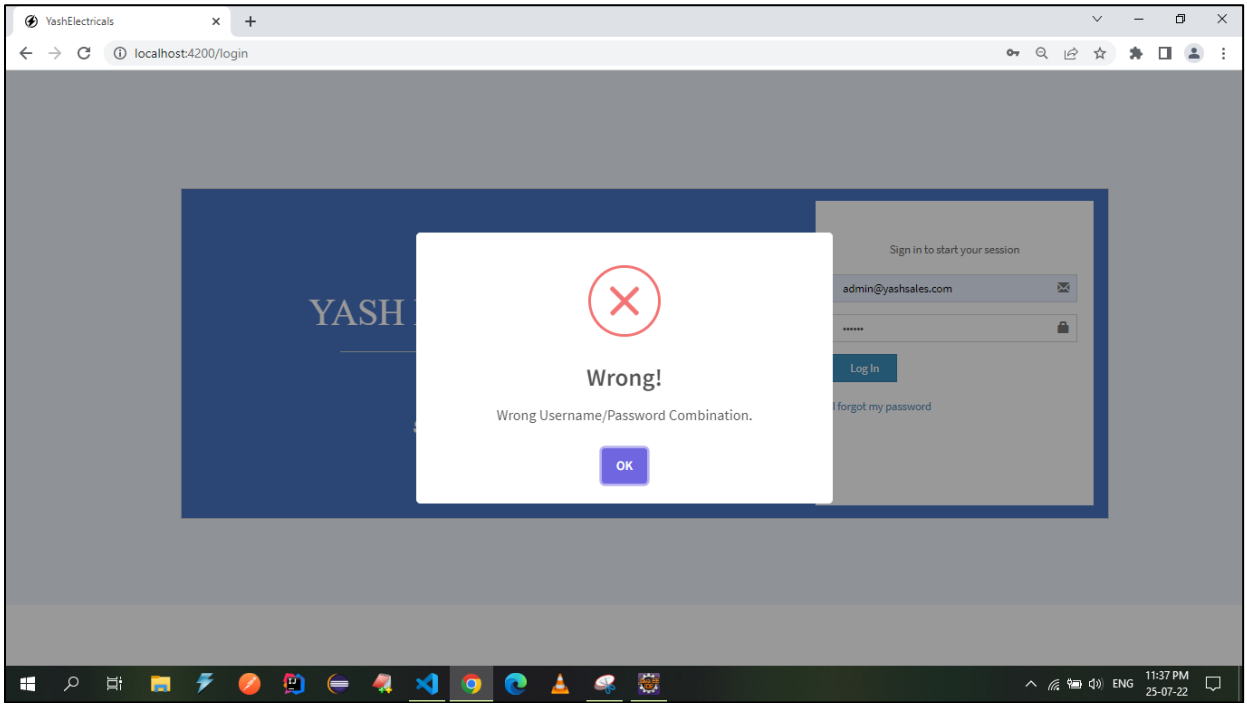
Login page for all users.



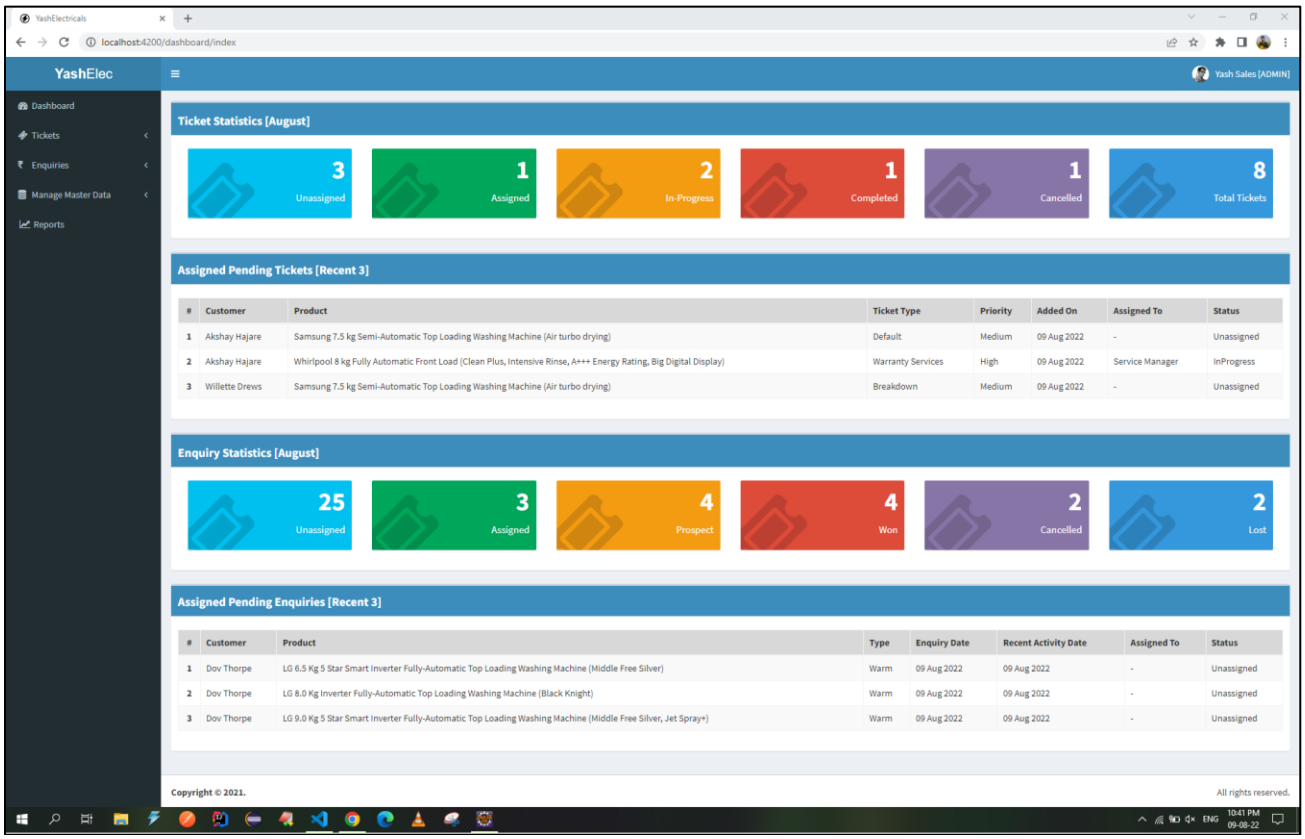
*Login screen validations –



*Invalid username or password message shown when user entered invalid credentials.

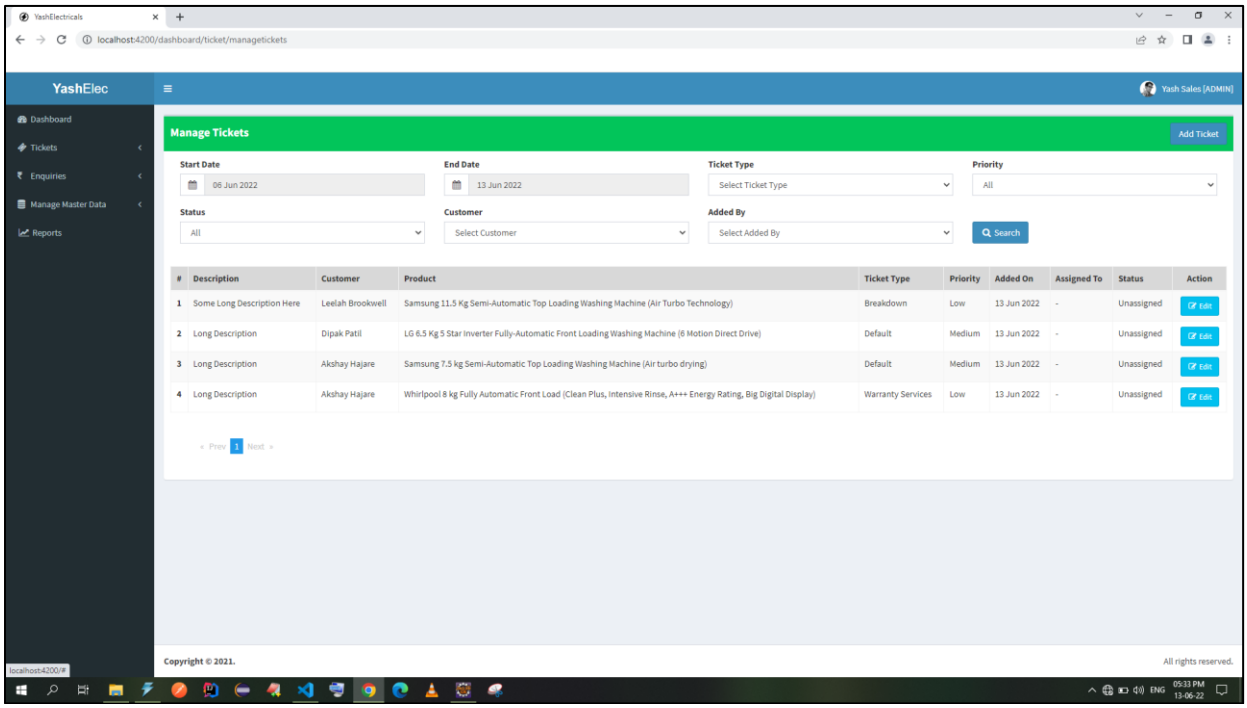


*Dashboard for admin

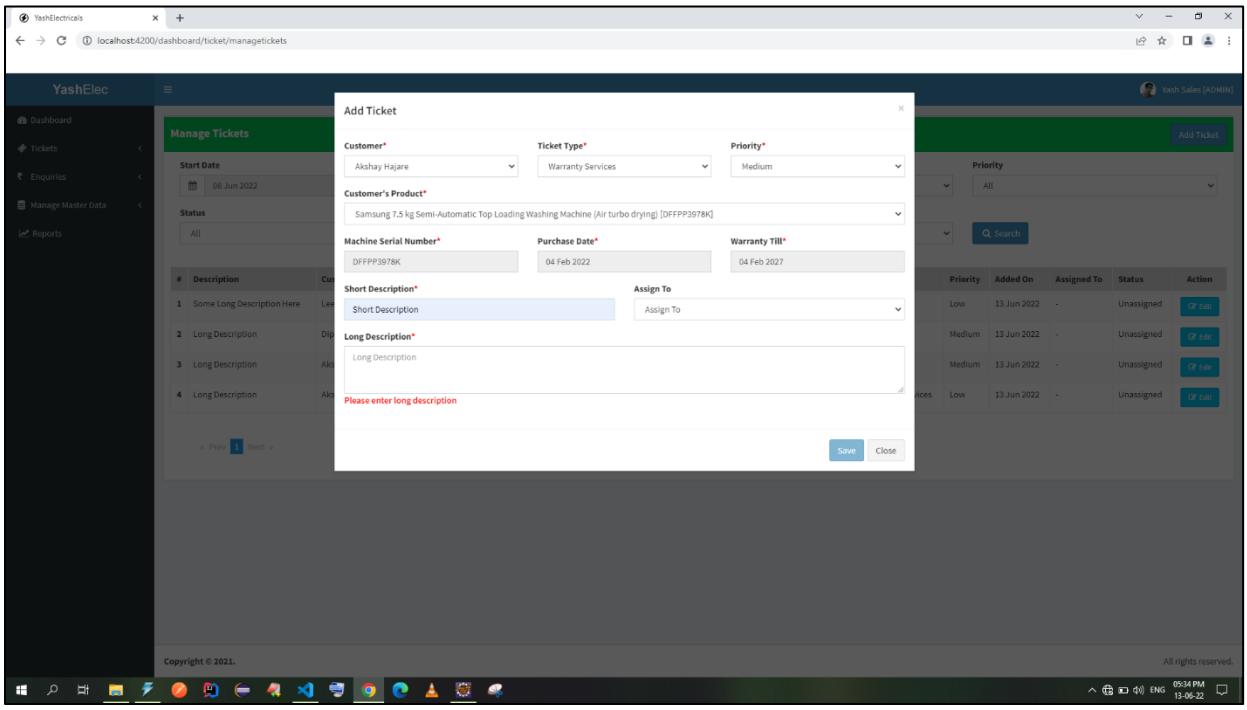


Tickets Module Screen Snaps

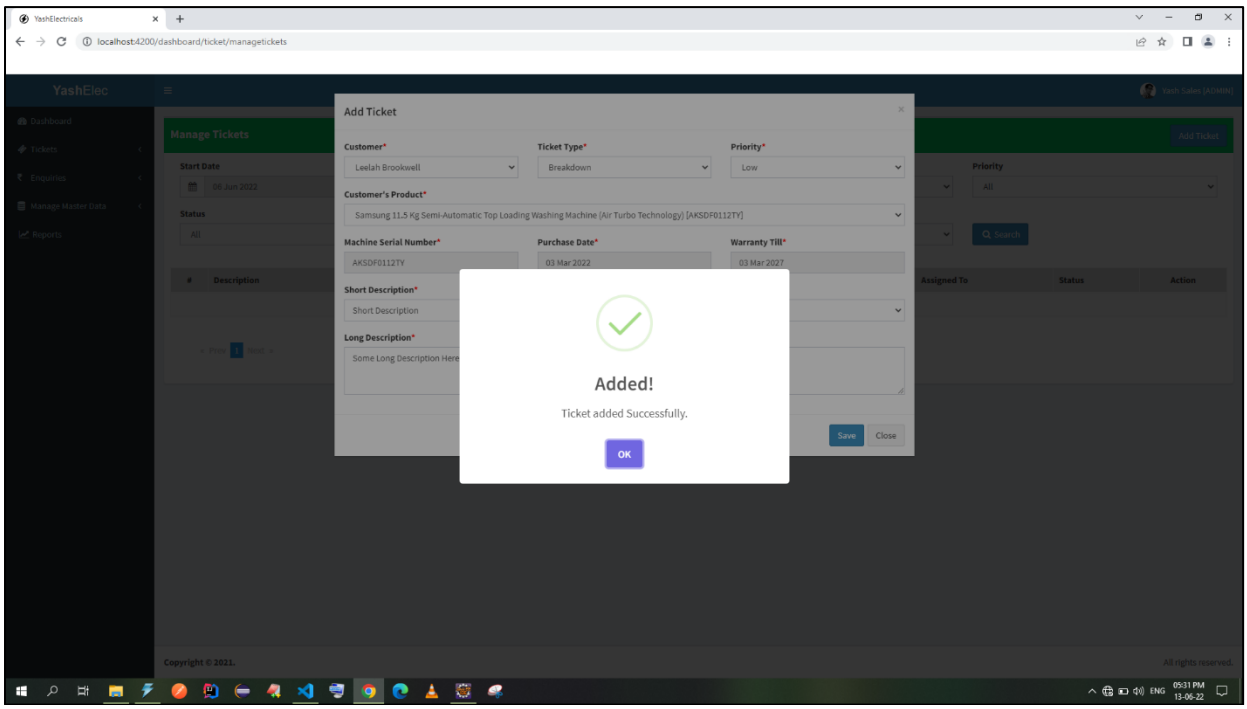
*Manage Ticket: View Tickets



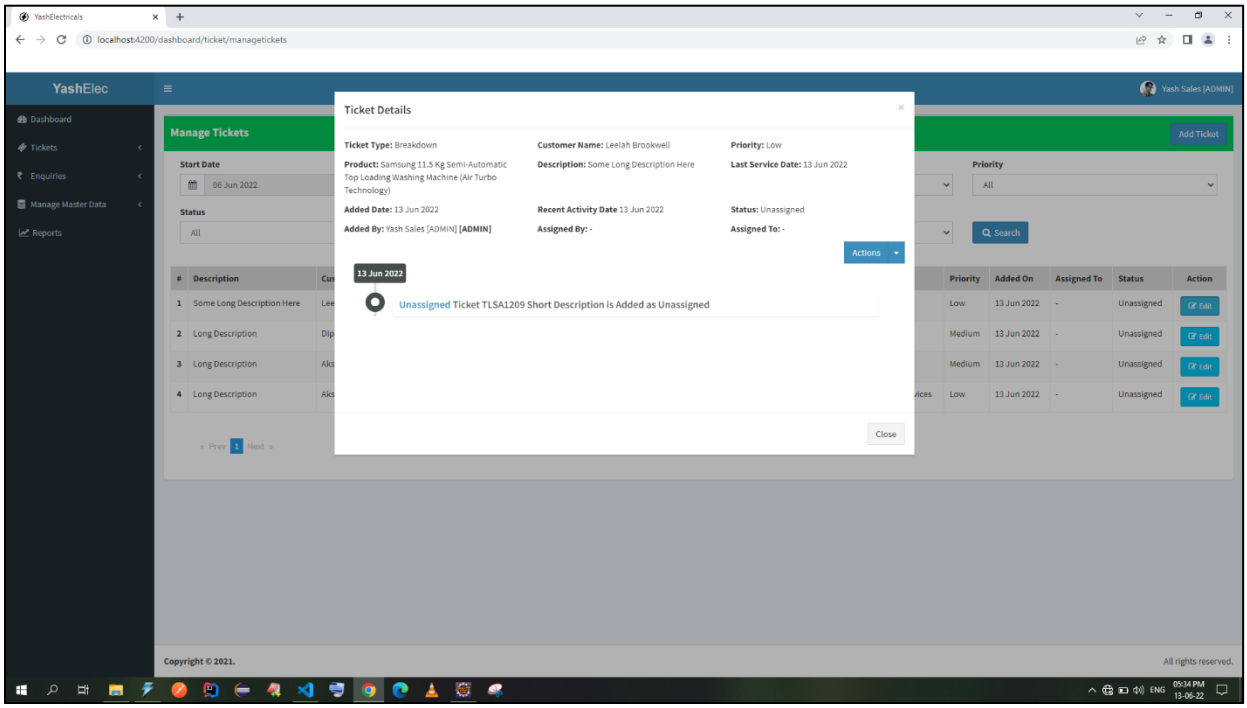
*Manage Ticket: Add Ticket



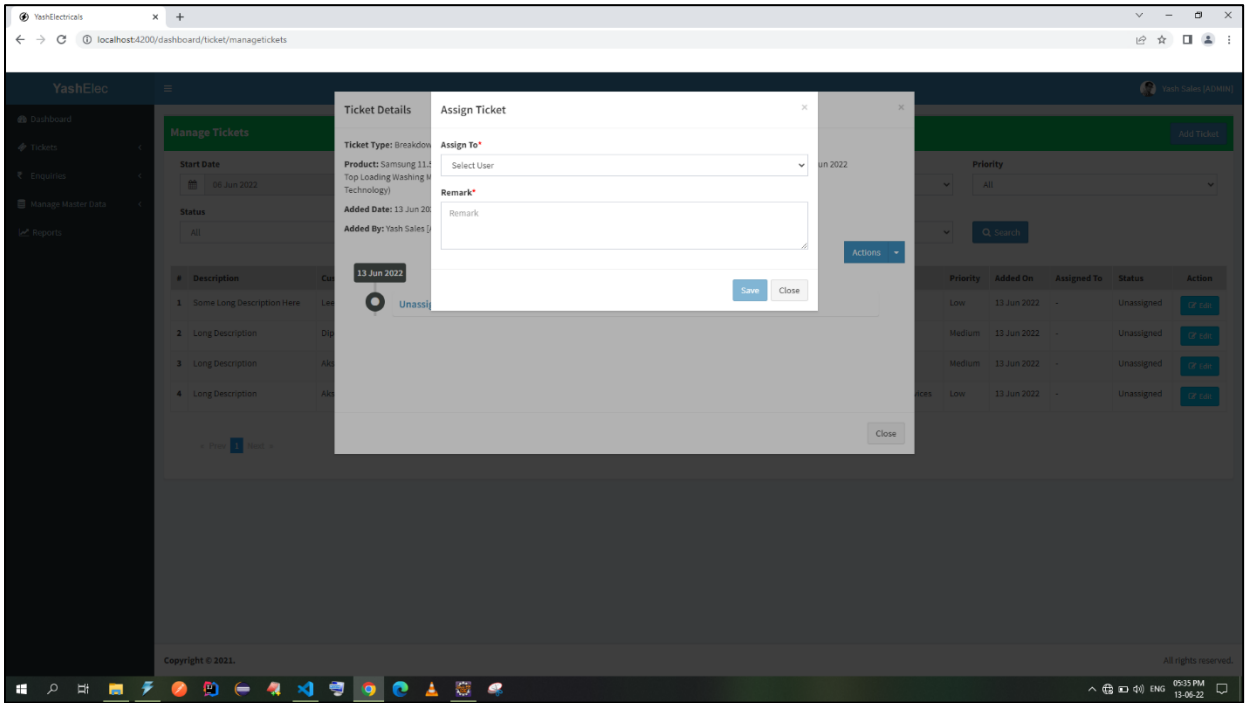
*Manage Ticket: Ticket Added Successfully.



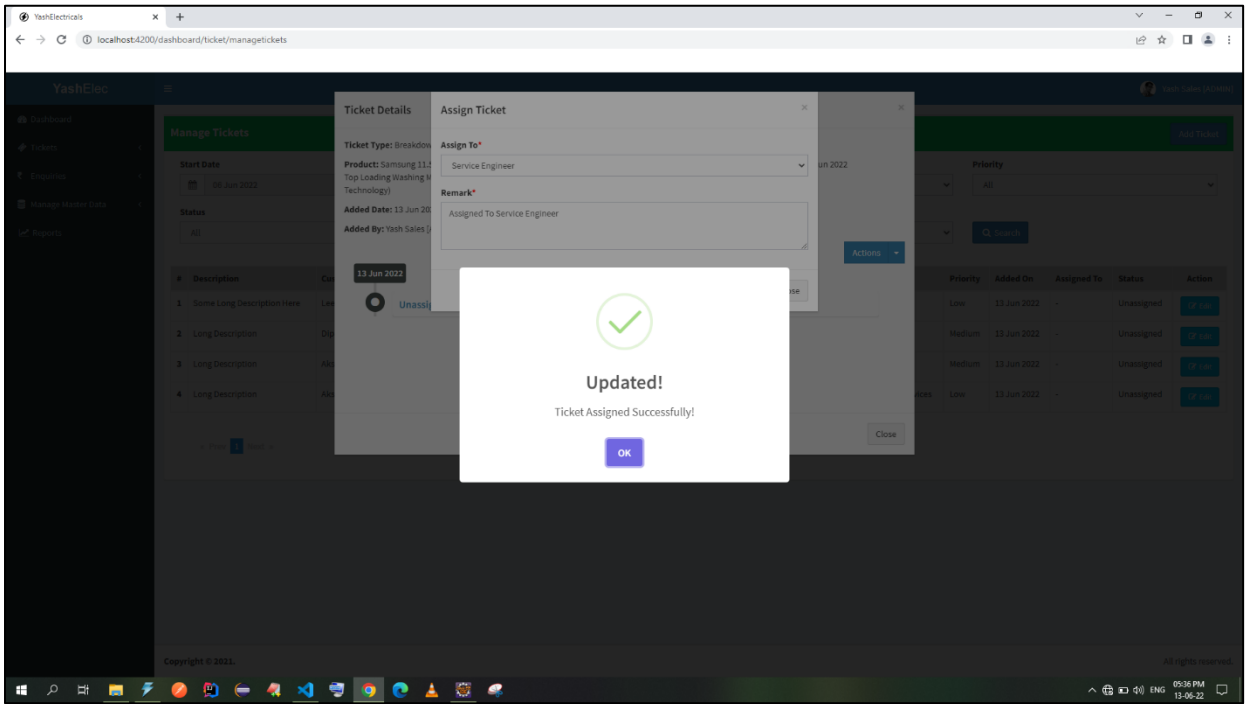
*Manage Ticket: Ticket status Unassigned.



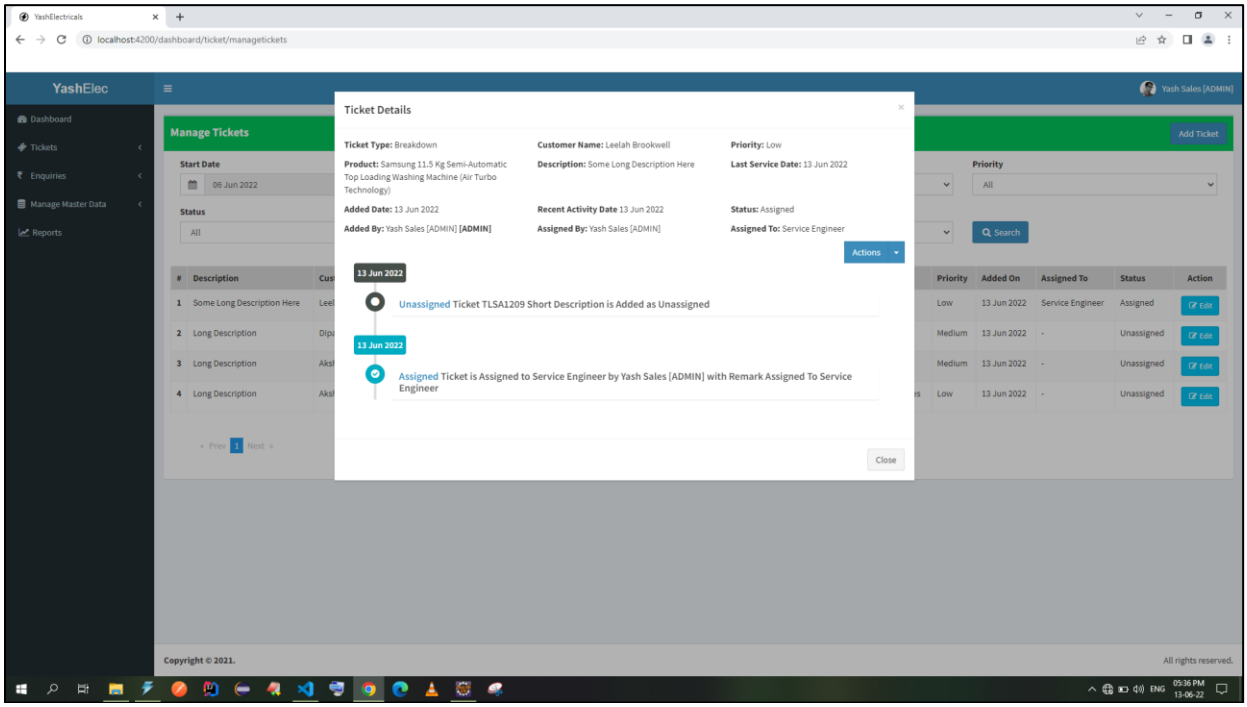
*Manage Ticket: Assign Ticket



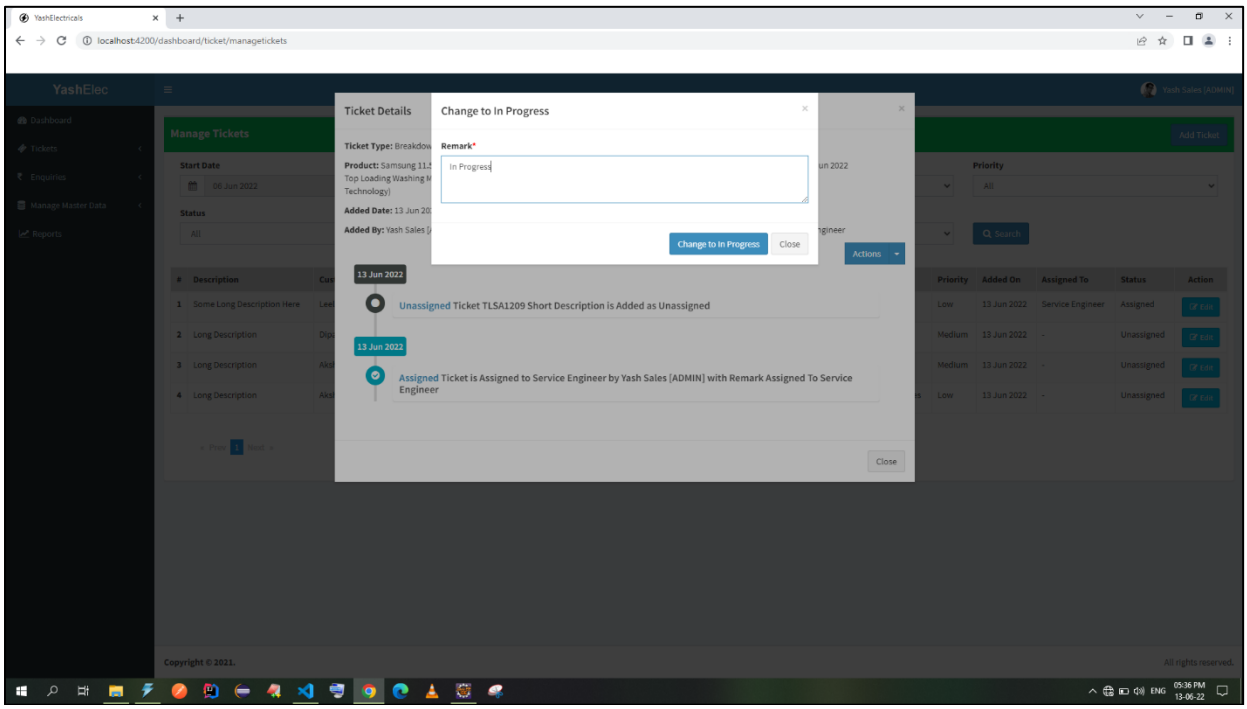
*Manage Ticket: Ticket Assigned Successfully.



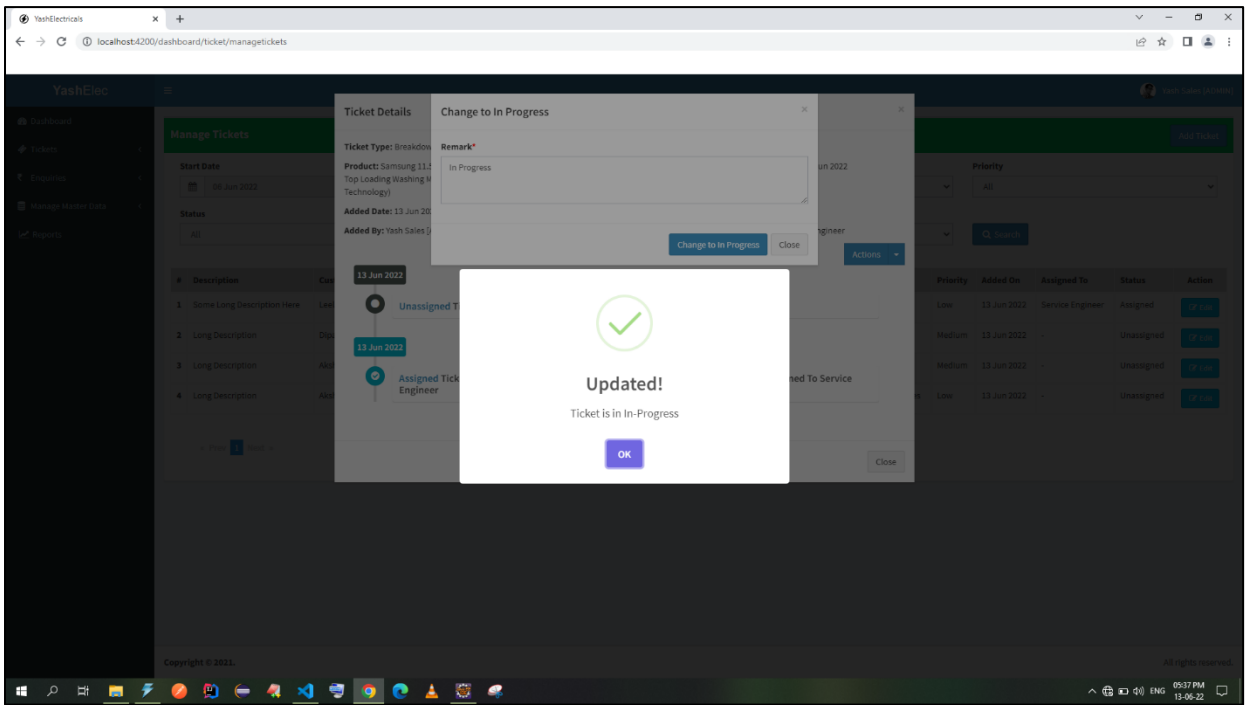
*Manage Ticket: Ticket status updated in ticket Activities on Ticket Details form.



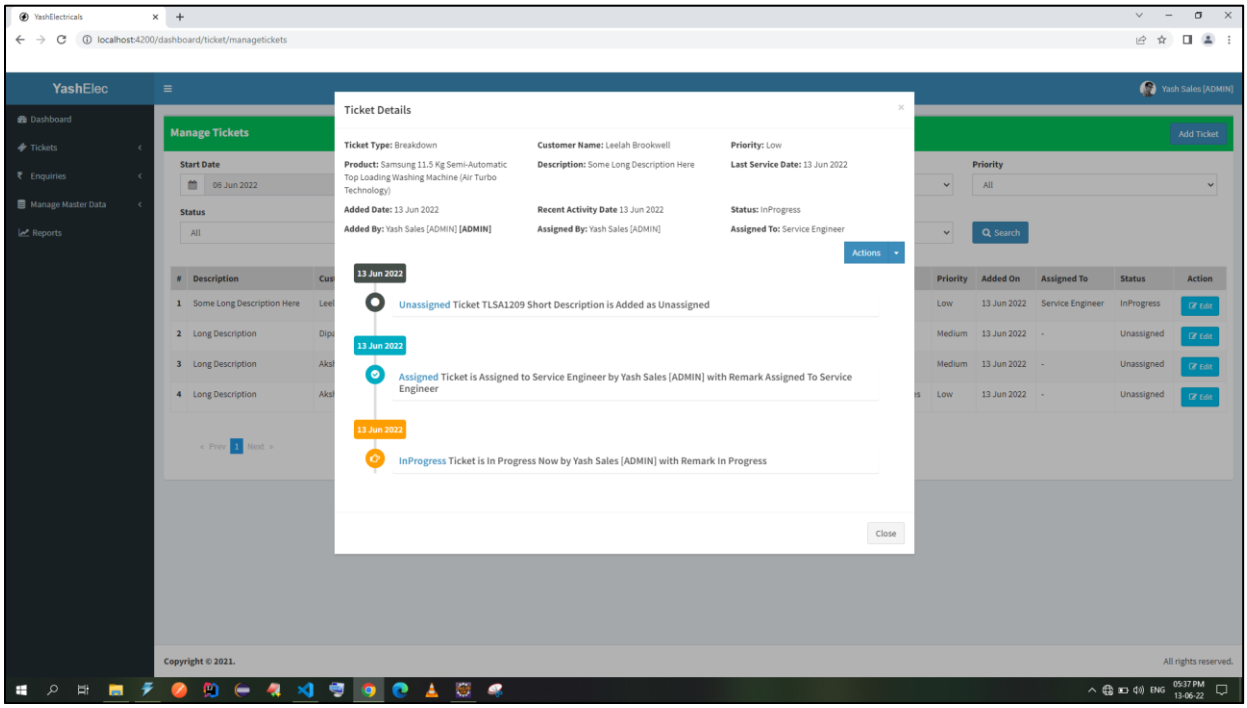
*Manage Ticket: Ticket Change to In Progress.



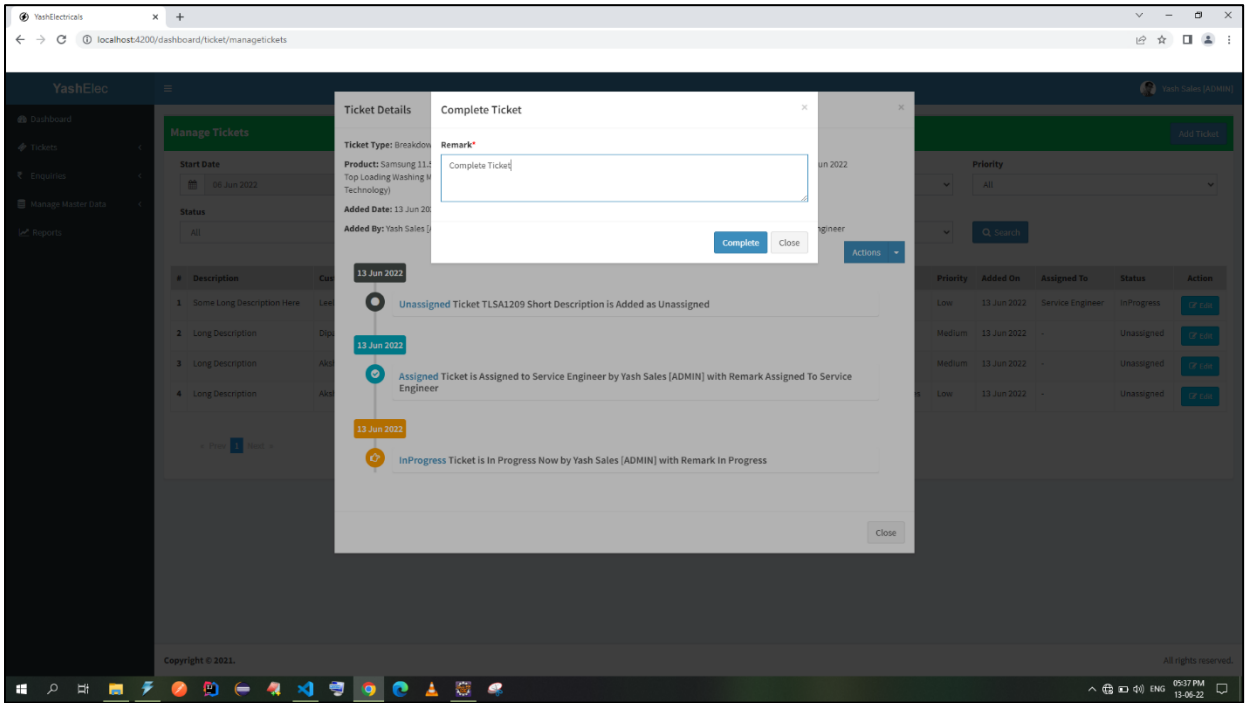
*Manage Ticket: Ticket status changed to In Progress successfully.



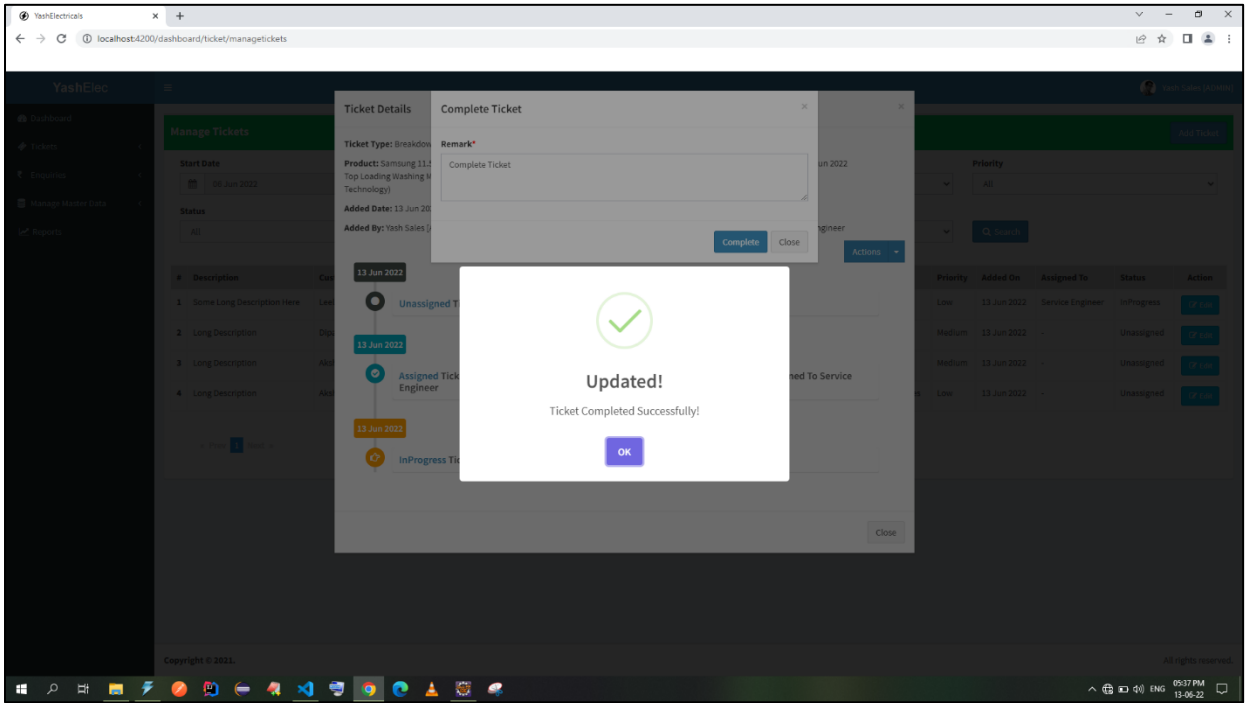
*Manage Ticket: Ticket status updated in ticket Activities on Ticket Details form.



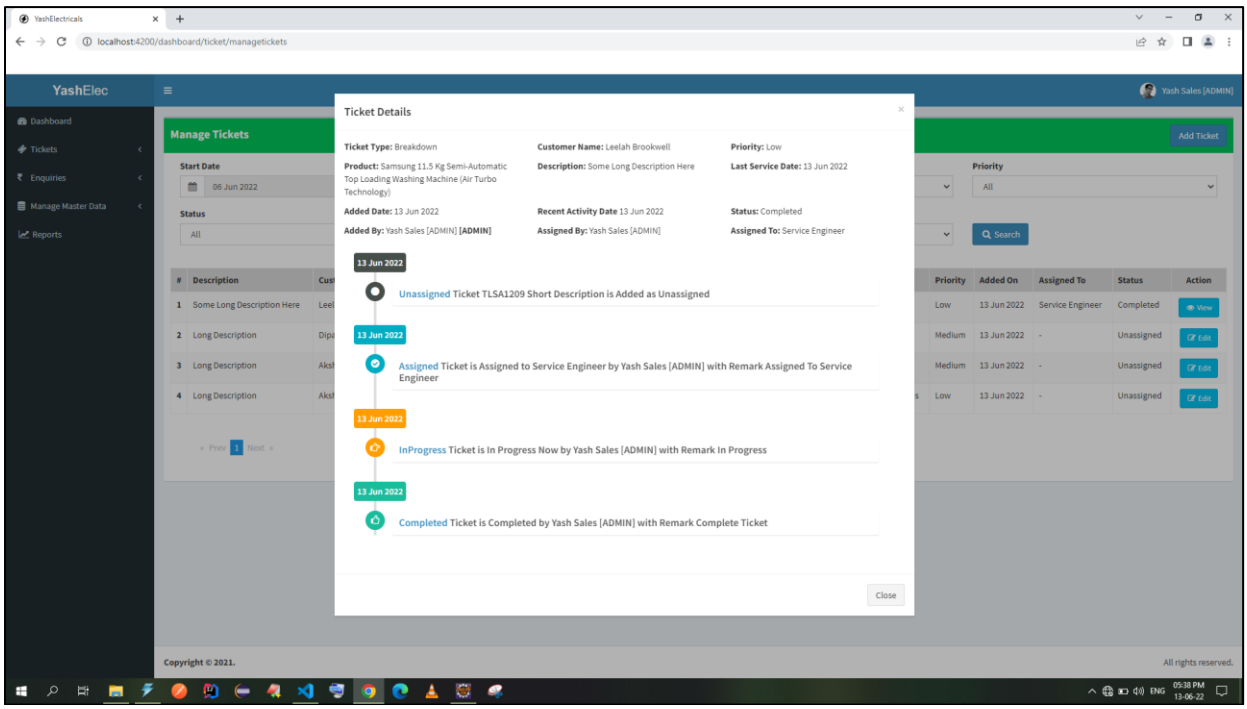
*Manage Ticket: Complete Ticket.



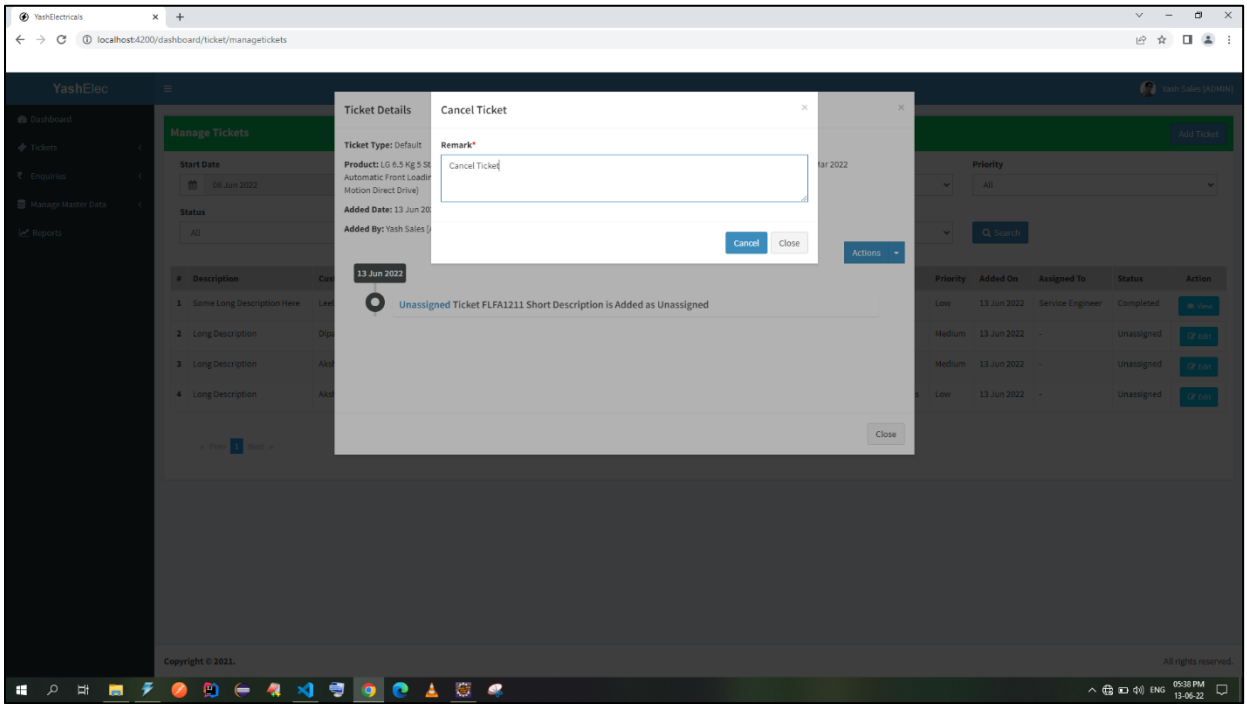
*Manage Ticket: Ticket status updated as Complete successfully.



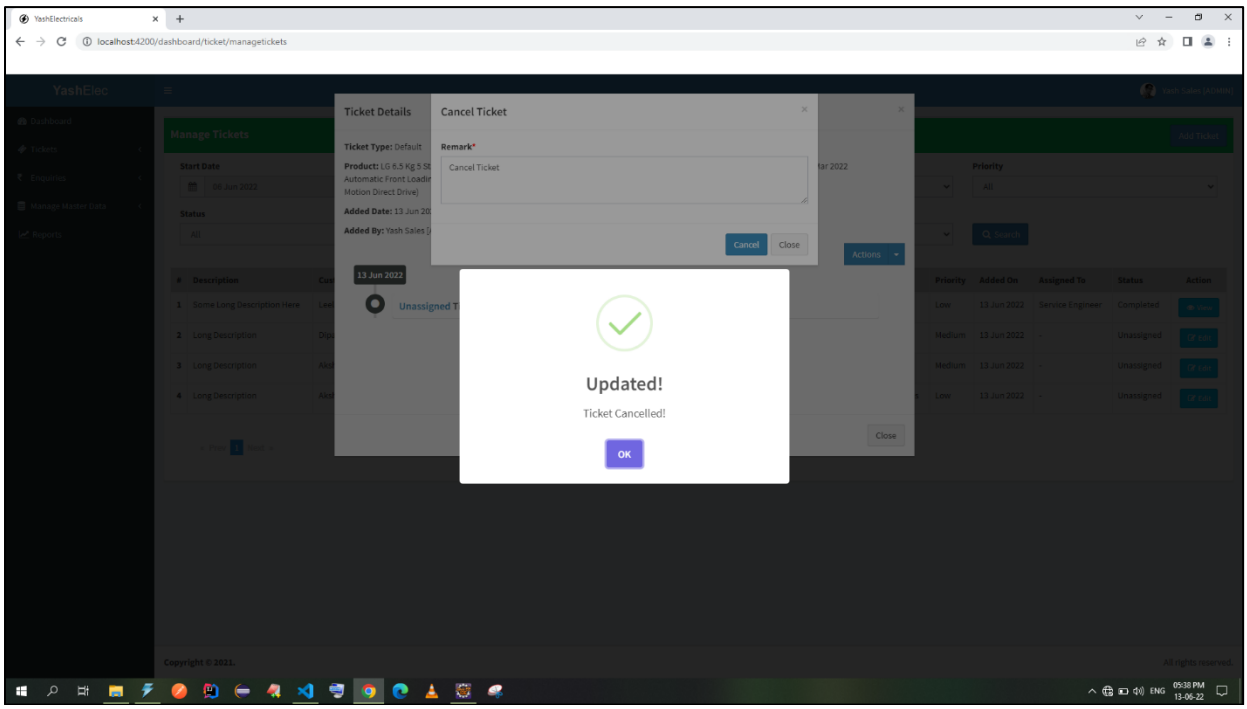
*Manage Ticket: Ticket status updated in ticket Activities on Ticket Details form.



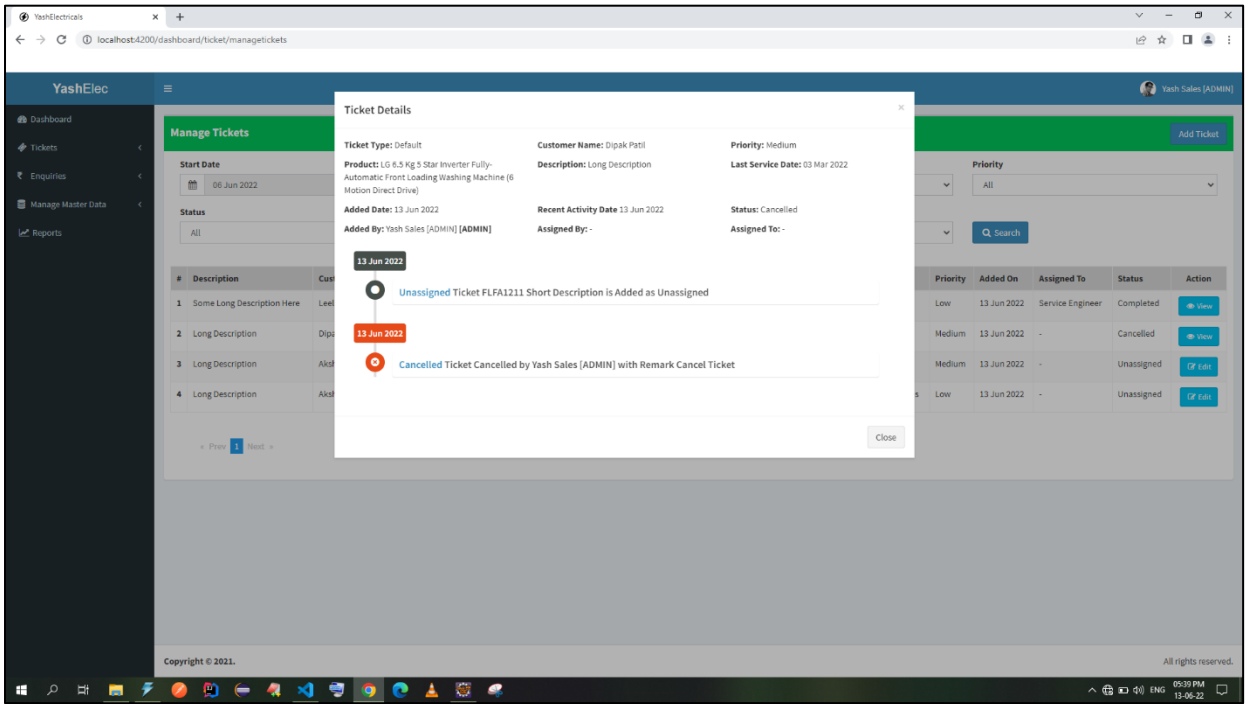
*Manage Ticket: Cancel Ticket.



*Manage Ticket: Ticket cancelled successfully.

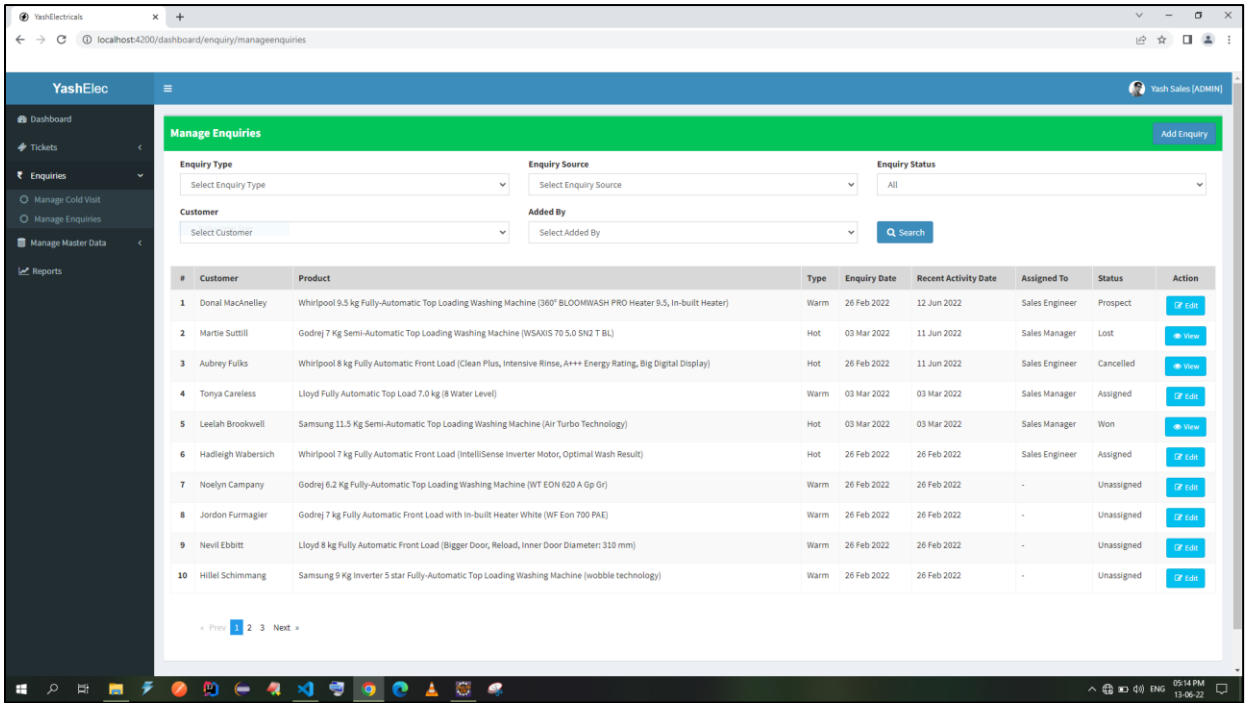


*Manage Ticket: Ticket status updated in ticket Activities on Ticket Details form.

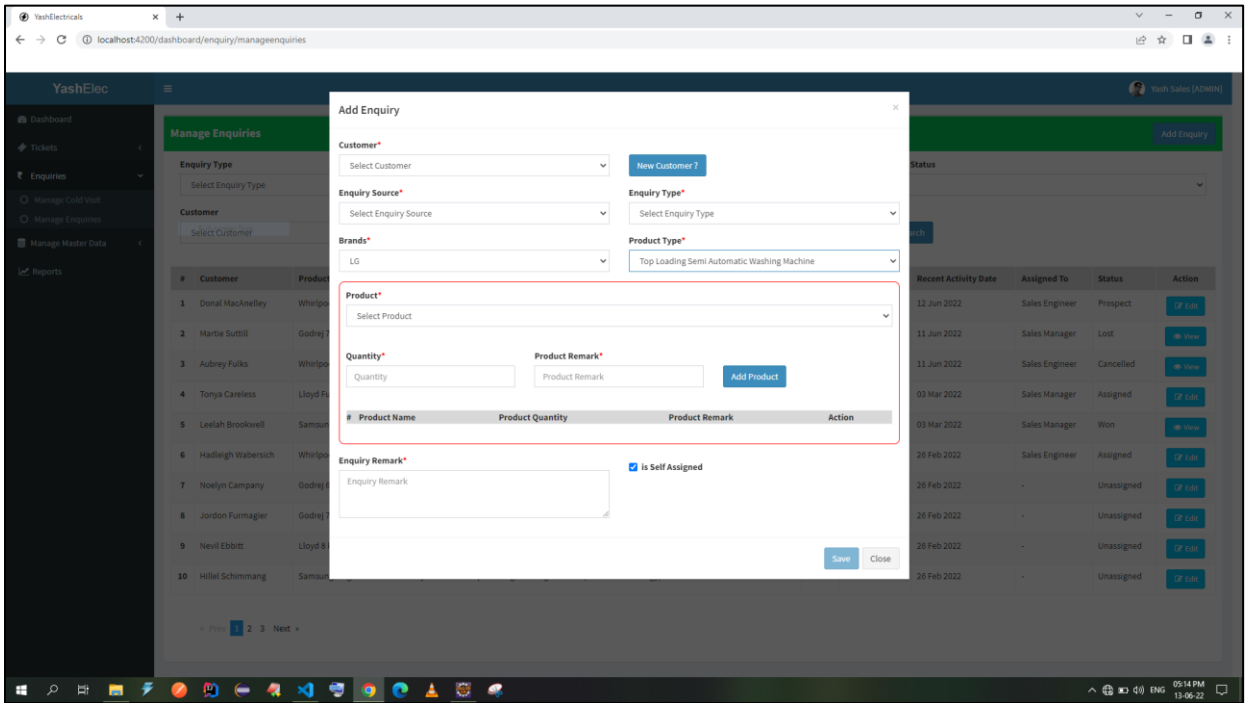


Enquiry Module Screen Snaps

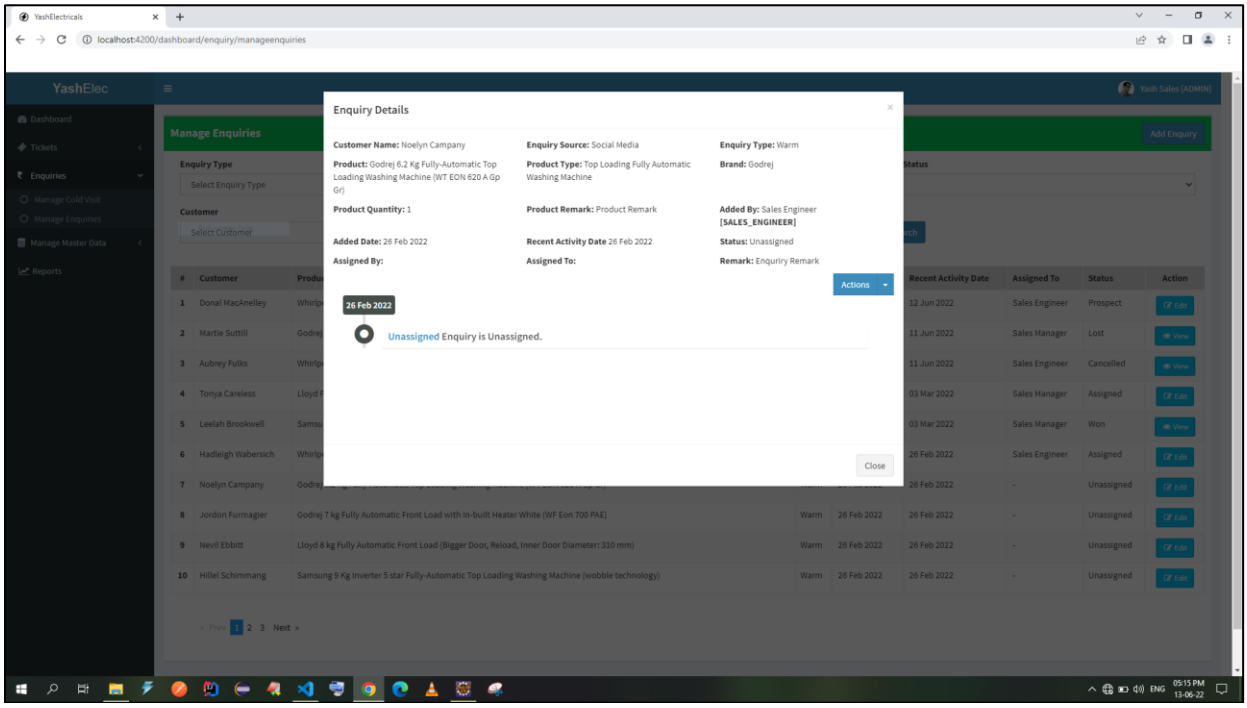
*Manage Enquiries: View Enquiries



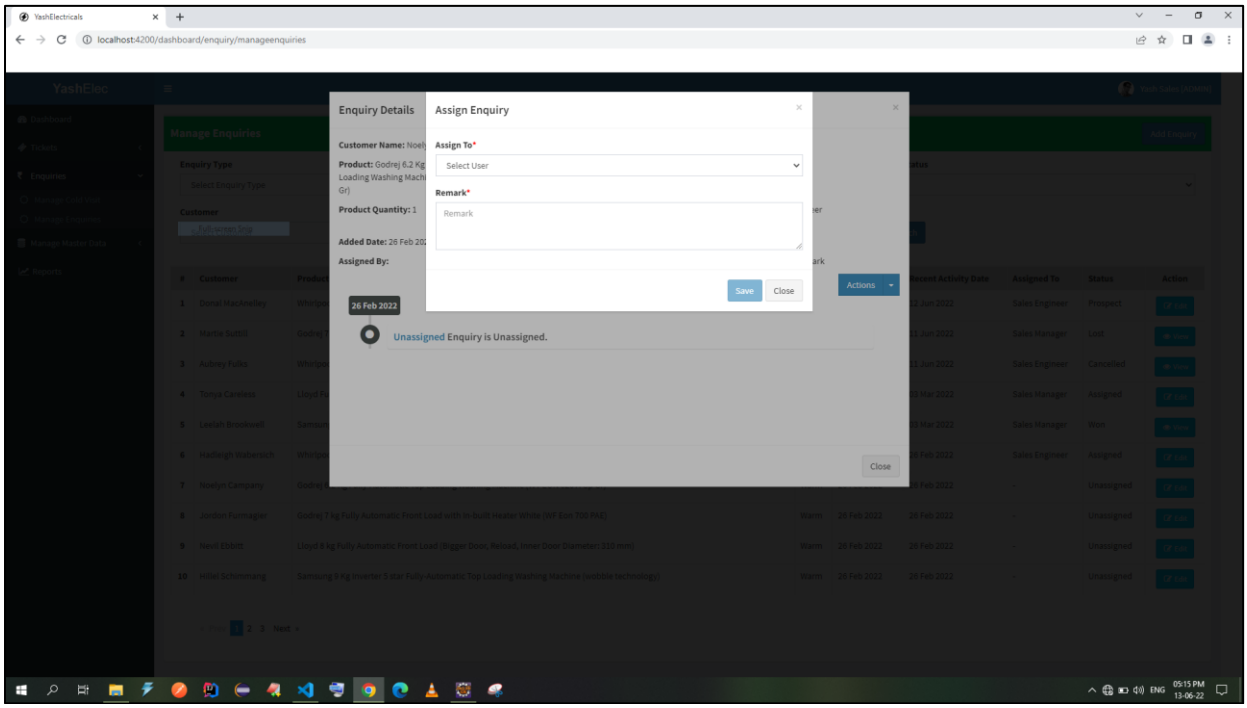
*Manage Enquiries: Add Enquiry



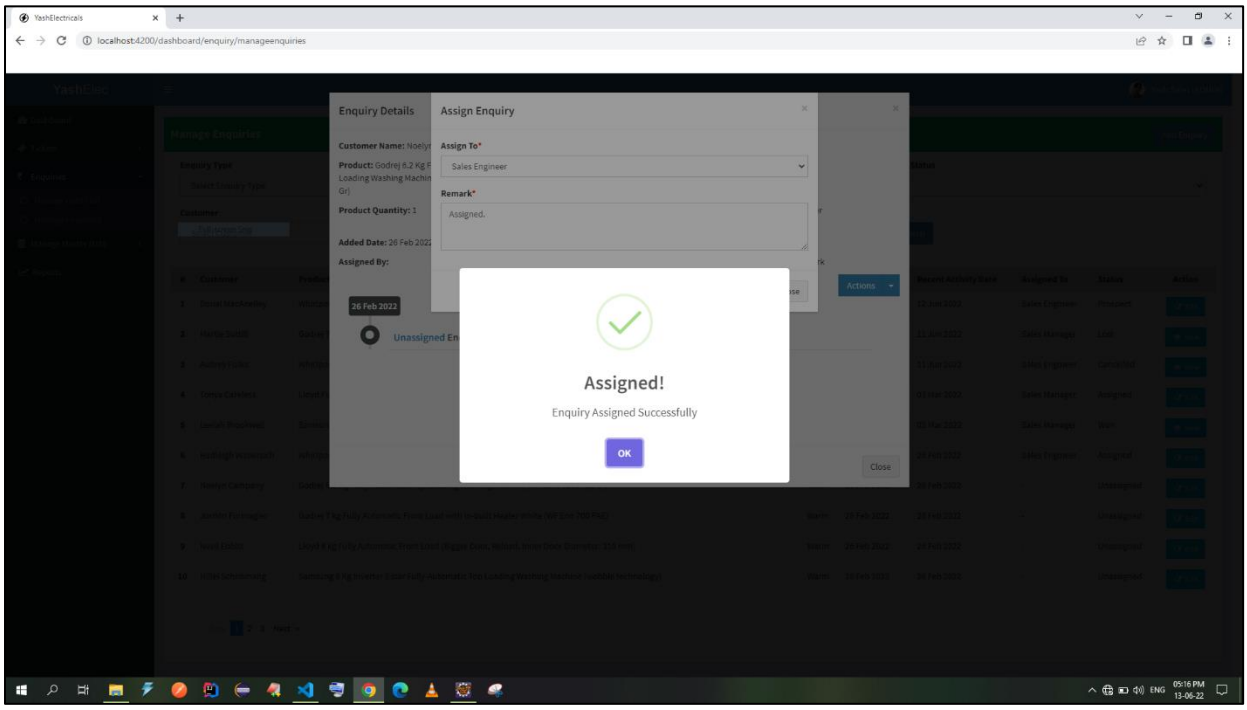
*Manage Enquiries: View Enquiry Details



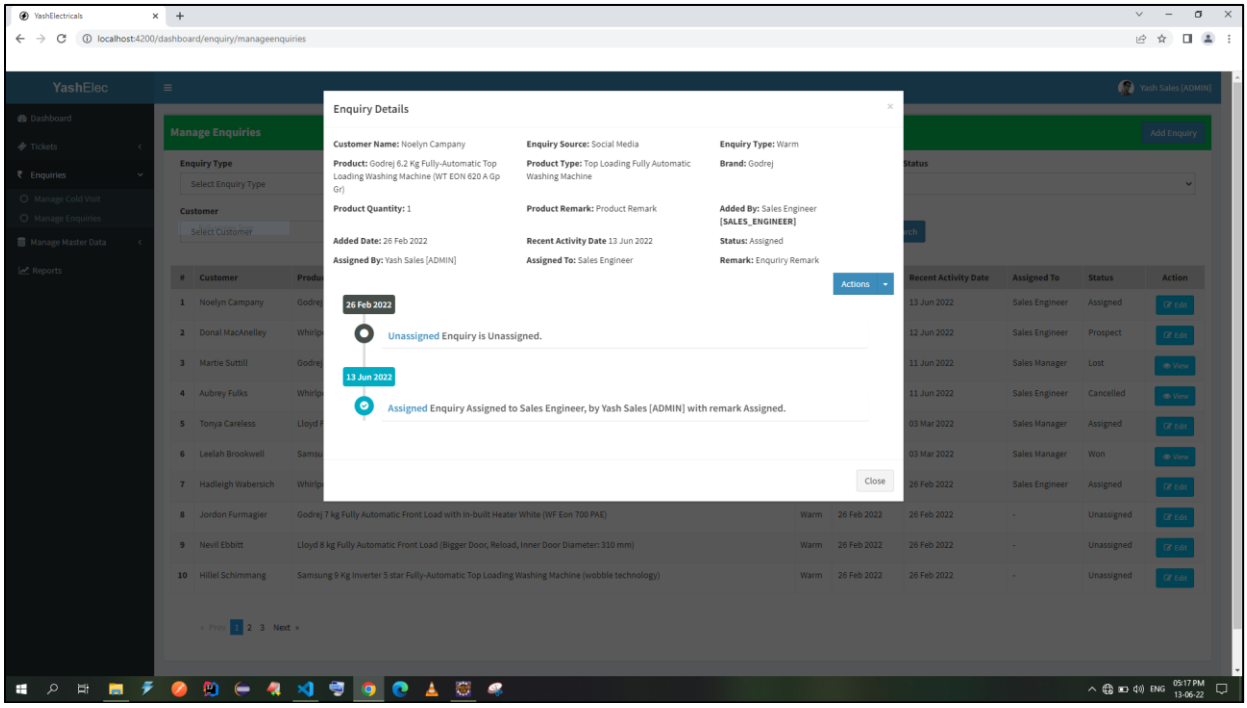
*Manage Enquiries: Assign Enquiry



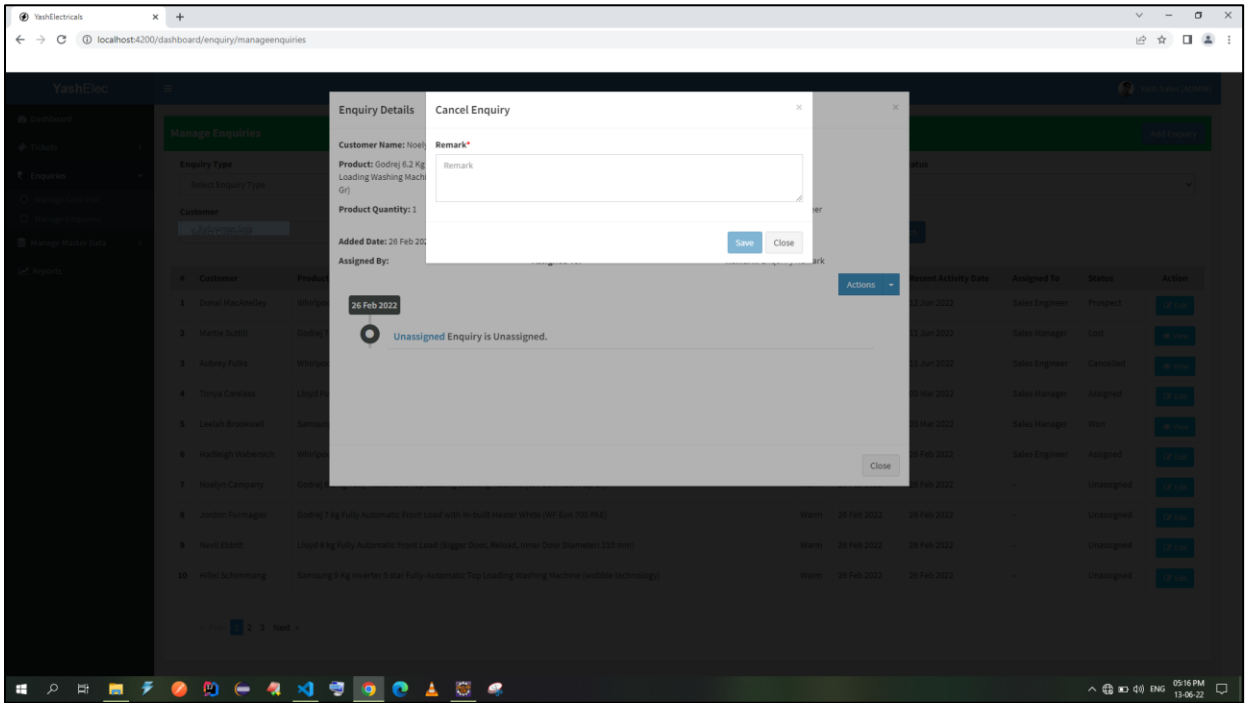
*Manage Enquiries: Enquiry Assigned Successfully



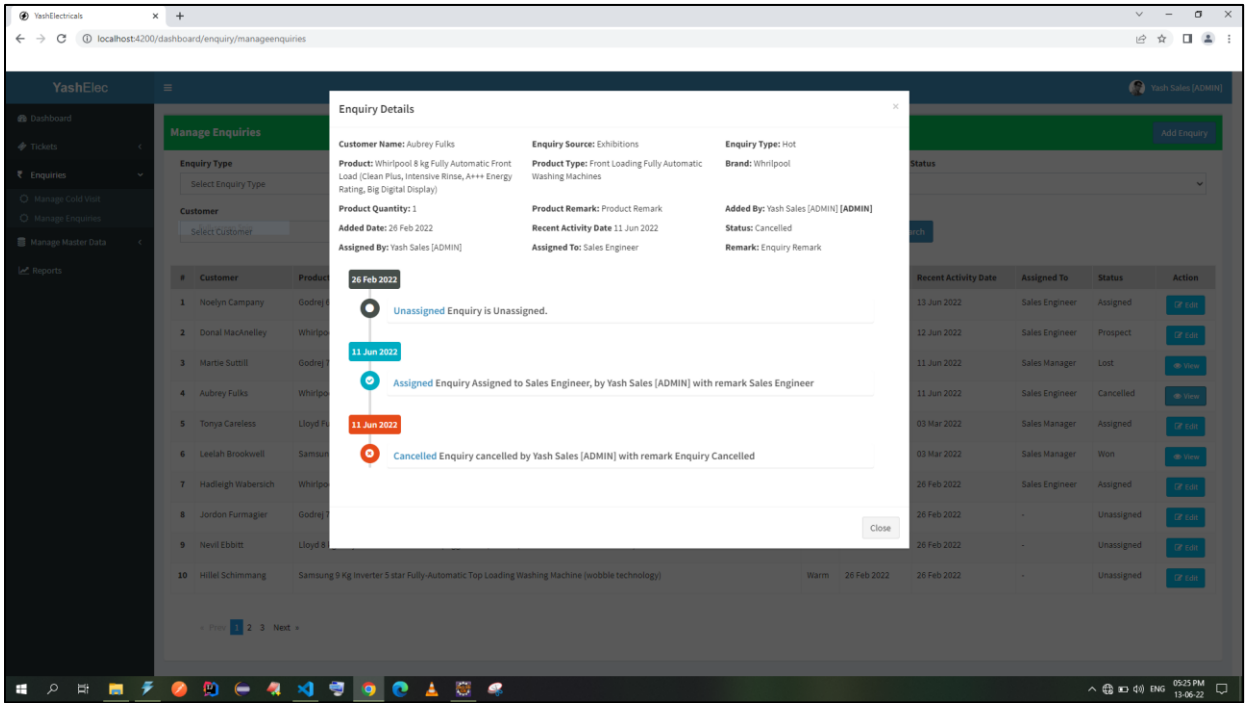
*Manage Enquiries: Enquiry Details after enquiry assigned. Enquiry Activities are changed.



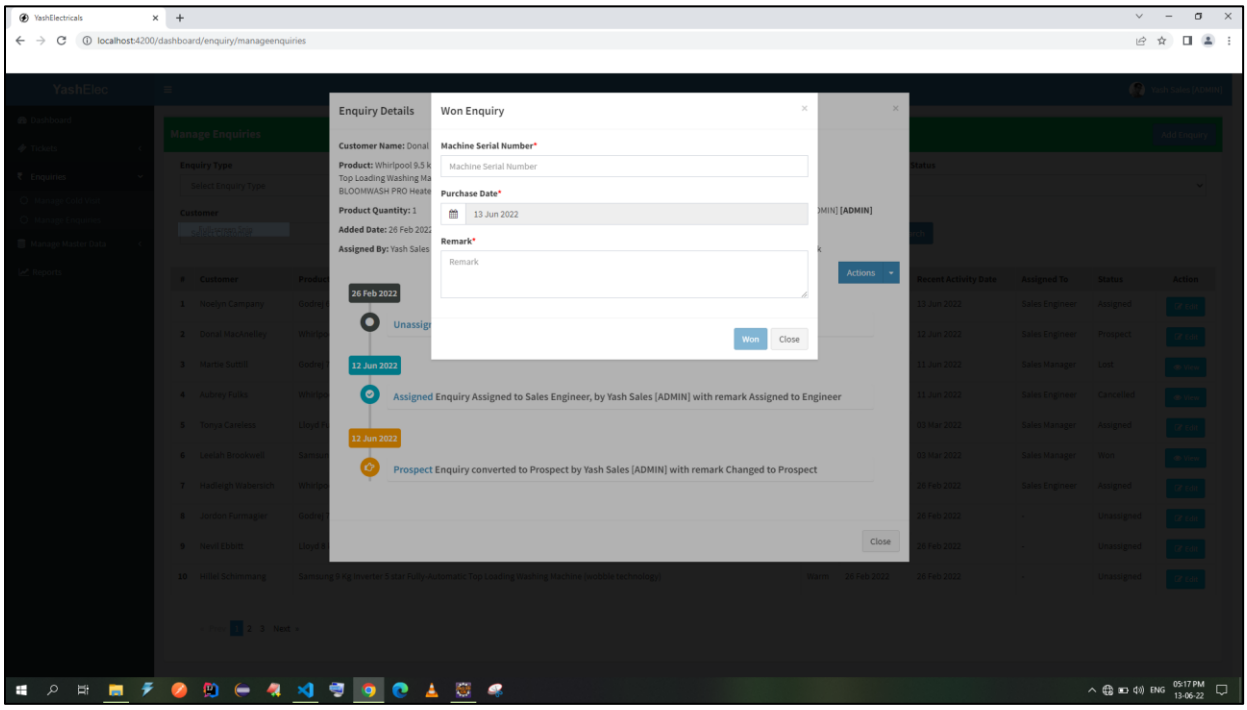
*Manage Enquiries: Cancel Enquiry



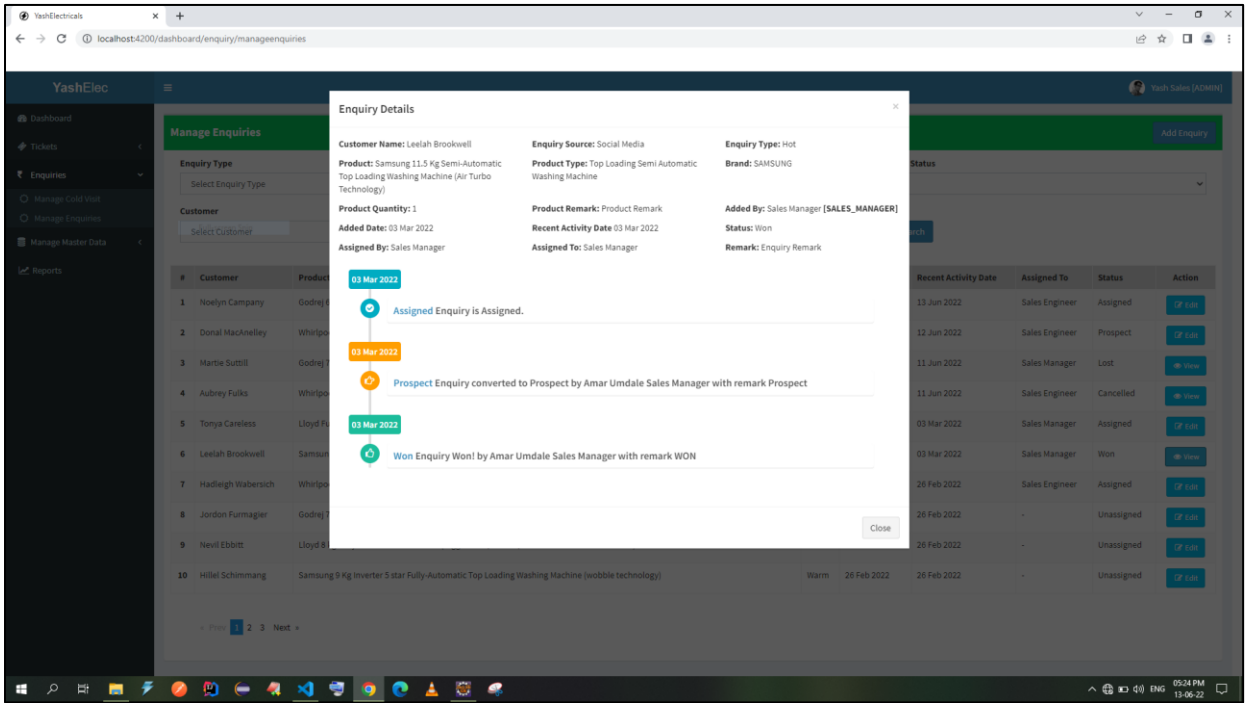
*Manage Enquiries: Enquiry Details after enquiry cancelled. Enquiry Activities are changed.



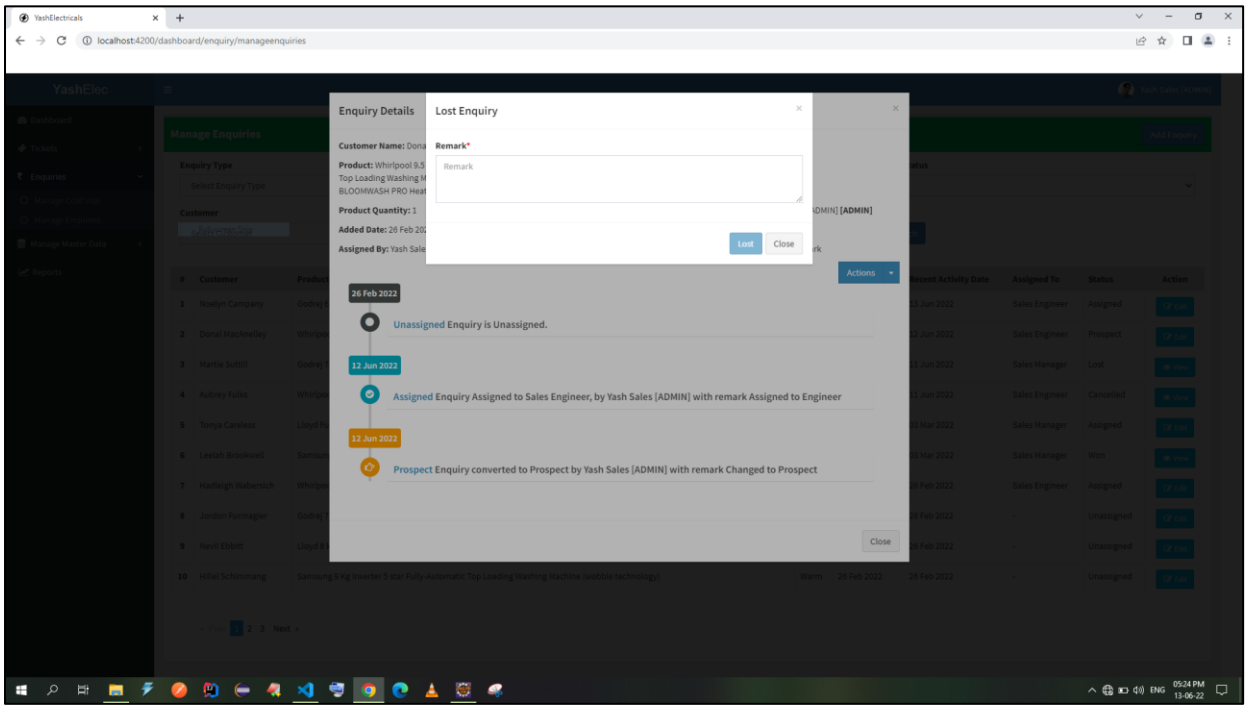
*Manage Enquiries: Won Enquiry



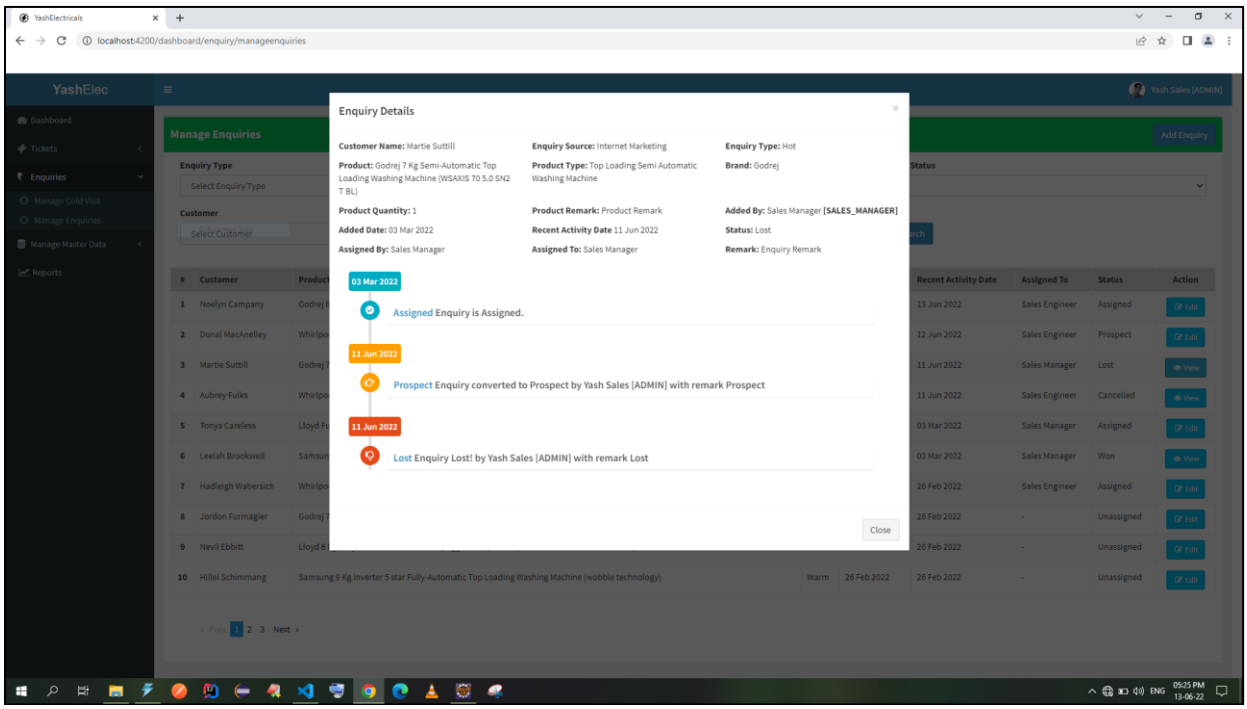
*Manage Enquiries: Enquiry Details after enquiry won. Enquiry Activities are changed.



*Manage Enquiries: Lost Enquiry.

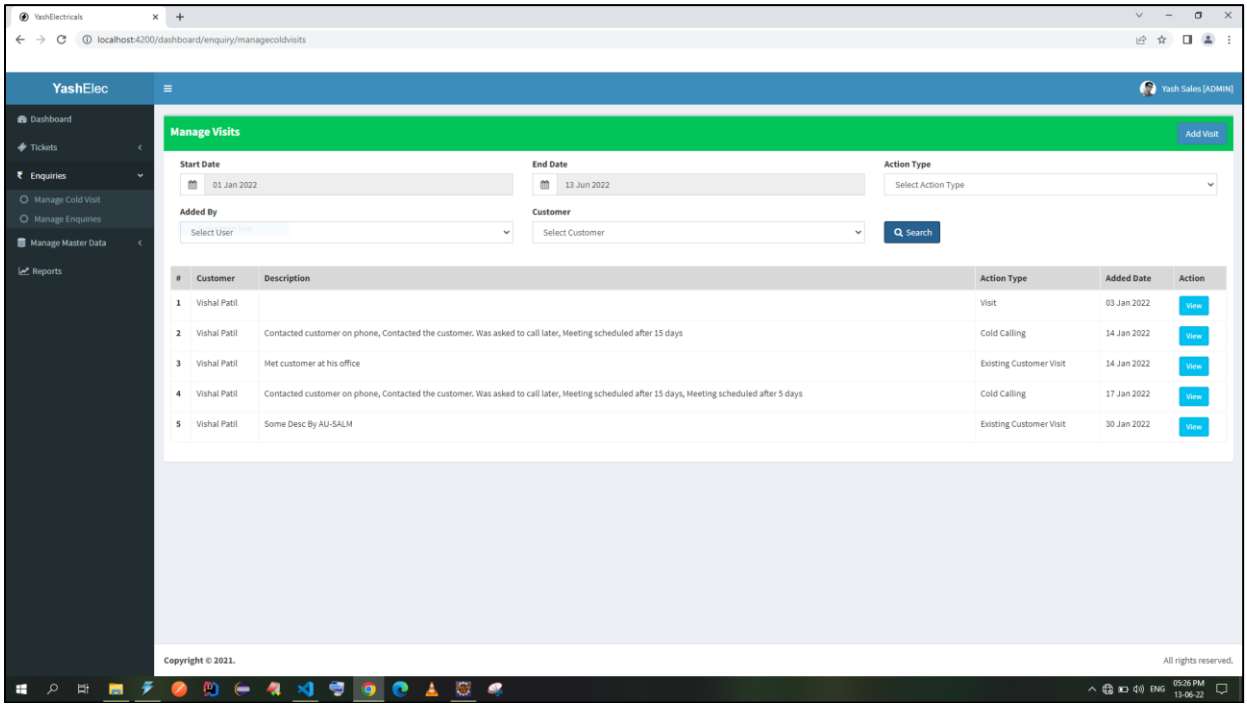


*Manage Enquiries: Enquiry Details after enquiry lost. Enquiry Activities are changed.

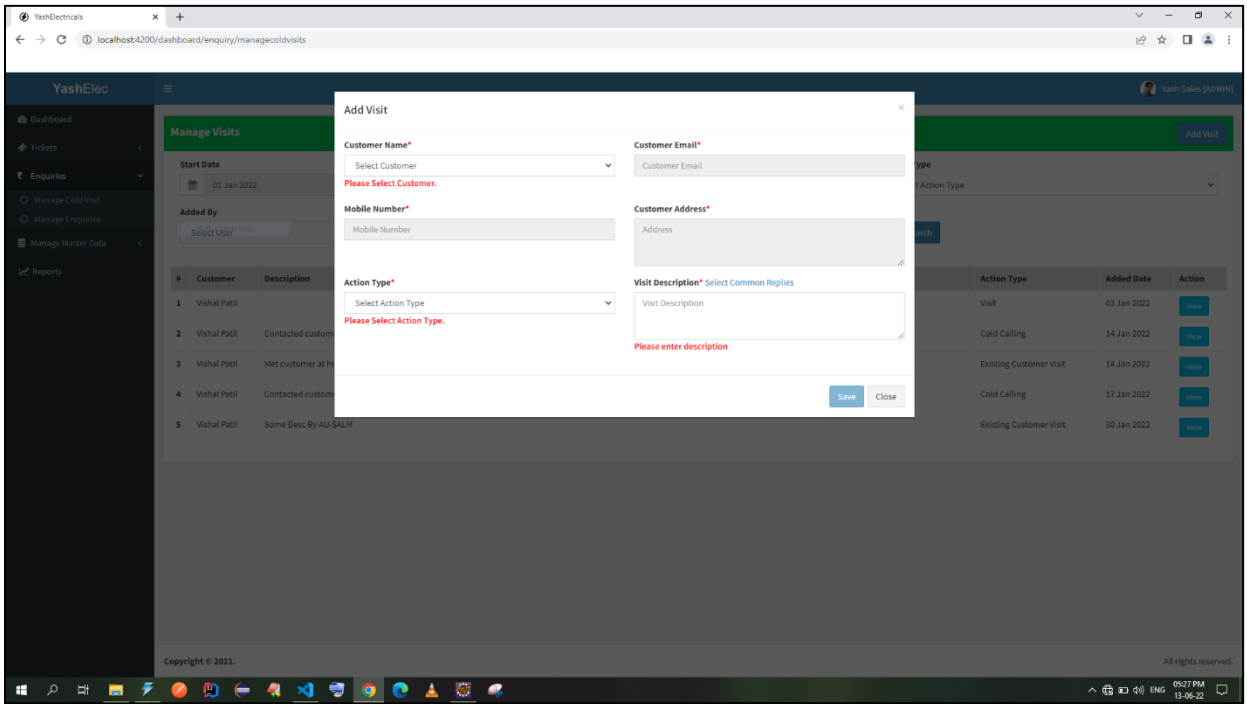


Visit Module Screen Snaps

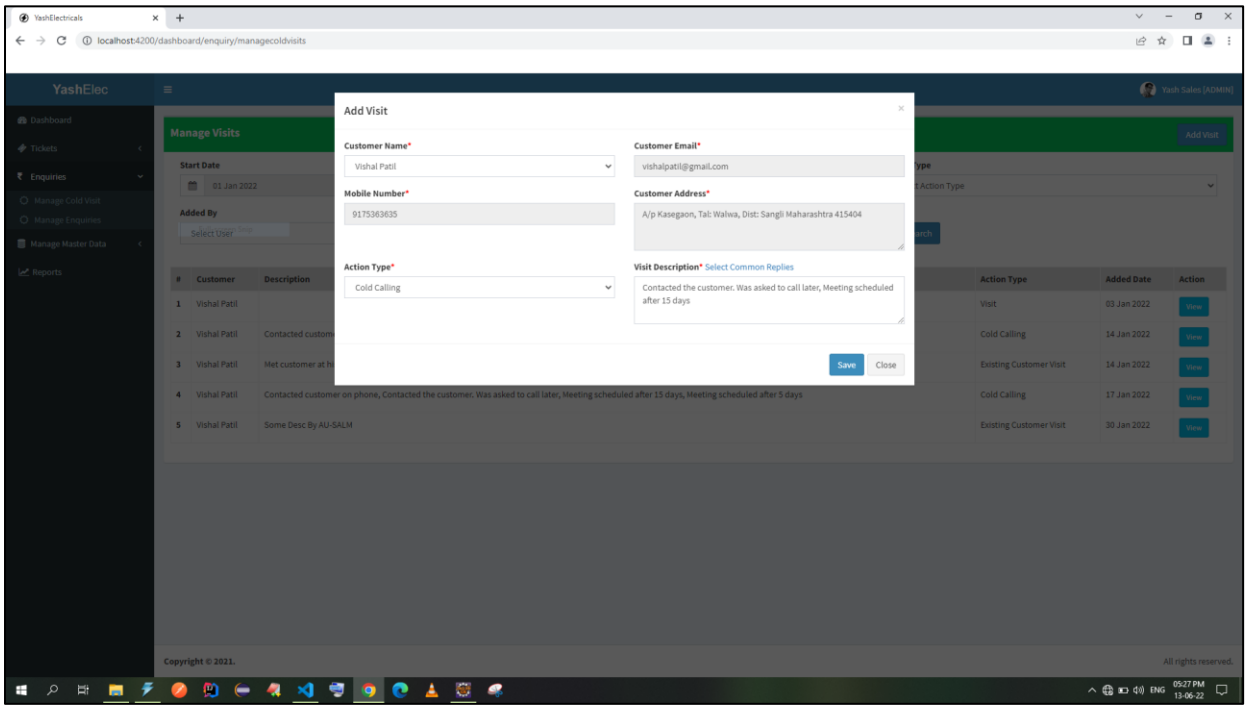
*Manage Visits: View Visits



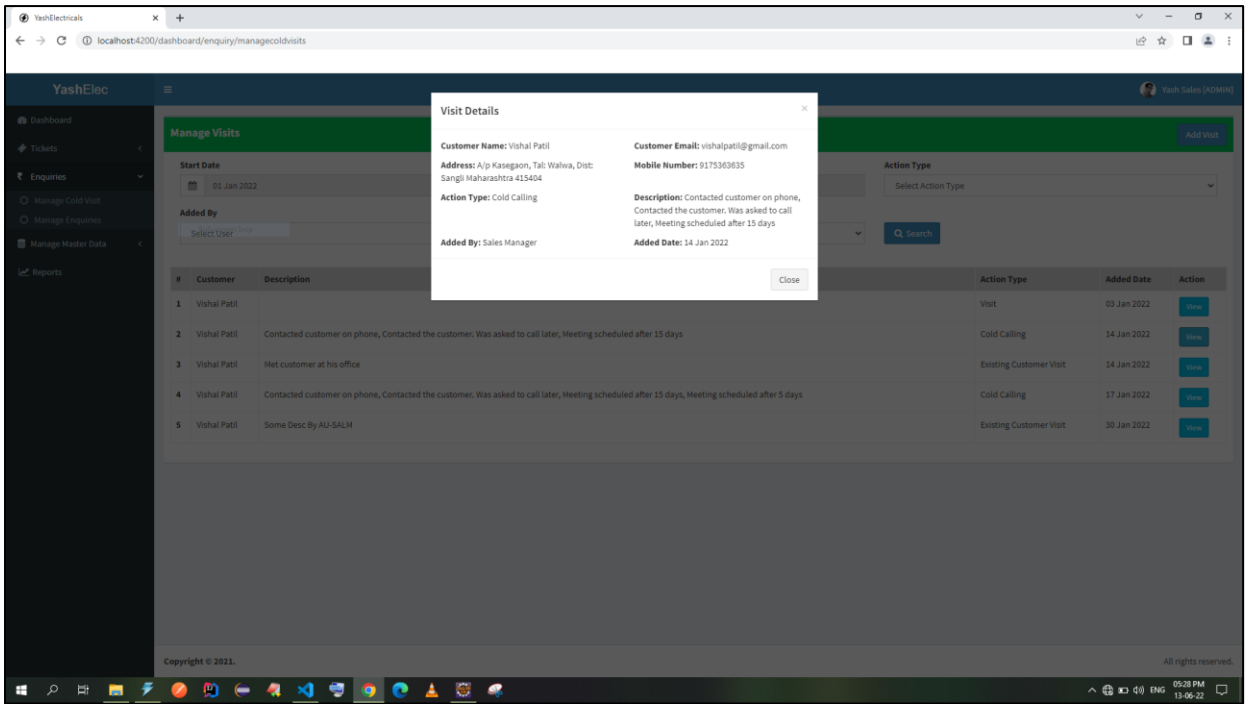
*Manage Visits: Add Visit Validations



*Manage Visits: Add Visit

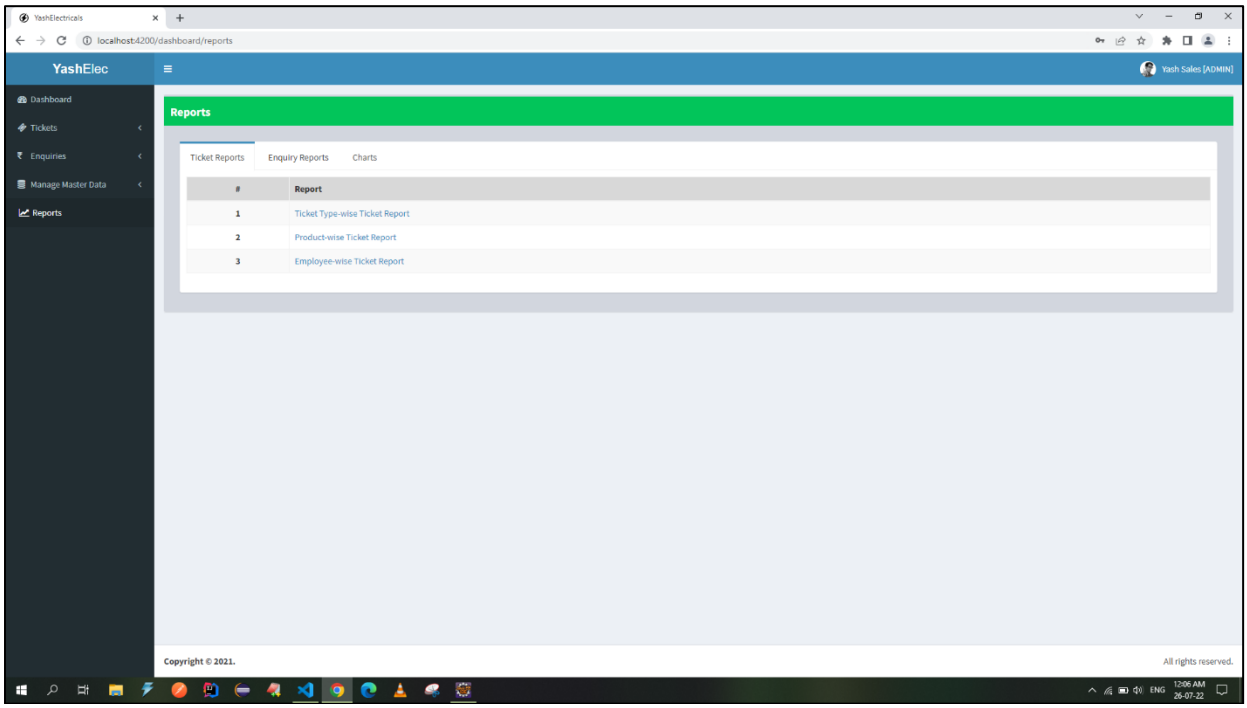


*Manage Visits: View Visit Details



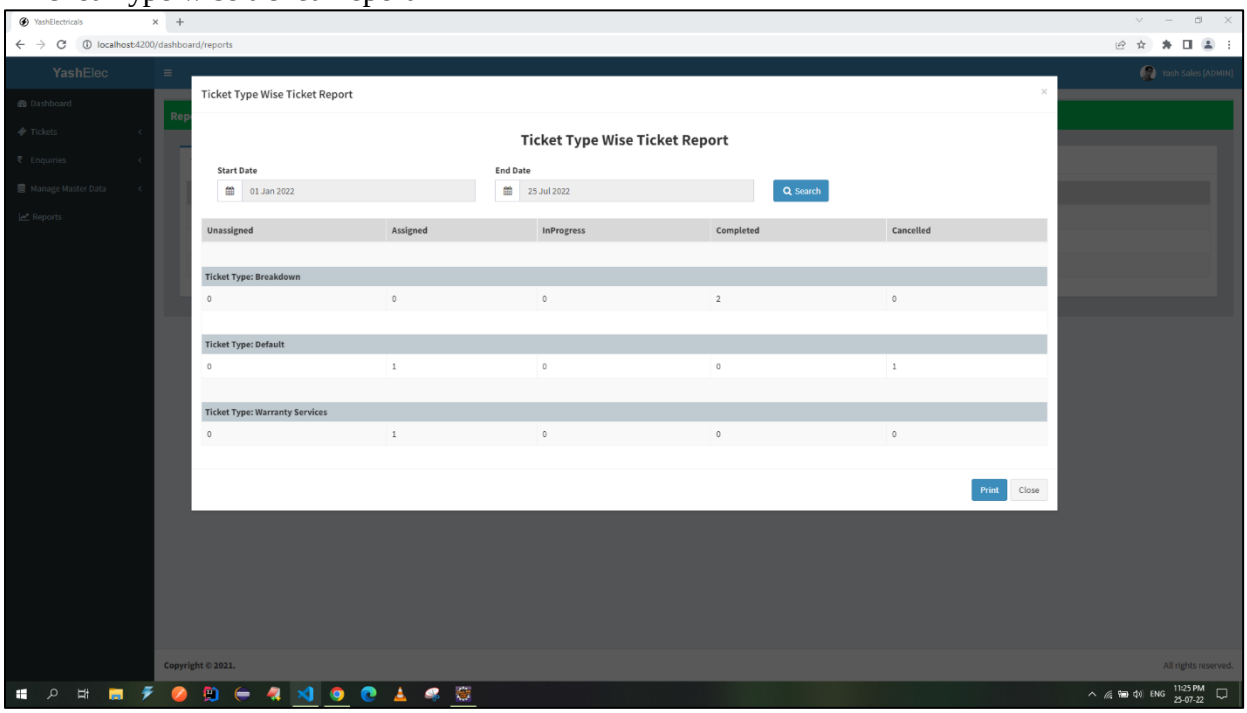
Reports Screen Snaps

*Reports Dashboard

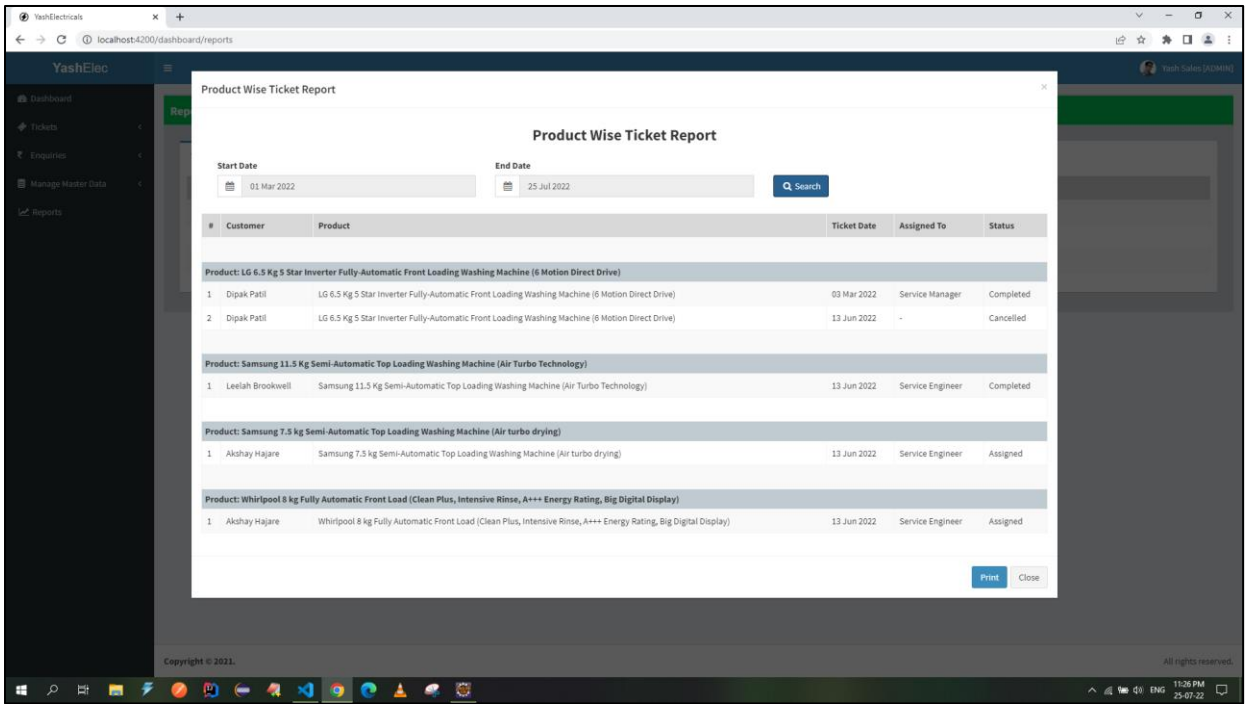


Ticket Reports

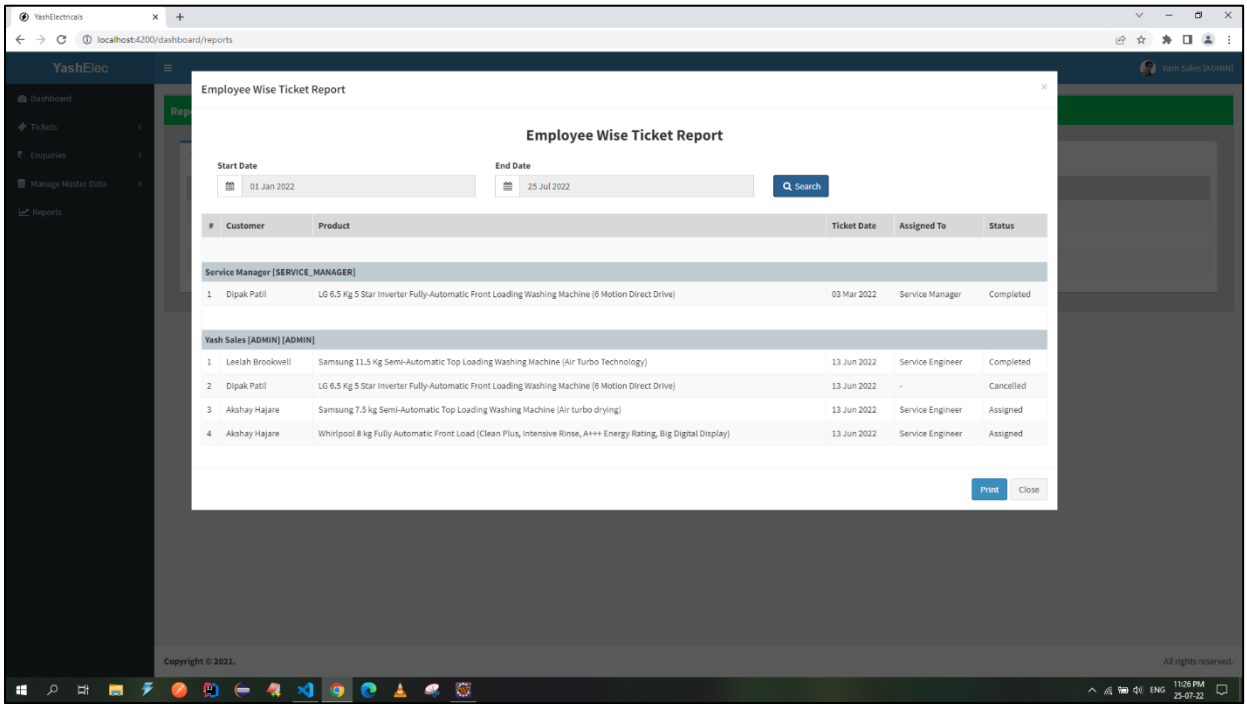
*Ticket Type wise ticket Report



*Product wise Ticket Report

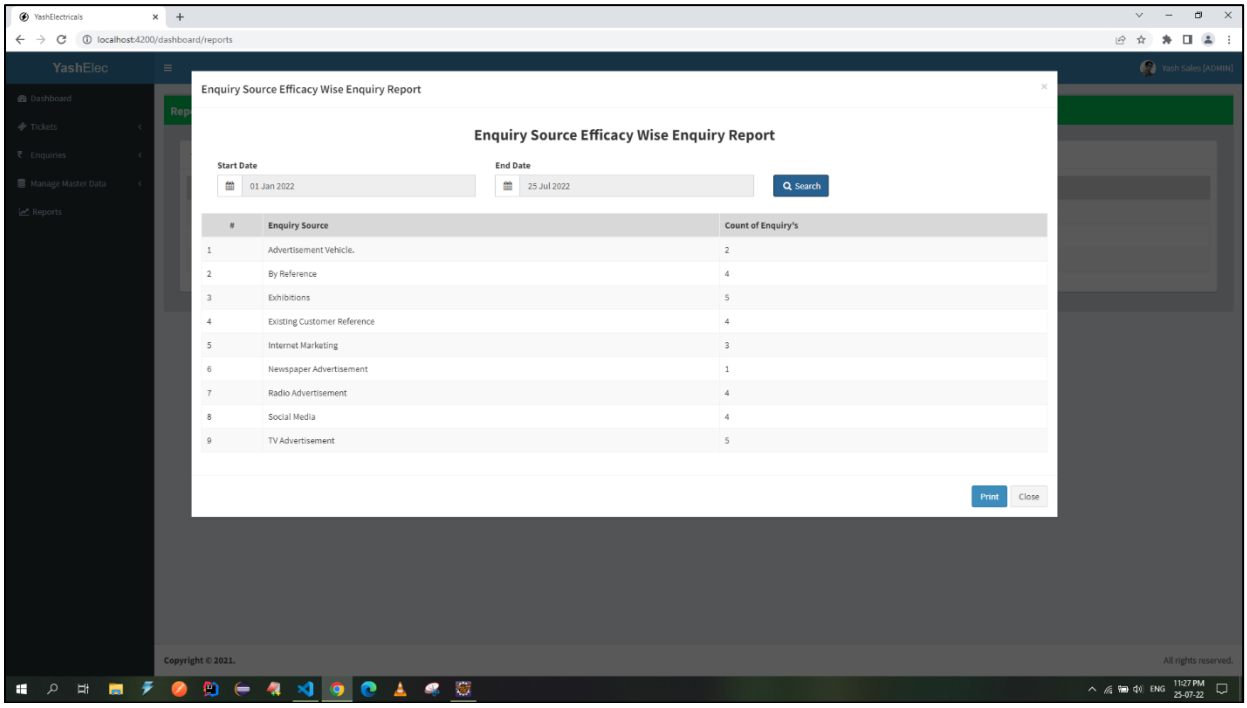


*Employee Wise Ticket Report

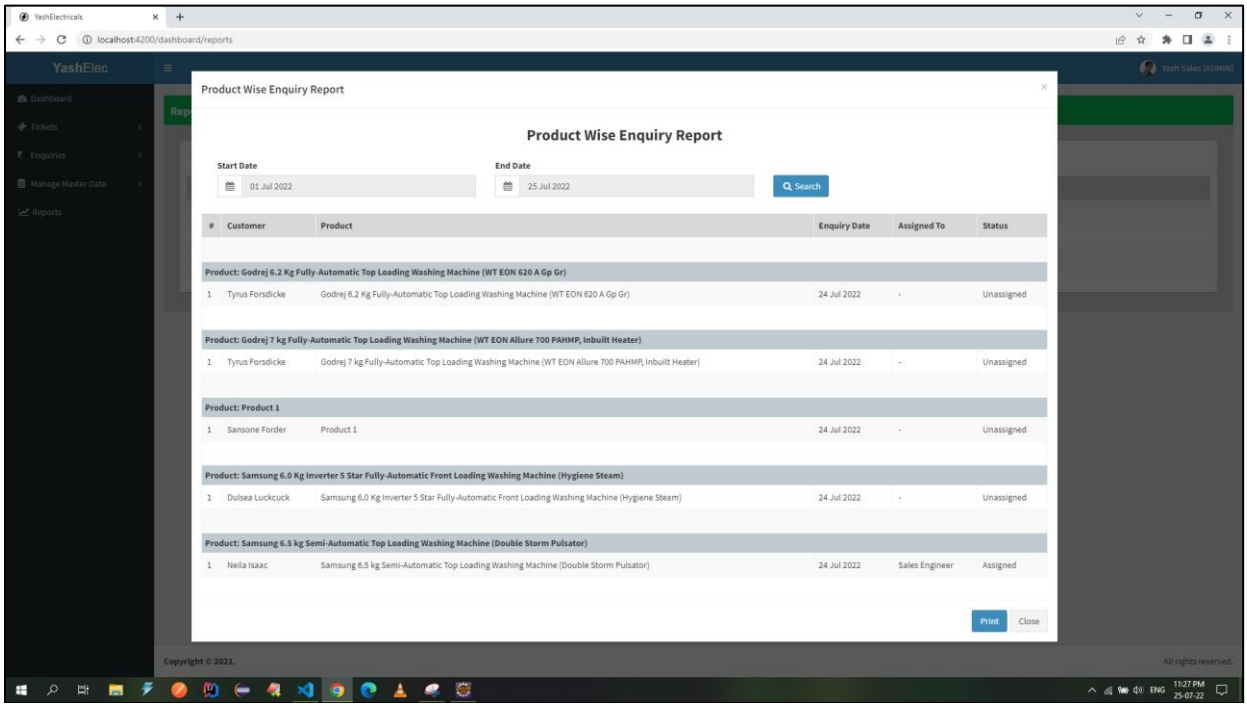


Enquiries Report

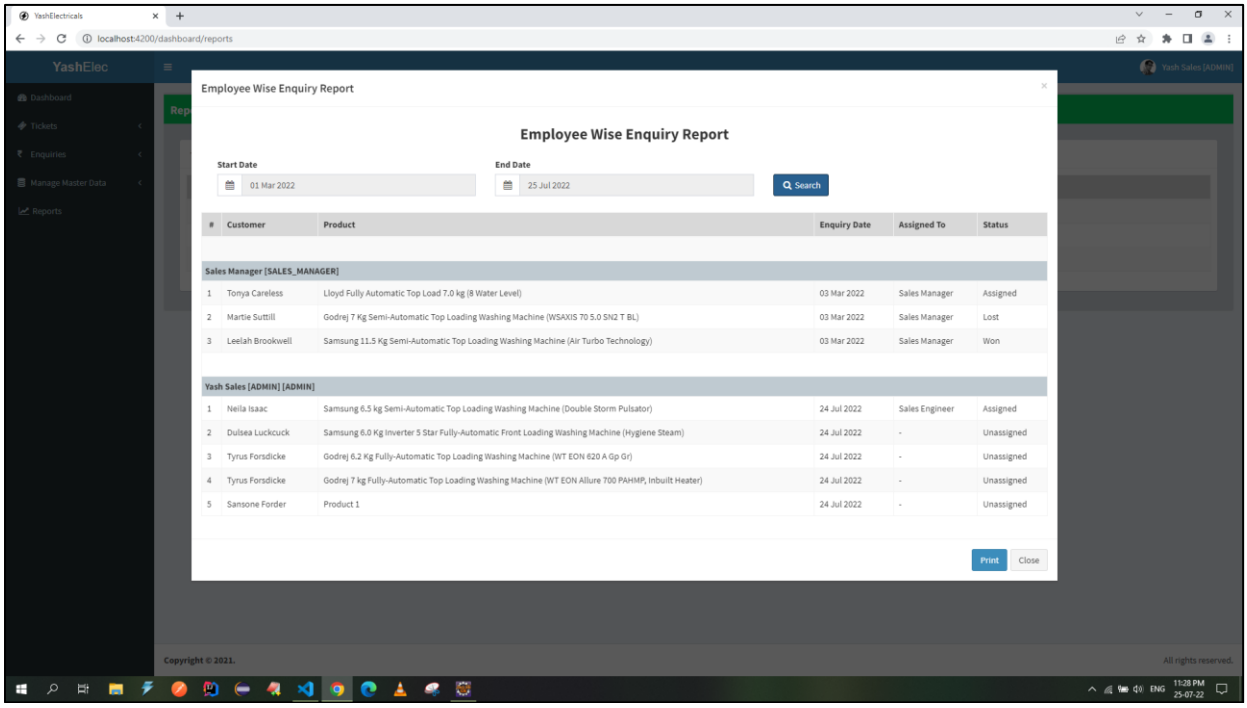
*Enquiry Source Efficacy Wise Enquiry Report



*Product Wise Enquiry Report

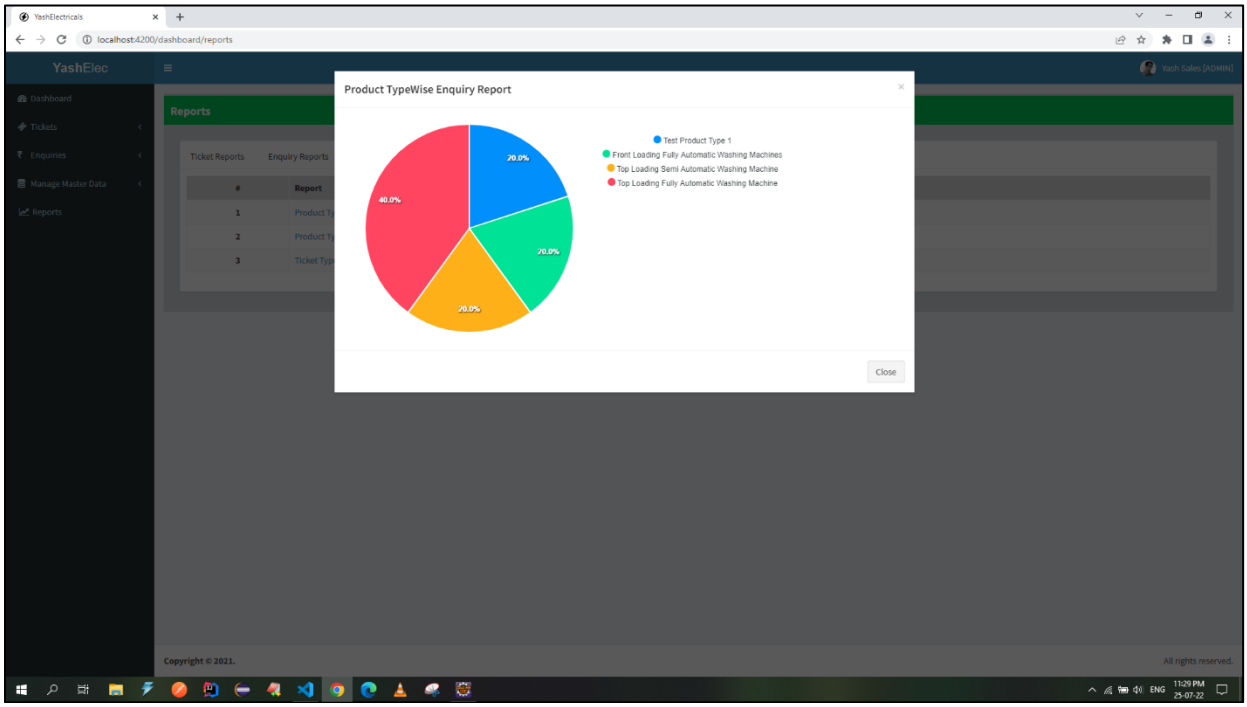


*Employee Wise Enquiry Report

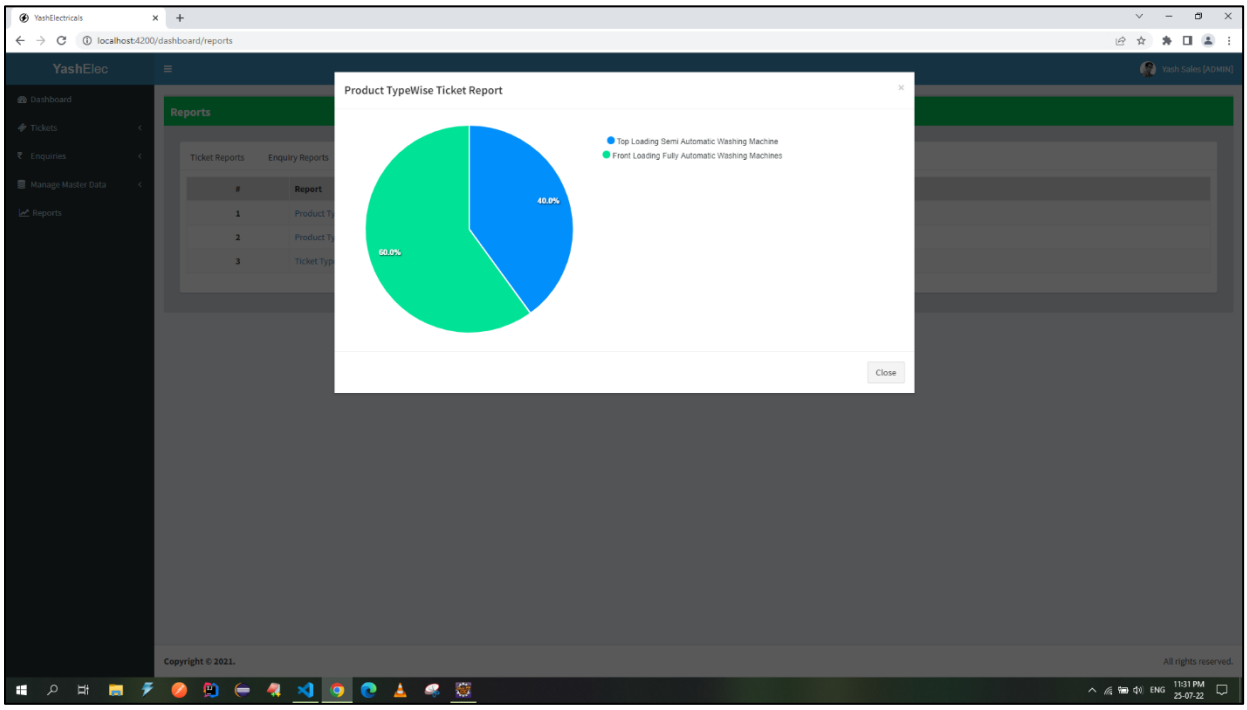


Pie Chart Reports

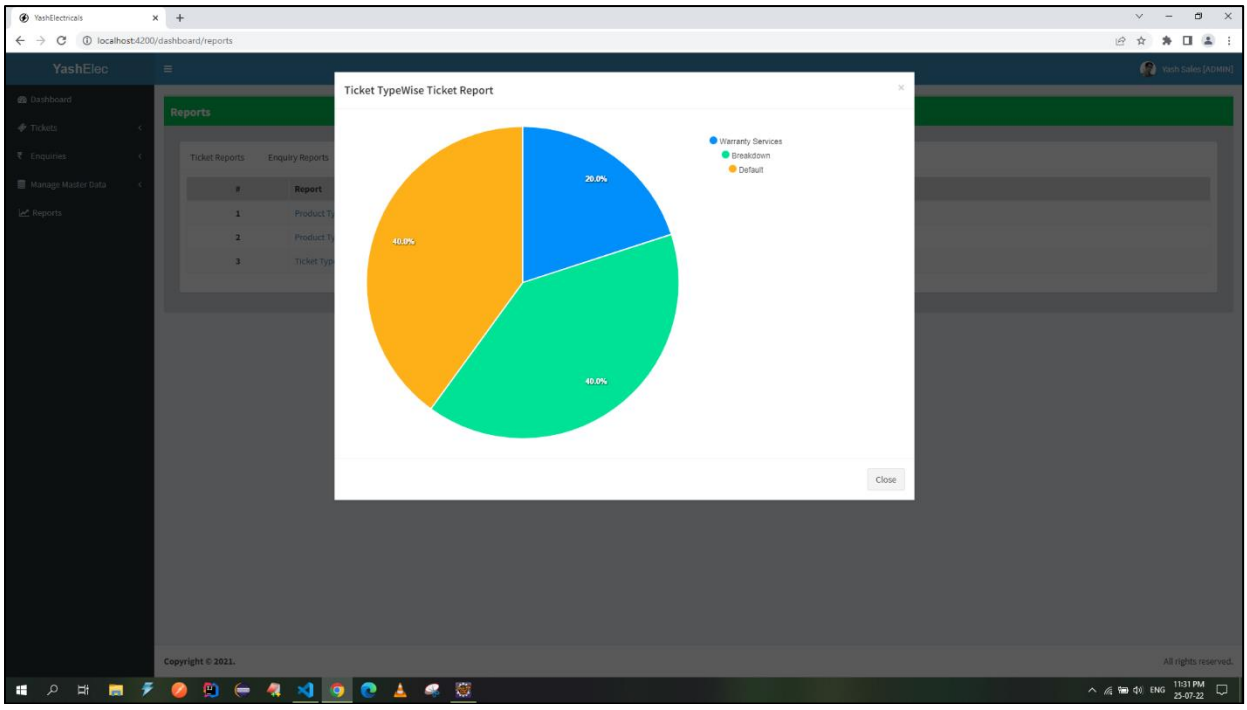
*Product Type Wise Enquiry Report



*Product Type Wise Ticket Report

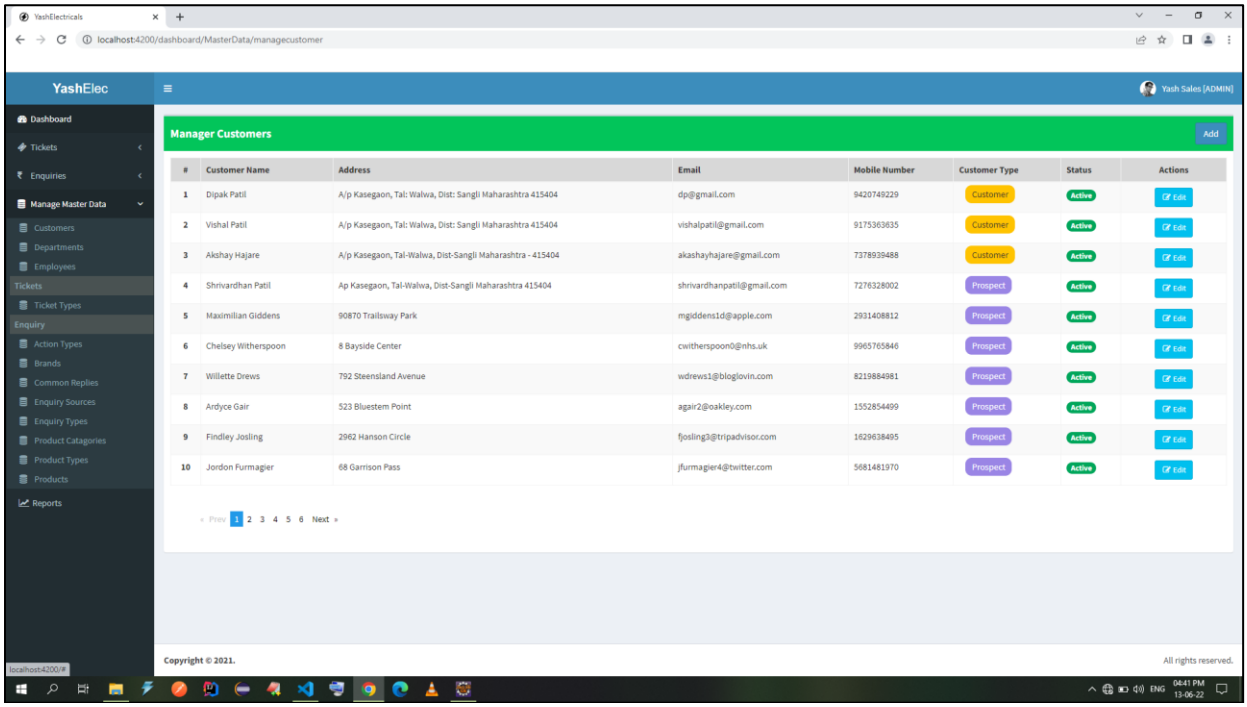


*Ticket Type Wise Ticket Report

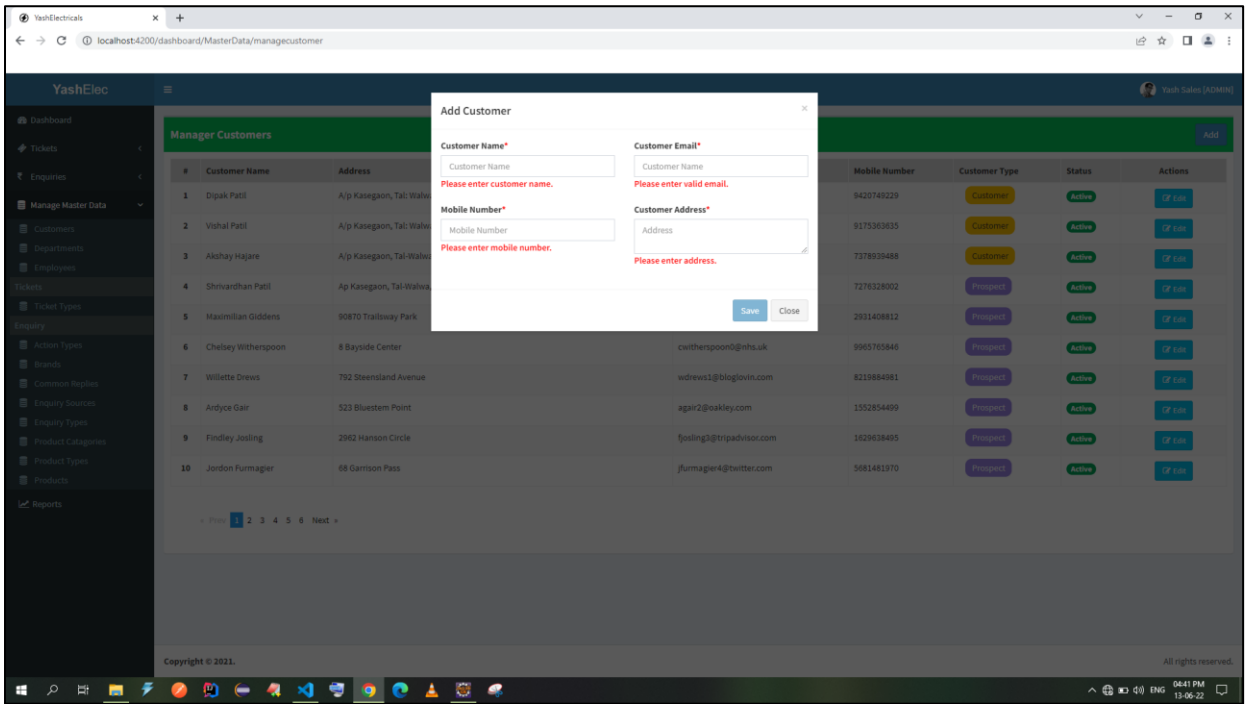


Master Data Screen Snaps

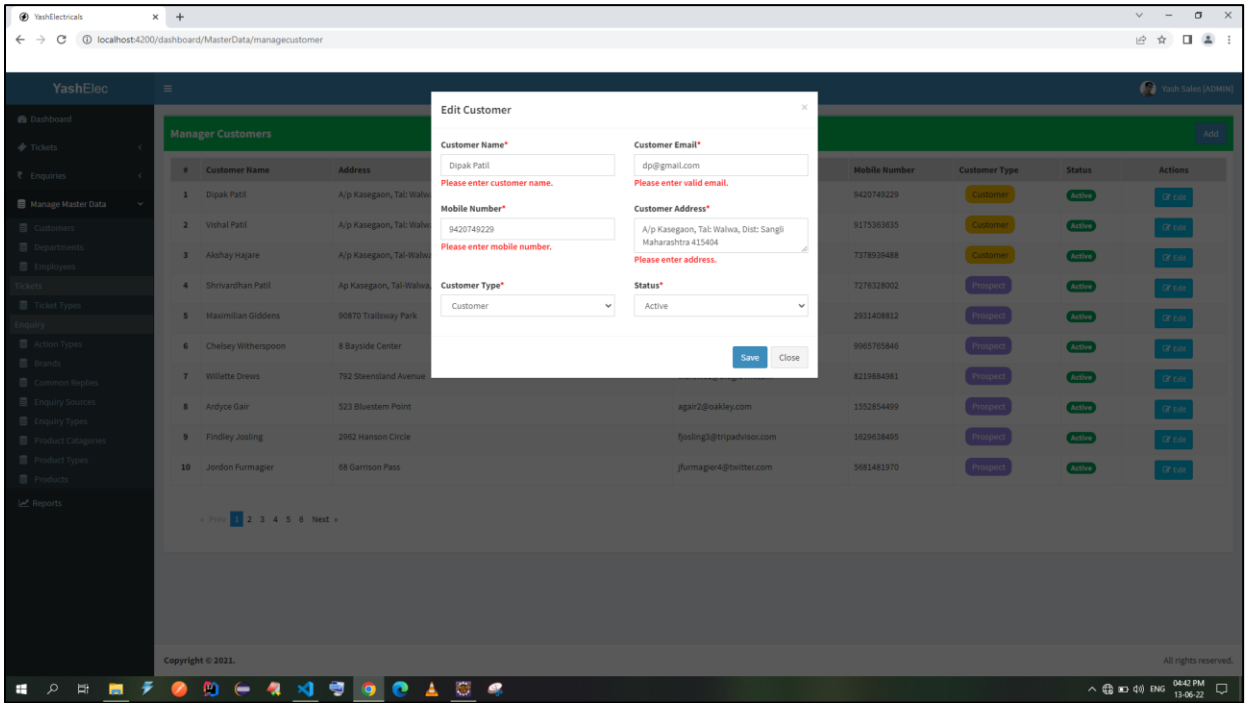
*Manage Customer: View Customer



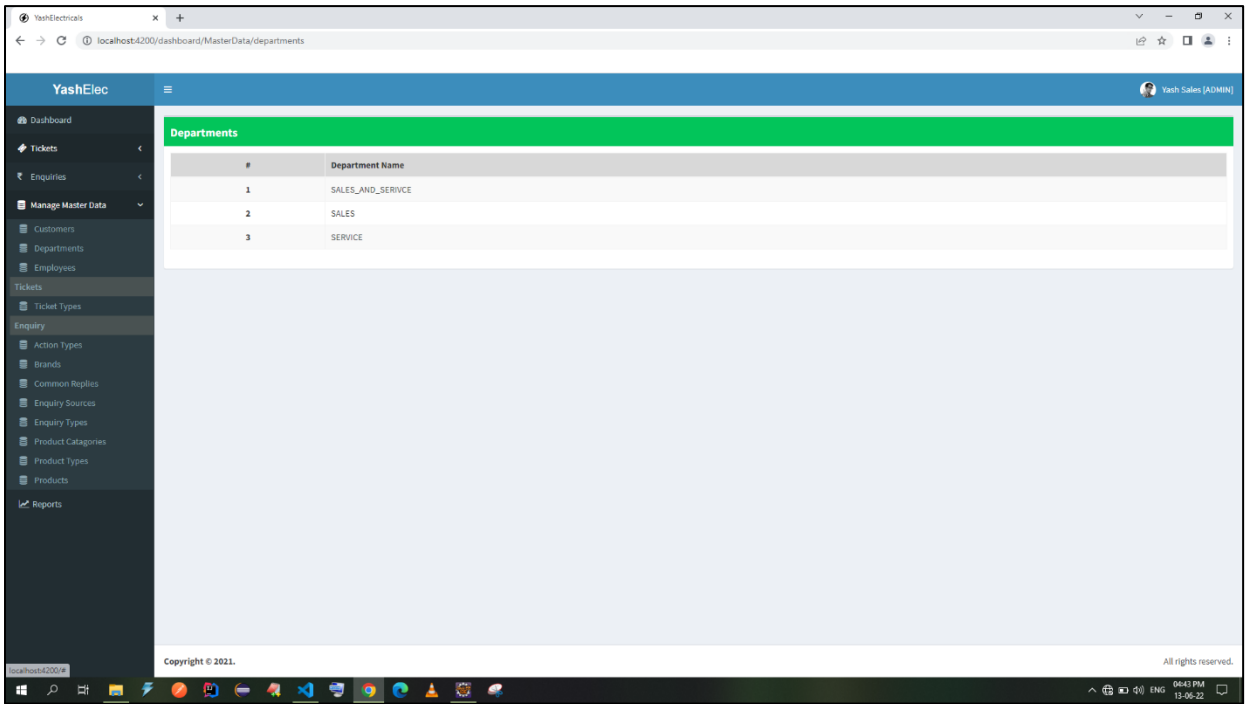
*Manage Customer: Add Customer Form



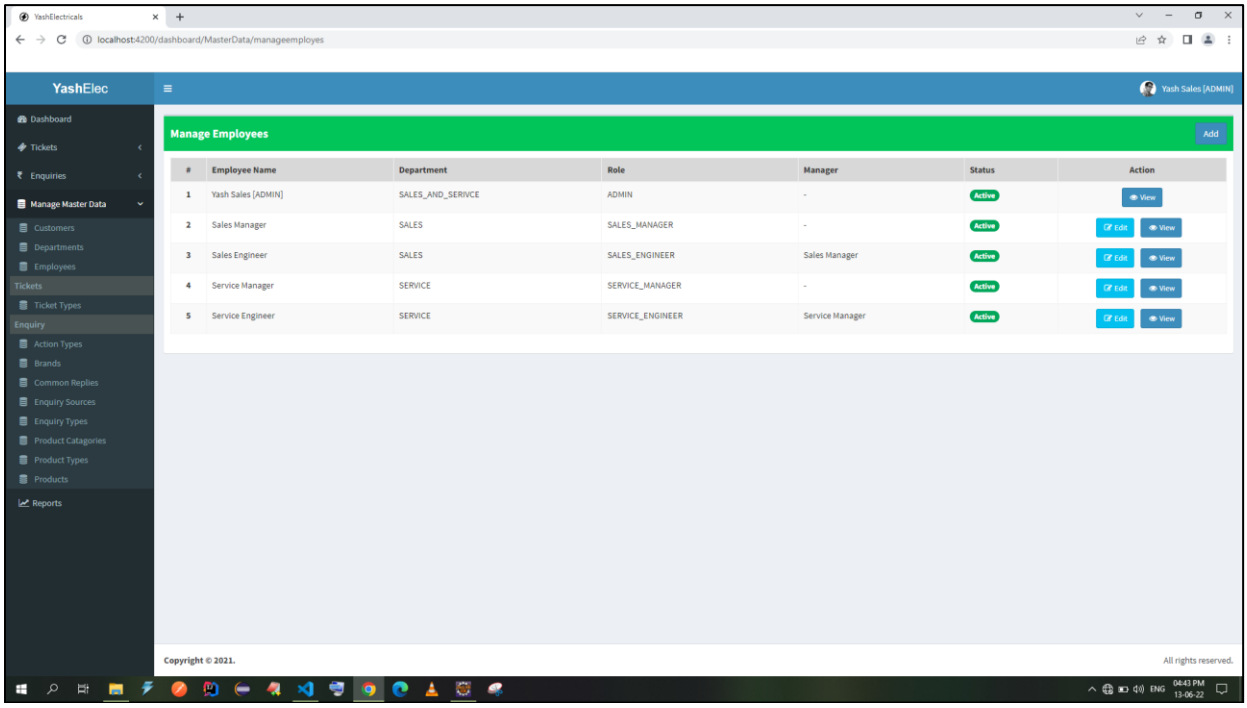
*Manage Customer: Edit Customer



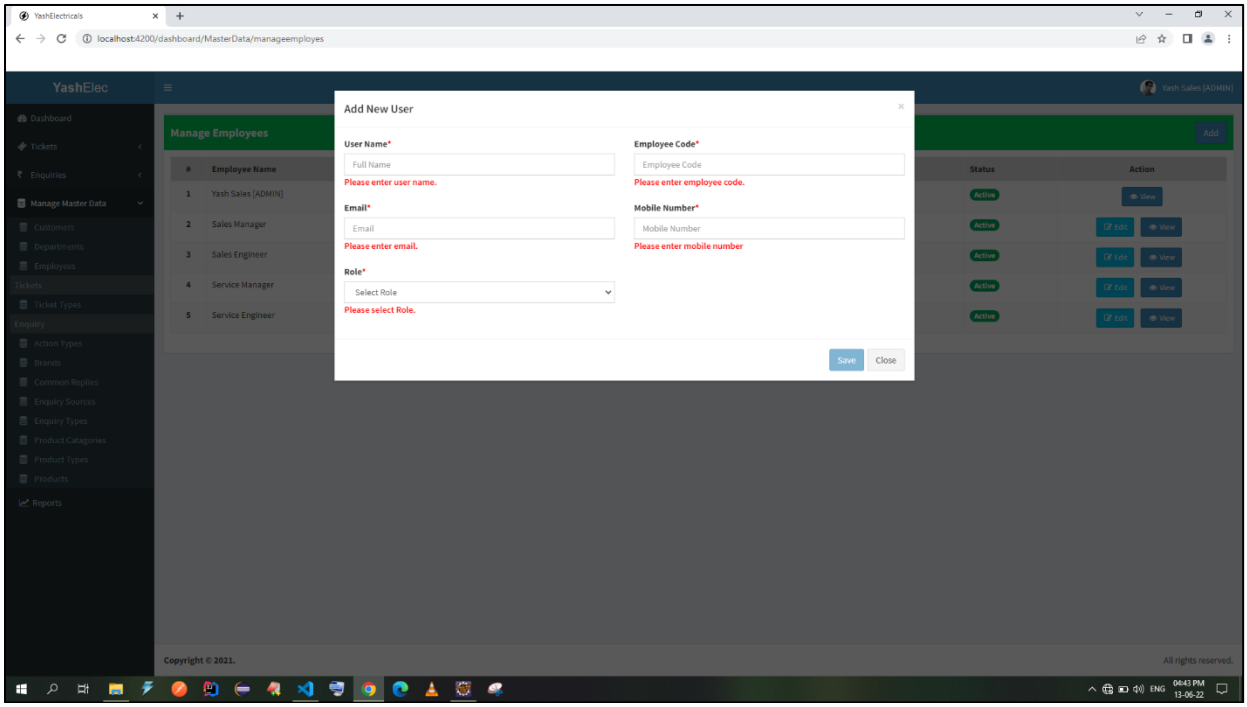
*Departments: List of Departments



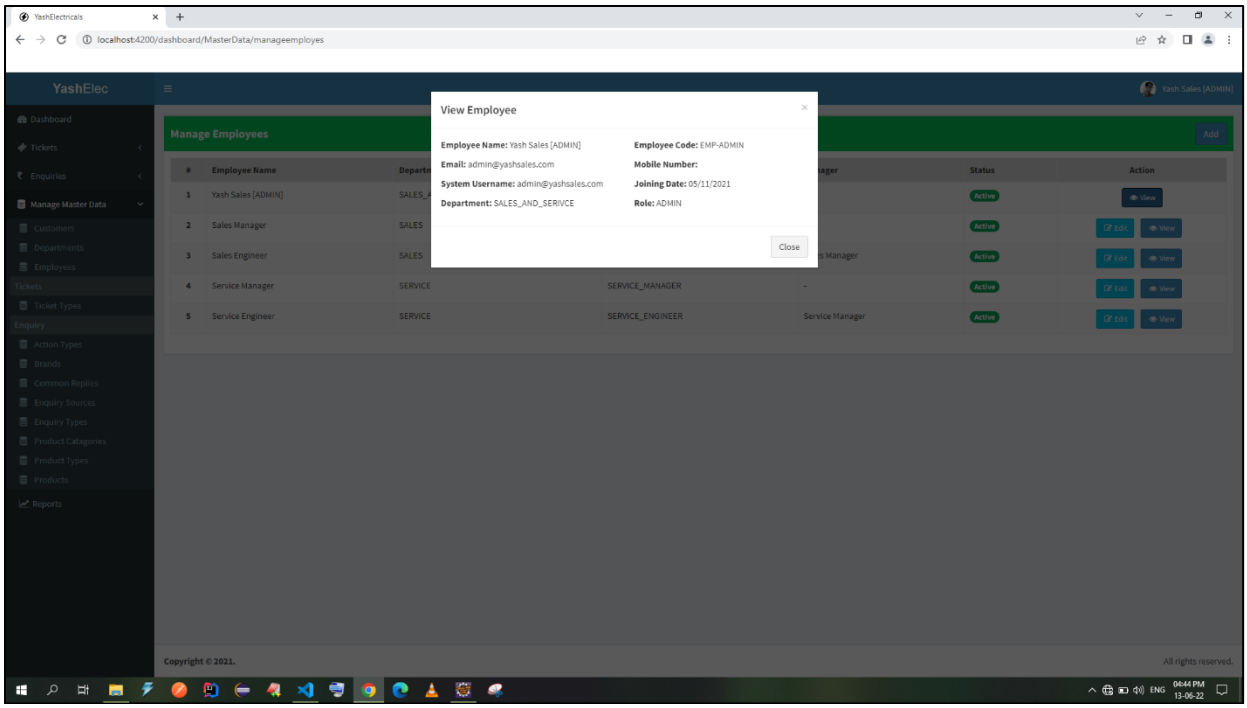
*Manage Employees: List of Employees



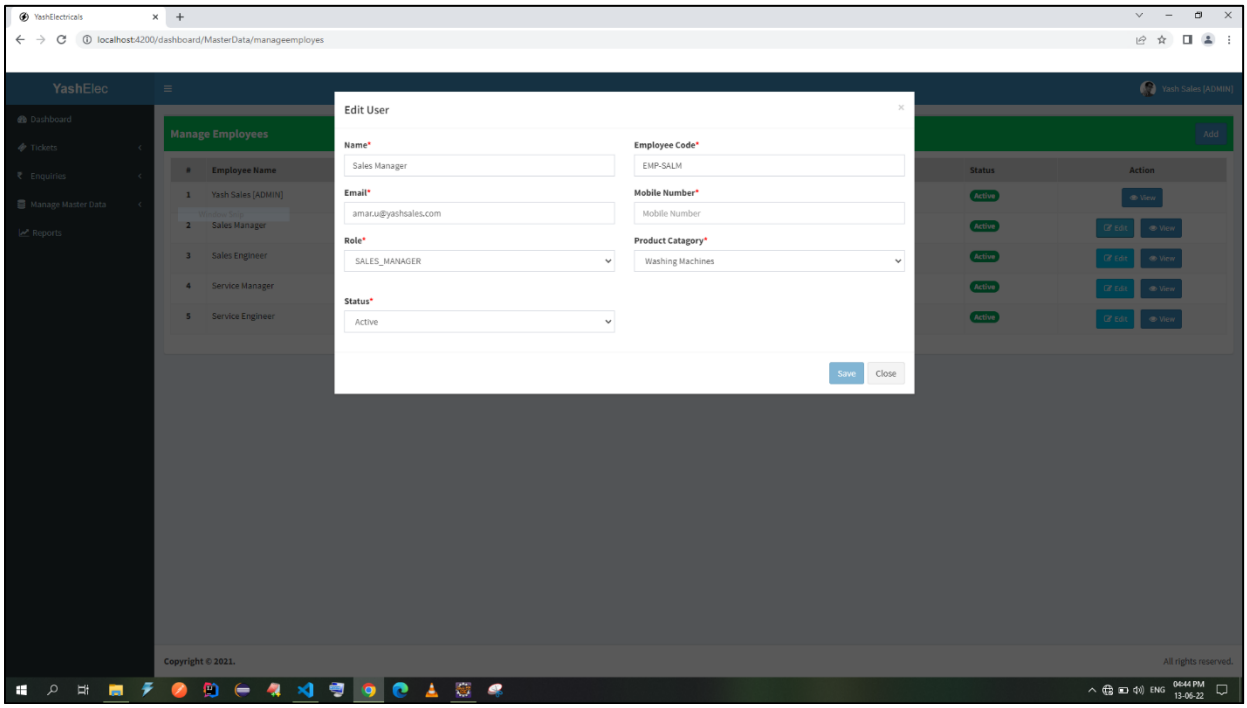
*Employee: Add New Employee



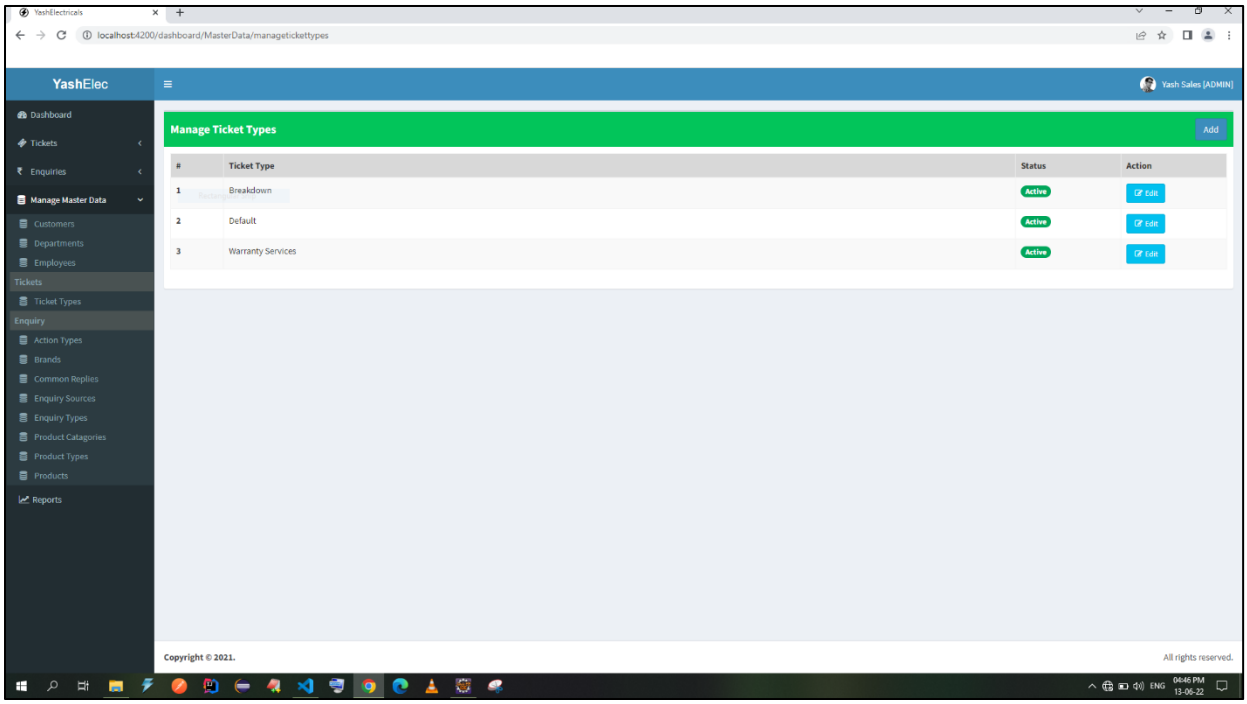
*Employee: View Employee Details



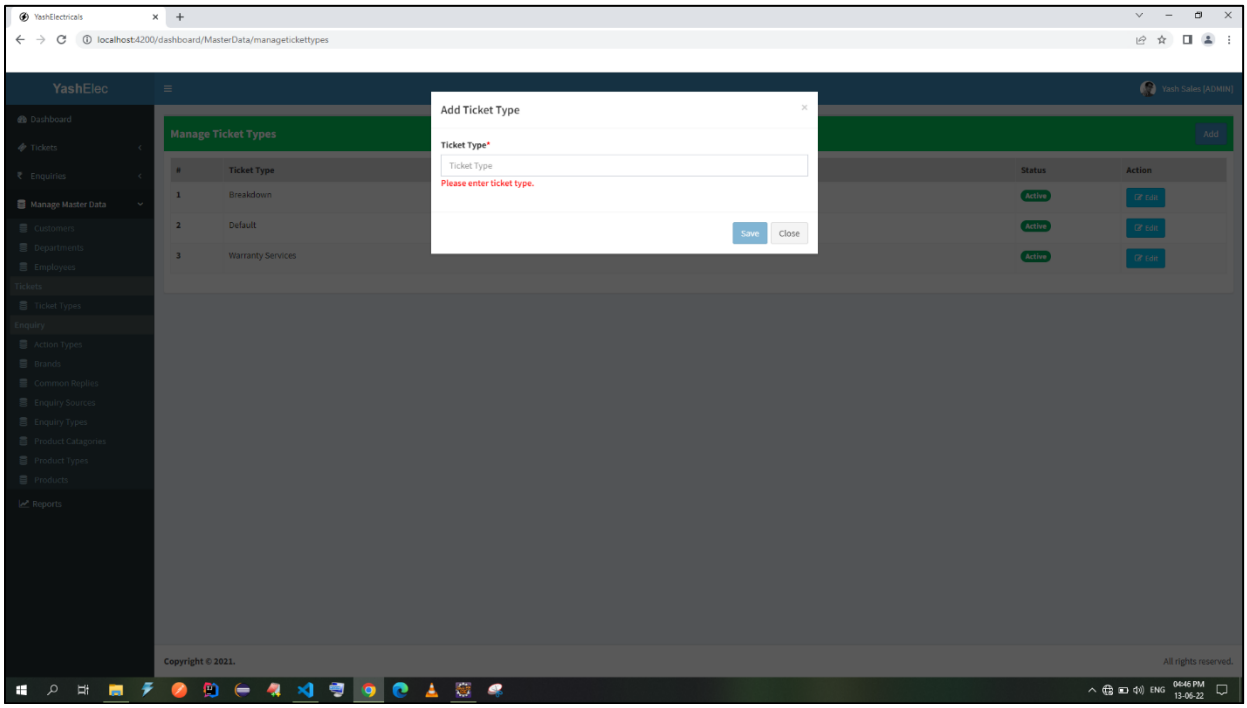
*Employee: Edit Employee Details



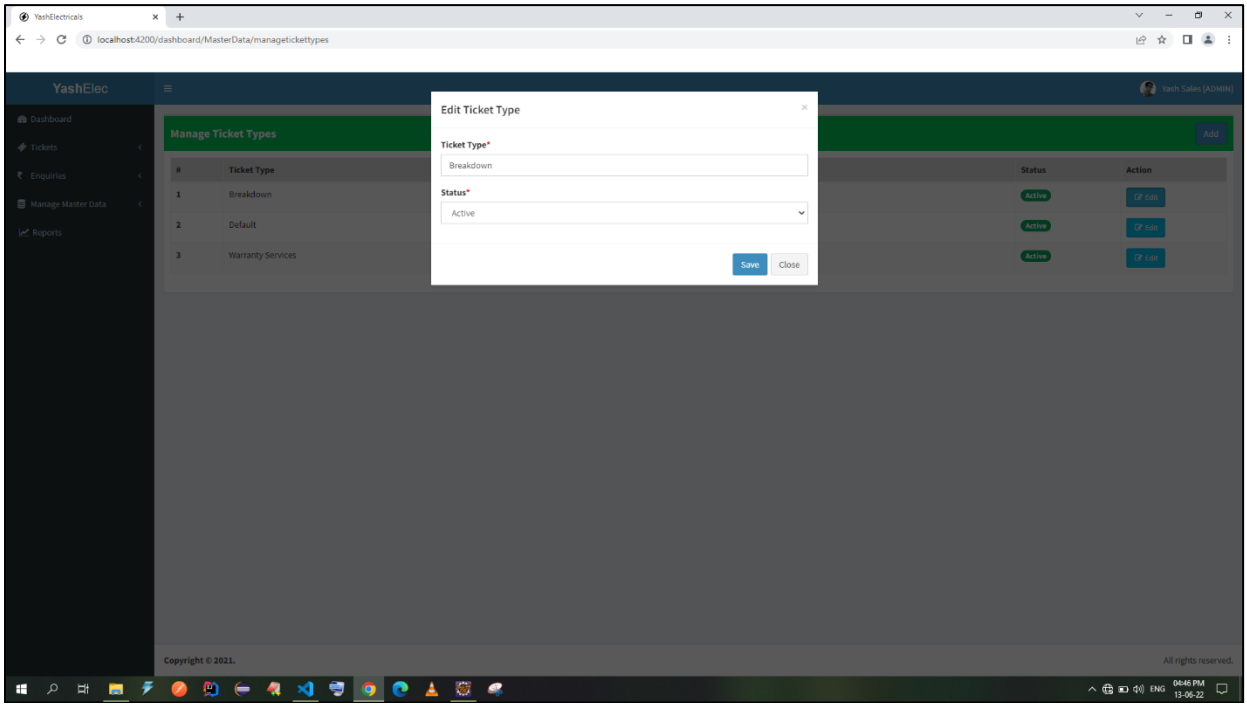
*Manage Ticket Types: View Ticket Types



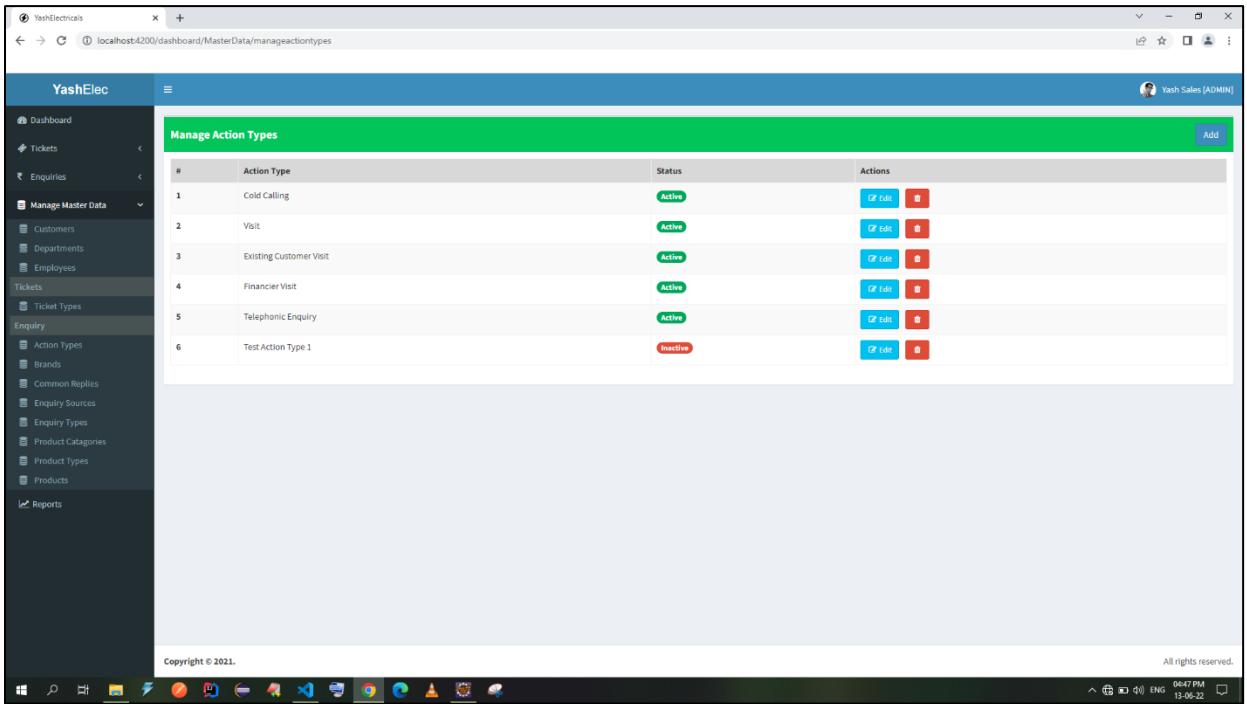
*Manage Ticket Types: Add Ticket Type



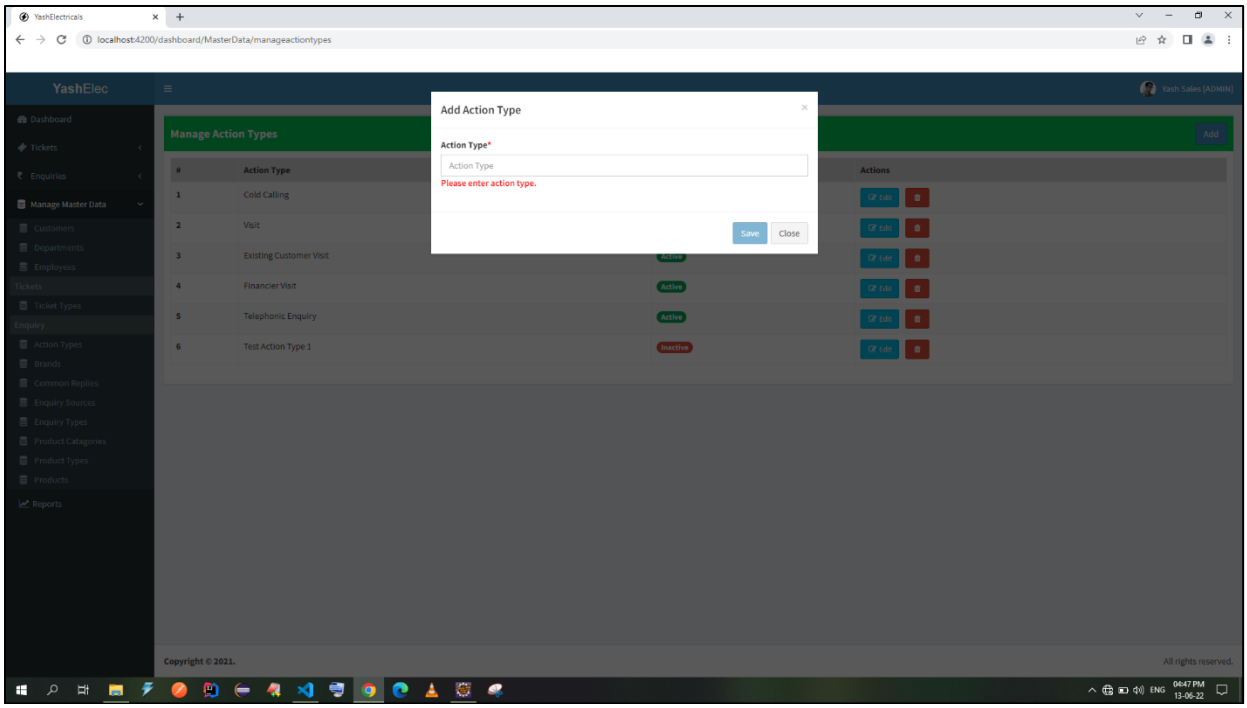
*Manage Ticket Types: Edit Ticket Types



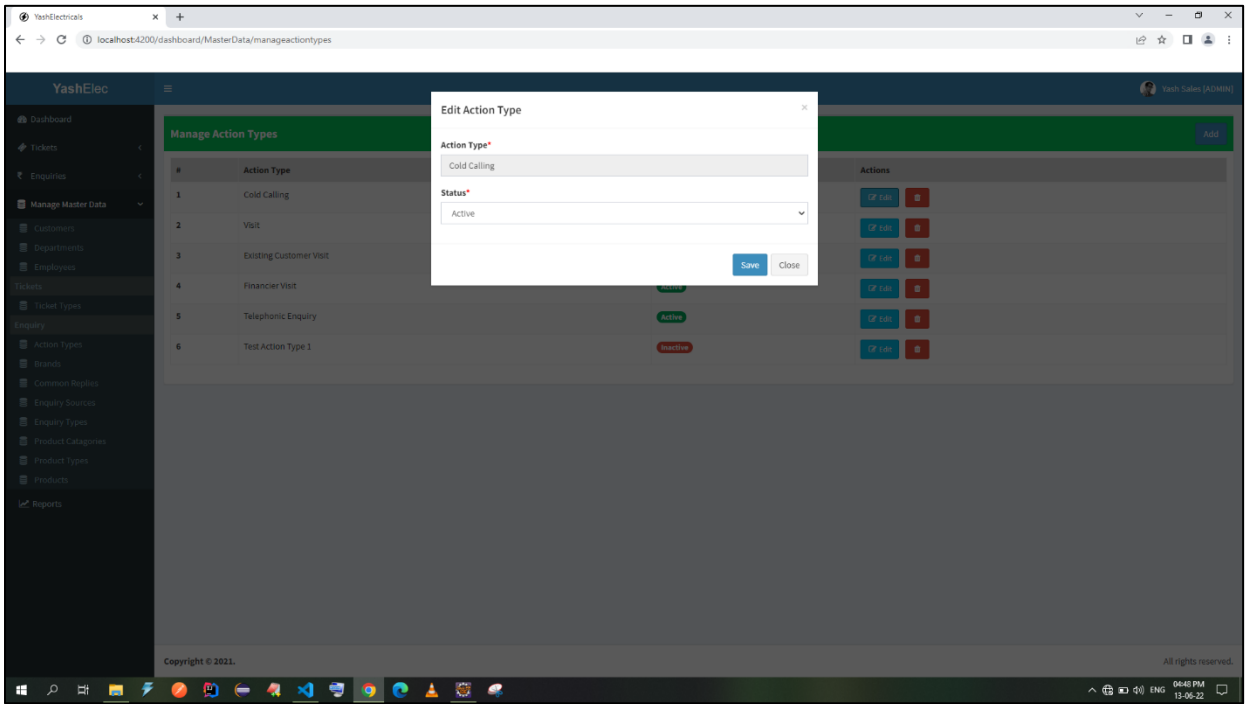
*Manage Action Types: View Action Types



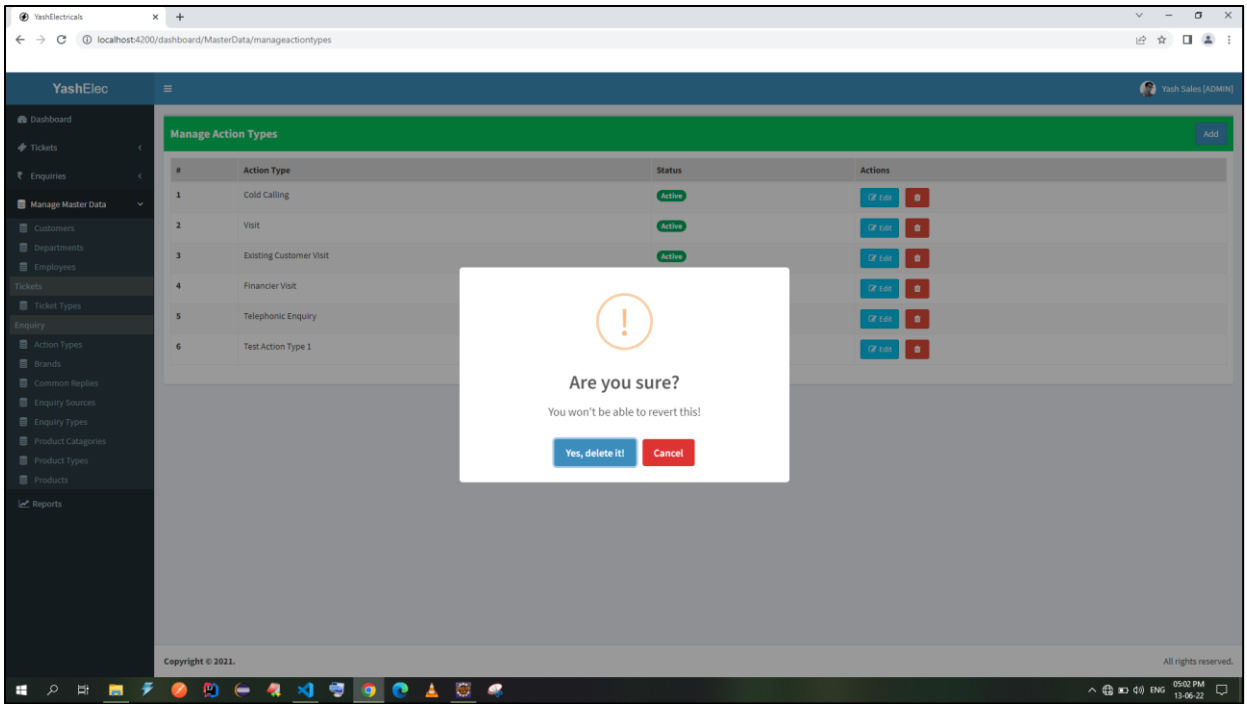
*Manage Action Types: Add Action Types



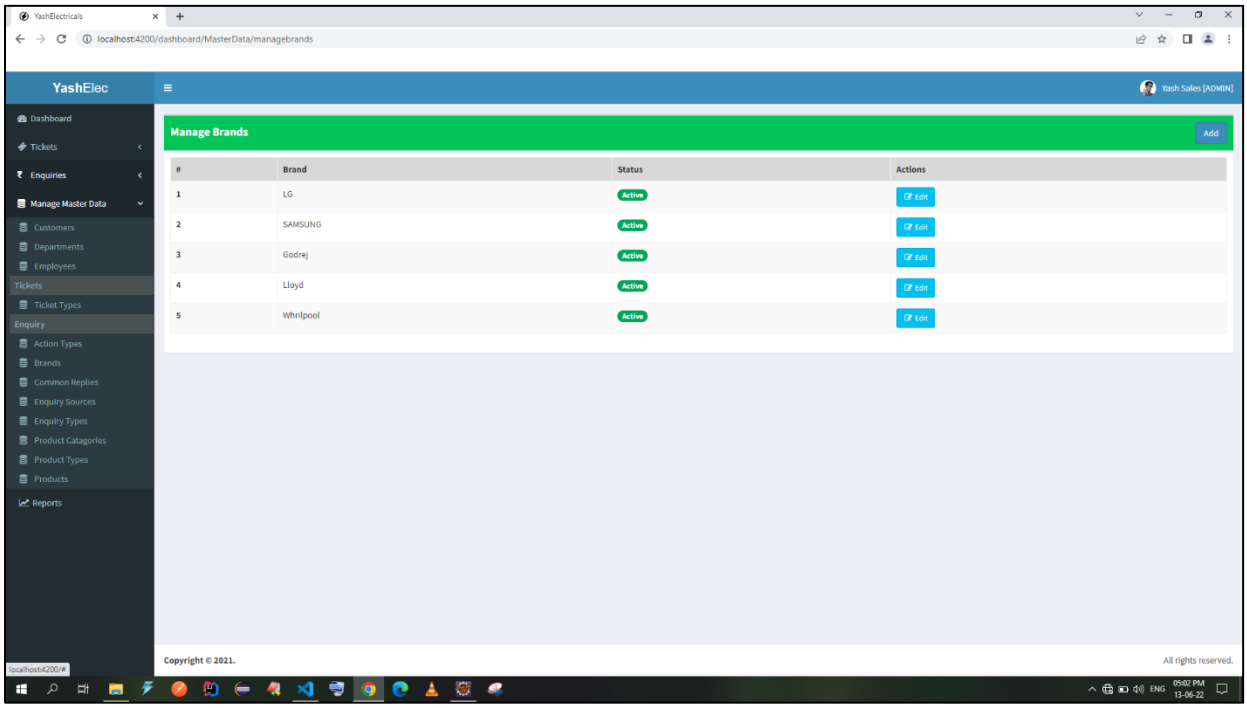
*Manage Action Types: Edit Action Types



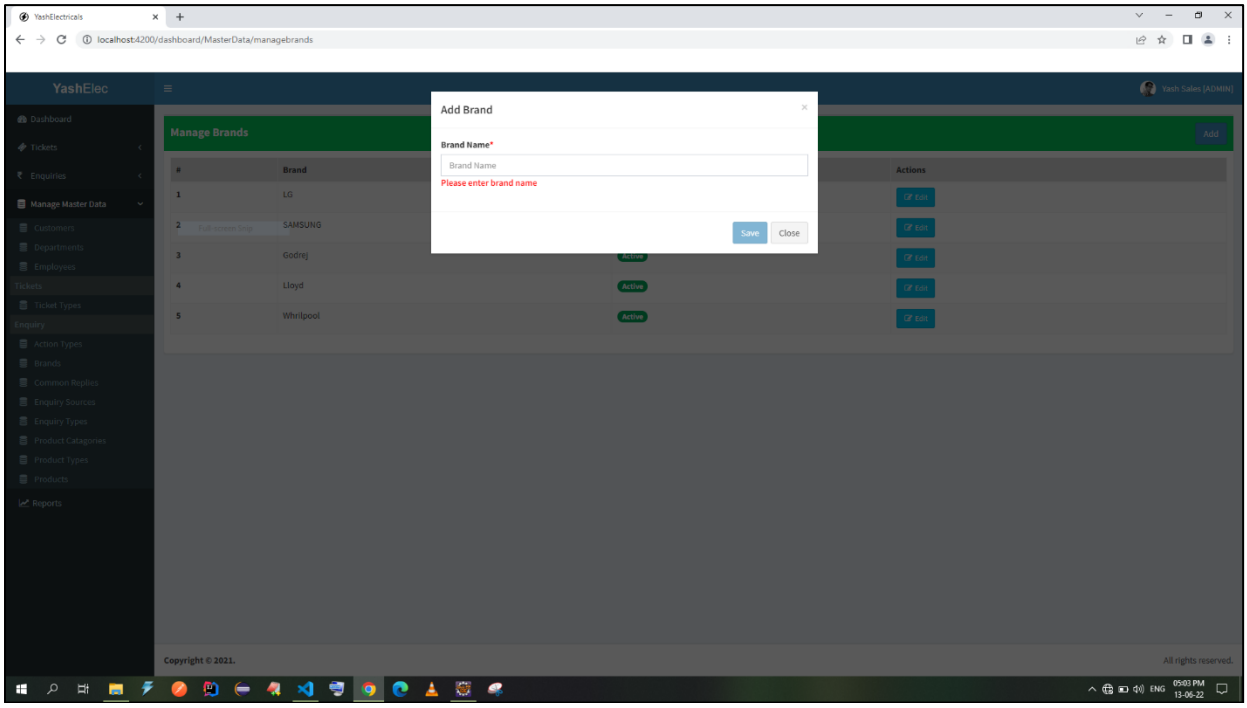
*Manage Action Types: Delete Action Types



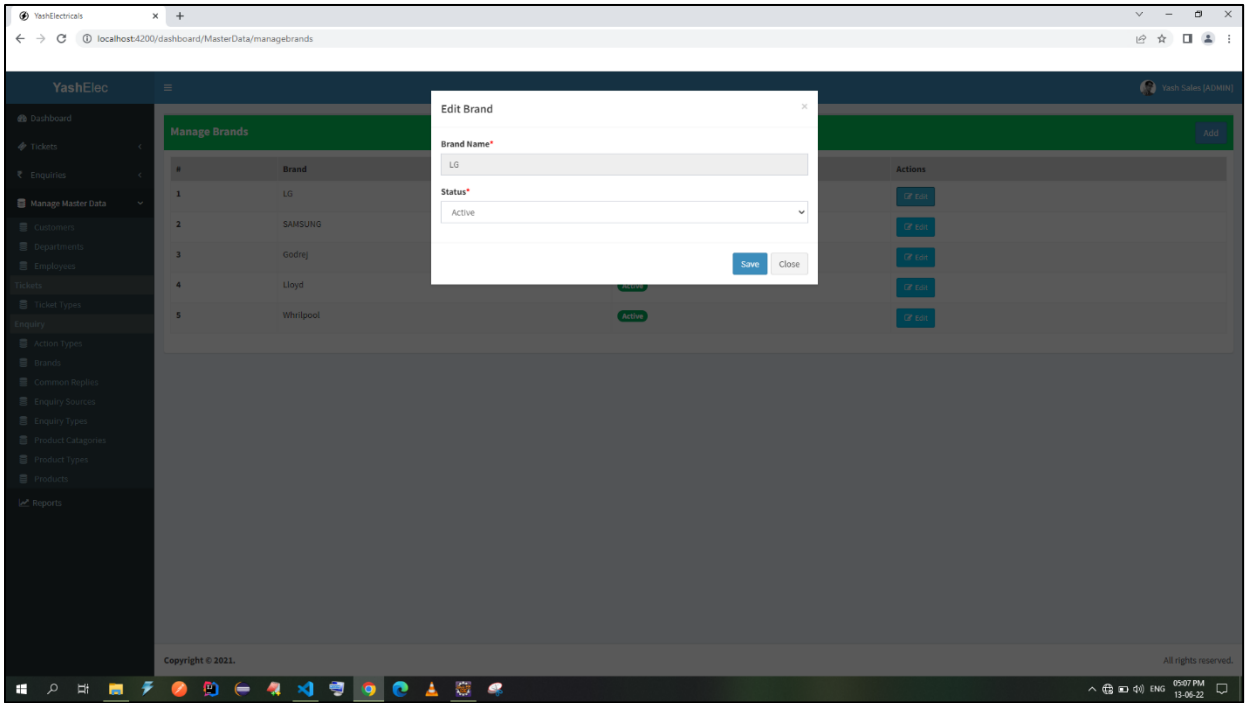
*Manage Brands: View Brands



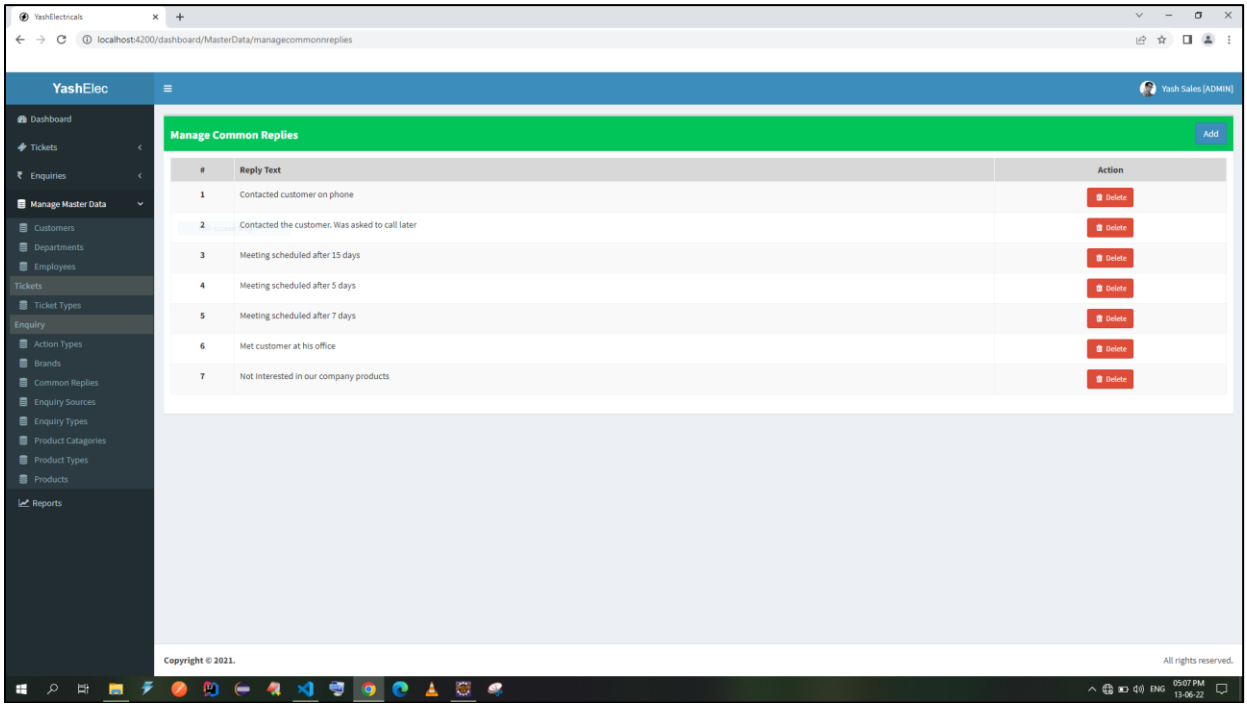
*Manage Brands: Add Brand



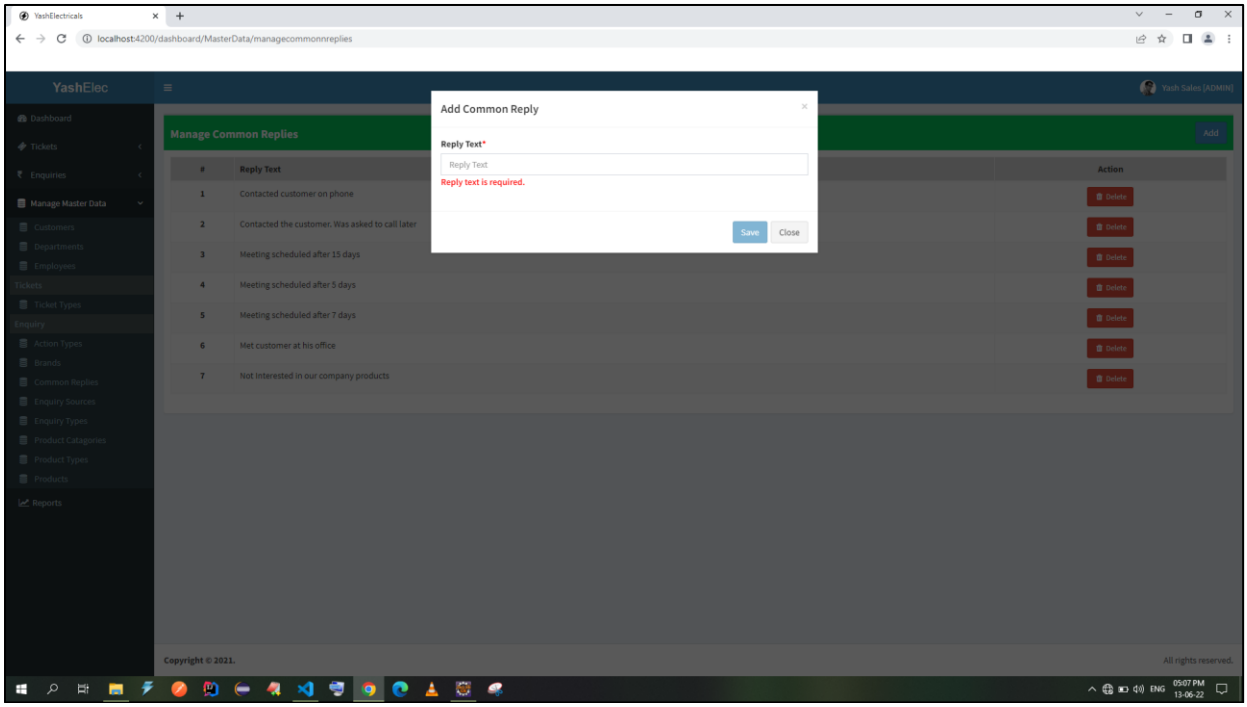
*Manage Brands: Edit Brand



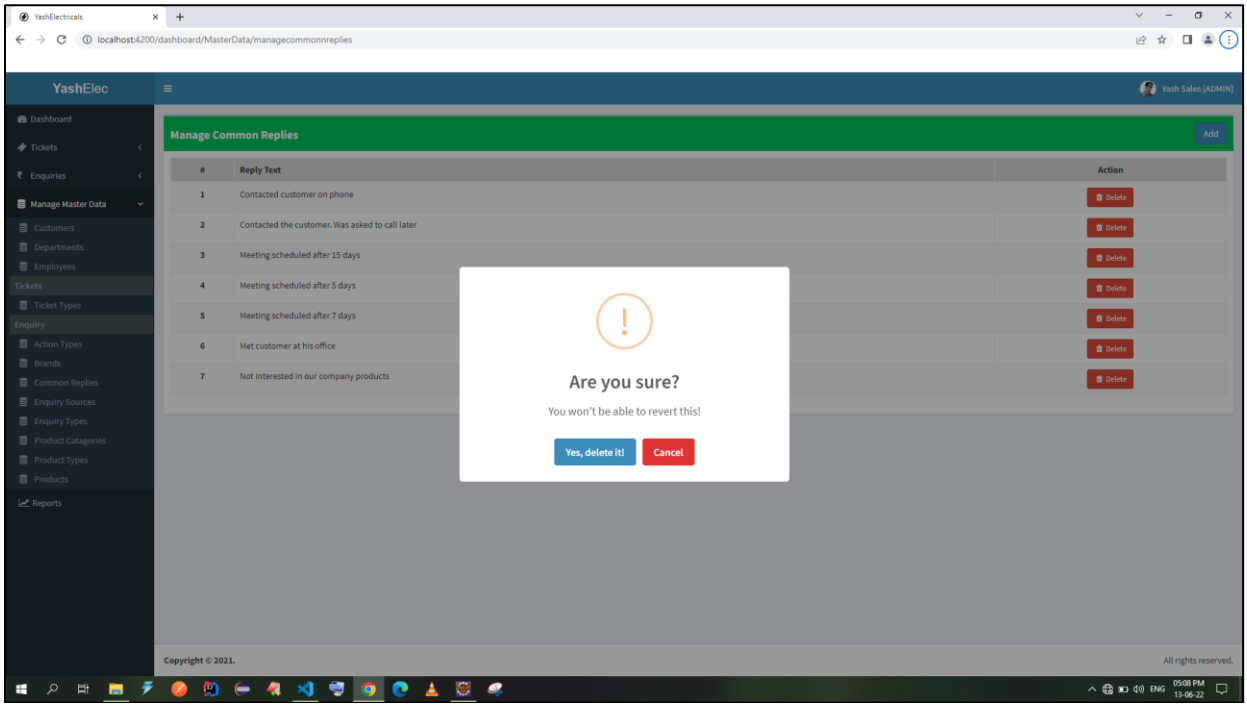
*Manage Common Replies: View Common Replies



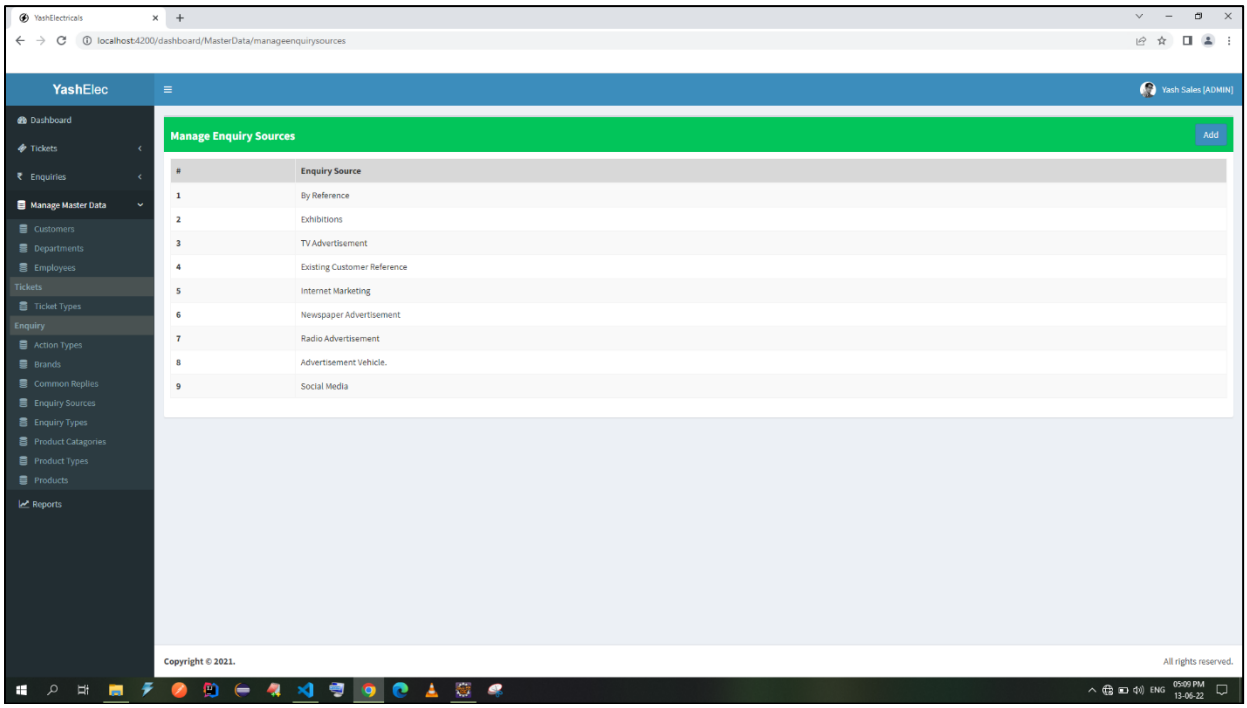
*Manage Common Replies: Add Common Replies



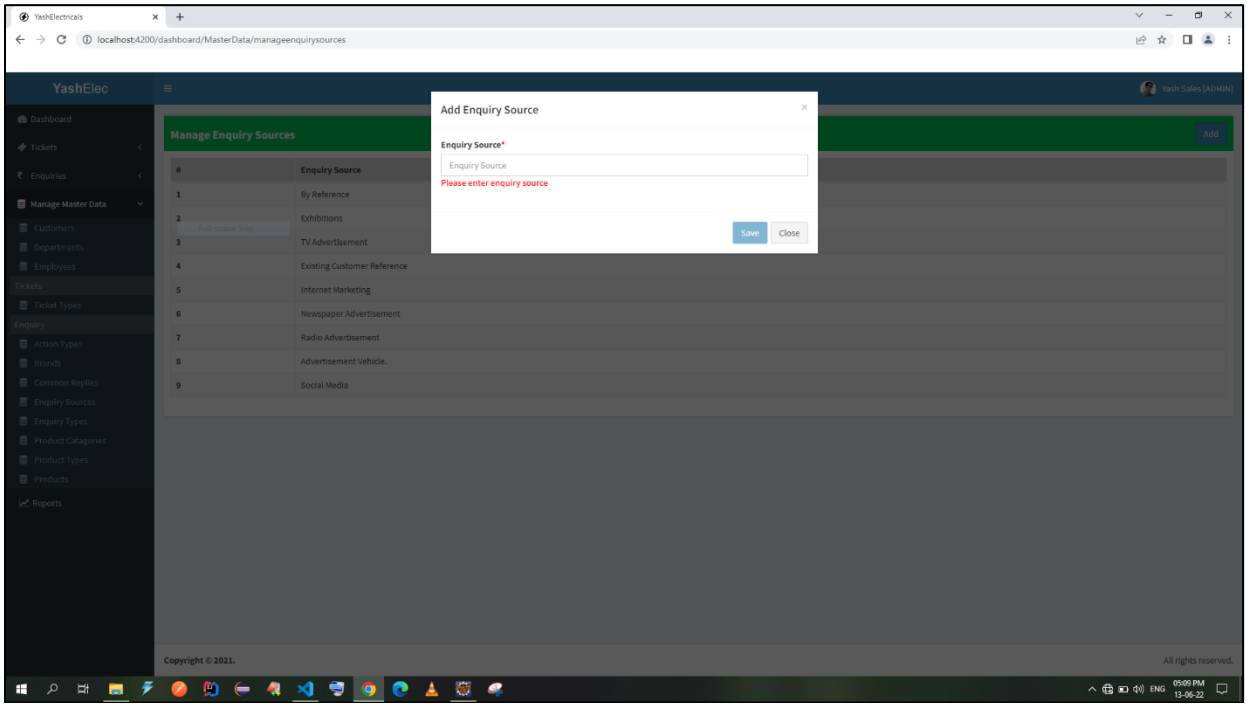
*Manage Common Replies: Delete Common Replies



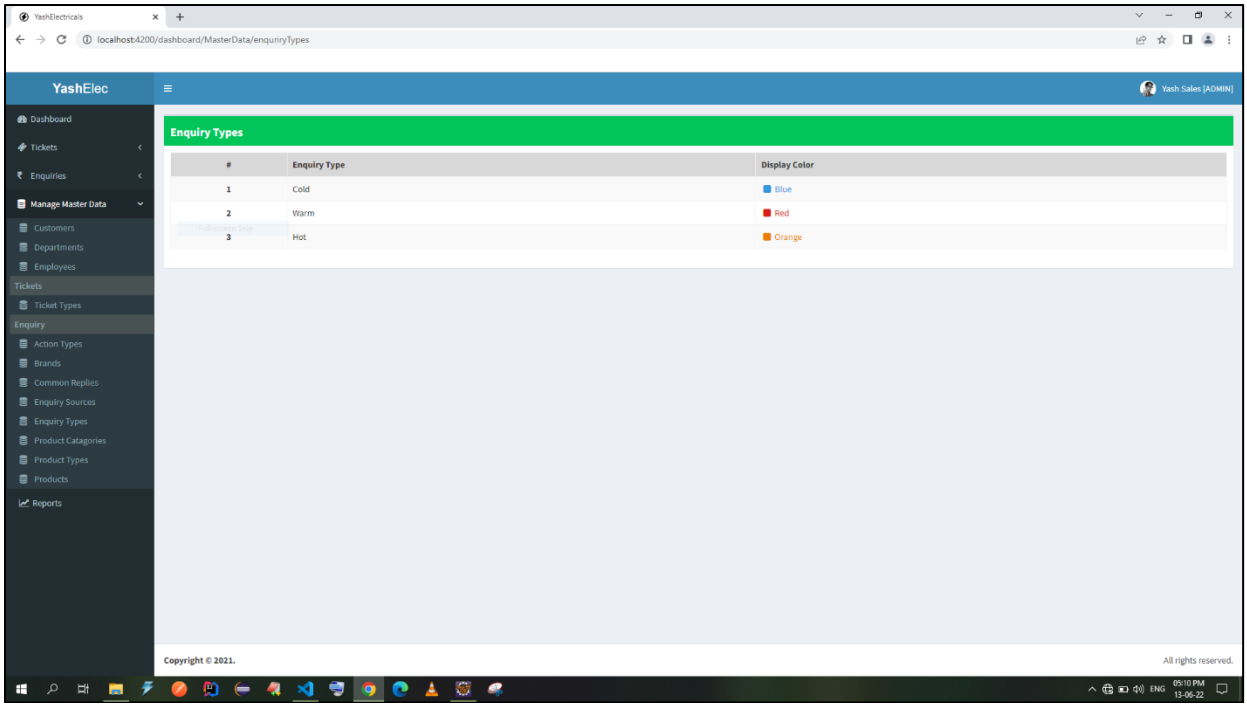
*Manage Enquiry Source: View Enquiry Source



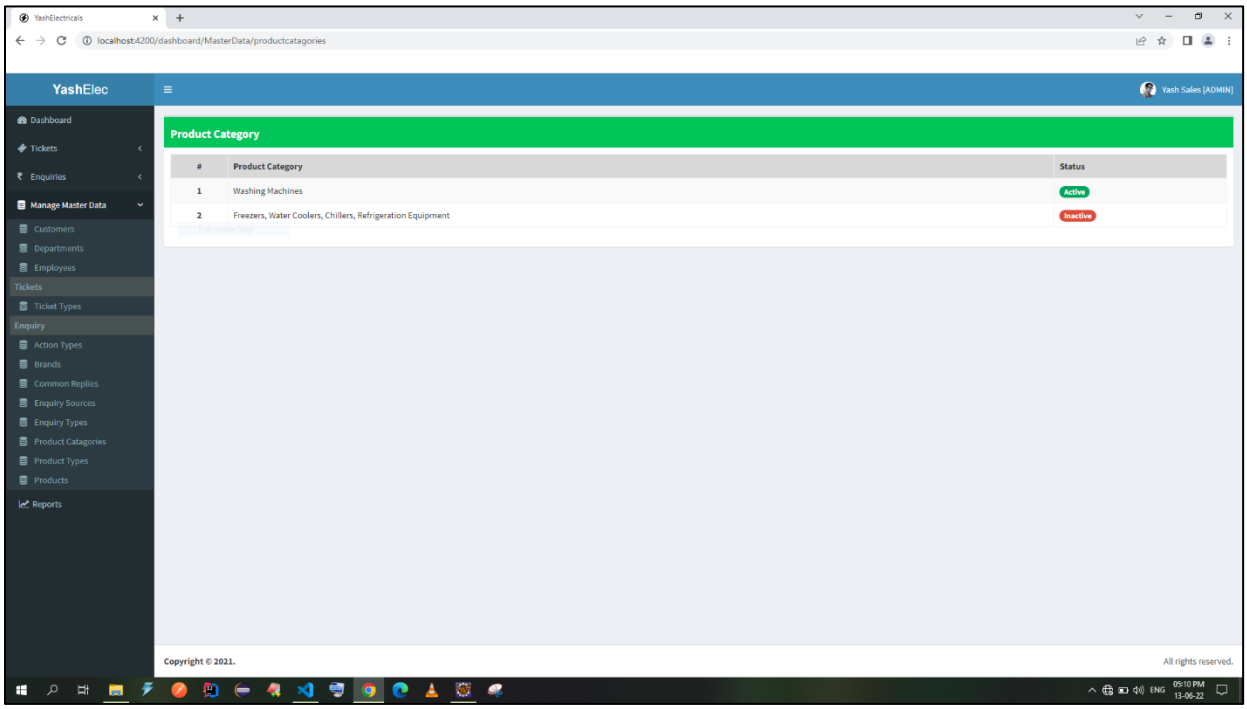
*Manage Enquiry Source: Add Enquiry Source



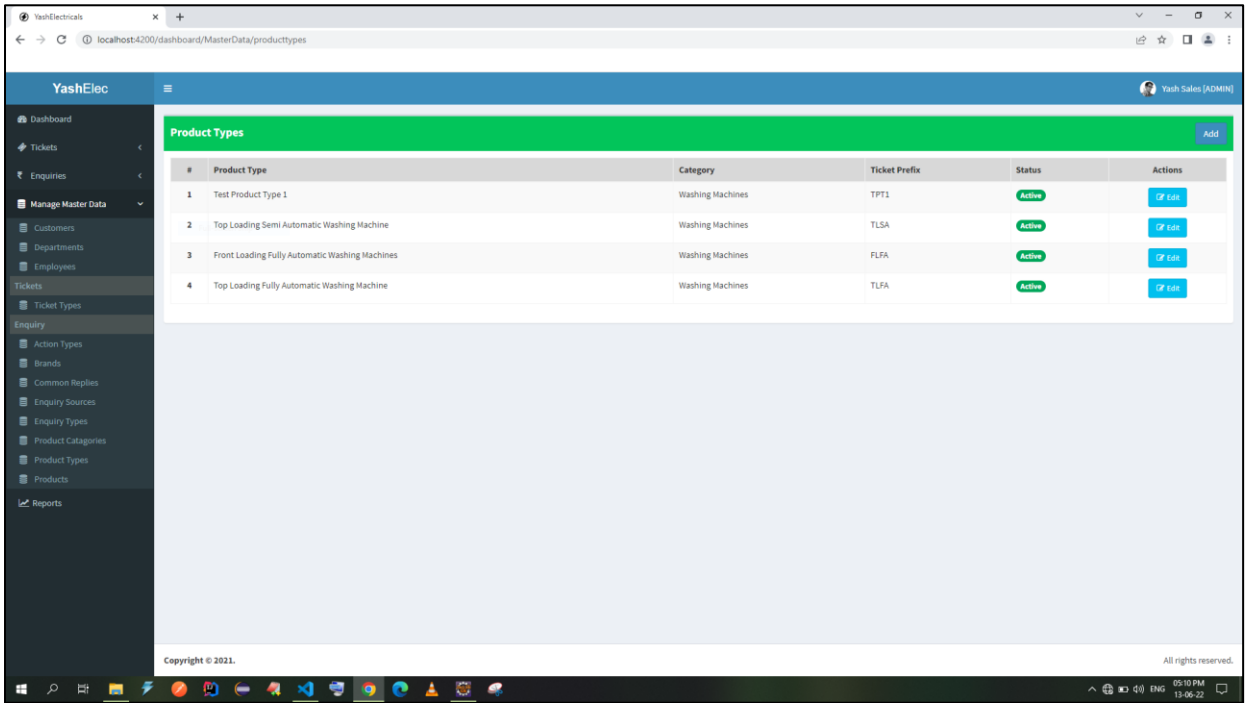
*Manage Enquiry Types: View Enquiry Types



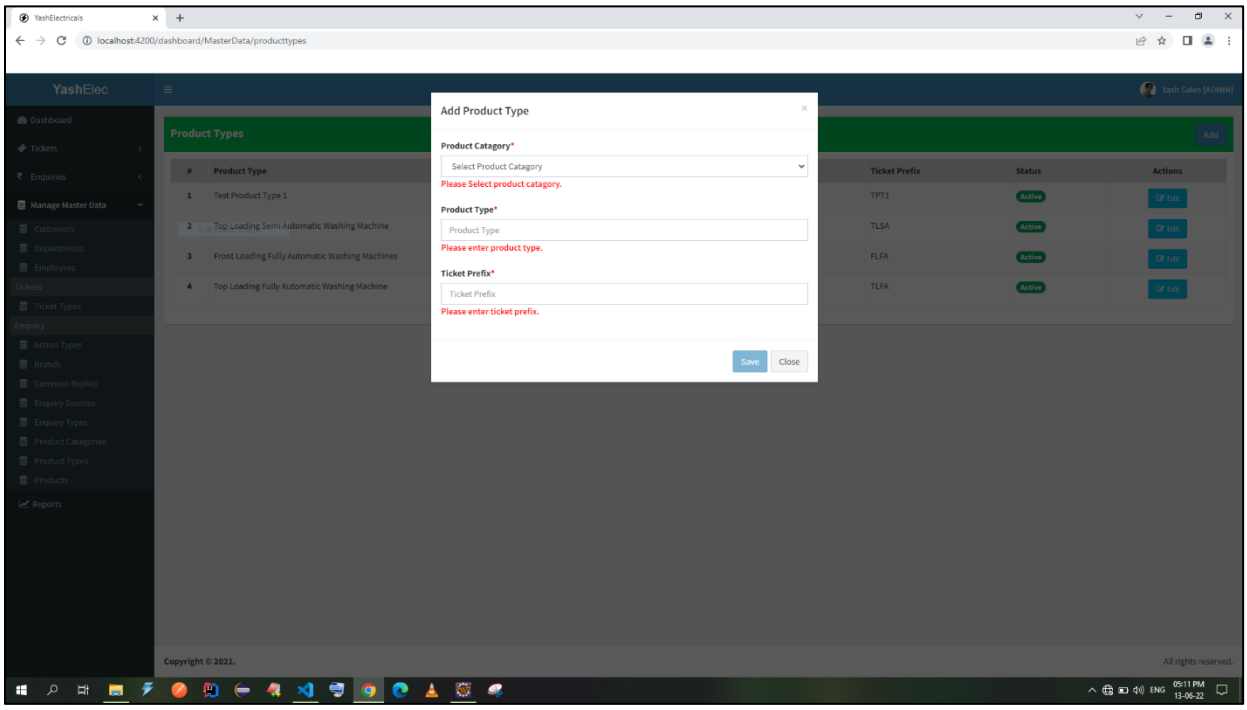
*Manage Product Category: View Product Category



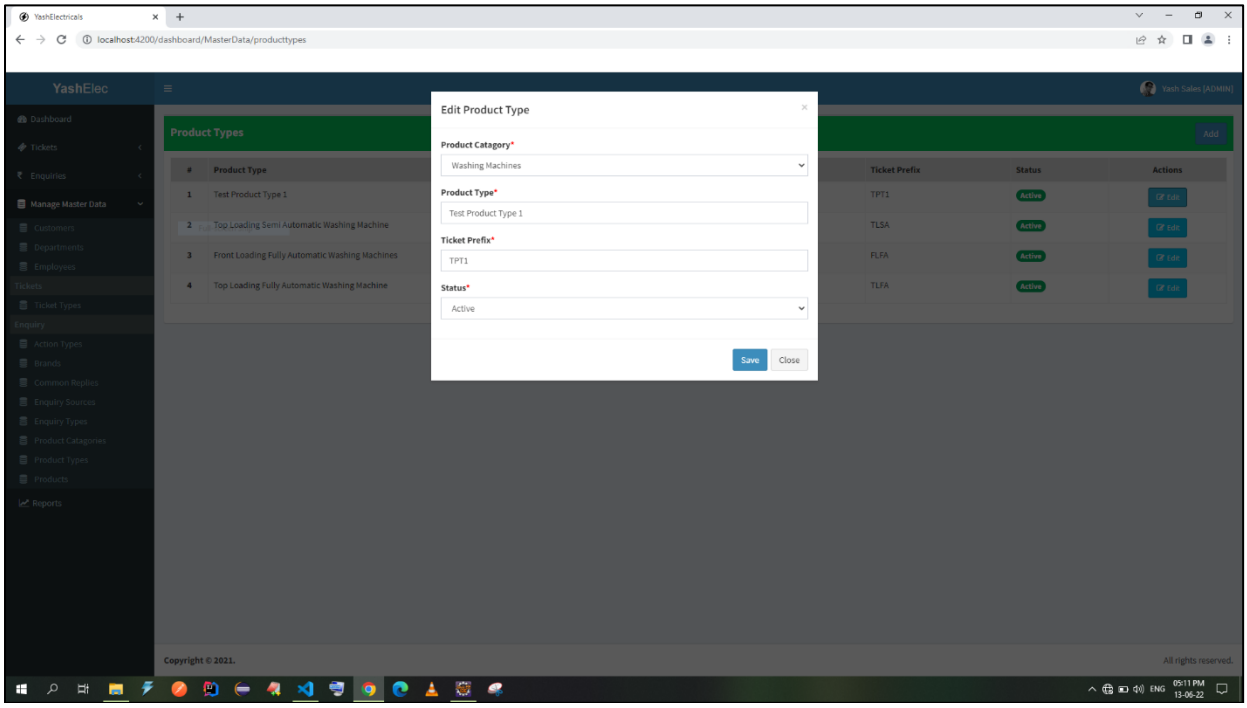
*Manage Product Types: View Product Types



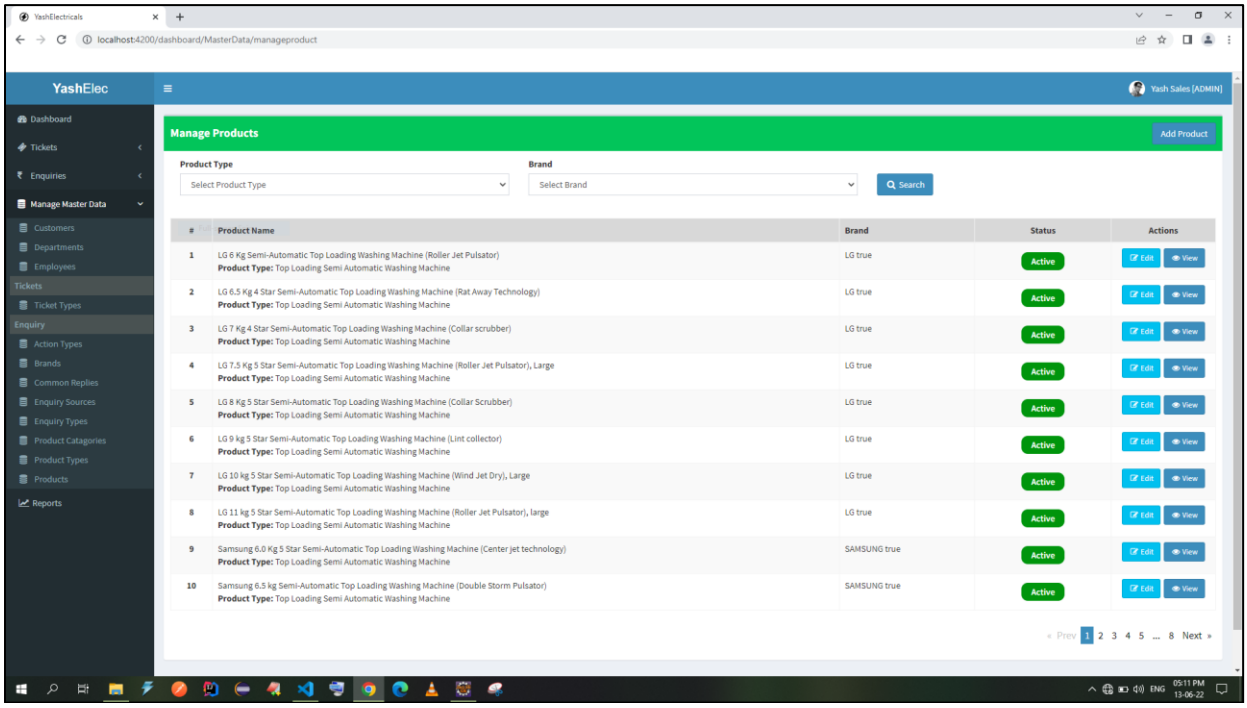
*Manage Product Types: Add Product Types



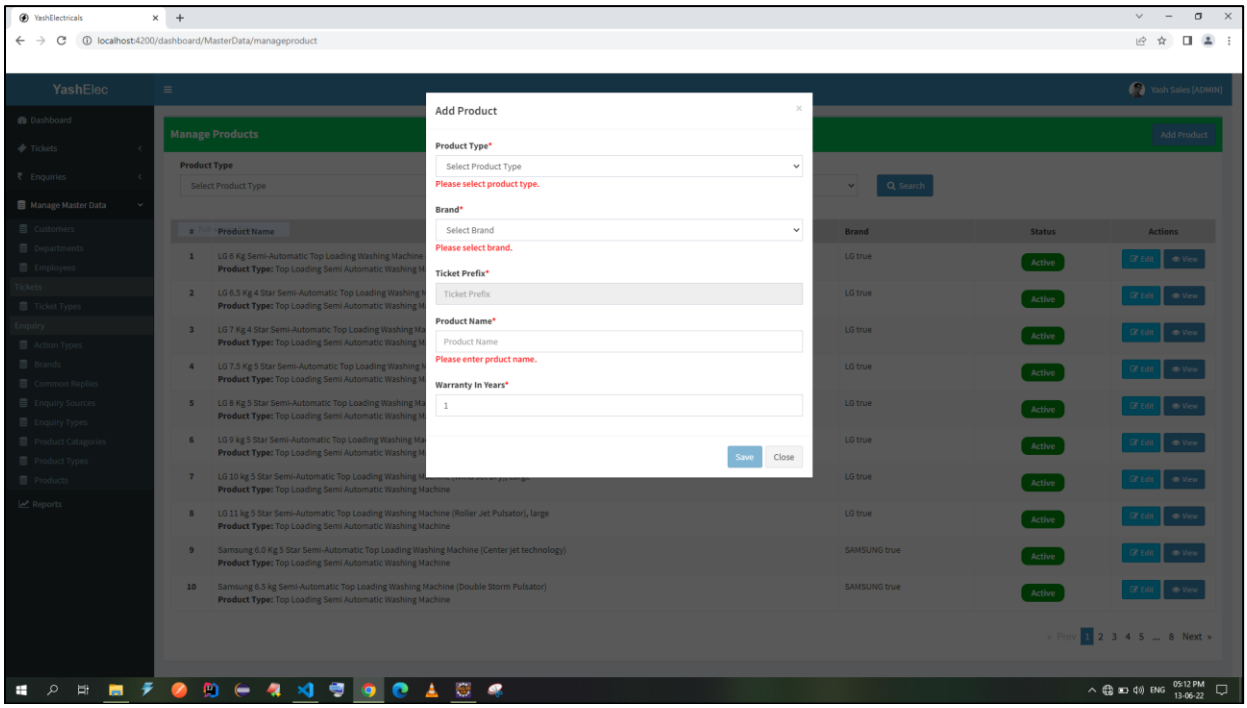
*Manage Product Types: Edit Product Types



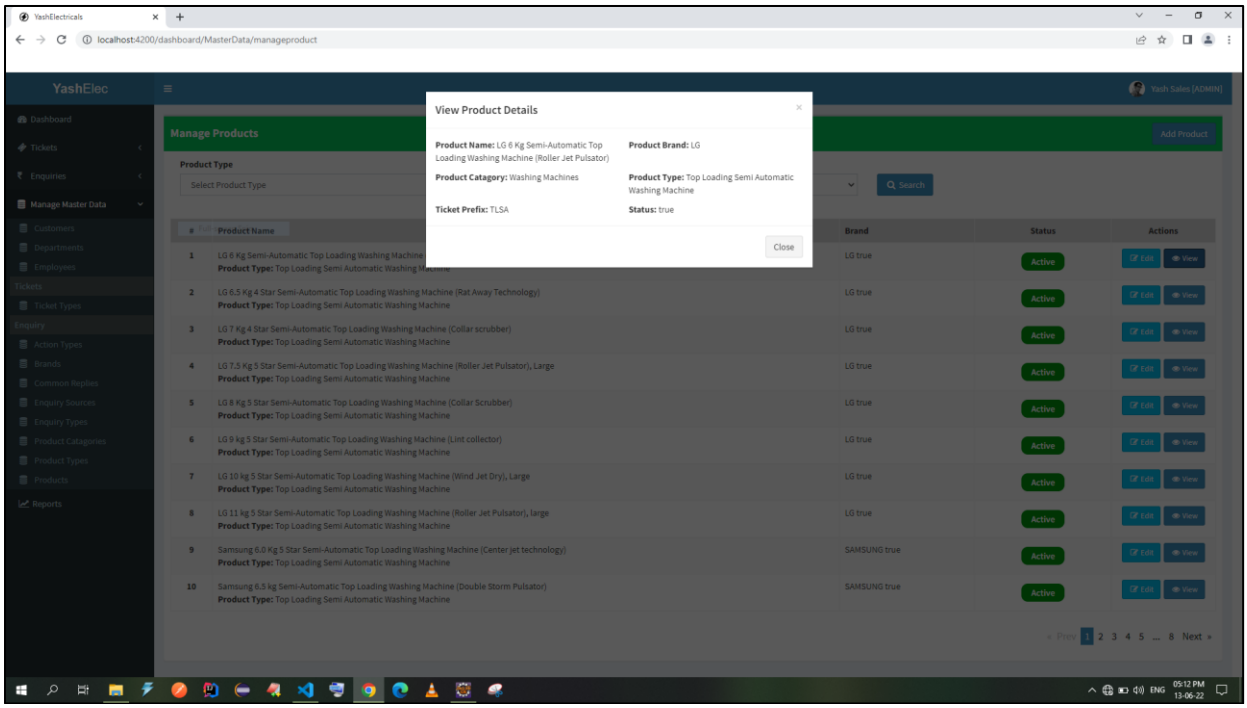
*Manage Products: View Products



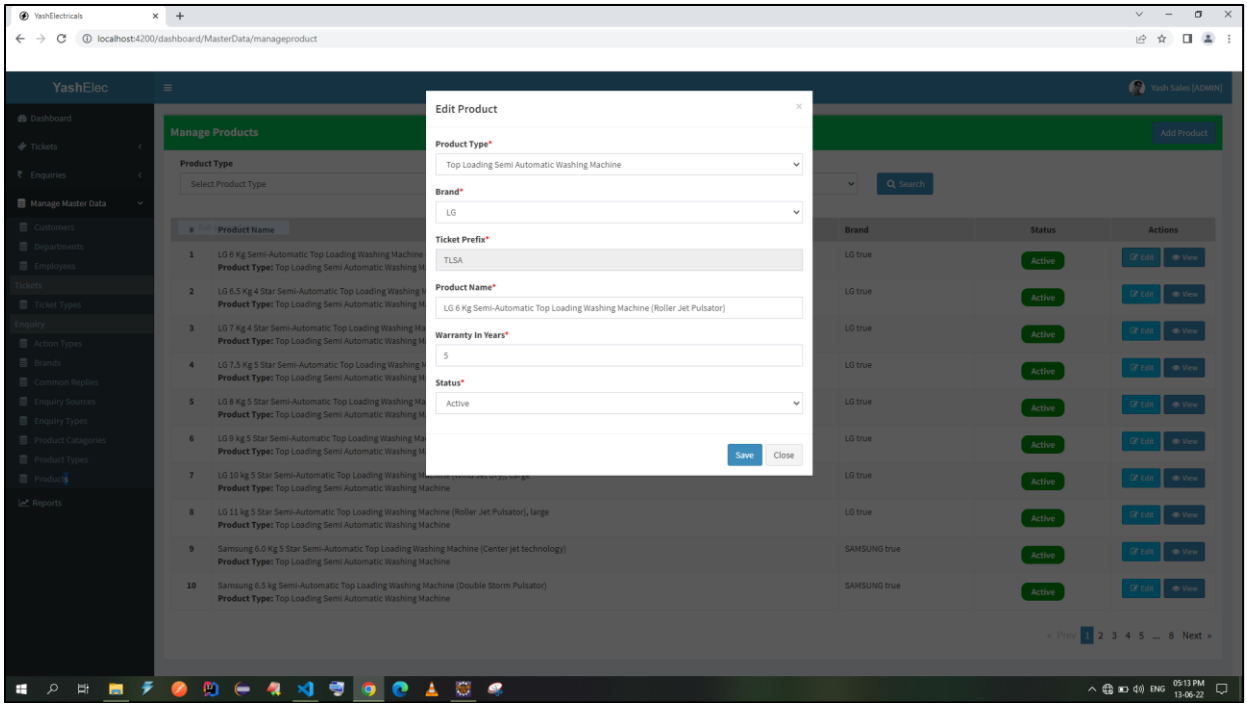
*Manage Product: Add Product



*Manage Product: View Product Details



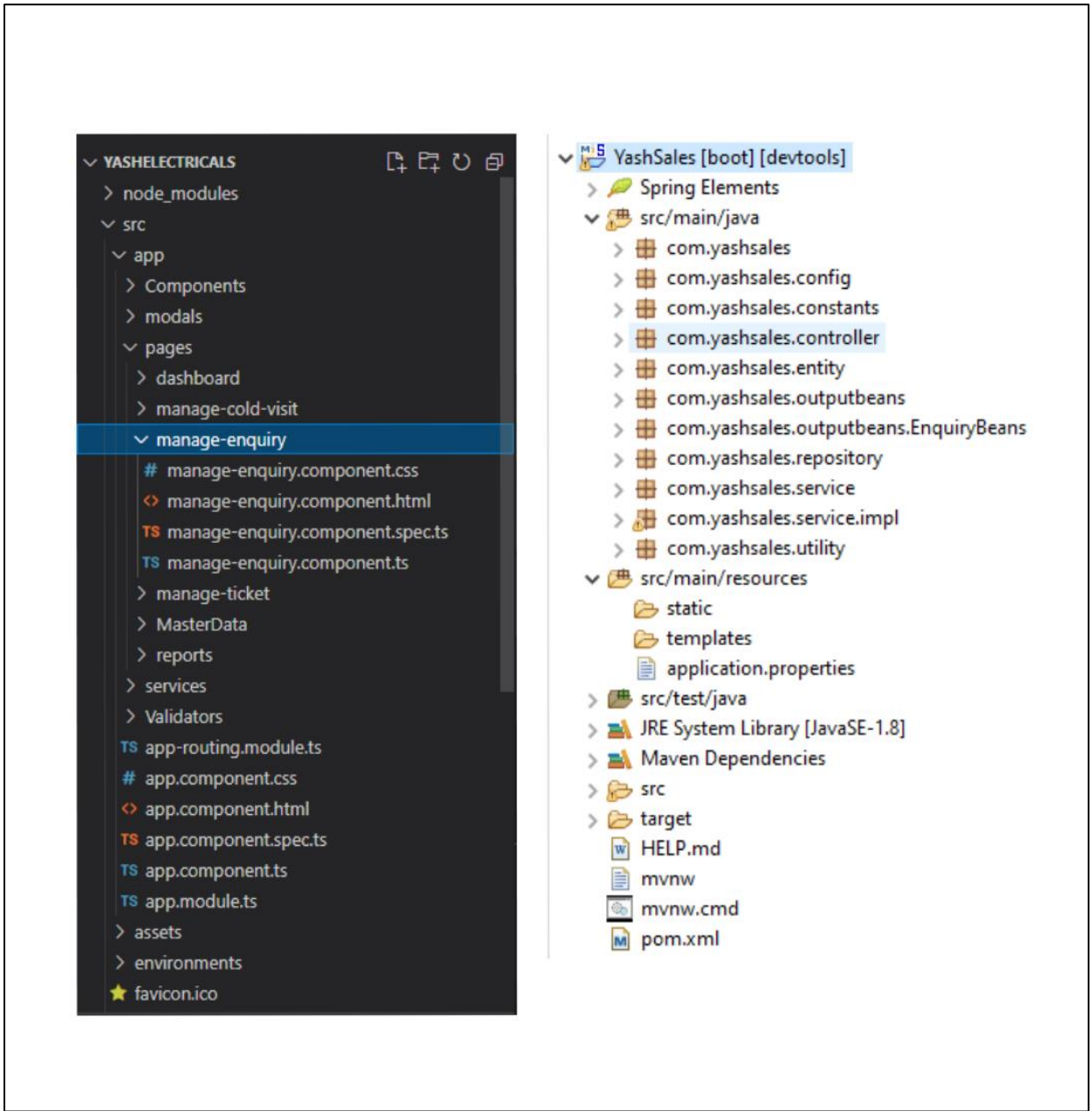
*Manage Product: Edit Product Details



CHAPTER 4
CODING

4.1) Code Snippets of Enquiry Functionality

Directory Structure of Project



Enquiry Angular Code

```
<section class="content">
  <div class="row">
    <div class="col-xs-12">
      <div class="box">

        <div class="box-header" style="background-color: #02c55a; color: white;">
          <div class="box-title">
            <div style="font-weight: bold; margin-top: 5%;">Manage Enquiries</div>
          </div>
          <div style="float: right;">
            <button type="button" class="btn btn-primary" (click)="openAddEnquiryPopup()">Add Enquiry</button>
          </div>
        </div>

        <div class="box-body">
          <div class="container-fluid">
            <form id="searchEnquiryForm" name="searchEnquiryForm">

              <div class="row">
                <div class="col-sm-4">
                  <div class="form-group">
                    <label>Enquiry Type</label>
                    <select id="enquiryTypeSearch" class="form-control"
[(ngModel)]="enquirySearchBean.enquiryTypeId" name="enquiryTypeId">
                      <option value="0" selected>Select Enquiry Type</option>
                      <option *ngFor="let enquiryType of enquiryTypeList"
[value]="enquiryType.enquiryTypeId">{{enquiryType.enquiryType}}</option>
                    </select>
                  </div>
                </div>

                <div class="col-sm-4">
                  <div class="form-group">
                    <label>Enquiry Source</label>
                    <select id="enquirySourceSearch" class="form-control"
[(ngModel)]="enquirySearchBean.enquirySourceId" name="enquirySourceId">
                      <option value="0" selected>Select Enquiry Source</option>
                      <option *ngFor="let es of enquirySourceList"
[value]="es.enquirySourceId">{{es.enquirySource}}</option>
                    </select>
                  </div>
                </div>

                <div class="col-sm-4">
                  <div class="form-group">
                    <label>Enquiry Status</label>
                    <select id="enquiryStatus" class="form-control" [(ngModel)]="enquirySearchBean.status"
name="status">
                      <option value="Unassigned" selected>Unassigned</option>
                      <option value="ALL" selected>All</option>
                      <option value="Assigned">Assigned</option>
                      <option value="Won">Won</option>
                      <option value="Prospect">Prospect</option>
                      <option value="Cancelled">Cancelled</option>
                      <option value="Lost">Lost</option>
                    </select>
                  </div>
```

```
            </div>


          </div>

            <div class="col-sm-4">
              <div style="margin-top: 25px;">
                <button type="button" class="btn btn-primary" (click)="searchEnquiryFunction();">
                  <i class="fa fa-fw fa-search"></i> Search
                </button>
              </div>
            </div>
          </div>

        </form>
      </div>
    </div>

    <div class="box-body">
      <div class="table-responsive">
        <table class="table table-bordered table-striped">
          <thead>
            <tr>
              <th scope="col" class="center">#</th>
              <th scope="col">Customer</th>
              <th scope="col">Product</th>
              <th scope="col">Type</th>
              <th scope="col">Enquiry Date</th>
              <th scope="col">Recent Activity Date</th>
              <th scope="col">Assigned To</th>
              <th scope="col">Status</th>
              <th scope="col" style="text-align: center;">Action</th>
            </tr>
          </thead>
          <tbody>
            <tr *ngIf="!enquiryList">
              <td colspan="9" style="color: red; text-align: center;"><span>No records found!</span></td>
            </tr>

            <tr *ngFor="let enquiry of enquiryList | paginate : {
              itemsPerPage: 10,
              currentPage: page,
              totalItems: totalItems
            };
            let indexOfElements = index;
            ">

              <th scope="row" class="center">{{(page-1) * 10 + indexOfElements + 1}}</th>
              <td>
                <div>{{enquiry.customer.customerName}}</div>
                <!-- <div><span class="bold">Product Type: </span>{{product.productType.productName}}</div> --
>
              </td>
              <td>{{enquiry.product.productName}}</td>
              <td>{{enquiry.enquiryType.enquiryType}}</td>
              <td>{{enquiry.addedDate | date:"dd MMM yyyy"}}</td>
              <td>{{enquiry.recentActivityDate | date:"dd MMM yyyy"}}</td>
              <td *ngIf="enquiry.assignedTo == null">-</td>
              <td *ngIf="enquiry.assignedTo != null">{{enquiry.assignedTo.fullName}}</td>
              <td>{{enquiry.status}}</td>
```

```
            <td style="text-align: center;" *ngIf="enquiry.status === 'Won' || enquiry.status === 'Lost' ||
enquiry.status === 'Cancelled' ">
              <button type="button" class="btn btn-sm btn-info" (click)="getEnquiryDetails(enquiry.enquiryId)"><i
class="fa fa-fw fa-eye"></i> View</button>
            </td>

            <td style="text-align: center;" *ngIf="enquiry.status === 'Unassigned' || enquiry.status === 'Assigned' ||
enquiry.status === 'Prospect'">
              <button type="button" class="btn btn-sm btn-info" (click)="getEnquiryDetails(enquiry.enquiryId)"><i
class="fa fa-fw fa-edit"></i> Edit</button>
            </td>
           </tr>
          </tbody>
         </table>

         <div class="paginationDiv">
          <pagination-controls class="pagination my-pagination" (pageChange)="changePage($event)"
           previousLabel="Prev" nextLabel="Next"></pagination-controls>
         </div>

        </div>
       </div>

     </div>
    </div>
  </div>
</section>

<!-- View Enquiry Details Modal Starts -->
<div class="modal fade" id="viewEnquiryDetailsModal" data-backdrop="static" data-keyboard="false">
 <div class="modal-dialog modal-lg">
  <div class="modal-content">
   <div class="modal-header">
    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
     <span aria-hidden="true">&times;</span></button>
    <h4 class="modal-title">Enquiry Details</h4>
   </div>

   <div class="modal-body">
    <div *ngIf="enquiryDetailsBean">

     <div class="row">
      <div class="col-sm-4">
       <span class="bold">Customer Name: </span>
       <span>{{enquiryDetailsBean.customer.customerName}}</span>
      </div>
      <div class="col-sm-4">
       <span class="bold">Enquiry Source: </span>
       <span>{{enquiryDetailsBean.enquirySource.enquirySource}}</span>
      </div>
      <div class="col-sm-4">
       <span class="bold">Enquiry Type: </span>
       <span>{{enquiryDetailsBean.enquiryType.enquiryType}}</span>
      </div>
     </div>

     <div class="mtb"></div>

     <div class="row">
      <div class="col-sm-4">
       <span class="bold">Product: </span>
```

```
            <span>{{enquiryDetailsBean.product.productName}}</span>
           </div>
           <div class="col-sm-4">
            <span class="bold">Product Type: </span>
            <span>{{enquiryDetailsBean.product.productType.productName}}</span>
           </div>
           <div class="col-sm-4">
            <span class="bold">Brand: </span>
            <span>{{enquiryDetailsBean.product.brand.brandName}}</span>
           </div>
          </div>

          <div class="mtb"></div>

          <div class="row">
           <div class="col-sm-4">
            <span class="bold">Product Quantity: </span>
            <span>{{enquiryDetailsBean.quantity}}</span>
           </div>
           <div class="col-sm-4">
            <span class="bold">Product Remark: </span>
            <span *ngIf="enquiryDetailsBean.productRemark != null && enquiryDetailsBean.productRemark !=
"">{{enquiryDetailsBean.productRemark}}</span>
            <span *ngIf="enquiryDetailsBean.productRemark == null && enquiryDetailsBean.productRemark ==
"">-</span>
           </div>
           <div class="col-sm-4">
            <span class="bold">Added By: </span>
            <span>{{enquiryDetailsBean.addedBy.fullName}}
<b>[{{enquiryDetailsBean.addedBy.role.rolename}}]</b></span>
           </div>
          </div>
          <div class="row">
           <div class="col-sm-4">
            <span class="bold">Assigned By: </span>
            <span *ngIf="enquiryDetailsBean.assignedBy != null && enquiryDetailsBean.assignedBy !=
"">{{enquiryDetailsBean.assignedBy.fullName}}</span>
            <span *ngIf="enquiryDetailsBean.assignedBy == null && enquiryDetailsBean.assignedBy == "">-
</span>
           </div>
           <div class="col-sm-4">
            <span class="bold">Assigned To: </span>
            <span *ngIf="enquiryDetailsBean.assignedTo != null && enquiryDetailsBean.assignedTo !=
"">{{enquiryDetailsBean.assignedTo.fullName}}</span>
            <span *ngIf="enquiryDetailsBean.assignedTo == null && enquiryDetailsBean.assignedTo == "">-
</span>
           </div>
           <div class="col-sm-4">
            <span class="bold">Remark: </span>
            <span>{{enquiryDetailsBean.remark}}</span>
           </div>
          </div>

          <div class="mtb"></div>

          <div class="row" style="float:right" *ngIf="enquiryDetailsBean.status != 'Won' &&
enquiryDetailsBean.status != 'Lost' && enquiryDetailsBean.status != 'Cancelled' ">
           <div class="col-sm-12">

            <div class="btn-group">
             <button type="button" class="btn btn-primary btn-flat">Actions</button>
```

```
            <button type="button" class="btn btn-primary btn-flat dropdown-toggle" data-toggle="dropdown" aria-
expanded="false">
              <span class="caret"></span>
              <span class="sr-only">Toggle Dropdown</span>
            </button>
            <ul class="dropdown-menu" role="menu" *ngIf="enquiryDetailsBean.status === 'Unassigned'">
              <li><a class="action" (click)="openAssignEnquiryModal(enquiryDetailsBean.enquiryId);">Assign
Enquriry</a></li>
              <li><a class="action" (click)="openCancelEnquiryModal(enquiryDetailsBean.enquiryId);">Cancel
Enquriry</a></li>
            </ul>
            <ul class="dropdown-menu" role="menu" *ngIf="enquiryDetailsBean.status === 'Assigned'">
              <li><a class="action"
(click)="openChangeToProspectEnquiryModal(enquiryDetailsBean.enquiryId);">Change to Prospet</a></li>
              <li><a class="action" (click)="openCancelEnquiryModal(enquiryDetailsBean.enquiryId);">Cancel
Enquriry</a></li>
            </ul>
            <ul class="dropdown-menu" role="menu" *ngIf="enquiryDetailsBean.status === 'Prospect'">
              <li><a class="action" (click)="openWonEnquiryModal(enquiryDetailsBean.enquiryId);">Won
Enquiry</a></li>
              <li><a class="action" (click)="openLostEnquiryModal(enquiryDetailsBean.enquiryId);">Lost
Enquriry</a></li>
            </ul>
          </div>

        </div>

      </div>

    </div>

    <div class="mtb"></div>

    <section class="content">
      <div class="row">
        <div class="col-md-12">
          <ul class="timeline">

            <ul class="timeline" *ngFor="let ea of enquiryActivityList; let i = index">

              <li class="time-label" *ngIf="ea.status === 'Assigned'"><span class="bg-assigned">{{ea.recordDate |
date:"dd MMM yyyy"}}</span></li>
              <li class="time-label" *ngIf="ea.status === 'Unassigned'"><span class="bg-
unassigned">{{ea.recordDate | date:"dd MMM yyyy"}}</span></li>
              <li class="time-label" *ngIf="ea.status === 'Cancelled'"><span class="bg-cancelled">{{ea.recordDate |
date:"dd MMM yyyy"}}</span></li>
              <li class="time-label" *ngIf="ea.status === 'Prospect'"><span class="bg-prospect">{{ea.recordDate |
date:"dd MMM yyyy"}}</span></li>
              <li class="time-label" *ngIf="ea.status === 'Won'"><span class="bg-won">{{ea.recordDate | date:"dd
MMM yyyy"}}</span></li>
              <li class="time-label" *ngIf="ea.status === 'Lost'"><span class="bg-lost">{{ea.recordDate | date:"dd
MMM yyyy"}}</span></li>

              <li *ngIf="ea.status === 'Unassigned'">
                <i class="fa fa-fw fa-circle bg-unassigned"></i>
                <div class="timeline-item">
                  <h3 class="timeline-header no-border"><a>{{ea.status}}</a> {{ea.remark}}</h3>
                </div>
              </li>

              <li *ngIf="ea.status === 'Assigned'">
```

```
                <i class="fa fa-fw fa-check-circle bg-assigned"></i>
                <div class="timeline-item">
                  <h3 class="timeline-header no-border"><a>{{ea.status}}</a> {{ea.remark}}</h3>
                </div>
              </li>

              <li *ngIf="ea.status === 'Cancelled'">
                <i class="fa fa-fw fa-times-circle bg-cancelled"></i>
                <div class="timeline-item">
                  <h3 class="timeline-header no-border"><a>{{ea.status}}</a> {{ea.remark}}</h3>
                </div>
              </li>

              <li *ngIf="ea.status === 'Prospect'">
                <i class="fa fa-fw fa-hand-o-right bg-prospect"></i>
                <div class="timeline-item">
                  <h3 class="timeline-header no-border"><a>{{ea.status}}</a> {{ea.remark}}</h3>
                </div>
              </li>

              <li *ngIf="ea.status === 'Won'">
                <i class="fa fa-fw fa-thumbs-o-up bg-won"></i>
                <div class="timeline-item">
                  <h3 class="timeline-header no-border"><a>{{ea.status}}</a> {{ea.remark}}</h3>
                </div>
              </li>

              <li *ngIf="ea.status === 'Lost'">
                <i class="fa fa-fw fa-thumbs-o-down bg-lost"></i>
                <div class="timeline-item">
                  <h3 class="timeline-header no-border"><a>{{ea.status}}</a> {{ea.remark}}</h3>
                </div>
              </li>


            </ul>

          </ul>

        </div>

      </div>
    </section>
  </div>

  <div class="modal-footer">
    <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
  </div>
 </div>
</div>
</div>
<!-- View Enquiry Details Modal Ends -->
```

Java Code for Backend:

```
package com.yashsales.controller;

import com.yashsales.outputbeans.AddEnquiryBean;
import com.yashsales.outputbeans.EnquiryBeans.AssignEnquiryBean;
import com.yashsales.outputbeans.EnquiryBeans.CancelEnquiryBean;
import com.yashsales.outputbeans.EnquiryBeans.ConvertToProspectBean;
import com.yashsales.outputbeans.EnquiryBeans.WonLostEnquiryBean;
import com.yashsales.outputbeans.EnquirySearchBean;
import com.yashsales.service.EnquiryService;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController @RequestMapping("/enquiry") @CrossOrigin("*") public class EnquiryController
{

 @Autowired private EnquiryService enqService;

 @PostMapping("/")
 public Map<String, Object> addEnquiry(@RequestBody AddEnquiryBean enquiryBean)
 {
  return enqService.addEnquiry(enquiryBean);
 }

 @PostMapping("/get")
 public Map<String, Object> getEnquiries(@RequestBody EnquirySearchBean searchBean)
 {
  return enqService.getPaginatedEnquiries(searchBean);
 }

 @GetMapping("/getassignedtolist/") public Map<String, Object> getAssignedToList()
 {
  return enqService.getAssignedToList();
 }

 @GetMapping("/getenquirydetails/{enquiryId}")
 public Map<String, Object> getEnquiryDetails(@PathVariable Long enquiryId)
 {
  return enqService.getEnquiryDetails(enquiryId);
 }

 @PostMapping("/assign")
 public Map<String, Object> assignEnquiry(@RequestBody AssignEnquiryBean assignEnquiry)
 {
  return enqService.assignEnquiry(assignEnquiry);
 }

 @PostMapping("/cancel")
 public Map<String, Object> cancelEnquiry(@RequestBody CancelEnquiryBean cancelEnquiry)
 {
  return enqService.cancelEnquiry(cancelEnquiry);
```

```
 }

 @PostMapping("/convertToProspect")
 public Map<String, Object> convertToProspectEnquiry(
  @RequestBody ConvertToProspectBean convertToProspect)
 {
  return enqService.convertToProspectEnquiry(convertToProspect);
 }

 @PostMapping("/won")
 public Map<String, Object> wonEnquiry(@RequestBody WonLostEnquiryBean wonEnquiry)
 {
  return enqService.wonEnquiry(wonEnquiry);
 }

 @PostMapping("/lost")
 public Map<String, Object> lostEnquiry(@RequestBody WonLostEnquiryBean lostEnquiry)
 {
  return enqService.lostEnquiry(lostEnquiry);
 }
}
```

Service Class

```
@Service public class EnquiryServiceImpl implements EnquiryService
{

 @Override @Transactional public Map<String, Object> addEnquiry(AddEnquiryBean enquiryBean)
 {
  Map<String, Object> returnMap = new HashMap<String, Object>();
  returnMap.put("responseStatus",
            ApplicationConstants.ResponseConstants.RESPONSE_FAILURE);

  Customer customer = null;
  Enquiry enquiry = null;
  EnquirySource enquirySource = null;
  EnquiryType enquiryType = null;
  Product product = null;
  User addedBy = null;

  if (enquiryBean.getCustomerId() > 0) {
   customer = custRepo.findById(enquiryBean.getCustomerId()).orElse(null);
  }

  if (customer != null && customer.getCustomerId() > 0) {
   if (enquiryBean.getProductList() != null && enquiryBean.getProductList().size() > 0) {
    for (int i = 0; i < enquiryBean.getProductList().size(); i++) {

     Long productId = enquiryBean.getProductList().get(i).getProductId();
     if (productId > 0) {

      enquiry = new Enquiry();
      // Set Customer
      enquiry.setCustomer(customer);

      // set Enquiry Source
      if (enquiryBean.getEnquirySourceId() > 0) {
       enquirySource =
        enqSourceRepo.findById(enquiryBean.getEnquirySourceId()).orElse(null);
       if (enquirySource != null) {
        enquiry.setEnquirySource(enquirySource);
       }
```

```
}

if (enquiryBean.getEnquiryTypeId() > 0) {
  enquiryType =
    enqTypeRepo.findById(enquiryBean.getEnquiryTypeId()).orElse(null);
  if (enquiryType != null) {
    enquiry.setEnquiryType(enquiryType);
  }
}

product = null;
product = productRepo.findById(productId).orElse(null);
if (product != null) {
  enquiry.setProduct(product);
  enquiry.setQuantity(enquiryBean.getProductList().get(i).getQuantity());
  enquiry.setProductRemark(
    enquiryBean.getProductList().get(i).getProductRemark());
}

// addedBy
addedBy = userService.getCurrentLoggedInUser();
// addedBy = userRepo.findById(enquiryBean.getAddedById()).orElse(null); //TODO
// Old code.
enquiry.setAddedBy(addedBy);

// addedDate
enquiry.setAddedDate(DateUtils.getCurrentTimestamp());

// status
enquiry.setStatus(
  ApplicationConstants.EnquiryConstants.ENQUIRY_STATUS_UNASSIGNED);

if (enquiryBean.isSelfAssigned()) {
  enquiry.setAssignedTo(addedBy);
  enquiry.setAssignedBy(addedBy);
  enquiry.setStatus(
    ApplicationConstants.EnquiryConstants.ENQUIRY_STATUS_ASSIGNED);
} else if (enquiryBean.getAssignedToId() != null &&
        enquiryBean.getAssignedToId() > 0) {
  User assignedTo =
    userRepo.findById(enquiryBean.getAssignedToId()).orElse(null);
  enquiry.setAssignedTo(assignedTo);
  enquiry.setAssignedBy(addedBy);
  enquiry.setStatus(
    ApplicationConstants.EnquiryConstants.ENQUIRY_STATUS_ASSIGNED);
}

// recent activity date
// enquiry.setRecentActivityDate(DateUtils.getCurrentTimestamp());
enquiry.setRecentActivityDate(new Timestamp(System.currentTimeMillis()));

// remark
enquiry.setRemark(enquiryBean.getRemark());

// createdBy
enquiry.setCreatedBy(addedBy.getUserId());

// createdDate
enquiry.setCreatedDate(DateUtils.getCurrentTimestamp());

// save should be here
```

```
        enquiry = enqRepo.save(enquiry);

        setEnquiryActivities(enquiry);
       } // NewProduct

     } // for
     returnMap.put("responseStatus",
               ApplicationConstants.ResponseConstants.RESPONSE_SUCCESS);
     returnMap.put("message", "Enquriy added Successfully.");
    } else {
     returnMap.put("responseStatus",
               ApplicationConstants.ResponseConstants.RESPONSE_FAILURE);
     returnMap.put("message", "At least 1 product should be added for enquiry!");
     returnMap.put("errorCode", "WSEM008");
    }
   }
  return returnMap;
}

@Override public Map<String, Object> getPaginatedEnquiries(EnquirySearchBean searchBean)
{
  Map<String, Object> responseMap = new HashMap<String, Object>();
  Page<Enquiry> enquiryPage = null;
  int pageNumber = Math.subtractExact(searchBean.getPageNumber(), 1);
  Pageable pagable = PageRequest.of(pageNumber, searchBean.getPageSize());

  // Temp Code
  // User user = userRepo.findById(searchBean.getSysUserId()).orElse(null); //TODO change
  // after Rolewise
  User user = userService.getCurrentLoggedInUser();

  enquiryPage = enqRepo.findAll(new Specification<Enquiry>() {
   @Override
   public Predicate toPredicate(
    Root<Enquiry> root, CriteriaQuery<?> cq, CriteriaBuilder cb)
   {
    Predicate p = cb.conjunction();

    if (searchBean.getEnquirySourceId() > 0) {
     p = cb.and(p,
               cb.equal(root.join("enquirySource").get("enquirySourceId"),
                     searchBean.getEnquirySourceId()));
    }

    if (searchBean.getEnquiryTypeId() > 0) {
     p = cb.and(p,
               cb.equal(root.join("enquiryType").get("enquiryTypeId"),
                     searchBean.getEnquiryTypeId()));
    }

    if (searchBean.getStatus() != null && !searchBean.getStatus().equals("ALL")) {
     p = cb.and(p, cb.equal(root.get("status"), searchBean.getStatus()));
    }

    if (searchBean.getCustomerId() > 0) {
     p = cb.and(
       p,
       cb.equal(root.join("customer").get("customerId"), searchBean.getCustomerId()));
    }
```

```
    if (searchBean.getAddedBy() > 0) {
      p = cb.and(p,
              cb.equal(root.join("addedBy").get("userId"), searchBean.getAddedBy()));
    }

    if (user.getRole().getrolename().equals(
         ApplicationConstants.RoleConstants.ROLE_ADMIN)) {
      // all Enquiries should be visible
    } else if (user.getRole().getrolename().equals(
              ApplicationConstants.RoleConstants.ROLE_SALES_MANAGER)) {

      // all Enquiries should be visible that to for same Product Catagory
      UserProductCatagoryLink upcl =
        userProductCatagoryLinkRepo.findByUserId(user.getUserId());

      if (upcl.getUserProductCatagoryLinkId() > 0) {
        List<ProductType> productTypeList = productTypeRepo.findAllByProductCatagory(
          upcl.getProductCatagory().getProductCatagoryId());

        for (int i = 0; i < productTypeList.size(); i++) {
          p = cb.and(
            p,
            cb.equal(
              root.join("product")
                .join("productType")
                .join("productCatagory")
                .get("productCatagoryId"),
              productTypeList.get(i).getProductCatagory().getProductCatagoryId()));
        }
      }
    } else if (user.getRole().getrolename().equals(
              ApplicationConstants.RoleConstants.ROLE_SALES_ENGINEER)) {
      p = cb.and(p, cb.equal(root.join("assignedTo").get("userId"), user.getUserId()));
    }

    cq.where(p).orderBy(cb.desc(root.get("recentActivityDate")));

    return p;
  }
 }, pagable);

 responseMap.put("responseStatus",
             ApplicationConstants.ResponseConstants.RESPONSE_SUCCESS);
 responseMap.put("EnquiryList", enquiryPage.getContent());
 responseMap.put("CurrentPage", enquiryPage.getNumber() + 1);
 responseMap.put("TotalItems", enquiryPage.getTotalElements());
 responseMap.put("TotalPages", enquiryPage.getTotalPages());

 return responseMap;
}

@Transactional
@Override
public Map<String, Object> assignEnquiry(AssignEnquiryBean assignEnquiry)
{
 Map<String, Object> responseMap = new HashMap<String, Object>();
 responseMap.put("responseStatus",
             ApplicationConstants.ResponseConstants.RESPONSE_FAILURE);

 Enquiry enquiry = enqRepo.findById(assignEnquiry.getEnquiryId()).orElse(null);
 User assignedTo = userRepo.findById(assignEnquiry.getAssignToId()).orElse(null);
```

```
  // User assignedBy = userRepo.findById(assignEnquiry.getAssignById()).orElse(null); TODO
  // Old Code
  User assignedBy = userService.getCurrentLoggedInUser();

  // User sysUser = userRepo.findById(assignEnquiry.getSysUserId()).orElse(null); TODO Old
  // Code.

  if (enquiry != null && enquiry.getEnquiryId() > 0 &&
      enquiry.getStatus().equals(
        ApplicationConstants.EnquiryConstants.ENQUIRY_STATUS_UNASSIGNED) &&
      assignedTo != null && assignedTo.getUserId() > 0 && assignedBy != null &&
      assignedBy.getUserId() > 0) {

    enquiry.setAssignedTo(assignedTo);
    enquiry.setAssignedBy(assignedBy);
    enquiry.setRecentActivityDate(DateUtils.getCurrentTimestamp());
    enquiry.setStatus(ApplicationConstants.EnquiryConstants.ENQUIRY_STATUS_ASSIGNED);

    enquiry = enqRepo.save(enquiry);

    if (enquiry != null) {
      EnquiryActivity enqActivity = new EnquiryActivity();
      enqActivity.setEnquiry(enquiry);
      // enqActivity.setUser(sysUser);
      enqActivity.setUser(assignedBy);
      enqActivity.setStatus(enquiry.getStatus());
      enqActivity.setRecordDate(DateUtils.getCurrentTimestamp());
      enqActivity.setRemark("Enquiry Assigned to " + assignedTo.getFullName() + ", by " +
                    assignedBy.getFullName() + " with remark " +
                    assignEnquiry.getRemark());
      enqActivity = enqActivityRepo.save(enqActivity);
      if (enqActivity.getEnquiryActivityId() > 0) {
        responseMap.put("responseStatus",
                  ApplicationConstants.ResponseConstants.RESPONSE_SUCCESS);
        responseMap.put("message", "Enquiry Assigned Successfully");
      }
      responseMap.put("enquiryId", enquiry.getEnquiryId());
    }
  } else {
    responseMap.put("responseStatus",
              ApplicationConstants.ResponseConstants.RESPONSE_FAILURE);
    responseMap.put("message", "Unable to Assign Enquiry! Try Again");
  }
  return responseMap;
}

@Transactional
@Override
public Map<String, Object> cancelEnquiry(CancelEnquiryBean cancelEnquiry)
{
  Map<String, Object> responseMap = new HashMap<String, Object>();
  responseMap.put("responseStatus",
              ApplicationConstants.ResponseConstants.RESPONSE_FAILURE);

  Enquiry enquiry = enqRepo.findById(cancelEnquiry.getEnquiryId()).orElse(null);
  // User sysUser = userRepo.findById(cancelEnquiry.getSysUserId()).orElse(null); //Old
  // Code
  User currentUser = userService.getCurrentLoggedInUser();

  if (enquiry != null && enquiry.getEnquiryId() > 0 &&
```

```
      (enquiry.getStatus().equals(
         ApplicationConstants.EnquiryConstants.ENQUIRY_STATUS_UNASSIGNED) ||
       enquiry.getStatus().equals(
         ApplicationConstants.EnquiryConstants.ENQUIRY_STATUS_ASSIGNED)) &&
      currentUser.getUserId() > 0) {

    enquiry.setRecentActivityDate(DateUtils.getCurrentTimestamp());
    enquiry.setStatus(ApplicationConstants.EnquiryConstants.ENQUIRY_STATUS_CANCELLED);

    enquiry = enqRepo.save(enquiry);

    if (enquiry != null) {
     EnquiryActivity enqActivity = new EnquiryActivity();
     enqActivity.setEnquiry(enquiry);
     // enqActivity.setUser(sysUser);
     enqActivity.setUser(currentUser);
     enqActivity.setStatus(enquiry.getStatus());
     enqActivity.setRecordDate(DateUtils.getCurrentTimestamp());
     enqActivity.setRemark("Enquiry cancelled by " + currentUser.getFullName() +
                    " with remark " + cancelEnquiry.getRemark());
     enqActivity = enqActivityRepo.save(enqActivity);
     if (enqActivity.getEnquiryActivityId() > 0) {
      responseMap.put("responseStatus",
                  ApplicationConstants.ResponseConstants.RESPONSE_SUCCESS);
      responseMap.put("message", "Enquiry Cancelled Successfully");
     }
     responseMap.put("enquiryId", enquiry.getEnquiryId());
    }
   } else {
    responseMap.put("responseStatus",
                ApplicationConstants.ResponseConstants.RESPONSE_FAILURE);
    responseMap.put("message", "Unable to Cancel Enquiry! Try Again");
   }
   return responseMap;
  }
}
```

CHAPTER 5

TESTING

5.1 Test Strategy
5.2 Unit Test Plan
5.3 Acceptance Test Plan
5.4 Test Case / Test Script
5.5 Defect Report / Test Log

5.1 Test Strategy

To simplify the process of testing a software application, a team of testers plan a test strategy, which defines the testing approach that will be used to test the software. In the test strategy, the targets are set by the team regarding what needs to be accomplished and how it will be accomplished. It consists of a description of all the test cases and data that will be required for testing.

Overview of project:

The final product is a fully functional Sales and Service management system built on the basis of business managed platform built on long term support versions of the stable technologies of Java, Angular and MySQL. When the pitch of the project was made to the client, it was explained that many key features Managing tickets and Enquiries for the multiple customers and means to generate and view an visits and reports would significantly reduce the chances of duplication of data and provide the key differences in availability and customer interactions and increase the engagement levels of the viewers to become potential customers as well. The client after hearing the pitch wanted to develop the project because of the easy record tracking.

This release is a stable 1.0v release which includes a lot of key features like login, Managing Tickets, Managing Enquiries, Managing Visits, Search functionality including multiple filters, multiple roles login and Reports and Charts etc. Amongst these features throughout the project, since this being the academic project, manual tests would be performed on all related forms in every fields through both backend and frontend methodologies.

Features to be tested –

- Multiple Role Login
- Managing Ticket Feature – Add/Update/View Tickets
- Managing Enquiry Feature – Add/Update/View Enquiries
- Managing Visit Feature – Add/View Visits
- Managing Master Data Feature – Add/Update/Delete/View all required master data tables.
- Reports

Throughout this project Functional testing, UI testing, Load/Stress testing's are performed.

Functional Testing:

FUNCTIONAL TESTING is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application under Test. The testing can be done either manually or using automation.

In this case manual testing was conducted. Through the help of Junit and Postman the test scripts were made and tests were conducted to match their response return codes.

Integration Testing:

INTEGRATION TESTING is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated. Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as 'I & T' (Integration and Testing), 'String Testing' and sometimes 'Thread Testing'.

In this project, integration testing was used on every form submissions throughout the project and especially when testing of registration and authentication to login testing for different users and admin.

The main agenda between this being that the authenticated generated JWT token response was working as it should for both User and Admin sections and the authenticated API's were returning the valid data only when the valid token was being passed across the platform.

UI Testing:

UI Testing, also known as GUI Testing is basically a mechanism meant to test the aspects of any software that a user will come into contact with. This usually means testing the visual elements to verify that they are functioning according to requirements – in terms of functionality and performance. UI testing ensures that UI functions are bug-free.

Web Applications comprise web elements created with CSS, JavaScript, and numerous other programming languages. UI testing performs tests and assertions of these elements to validate their efficacy. It is focused on examining visual and structural parts of the software i.e. parts the user would be concerned with, rather than the internal logic of the software.

For this project, every form field is validated using standard regex to validate the entry from the user side, also return objects are working and no null pointer or exception throwing is taking place.

Load and Stress Testing:

Load Testing:

Load Testing is a type of performance testing which determines the performance of a system, software product or software application under real life based load conditions For this project since it was running on localhost under a specific port, load testing was performed by putting load for resources in the development machine and letting limited resources to interact with the rental management application.

Stress Testing:

Stress testing is a type of software testing that verifies the stability and reliability of the system. This test particularly determines the system on its robustness and error handling under extremely heavy load conditions.

For this project since it was using LTS versions of the backend and frontend software's, stress testing was conducted by involving frontend Angular to handle null pointers and backend spring boot to handle throwing exceptions and null values to be handled under a try catch statement and specific log's recognized instead of standard 'System.out.println()' for safe console printing, log4j was also used for this purpose.

5.2 Unit Test Plan

A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system. Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. The main objective of unit testing is to isolate written code to test and determine if it works as intended.

Unit testing is a component of test-driven development (TDD), a pragmatic methodology that takes a meticulous approach to building a product by means of continual testing and revision. This testing method is also the first level of software testing, which is performed before other testing methods such as integration testing. Unit tests are typically isolated to ensure a unit does not rely on any external code or functions. Testing can be done manually but is often automated.

The three stages used during unit testing in this project are as follows:

- Unit test plan: In this phase, Junit5 was used as the stable testing mechanism for testing the flow of the API data's and to process the url's.

- Unit test cases: In this phase, Junit5 was used to build the individual flow and cases for the project and also the different testing criteria was met with the try {} catch statements to handle runtime exception handling.

- Unit test scripting: In this phase, for example since STS (Spring Tool Suite) is used, Junit is integrated into the runner while testing with or without test cases. The process to execute: Right click on project -> Run as Junit test.

5.3 Acceptance Test Plan

The Acceptance Test Report (ATR) contains the summary of results obtained from executing the Acceptance Test Plan to verify the system meets all necessary requirements to satisfaction. For requirements being proven through analysis, required memos and additional documents are included in the ATR. For requirements being proven by inspection, a collection of numbered and annotated photographs is included to conclusively demonstrate the meeting of each requirement. Results of all tests run during ATP are also included to prove those requirements being met through testing. The Acceptance test plan attributes for the project are as follows:

Introduction:

The main testing for this project for acceptance is done keeping in mind of the client needs and further details on the nature of the project. Every form field is individually tested and linked together before each form submission through the frontend and the backend.

Acceptance Test Category:

User acceptance test is prioritized as per the client's demand. As the tests are targeted to fit the client needs and resources available to them. From the Admin point of view, the business acceptance tests are also kept in mind and further testing is followed via manual path to ensure the structural data integrity of the test.

Operation Environment:

The operation environment used is the minimum environment required to run the project, the testing is made in such a manner that the tests perform even under minimum requirement of the technical specifications to run the project. Ensuring data integrity is a critical path before any form submission.

Test Case Id, title and objective:

The test cases along with their title and objectives are separated in the following point below, following the methods of manual testing and thus the entire architecture of testing is kept intact and the form fields are tested individually and as a collective before submitting the document from the frontend Angular pathway and the Spring backend as well.

Test Procedure:

The procedure for testing the fields are by generating a regex for each field and taking valid, invalid and whitespace inputs to test whether the form validations suffice or not. The functional submissions are tested for the project to handle null pointers by placing proper Exception Handling mechanisms and placing the important codes within the try catch blocks wherever necessary.

Test Schedule:

The tests are scheduled whenever the user hits the certain link and the link generates the correct form needed in that page and the data is either placed for submission or retrieved and modified. Since Angular framework is a one page application, the validation tests are done instantly in every case.

Resources:

The resources from the test cases are taken from the spring starter test libraries and Junit libraries and general approaches as well as Angular standard validation class utilization approaches throughout the project and implemented on the respective forms.

5.4 Test Case / Test Script

Test Scenario ID	TC_001
Test Scenario Description	Verify the Login Functionality of Login Page
Module	Login
Sub - Module	NA
Test Case ID	TC_Login_001
Test Case Description	1. Application should be open. 2. Environment should be available
Test Steps	Open URL http://localhost:4200/
Test Data	NA
Preconditions	NA
Post Conditions	User should able to see the Login Page
Expected Result	Login Page should visible
Actual Result	Login Page is visible
Status	PASS

Test Scenario ID	TC_001
Test Scenario Description	Verify the Login Functionality of Login Page
Module	Login
Sub - Module	NA
Test Case ID	TC_Login_002
Test Case Description	Enter Valid Username and Password
Test Steps	1. Enter valid username 2. Enter valid password
Test Data	Username: admin Password: 0000
Preconditions	NA
Post Conditions	User should able to see the Dashboard
Expected Result	User should able to see the Dashboard
Actual Result	User able to see the Dashboard
Status	PASS

Test Scenario ID	TC_001
Test Scenario Description	Verify the Login Functionality of Login Page
Module	Login
Sub - Module	NA
Test Case ID	TC_Login_003
Test Case Description	Enter Invalid Username or Password
Test Steps	Enter invalid username or password
Test Data	admin 00001
Preconditions	Provide Wrong username/Password
Post Conditions	User should not able to login and see the "Wrong Username/Password! Please try again" error Message
Expected Result	User should not able to login and see the "Wrong Username/Password! Please try again" error Message
Actual Result	User is unable to login and can see the error Message
Status	PASS

Test Scenario ID	TC_001
Test Scenario Description	Verify the Login Functionality of Login Page
Module	Login
Sub - Module	NA
Test Case ID	TC_Login_004
Test Case Description	Verify that when user clicks on forgot password link, forgot password page is displayed or not.
Test Steps	Click on Forgot Password Link
Test Data	
Preconditions	NA
Post Conditions	Forgot Password Page Should Displayed
Expected Result	Forgot Password Page should be displayed
Actual Result	Forgot Password Page is displaying
Status	PASS

Test Scenario ID	TC_001
Test Scenario Description	Verify the Login Functionality of Login Page
Module	Login
Sub - Module	NA
Test Case ID	TC_Login_005
Test Case Description	1. User should enter his username and on clicking "Get OTP" if he is an employee of the organization he should get the OTP on his registered email. 2. After sending an OTP he should be able to see next screen containing OTP, New Password, Confirm New Password with "Change Password" button 3. After clicking on the "Change Password Button" OTP Should be verified and if it's correct then the given new password should be set as user's current password. if not error message should be displayed accordingly
Test Steps	1. Click on "Send OTP" Button 2. Enter the OTP received on the registered email. 3. Enter the new password. 4. Enter the same password as confirming Password. 5. Click on the "Change Password Button"
Test Data	1. OTP --> 123456 2. New Password - newpassword@123 3. Confirm new Password: newpassword@123
Preconditions	NA
Post Conditions	User will able to login with his new Password
Expected Result	1. Old Password should get changed with New Password 2. User should be able to login with New Password
Actual Result	1. Old Password is changed with New Password 2. User is able to login with New Password
Status	PASS

Test Scenario ID	TC_002
Test Scenario Description	Verify the user should able to see his own customised Dashboard after login
Module	Dashboard
Sub - Module	NA
Test Case ID	TC_Dashboard_001
Test Case Description	**Admin** Should able to see all the details of the System. i.e. Admin can see all the Tickets, Enquiries, Visits Count
Test Steps	Login using Admin credentials

Test Data	NA
Preconditions	Post-Login as Admin
Post Conditions	Admin Should able to see all the details of the System. i.e. Admin can see all the Tickets, Enquiries, Visits Count
Expected Result	Admin Should able to see all the details of the System. i.e. Admin can see all the Tickets, Enquiries, Visits Count
Actual Result	Admin Should able to see all the details of the System. i.e. Admin can see all the Tickets, Enquiries, Visits Count
Status	PASS

Test Scenario ID	**TC_003**
Test Scenario Description	Admin Should able to see Menu's in Dashboard
Module	Dashboard
Sub - Module	NA
Test Case ID	**TC_Dashboard_002**
Test Case Description	Verify that the Admin should able to See Following Menu's in his dashboard. 1. Tickets i) Manage Tickets 2. Enquiries i) Manage Visits ii) Manage Enquiries 3. Master Data i) Customers ii) Departments iii) Employees iv) Ticket Types v) Action Types vi) Brands vii) Common Replies viii) Enquiry Sources ix) Enquiry Types x) Product Categories xi) Product Types xii) Products 4. Reports
Test Steps	Login using Admin Credentials
Test Data	NA
Preconditions	Post-Login as Admin
Post Conditions	Admin able to see all Menu's on Dashboard
Expected Result	Admin should be able to see All Menu's
Actual Result	Admin is able to see the All Menu's
Status	PASS

Test Scenario ID	**TC_004**
Test Scenario Description	Sales Manager/Sales Engineer should be able to see his Menu's on Dashboard
Module	Dashboard
Sub - Module	NA
Test Case ID	**TC_Dashboard_003**
Test Case Description	Verify that the Sales Manager / Sales Engineer should able to see Following Menu's in his dashboard 1. Enquiries i) Manage Visits ii) Manage Enquiries
Test Steps	Login using Sales Manager/Sales Engineer Credentials
Test Data	NA
Preconditions	Post - Login as Sales Manager/Sales Engineer
Post Conditions	Sales Manager/Sales Engineer able to see his accessible Menus

Expected Result	Sales Manager/Sales Engineer should able to see his accessible Menus
Actual Result	Sales Manager/Sales Engineer is able to see his accessible Menus
Status	PASS

Test Scenario ID	**TC_005**
Test Scenario Description	Service Mangers/ Service Engineer should be able to see his Menu on dashboard
Module	Dashboard
Sub - Module	NA
Test Case ID	**TC_Dashboard_004**
Test Case Description	Verify that the Service Manager/ Service Engineer should be able to see following Menu's in his dashboard 1. Tickets i) Manage Tickets
Test Steps	Login using Service Manager/ Service Engineer Credentials
Test Data	NA
Preconditions	Post-Login as Service Manager / Service Engineer
Post Conditions	Service Manager / Service Engineer able to see his accessible Menus
Expected Result	Service Manager / Service Engineer should able to see his accessible Menus
Actual Result	Service Manager / Service Engineer is able to see his accessible Menus
Status	PASS

Test Scenario ID	**TC_006**
Test Scenario Description	User should be able to Logout
Module	Login
Sub - Module	NA
Test Case ID	**TC_Login_006**
Test Case Description	User should able to Logout via environment
Test Steps	Click on Logout button
Test Data	NA
Preconditions	Post-Login
Post Conditions	User is logged out
Expected Result	User should be logged out
Actual Result	User is logged out
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	**TC_ENQ_001**
Test Case Description	On clicking "Add Enquiry" enquiry page should visible
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	Add Enquiry form is visible
Expected Result	Add Enquiry form should be visible
Actual Result	Add Enquiry form is visible
Status	PASS

Test Scenario ID	TC_007
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	TC_ENQ_002
Test Case Description	Customer List Should be Prepopulated in dropdown
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	Customer List in dropdown is prepopulate and visible
Expected Result	Customer List in dropdown should be prepopulate and visible
Actual Result	Customer List in dropdown is prepopulate and visible
Status	PASS

Test Scenario ID	TC_007
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	TC_ENQ_003
Test Case Description	Enquiry source List Should be Prepopulated in dropdown
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	Enquiry Source List in dropdown is prepopulate and visible
Expected Result	Enquiry Source List in dropdown should be prepopulate and visible
Actual Result	Enquiry Source List in dropdown is prepopulate and visible
Status	PASS

Test Scenario ID	TC_007
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	TC_ENQ_004
Test Case Description	Enquiry Type List Should be Prepopulated in dropdown
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	Enquiry Type List in dropdown is prepopulate and visible
Expected Result	Enquiry Type List in dropdown should be prepopulate and visible
Actual Result	Enquiry Type List in dropdown is prepopulate and visible
Status	PASS

Test Scenario ID	TC_007
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	TC_ENQ_005
Test Case Description	Brands List Should be Prepopulated in dropdown

Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	Brands List in dropdown is prepopulate and visible
Expected Result	Brands List in dropdown should be prepopulate and visible
Actual Result	Brands List in dropdown is prepopulate and visible
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	**TC_ENQ_006**
Test Case Description	Product Type List Should be Prepopulated in dropdown
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	Product Type List in dropdown is prepopulate and visible
Expected Result	Product Type List in dropdown should be prepopulate and visible
Actual Result	Product Type List in dropdown is prepopulate and visible
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	**TC_ENQ_007**
Test Case Description	"Is Self Assigned" checkbox is should be prepopulated checked
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	"Is Self Assigned" Check box is prepopulated checked
Expected Result	"is Self Assigned" Check box should be prepopulated checked
Actual Result	"is Self Assigned" Check box is prepopulated checked
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	**TC_ENQ_008**
Test Case Description	Customer List is required to be selected
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	Customer List is required to be selected otherwise error appropriate message should be displayed
Expected Result	Customer List is required to be selected otherwise error appropriate message should be displayed
Actual Result	Customer List is required if not selected error appropriate message

	should be displayed
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	**TC_ENQ_009**
Test Case Description	Enquiry Source List is required to be selected
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	Enquiry Source List is required to be selected otherwise error appropriate message should be displayed
Expected Result	Enquiry Source List is required to be selected otherwise error appropriate message should be displayed
Actual Result	Enquiry Source List is required if not selected error appropriate message should be displayed
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	**TC_ENQ_010**
Test Case Description	Enquiry type List is required to be selected
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	Enquiry type List is required to be selected otherwise error appropriate message should be displayed
Expected Result	Enquiry type List is required to be selected otherwise error appropriate message should be displayed
Actual Result	Enquiry type List is required if not selected error appropriate message should be displayed
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	**TC_ENQ_011**
Test Case Description	Brands List is required to be selected
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	Brands List is required to be selected otherwise error appropriate message should be displayed
Expected Result	Brands List is required to be selected otherwise error appropriate message should be displayed
Actual Result	Brands List is required if not selected error appropriate message should be displayed

Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	**TC_ENQ_012**
Test Case Description	Product Type List is required to be selected
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	Product Type List is required to be selected otherwise error appropriate message should be displayed
Expected Result	Product Type List is required to be selected otherwise error appropriate message should be displayed
Actual Result	Product Type List is required if not elected error appropriate message should be displayed
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	**TC_ENQ_013**
Test Case Description	Enquiry Remark is required.
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	If Enquiry remark not entered an error message is displayed.
Expected Result	If Enquiry remark not entered an error message should displayed.
Actual Result	If Enquiry remark not entered an error message is displayed.
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	**TC_ENQ_014**
Test Case Description	Enquiry Remark should contain at least 5 characters.
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	if enquiry remark is containing less than 5 character then an error message is displayed.
Expected Result	if enquiry remark is containing less than 5 character then an error message should displayed.
Actual Result	if enquiry remark is containing less than 5 character then an error message is displayed.
Status	PASS

Test Scenario ID	TC_007
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	TC_ENQ_015
Test Case Description	Unless all required fields selected and entered "Save" button should not be enabled
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	After entering all required fields selected and entered "Save" button is enabled
Expected Result	Unless all required fields selected and entered "Save" button should not be enabled
Actual Result	After entering all required fields selected and entered "Save" button is enabled
Status	PASS

Test Scenario ID	TC_007
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	TC_ENQ_016
Test Case Description	After selecting Brands and Product Type following fields should be visible. 1. Product List 2. Product Quantity 3. Product Remark 4. Add Product Button.
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	After selecting Brands and Product Type following fields is visible. 1. Product List 2. Product Quantity 3. Product Remark 4. Add Product Button.
Expected Result	After selecting Brands and Product Type following fields should be visible. 1. Product List 2. Product Quantity 3. Product Remark 4. Add Product Button.
Actual Result	After selecting Brands and Product Type following fields is visible. 1. Product List 2. Product Quantity 3. Product Remark 4. Add Product Button.
Status	PASS

Test Scenario ID	TC_007
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry

Test Case ID	**TC_ENQ_017**
Test Case Description	At least one product is should be select to add Enquiry successfully
Test Steps	Enquiries > Manage Enquiries > Add Enquiry
Test Data	NA
Preconditions	Post Login
Post Conditions	At least one product is should be select to add Enquiry successfully. After adding at least 1 product user is able to add Enquiry. Otherwise error message is displayed.
Expected Result	At least one product is should be select to add Enquiry successfully. Otherwise error message should be displayed.
Actual Result	At least one product is should be select to add Enquiry successfully. After adding at least 1 product user is able to add Enquiry. Otherwise error message is displayed.
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	**TC_ENQ_018**
Test Case Description	If user wants to add a new customer, on-clicking "New Customer?" button "Add customer" form should be populated.
Test Steps	Enquiries > Manage Enquiries > Add Enquiry > New Customer?
Test Data	NA
Preconditions	Post Login
Post Conditions	On clicking "New Customer?" add customer form is opened.
Expected Result	On clicking "New Customer?" add customer form should be opened.
Actual Result	On clicking "New Customer?" add customer form is opened.
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	Add Enquiry
Test Case ID	**TC_ENQ_019**
Test Case Description	All Fields are required from the "Add Customer" form
Test Steps	Enquiries > Manage Enquiries > Add Enquiry > New Customer?
Test Data	NA
Preconditions	Post Login
Post Conditions	All Fields are validated. If user fails to fill any field then appropriate error message is displayed. Once all fields are entered then only save button is enabled.
Expected Result	All Fields should be validated. If user fails to fill any field then appropriate error message should be displayed. Once all fields are entered then only save button should be enabled.
Actual Result	All Fields are validated. If user fails to fill any field then appropriate error message is displayed. Once all fields are entered then only save button is enabled.
Status	PASS

Test Scenario ID	TC_007
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	TC_ENQ_020
Test Case Description	Admin/ Sales Manager/ Sales Engineer should be able to see enquires that are available
Test Steps	Enquiries > Manage Enquires
Test Data	NA
Preconditions	Post Login
Post Conditions	Mentioned users are able to see all the enquiries available
Expected Result	Mentioned users should able to see all the enquiries available
Actual Result	Mentioned users are able to see all the enquiries available
Status	PASS

Test Scenario ID	TC_007
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	TC_ENQ_021
Test Case Description	All the Search Filters should be working fine and should be returning the selected filters filtered data only
Test Steps	Enquiries > Manage Enquires > Search Filters
Test Data	NA
Preconditions	Post Login
Post Conditions	All search filters are working fine and returning selected filters filtered data.
Expected Result	All search filters should be working fine and returning selected filters filtered data.
Actual Result	All search filters are working fine and returning selected filters filtered data.
Status	PASS

Test Scenario ID	TC_007
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	TC_ENQ_022
Test Case Description	Pagination should work. Following are the conditions. 1. On single page there should be only 10 maximum entries 2. Next, Previous button should be working 3. Next, Prev buttons should be disabled if there is no data further.
Test Steps	NA
Test Data	NA
Preconditions	Post Login
Post Conditions	1. There are 10 maximum entries on each page. 2. Next, Previous buttons are working 3. Next, Prev buttons are disabled when there is no data further.
Expected Result	Pagination should work. Following are the conditions. 1. On single page there should be only 10 maximum entries 2. Next, Previous button should be working 3. Next, Prev buttons should be disabled if there is no data further.

Actual Result	Pagination is working. With the following conditions. 1. On single page there are 10 maximum entries 2. Next, Previous button are working 3. Next, Prev buttons are disabled when there is no data further.
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	**TC_ENQ_023**
Test Case Description	Verify that on the Manage Enquiry Page check for every enquiry in the enquiry table there should be appropriate button should be visible on the basis of enquiry status. According to enquiry status following buttons should be visible. 1. Unassigned - Edit Button 2. Assigned - Edit Button 3. Cancel - View Button 4. Prospect - Edit Button 5. Won - View Button 6. Lost - View Button.
Test Steps	Enquiries > Manage Enquiries
Test Data	At least 10 enquiries should be added and modify according wrt status
Preconditions	Post Login
Post Conditions	According to enquiry status following buttons is visible. 1. Unassigned - Edit Button 2. Assigned - Edit Button 3. Cancel - View Button 4. Prospect - Edit Button 5. Won - View Button 6. Lost - View Button.
Expected Result	According to enquiry status following buttons should be visible. 1. Unassigned - Edit Button 2. Assigned - Edit Button 3. Cancel - View Button 4. Prospect - Edit Button 5. Won - View Button 6. Lost - View Button.
Actual Result	According to enquiry status following buttons is visible. 1. Unassigned - Edit Button 2. Assigned - Edit Button 3. Cancel - View Button 4. Prospect - Edit Button 5. Won - View Button 6. Lost - View Button.
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	**TC_ENQ_024**
Test Case Description	Verify that for "Unassigned" Status on clicking "Edit button following

	things 1. On Clicking "Edit Button" Enquiry Details page should be open. 2. All the enquiry details should be there. 3. For "Unassigned" Status an "Action" Button should be visible and contain following actions a) Assign Enquiry b) Cancel Enquiry. 4. Enquiry Details page contains at least one enquiry activities log.
Test Steps	Enquiries > Manage Enquiries > Unassigned Status Enquiry > Edit
Test Data	Enquiry with "Unassigned" Status
Preconditions	Enquiry should have status "Unassigned"
Post Conditions	User is able to take following actions on enquiry with "Unassigned' Status 1. User is able assign enquiry to the employee. 2. User is able to cancel enquiry. 3. Enquiry Details page contains Enquiry activities logs
Expected Result	User should able to take following actions on enquiry with "Unassigned' Status 1. User should able assign enquiry to the employee. 2. User should able to cancel enquiry. 3. Enquiry Details page contains Enquiry activities logs
Actual Result	User is able to take following actions on enquiry with "Unassigned' Status 1. User is able assign enquiry to the employee. 2. User is able to cancel enquiry. 3. Enquiry Details page contains Enquiry activities logs
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	**TC_ENQ_025**
Test Case Description	Verify that for "Assigned" Status on clicking "Edit button following things 1. On Clicking "Edit Button" Enquiry Details page should be open. 2. All the enquiry details should be there. 3. For "Assigned" Status an "Action" Button should be visible and contain following actions a) Change to Prospect b) Cancel Enquiry. 4. Enquiry Details page have an enquiry activities log.
Test Steps	Enquiries > Manage Enquiries > Assigned Status Enquiry > Edit
Test Data	Enquiry with "Assigned" Status
Preconditions	Enquiry should have status "Assigned"
Post Conditions	
Expected Result	User should able to take following actions on enquiry with "Assigned' Status 1. User should able change enquiry to "Change to Prospect" 2. User should able to cancel enquiry. 3. Enquiry Details page contains Enquiry activities logs
Actual Result	User is able to take following actions on enquiry with "Assigned' Status

	1. User is able change enquiry to "Change to Prospect" 2. User is able to cancel enquiry. 3. Enquiry Details page contains Enquiry activities logs
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	**TC_ENQ_026**
Test Case Description	Verify that for "Prospect" Status on clicking "Edit button following things 1. On Clicking "Edit Button" Enquiry Details page should be open. 2. All the enquiry details should be there. 3. For "Prospect" Status an "Action" Button should be visible and contain following actions a) Won Enquiry b) Lost Enquiry
Test Steps	Enquiries > Manage Enquiries > Prospect Status Enquiry > Edit
Test Data	Enquiry with "Prospect" Status
Preconditions	Enquiry should have status "Prospect"
Post Conditions	User is able to take following actions on enquiry with "Prospect" Status 1. User is able to Won enquiry 2. User is able to lost enquiry. 3. Enquiry Details page contains Enquiry activities logs
Expected Result	User is able to take following actions on enquiry with "Prospect" Status 1. User should able to Won enquiry 2. User should able to lost enquiry. 3. Enquiry Details page contains Enquiry activities logs
Actual Result	User is able to take following actions on enquiry with "Prospect" Status 1. User is able to Won enquiry 2. User is able to lost enquiry. 3. Enquiry Details page contains Enquiry activities logs
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	**TC_ENQ_027**
Test Case Description	Verify that for "Won" Status on clicking "Edit button following things 1. On Clicking "View" Button Enquiry Details page should be open. 2. All the enquiry details should be there. 3. For "Won" Status an "Action" Button should not be visible 4. Enquiry Details page contains Enquiry activities logs
Test Steps	Enquiries > Manage Enquiries > Won Status Enquiry > Edit
Test Data	Enquiry with "Won" Status
Preconditions	Enquiry should have status "Won"
Post Conditions	User is able to take following actions on enquiry with "Won" Status 1. User is able to see All Enquiry Details 2. "Action" button is not visible

	3. Enquiry Details page contains Enquiry activities logs
Expected Result	User is able to take following actions on enquiry with "Won" Status 1. User should able to see All Enquiry Details 2. "Action" button is not visible 3. Enquiry Details page contains Enquiry activities logs
Actual Result	User is able to take following actions on enquiry with "Won" Status 1. User is able to see All Enquiry Details 2. "Action" button is not visible 3. Enquiry Details page contains Enquiry activities logs
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	**TC_ENQ_028**
Test Case Description	Verify that for "Lost" Status on clicking "Edit button following things 1. On Clicking "View" Button Enquiry Details page should be open. 2. All the enquiry details should be there. 3. For "Lost" Status an "Action" Button should not be visible 4. Enquiry Details page contains Enquiry activities logs
Test Steps	Enquiries > Manage Enquiries > "Lost" Status Enquiry > Edit
Test Data	Enquiry with "Lost" Status
Preconditions	Enquiry should have status "Lost"
Post Conditions	User is able to take following actions on enquiry with "Lost" Status 1. User is able to see All Enquiry Details 2. "Action" button is not visible 3. Enquiry Details page contains Enquiry activities logs
Expected Result	User is able to take following actions on enquiry with "Lost" Status 1. User should able to see All Enquiry Details 2. "Action" button is not visible 3. Enquiry Details page contains Enquiry activities logs
Actual Result	User is able to take following actions on enquiry with "Lost" Status 1. User is able to see All Enquiry Details 2. "Action" button is not visible 3. Enquiry Details page contains Enquiry activities logs
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	**TC_ENQ_029**
Test Case Description	Verify that for "Cancelled" Status on clicking "Edit button following things 1. On Clicking "View" Button Enquiry Details page should be open. 2. All the enquiry details should be there. 3. For "Cancelled" Status an "Action" Button should not be visible 4. Enquiry Details page contains Enquiry activities logs
Test Steps	Enquiries > Manage Enquiries > "Cancelled" Status Enquiry > Edit
Test Data	Enquiry with "Cancelled" Status
Preconditions	Enquiry should have status "Cancelled"

Post Conditions	User is able to take following actions on enquiry with "Cancelled" Status 1. User is able to see All Enquiry Details 2. "Action" button is not visible 3. Enquiry Details page contains Enquiry activities logs
Expected Result	User is able to take following actions on enquiry with "Cancelled" Status 1. User should able to see All Enquiry Details 2. "Action" button is not visible 3. Enquiry Details page should contains Enquiry activities logs
Actual Result	User is able to take following actions on enquiry with "Cancelled" Status 1. User is able to see All Enquiry Details 2. "Action" button is not visible 3. Enquiry Details page contains Enquiry activities logs
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	**TC_ENQ_030**
Test Case Description	Verify that "Assign Enquiry" form contains following 1. Assign to List 2. Remark 3. All fields are required and should not be empty.
Test Steps	Enquiries > Manage Enquiries > Edit
Test Data	Enquiry with "Unassigned" Status
Preconditions	Enquiry should have status "Unassigned"
Post Conditions	"Assign Enquiry" form contains following 1. Assign to List 2. Remark 3. All fields are required and should not be empty. If empty appropriate error message is displayed.
Expected Result	"Assign Enquiry" form contains following 1. Assign to List 2. Remark 3. All fields should be required and should not be empty. If empty appropriate error message is displayed.
Actual Result	"Assign Enquiry" form contains following 1. Assign to List 2. Remark 3. All fields are required and should not be empty. If empty appropriate error message is displayed.
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	**TC_ENQ_031**
Test Case Description	Verify that "Change to prospect" form contains following 1. Remark

	2. Remark field is required and should not be empty.
Test Steps	Enquiries > Manage Enquiries > Edit
Test Data	Enquiry with "Assigned" Status
Preconditions	Enquiry should have status "Assigned"
Post Conditions	"Change to prospect" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Expected Result	"Change to prospect" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Actual Result	"Change to prospect" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	**TC_ENQ_032**
Test Case Description	Verify that "Won" form contains following 1. Machine Serial Number 2. Purchase Date 3. Remark 4. Machine Serial Number, Purchase Date and Remark fields is required and should not be empty.
Test Steps	Enquiries > Manage Enquiries > Edit
Test Data	Enquiry with "Prospect" Status
Preconditions	Enquiry should have status "Prospect"
Post Conditions	"Won" form contains following 1. Machine Serial Number 2. Purchase Date 3. Remark 4. Machine Serial Number, Purchase Date and Remark fields is required and should not be empty. If it is empty then appropriate error message is displayed.
Expected Result	"Won" form contains following 1. Machine Serial Number 2. Purchase Date 3. Remark 4. Machine Serial Number, Purchase Date and Remark fields is required and should not be empty. If it is empty then appropriate error message is displayed.
Actual Result	"Won" form contains following 1. Machine Serial Number 2. Purchase Date 3. Remark 4. Machine Serial Number, Purchase Date and Remark fields is required and should not be empty. If it is empty then appropriate error message is displayed.
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access

	Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	**TC_ENQ_033**
Test Case Description	Verify that "Lost" form contains following 1. Remark 2. Remark field is required and should not be empty.
Test Steps	Enquiries > Manage Enquiries > Edit
Test Data	Enquiry with "Prospect" Status
Preconditions	Enquiry should have status "Prospect"
Post Conditions	"Lost" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Expected Result	"Lost" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Actual Result	"Lost" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Status	PASS

Test Scenario ID	**TC_007**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Enquiries
Module	Enquiries
Sub - Module	View Enquiry
Test Case ID	**TC_ENQ_034**
Test Case Description	Verify that "Cancelled" form contains following 1. Remark 2. Remark field is required and should not be empty.
Test Steps	Enquiries > Manage Enquiries > Edit
Test Data	Enquiry with "Assigned", "Unassigned" Status
Preconditions	Enquiry should have status "Assigned", "Unassigned"
Post Conditions	"Cancelled" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Expected Result	"Cancelled" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Actual Result	"Cancelled" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Status	PASS

Test Scenario ID	**TC_008**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Visits
Module	Visit
Sub - Module	NA
Test Case ID	**TC_VISIT_001**
Test Case Description	On Clicking "Manage Visits" all visits should be displayed.

Test Steps	Enquiries > Manage Visits
Test Data	NA
Preconditions	Post Login
Post Conditions	User is able to see the Manage Visit Page
Expected Result	User should able to see the Manage Visit Page
Actual Result	User is able to see the Manage Visit Page
Status	PASS

Test Scenario ID	**TC_008**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Visits
Module	Visit
Sub - Module	NA
Test Case ID	**TC_VISIT_002**
Test Case Description	On page opening filter should be set as following. 1. Start Date should be 7 days before current date. 2. End date should be todays date. 3. All other dropdown Filters should be prepopulated.
Test Steps	Enquiries > Manage Visits
Test Data	NA
Preconditions	Post Login
Post Conditions	On page opening filter working as following. 1. Start Date is 7 days before current date. 2. End date is todays date. 3. All other dropdown Filters are prepopulated.
Expected Result	On page opening filter should be set as following. 1. Start Date should be 7 days before current date. 2. End date should be todays date. 3. All other dropdown Filters should be prepopulated.
Actual Result	On page opening filter working as following. 1. Start Date is 7 days before current date. 2. End date is todays date. 3. All other dropdown Filters are prepopulated.
Status	PASS

Test Scenario ID	**TC_008**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Visits
Module	Visit
Sub - Module	NA
Test Case ID	**TC_VISIT_003**
Test Case Description	On "Manage Visit" Each visit record contains an "View" button. On-clicking "View" button visit details should be displayed on a modal
Test Steps	Enquiries > Manage Visits > View
Test Data	NA
Preconditions	Post Login
Post Conditions	On Clicking "View" Button all the visits details are displayed on a modal.
Expected Result	On Clicking "View" Button all the visits details should be displayed on a modal.
Actual Result	On Clicking "View" Button all the visits details are displayed on a modal.
Status	PASS

Test Scenario ID	TC_008
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Visits
Module	Visit
Sub - Module	Add Visit
Test Case ID	TC_VISIT_004
Test Case Description	On Clicking "Add Visit" Button an Add Visit Form should be Opened.
Test Steps	Enquiries > Manage Visits > Add Visit
Test Data	NA
Preconditions	Post Login
Post Conditions	On Clicking "Add Visit" Button an Add Visit Form is Opened.
Expected Result	On Clicking "Add Visit" Button an Add Visit Form should be Opened.
Actual Result	On Clicking "Add Visit" Button an Add Visit Form is Opened.
Status	PASS

Test Scenario ID	TC_008
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Visits
Module	Visit
Sub - Module	Add Visit
Test Case ID	TC_VISIT_005
Test Case Description	Verify that on opening of "Add Visit" Page Customer list should be Prepopulated.
Test Steps	Enquiries > Manage Visits > Add Visit
Test Data	NA
Preconditions	Post Login
Post Conditions	On opening of "Add Visit" Page Customer list is being Prepopulated.
Expected Result	On opening of "Add Visit" Page Customer list should be Prepopulated.
Actual Result	On opening of "Add Visit" Page Customer list is being Prepopulated.
Status	PASS

Test Scenario ID	TC_008
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Visits
Module	Visit
Sub - Module	Add Visit
Test Case ID	TC_VISIT_006
Test Case Description	Verify that on opening of "Add Visit" Page Action Type list should be Prepopulated.
Test Steps	Enquiries > Manage Visits > Add Visit
Test Data	NA
Preconditions	Post Login
Post Conditions	On opening of "Add Visit" Page Action Type list is being Prepopulated.
Expected Result	On opening of "Add Visit" Page Action Type list should be Prepopulated.
Actual Result	On opening of "Add Visit" Page Action Type list is being Prepopulated.
Status	PASS

Test Scenario ID	TC_008
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Visits
Module	Visit
Sub - Module	Add Visit
Test Case ID	TC_VISIT_007
Test Case Description	Verify on Selecting a customer from "Customer List" Dropdown below

	mentioned all information of the customer should be fetched and displayed to their places accordingly. Information should be disabled and restrict user from changing/updating any kind of data of customer. 1. Customer Email. 2. Mobile Number. 3. Customer Address
Test Steps	Enquiries > Manage Visits > Add Visit > Customer List
Test Data	NA
Preconditions	Post Login
Post Conditions	On Selecting a customer from "Customer List" Dropdown below mentioned all information of the customer is fetched and displayed to their places accordingly. Information is disabled and restrict user from changing/updating any kind of data of customer. 1. Customer Email. 2. Mobile Number. 3. Customer Address
Expected Result	On Selecting a customer from "Customer List" Dropdown below mentioned all information of the customer should be fetched and displayed to their places accordingly. Information should be disabled and restrict user from changing/updating any kind of data of customer. 1. Customer Email. 2. Mobile Number. 3. Customer Address
Actual Result	On Selecting a customer from "Customer List" Dropdown below mentioned all information of the customer is fetched and displayed to their places accordingly. Information is disabled and restrict user from changing/updating any kind of data of customer. 1. Customer Email. 2. Mobile Number. 3. Customer Address
Status	PASS

Test Scenario ID	**TC_008**
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Visits
Module	Visit
Sub - Module	Add Visit
Test Case ID	**TC_VISIT_008**
Test Case Description	Verify that fields Action Type and Visit Description are mandatory.
Test Steps	Enquiries > Manage Visits > Add Visit
Test Data	NA
Preconditions	Post Login
Post Conditions	Fields Action Type and Visit Description are mandatory. If not filled or selected an error message is displayed.
Expected Result	Fields Action Type and Visit Description are mandatory. If not filled or selected an error message should be displayed.
Actual Result	Fields Action Type and Visit Description are mandatory. If not filled or selected an error message is displayed.
Status	PASS

Test Scenario ID	TC_008
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Visits
Module	Visit
Sub - Module	Add Visit
Test Case ID	TC_VISIT_009
Test Case Description	Verify Following: 1. On Clicking "Select common Replies" Link a modal should open and a list of common replies fetched. 2. User should able to select these common replies. 3. After user selected the common Replies they should be added in Visit Description.
Test Steps	Enquiries > Manage Visits > Add Visit > Select Common Replies
Test Data	NA
Preconditions	Post Login
Post Conditions	1. On Clicking "Select common Replies" Link a modal is opened and a list of common replies is fetched. 2. User is able to select these common replies. 3. After user selected the common Replies they gets added in Visit Description.
Expected Result	1. On Clicking "Select common Replies" Link a modal should open and a list of common replies fetched. 2. User should able to select these common replies. 3. After user selected the common Replies they should be added in Visit Description.
Actual Result	1. On Clicking "Select common Replies" Link a modal is opened and a list of common replies is fetched. 2. User is able to select these common replies. 3. After user selected the common Replies they gets added in Visit Description.
Status	PASS

Test Scenario ID	TC_008
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Visits
Module	Visit
Sub - Module	Add Visit
Test Case ID	TC_VISIT_010
Test Case Description	Verify that till All mandatory fields are not filled the "Save" button should be disabled.
Test Steps	Enquiries > Manage Visits > Add Visit
Test Data	NA
Preconditions	Post Login
Post Conditions	Till All mandatory fields are not filled the "Save" button is disabled.
Expected Result	Till All mandatory fields are not filled the "Save" button should be disabled.
Actual Result	Till All mandatory fields are not filled the "Save" button is disabled.
Status	PASS

Test Scenario ID	TC_008
Test Scenario Description	Admin/ Sales Manager/ Sales Engineer should be able to access Visits
Module	Visit
Sub - Module	Add Visit
Test Case ID	TC_VISIT_011
Test Case Description	Verify that on Clicking "Save" Button a new visit should get added.
Test Steps	Enquiries > Manage Visits > Add Visit

Test Data	All Mandatory fields should be entered/Selected.
Preconditions	Post Login
Post Conditions	On Clicking "Save" Button a new visit get added.
Expected Result	On Clicking "Save" Button a new visit should get added.
Actual Result	On Clicking "Save" Button a new visit get added.
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	Add Tickets
Test Case ID	**TC_TIC_001**
Test Case Description	On clicking "Add Ticket" add ticket form page should visible
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	Add Ticket form is visible
Expected Result	Add Ticket form should be visible
Actual Result	Add Ticket form is visible
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	Add Tickets
Test Case ID	**TC_TIC_002**
Test Case Description	Customer List Should be Prepopulated in dropdown
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	Customer List in dropdown is prepopulate and visible
Expected Result	Customer List in dropdown should be prepopulate and visible
Actual Result	Customer List in dropdown is prepopulate and visible
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	Add Tickets
Test Case ID	**TC_TIC_003**
Test Case Description	Ticket Type List Should be Prepopulated in dropdown
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	Ticket Type List in dropdown is prepopulate and visible
Expected Result	Ticket Type List in dropdown should be prepopulate and visible
Actual Result	Ticket Type List in dropdown is prepopulate and visible
Status	PASS

Test Scenario ID	TC_009
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	Add Tickets
Test Case ID	TC_TIC_004
Test Case Description	Priority List Should be Prepopulated in dropdown
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	Priority List in dropdown is prepopulate and visible
Expected Result	Priority List in dropdown should be prepopulate and visible
Actual Result	Priority List in dropdown is prepopulate and visible
Status	PASS

Test Scenario ID	TC_009
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	Add Tickets
Test Case ID	TC_TIC_005
Test Case Description	Assign To List Should be Prepopulated in dropdown
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	Assign To List in dropdown is prepopulate and visible
Expected Result	Assign To List in dropdown should be prepopulate and visible
Actual Result	Assign To List in dropdown is prepopulate and visible
Status	PASS

Test Scenario ID	TC_009
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	Add Tickets
Test Case ID	TC_TIC_006
Test Case Description	Customer List is required to be selected
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	Customer List is required to be selected otherwise error appropriate message should be displayed
Expected Result	Customer List is required to be selected otherwise error appropriate message should be displayed
Actual Result	Customer List is required if not selected error appropriate message should be displayed
Status	PASS

Test Scenario ID	TC_009
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	Add Tickets
Test Case ID	TC_TIC_007

Test Case Description	Ticket Type List is required to be selected
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	Ticket Type List is required to be selected otherwise error appropriate message should be displayed
Expected Result	Ticket Type List is required to be selected otherwise error appropriate message should be displayed
Actual Result	Ticket Type List is required if not selected error appropriate message should be displayed
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	Add Tickets
Test Case ID	**TC_TIC_008**
Test Case Description	Priority List is required to be selected
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	Priority List is required to be selected otherwise error appropriate message should be displayed
Expected Result	Priority List is required to be selected otherwise error appropriate message should be displayed
Actual Result	Priority List is required if not selected error appropriate message should be displayed
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	Add Tickets
Test Case ID	**TC_TIC_009**
Test Case Description	Customer Product List Should be fetched once customer is selected from customer list
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	Customer Product List is fetched when customer is selected from customer list
Expected Result	Customer Product List Should be fetched once customer is selected from customer list
Actual Result	Customer Product List is fetched when customer is selected from customer list
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	Add Tickets
Test Case ID	**TC_TIC_010**

Test Case Description	Verify following validations for "Short Description" Field 1. "Short Description" field is mandatory. 2. "Short Description" field should not be empty, if it is empty show appropriate error message
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	1. "Short Description" field is mandatory. 2. "Short Description" field should not be empty, if it is empty show appropriate error message
Expected Result	Verify following validations for "Short Description" Field 1. "Short Description" field is mandatory. 2. "Short Description" field should not be empty, if it is empty show appropriate error message
Actual Result	Verify following validations for "Short Description" Field 1. "Short Description" field is mandatory. 2. "Short Description" field should not be empty, if it is empty show appropriate error message
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	Add Tickets
Test Case ID	**TC_TIC_011**
Test Case Description	Verify following validations for "Long Description" Field 1. "Long Description" field is mandatory. 2. "Long Description" field should not be empty, if it is empty show appropriate error message
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	1. "Long Description" field is mandatory. 2. "Long Description" field should not be empty, if it is empty show appropriate error message
Expected Result	1. "Long Description" field is mandatory. 2. "Long Description" field should not be empty, if it is empty show appropriate error message
Actual Result	1. "Long Description" field is mandatory. 2. "Long Description" field should not be empty, if it is empty show appropriate error message
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	Add Tickets
Test Case ID	**TC_TIC_012**
Test Case Description	Verify that "Save" button on "Add Ticket" Form is disabled till all mandatory fields are filled.
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	"Save" button on "Add Ticket" Form is disabled till all mandatory fields are filled.

Expected Result	"Save" button on "Add Ticket" Form should be disabled till all mandatory fields are filled.
Actual Result	"Save" button on "Add Ticket" Form is disabled till all mandatory fields are filled.
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_013**
Test Case Description	Pagination should work. Following are the conditions. 1. On single page there should be only 10 maximum entries 2. Next, Previous button should be working 3. Next, Prev buttons should be disabled if there is no data further.
Test Steps	Tickets > Manage Tickets > Add Ticket
Test Data	NA
Preconditions	Post Login
Post Conditions	1. There are 10 maximum entries on each page. 2. Next, Previous buttons are working 3. Next, Prev buttons are disabled when there is no data further.
Expected Result	Pagination should work. Following are the conditions. 1. On single page there should be only 10 maximum entries 2. Next, Previous button should be working 3. Next, Prev buttons should be disabled if there is no data further.
Actual Result	Pagination is working. With the following conditions. 1. On single page there are 10 maximum entries 2. Next, Previous button are working 3. Next, Prev buttons are disabled when there is no data further.
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_014**
Test Case Description	On page opening filter should be set as following. 1. Start Date should be 7 days before current date. 2. End date should be todays date. 3. All other following dropdown Filters should be prepopulated. i) Ticket Types ii) Priority iii) Status iv) Customer v) Added By (List of System Users)
Test Steps	Tickets > Manage Tickets
Test Data	NA
Preconditions	Post Login
Post Conditions	On page opening filter working as following. 1. Start Date is 7 days before current date. 2. End date is todays date. 3. All other following dropdown Filters are prepopulated. i) Ticket Types ii) Priority

	iii) Status iv) Customer v) Added By (List of System Users)
Expected Result	On page opening filter should be set as following. 1. Start Date should be 7 days before current date. 2. End date should be todays date. 3. All other following dropdown Filters should be prepopulated. i) Ticket Types ii) Priority iii) Status iv) Customer v) Added By (List of System Users)
Actual Result	On page opening filter working as following. 1. Start Date is 7 days before current date. 2. End date is todays date. 3. All other following dropdown Filters are prepopulated. i) Ticket Types ii) Priority iii) Status iv) Customer v) Added By (List of System Users)
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_015**
Test Case Description	Verify that on the Manage Ticket Page check for every ticket in the ticket table there should be appropriate button visible on the basis of ticket status. According to ticket status following buttons should be visible. 1. Unassigned - Edit Button 2. Assigned - Edit Button 3. Cancelled - View Button 4. In Progress - Edit Button 5. Completed - View Button 6. Closed - View Button
Test Steps	Tickets > Manage Tickets
Test Data	At least 10 tickets should be added and modify according wrt status
Preconditions	Post Login
Post Conditions	According to ticket status following buttons are be visible. 1. Unassigned - Edit Button 2. Assigned - Edit Button 3. Cancelled - View Button 4. In Progress - Edit Button 5. Completed - View Button 6. Closed - View Button
Expected Result	According to ticket status following buttons should be visible. 1. Unassigned - Edit Button 2. Assigned - Edit Button 3. Cancelled - View Button 4. In Progress - Edit Button 5. Completed - View Button 6. Closed - View Button
Actual Result	According to ticket status following buttons are be visible. 1. Unassigned - Edit Button 2. Assigned - Edit Button 3. Cancelled - View Button 4. In Progress - Edit Button

	5. Completed - View Button 6. Closed - View Button
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_016**
Test Case Description	Verify that for "Unassigned" Status on clicking "Edit button following things 1. On Clicking "Edit Button" ticket Details page should be open. 2. All the ticket details should be there. 3. For "Unassigned" Status an "Action" Button should be visible and contain following actions i) Assign Ticket. ii) Cancel Ticket. 4. Enquiry Details page contains at least one enquiry activities log.
Test Steps	Tickets > Manage Tickets > Unassigned Status Ticket > Edit
Test Data	Ticket with "Unassigned" Status
Preconditions	Ticket should have status "Unassigned"
Post Conditions	User is able to take following actions on Ticket with "Unassigned' Status 1. User is able assign Ticket to the employee. 2. User is able to cancel Ticket. 3. Ticket Details page contains ticket activities logs
Expected Result	User should able to take following actions on Ticket with "Unassigned' Status 1. User should able assign Ticket to the employee. 2. User should able to cancel Ticket. 3. Ticket Details page contains ticket activities logs
Actual Result	User is able to take following actions on Ticket with "Unassigned' Status 1. User is able assign Ticket to the employee. 2. User is able to cancel Ticket. 3. Ticket Details page contains ticket activities logs
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_017**
Test Case Description	Verify that for "Assigned" Status on clicking "Edit button following things 1. On Clicking "Edit Button" Ticket Details page should be open. 2. All the Ticket details should be there. 3. For "Assigned" Status an "Action" Button should be visible and contain following actions i) Change to In Progress ii) Cancel Ticket 4. Ticket Details page have an enquiry activities log.
Test Steps	Tickets > Manage Tickets > "Assigned" Status Ticket > Edit

Test Data	Ticket with "Assigned" Status
Preconditions	Ticket should have status "Assigned"
Post Conditions	User is able to take following actions on Ticket with "Assigned' Status 1. User is able change Ticket to "Change to In Progress" 2. User is able to cancel Ticket. 3. Ticket Details page contains ticket activities logs
Expected Result	User should able to take following actions on enquiry with "Assigned' Status 1. User should able change enquiry to "Change to In Progress" 2. User should able to cancel Ticket. 3. Ticket Details page contains ticket activities logs
Actual Result	User is able to take following actions on Ticket with "Assigned' Status 1. User is able change Ticket to "Change to In Progress" 2. User is able to cancel Ticket. 3. Ticket Details page contains ticket activities logs
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_018**
Test Case Description	Verify that for "In Progress" Status on clicking "Edit button following things 1. On Clicking "Edit Button" ticket Details page should be open. 2. All the ticket details should be there. 3. For "In Progress" Status an "Action" Button should be visible and contain following actions i) Complete Ticket ii) Cancel Ticket
Test Steps	Tickets > Manage Tickets > In Progress Status Ticket > Edit
Test Data	Ticket with "In Progress" Status
Preconditions	Ticket should have status "In Progress"
Post Conditions	User is able to take following actions on ticket with "In Progress" Status 1. User is able to complete ticket 2. User is able to cancel ticket 3. Ticket Details page contains ticket activities logs
Expected Result	User is able to take following actions on ticket with "In Progress" Status 1. User should able to complete ticket. 2. User should able to cancel ticket 3. Ticket Details page contains ticket activities logs
Actual Result	User is able to take following actions on ticket with "In Progress" Status 1. User is able to complete ticket 2. User is able to cancel ticket 3. Ticket Details page contains ticket activities logs
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_019**
Test Case Description	Verify that for "Completed" Status on clicking View button following

	things 1. On Clicking "View" Button Ticket Details page should be open. 2. All the ticket details should be there. 3. For Completed Status an "Action" Button should not be visible 4. Ticket Details page contains ticket activities logs
Test Steps	Tickets > Manage Tickets > Completed Status Ticket > Edit
Test Data	Ticket with "Completed" Status
Preconditions	Ticket should have status "Completed"
Post Conditions	User is able to take following actions on enquiry with "Completed" Status 1. On Clicking "View" Button Ticket Details page is opened. 2. All the ticket details are there. 3. For Completed Status an "Action" Button is not visible 4. Ticket Details page contains ticket activities logs
Expected Result	User is able to take following actions on enquiry with "Cancelled" Status 1. On Clicking "View" Button Ticket Details page should be open. 2. All the ticket details should be there. 3. For Completed Status an "Action" Button should not be visible 4. Ticket Details page contains ticket activities logs
Actual Result	User is able to take following actions on enquiry with "Completed" Status 1. On Clicking "View" Button Ticket Details page is opened. 2. All the ticket details are there. 3. For Completed Status an "Action" Button is not visible 4. Ticket Details page contains ticket activities logs
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_020**
Test Case Description	Verify that for "Cancelled" Status on clicking View button following things 1. On Clicking "View" Button ticket Details page should be open. 1. All the ticket details should be there. 2. For "Cancelled" Status an "Action" Button should not be visible 3. Ticket Details page contains ticket activities logs
Test Steps	Tickets > Manage Tickets > "Cancelled" Status Ticket > Edit
Test Data	Ticket with "Cancelled" Status
Preconditions	Ticket should have status "Cancelled"
Post Conditions	User is able to take following actions on enquiry with "Cancelled" Status 1. On Clicking "View" Button ticket Details page is opened. 2. All the ticket details are there. 3. For "Cancelled" Status an "Action" Button is not visible 4. Ticket Details page contains ticket activities logs

Expected Result	User is able to take following actions on enquiry with "Cancelled" Status 1. On Clicking "View" Button ticket Details page should be open. 2. All the ticket details should be there. 3. For "Cancelled" Status an "Action" Button should not be visible 4. Ticket Details page contains ticket activities logs
Actual Result	User is able to take following actions on enquiry with "Cancelled" Status 1. On Clicking "View" Button Ticket Details page is opened. 2. All the ticket details are there. 3. For "Cancelled" Status an "Action" Button is not visible 4. Ticket Details page contains ticket activities logs
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_021**
Test Case Description	Verify that for "Closed" Status on clicking View button following things 1. On Clicking "View" Button Ticket Details page should be open. 2. All the ticket details should be there. 3. For "Closed" Status an "Action" Button should not be visible 4. Ticket Details page contains ticket activities logs
Test Steps	Tickets > Manage Tickets > "Closed" Status Ticket > Edit
Test Data	Ticket with "Closed" Status
Preconditions	Ticket should have status "Closed"
Post Conditions	User is able to take following actions on enquiry with "Closed" status 1. On Clicking "View" Button ticket Details page is opened. 2. All the ticket details are there. 3. For "Closed" status an "Action" Button is not visible 4. Ticket Details page contains ticket activities logs
Expected Result	User is able to take following actions on enquiry with "Closed" Status 1. On Clicking "View" Button ticket Details page should be open. 2. All the ticket details should be there. 3. For "Closed" status an "Action" Button should not be visible 4. Ticket Details page contains ticket activities logs
Actual Result	User is able to take following actions on enquiry with "Closed" status 1. On Clicking "View" Button ticket Details page is opened. 2. All the ticket details are there. 3. For "Closed" status an "Action" Button is not visible 4. Ticket Details page contains ticket activities logs
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_022**
Test Case Description	Verify that "Assign Ticket" form contains following

	1. Assign to List 2. Remark 3. All fields are required and should not be empty.
Test Steps	Ticket > Manage Ticket > Edit
Test Data	Ticket with "Unassigned" Status
Preconditions	Ticket should have status "Unassigned"
Post Conditions	"Assign Ticket" form contains following 1. Assign to List 2. Remark 3. All fields are required and should not be empty. If empty appropriate error message is displayed.
Expected Result	"Assign Ticket" form contains following 1. Assign to List 2. Remark 3. All fields should be required and should not be empty. If empty appropriate error message is displayed.
Actual Result	"Assign Ticket" form contains following 1. Assign to List 2. Remark 3. All fields are required and should not be empty. If empty appropriate error message is displayed.
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_023**
Test Case Description	Verify that "Change to In Progress" form contains following 1. Remark 2. Remark field is required and should not be empty.
Test Steps	Ticket > Manage Ticket > Edit
Test Data	Ticket with "Assigned" Status
Preconditions	Ticket should have status "Assigned"
Post Conditions	"Change to In Progress" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Expected Result	"Change to In Progress" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Actual Result	"Change to In Progress" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_024**
Test Case Description	Verify that "Completed" form contains following 1. Remark 2. Remark fields is required and should not be empty.

Test Steps	Ticket > Manage Ticket > Edit
Test Data	Ticket with "Change to In Progress" Status
Preconditions	Ticket should have status "Change to In Progress"
Post Conditions	"Completed" form contains following 1. Remark 2. Remark fields is required and should not be empty. If it is empty then appropriate error message is displayed.
Expected Result	"Completed" form contains following 1. Remark 2. Remark fields is required and should not be empty. If it is empty then appropriate error message is displayed.
Actual Result	"Completed" form contains following 1. Remark 2. Remark fields is required and should not be empty. If it is empty then appropriate error message is displayed.
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_025**
Test Case Description	Verify that "Closed" form contains following 1. Remark 2. Remark field is required and should not be empty.
Test Steps	Ticket > Manage Ticket > Edit
Test Data	Ticket with "Unassigned" Status
Preconditions	Ticket should have status "Unassigned"
Post Conditions	"Closed" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Expected Result	"Closed" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Actual Result	"Closed" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Status	PASS

Test Scenario ID	**TC_009**
Test Scenario Description	Verify that the "Ticket" and "Manage Ticket" are working for following Test
Module	Ticket
Sub - Module	View Tickets
Test Case ID	**TC_TIC_026**
Test Case Description	Verify that "Cancelled" form contains following 1. Remark 2. Remark field is required and should not be empty.
Test Steps	Ticket > Manage Ticket > Edit
Test Data	Ticket with "Assigned", "Unassigned" Status
Preconditions	Ticket should have status "Assigned", "Unassigned"
Post Conditions	"Cancelled" form contains following 1. Remark

	2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Expected Result	"Cancelled" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Actual Result	"Cancelled" form contains following 1. Remark 2. Remark field is required and should not be empty. If it is empty then appropriate error message is displayed.
Status	PASS

5.5 Defect report / Test Log

Defects found during test execution may be recorded either manually or by using an automated tool. The Test Defect Log is a tool for recording, analyzing, tracking, and closing defects.

Supply the "appropriate level of detail" when recording a defect. Simply put, supply enough information for the developer to find, recreate and repair the defect. Specify the screen, field, behavior and actual result that occurred. Include a screen capture, when possible.

General Information:

1. **Project Name:** The name assigned to the project.
2. **System Name:** The name of the system under test.

Defect Information:

3. **Defect ID:** A unique, sequential number used to identify the defect.
4. **Description:** A brief description of the problem.
5. **Detailed Steps:** Steps to reproduce the defect.
6. **Reported by:** The name of the person who discovered the defect.
7. **Date Reported:** The date the defect was reported.
8. **Severity:** Based on the severity (Critical, High, Medium, Low) it tells us about impact of the defect or bug in the software application

Severity Level 1 – Critical

- Any defect that compromises system security
- Loss of system functionality critical to user operations with no suitable workaround. System crash or hang that prevents further testing or operation of the complete application or a section of the application.
- Loss of functionality resulting in erroneous eligibility/enrollment determinations or communications not being sent.

Severity Level 2 - High

- A major defect in the functionality which does not result in corruption of data.
- Any defect that results in invalid authentication or authentication of an invalid end user

Severity Level 3 - Medium

- Minor functionality is not working as intended and a workaround exists but is not suitable for long term use.
- The inability of a valid user to access the system consistent with granted privileges

Severity Level 4 - Low

- Minor loss of or defect in the functionality where a long term use exists.
- Low level cosmetic issues

9. **Priority:** Priority to fix defect.
10. **Status:** A state through which a defect passes from identification to closure. (eg. New, Assigned, Open, Reopened, Verification, Closed, Failed, Deferred)
11. **Date Resolved:** Date when defect is resolved.

Test Defect Log

Project Name		Sales and Service Management	**System Name**			Yash Electricals		
Defect ID	Description	Detailed Steps to replicate.	Reported By	Date Reported	Severity	Priority	Status	Date Resolved
00001	Bug in Searching Visit	Apply All Filters data is not sorting accordingly.	Amar	05-05-22	Medium	Medium	Resolved	10-05-22
00002	User can choose Start date and End date greater than today's date	In Search Filters Start date and end date should not be greater than today's date. Currently user is able to select the greater date.	Amar	20-05-22	Medium	High	Resolved	21-05-22
00003	Pagination should be implemented.	Most tables carrying big data and it is unnecessary loading	Amar	25-05-22	Low	Low	Resolved	05-06-22
00004	Product List not updating after edit a product details	Go to Product > Select Product from List > Edit > Observe the Product List > Updated Details of Product not reflected.	Amar	01-06-22	Critical	High	Resolved	01-06-22
00005	Enquiry status not updating in enquiry activities	Enquiry > Update status of Enquiry > Enquiry activity details not updated.	Amar	15-06-22	Critical	High	Resolved	15-06-22
00006	Unable to take print of the Report	Report > See Report > Unable to take print	Amar	22-06-22	Low	Low	Resolved	25-06-22
00007	Chart Size	On Opening of the chart report pie chart size is too large.	Amar	01-07-22	Low	Low	Open	-
00008	Web Api Not working	Web-api getemployewiseenqiryreport not working.	Amar	03-07-22	Medium	Medium	Resolved	04-07-22

CHAPTER 6

LIMITATIONS OF PROPOSED SYSTEM

Limitations of the proposed system:

- Mobile Phone, Tablet or Computer is minimum requirement.
- Users should have an internet connection to connect with the system.
- Quotation for the product not available.
- Customer didn’t receive SMS/Email after every successful visit.
- Managers didn’t get notify when any action done on any enquiry or ticket.
- Customer didn’t get a notification for any action done on their ticket for a product.
- Unwanted enquiries not closed. Scheduler should be added to schedule this task.
- For Sales Department Android App not available.
- Referral marketing is not implemented.

CHAPTER 7

PROPOSED ENHANCEMENTS

Proposed Enhancements:

1) Planning for adding New Roles Sales Head and Service Head.
2) Quotation for the product should be sent to the customer when an new enquiry is added.
3) Customer should receive an SMS/Email after every successful visit.
4) Managers should be notify when any action done on any enquiry or ticket.
5) Customer should be get a notification for any action done on their ticket for a product.
6) Unwanted enquiries should be closed. Scheduler should be added to schedule this task.
7) For Sales Department there should be introduced Android app to add Enquiries/Visits remotely.
8) Referral marketing.

CHAPTER 8
CONCLUSION

In this academic project purely developed for the completion of the academic purposes and then given to the associated company. It was set out to design and construct a frontend and backend managed system which would enable weighted solution for Sales and Service management with relevance feedback will be carried out on a conventional Boolean host like AWS. In this endeavor I believe I was successful, and the resulting system has been described in these pages.

The purpose of this academic project application was to identify effective strategies for dealing with Sales and Service management in and around the locality of the clients business with limited resources this academic project was developed using the core foundations of Angular and Java Spring boot with MySQL as the database management system. Based on the progress made, it could be concluded that multiple modifications now exist which improves the overall workaround system currently in place for the client by a lot. Once finished and deployed, the project would certainly make a heavy significant beneficial impact for the client and improve their customers overall experiences with the Pre-Sale and Post-Sale.

Although technology being an everlasting improving field and the system certainly has limitations, requiring development in certain directions. Some of the limitations are of a fundamental nature, and may not be removable within the present. Nevertheless, it would appear that the system, with a little further development could form the basis of the sort of evaluation project originally envisaged for the rental management application. The application needs constant maintenance of version control and bug fixes. But it leaves room for future enhancements and structured approaches which develops the overall experience of the business and helps maintain records and generates reports for the client as well.

CHAPTER 9

BIBLIOGRAPHY

1. Stackoverflow.com
2. angular.io
3. spring.io
4. Maven repository: https://mvnrepository.com/
5. Admin LTE Free bootstrap Admin Template