**QProperty Information Management System**

Software Requirement Elicitation, Modelling and Design Document



Client: QProperty real estate agency

Vendor: Duane Powell - 12174555

Table of Contents

[1 SYSTEM REQUIREMENTS 2](#_Toc166850703)

[1.1 Functional requirements 2](#_Toc166850704)

[1.2 Non-Functional requirements 3](#_Toc166850705)

[2 SYSTEM ARCHITECTURE 4](#_Toc166850706)

[2.1 Architectural Design 4](#_Toc166850707)

[2.2 Use Case Diagram 5](#_Toc166850708)

[2.3 Class Diagram 8](#_Toc166850709)

[2.4 Sequence Diagrams 9](#_Toc166850710)

[3 DATA DESIGN 10](#_Toc166850711)

[4 INTERFACE DESIGN 11](#_Toc166850712)

5 DESIGN PATTERN 19

6 DATA VALIDATION 20

[7. REQUIRE MATRIX 20](#_Toc166850714)1

8 TEST PLAN 22

# SYSTEM REQUIREMENTS

## Functional requirements

For QPIMS, the functional requirements focus on the essential capabilities the software program must provide its users. These essential functions allow the user to manage customers, properties and maintenance, as well as providing statistical data in reporting. I have split the functional requirements into 5 main categories:

1. **Main Screen Management:** The system should provide capabilities to:
   * Add new users and provide password encryption.
   * Provide quick access to the four main Management Classes.
2. **Customer Management:** The application should allow users to:
   * Add new customer data like name, address, phone number.
   * Update existing customer information.
   * Search for customers by phone number or last name.
   * Automatic unique customer IDs autogenerated.
3. **Property Management:** Users should be able to:
   * Add property details such as address, description, year built, managing agent’s name, and property type.
   * Description information table separate from main table to allow ease of scaling.
   * Search for property by address.
   * Associate a property to customer with button click.
4. **Repair Jobs Management:** The system should provide capabilities to:

* Record and manage maintenance service.
* Record and manage repair jobs with automated repair job IDs.
* Allow for the booking, viewing, cancellation, and searching of repair jobs.
* Organize repair jobs by categories like electrical, plumbing, structural, cleaning, gardening, pest control and other.

1. **Reporting:** There should be a manager report view that displays:
   * Bar charts on repair jobs across various categories.
   * Statistical data for visual representation.
   * Minimum, maximum, and average charges for repair jobs.
2. **System Administration:**
   * Users must be able to log in with a username and password to access the application.
   * Provision for users to exit to the login screen.
   * Access help information via an About button that will be relevant as per category.

## Non-Functional requirements

Non-Functional requirements describe how the system should work by describing areas such as performance, usability, reliability, and security. Although non-functional, they still affect the overall design and development of the system. I have split these into 7 categories:

1. **Usability:** The user interface should be intuitive and easy to navigate. This will allow users to operate the system without extensive training or guidance. Use clear language, consistent design elements, and helpful prompts.
2. **Performance:** QPIS should respond quickly and efficiently, even when multiple users are accessing it simultaneously or when there is a large volume of data being processed. Load times should be minimized, and there should be no significant delays in performing tasks.
3. **Security:**  QPIS should prioritize protecting user data and preventing unauthorized access. Sensitive information like passwords and personal details must be encrypted, and appropriate security measures should be in place to reduce the risk of data breaches.
4. **Scalability:** QPIS should be built with scalability in mind. It should be able to handle an increasing number of users and a growing amount of data without sacrificing performance or functionality.
5. **Reliability:** QPIS should be available and functional when the user requires, with limited downtime. If errors or failures occur, they should be handled gracefully, with clear error messages provided to users. The system should also be able to recover quickly from any disruptions.
6. **Maintainability:** The codebase should be well-structured, documented, and easy to understand. This will allow for efficient updates, bug fixes, and the addition of new features in the future.
7. **Database Connectivity:** QPIS must establish a reliable connection to a MySQL Server 8.0 database. It should have robust mechanisms to handle connection failures, ensuring data integrity and continuity of service.

# SYSTEM ARCHITECTURE

## Architectural Design

The following package diagram illustrates the high-level architecture of the QPIMS. It showcases the interactions between the Model, View and Controller highlighting the flow of data.

A diagram of a software application

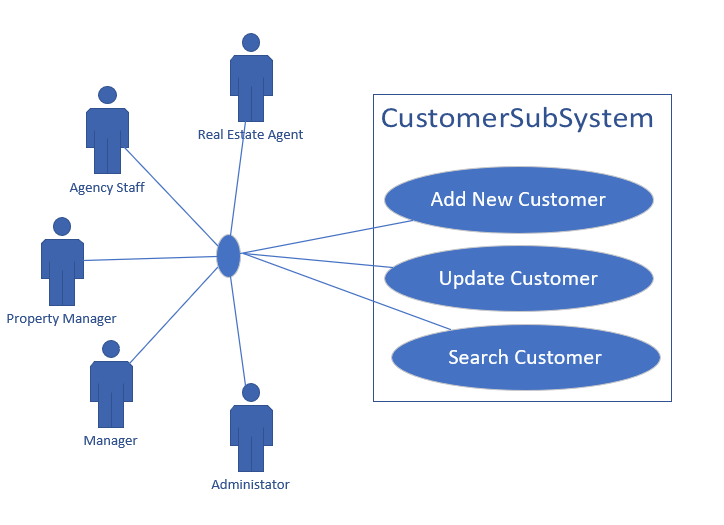
Description automatically generated

## Use Case Diagram

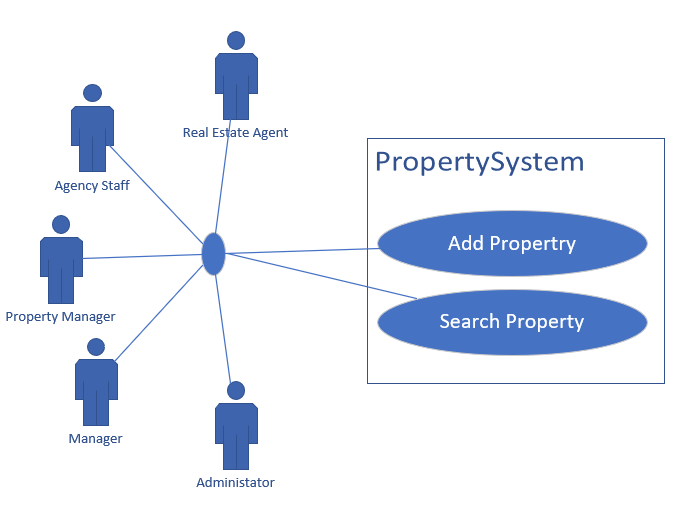
The following use case diagrams are a vital part of system design that capture the functional requirements of a system from a user perspective. I have identified six actors who will engage with QPIMS:

A diagram of a system

Description automatically generated



A diagram of a repair job system

Description automatically generatedA diagram of a company's performance

Description automatically generated

## Class Diagram

This class diagram effectively captures the core components of the QPIMS. It shows how the various entities such as users, customers, properties, repair jobs and reports are represented and interact within the system.

A computer screen shot of a diagram

Description automatically generated

## Sequence Diagrams

These sequence diagrams provide a clear and structured visualization of the interactions and data flow within QPIMS, ensuring that the system's behaviour aligns with the user requirements and functional specifications.

A diagram of a software process

Description automatically generated

Update operations.

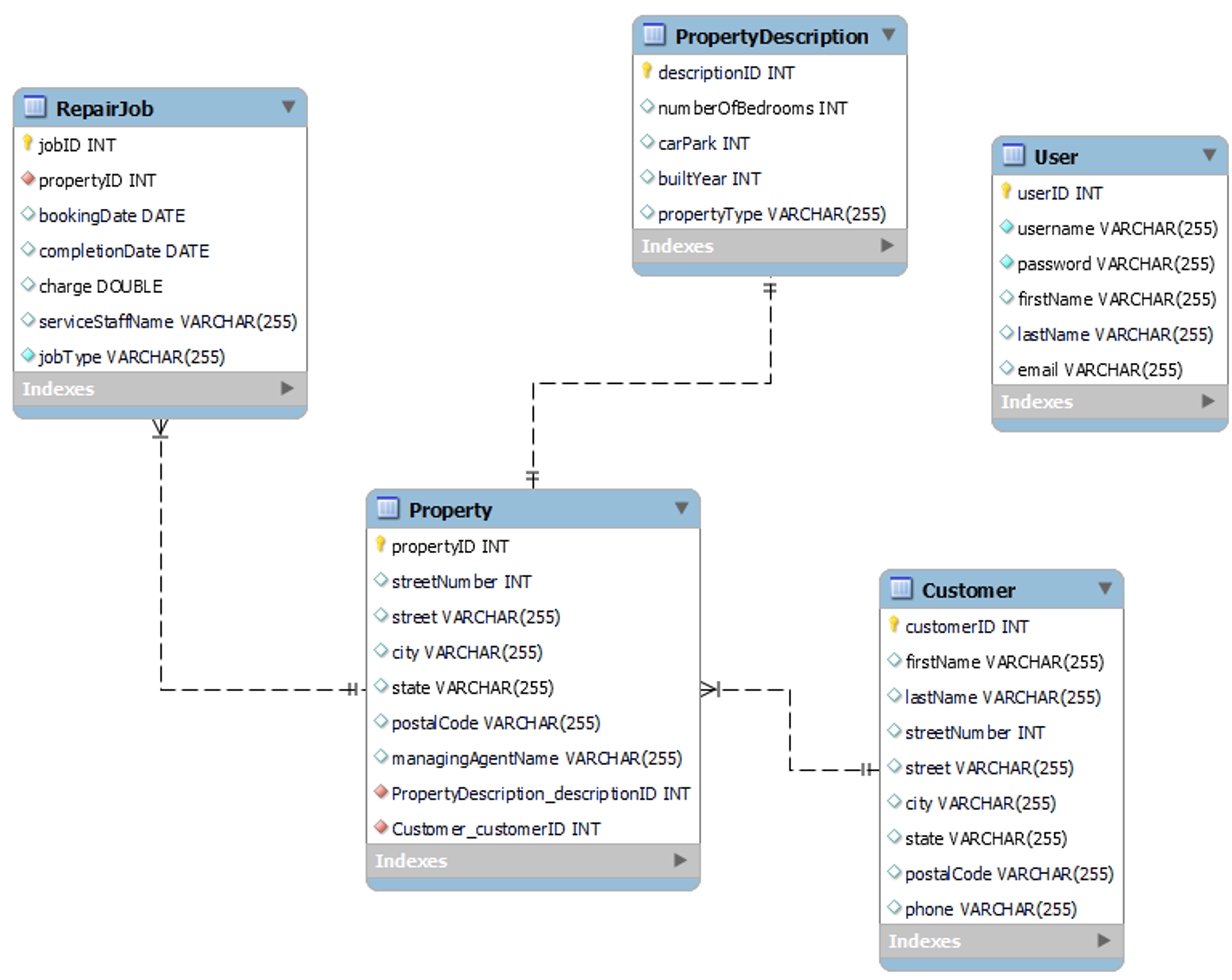
A diagram of a software process

Description automatically generated

Search operations.

# DATA DESIGN

The Entity-Relationship (ER) diagram for QPIMS provides a detailed visualization of the database structure. The diagram maps out the data structure indicating how different entities like users, customers, properties, property descriptions, and repair jobs are interrelated. This structure supports the efficient organization and management of data within the system, facilitating robust database operations and queries.



# Interface Design

Login Page

# 

Main Page

A screenshot of a computer

Description automatically generated

Customer Screen

A screenshot of a computer

Description automatically generated

Property Screen

A screenshot of a computer

Description automatically generated

Repair Screen

A screenshot of a computer

Description automatically generated

Reporting Screen

A screenshot of a computer

Description automatically generated

About Screen

A screenshot of a computer

Description automatically generated

**5. DESIGN PATTERN**

QPIMS employs two design patterns to enhance the architecture, maintainability, and improve overall system efficiency:

**5.1**. **Model-View-Controller (MVC) Pattern:** This pattern is used at the starting architecture. The separation of the three layers allows for a more modular and testable codebase. This in turn makes the system easier to maintain.

* **Model:** This layer contains the core data and business logic of QPIMS. This includes classes such as UserModel, Customer, Property, RepairJob, and Report.
* **View:** The view layer consists of the graphical interface and comprises LoginView, AboutView MainView, CustomerView, PropertyView, RepairJobView, and ReportView.
* **Controller:** The controller layer acts an intermediary between the Model and the View, processing user inputs from the View, updating the Model, and reflecting the changes back in the View. QPIMS controllers are LoginController, MainController, CustomerController, PropertyController, RepairJobController, and ReportController.

**5.2. Data Access Object (DAO) Pattern:** The DAO pattern is used to access all data via abstraction and encapsulation. It provides a clean separation between the data access logic and the business logic. This pattern does allow for a more modular and maintainable code. DAOs manage the connection with the data source to obtain and store data. Examples in use include:

* **CustomerDAO**: Manages database operations related to customer entities, such as adding, updating, and searching for customers.
* **PropertyDAO:** Handles database interactions for property-related data, including adding properties and fetching property details.
* **RepairJobDAO:** Manages the CRUD operations for repair jobs, including scheduling and cancelling jobs.

* **ReportDAO**: Could handle the retrieval of statistical data for generating reports.

**6. Data validation**

QPIMS is like any system and requires robust data validation. It is a critical aspect that ensure that data inputted by all users is accurate, complete and within acceptable parameters. Poor data validation will increase errors and lead to incorrect processing of data.

The key areas are:

1. User Input Validation:
   1. Password meets security requirements.
   2. Validate the email follows correct format.
   3. Validate phone number is in correct format.
   4. Ensure all fields have been entered.
2. Customer Data Validation:
   1. Ensure first and last names are not empty and do not contain any characters.
   2. All address fields have been entered.
   3. Post Codes are only 4 digits.
   4. Validate phone number is in correct format.
3. Property Data Validation:
   1. All address fields have been entered.
   2. Post Codes are only 4 digits.
   3. Validate property descriptions by ensuring data has been entered.
   4. Validate agent’s name.
4. Repair Job Data Validation:
   1. Booking date should not be in the past.
   2. The cost of the repair needs to be a positive number.
   3. Ensure staff name is entered.
   4. Ensure job type fits within categories already defined.

It is vital that all checks are done as the data is being inputted. The rules need to be consistent across all systems.

# 7. REQUIREMENTS MATRIX

|  |  |  |
| --- | --- | --- |
| Requirement | Button | Inputs Required |
| User Login | Login | Username, Password |
| User Registration | Register | First Name, Last Name, Username, Password, Email |
| Open Customer Screen | Customer Data | Mouse Click |
| Open Property Screen | Property Data | Mouse Click |
| Open Repair Screen | Repair Maintenace | Mouse Click |
| Open Reporting Screen | Reporting | Mouse Click |
| Add Customer | Add / Update | First Name, Last Name, Street Number, Street, City, State, Postal Code, Phone |
| Update Customer | Add / Update | Customer ID, First Name, Last Name, Street Number, Street, City, State, Postal Code, Phone |
| Search Customer | By Phone | Phone Number |
| Search Customer | By Last Name | Last Name |
| Add Property | Add /Update | Street Number, Street, City, State, Postal Code, Managing Agent Name, Property Description |
| Update Property | Add /Update | Street Number, Street, City, State, Postal Code, Managing Agent Name, Property Description |
| Search Property | Search Property | Address |
| Add Repair Job | Add | Property ID, Description, Booking Date, Completion Date, Charge, Service Staff Name, Job Type |
| Cancel Repair Job | Cancel | JobID |
| Search Repair Jobs | Search (ID) | JobID |
| Search Repair Jobs | Search (type) | Job Type |
| View Report |  | Mouse Click |
| System Help | About | Mouse Click |
| Logout | Exit | Mouse Click |

**8.Test Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Test | Input | Expected Result |
| 001 | User Login | Enter valid username and password | Login successfully |
| 002 | User Login - Invalid Credentials | Enter valid username and password | Error message is displayed. |
| 003 | |  | | --- | | User Registration |  |  | | --- | |  | | Enter all required details (first name, last name, username, password, email). | User is registered successfully. |
| 004 | User Registration - Missing Details | Leave one or more required fields empty. | Error message is displayed. |
| 005 | Add Customer | Enter all required customer details. | Customer is added successfully. |
| 006 | Update Customer | Enter updated details for an existing customer. | Customer details are updated successfully. |
| 007 | Search Customer | Enter search criteria (phone or last name). | Customer details are displayed. |
| 008 | Search Customer - No Results | Enter search criteria (phone or last name). | No results found message is displayed. |
| 009 | Click on Search Result to fill in inputs | Mouse Click – after search | Customer details are added to text fields for updating |
| 010 | Add Property | Enter all required property details. | Property is added successfully. |
| 011 | Search Property | Enter search criteria (address). | Property details are displayed. |
| 012 | |  | | --- | | Add Repair Job |  |  | | --- | |  | | Enter all required job details. | Repair job is added successfully. |
| 013 | Cancel Repair Job | Search JobID and click on table to display results in text field. | Repair job is cancelled successfully. |
| 014 | View Report | Click View Report. | Report is generated and displayed. |
| 015 | Exit Button | Mouse Click | Login screen is displayed |
| 016 | About Button | Mouse Click | System Help is displayed |
| 017 | Back Button | Mouse Click in system help | Previous screen is displayed. |