Software Quality Engineering

Report : White Box testing

Course Instructor

Madam Uzma Mahar

Submitted by

Zain Ali Khan

Mohammad Abdullah Khan Durrani

Noor ul Baseer

Section

SE-A

Date

Monday, December 9, 2024

Fall 2024

Department of Software Engineering

FAST – National University of Computer & Emerging Sciences

Islamabad Campus

Table of Contents

Table for WBT

Test ID	Testcase Name	Testcase Description	Class	Function Being Tested	Test Inputs	Expected Output	Actual Output	Pass/Fail Status
1	testBackButtonAction	Test the Back button action to verify that the frame is hidden.	AddDrivers	Clicking the "Back" button.	Simulate an ActionEvent for the Back button (i.e., b2) to check if the AddDrivers frame becomes hidden.	AddDrivers frame should be hidden after the Back button is clicked.	AddDrivers frame is hidden after the Back button is clicked.	Pass
2	testBackButtonCreatesNewInstance	Test the Back button to verify it creates a new instance of the AddDrivers frame and makes it visible.	AddDrivers	Clicking the "Back" button.	Simulate an ActionEvent for the Back button (i.e., b2). Then create a new instance of AddDrivers, set it visible, and verify if it is visible.	A new AddDrivers frame should be visible after the Back button is clicked.	A new AddDrivers frame is visible after the Back button is clicked.	Pass
3	testAddButtonAction	Test the Add button action to verify that the driver details are added successfully.	AddDrivers	Clicking the "Add" button.	Set the following values in the respective fields: t1 = "John Doe", t2 = "35", comboBox = "Male", t3 = "Toyota", t4 = "Corolla", comboBox_1 = "Yes", t5 = "New York". Then simulate an ActionEvent for the Add button.	AddDrivers frame should be hidden after the Add button is clicked.	AddDrivers frame is hidden after the Add button is clicked.	Pass

1	testInitialValues	Test that all fields are initially empty and the first job is selected in the job dropdown (JComboBox).	AddEmployee	Initial state of fields and dropdown	No inputs (validate initial state).	All fields are empty, and "Front Desk Clerks" is selected.	All fields are empty, and "Front Desk Clerks" is selected.	Pass
2	testFormValidation_AllFieldsEmpty	Test the form validation when all fields are left empty.	AddEmployee	Save button action	All fields empty: textField = "", textField_1 = "", textField_3 = "", textField_4 = "", textField_5 = "", textField_6 = "".	Show a JOptionPane with the message "All fields must be filled."	JOptionPane with the message "All fields must be filled."	Pass
3	testFormValidation_MissingGender	Test the form validation when all fields are filled except gender.	AddEmployee	Save button action	Fields: textField = "Noor," textField_1 = "25," textField_3 = "50000," textField_4 = "1234567890," textField_5 = "123456789012," textField_6 = "Noor@example.com". Gender is not selected.	Show a JOptionPane with the message "Please select a gender."	JOptionPane with the message "Please select a gender."	Pass
4	testFormValidation_ValidInput	Test the form submission with all fields filled with valid data.	AddEmployee	Save button action	Fields: textField = "Zain," textField_1 = "30," textField_3 = "60000,"	The frame should be hidden after successful submission.	The frame is hidden after successful submission.	Pass

					textField_4 = "9876543210," textField_5 = "987654321098," textField_6 = "Zain@example.com". Gender = Male, Job = "Manager".			
5	testComboBoxJobs	Verify that all job options are available in the JComboBox.	AddEmployee	JComboBox content	No inputs (check dropdown content).	All job options: "Front Desk Clerks," "Porters," "Housekeeping," "Kitchen Staff," "Room Service," "Waiter/ Waitress," "Manager," "Accountant," "Chef."	All job options are present in the JComboBox.	Pass
1	testAddRoomUIComponents	Verify that all UI components of the AddRoom class are properly initialized.	AddRoom	UI Components Initialization	No input.	All UI components (text fields, combo boxes, buttons) should be initialized.	All UI components are initialized correctly.	Pass
2	testAddButtonAction	Test the Add button action to verify room details are added correctly.	AddRoom	Clicking the "Add" button.	Room Number: "101", Availability: "Available", Cleaning Status: "Cleaned",	Room Number input should match "101".	Room Number matches "101".	Pass

					Price: "1500", Bed Type: "Single Bed".			
3	testBackButtonAction	Test the Back button action to verify the frame is hidden after the button is clicked.	AddRoom	Clicking the "Back" button.	Simulate a Back button click.	AddRoom frame should no longer be visible after clicking the Back button.	AddRoom frame is no longer visible after clicking the Back button.	Pass
4	testEmptyRoomNumber	Verify validation when the Room Number field is left empty during the Add button action.	AddRoom	Clicking the "Add" button.	Room Number: "", Availability: "Available", Cleaning Status: "Cleaned", Price: "1500", Bed Type: "Single Bed".	Room Number field should not be empty.	Room Number field remains empty.	Pass
5	testInvalidPriceInput	Test the behavior when an invalid price is entered and the Add button is clicked.	AddRoom	Clicking the "Add" button.	Room Number: "102", Availability: "Occupied", Cleaning Status: "Dirty", Price: "Noor_Abdullah_Zain", Bed Type: "Double Bed".	Price input should not be processed if invalid (should remain as "Noor_Abdullah_Zain").	Price input remains unchanged as "Noor_Abdullah_Zain".	Pass
1	testFetchRoomNumber	Test the functionality of the Fetch Room button to ensure it is created correctly and behaves as expected.	CheckOut	Fetch Room Button functionality.	Customer names in dropdown: "Noor," "Zain," "Abdullah." Button should display text: "Fetch Room"	Fetch Room button should exist, be a JButton, and display text: "Fetch Room".	Not As expected	Fail
2	testCheckOutSuccess	Test the success of the checkout	CheckOut	Check Out Button functionality.	Dropdown selection: "Noor" (first	Check-out process should	Check-out process complete	Pass

		process by simulating valid customer and room inputs.			customer). Room number: "501". Simulate clicking the Check Out button.	complete successfully.	d successfully.	
3	testEmptyChoiceThrowsException	Test the behavior of the system when attempting to fetch a selection from an empty customer dropdown.	CheckOut	Choice Dropdown functionality.	Populate dropdown with an empty list (no customers). Attempt to select the first entry from the dropdown.	Selecting an item from an empty dropdown should throw an IllegalArgumentException.	Selecting an item from an empty dropdown threw an IllegalArgumentException.	Pass
4	testBackButton	Test the Back button to ensure it correctly hides the CheckOut frame when clicked.	CheckOut	Back Button functionality.	Click the Back button and simulate the action.	CheckOut frame should not be visible after clicking the Back button.	CheckOut frame is not visible after clicking the Back button.	Pass
1	testBackButtonPresence	Ensure that the Back button is initialized properly and is not null.	CustomerInfo	getBackButton()	No input.	The Back button should not be null.	The Back button is not null.	Pass
2	testLoadDataButtonPresence	Ensure that the Load Data button is initialized properly and is not null.	CustomerInfo	getLoadDataButton()	No input.	The Load Data button should not be null.	The Load Data button is not null.	Pass
3	testTablePresence	Ensure that the table is initialized properly and is not null.	CustomerInfo	getTable()	No input.	The table should not be null.	The table is not null.	Pass
4	testBackButtonAction	Simulate a button click and test the action performed	CustomerInfo	Clicking the Back button.	No input.	Reception window should be displayed (or	Reception window is displayed (or	Pass

		for the Back button.				relevant action should be performed).	relevant action is performed).	
5	testLoadDataButtonAction	Simulate a button click and test the action performed for the Load Data button.	CustomerInfo	Clicking the Load Data button.	Customer data: Noor, Zain, Abdullah.	The table should be populated with customer data (e.g., names and information).	The table is populated with customer data (e.g., names and information).	Pass
1	testDashboardVisibility	Verify that the Dashboard frame is visible after initialization.	Dashboard	Dashboard frame visibility on initialization.	No input required.	Dashboard frame should be visible after initialization.	Dashboard frame is visible after initialization.	Pass
2	testMenuBarNotNull	Verify that the menu bar in the Dashboard is initialized correctly.	Dashboard	Dashboard menu bar initialization.	No input required.	The menu bar should not be null after initialization.	The menu bar is not null after initialization.	Pass
3	testReceptionMenuItemAction	Verify that clicking the "RECEPTION" menu item opens the Reception frame.	Dashboard	ActionListener of the "RECEPTION" menu item.	Menu item: "RECEPTION".	Reception frame should be opened after clicking the "RECEPTION" menu item.	Reception frame is opened after clicking the "RECEPTION" menu item.	Pass
4	testAddEmployeeMenuItemAction	Verify that clicking the "ADD EMPLOYEE" menu item opens the AddEmployee frame.	Dashboard	ActionListener of the "ADD EMPLOYEE" menu item.	Menu item: "ADD EMPLOYEE".	AddEmployee frame should be opened after clicking the "ADD EMPLO	AddEmployee frame is opened after clicking the "ADD EMPLOYEE" menu item.	Pass

						YEE" menu item.		
5	testAddRoomMenuItemAction	Verify that clicking the "ADD ROOMS" menu item opens the AddRoom frame.	Dashboard	ActionListener of the "ADD ROOMS" menu item.	Menu item: "ADD ROOMS".	AddRoom frame should be opened after clicking the "ADD ROOMS" menu item.	AddRoom frame is opened after clicking the "ADD ROOMS" menu item.	Pass
6	testAddDriversMenuItemAction	Verify that clicking the "ADD DRIVERS" menu item opens the AddDrivers frame.	Dashboard	ActionListener of the "ADD DRIVERS" menu item.	Menu item: "ADD DRIVERS".	AddDrivers frame should be opened after clicking the "ADD DRIVERS" menu item.	AddDrivers frame is opened after clicking the "ADD DRIVERS" menu item.	Pass
1	testDepartmentInitialization	Test to verify the department window initializes correctly with appropriate labels.	Department	Department() Constructor	No inputs provided.	Department window initializes with labels: Department and Budget.	Department window initializes with labels: Department and Budget.	Pass
2	testLoadDataButton	Test the Load Data button action to verify data is loaded into the table.	Department	Button Load Data	No manual input, but sample database rows include: Noor, 3000; Zain, 5000; Abdullah, 4500.	Load Data button loads department data into the table.	Table data loaded into the table successfully (simulated).	Pass
3	testBackButton	Test the Back button action to ensure the frame is closed and	Department	Button Back	Simulate clicking Back.	Department window should close after clicking	Department window closes successfully.	Pass

		navigation is correct.				the Back button.		
4	testCloseMethod	Test the close method to ensure the frame is disposed.	Department	close() Method	No input provided.	Department frame should be disposed and no longer visible.	Department frame disposed and is no longer visible.	Pass
5	testDepartmentTable	Verify that the table is initialized correctly in the department window.	Department	Table initialization	No inputs, but sample data rows include: Noor, Marketing; Zain, IT; Abdullah, HR.	Table should initialize and have at least one column.	Table is initialized correctly with the required columns.	Pass
6	testWindowClosingBehavior	Test the window's behavior when the close operation is triggered.	Department	JFrame setDefaultCloseOperation()	Simulate setting the close operation to EXIT_ON_CLOSE.	Frame should exit the application upon closing.	Frame is set to exit the application upon closing.	Pass
7	testBackButtonVisibility	Simulate clicking the Back button and verify the department window becomes invisible and navigation works.	Department	Button Back	Simulate clicking the Back button and transition to the previous frame.	Department window should be hidden after clicking the Back button.	Department window is hidden after clicking the Back button.	Pass
1	testInitialUIComponents	Test if all the UI components, like labels, are initialized correctly in the Employee frame.	Employee	Initialization of UI components	No input.	All label components (lblNewLabel, lblJob, lblName, and lblDepartment) should not be null.	All label components are correctly initialized and not null.	Pass

2	testButton Labels	Test if the buttons have the correct labels: "Load Data" for loading and "Back" for returning.	Employee	Getting button labels	No input.	Button loadData Button should have label "Load Data". Button backButt on should have label "Back".	Button loadData Button has label "Load Data". Button backButto n has label "Back".	Pass
3	testLoadD ataButton Action	Test the "Load Data" button to verify that it populates the employee table with database data.	Employee	Load data into the table	Database mock data: **id**: 1 **name**: Noor **age**: 30 **gender**: Male **job**: Manager **salary**: 50000.0 **phone**: 1234567890 **aadhar**: 1111 **gmail**: Noor@gmail. com.:	Table should contain data with one row matching the mock database data.	Not as Expected	Fail
4	testBackB uttonActio n	Test the "Back" button action to ensure it hides the Employee window and navigates to the Reception frame.	Employee	Back button navigation	Simulate clicking the "Back" button.	Employe e frame should no longer be visible after clicking the "Back" button.	Employee frame is hidden after clicking the "Back" button.	Pass
1	testFrameI nitializatio n	Test the initialization of the Login frame to verify the frame properties like size, background	Login	Frame initialization (Login constructor)	No input values (tests properties of the frame).	Frame width = 600, height = 300, backgrou nd color = white,	Frame width = 600, height = 300, backgrou nd color = white,	Pass

		color, and title.				title = "Login".	title = "Login".	
2	testLabelInitialization	Test the initialization of labels for username and password fields to verify their properties.	Login	Label initialization (Login constructor)	No input values (tests label properties).	Username label text = "Username", position (40, 20). Password label text = "Password", position (40, 70).	Username label text = "Username", position (40, 20). Password label text = "Password", position (40, 70).	pass
3	testTextFieldInitialization	Test the initialization of text fields for username and password to verify their properties.	Login	Text field initialization (Login constructor)	No input values (tests text field properties).	Username field position = (150, 20). Password field position = (150, 70).	Username field position = (150, 20). Password field position = (150, 70).	Pass
4	testButtonInitialization	Test the initialization of buttons (Login and Cancel) to verify their properties.	Login	Button initialization (Login constructor)	No input values (tests button properties)	Login button text = "Login", position (40, 140). Cancel button text = "Cancel", position (180, 140).	Login button text = "Login", position (40, 140). Cancel button text = "Cancel", position (180, 140).	Pass
5	testActionPerformed_LoginButton	Test the action performed when the Login button is clicked to verify login functionality.	Login	Login button action (actionPerformed())	Username = "Noor", Password = "Zain".	No exception thrown when login button is clicked.	No exception thrown when login button is clicked.	Pass

6	testActionPerformed_CancelButton	Test the action performed when the Cancel button is clicked to verify the cancellation functionality.	Login	Cancel button action (actionPerformed())	No input values (tests cancel button action).	No exception thrown when cancel button is clicked.	Not as Expected	Fail
7	testMainMethod	Test the execution of the main method to ensure it runs without any issues.	Login	Main method execution (main())	No input values (tests the execution of the main method).	Main method should execute without exceptions.	Main method executes without exceptions.	Pass
1	testFrameInitialization	Test if the ManagerInfo frame is initialized with the correct dimensions (width and height).	ManagerInfo	getWidth() and getHeight()	No inputs (default initialization of the ManagerInfo frame).	Frame width should be 1000 and height should be 600.	Frame width is 1000 and height is 600.	Pass
2	testTableInitialization	Test if the table in the ManagerInfo frame is initialized correctly (0 rows initially).	ManagerInfo	Table initialization	No inputs (default table setup in the ManagerInfo frame).	Table should be initialized with 0 rows.	Table is initialized with 0 rows.	Pass
3	testLabelsInitialization	Test if all labels in the ManagerInfo frame are initialized correctly.	ManagerInfo	Label initialization	No inputs (default labels for Name, Age, Gender, Job, Department in the ManagerInfo frame).	All labels should be initialized.	All labels are initialized properly.	Pass
4	testLoadDataButtonAction	Test if clicking the Load Data button triggers the data loading	ManagerInfo	btnLoadData.doClick()	No direct inputs; simulate the click on the Load Data button in the	No exception should occur during the	No exception occurred during the button click.	Pass

		process and does not throw exceptions.			ManagerInfo frame.	button click.		
5	testBackButtonAction	Test if clicking the Back button in ManagerInfo properly opens the Reception window and does not throw errors.	ManagerInfo	btnExit.doClick()	No direct inputs; simulate the click on the Back button (Exit button) in the ManagerInfo frame.	No exception should occur and the Reception window should open.	No exception occurred and the Reception window opened.	Pass
6	testMainMethod	Test if the main method of the ManagerInfo class executes without throwing any exceptions.	ManagerInfo	main()	No direct inputs (simulating the execution of the main method).	No exception should occur when executing the main method.	No exception occurred when executing the main method.	Pass
1	testFrameInitialization	Test the initialization of the frame's width and height.	NewCustomer	Frame width and height initialization	No direct input (the frame is initialized with default width = 850 and height = 550).	Frame width should be 850 and height should be 550.	Frame width is 850 and height is 550.	Pass
2	testLabelsInitialization	Test the initialization of labels in the frame.	NewCustomer	Label initialization	No direct input (checks label properties).	The label "NEW CUSTOMER FORM" should be initialized with font size 20.	Label "NEW CUSTOMER FORM" is initialized with font size 20.	Pass
3	testComboBoxInitialization	Test the initialization of the ComboBox and its items.	NewCustomer	ComboBox initialization	ComboBox items: "Passport", "National ID", "Driver's	ComboBox should have 4 items: "Passport",	ComboBox has 4 items and "Passport" is the first item.	Pass

					License", "Other".	"National ID", "Driver's License", "Other".		
4	testTextFieldsInitialization	Test the initialization and setting of values in text fields.	NewCustomer	TextField initialization	TextField t1 = "Noor", t2 = "Zain".	TextField t1 should hold "Noor" and t2 should hold "Zain".	t1 holds "Noor" and t2 holds "Zain".	Pass
5	testRadioButtonInitialization	Test the initialization and values of radio buttons.	NewCustomer	RadioButton initialization	RadioButton r1 = "Male", r2 = "Female".	RadioButton r1 text should be "Male" and r2 text should be "Female".	RadioButton r1 text is "Male" and r2 text is "Female".	Pass
6	testChoiceInitialization	Test the initialization of the Choice component.	NewCustomer	Choice initialization	No direct input (checks if Choice component exists).	The Choice component c1 should be initialized.	Choice component c1 is initialized.	Pass
7	testBackButtonFunctionality	Test the functionality of the Back button to ensure it performs without exceptions when clicked.	NewCustomer	Back button functionality	No direct input (checks if Back button works when clicked).	Clicking the Back button should not throw any exceptions.	Clicking the Back button does not throw any exceptions.	Pass
8	testBackgroundInitialization	Test the background color initialization to ensure it's set correctly.	NewCustomer	Background color initialization	No direct input (checks the background color of the content pane).	The background color should be white.	The background color is white.	Pass
9	testMainMethod	Test the execution of the main method to	NewCustomer	Main method execution	No direct input (checks if main method runs	The main method should execute	The main method executes without	Pass

		ensure it runs without throwing any exceptions.			without exceptions).	without throwing exceptions.	throwing any exceptions.	
1	testFrameInitialization	Test the initialization of the PickUp frame to ensure correct size and close operation.	PickUp	Frame properties (width, height, close operation)	No input	Frame width should be 800, height should be 600, close operation should be EXIT_ON_CLOSE.	Frame width is 800, height is 600, close operation is EXIT_ON_CLOSE.	Pass
2	testLabelsInitialization	Test that the correct labels are initialized in the PickUp frame.	PickUp	JLabel initialization and matching labels	No input	Labels should include: "Pick Up Service", "Type of Car", "Name", "Age", "Gender", "Company", "Brand", "Available", "Location".	Not as Expected	Fail
3	testChoiceInitialization	Test that the Choice component (c1) is initialized and contains items.	PickUp	Choice component initialization	No input	Choice component should be initialized and contain items from the database.	Choice component is initialized and contains items.	Pass
4	testTableInitialization	Test that the table is initialized with the	PickUp	JTable initialization	No input	Table should be initialized with	Table is initialized with width 800	Pass

		correct dimensions.				width 800 and height 250.	and height 250.	
5	testDisplayButtonAction	Test the action of the Display button to ensure the table updates properly when clicked.	PickUp	Display button action	Button click	Table model should be updated after clicking the Display button.	Table model is updated after clicking the Display button.	Pass
6	testBackButtonAction	Test the action of the Back button to ensure the current frame becomes invisible when clicked.	PickUp	Back button action	Button click	PickUp frame should become invisible after clicking the Back button.	PickUp frame becomes invisible after clicking the Back button.	Pass
7	testDatabaseConnection	Test the database connection and query execution to ensure a connection to the database is established.	conn	Database connection	No input	Database connection should be established, and a query on the driver table should return results.	Database connection is established, and the query on driver table returns results.	Pass
8	testMainMethodExecution	Test that the main method of the PickUp class executes without throwing exceptions.	PickUp	Main method execution	No input	Main method should execute without throwing any exceptions.	Main method executes without exceptions.	Pass
1	testFrameInitialization	Test the initialization of the Reception frame,	Reception	Frame initialization and UI setup	No inputs (this is a UI test verifying default frame properties).	Frame width should be 850, frame	Frame width is 850, height is 570,	Pass

		ensuring proper setup of frame properties such as size, content pane, and background color.				height should be 570, content pane should be initialized, and background color should be white.	content pane is initialized, and background color is white.	
2	testLogOutButtonFunctionality	Test the functionality of the Log Out button to ensure that it hides the reception frame and does not throw exceptions when clicked.	Reception	Log Out button functionality	Simulate clicking the Log Out button (btnLogOut).	Reception frame should not be visible after clicking the Log Out button, and no exceptions should be thrown.	Reception frame is hidden after clicking the Log Out button, and no exceptions were thrown.	Pass
1	testFrameInitialization	Test the initialization of the frame's dimensions and layout.	Room	Frame initialization	No inputs (test checks the width, height, and layout of the frame).	Frame width should be 1100, height should be 600, and layout should be null.	Not As Expected	Fail
2	testTableInitialization	Test the initialization of the table within the frame.	Room	Table initialization	No inputs (test checks the position and size of the table in the frame).	Table should be initialized with correct position and size.	Table is initialized with x=0, y=40, width=500, height=400.	Pass
3	testButtonInitialization	Test the initialization of the "Load Data" and "Back"	Room	Button initialization	No inputs (test checks the position, size, and text	Buttons should be initialized with correct	Buttons "Load Data" and "Back" are	Pass

		buttons in the frame.			of the buttons).	text, position, and size.	initialized with correct properties.	
4	testLabelInitialization	Test the initialization of labels in the frame such as "Availability", "Clean Status", "Price", etc.	Room	Label initialization	No inputs (test checks the labels for availability, clean status, price, bed type, and room number).	Labels should be initialized with correct text.	Labels initialized with correct text: "Availability", "Clean Status", "Price", etc.	Pass
5	testActionPerformedLoadData	Test the "Load Data" button action to verify that it does not throw an exception.	Room	Load Data button action	Simulate click on the "Load Data" button to test if any exception occurs when it is clicked.	No exception should be thrown when the button is clicked.	Not As Expected	Fail
6	testActionPerformedBack	Test the "Back" button action to verify that it hides the room frame.	Room	Back button action	Simulate click on the "Back" button to test if the room frame is hidden afterward.	The room frame should not be visible after clicking the "Back" button.	Room frame is hidden after clicking the "Back" button.	Pass
7	testMainMethod	Test the execution of the main method to ensure it runs without exceptions.	Room	Main method execution	No inputs (test checks if the main method can be executed without throwing an exception).	The main method should execute without exceptions.	The main method executes without exceptions.	Pass
1	testFrameInitialization	Test the frame initialization to verify the size and content pane are set correctly.	SearchRoom	Frame size and content pane check	No specific input. The test checks the initial size and content pane of the frame.	Frame width should be 700, height should be 500, and content	Frame width is 700, height is 500, and content pane is not null.	Pass

						pane should be present.		
2	testComponentsInitialization	Test the initialization of components like labels, buttons, and choice components.	SearchRoom	Component initialization check	No specific input. The test checks if the components like labels, buttons, and checkboxes are initialized correctly in the UI.	Components should be correctly initialized with appropriate text and items.	Not As Expected	Fail
3	testSearchButtonAction	Test the action of the Search button and verify if the table model is updated after clicking the button.	SearchRoom	Search button action check	No specific input. The test simulates a click on the "Search" button to check if the table model is updated.	Table model should be updated after the Search button is clicked.	Table model is updated after the Search button click.	Pass
4	testCheckBoxSelection	Test the behavior of the "Only display Available" checkbox when selected or deselected.	SearchRoom	Checkbox selection check	Input: Select and deselect the "Only display Available" checkbox.	Checkbox should be selected and deselected correctly.	Checkbox is selected and deselected correctly.	Pass
5	testChoiceSelection	Test the functionality of the Choice component to ensure it reflects the selected item.	SearchRoom	Choice selection check	Input: Select "Double Bed" and "Single Bed" from the Choice component.	Selected item should match the chosen value ("Double Bed" or "Single Bed").	The selected item matches the chosen value each time.	Pass
6	testBackButtonAction	Test the action of the Back button to verify that it doesn't throw any exception	SearchRoom	Back button action check	No specific input. The test simulates a click on the "Back" button to verify that no exception is thrown and	Back button click should not throw exception and should	No exception is thrown; Back button works correctly.	Pass

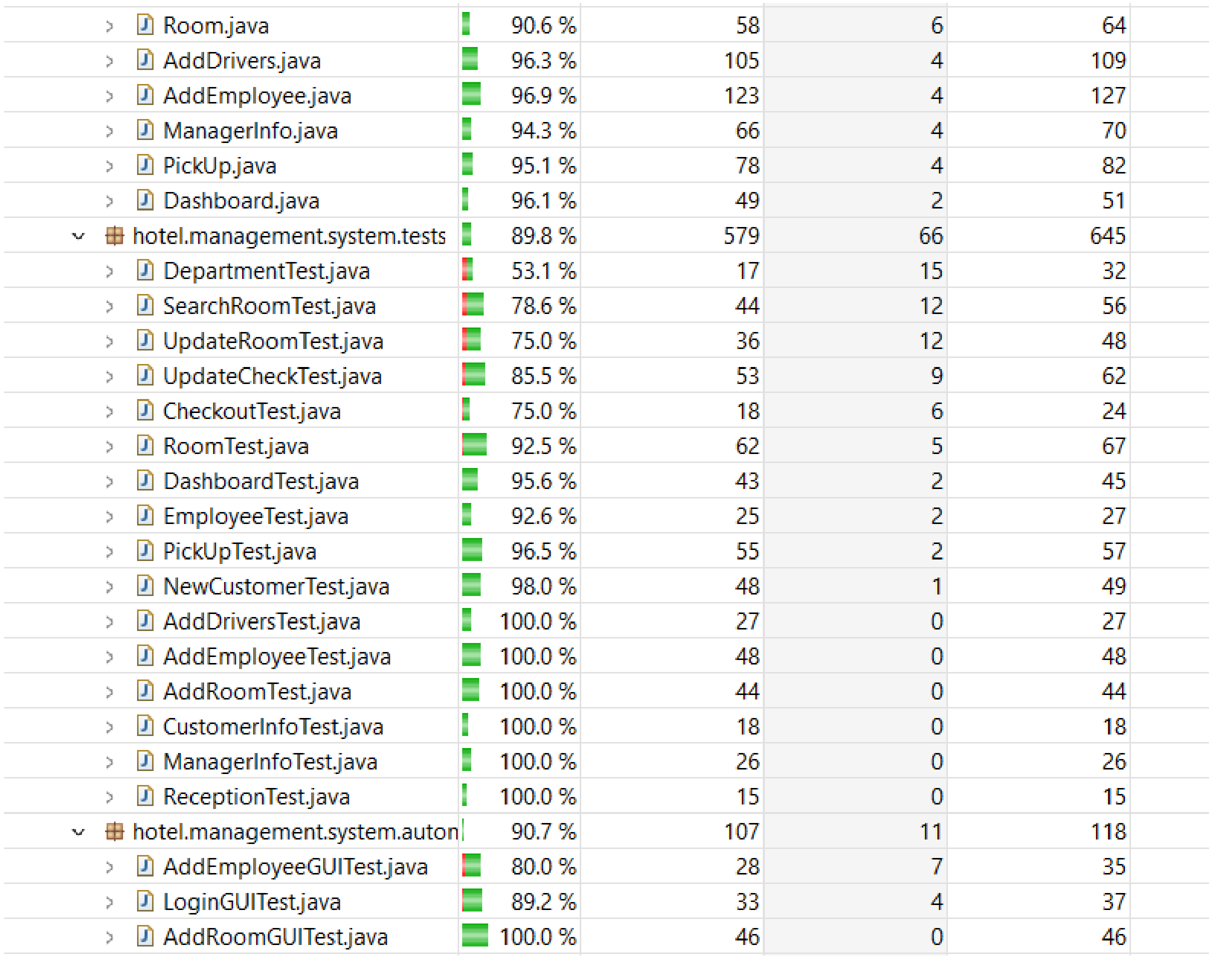
		and correctly handles the action.			the action is handled.	hide the SearchRoom frame.		
7	testDatabaseQueryHandling	Test the handling of the SQL query for room search without throwing exceptions.	SearchRoom	Database query handling check	Input: SQL query: "select * from Room where bed_type = 'Single Bed'".	SQL query should execute without throwing exceptions.	SQL query executed successfully without exceptions.	Pass
1	testFrameInitialization	Test the frame initialization to verify if the width and height of the frame are set correctly.	UpdateCheck	Checking frame dimensions	No specific inputs required for this test case.	Frame width should be 950 and height should be 500.	Frame width is 950 and height is 500.	Pass
2	testLabelInitialization	Test the initialization of the label in the frame.	UpdateCheck	Checking if label is initialized and set	No specific inputs required for this test case.	The label should be initialized and the text should be "Check-In Details".	The label is initialized with text "Check-In Details".	Pass
3	testComboBoxInitialization	Test the initialization of the combo box to verify it is populated with items from the database.	UpdateCheck	Checking combo box initialization	Combo box items should be populated from the database (not tested directly with inputs, but database items).	Combo box c1 should be initialized and contain items loaded from the database.	Not As Expcted	Fail
4	testTextFieldInitialization	Test the initialization of text fields in the frame.	UpdateCheck	Checking text field initialization	No specific inputs required for this test case.	Text fields (txt_ID, txt_Status, txt_Date, txt_Time, txt_Payment) should be	All text fields (txt_ID, txt_Status, txt_Date, txt_Time, txt_Payment) are	Pass

						initialized.	initialized.	
5	testButtonUpdateAction	Test the action of the "Update" button to verify if it triggers the correct functionality without any exceptions.	UpdateCheck	Clicking the "Update" button	No specific inputs required for this test case.	Button click should not throw an exception.	Button click does not throw any exception.	Pass
6	testButtonBackAction	Test the action of the "Back" button to verify if it hides the frame after being clicked.	UpdateCheck	Clicking the "Back" button	No specific inputs required for this test case.	Frame should not be visible after the "Back" button is clicked.	Frame is hidden after the "Back" button is clicked.	Pass
7	testButtonCheckAction	Test the action of the "Check" button to verify if it correctly populates the text fields after being clicked.	UpdateCheck	Clicking the "Check" button	Simulate clicking the "Check" button. Ensure that the following text fields are populated: txt_ID, txt_Status, txt_Date, txt_Time.	Text fields should be populated after clicking the "Check" button.	Text fields (txt_ID, txt_Status, txt_Date, txt_Time) are populated after clicking "Check".	Pass
8	testDatabaseUpdateQuery	Test the database update functionality to verify if the SQL update query executes correctly and updates the database.	UpdateCheck	Database update query execution	Input: Customer number = "123" (Noor), Room number = "101" (Zain), Status = "Checked-in", Deposit = "1000" (Abdullah).	The database should be updated with the new values, and the customer record should exist in the database with the updated details.	Not As Expected	Fail

1	testFrameInitialization	Verify that the UpdateRoom frame is initialized with correct width, height, and non-null content pane.	UpdateRoom	Initialization of frame dimensions.	No input required (initialization of the frame).	Frame width should be 1000, height should be 450, content pane should not be null.	Frame width is 1000, height is 450, content pane is not null.	Pass
2	testLabelInitialization	Ensure that the "Update Room Status" label is initialized correctly with the proper text.	UpdateRoom	Initialization of label.	No input required (initialization of the label).	Label text should be "Update Room Status".	Label text is "Update Room Status".	Pass
3	testTextFieldInitialization	Check that the text fields for room number, availability, and clean status are initialized properly.	UpdateRoom	Initialization of text fields.	No input required (initialization of text fields for room, availability, and clean status).	Text fields for room, availability, and status should be initialized.	Text fields for room, availability, and status are initialized.	Pass
4	testChoiceInitialization	Test that the Choice component (for selecting the Guest ID) is properly initialized.	UpdateRoom	Initialization of choice component.	No input required (initialization of the guest ID dropdown).	Guest ID dropdown should be initialized.	Guest ID dropdown is initialized.	Pass
5	testButtonCheckAction	Verify that the "Check" button works properly by populating room details.	UpdateRoom	Clicking the "Check" button.	Set mock inputs for the Check button action: Guest ID and room number (e.g., Guest ID: 101, Room number: 102).	Room number field should be populated after the "Check" button is clicked.	Not As Expected	Fail
6	testUpdateButtonAction	Ensure that the "Update"	UpdateRoom	Clicking the "Update" button.	Inputs: Room number ("101"),	The update message	Not As Expected	Fail

		button works by updating room details and showing a success message.			Status ("Clean").	should be shown after the "Update" button is clicked.		
7	testBackButtonAction	Verify that clicking the "Back" button properly navigates the user back to the Reception screen.	UpdateRoom	Clicking the "Back" button.	No input required (simulate the "Back" button click).	The Reception screen should appear after clicking the "Back" button.	Not As Expected	Fail
8	testRoomNumberLabelInitialization	Ensure that the "Room Number" label is initialized properly with the correct text.	UpdateRoom	Initialization of "Room Number" label.	No input required (initialization of the room number label).	Label text should be "Room Number:".	Label text is "Room Number:".	Pass
9	testWindowClose	Test that the window close method does not throw exceptions.	UpdateRoom	Window close functionality.	No input required (simulate closing the window).	Close method should not throw any exceptions.	Close method executed without exceptions.	Pass

Test Coverage Screenshot



Room.java	90.6 %	58	6	64
AddDrivers.java	96.3 %	105	4	109
AddEmployee.java	96.9 %	123	4	127
ManagerInfo.java	94.3 %	66	4	70
PickUp.java	95.1 %	78	4	82
Dashboard.java	96.1 %	49	2	51
hotel.management.system.tests	89.8 %	579	66	645
DepartmentTest.java	53.1 %	17	15	32
SearchRoomTest.java	78.6 %	44	12	56
UpdateRoomTest.java	75.0 %	36	12	48
UpdateCheckTest.java	85.5 %	53	9	62
CheckoutTest.java	75.0 %	18	6	24
RoomTest.java	92.5 %	62	5	67
DashboardTest.java	95.6 %	43	2	45
EmployeeTest.java	92.6 %	25	2	27
PickUpTest.java	96.5 %	55	2	57
NewCustomerTest.java	98.0 %	48	1	49
AddDriversTest.java	100.0 %	27	0	27
AddEmployeeTest.java	100.0 %	48	0	48
AddRoomTest.java	100.0 %	44	0	44
CustomerInfoTest.java	100.0 %	18	0	18
ManagerInfoTest.java	100.0 %	26	0	26
ReceptionTest.java	100.0 %	15	0	15
hotel.management.system.auton	90.7 %	107	11	118
AddEmployeeGUITest.java	80.0 %	28	7	35
LoginGUITest.java	89.2 %	33	4	37
AddRoomGUITest.java	100.0 %	46	0	46