GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT044-3-1-IOOP

INTRODUCTION TO OBJECT ORIENTED PROGRAMMING

Lecturer Name: MARY TING

Intake Code: APU1F2403CS(CYB)

Group Number: Group 26

Group Members:

NO.	NAME	TP NO.
1.	Lee Yi Chern (L)	TP081340
2.	Ho Kun Yuan	TP080482
3.	Hwang XiaoShun	TP077723
4.	Ibraheem Shiraz Omar	TP074191

Date Assigned: 6 September 2024

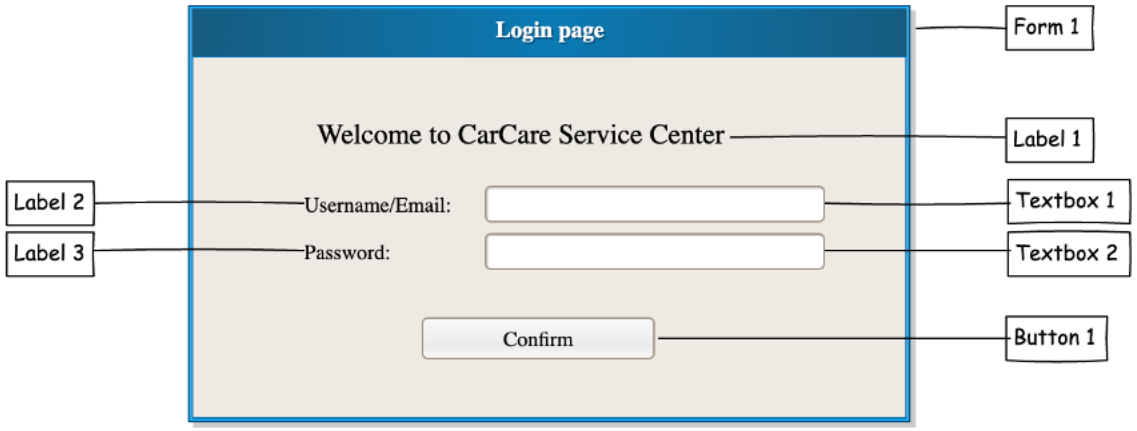
Date Completed: 17 November 2024

2. Table Content

1. Front page ---------- 1
2. Table Content ---------- 2
3. Storyboard & Test plan
   3.1 Admin Storyboard & Test plan ---------- 3
   3.2 Receptionist Storyboard & Test plan ---------- 12
   3.3 Customer Storyboard & Test plan ---------- 19
   3.4 Mechanic Storyboard & Test plan ---------- 25
4. Use-case Diagram ---------- 34
5. Class Diagram
   5.1 Admin Class Diagram ---------- 35
   5.2 Receptionist Class Diagram ---------- 36
   5.3 Customer Class Diagram ---------- 37
   5.4 Mechanic Class Diagram ---------- 38
6. Code Explanations
   6.1 Admin Code Explanations ---------- 39
   6.2 Receptionist Code Explanations ---------- 42
   6.3 Customer Code Explanations ---------- 45
   6.4 Mechanic Code Explanations ---------- 47
7. Conclusion ---------- 48
8. References ---------- 49
9. Workload Matrix ---------- 50

3. Storyboard

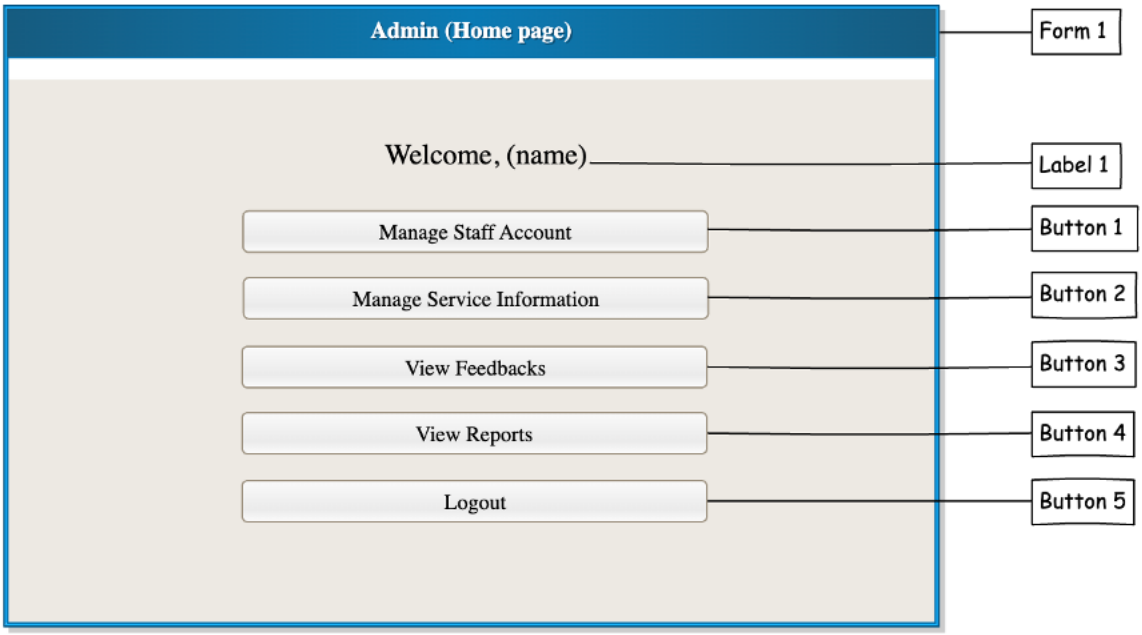
3.1 Admin Storyboard & Test plan



Control	Control Name	Description
Form 1	formLogin	Form of Login page
Label 1	lblWelcome	To welcome the user to this software
Label 2	lblUsername	To label the related controls to the right
Label 3	lblPassword	
Textbox 1	txtUsername	To insert the username
Textbox 2	txtPassword	To insert the password
Button 1	btnConfirm	To login to the remaining home page

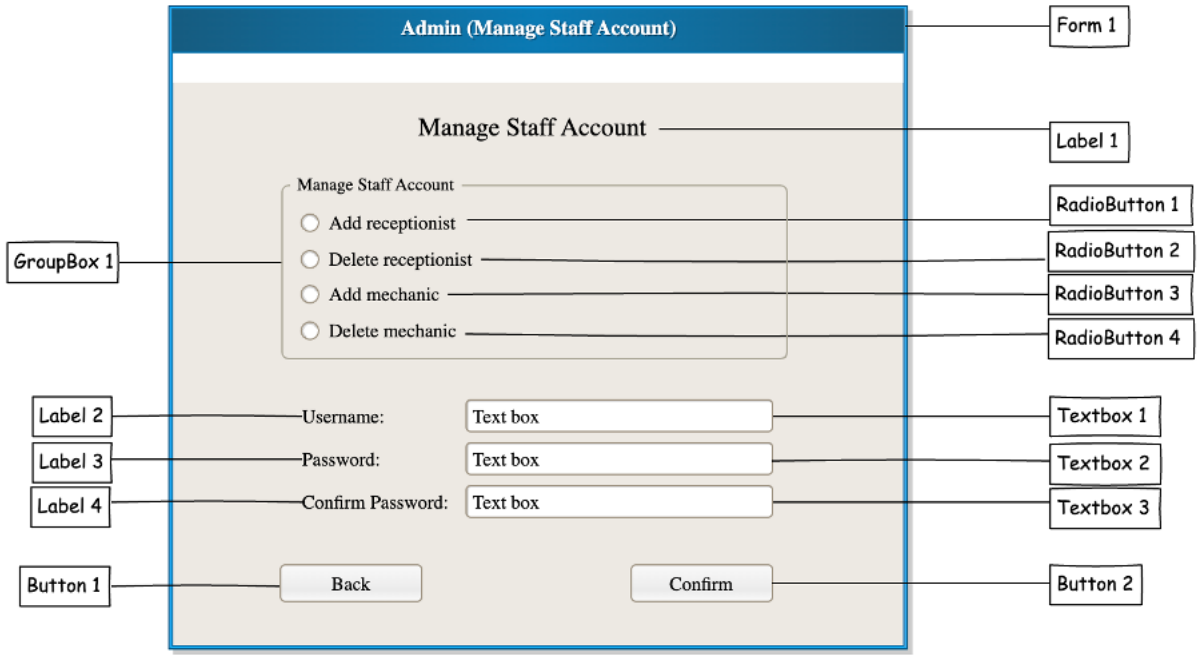
Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1.	Login – Admin	To test admin can login to its page with correct username and password	Admin can login to its page successfully	Admin login successfully, admin home page displayed.	
2.	Login – Receptionist	To test receptionist can login to its page with correct username and password	Receptionist can login to its page successfully	Receptionistlogin successfully, admin home page displayed.	
3.	Login – Mechanic	To test mechanic can login to its page with correct username and password	Mechanic can login to its page successfully	Admin login successfully, admin home page displayed.	

4.	Login – Customer	To test customer can login to its page with correct username and password	Customer can login to its page successfully	Customer login successfully, admin home page displayed.	
5.	Login – Wrong username/email	To test how if user key in the wrong username	Users will be required to key its username/email and password again.	Message box shows "Incorrect username/password! Please try again.	
6.	Login – Wrong password	To test how if user key in the wrong password	Users will be required to key its username/email and password again.	Message box shows "Incorrect username/password! Please try again.	



Control	Control Name	Description
Form 1	formAdHome	Form of Admin Home page
Label 1	lblWelcome	To welcome the admin
Button 1	btnManageStaffAcc	To move to the form of Admin Manage Staff Account
Button 2	btnManageService	To move to the form of Admin Manage Service Information
Button 3	btnFeedback	To move to the form of Admin View Feedback
Button 4	btnReport	To move to the form of Admin View Reports
Button 5	btnLogout	To logout and move to the form of Login Page

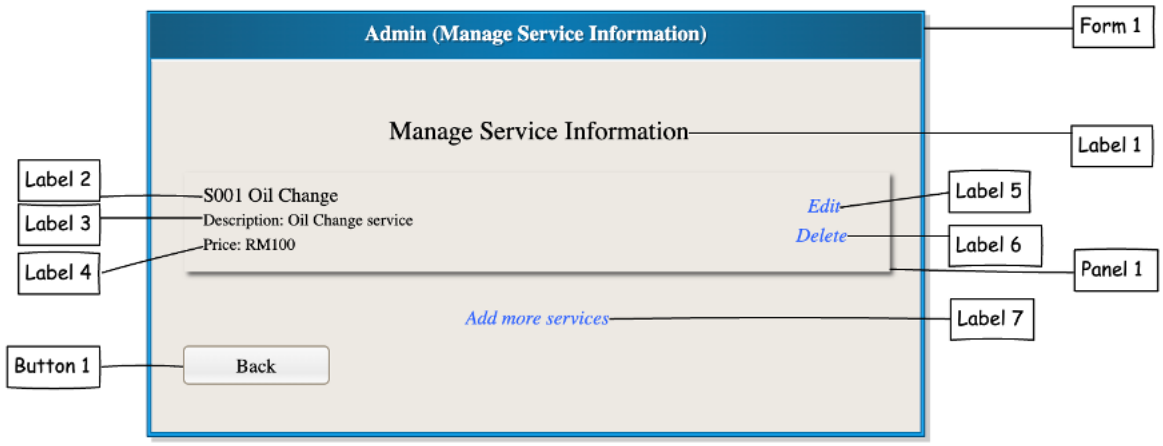
Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1.	Enter Manage Staff Account page	To test whether user can enter Manage Staff Account page	User can enter Manage Staff Account page	Enter Manage Staff Account page successfully	
2.	Enter Manage Service Information page	To test whether user can enter Manage Service Information page	User can enter Manage Service Information page	Enter Manage Service Information page successfully	
3.	Enter View Feedback page	To test whether user can enter View Feedback page	User can enter View Feedback page	Enter View Feedback page successfully	
4.	Enter View Reports page	To test whether user can enter View Reports page	User can enter View Reports page	Enter View Report page successfully	
5.	Logout	To test whether user can logout its account	User can logout its account successfully	Logout successfully. Login page displayed.	



Control	Control Name	Description
Form 1	formAdManageStaff	Form of Admin Manage Staff Account
Label 1	lblManageStaff	To label Manage Staff Account page
Label 2	Label 2	To label the related controls to the right
Label 3	Label 3	
Label 4	Label 4	
GroupBox 1	grpboxChoices	To group all choices
RadioButton 1	radbtnAddRecept	To select action, add receptionist
RadioButton 2	radbtnDelRecept	To select action, remove receptionist
RadioButton 3	radbrnAddMechanic	To select action, add mechanic
RadioButton 4	radbtnDelMechanic	To select action, remove mechanic
Textbox 1	txtUsername	To insert the username/email
Textbox 2	txtPass	To insert the password
Textbox 3	txtConfirmPass	To insert the confirmation password
Button 1	btnBack	To go back to the previous page
Button 2	btnConfirm	To confirm action

Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1.	Add receptionist	To test whether admin could add receptionist	Success to add receptionist	Add receptionist successfully. Message box shows	

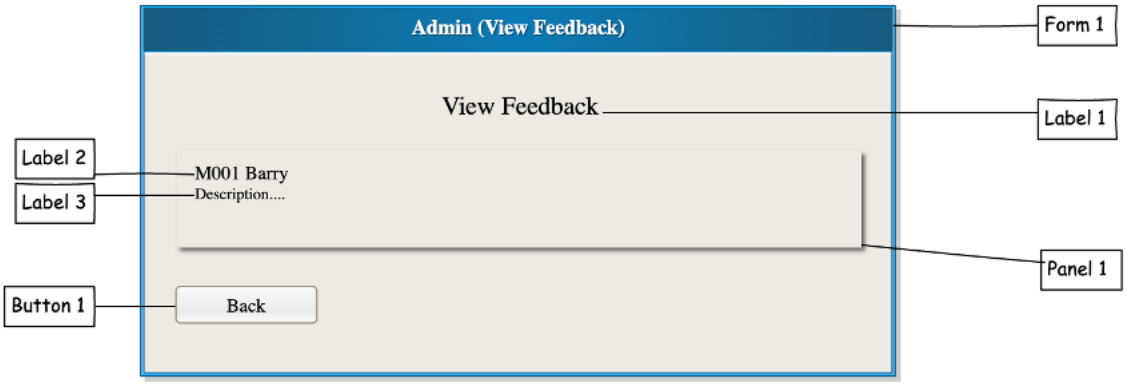
				"Receptionist added successfully".	
2.	Delete receptionist	To test whether admin could delete receptionist	Success to delete receptionist	Delete receptionist successfully. Message box shows "Receptionist deleted successfully"	
3.	Add mechanic	To test whether admin could add mechanic	Success to add mechanic	Add mechanic successfully. Message box shows "Mechanic added successfully".	
4.	Delete mechanic	To test whether admin could delete mechanic	Success to delete mechanic	Delete mechanic successfully. Message box shows "Mechanic deleted successfully"	
5.	Add receptionist with same username/email	To test how if adding a receptionist with same username/email	Message box show "Invalid username/email, please try again."	Message box shows "Error: Username {input} already exist".	
6.	Add mechanic with same username/email	To test how if adding a mechanic with same username/email	Message box show "Invalid username/email, please try again."	Message box shows "Error: Username {input} already exist".	
7.	Delete receptionist with invalid username/username	To test how if deleting a receptionist with invalid username/email	Message box show "Invalid username/email, please try again."	Message box shows "Receptionist not found".	
8.	Delete mechanic with invalid username/username	To test how if deleting a mechanic with invalid username/email	Message box show "Invalid username/email, please try again."	Message box shows "Mechanic not found".	
9.	Back to home	To test back to home	Back to home page successfully	Back to home page successfully.	



Control	Control Name	Description
Form 1	formAdManageService	Form of Admin Manage Service Information
Label 1	lblManageService	To label Manage Service Information
Label 2	lblService1	To label service 1 code and name
Label 3	lblService1Description	To label information of service 1
Label 4	lblService1Price	To label service 1 price
Label 5	lblService1Edit	To edit service 1
Label 6	lblService1Delete	To delete service 1
Label 7	lblAddServices	To add more service
Panel 1	panelService1	To group information of service 1
Button 1	btnBack	To go back to the previous page

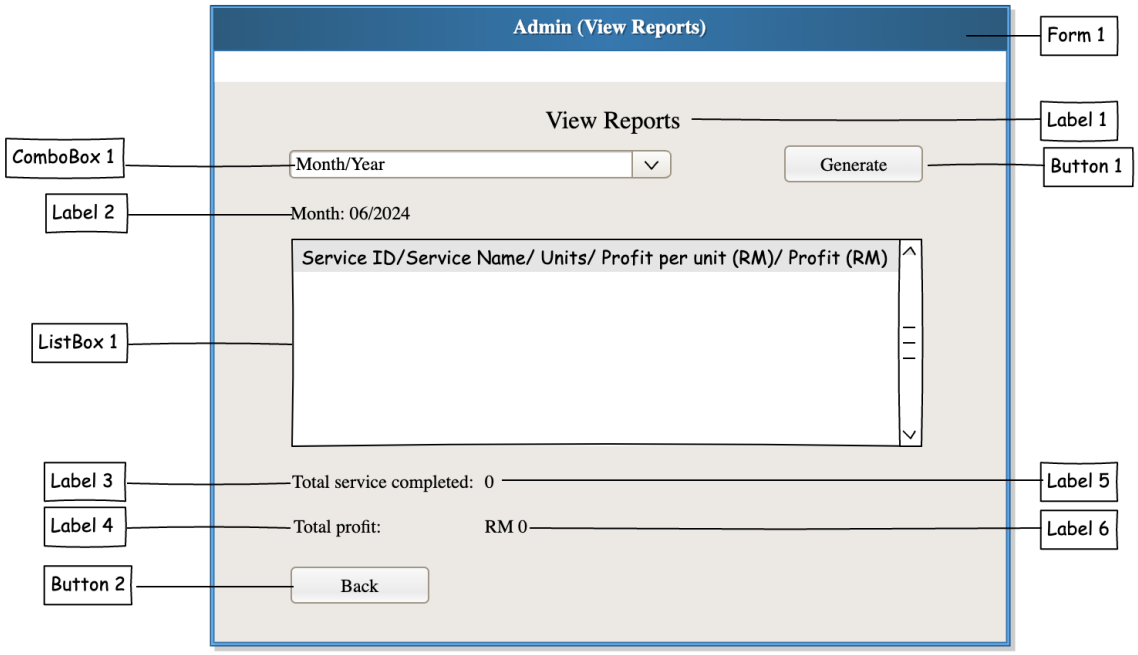
Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1.	View service information	To test whether service information been displayed properly	Service information displayed properly	Service information displayed properly	*Able to sort by price, name and type
2.	Edit service information	To test edit service information	Service information been edited	Able to edit service information.	*Edit service form displayed
3.	Delete service information	To test delete service information	Service been deleted	Able to delete service.	
4.	Add more services	To test adding more services	Service been added	Able to add service.	*Add service

					form displayed
5.	Add more services with same service code	To test how if adding services with same service code	Message box show “Invalid service code, please try again.”	Message box shows “Service ID {input} already exist”.	
6.	Back to home	To test back to home	Back to home page successfully	Back to home page successfully.	



Control	Control Name	Description
Form 1	formAdFeedback	Form of Admin View Feedback
Label 1	lblViewFeedback	To label View Feedback
Label 2	lblFeedback1	To label service 1 code and name
Label 3	lblFeedback1Description	To label information of service 1
Panel 1	panelFeedback1	To group information of service 1
Button 1	btnBack	To go back to the previous page

Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1.	View all feedback	To test whether feedback displayed properly	All feedback are displayed properly	All feedback are displayed properly	*User able to sort by date
2.	Back to home	To test back to home	Back to home page successfully	Back to home page successfully	

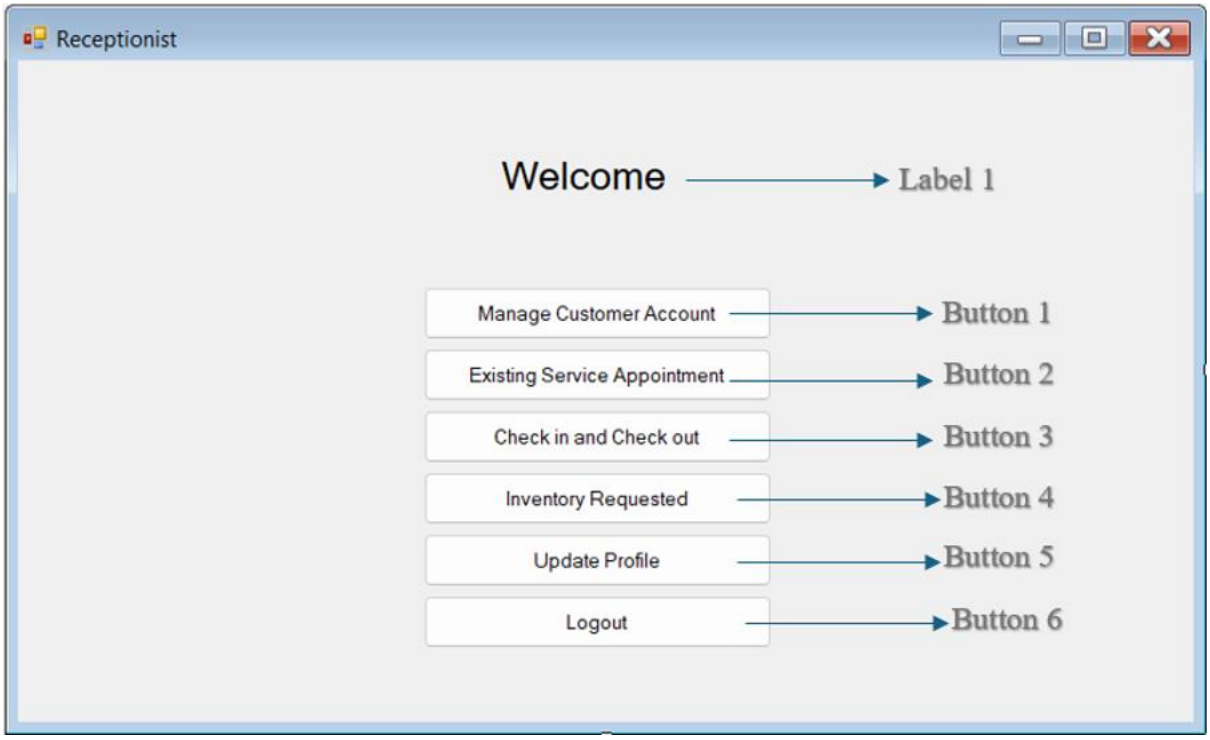


Control	Control Name	Description
Form 1	formAdReport	Form of Admin View Reports
Label 1	lblViewReport	To label View Reports page
Label 2	lblMonth	To label the month that chosen
Label 3	Label 3	To label the related controls to the right
Label 4	Label 4	
Label 5	lblServiceCompleted	To label the total service completed on that month
Label 6	lblTotalProfit	To label the total profit on that month
ListBox 1	listService	To list all the service completed
Button 1	btnGenerate	To generate the service that completed at that month
Button 2	btnBack	To go back to the previous page

Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1.	Generate monthly reports	To test whether system show the correct monthly report	Monthly report generated with correct month	Monthly report generated with	*User able to download

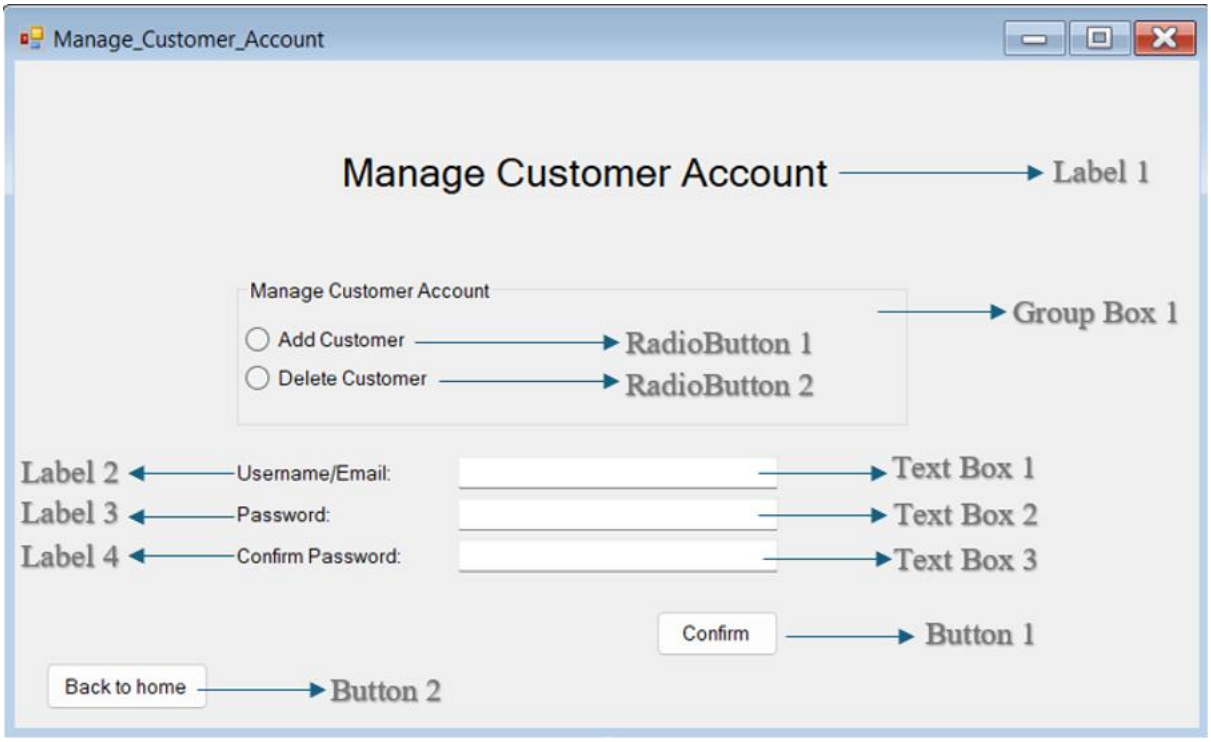
				correct month	monthly report
2.	Back to home	To test back to home	Back to home page successfully	Back to home page successfully	

3.2 Receptionist Storyboard & Test Plan



Control	Control Name	Description
Form 1	Receptionist	Form of Receptionist Home page
Label 1	lblWelcome	To welcome the receptionist
Button 1	btnManageCusAcc	To move to the form of Receptionist Manage Customer Account
Button 2	btnServiceApp	To move to the form of Receptionist Existing Service Appointment
Button 3	btnCheckinout	To move to the form of Receptionist Check in & Check out
Button 4	btnInventoryRequest	To move to the form of Receptionist Inventory Request
Button 5	btnUpdateProfile	To move to the form of Receptionist Update Profile
Button 6	btnLogout	To logout and move to the form of Login Page

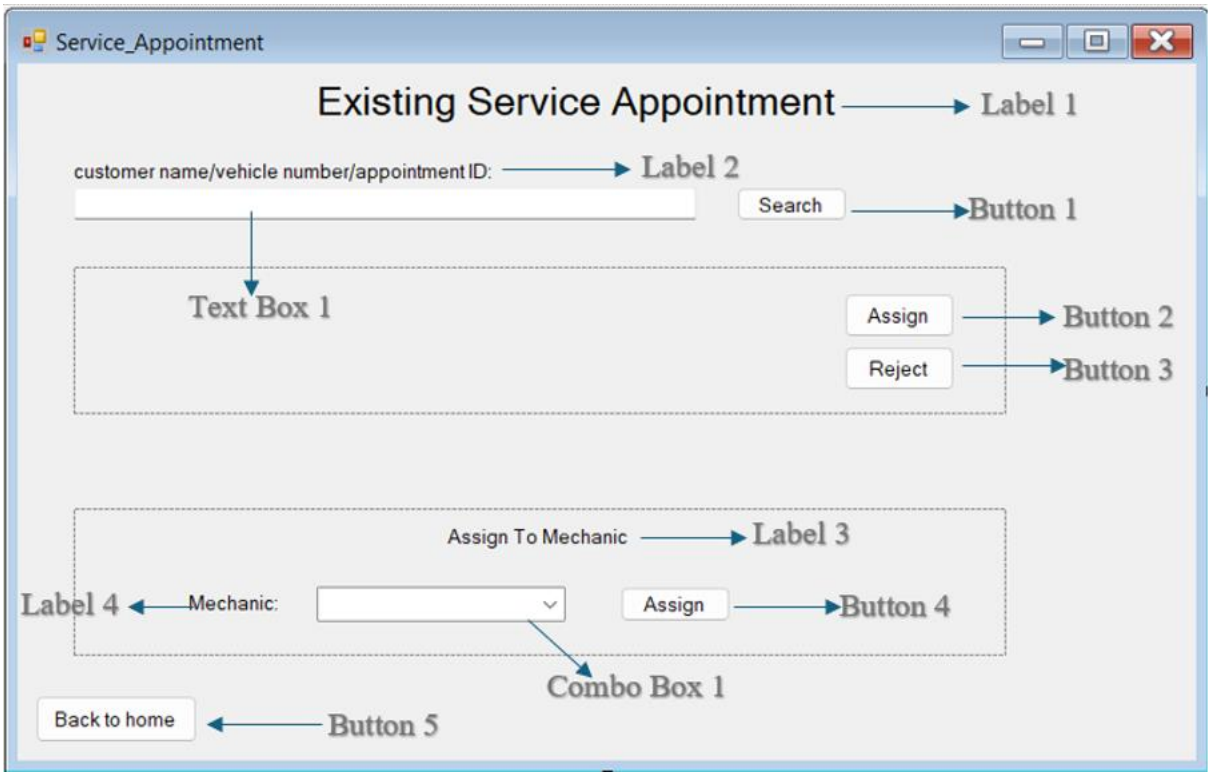
Test Case	Function	Test Objective	Expected Result	Actual Result	Remarks
1.	Logout	Verify receptionist can log out of the system	User is logged out and redirected to the login page	User successfully logged out, redirected to login page	



Control	Control Name	Description
Form 1	Manage_Customer_Account	Form of Receptionist Manage Customer Account
Label 1	lblManageCus	To label Manage Customer Account page
Label 2	label1	To label the related controls to the right
Label 3	label2	
Label 4	label3	
GroupBox 1	grpboxChoices	To group all choices
RadioButton 1	radbtnAddCus	To select action add customer
RadioButton 2	radbtnDelCus	To select action remove customer
Textbox 1	txtboxUsername	To insert the username/email
Textbox 2	txtboxPassword	To insert the password
Textbox 3	txtboxConfirmPassword	To insert the confirmation password
Button 1	btnConfirm	To confirm action
Button 2	btnBack	To go back to home

Test Case	Function	Test Objective	Expected Result	Actual Result	Remarks
1.	Add Customer Account	Verify that the receptionist can add a new customer account	New customer account is created successfully and a confirmation	Customer account successfully created, confirmation message displayed	

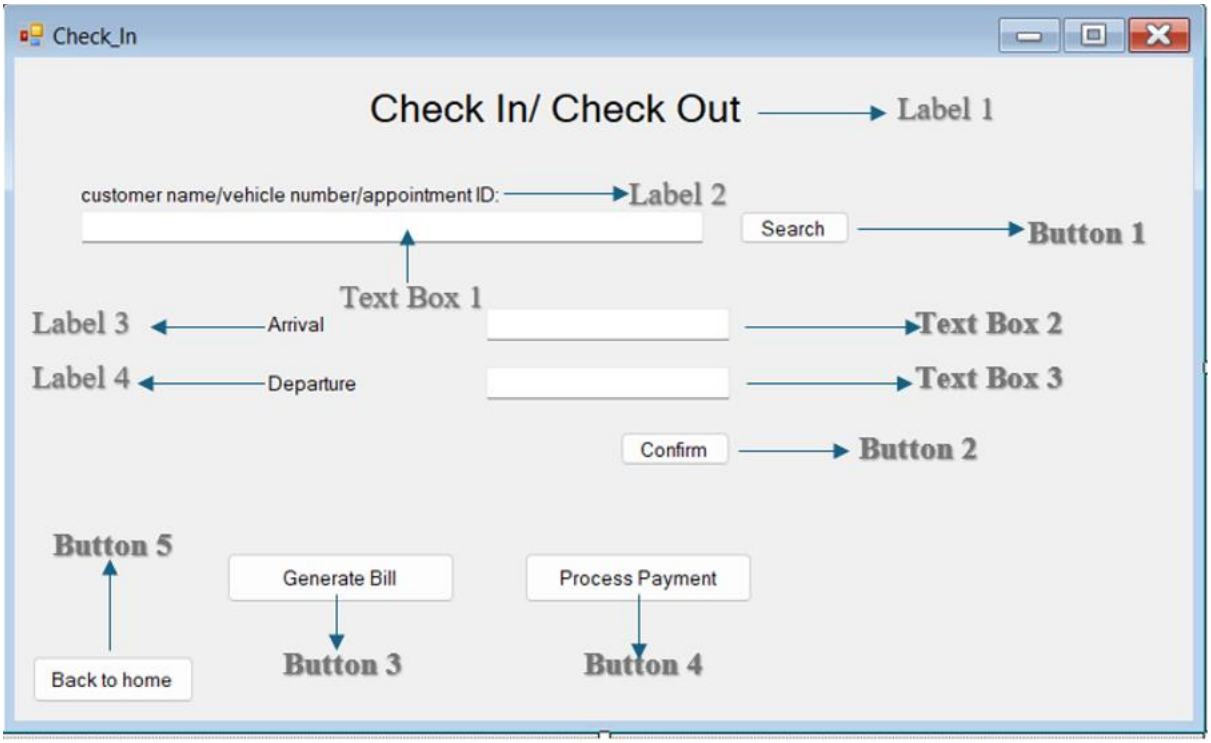
			message is displayed		
2.	Delete Customer Account	Ensure the receptionist can delete an existing customer account	Customer account is deleted, and a confirmation message appears	Customer account successfully deleted, confirmation message displayed	
3.	Validate Email Address	Test email validation during customer account creation	Error message displayed if email lacks "@" or "."	Error message displayed as expected for invalid email format	



Control	Control Name	Description
Form 1	Service_Appointment	Form of Receptionist Existing Service Appointment
Label 1	lblSearchApp	To label Existing Service Appointment page
Label 2	label2	To label Text Box 1
Label 3	label3	To label Assign to Mechanic
Label 4	label4	To label Combo Box 1
Textbox 1	txtboxUsername	To insert the username/email
Button 1	btnSearch	To perform search
Button 2	btnAssign	To perform assign
Button 3	btnReject	To perform reject

Button 4	btnConfirm	To confirm assignment
Button 5	btnBack	To go back to home form
Combo Box 1	comboBoxAssign	To choose mechanic

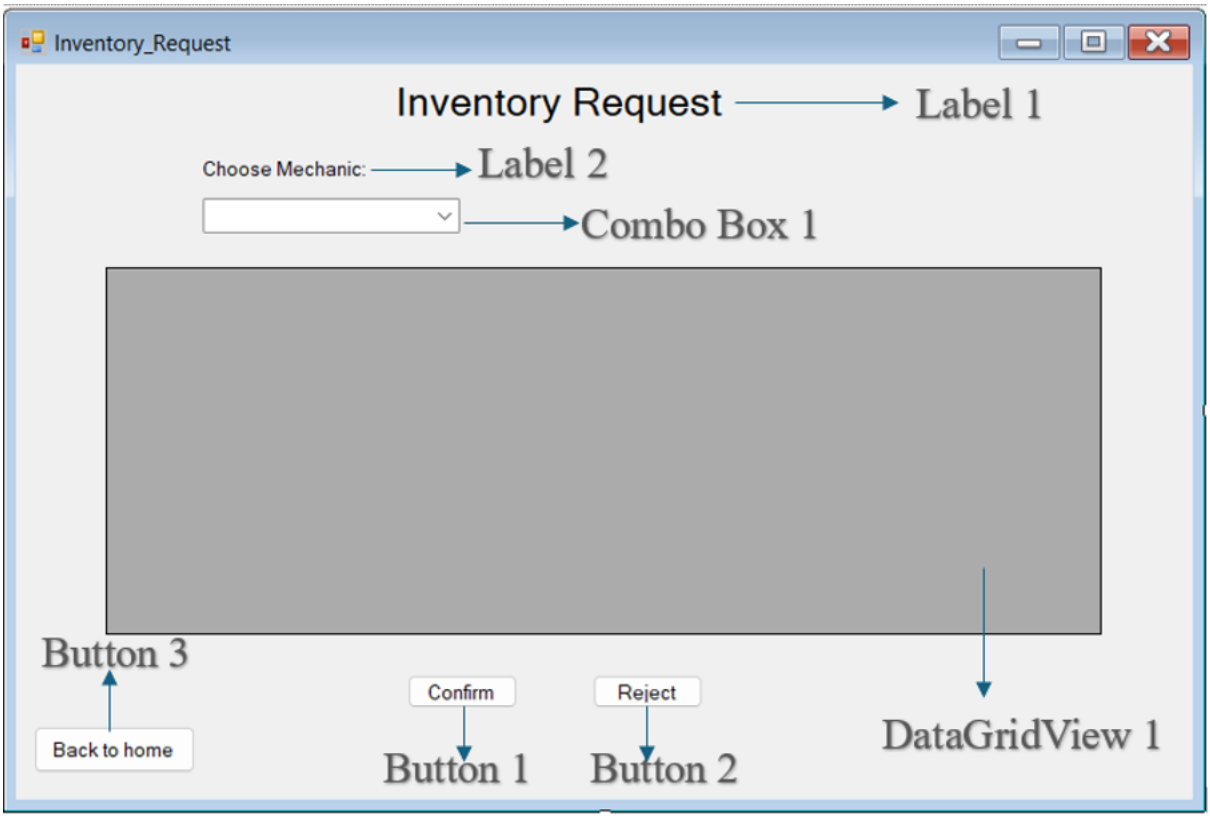
Test Case	Function	Test Objective	Expected Result	Actual Result	Remarks
1.	Search Existing Service Appointment	Ensure receptionist can search for an existing appointment	Appointment details are displayed when valid details are entered	Appointment details successfully displayed	
2.	Assign Mechanic	Verify mechanic can be assigned to a service appointment	Mechanic is assigned, and confirmation message is displayed	Mechanic successfully assigned, confirmation message displayed	



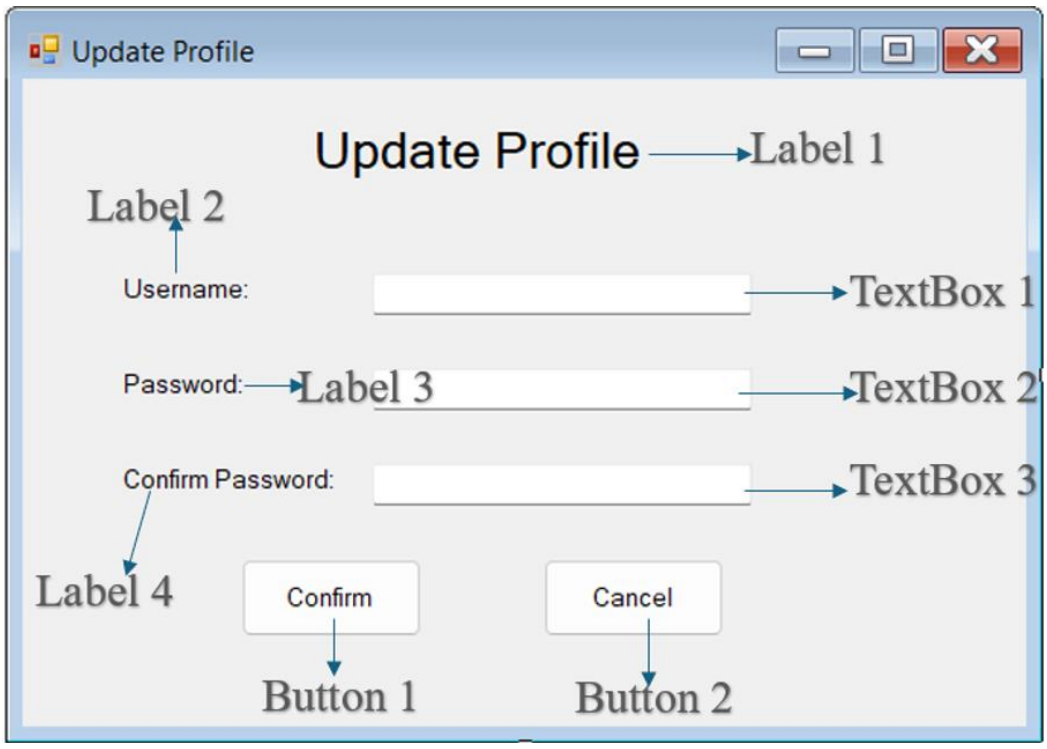
Control	Control Name	Description
Form 1	Check_In	Form of Receptionist Check in/ Check out
Label 1	label1	To label Check in/ Check out page
Label 2	label2	To label the related controls to the right
Label 3	label3	
Label 4	label4	

Textbox 1	txtboxUsername	To insert the customer name/ vehicle number/ appointment id
Textbox 2	txtboxArrival	To insert the arrival date and time
Textbox 3	txtboxDeparture	To insert the departure date and time
Button 1	btnSearch	To search the customer name/
Button 2	btnConfirm	To confirm check in/out
Button 3	btnBill	To generate bill
Button 4	btnPayment	To process payment
Button 5	btnBack	To go back to home form

Test Case	Function	Test Objective	Expected Result	Actual Result	Remarks
1.	Check In - Arrival	Ensure receptionist can mark the arrival of a customer	Arrival is recorded and confirmation message displayed	Arrival successfully recorded, confirmation message displayed	
2.	Check Out – Departure	Ensure receptionist can mark the departure of a customer	Departure is recorded and confirmation message displayed	Departure successfully recorded, confirmation message displayed	
3.	Generate Bill	Verify that the bill can be generated after service	Bill generated and displayed successfully	Bill generated successfully and displayed	
4.	Process Payment	Ensure payment processing functionality works as expected	Payment is processed, and a confirmation message is displayed	Payment processed successfully, confirmation message displayed	



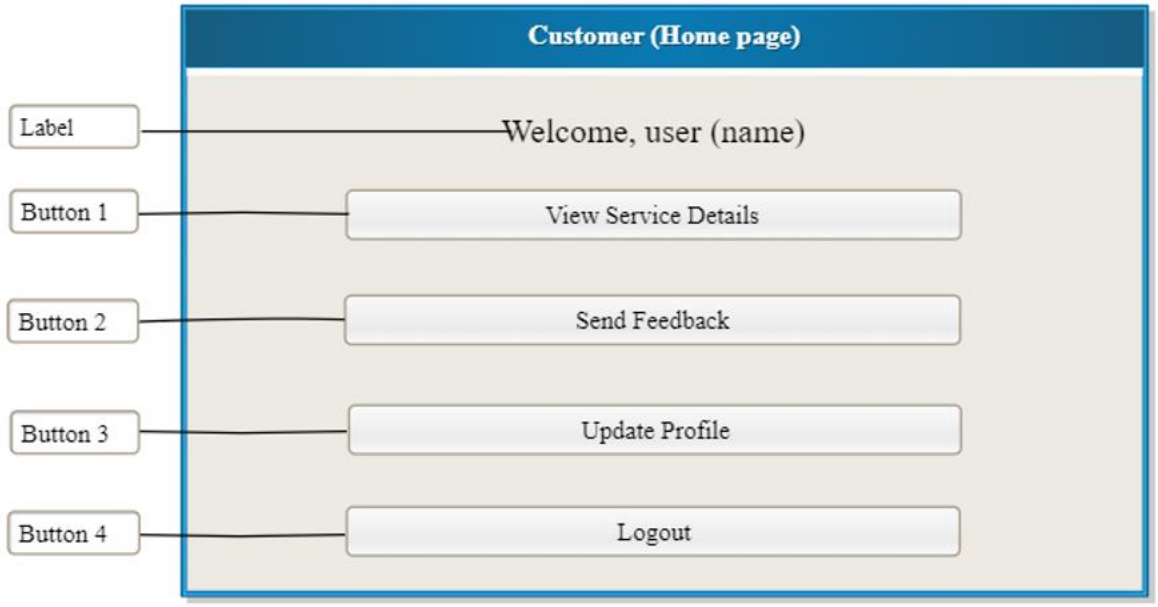
Control	Control Name	Description
Form 1	Inventory_Request	Form name
Label 1	label1	To display form information
Label 2	label2	To label combo boxes
Combo Box 1	comboBoxMechanic	To sort by mechanic
DataGridView	dataGridViewInventory	To display inventory requests
Button 1	btnConfirm	To accept request
Button 2	btnReject	To reject request
Button 3	btnBack	To go back to home form



Control	Control Name	Description
Form 1	Update Profile	Form name
Label 1	label1	To display form information
Label 2	label2	To label text boxes
Label 2	label2	To label text boxes
Label 2	label2	To label text boxes
Text Box 1	textBoxUsername	To enter username
Text Box 2	textBoxPassword	To enter password
Text Box 3	textBoxConfirmPassword	To confirm password
Button 1	btnConfirm	To update profile
Button 2	btnCancel	To cancel

Test Case	Function	Test Objective	Expected Result	Actual Result	Remarks
1.	Update Profile	Ensure receptionist can update their profile information	Profile updated successfully, and confirmation message displayed	Profile successfully updated, confirmation message displayed	

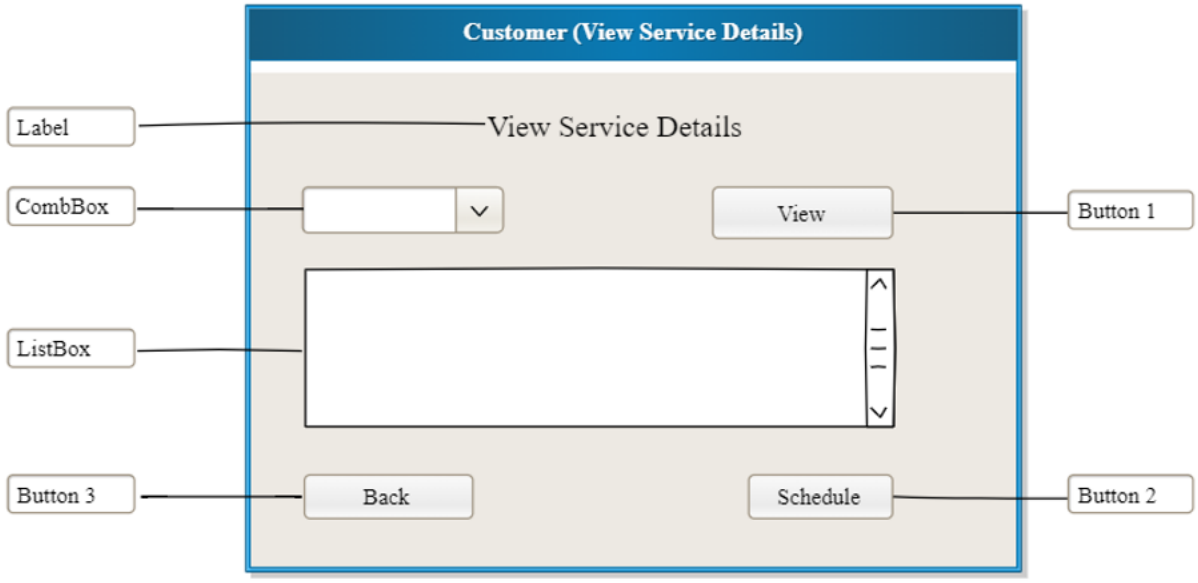
3.3 Customer Storyboard & Test Plan



Control	Control Name	Description
Label	Label	To display word of welcome customer.
Button 1	btnView	To allow customer to view service details.
Button 2	btnSend	To send feedback for service received.
Button 3	btnUpdate	Let customer update own profile.
Button 4	btnLogout	Let customer log out their own account.

Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1	Enter view service details page	To test whether customer can enter view service details page.	Customer can enter view service details page.	Customer can enter view service details page.	
2	Enter send feedback for service received page	To test whether customer can send feedback for service received page.	Customer can enter send feedback for service received page.	Customer can enter send feedback for service received page.	
3	Enter update profile page	To test whether customer can enter update profile page.	Customer can enter update profile page.	Customer can enter update profile page.	

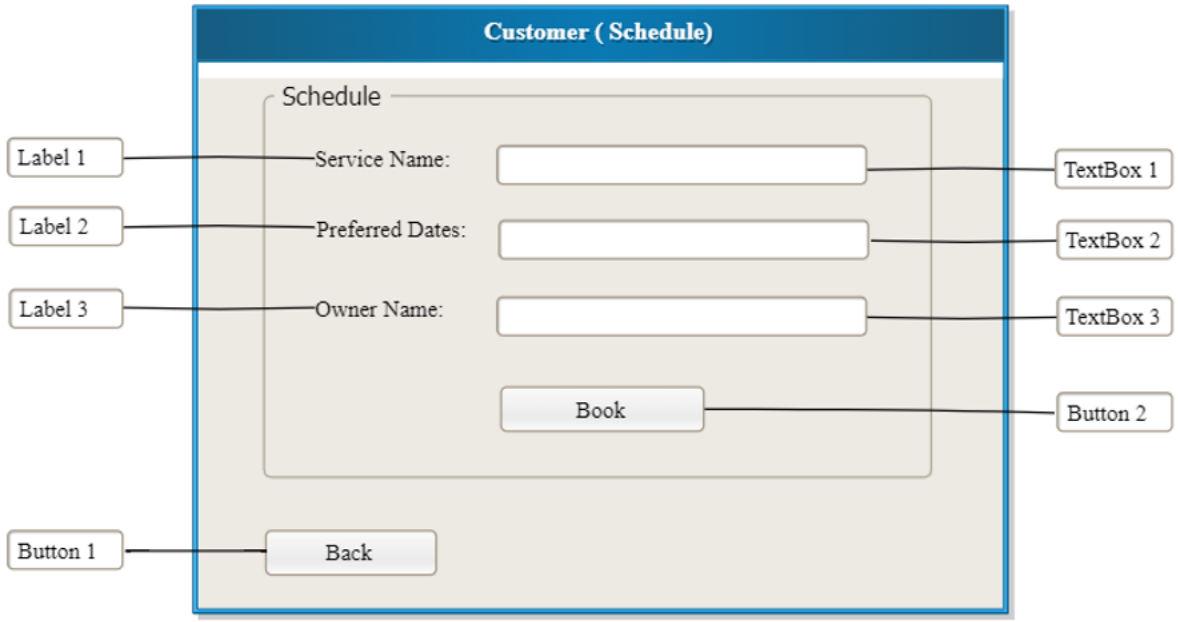
4	Enter log out page	To test whether customer can enter log out page.	Customer can enter log out page.	Customer can enter log out page.	



Control	Control Name	Description
Label	Label	To display words of view service details
ComboBox	cboView	Let customer choose what service type need to view
ListBox	listView	To display description, price and estimated time of service type in list box
Button 1	btnView	To view that customer want to view the service type
Button 2	btnSchedule	To allow customer go to schedule page
Button 3	btnBack	To allow customer can back to previous page

Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1	Selected a service type in combo box	To test whether customer can select service type from the combo box.	Customer can select service type from the combo box.	Customer can select service type from the combo box.	

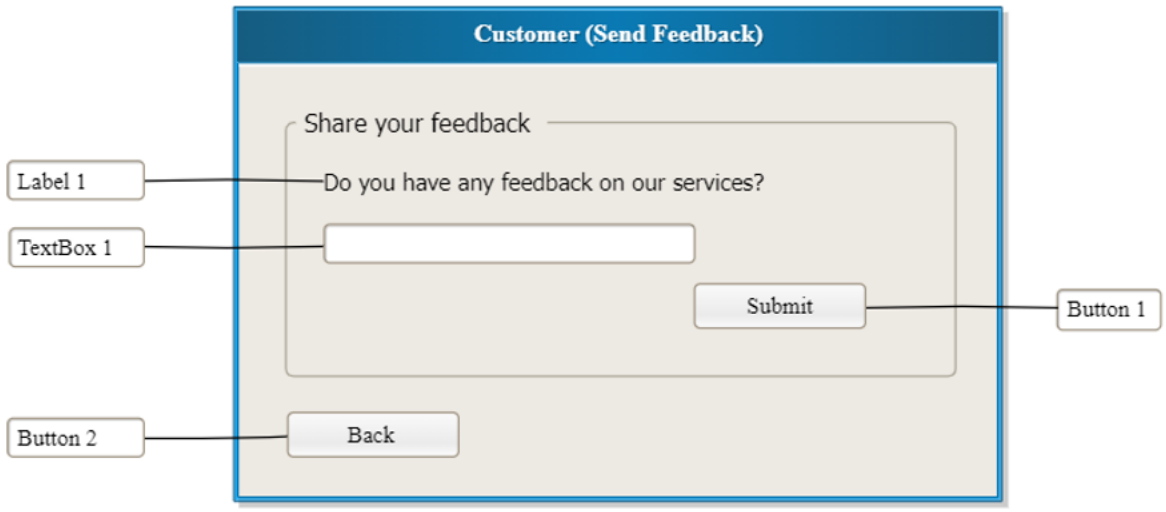
2	Enter view button	To test whether customer can view the service details in list box.	Customer can view service details in list box.	Customer can view service details in list box.	
3	Display description, price and estimated time in list box.	To test customer can view description, price and estimated time in list box.	Customer press view button can view description, price and estimated time in list box.	Customer press view button can view description, price and estimated time in list box.	
4	Enter schedule page	To test whether customer can enter schedule page.	Customer can enter schedule page.	Customer can enter schedule page	
5	Enter back page	To test whether customer can enter back page.	Customer can enter back to home page.	Customer can enter back to home page.	



Control	Control Name	Description
Label 1	Label	To label the control related to the right
Label 2	Label 2	
Label 3	Label 3	
TextBox 1	txtServiceName	To allow customer select service name
TextBox 2	txtPreferredDates	To allow customer select prefer date
TextBox 3	txtOwnerName	To allow customer fill in their name
Button 1	btnBack	To allow customer back to previous page

Button 2	btnBook	To allow customer book their appointment

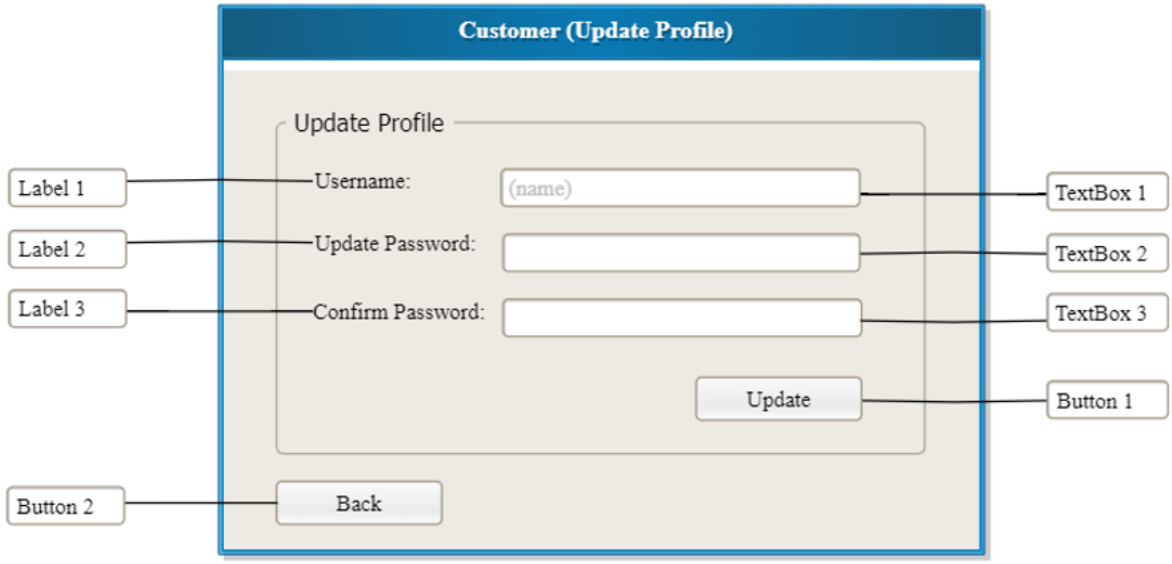
Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1	Fill in service name	To test whether customer want what service type to fill in.	Customer can fill in want what service type in text box.	Customer can fill in want what service type in text box.	
2	Fill in preferred date	Let the customer to fill in the prefer date.	Customer can fill in the prefer date.	Customer can fill in the prefer date.	
3	Fill in own name	Let customer to fill in their name.	Customer can fill in their name.	Customer can fill in their name.	
4	Enter back button	To test whether customer can back to previous page.	Customer can back to previous page.	Customer can back to previous page.	
5	Enter book page	To test whether customer can success to book appointment	Customer can book appointment.	Customer can book appointment.	



Control	Control Name	Description
Label 1	Label 1	To label the control related to the below

TextBox 1	TextBox 1	Let customer fill in feedback
Button 1	btnSubmit	Let customer submit the feedback
Button 2	btnBack	Let customer back to previous page

Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1	Write feedback after service received	To test whether customer can write feedback.	Customer can write feedback.	Customer can write feedback.	
2	Enter submit button	To test whether customer can submit feedback	Customer can submit feedback.	Customer can submit feedback.	
3	Enter back button	To test whether customer can back to previous page	Customer can back to previous page.	Customer can back to previous page.	

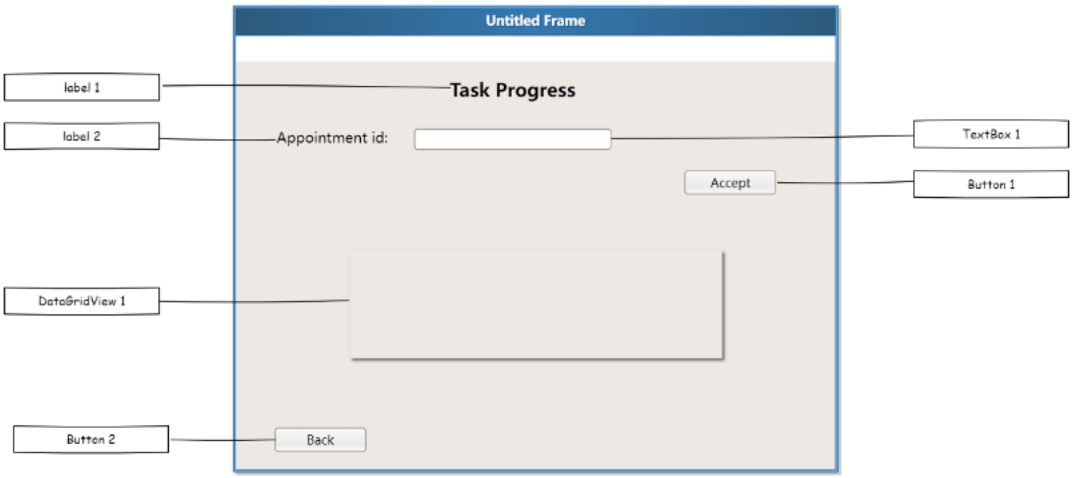


Control	Control Name	Description
Label 1	Label 1	To label the related control to the right
Label 2	Label 2	
Label 3	Label 3	
TextBox 1	txtUsername	Already display customer' name
TextBox 2	txtPassword	To allow customer fill in their password
TextBox 3	txtConfirmPassword	To allow customer fill in again their password

Button 1	btnUpdate	To allow customer to update their profile
Button 2	btnBack	To allow customer back to previous page

Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1	Customer name have display in text box	To test whether customer name have display in text box.	Customer name already display in text box.	Customer name already display in text box.	
2	Write password in text box	To test customer can write password in text box.	Customer can write password in text box.	Customer can write password in text box.	
3	Write again password in text box.	To test customer write again the same password.	Customer write again the same password.	Customer write again the same password.	
4	Enter update button	To test whether customer can update their profile	Customer can update their profile.	Customer can update their profile.	
5	Enter back page	To test whether customer can back to previous page	Customer can back to previous page	Customer can back to previous page.	

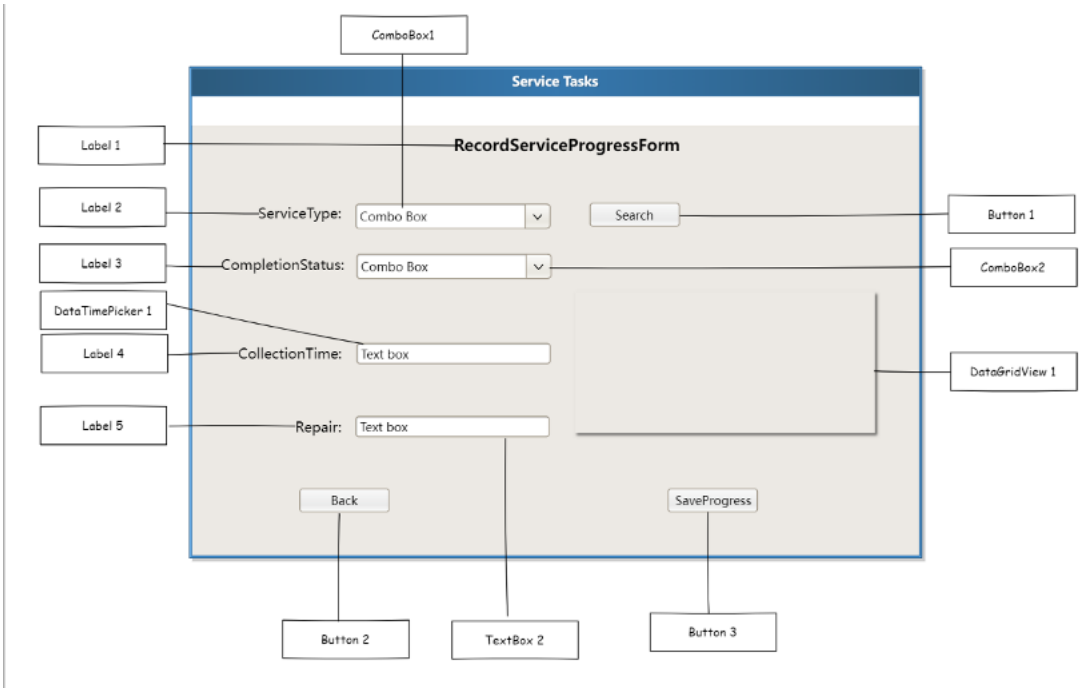
3.4 Mechanic Storyboard & Test Plan



Control	Control Name	Description
Label 1	lblHeader	Displays the form title "Task Progress"”
Label 2	lblTaskCode	Displays "Task Code:" for identifying a specific task.
Button 1	btnLoadTasks	Triggers the loading of appointment data based on the entered ID
Button 2	btnBack	Closes the current form
DataGridView 1	ConfigureDataGridView	Displays the appointment data in a tabular format

Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1	Load Appointment Data	Verify if the application can load appointment data from the database	TheConfigureDataGrid View is populated with appointment data from the database	ConfigureDataGrid View is populated with appointment data from the database	
2	Enter Appointment ID	Verify if the user can input an Appointment ID	The Appointment ID is entered correctly in the textbox.	Appointment ID is entered correctly in the textbox	

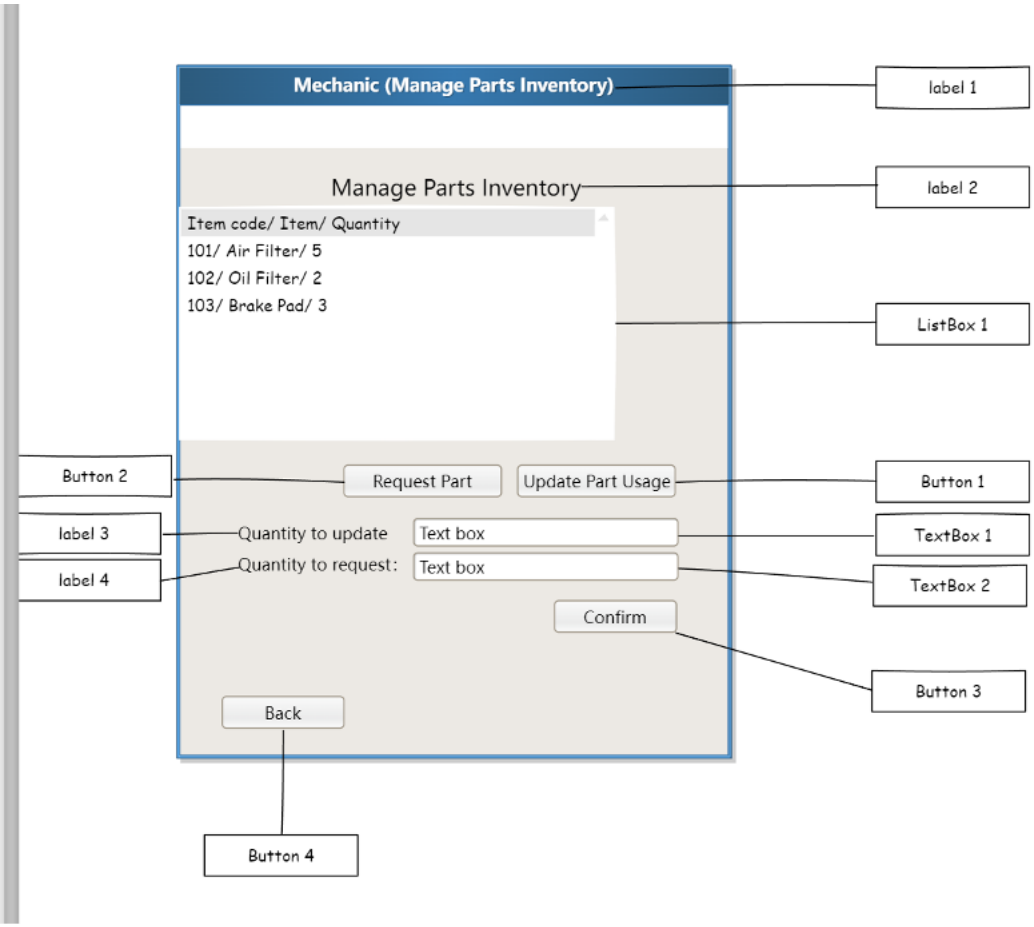
3	Filter by Appointment ID	Verify if the application can filter appointment data by Appointment ID	Only the appointment data matching the entered Appointment ID is displayed in the ConfigureDataGridView	ConfigureDataGridView displays all appointment data as there is no filtering functionality implemented in the code	
4	Accept Button Functionality	Verify if the Accept button triggers the filtering of data	The LoadAppointmentData() method is called and the ConfigureDataGridView is updated with filtered data	LoadAppointmentData() method is called and the ConfigureDataGridView displays all appointment data	
5	Back Button Functionality	Verify if the Back button closes the form	The form closes successfully when the Back button is clicked	The form closes successfully when the Back button is clicked	
6	Handle Invalid Input	Verify if the application handles invalid input (e.g., non-numeric characters) for Appointment ID	An error message is displayed, or the application prevents invalid input	The application does not handle invalid input	
7	Handle Empty Input	Verify if the application handles an empty input for Appointment ID	An error message is displayed, or the application prevents submitting an empty input	The application displays a message when the input is empty	



Control	Control Name	Description
Label 1	lblServiceType	Displays "Service Type:" for selecting the type of service
Label 2	lblCompletionStatus	Displays "Completion Status:" to indicate the progress of the service
Label 3	lblCollectionTime	Displays "Collection Time:" to specify the scheduled pick-up time
Label 4	lblRepair	Displays "Repair:" for any additional repair details
ComboBox 1	cboServiceType	Dropdown control to select the service type (e.g., "Repair", "Maintenance")
ComboBox 2	cboCompletionStatus	Dropdown control to select the completion status (e.g., "In progress", "Completed")
DateTimePicker 1	dateTimePickerCollectionTime	Allows the user to select a date and time for collection
TextBox 1	txtAdditionalRepairs	Textbox to input any additional repair details or notes
Button 1	btnSearch	Initiates the search for appointments based on the selected service type
Button 2	btnSaveProgress	Saves the entered service progress information to the database
Button 3	btnBack	Closes the current form and returns to the previous form
DataGridView 1	dataGridView1	Displays the appointment information retrieved based on the selected service type

Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remark
1	Load Service Types	Verify if the application can load service types from the database into the `cboServiceType` combobox	The `cboServiceType` combobox is populated with service types from the database	The cboServiceType combobox is populated with service types from the database	
2	Select Service Type	Verify if the user can select a service type from the `cboServiceType` combobox	A service type is selected from the `cboServiceType` combobox	A service type is selected from the cboServiceType combobox	
3	Search for Appointments	Verify if the application can search and display appointments based on the selected service type	The `dataGridView1` is populated with appointment data matching the selected service type	The dataGridView1 is populated with appointment data matching the selected service type	
4	Select Completion Status	Verify if the user can select a completion status from the `cboCompletionStatus` combobox	A completion status is selected from the `cboCompletionStatus` combobox	A completion status is selected from the cboCompletionStatus combobox	
5	Set Collection Time	Verify if the user can set the collection time using the `dateTimePickerCollectionTime` control	The collection time is set correctly in the `dateTimePickerCollectionTime`	The collection time is set correctly in the dateTimePickerCollectionTime	
6	Enter Additional Repairs	Verify if the user can enter additional repair details in the `txtAdditionalRepairs` textbox	Text is entered correctly in the `txtAdditionalRepairs` textbox	Text is entered correctly in the txtAdditionalRepairs textbox	
7	Save Service Progress	Verify if the application can save the entered service progress details to the database	The service progress details (completion status, collection time, additional repairs) are saved to the database for the selected appointment	The service progress details (completion status, collection time, additional repairs) are saved to the database for the selected service type. Since there is no appointment selection in the form, the code updates all	

				appointments with the selected service type	
8	Back Button Functionality	Back Button Functionality | Verify if the Back button closes the form	The form closes successfully when the Back button is clicked	The form closes successfully when the Back button is clicked	
9	Handle Empty Service Type Selection	Verify if the application handles cases where no service type is selected when searching or saving	An error message is displayed indicating that a service type must be selected	An error message is displayed indicating that a service type must be selected	

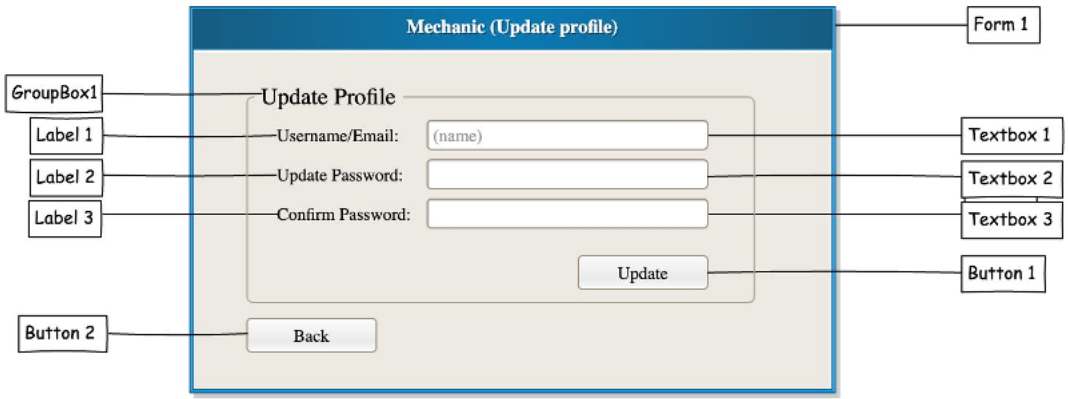


Control	Control Name	Description
Label 1	Label 1	Displays the form title "Mechanic (Manage Parts Inventory)"
Label 2	Label 2	Displays "Manage Parts Inventory" as a heading for the inventory list

Label 3	lblQuantityUpdate	Displays "Quantity to update" for updating part quantity
Label 4	lblQuantityRequest	Displays "Quantity to request" for requesting new parts
DataGridView 1	dataGridView1	Displays the current parts inventory with item code, item name, and quantity
TextBox 1	txtQuantityUpdate	Textbox to input the quantity for updating the existing inventory
TextBox 2	txtQuantityRequest	Textbox to input the quantity for requesting new parts
Button 1	btnUpdatePartUsage	Button to request a part from inventory
Button 2	btnRequestPart	Processes a request for new parts based on the entered quantity
Button 3	btnConfirm	Confirms both the inventory usage update and the part request
Button 4	btnBack	Closes the current form and returns to the previous form

Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1	Open Manage Inventory Form	Verify if the user can open the Manage Parts Inventory form	The Manage Parts Inventory form opens successfully	The Manage Parts Inventory form opens successfully	
2	Load Inventory Data	Test if the parts inventory data is loaded correctly when the "Manage Parts Inventory" form loads.	Parts inventory data is successfully loaded into dataGridView1	Parts inventory data is successfully loaded into dataGridView1	
3	Select Inventory Item	Test if the user can select a parts inventory item from dataGridView1	An inventory item is selected successfully	An inventory item is selected successfully	
4	Enter Quantity to Request	Verify if the user can enter a quantity to update in the "Quantity to update" textbox	The quantity is entered correctly in txtQuantityRequest	The quantity is entered correctly in txtQuantityRequest	

5	Update Part Usage	Test if the user can click the "Update Part Usage" button to update the usage of the selected part	Part usage is updated successfully, and the data in dataGridView1 is updated accordingly	Part usage is updated successfully, and the data in dataGridView1 is updated accordingly	
6	Enter Request Quantity	Verify if the user can enter a quantity to request in the "Quantity to request" textbox	A positive integer is entered correctly in txtQuantityRequest	A positive integer is entered correctly in txtQuantityRequest	
7	Request Part	Test if the user can click the "Request Part" button to request the selected part	Part request is processed successfully, and the data in dataGridView1 is updated accordingly	Part request is processed successfully, and the data in dataGridView1 is updated accordingly	
8	Confirm Update and Request	Test if the user can click the "Confirm" button to simultaneously update part usage and request a part	Both part usage and request are processed successfully, and the data in dataGridView1 is updated accordingly	Both part usage and request are processed successfully, and the data in dataGridView1 is updated accordingly	
9	Back to Previous Form	Test if the user can click the "Back" button to return to the previous form	The current form is closed successfully, and the user is returned to the previous form	The current form is closed successfully, and the user is returned to the previous form	

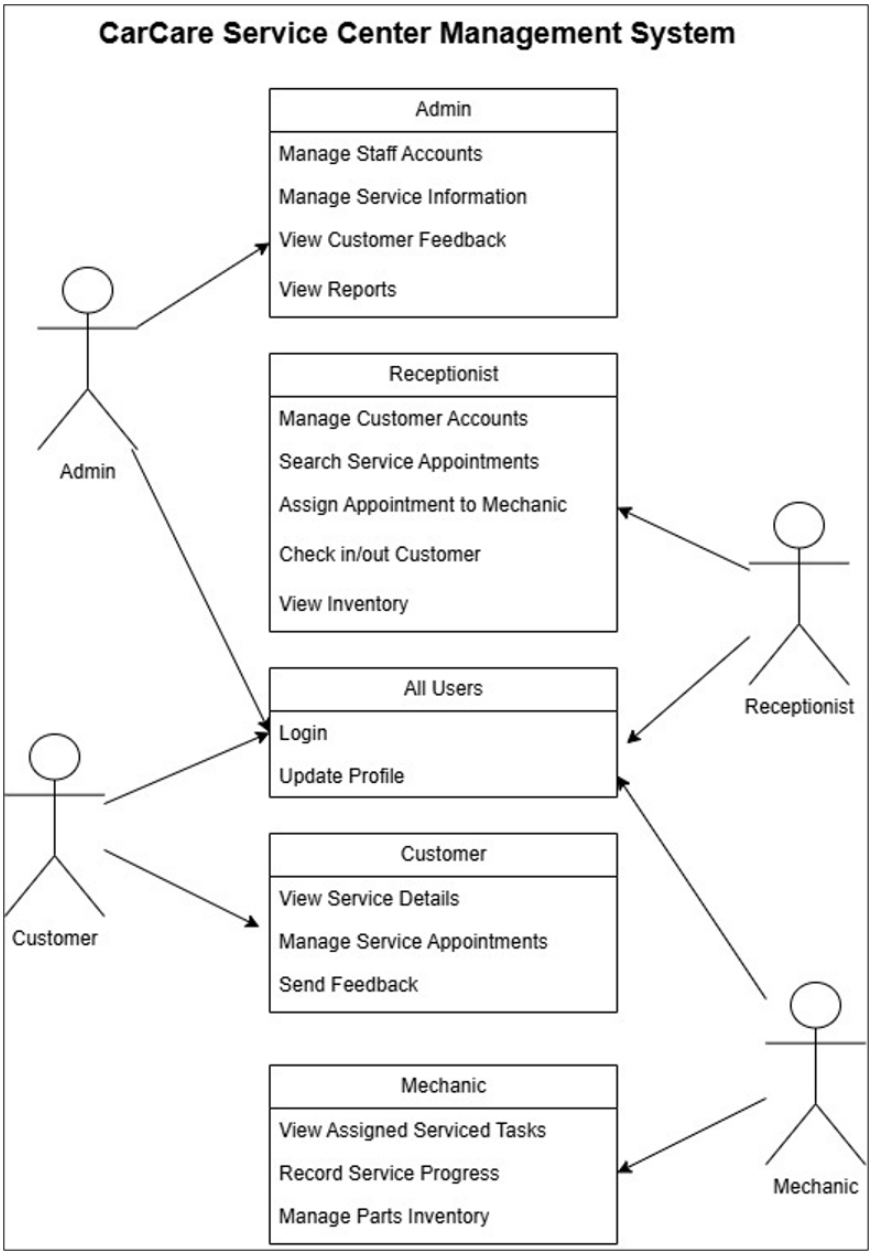


Control	Control Name	Description
GroupBox 1	grpUpdateProfile	Group box to contain profile update fields and labels
Label 1	lblUsernameEmail	Displays "Username/Email" label
Label 2	lblUpdatePassword	Displays "Update Password" label for entering a new password
Label 3	lblConfirmPassword	Displays "Confirm Password" label for re-entering the password
TextBox 1	txtUsernameEmail	Textbox that displays the current username/email
TextBox 2	txtUpdatePassword	Textbox for entering a new password
TextBox 3	txtConfirmPassword	Textbox for confirming the new password
Button 1	btnUpdate	Button labeled "Update" to save the updated profile information
Button 2	btnBack	Button labeled "Back" to return to the previous screen

Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remarks
1	Open Update Profile Form	Verify if the user can open the Update Profile form	The Update Profile form opens successfully	The Update Profile form opens successfully	
2	Display Username/Email	Check if the current username/email is displayed	The username/email is displayed in TextBox1	The username/email is displayed in TextBox1	
3	Enter New Password	Test if the user can input a new password	The new password is entered correctly in TextBox2	The new password is entered correctly in TextBox2	
4	Confirm New Password	Test if the user can input the confirmation password	The confirmation password is entered in TextBox3	The confirmation password is entered in TextBox3	
5	Password Match Check	Verify if the form checks if passwords match	If passwords match, the form proceeds to update	If passwords match, the form proceeds to update	
6	Password Mismatch Error	Check if an error is shown when passwords do not match	Error message "Passwords do not match, please re-enter" is displayed	Error message "Password does not match! Please re-enter your	

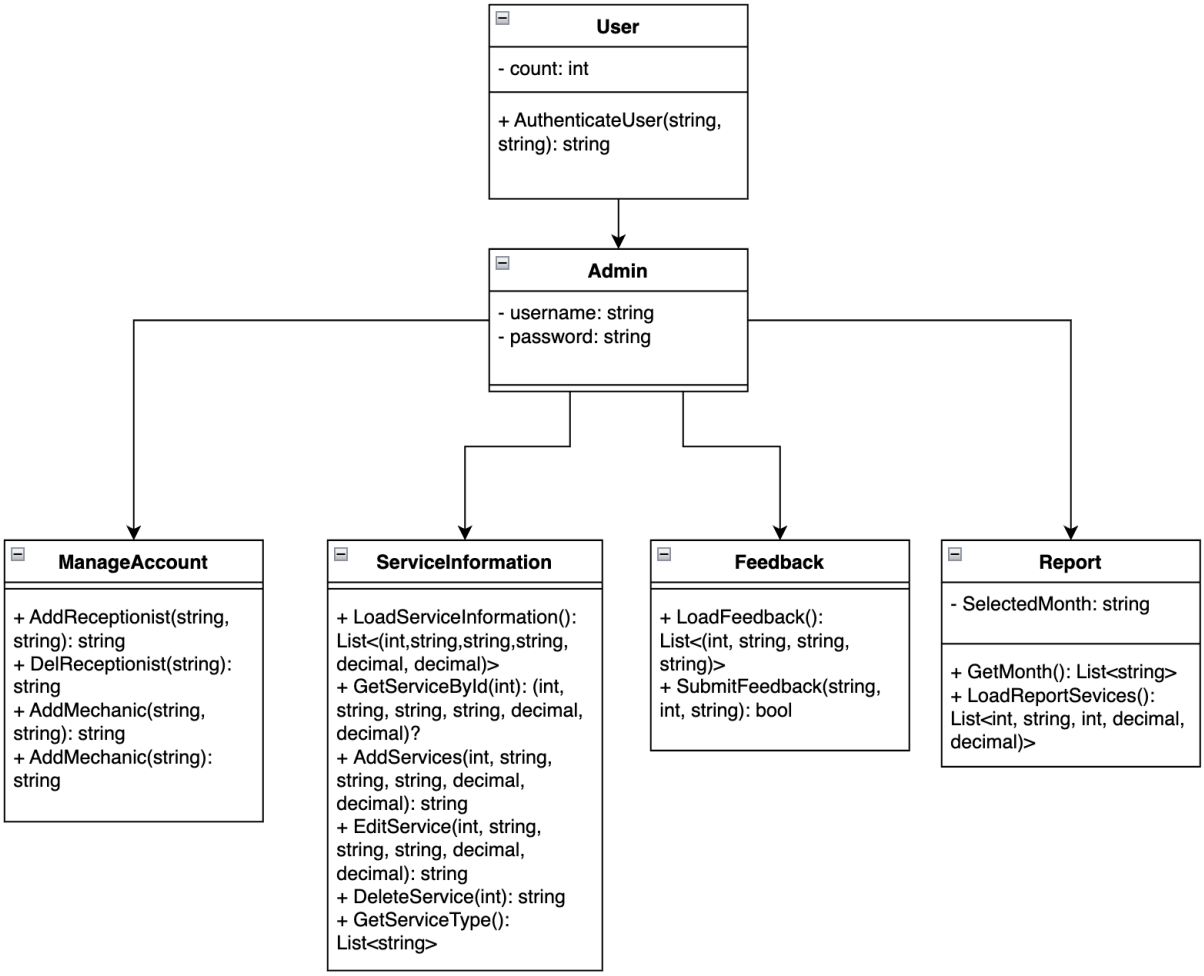
				password." is displayed	
7	Empty Password Fields	Verify if the form displays an error if password fields are empty	Error message "Password cannot be empty" is displayed	Error message "Fill in all fields, Please re-enter." is displayed	
8	Update Profile	Verify if clicking "Update" saves the updated profile	Message "Profile updated successfully" is displayed	Message "Update successful!" is displayed	
9	Back Button	Test if clicking "Back" returns the user to the previous screen	The form closes, and the user is returned to the previous screen	The form hides, and the MainForm is shown	

4. Use-case Diagram

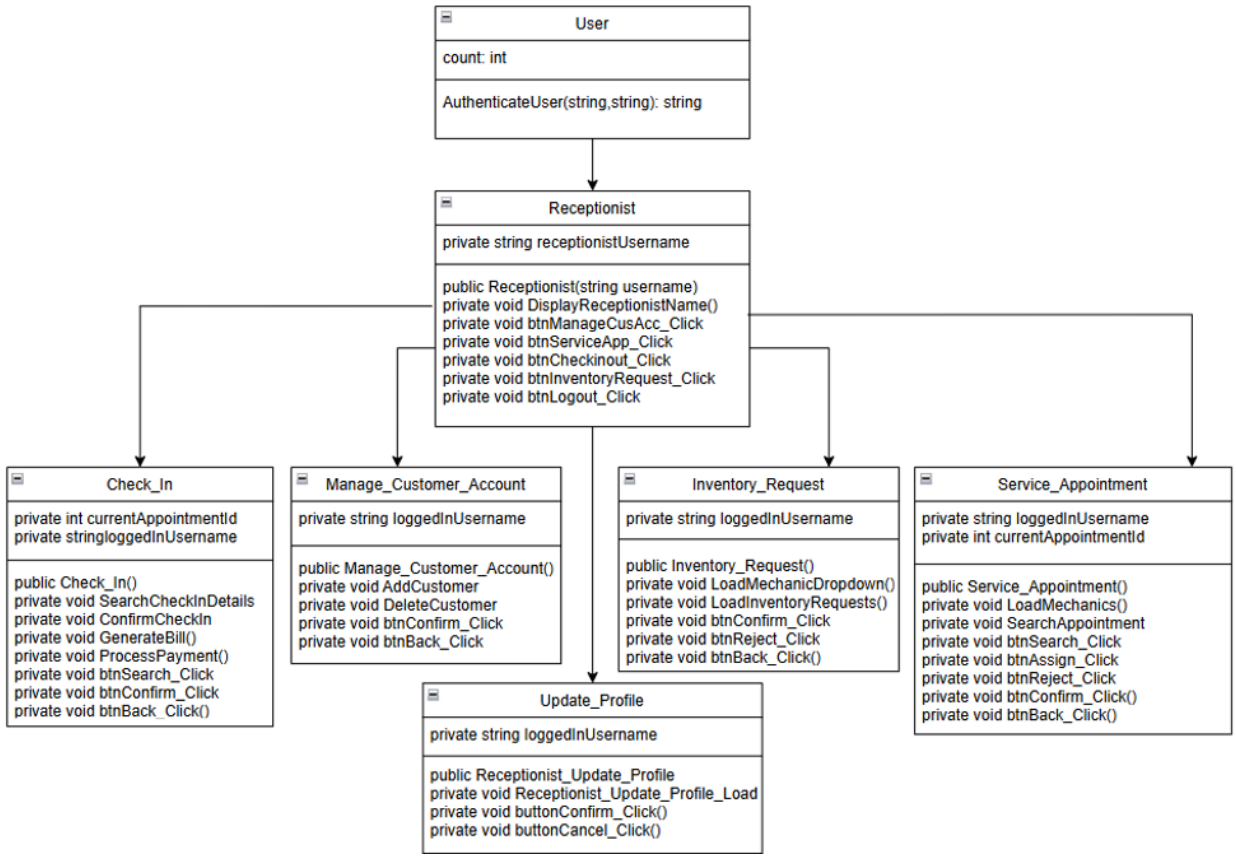


5. Class Diagram

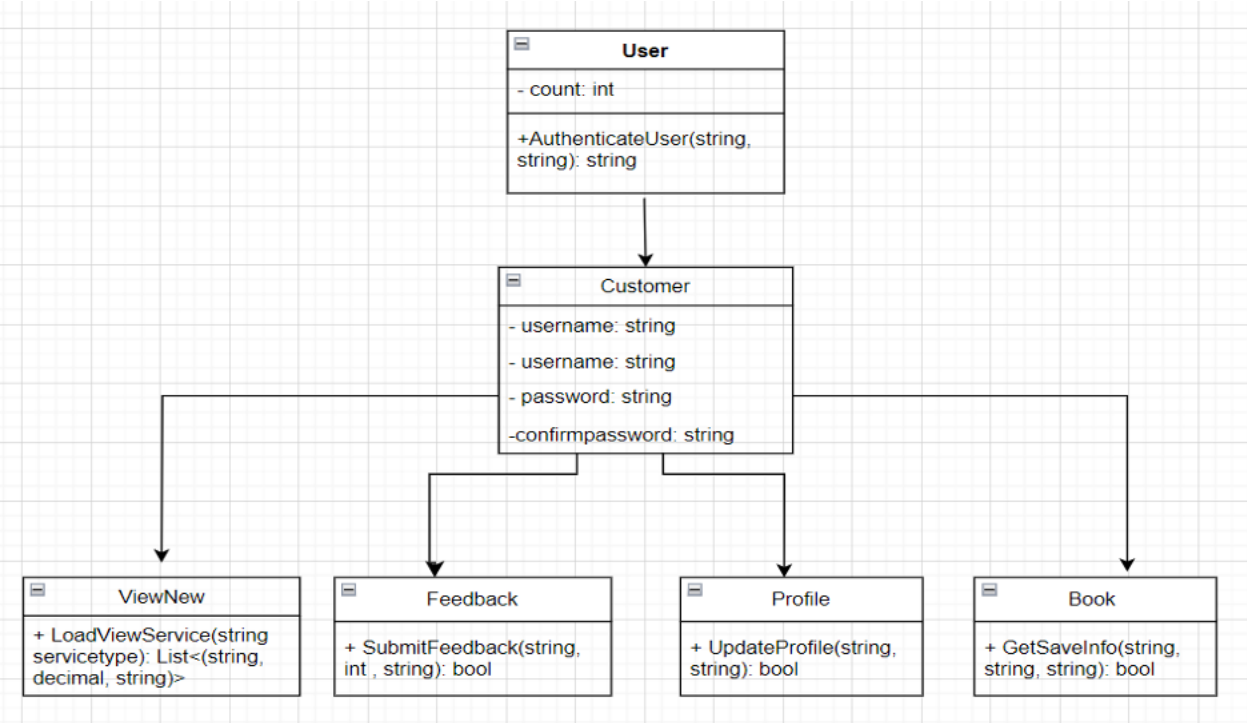
5.1 Admin Class Diagram



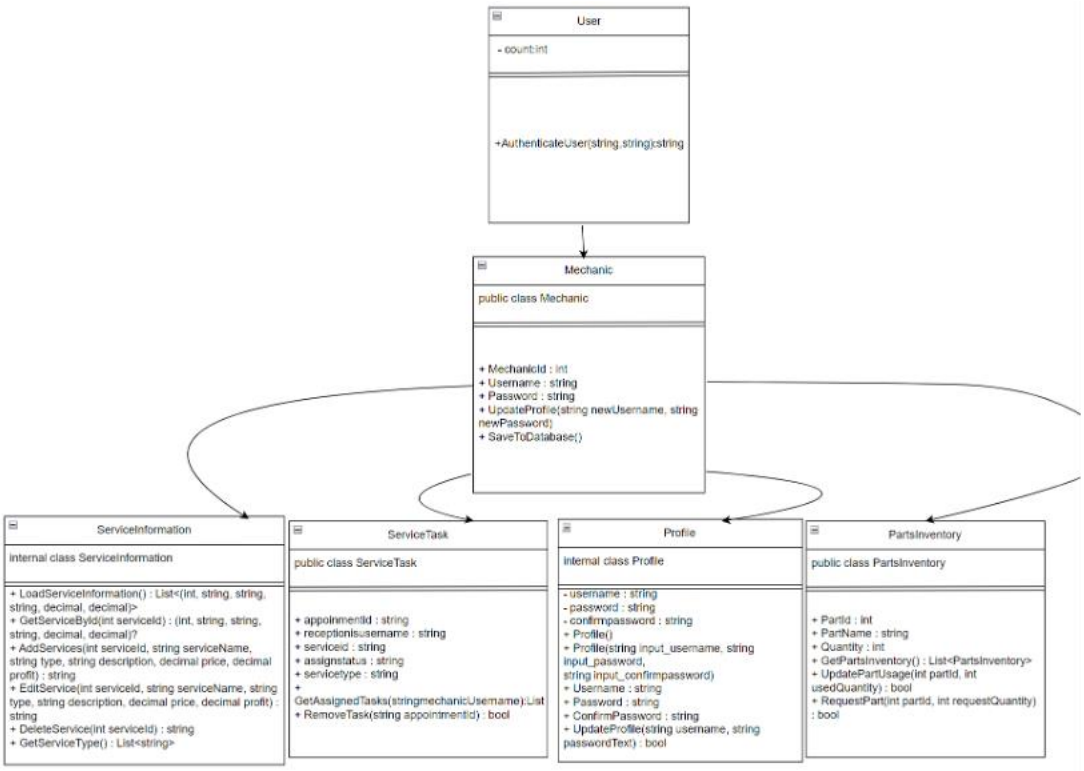
5.2 Receptionist Class Diagram



5.3 Customer Class Diagram



5.4 Mechanic Class Diagra



6. Code Explanation

6.1 Admin Code Explanation

Class Explanation

In admin code, the class can be categorized into two, which is Windows Form class and class. Windows Form class is class that automatically generates while form been created, while class are created by designer. Windows Form class is a class that represents user interface component in the application. In admin, there are seven Windows Form which are formHomePage, formManageStaff, formManageService, formFeedback, formReport, formAddService and formEditService. Whereby the class are non-UI classes that handle the logic and data processing for the application. In admin, there are four classes which are ManageAccount, ServiceInformation, Feedback and Report.

Example:

```
public partial class formAdHome : Form
{
    private Admin admin;
    public formAdHome(Admin admin)
    {
        InitializeComponent();
        this.admin = admin;
    }
}
```

This is one of the Windows Form classes, formAdHome. The first line of the code clarifies the access of the form and the name of the form. While admin login to its account, this form will be displayed and execute the function.

```
namespace IOOP_Assignment
```

```
{
    public class Admin
    {
        private string username;
        private string password;
    }
}
```

While this is one of the classes, Admin. In the field of the class can be included attributes, constructor, and method. As the example shows, there are private attributes username and password in this class. For class, the application should call it out to use the method in it.

Method Explanation

Method is a group of multiple lines of code that performs an/several action. It will always be in the field of a class. While it is in a Windows Form class, the method will usually interact with the object created in the form which is the UI design. Whereby the method in class is usually performing specific actions for data processing.

Example:

private void btnLogout_Click(object sender, EventArgs e)

This is one of the methods that are included in Admin Home Page (Windows Form class). It will execute the action in the field (Logout) while the button called btnLogout was clicked.

public string AddReceptionist(string username, string password)

While this is a method that is included in ManageStaff class. It will execute the action in the field (Add receptionist name and password to database) while been called.

Object Explanation

C# is an object-oriented program. While we are creating something in a form or in a class, it is considered as an object. For example, when we are adding a button inside a form by drag and drop, both button and form are considered as an object. Not only that, when we are using class functions in form, we should create this class object to call its method.

Example:

this.btnLogout.Font = new System.Drawing.Font("Times New Roman", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

This is button is created as an object while we drag and drop it inside our form.

Admin admin = new Admin

While we are creating a class object, this object is considered as an instance of class. The class that created this object could use the method inside this admin class.

6.2 Receptionist Code Explanation

User Clas

count: int: Tracks the number of users or sessions (purpose depends on its usage in the program).

Methods:

- **AuthenticateUser(string username, string password): string**: Authenticates the user by checking the username and password against the database. Returns the role of the user (e.g., Admin, Receptionist, Mechanic, Customer).

Receptionist Class

private string receptionistUsername: Stores the logged-in receptionist's username

Methods:

- **Receptionist(string username)**: Constructor that initializes the receptionist with the logged-in username.
- **DisplayReceptionistName()**: (If implemented) Displays the receptionist's name on the form.
- **btnManageCusAcc_Click()**: Opens the **Manage Customer Account** form.
- **btnServiceApp_Click()**: Opens the **Service Appointment** form.
- **btnCheckinout_Click()**: Opens the **Check In** form.
- **btnInventoryRequest_Click()**: Opens the **Inventory Request** form.
- **btnLogout_Click()**: Logs out the user and returns to the login screen.

Check_In Class

private int currentAppointmentId: Stores the Appointment ID of the currently selected appointment.

private string loggedInUsername: Stores the logged-in receptionist's username.

Methods:

- **Check_In(string username)**: Constructor that initializes the form with the logged-in username.
- **SearchCheckInDetails(string searchQuery)**: Searches for an appointment by either Appointment ID or Customer Username.
- **ConfirmCheckIn(int appointmentId, DateTime arrival, DateTime departure)**: Updates the appointment with check-in and check-out times.
- **GenerateBill()**: Generates a bill for the selected appointment by fetching service details.

- **btnSearch_Click()**: Triggers the search functionality for appointments.
- **btnConfirm_Click()**: Confirms the check-in/check-out details for the selected appointment.
- **btnBill_Click()**: Calls GenerateBill() to display the bill.
- **btnPayment_Click()**: Displays a message for payment instructions.
- **btnBack_Click()**: Returns to the receptionist's home page.

Manage_Customer_Account Class

private string loggedInUsername: Stores the logged-in receptionist's username.

Methods:

- **Manage_Customer_Account(string username)**: Constructor that initializes the form with the logged-in username.
- **AddCustomer(string username, string password)**: Adds a new customer to the database with the provided username and password.
- **DeleteCustomer(string username)**: Deletes a customer from the database based on the username.
- **btnConfirm_Click()**: Determines whether to add or delete a customer based on the selected radio button.
- **btnBack_Click()**: Returns to the receptionist's home page.

Inventory_Request Class

private string loggedInUsername: Stores the logged-in receptionist's username.

Methods:

- **Inventory_Request(string username)**: Constructor that initializes the form with the logged-in username.
- **LoadMechanicDropdown()**: Loads a list of mechanics into a dropdown from the database.
- **LoadInventoryRequests()**: Loads inventory requests from the database into a data grid view.
- **btnConfirm_Click()**: Confirms the selected inventory request by updating the database.
- **btnReject_Click()**: Rejects the selected inventory request.
- **btnBack_Click()**: Returns to the receptionist's home page.

Service_Appoitnment Class

private string loggedInUsername: Stores the logged-in receptionist's username.

private int currentAppointmentId: Stores the current Appointment ID.

Methods:

- **Service_Appointment(string username)**: Constructor that initializes the form with the logged-in username.
- **LoadMechanics()**: Loads a list of mechanics into a dropdown from the database.
- **SearchAppointment(string searchQuery)**: Searches for an appointment by Appointment ID or Customer Username.
- **btnAssign_Click()**: Displays a message indicating an appointment is being assigned.
- **btnReject_Click()**: Displays a message indicating an appointment is rejected.
- **btnConfirm_Click()**: Confirms the assignment of a mechanic to an appointment.
- **btnSearch_Click()**: Triggers the search functionality for appointments.
- **btnBack_Click()**: Returns to the receptionist's home page.

Update_Profile Class

private string loggedInUsername: Stores the logged-in receptionist's username.

Methods:

- **Receptionist_Update_Profile(string username)**: Constructor that initializes the form with the logged-in username.
- **Receptionist_Update_Profile_Load()**: Loads the current receptionist's username into the username text box.
- **buttonConfirm_Click()**: Updates the receptionist's profile in the database with the new username and password.
- **buttonCancel_Click()**: Returns to the receptionist's home page.

6.3 Customer Code Explanation

Class Explanation

- Customer: This class represents a customer. It has properties for storing the customer's username and password and methods for updating their profile. The class interacts with a "Customer" table in the database.
- Book: This class represents customer choose they want service type then fill in their booking appointment, it will store their information in “Appointment”table.
- Profile: This class appears to be a general class for managing user profiles, with properties for username, password, and password confirmation. It has a method for updating a profile, which interacts with a "Customer" table. This might indicate that it's used for both customer and mechanic profiles, but more context is needed to be sure.
- ViewNew: This class is represent let customer view service details. The customer choose want service want to view, then it will display description, price and estimated time.
- Feedback: This represents a customer write feedback of service received,then it will save in “Feedback” table.

Method Explanation

- boookschedule(List<string>ServiceTypes, List<DateTime>PreferredDates, string OwnerName) (in Schedule): Sore the appointment information in the database.
- GetSaveInfo(string name, string date, string owner) (in Schedule): Save the customer's requested service type, preferred date and name into the database
- List<(int, string, string, string)> LoadFeedback() (in Feedback): Let customers to fill in feedback after receiving the service, and then save it in the database
- UpdateProfile(string username, string passwordText) (in Update Profile): Update the password for a user in database.
- AuthenticateUser(string username, string password) (in Customer): Updates the customer’s profile in the database.
- List<(string, decimal, string)> LoadViewService(string servicetype) (in View Service Details): Call from the database to store description, price and estimated time in the database

Objects Explanation

Rather than defining particular objects, the code snippets mainly define classes and functions. When the program runs, objects are produced from these classes. For example, a customer object will represent a specific customer with their own username and password.

6.4 Mechanic Code Explanation

Classes Explanations

- **ServiceTask: This class represents the service task and if there is a saved task, the saved task will be assigned to the task information of the mechanic. It is designed to interact with a database table named "ServiceTasks".**
- **ServiceInformation:** This class is responsible for managing service information and will be stored in the "Service" table of the database. There are functions to search, add, edit and delete services.
- **Mechanic:** This class is the Mechanic. There are methods for saving a mechanic's username and password and for updating their profile. This class interacts with the "Mechanics" table in the database.
- **Profile:** This class is a generic class for managing user profiles, with username, password and password confirmation attributes. It has a method that updates the configuration file, which interacts with the "Customers" table. This indicates that it is used for both customer and mechanic profiles.

Methods Explanation

- **GetAssignedTasks(string mechanicUsername) (in ServiceTask):** Retrieves a list of service tasks assigned to a specific mechanic from the database.
- **RemoveTask(string appointmentId) (in ServiceTask):** Removes a service task from the database.
- **LoadServiceInformation() (in ServiceInformation):** Retrieves all service information from the database.
- **GetServiceById(int serviceId) (in ServiceInformation):** Retrieves service information for a specific service ID.
- **AddServices(...) (in ServiceInformation):** Adds a new service to the database.
- **EditService(...) (in ServiceInformation):** Updates an existing service in the database.
- **DeleteService(int serviceId) (in ServiceInformation):** Deletes a service from the database.
- **GetServiceType() (in ServiceInformation):** Retrieves a list of distinct service types from the database.
- **UpdateProfile(string newUsername, string newPassword) (in Mechanic):** Updates the mechanic's profile in the database.
- **SaveToDatabase() (in Mechanic):** Saves the mechanic's profile to the database.
- **GetPartsInventory() (in PartsInventory):** Retrieves the entire parts inventory from the database.
- **UpdatePartUsage(int partId, int usedQuantity) (in PartsInventory):** Updates the quantity of a part in the inventory after usage.
- **RequestPart(int partId, int requestQuantity) (in PartsInventory):** Increases the quantity of a part in the inventory, simulating a request.

- **UpdateProfile(string username, string passwordText) (in Profile):** Updates the password for a user in the database.

Objects Explanations

The code snippets primarily define classes and methods, not specific objects. Objects would be created (instantiated) from these classes when the program runs. For example, a Mechanic object would represent a specific mechanic with their own username, password, and potentially other data.

7. Conclusion

In conclusion, this application shows a solid basis in database organization while successfully meeting the needs of administrators, receptionists, clients, and technicians. Effective data processing and interconnection are ensured by the database's structure, which permits smooth table interaction. There are, nevertheless, certain areas that need improvement. The user experience may be impacted by the user interface's lack of visual appeal and consistency. Another issue is security because the system does not fully terminate forms, and if different roles use the same username and password, login conflicts may occur. Notwithstanding these difficulties, the application provides users with useful customization choices that allow for customized services. To solve login problems and provide a more polished and secure user experience, future enhancements will concentrate on strengthening security protocols, improving the user interface, and putting in place a strong role-based authentication system.

8. References

OpenAI. (2023). *ChatGPT* (Feb 13 version) [Large language model]. https://chat.openai.com

W3Schools.com. (n.d.). https://www.w3schools.com/cs/index.php

Iqbal, Z. (2022, August 31). *An Introduction to Microsoft Visual Studio IDE*. Bridgeall. https://www.bridgeall.com/2022/08/25/an-introduction-to-microsoft-visual-studio-ide/

Visual Studio Code - Code editing. Redefined. (2021, November 3). https://code.visualstudio.com/

Del Sole, A. (2021). Visual Studio Code distilled. In *Apress eBooks*. https://doi.org/10.1007/978-1-4842-6901-5

9. Workload Matrix

Student Name	**Lee Yi Chern**	**Ho Kun Yuan**	**Hwang XiaoShun**	**Ibraheem Shiraz Omar**
Student TP	**(TP081340)**	**(TP080482)**	**(TP077723)**	**(TP074191)**
Admin (code)	/			
Receptionist (code)				/
Customer (code)		/		
Mechanic (code)			/	
Database	/			
Front page	/			
Admin Storyboard	/			
Receptionist Storyboard				/
Customer Storyboard		/		
Mechanic Storyboard			/	
Use-case Diagram				/
Class Diagram	/	/	/	/
Conclusion	/			
References	/	/	/	/
Workload Matrix	/			
Contribution percentage (%)	25%	25%	25%	25%
Signature