

بسم الله الرحمن الرحيم

Kingdom of Saudi Arabia
Ministry of Education
College of Computer
Department of Computer Science



المملكة العربية السعودية
وزارة التعليم
كلية الحاسب
قسم علوم الحاسب

Visual Programming - CS342

Semester:461

Project Report

Car Rental Software

Muhannad Majed Al-Fawzan | 431108107 | Group:346

Mohammed Mansour Al-Bulihy | 431107867 | Group:346

Eyad Fahad Al-Arfaj | 431107974 | Group:348

Amer Mohammed Al-Senani | 431109292 | Group:346

Mohammed Saleh Al-Saif | 431108031 | Group:346

Faisal Qasim Saleh Thabit | 422117041 | Group:346

Supervisor: Dr.Walid Karamti

Table of Contents

1. Application Overview.....	2
2. GUI Elements.....	3
3. Database Connection.....	17
4. Exception Handling.....	19
5. Testing and Validation.....	20

1. Application Overview

The Car Rental Management System is a software application designed to make renting cars easier for both customers and administrators. It provides a simple and easy-to-use interface that allows users to perform all necessary actions efficiently.

For customers, the system lets them browse available cars, filter them by type or rental dates, and book the car they want. Customers can also view their current bookings, get detailed invoices, and receive notifications, such as reminders to return a car.

For administrators, the system provides tools to manage the cars, track bookings, and generate reports about revenue, customer activity, and overall performance. Admins can also view customer details.

The system includes a login feature to ensure secure access and provides different features based on whether the user is an admin or a customer. It is connected to a database that updates in real-time, so all information is always accurate and up-to-date. The Car Rental Management System is a simple yet effective tool to make the car rental process smoother and more organized.

Click [here](#) to check the repository in GitHub.

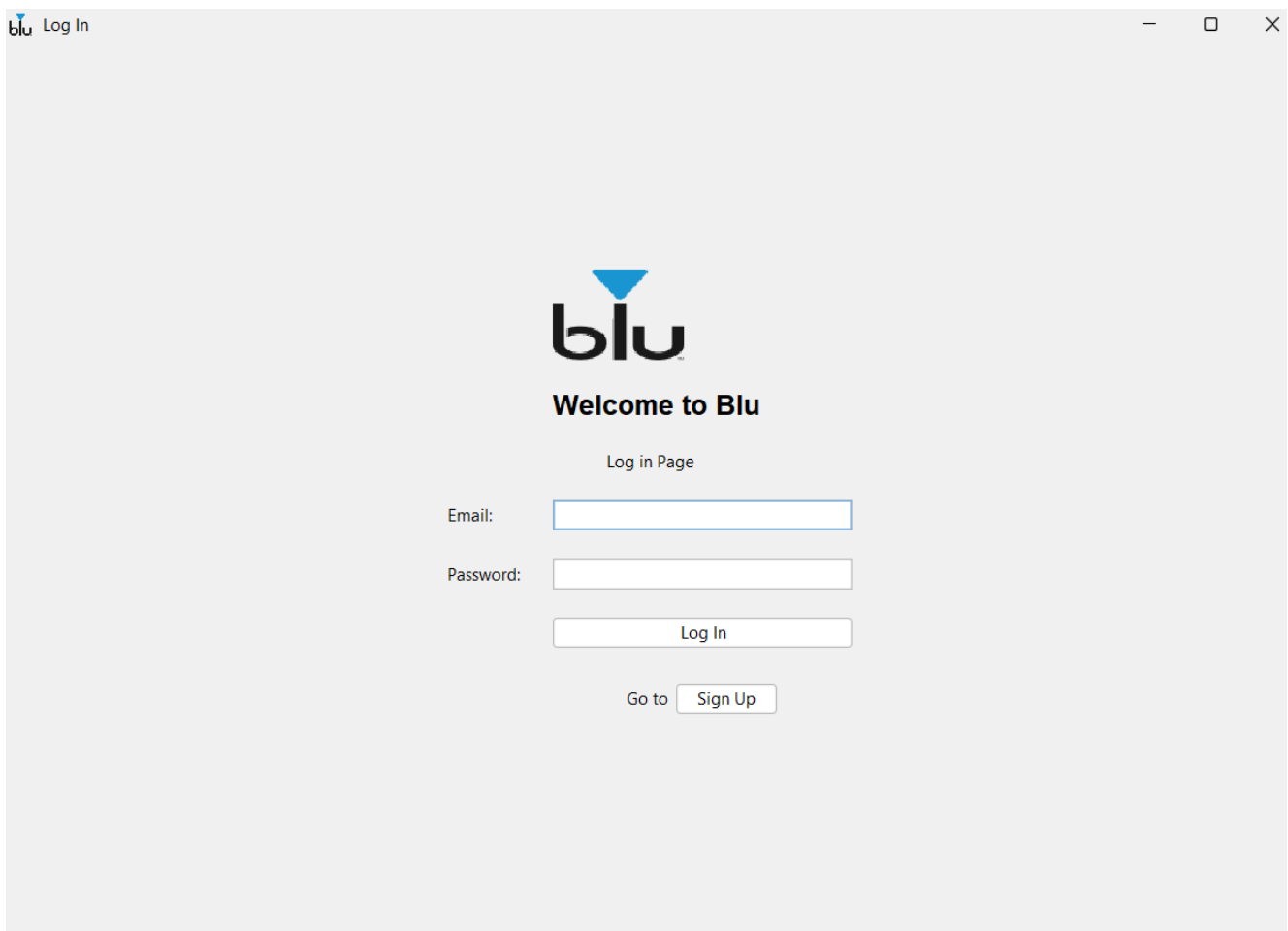


Fig. 1 (Login page GUI)

2. GUI Elements

Our program's GUI is designed to be informative, simple, and user-friendly. To enhance efficiency, we avoided using multiple JFrames and chose to use CardLayout with JPanels instead. CardLayout allows for seamless switching between different panels within a single container, making it ideal for creating a step-by-step interface or navigating between various views. Each panel is added to the layout with a unique identifier, enabling smooth transitions and reducing the need for multiple windows.

The user interface is divided into two main sections: a side navigation bar and a central content panel. The side navigation bar features three icon buttons: Account Page, Browse Vehicles, and My Bookings. When a user clicks on any of these buttons, the content panel seamlessly updates to display the corresponding screen, enabled by the efficiency of the 'CardLayout'. The content panel, which is a 'CardLayout', hosts all three pages and switches between them as needed. In fig.2, the orange rectangle locates the contentPanel and the purple locates the side navigation bar.

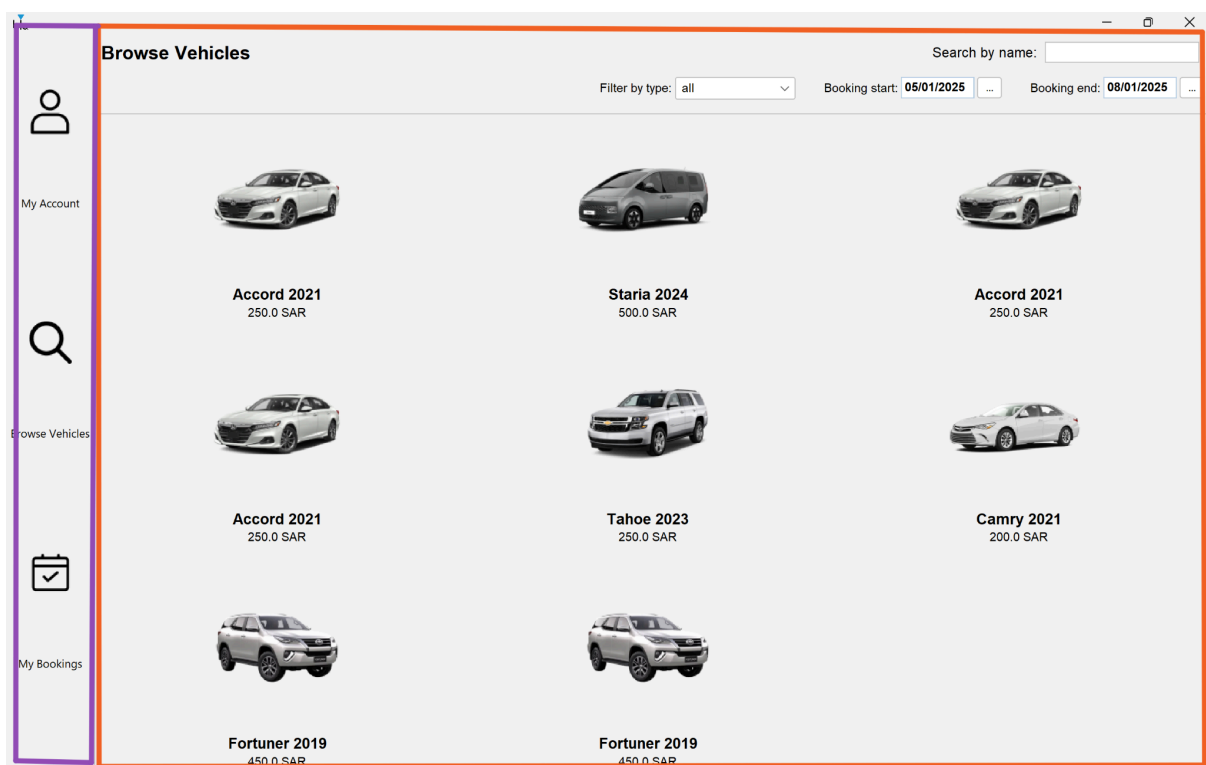


Fig. 2 (User Interface)

The Admin Dashboard is designed to be both informative and efficient, following a similar UI structure to the user interface. The 'AdminDashboard' JFrame is divided into two main sections: the navigation bar and the content panel. The navigation bar, located at the top (BorderLayout.NORTH), contains navigation buttons and quick actions like "Logout" and "Dark Mode". The content panel, which is a 'CardLayout', hosts various functional panels such as 'ManageVehicles' and 'ManageBookings'. When a user clicks a navigation button, the content panel seamlessly switches to the corresponding screen, ensuring a smooth and responsive experience.



Admin Dashboard								
Manage Vehicles		Manage Bookings		Reports		Customers		Logout
Image	ID	Name	Type	Price-per-day	Color	Year	Serial Number	
	11	Accord	Mid-Sedan	250.0	White	2021	1A253D	
	12	Staria	Minivan	500.0	Black	2024	2S3D4C	
	13	Elantra	Compact Sedan	190.0	Grey	2023	9Z0K8U	
	14	Accord	Mid-Sedan	250.0	White	2021	1A254B	
	15	Accord	Mid-Sedan	250.0	White	2021	1A253A	
	16	Maxcruz	SUV	300.0	White	2015	3UI32A	
								

Fig. 3 (Admin Dashboard Interface)

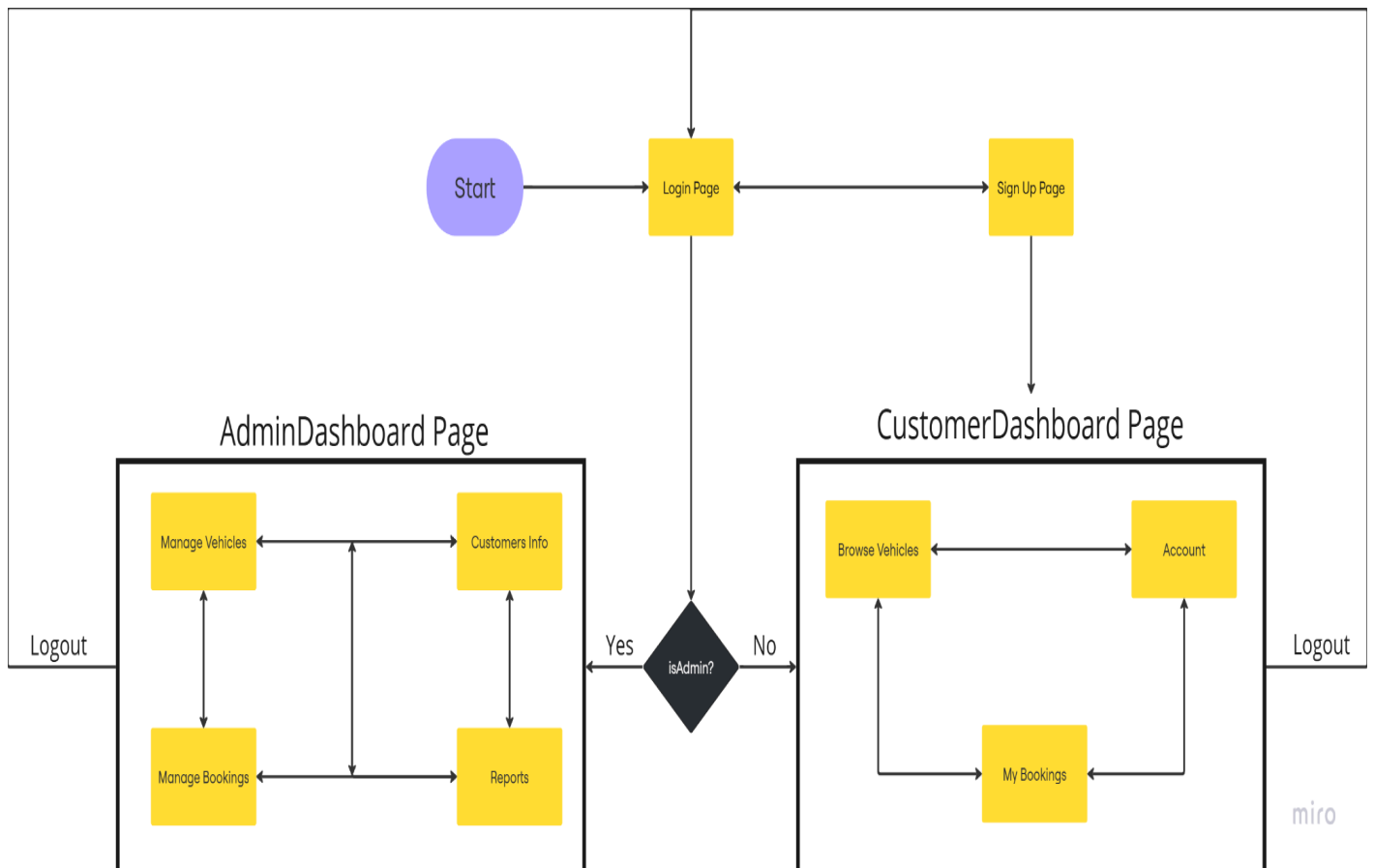


Fig. 4 (Flow Diagram for The Different Interfaces of The System)

GUI Elements:

1. **Buttons:**
 - Used for actions like booking a vehicle, confirming agreements, refreshing data, canceling bookings, logging in, and signing up.
2. **Text Fields:**
 - Allow users to input information such as email, phone numbers, names, and passwords during login or registration.
3. **Combo Boxes:**
 - Provide drop-down options for selecting vehicle types, colors, or car models.
4. **Tables:**
 - Display data like bookings, invoices, or vehicle details in an organized, tabular format.
5. **Labels:**
 - Display static text or dynamic information like revenue, notifications, or user details.
6. **Panels:**
 - Divide the interface into sections to organize different functionalities like reports, vehicle management, or user accounts.
7. **JLists:**
 - Used for displaying notifications in a scrollable list.
8. **Date Pickers:**
 - Allow users to select dates for booking start and end periods.

Each element ensures a user-friendly experience and helps efficiently navigate the system's functionality.

GUI Elements For Each Page:

Log In Page:

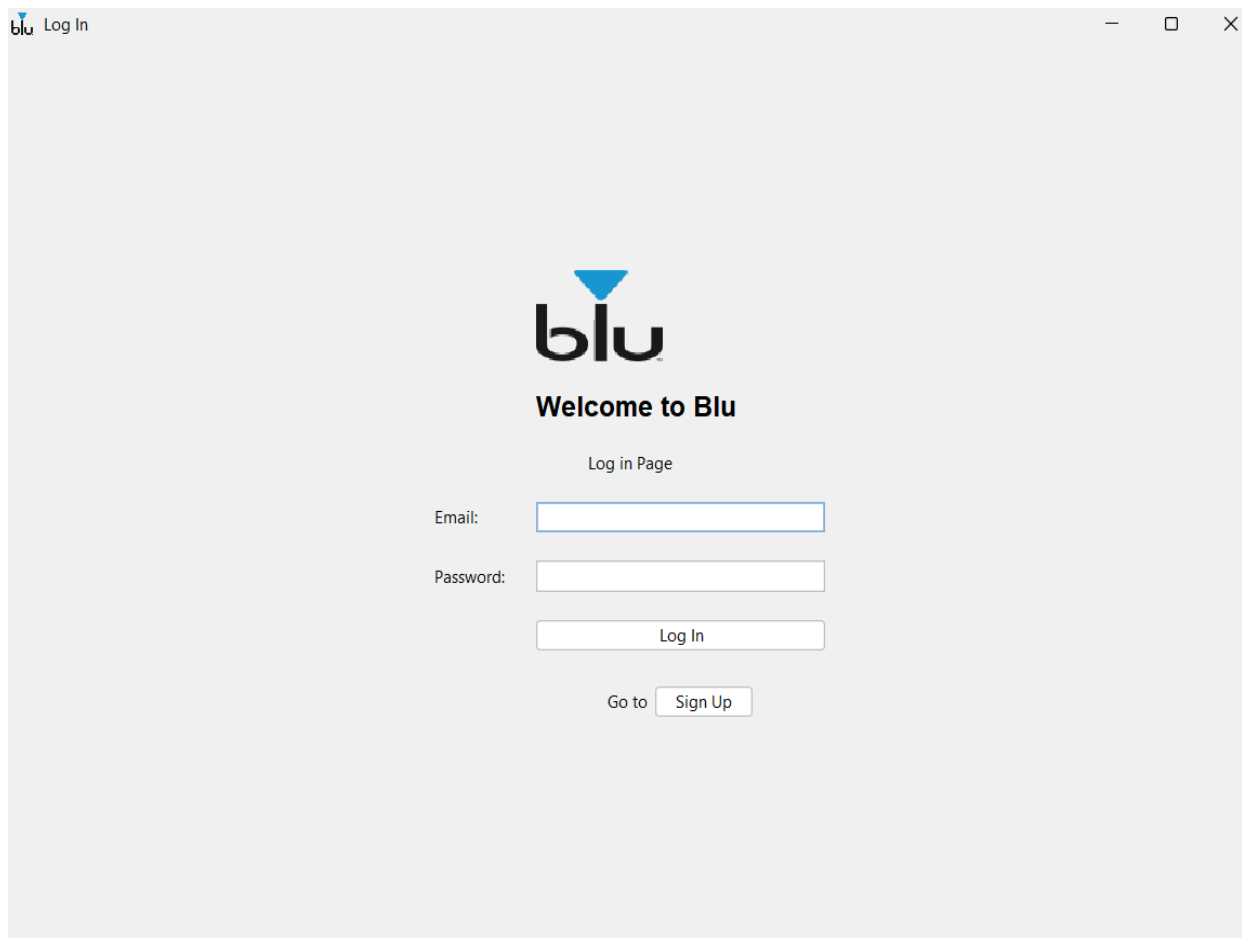


Fig. 5 (Log In Panel)

Buttons:

Log in: Triggers the login process. Validates the input data. Registers the user if all validations pass. If successful: If the user is an admin, redirects to the Admin Dashboard. Otherwise, redirects to the User UI Window.

Sign Up: Switch to the Sign Up page.

Text Fields:

Labeled Email: Holds the user input as a string, Checked for proper format.

Labeled Password: Holds the user input as an encrypted string. Must be at least 6 characters long and contain only letters and numbers.

Labels:

Welcome Message: Shows "Welcome to Blu" as a friendly greeting for new users.

Title: Displays "Sign Up Page" to indicate the purpose of the page.

Sign Up Page:

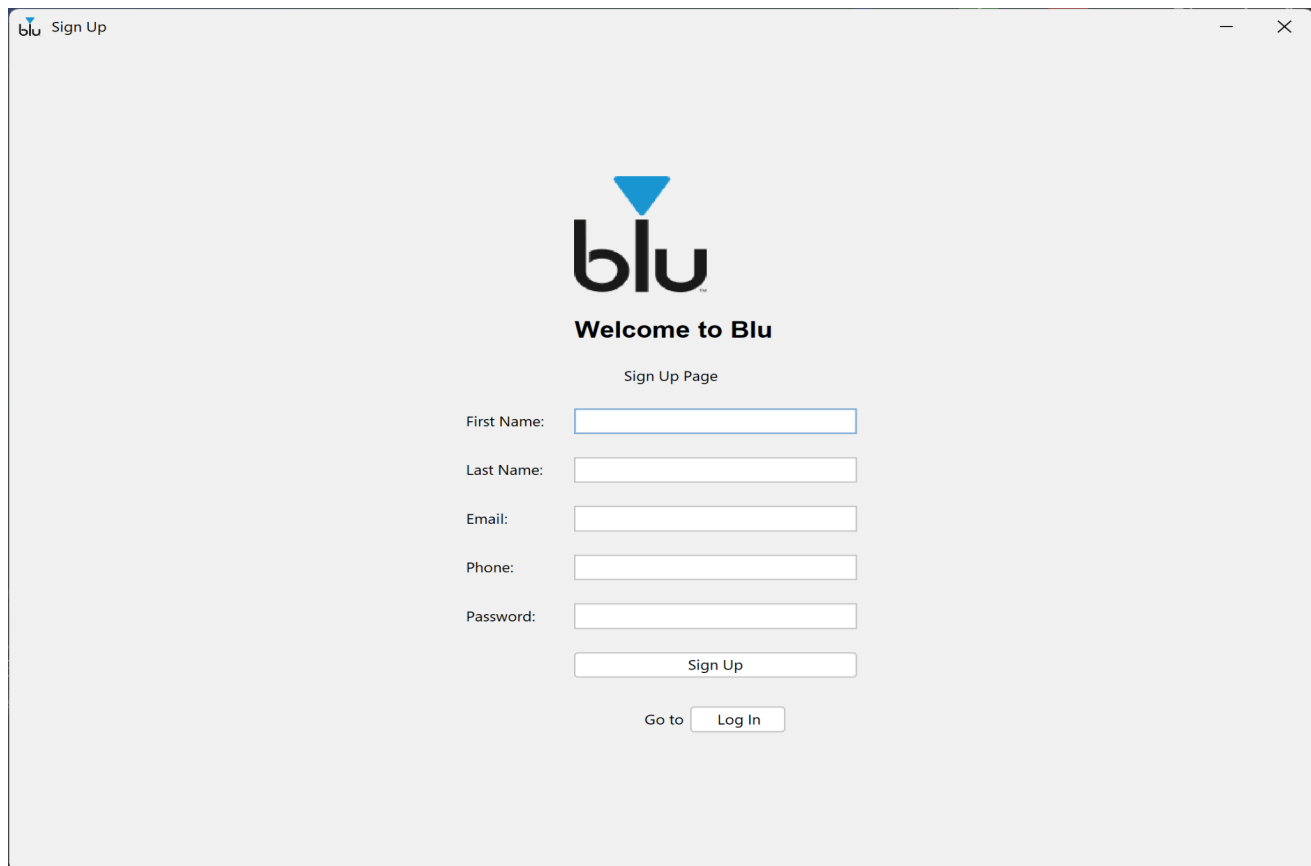
A screenshot of a web browser window titled "Sign Up". The page features the "blu" logo at the top center, which consists of a blue downward-pointing triangle above the lowercase letters "blu". Below the logo, the text "Welcome to Blu" is displayed in a bold, black font. Underneath this, the text "Sign Up Page" appears in a smaller, regular font. The main content area contains five text input fields, each preceded by a label: "First Name:", "Last Name:", "Email:", "Phone:", and "Password:". These labels are in a regular black font. Below the "Password:" field is a "Sign Up" button with a light gray background and black text. At the bottom of the form area, there is a "Go to" label followed by a "Log In" button, also with a light gray background and black text. The browser window has a standard title bar with a minus sign and a close button (X) in the top right corner.

Fig. 6 (Sign Up Panel)

Labels:

Welcome Message: Shows "Welcome to Blu" as a friendly greeting for new users.

Title: Displays "Sign Up Page" to indicate the purpose of the page.

Go to: a message to guide the user.

Text Fields:

Labeled First Name: Holds the user input as a string. Must contain only alphabetic characters.

Labeled Last Name: Holds the user input as a string. Must contain only alphabetic characters.

Labeled Email: Holds the user input as a string. Checked for proper format.

Labeled Phone: Must be exactly 10 digits.

Labeled Password: Holds the user input as an encrypted string. Must be at least 6 characters long and contain only letters and numbers.

Buttons:

Sign Up: Triggers the signup process. Validates the input data. Registers the user if all validations pass. If successful: redirects to the User UI Window.

Log In: Switch to the Log In page.

Account Page:

Account Page

My Account

First Name

Eyad

Last Name

Fahad

Email

eyad@gmail.com

Browse Vehicles

Password

123456

Phone Number

0540090569

My Bookings

[Save Changes](#) [Legacy UI](#)

Notifications

- * Your booking number #130 has been accepted
- * Your booking number #131 has been accepted
- * Your booking number #132 has been accepted
- * Your booking number #133 has been accepted

[Log out](#)

Fig. 7 (User Account Panel)

Labels:

Title: The header "Account Page" identifies the currently visible section.

Text Fields:

First Name: Presents the user's first name and allows the user to edit their data.

Last Name: Presents the user's last name and allows the user to edit their data.

Email: Only presents the user's email.

Password: Presents the user's password and allows the user to edit their data.

Phone Number: Presents the user's phone number and allows the user to edit their data.

Clickable Labels:

Save Changes: A clickable label that becomes active (blue color) when user information changes. Clicking it saves updates to the backend.

Legacy UI Link: A clickable label to switch to a legacy version of the interface.

Notifications Panel (Right Panel):

Label Title: "Notifications" indicates this section is for user alerts.

Content: Displays notifications dynamically, such as booking updates (e.g., "Your booking number #107 has been accepted").

Clickable Label Logout Link: Located at the bottom of the notifications panel.

Clicking it logs the user out and navigates to the login screen.

Buttons:

My Account: Goes to **My Account** page.

Browse Vehicles: Goes to **Browse Vehicles** page.

My Bookings: Goes to **My Bookings** page.

Browse Vehicles Page:

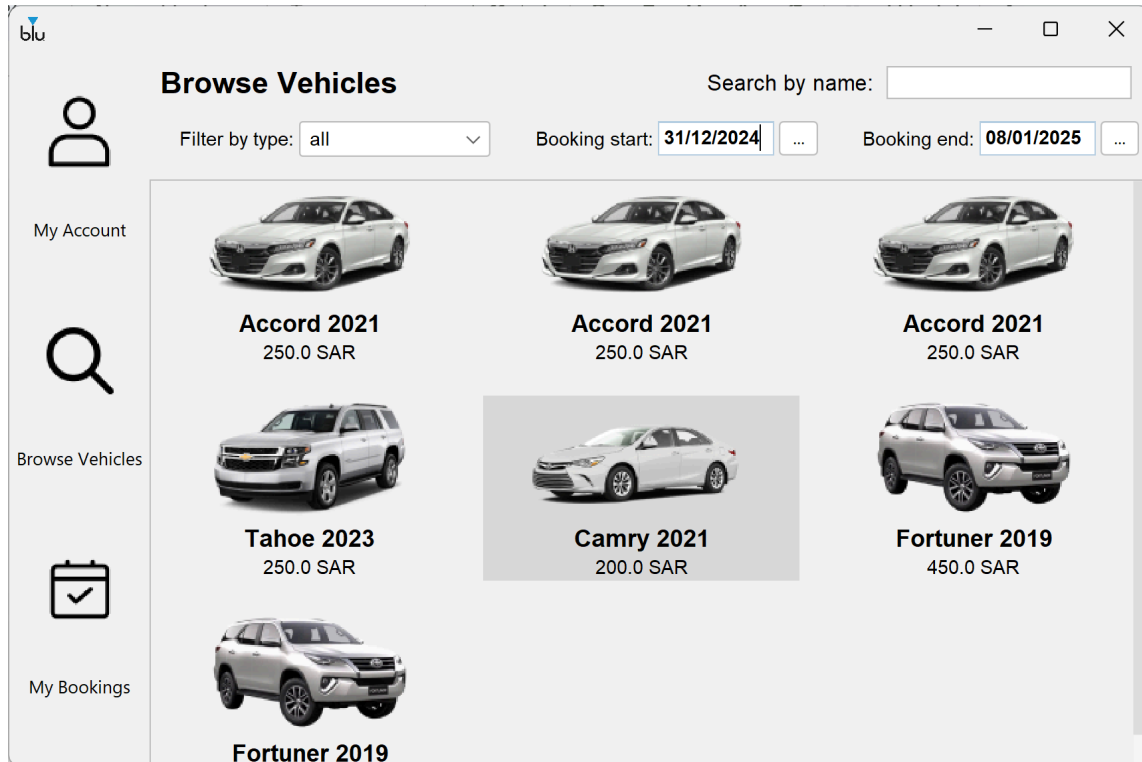


Fig. 8 (User Browse Vehicles panel)

Labels:

Title: Displays "Browse Vehicles" as the title of the current screen.

Text Fields:

Labeled Search by Name Field: A text box for searching vehicles by name. Updates the vehicle list dynamically as the user types.

Booking Date Fields:

Booking Start: Allows the user to select a start date for their booking.

Booking End: Allows the user to select an end date for their booking.

Both fields use a date picker with restrictions:

Start date must be from today to 60 days in the future.

End date must be after the start date and within 60 days.

Combo Box:

Labeled Filter by Type Dropdown: A combo box to filter vehicles by their type (e.g., Sedan, SUV, etc.). Default option is "all," which shows all available vehicles.

Grid List:

Vehicle Grid List: Displays the available vehicles in a grid format with:

Vehicle Image: Shows a picture of the vehicle.

Vehicle Name: Displays the vehicle's name and model year.

Price: Shows the rental price per day.

Interactive Feature: Hovering over a vehicle highlights the card for better visibility.

Clicking on a vehicle card navigates to a detailed page about that vehicle.

Scroll Pane: Ensures the vehicle grid list is scrollable for better usability when the list is extensive.

Booking Done Page:

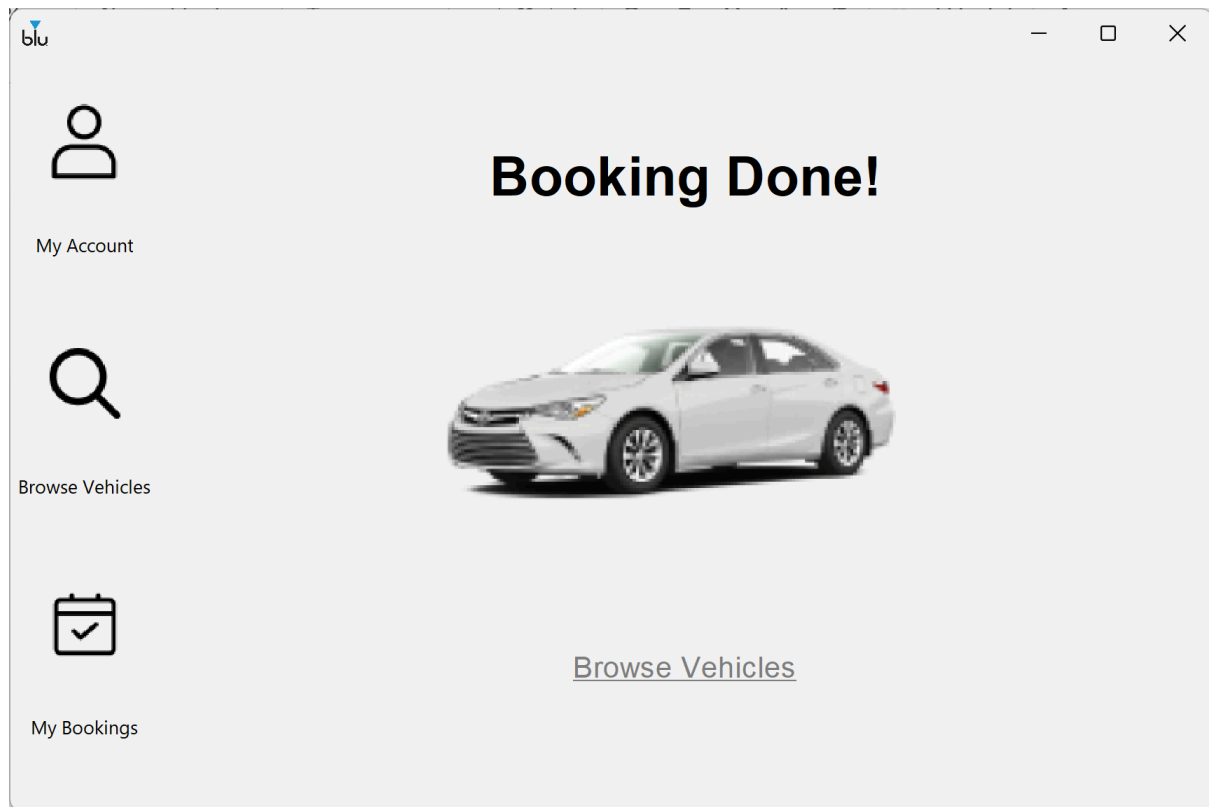


Fig. 9 (User Booking Confirmation panel)

Labels:

Booking Done! Message: A bold, centered label ("Booking Done!") informs the user that their booking was successful.

Styled with a large font size for clear visibility.

Car Image:

Displays an image of the booked vehicle.

Clickable Labels:

Browse Vehicles Link: A clickable label to navigate back to the Browse Vehicles page. Styled with an underline and changes to bold on hover for interactivity.

Buttons:

My Account: Goes to **My Account** page.

Browse Vehicles: Goes to **Browse Vehicles** Page.

My Bookings: Goes to **My Bookings** page.

Car Details Page:

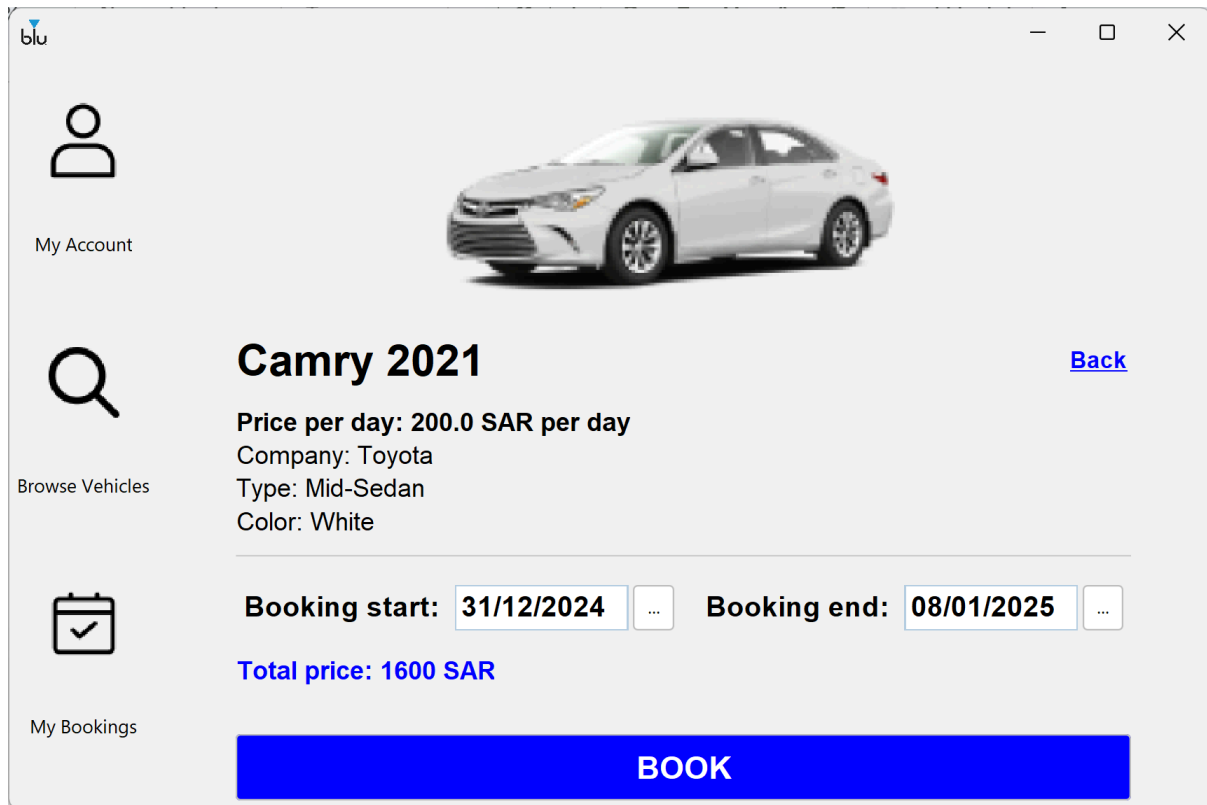


Fig. 10 (User Car Details panel)

Vehicle Image:

Displays a high-quality image of the selected vehicle.

Labels:

Vehicle Name: Shows the name and model year of the selected car (e.g., "Camry 2021").

Price per day: Displays the rental price per day for the vehicle.

Company: Shows the vehicle manufacturer (e.g., Toyota).

Type: Indicates the type of vehicle (e.g., Mid-Sedan).

Color: Displays the color of the car.

Total Price: Displays the calculated total rental cost based on the selected dates.

Updates in real-time as the user changes the booking dates.

Clickable Labels:

Back: A clickable label that navigates back to the Browse Vehicles screen.

The text "Back" is styled with an underline and changes font size when hovered.

Buttons:

Book: A prominent, blue "BOOK" button that confirms the booking.

Clicking it navigates to the Rental Agreement screen, passing the selected vehicle and booking details.

Booking Dates Fields:

Start Date: A date picker allowing the user to select the booking start date. Dates are restricted to today or up to 60 days ahead.

End Date: A date picker for the booking end date. It must be after the start date and within the next 60 days.

Both date pickers dynamically update the total price when dates are changed.

My Bookings Page:

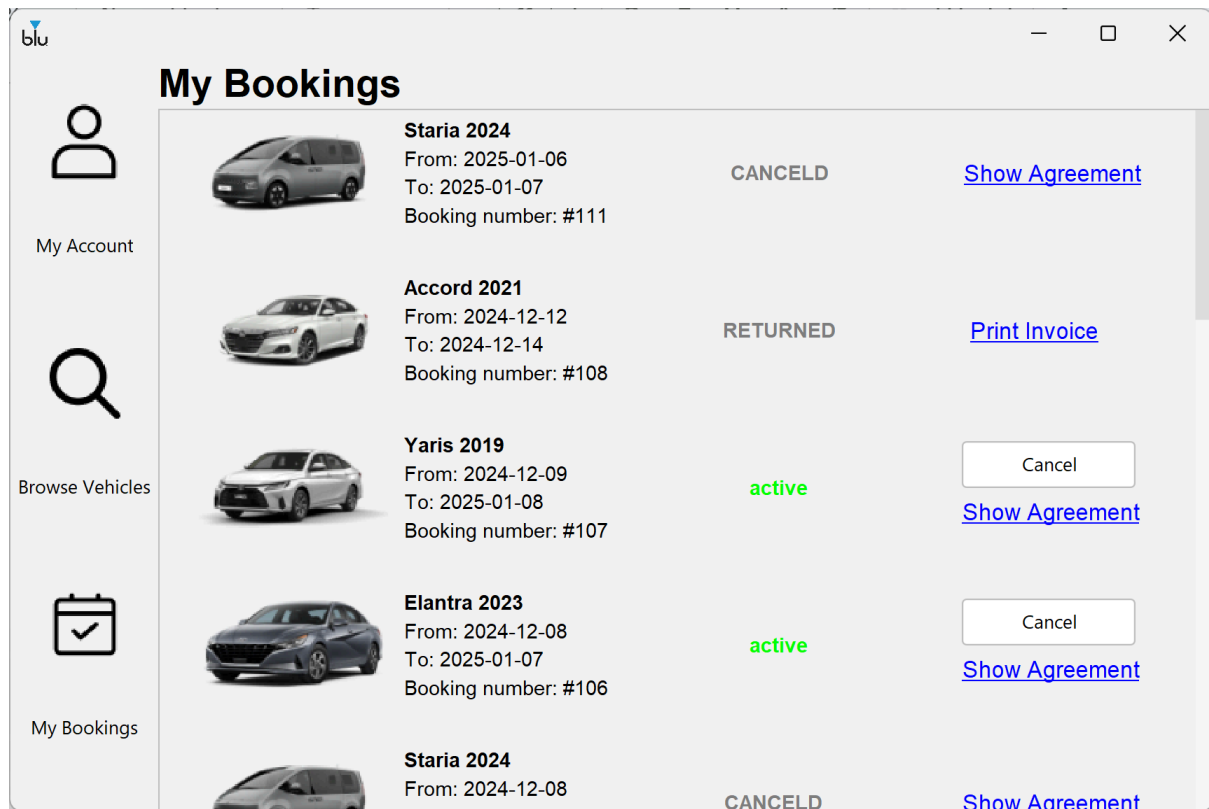


Fig. 11 (User Bookings Details panel)

Labels:

Title: Displays "My Bookings" to indicate the current section.

Vehicle Details: Name and model year.

Label Booking status: (e.g., active, canceled, returned) displayed in a color-coded label:

Green: Active bookings.

Gray: Canceled or returned bookings.

Booking number and duration: (start and end dates).

Booking List (Scrollable Panel)

Dynamic Booking Entries:

Each booking is displayed as a horizontal card, including:

Vehicle Image: Shows a picture of the booked vehicle.

Vehicle Details: Name and model year.

Label Booking status: (e.g., active, canceled, returned) displayed in a color-coded label:

Green: Active bookings.

Gray: Canceled or returned bookings.

Booking number and duration: (start and end dates).

Button:

Cancel: Available for active bookings, allowing the user to cancel the booking.

Print Invoice: Available for returned bookings with an invoice, allowing the user to print the

Clickable Label Show Agreement: Opens a detailed rental agreement for the booking.
invoice.

AdminDashboard Page-Manage Vehicle:

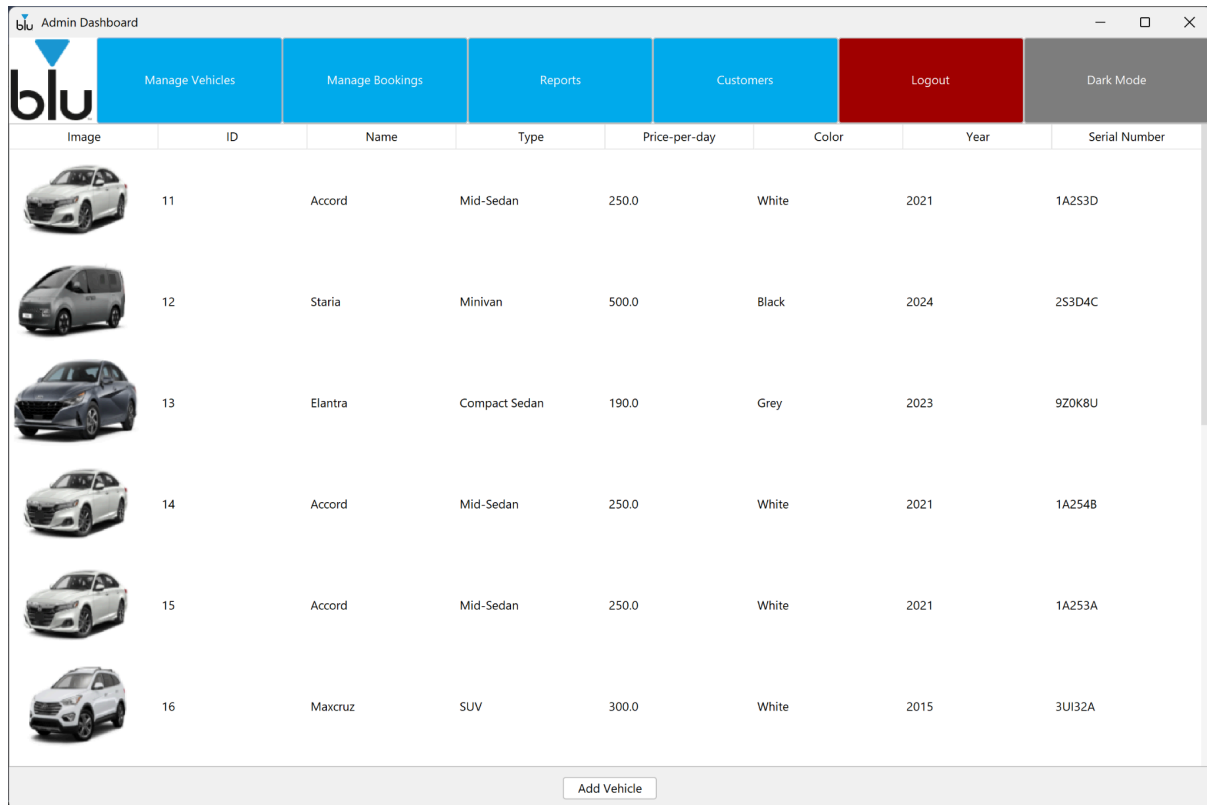








Image	ID	Name	Type	Price-per-day	Color	Year	Serial Number
	11	Accord	Mid-Sedan	250.0	White	2021	1A253D
	12	Staria	Minivan	500.0	Black	2024	2S3D4C
	13	Elantra	Compact Sedan	190.0	Grey	2023	9Z0K8U
	14	Accord	Mid-Sedan	250.0	White	2021	1A254B
	15	Accord	Mid-Sedan	250.0	White	2021	1A253A
	16	Maxcruz	SUV	300.0	White	2015	3UI32A

Add Vehicle

Fig. 12 (Admindashboard Manage Vehicles panel)

Tabs (Panels):

Manage Vehicles: The main vehicle management section.

Manage Bookings: Highlights the current section for managing bookings.

Reports: Opens the reports section.

Customers: Navigates to customer management.

Buttons:

Manage Vehicles: Goes to **Manage Vehicles** tab.

Manage Bookings: Goes to **Manage Bookings** tab.

Reports: Goes to **Reports** tab.

Customers: Goes to **Customers** tab.

Logout: Logs the admin out of the dashboard to the Log In page.

Dark Mode: Toggles between light and dark themes for better visual preference.

labeled Add Vehicle: Toggles Add vehicle page to add a new vehicle.

The navigation bar allows admins to switch between different sections seamlessly.

Search Panel (Top of Booking Section)

Table:

Displays all Cars in a tabular format with the following columns:

Image: display car's image.

ID: Unique identifier for the booking.

Name: displays car's name.

Type: displays car's type.

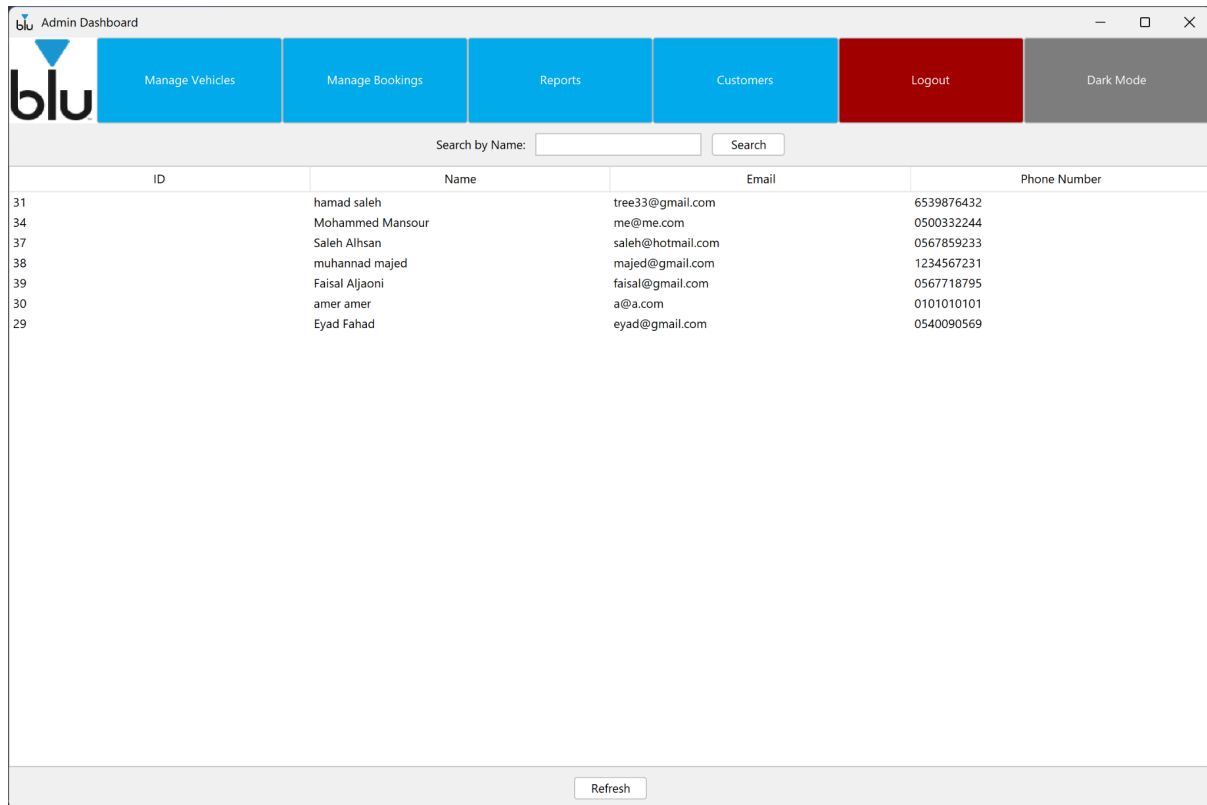
Price-per-day: displays car's price per day.

Color: displays car's color.

Year: displays car's year.

Serial Number: displays car's serial number.

AdminDashboard Page-Customers



ID	Name	Email	Phone Number
31	hamad saleh	tree33@gmail.com	6539876432
34	Mohammed Mansour	me@me.com	0500332244
37	Saleh Alhsan	saleh@hotmail.com	0567859233
38	muhammad majed	majed@gmail.com	1234567231
39	Faisal Aljaoni	faisal@gmail.com	0567718795
30	amer amer	a@a.com	0101010101
29	Eyad Fahad	eyad@gmail.com	0540090569

Fig. 13 (Customers Panel in Admindashboard)

Buttons:

Manage Vehicles: Goes to **Manage Vehicles** tab.

Manage Bookings: Goes to **Manage Bookings** tab.

Reports: Goes to **Reports** tab.

Customers: Goes to **Customers** tab.

Logout: Logs the admin out of the dashboard to the Log In page.

Dark Mode: Toggles between light and dark themes for better visual preference.

Search: A Search Button triggers a search based on the entered names. Using Search by Name Text field input.

Refresh: Reloads and refreshes the customer data displayed in the table.

Clears any search results and resets the table to show all customers.

Text field:

Search by Name: A text field for entering the customer name to search for.

A Search Button triggers a search based on the entered name.

Tables:

Customers Table (Center Panel):

Displays all customers in a tabular format with the following columns:

ID: Unique identifier for each customer.

Name: Full name of the customer.

Email: Customer's email address.

Phone Number: displays user's phone number.

AdminDashboard Page-Manage Bookings

ID	Customer Name	Car ID	Start Date	End Date	Status	Cost
86	Eyad Fahad	12	2024-12-07	2024-12-09	RETURNED	1000.0
129	Eyad Fahad	11	2024-12-07	2024-12-08	RETURNED	250.0
98	amer amer	13	2024-12-08	2025-01-07	active	5700.0
100	Eyad Fahad	14	2024-12-08	2025-01-07	CANCELLED	7500.0
101	Eyad Fahad	21	2024-12-08	2024-12-09	active	450.0
105	Faisal Aljaoni	11	2025-12-23	2025-12-28	RETURNED	1250.0
87	Mohammed Mansour	19	2024-12-08	2024-12-09	RETURNED	450.0
85	Eyad Fahad	21	2024-12-07	2025-01-06	RETURNED	13500.0
114	Mohammed Mansour	17	2024-12-09	2024-12-10	active	250.0

Fig. 14 (Admindashboard Manage Bookings panel)

Buttons:

Manage Vehicles: Goes to **Manage Vehicles** tab.

Manage Bookings: Goes to **Manage Bookings** tab.

Reports: Goes to **Reports** tab.

Customers: Goes to **Customers** tab.

Logout: Logs the admin out of the dashboard to the Log In page.

Dark Mode: Toggles between light and dark themes for better visual preference.

Search: A Search Button triggers a search based on the entered ID. Using Search by ID Text field input.

Refresh: Reloads the table with the most up-to-date booking information.

Cancel Booking: Cancels the selected booking and updates its status to "CANCELED"

Mark as Returned: Marks the selected booking as "RETURNED".

Text field:

Search by ID: Input field to enter a booking ID.If the input is empty or invalid, an error message is displayed. Resets the table to show all bookings when no ID is entered.

Tables:

Bookings Table:

ID: Unique identifier for each booking.

Customer Name: The name of the customer associated with the booking.

Car ID: Identifier for the booked vehicle.

Start Date: The start date of the booking.

End Date: The end date of the booking.

Status: The current status of the booking (e.g., active, returned, canceled).

Cost: The total cost of the booking.

AdminDashboard Page-Reports

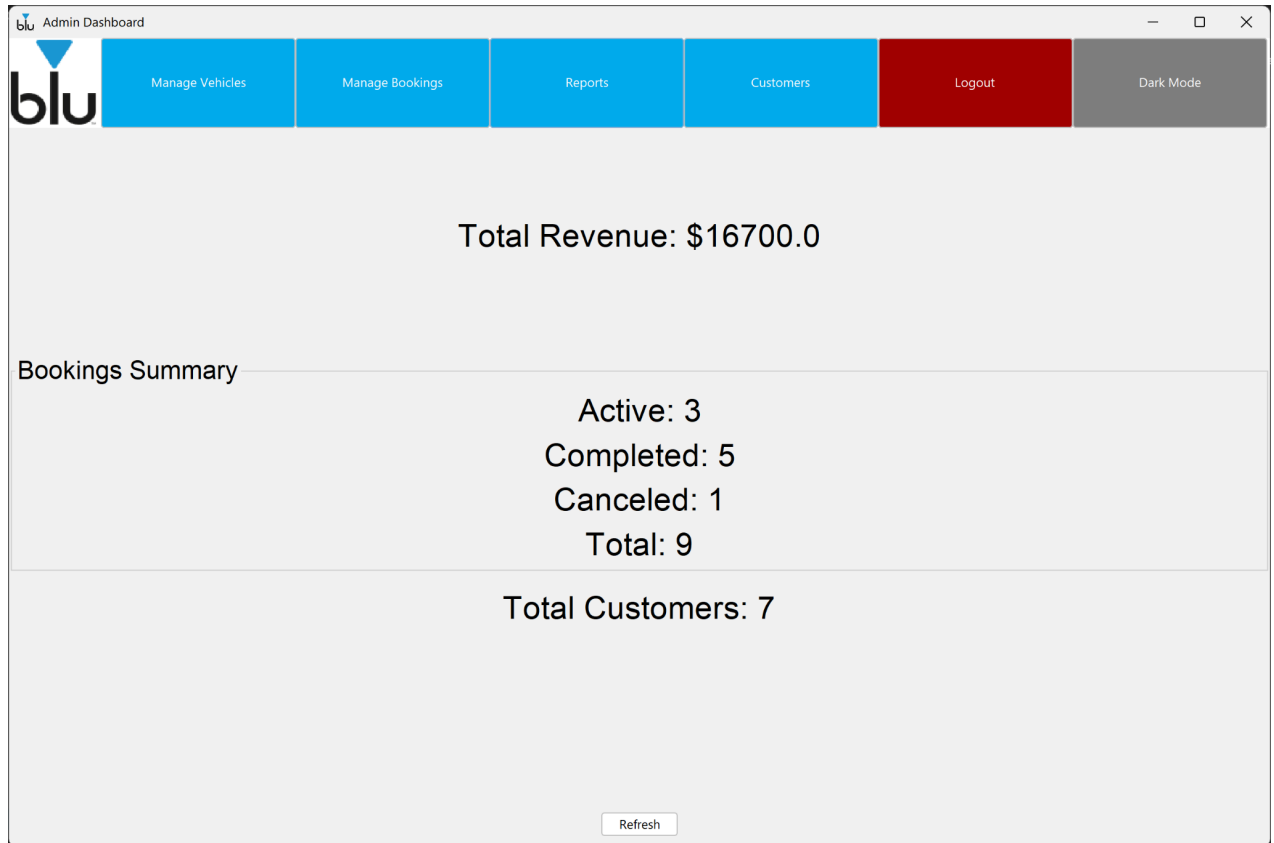


Fig. 15 (Admindashboard Reports panel)

Buttons:

Manage Vehicles: Goes to **Manage Vehicles** tab.

Manage Bookings: Goes to **Manage Bookings** tab.

Reports: Goes to **Reports** tab.

Customers: Goes to **Customers** tab.

Logout: Logs the admin out of the dashboard to the Log In page.

Dark Mode: Toggles between light and dark themes for better visual preference.

Refresh: Reloads the page with the most up-to-date report information.

Labels:

Total Revenue: Displays the total Revenue till this moment.

Active: Displays the total active bookings till this moment.

Completed: Displays the total completed bookings till this moment.

Canceled: Displays the total canceled bookings till this moment.

Total: Displays the total number of bookings till this moment.

Total Customers: Displays the total number of customers till this moment.

Booking Summary: Partitions the page for better visuals and simplicity.

3. Database Connection

In our project, we use a **PostgreSQL** database hosted on **AWS Cloud** to store and manage all the data. We chose AWS because it's secure, reliable, and provides good performance for cloud-hosted applications.

Database Connection

- The application connects to the database by opening a new connection for each query. Once the operation is completed, the connection is immediately closed to avoid resource overuse and keep the system efficient.
- We use **prepared statements** to send queries to the database. This method not only makes the queries faster but also protects against SQL injection attacks, keeping the data safe.
- A centralized class, `DatabaseHandler`, manages all interactions with the database. It takes care of:
 - Executing queries and updates.
 - Setting query parameters securely.
 - Automatically closing the connection after each operation.
- We also implemented error handling with the `ErrorHandler` class to display clear messages when something goes wrong with the database.

Controllers

To make the system organized, we divided its functionality into several **controllers**. Each controller is responsible for a specific part of the system and acts as a bridge between the user interface and the database.

- **BookingController**: Manages all booking-related operations, like creating a booking, checking if a car is available, and updating booking details.
- **VehicleController**: Handles everything related to vehicles, such as retrieving car details and dynamically fetching car models and types for the interface.
- **UserController**: Manages user accounts, including login, registration, and updates. It ensures data is validated before being saved.
- **InvoiceController**: Takes care of generating and retrieving invoices. It also works with the PDF generation feature to let users save or print their invoices.
- **AgreementController**: Handles rental agreements by creating them and ensuring all agreements follow the standard terms and conditions.
- **ReportingController**: Collects data for reports, such as revenue, bookings, and customer stats. It is optimized to handle large datasets efficiently.

How Controllers Work with the Database

Each controller uses the `DatabaseHandler` to run queries securely. This separation keeps the logic clean:

- Controllers handle the business logic and decide what data is needed or updated.
- The database handler takes care of safely running queries and returning results.

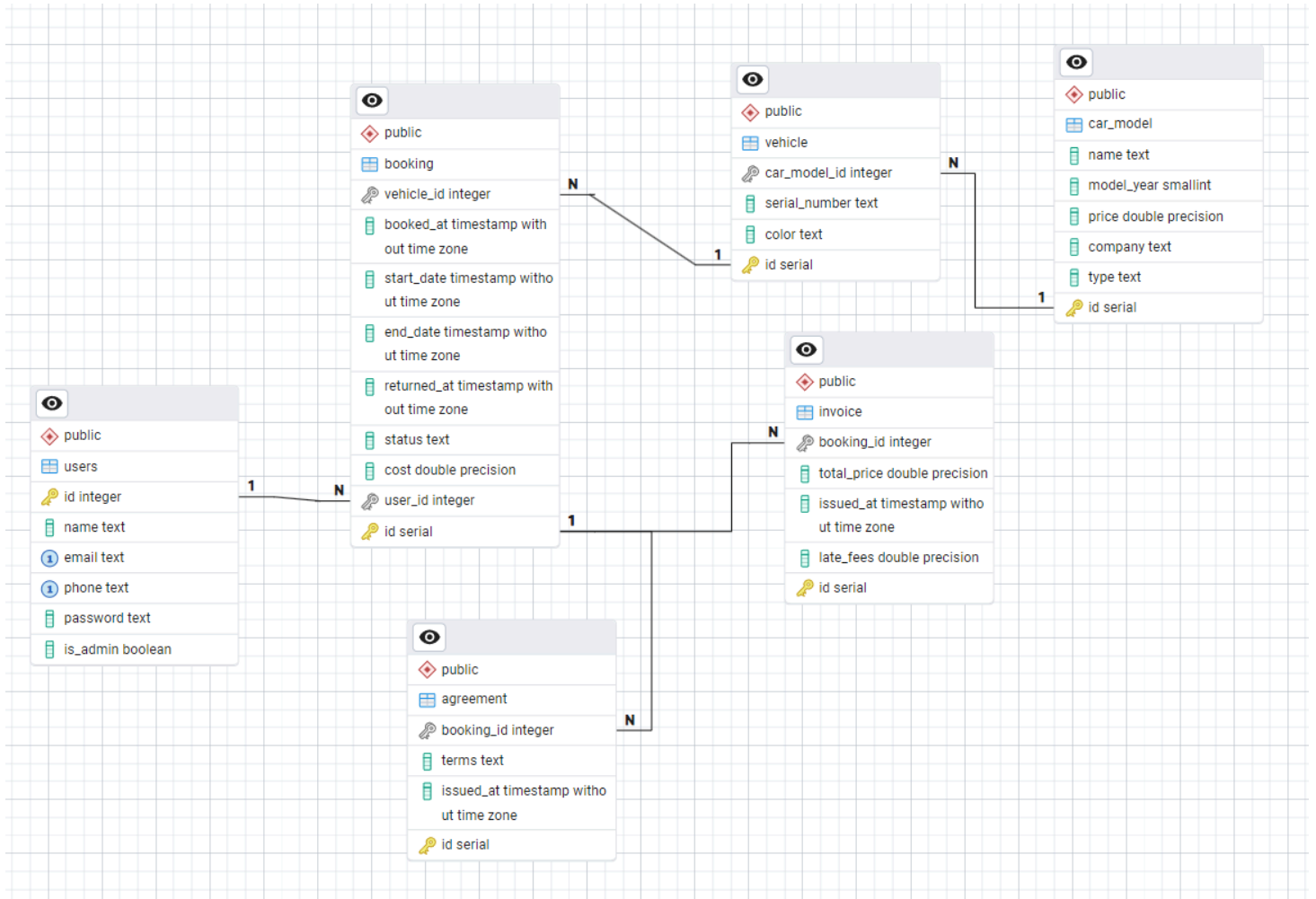


Fig. 16 (Entity Relationship Diagram of the Car Rental Software Database)

4. Exception Handling

In our project, we made sure to handle errors properly so the app works smoothly and users get clear feedback when something goes wrong. We used a special class called ErrorHandler to manage all errors in one place.

How We Handle Exceptions

1. Centralized Error Management:

- The ErrorHandler class is used throughout the app to handle errors.
- It has three main methods:
 - `handleException`: For showing errors, like when a database query fails.
 - `handleWarning`: For warnings, like trying to book a car that is already reserved.
 - `handleInfo`: For giving messages, like confirming a booking or showing successful actions.

2. Database Errors:

- If there's an issue connecting to the database or a query fails, we catch it and show an error message using `handleException`. This way, the app doesn't crash, and the user knows what went wrong.

3. User Input Validation:

- Before doing any action, we check the user's input to make sure it's valid (e.g., proper email format or non-empty fields).
- If the input is invalid, we show a warning message so the user can fix it.

4. Controller-Level Errors:

- Each controller (e.g., `BookingController`, `UserController`) has try-catch blocks to handle specific errors, like invalid booking IDs or unavailable cars.
- Controllers also send error messages back to the interface so the user knows what went wrong.

5. GUI Errors:

- In the graphical interface, we added error handling to avoid crashes when users perform invalid actions. For example:
 - If no row is selected when canceling a booking or generating an invoice, we show a warning.
 - If a user enters something not an alphabetic string in searching for customers, we show a warning.

Scenarios We Covered

- **Invalid Login:** If the email or password is wrong, the user gets a message to check their credentials.
- **Database Issues:** If the app can't connect to the database, the user is informed with a clear message.
- **Empty or Incorrect Input:** Users are asked to fix empty fields or invalid input like incorrect phone numbers.
- **Multiple Selections:** For actions like canceling bookings, only one row can be selected. If multiple rows are selected, the app warns the user.

5. Testing and Validation

We tested the application to make sure it works correctly and handles different scenarios smoothly.

How We Tested

1. **Unit Testing:** Tested individual functions like checking car availability (`CarIsBusy()`) and user registration to ensure they work as expected.
2. **Integration Testing:** Verified how controllers interact with the database, ensuring bookings and other actions update data correctly.
3. **GUI Testing:** Checked the interface, making sure buttons, forms, and panels respond as intended, such as handling invalid inputs or switching views.
4. **Real User Scenarios:** Simulated real actions like booking a car, canceling, and generating invoices to confirm everything works seamlessly.

Validation

1. **Input Validation:** Ensured emails, phone numbers, and passwords are valid before processing.
2. **Database Validation:** Verified the database updates correctly after actions like bookings and cancellations.
3. **Error Handling:** Tested with invalid inputs and system errors to ensure proper error messages are shown without crashing.

Outcome

Through this process, we ensured the application is functional, user-friendly, and reliable for real-world use.