Faculty of Computers and Artificial Intelligence
Cairo University
Assignment DB

Submitted to

Dr. Ali Zidan

Prepared by

Course Name: IS211-Introduction to Database Systems
Phase Number: Phase 2
TA Name: Fatma Hussein
Project: Car Insurance Company System
Program: General

ID	NAME	SECTION
20230650	**Ahmed Mahmoud Ibrahim**	21&22
20230157	**Zeyad Ali Azzap**	21&22
20230368	**Mahmoud Ahmed Ibrahim**	21&22
20230420	**Menna Mohammed Abdullah**	21&22
20230473	**Youmna Salah Abd El-hamid**	All
20230404	**Mostafa Mohammed Essa**	All

Table of content

Functional Requirements:

Req.ID	Requirements	Description
RQ001	Sign in Customer	The system shall allow users to sign in either as customers or administrators. Authentication shall be performed using the user's name, email, phone number, address, and password.

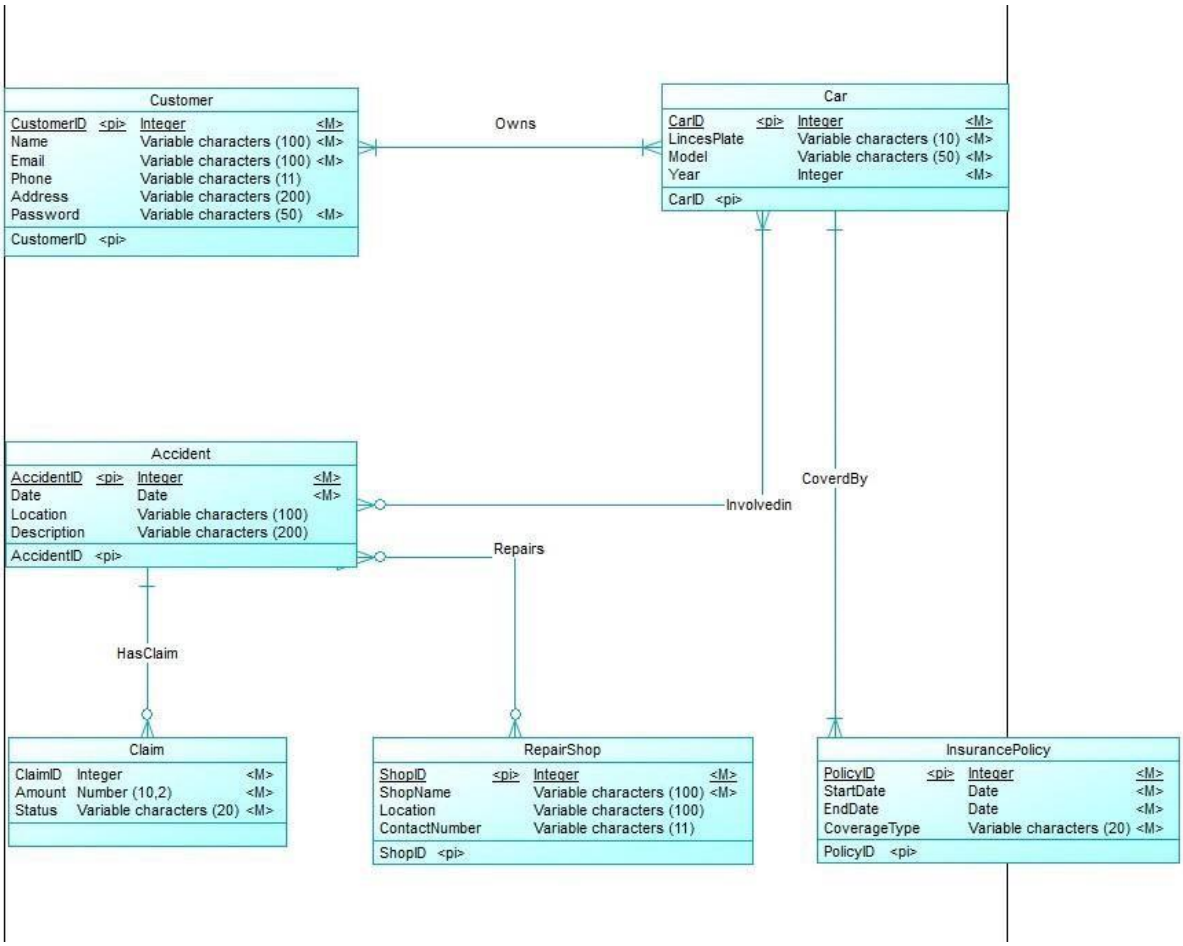
RQ002	Add Customer	The system shall allow the creation of a new customer account. A unique customer ID shall be generated, and the following details shall be stored: name, email, phone number, address, and password. Customers can be added by themselves or by an administrator.
RQ003	Update customer information	The system shall allow existing customers to update their registered information, including their name, contact details, and address.
RQ004	Delete customer	The system shall allow customers to delete their own accounts. Additionally, administrators shall have the ability to remove customer accounts from the system.
RQ005	Add new car	The system shall allow customers to register a new car either by themselves or through the administrator. Each car shall be assigned a unique identifier and store information such as license plate number, model, and manufacturing year. The system shall support the following ownership models: - A car may be owned by multiple customers. - A customer may own multiple cars.

		- A customer may own only one car, depending on business rules.

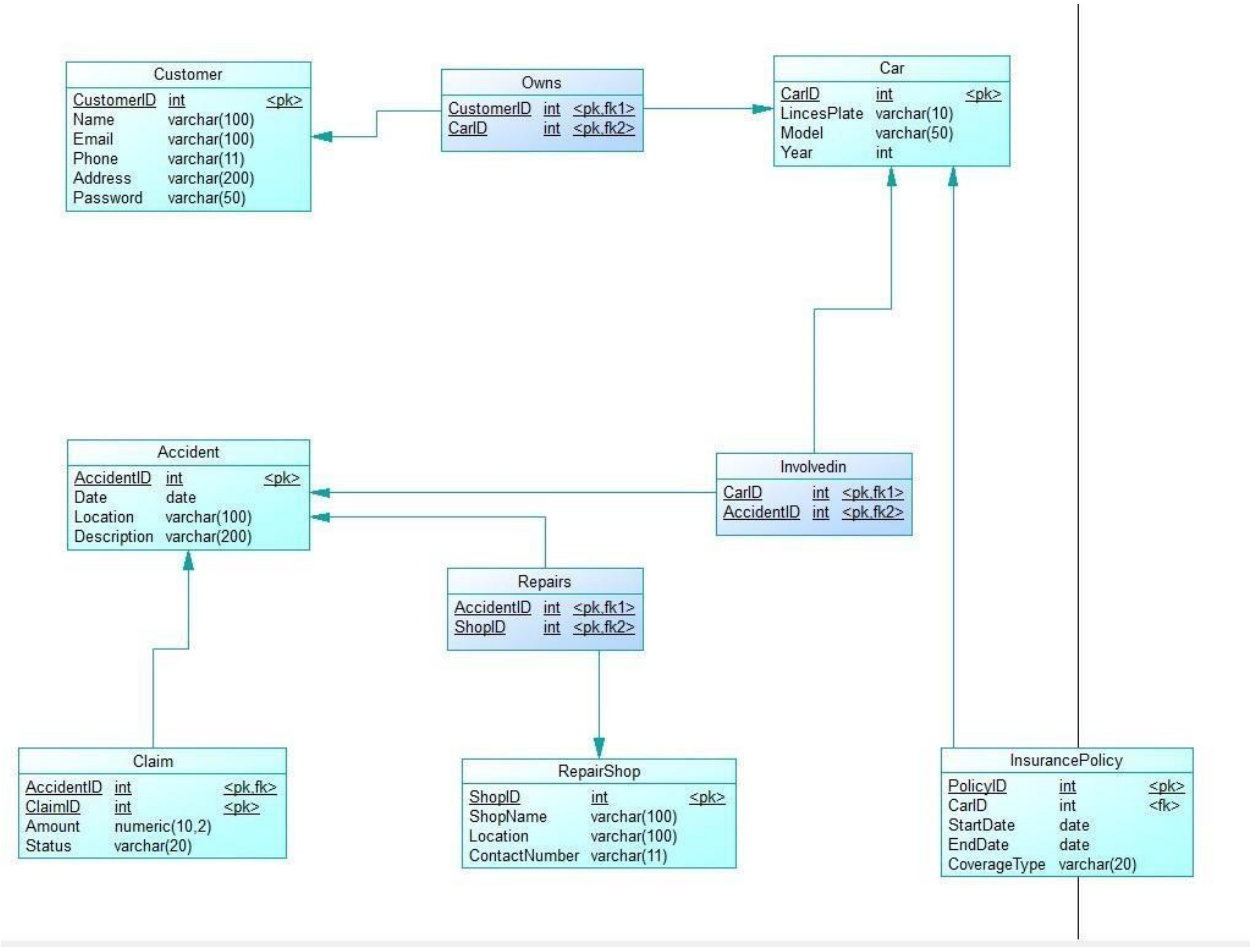
RQ006	Update car information	The system shall allow customers or administrators to modify existing car details, including the license plate number, model, and year of manufacture. The system must ensure that all ownership relationships and related records are consistently updated to reflect any changes.
RQ007	Delete car	The system shall allow either a customer or an admin to delete a saved car record. Upon deletion, the system must automatically update or remove all associated relationships and references—such as customer ownership links or dependent recordsto maintain data integrity and ensure consistency across the system.
RQ008	Add accident	In the event of an accident, the system shall allow either the customer or the admin to add detailed information about the accident. Each accident record must include a unique accident ID, the date of the accident, the location, and a description of the event. The system must also support many-to-many relationships between accidents and cars; a single accident may involve multiple cars, and a single car may be associated with multiple accident records.
RQ009	Update accident information	The system must let customers or admins edit details of an existing accident, like the date, location, or description. Any changes should keep all links between

		accidents and cars consistent, so no data gets messed up. The system needs to update all related records properly to reflect the new info.
RQ010	View accident information	The system should allow customers and admins to check out full details of an accident, including its ID, date, location, description, and a list of all cars involved. It should also let users filter accidents by stuff like date, car, or customer to quickly find what they need.
RQ011	Generate monthly reports for the total accidents	The system needs to create monthly reports that sum up all accidents for a specific month. The report should show the total number of accidents, how many cars were involved, and a breakdown by car model. Admins should be able to download it as a PDF for easy sharing or analysis.

Conceptual ERD



Physical ERD



DB Schema

1.Customer

Customers who own cars and deal with accidents and insurance.

Columns

CustomerID (PK, int)
Name (varchar(100), NOT NULL)
Email (varchar(100), NOT NULL)
Phone (varchar(11), NOT NULL)
Address (varchar(200), NOT NULL)
Password (varchar(50), NOT NULL)

2.Car

Cars owned by customers and may be covered by insurance policies.

Columns

CarID (PK, int)
LincesPlate (varchar(10), NOT NULL)
Model (varchar(50), NOT NULL)
Year (int, NOT NULL)
CustomerID (FK, int, NOT NULL)

Keys & Constraints

FK: CustomerID → Customer(CustomerID)

3.Owns

Represents the ownership relationship between customers and the cars they own.

Columns

CustomerID (FK, int, NOT NULL)
CarID (FK, int, NOT NULL)

Keys & Constraints

PK: (CustomerID, CarID)
FK: CustomerID → Customer(CustomerID)
FK: CarID → Car(CarID)

4.Accident

Accidents involving the cars.

Columns

AccidentID (PK, int)
Date (DATE, NOT NULL)
Location (varchar(100), NOT NULL)
Description (varchar(200), NOT NULL)
CarID (FK, int, NOT NULL)

Keys & Constraints
FK: CarID → Car(CarID)

5.InvolvedIn
Represents the involvement of a car in a specific accident.
Columns
CarID (FK, int, NOT NULL)
AccidentID (FK, int, NOT NULL)
Keys & Constraints
PK: (CarID, AccidentID)
FK: CarID → Car(CarID)
FK: AccidentID → Accident(AccidentID)

6.Claim
Claims related to accidents, which may be submitted to the insurance company. **Columns**
ClaimID (PK, int)
Amount (decimal(10,2), NOT NULL)
Status (varchar(20), NOT NULL)
AccidentID (FK, int, NOT NULL)
Keys & Constraints
FK: AccidentID → Accident(AccidentID)

7.RepairShop
Repair shops that handle car repairs after accidents.
Columns
ShopID (PK, int)
ShopName (varchar(100), NOT NULL)
Location (varchar(100), NOT NULL)
ContactNumber (varchar(11), NOT NULL)

8.Repairs
Represents the relationship between accidents and the repair shops that handled the repairs.
Columns
AccidentID (FK, int, NOT NULL)
ShopID (FK, int, NOT NULL)
Keys & Constraints
FK: AccidentID → Accident(AccidentID)
FK: ShopID → RepairShop(ShopID)

9.InsurancePolicy

Policies held by customers and covering cars.

Columns

PolicyID (PK, int)
StartDate (DATE, NOT NULL)
EndDate (DATE, NOT NULL)
CoverageType (varchar(20), NOT NULL)
CustomerID (int, NOT NULL)
Cid (int, NOT NULL)

Keys & Constraints

FK: CustomerID → Customer(CustomerID)
FK: Cid → Car(CarID)

Inquiries in Requirements:

A.

```
SELECT COUNT(DISTINCT o.CUSTOMERID) AS TotalOwners
FROM OWNS o
JOIN INVOLVEDIN i ON o.CARID = i.CARID
JOIN ACCIDENT a ON i.ACCIDENTID = a.ACCIDENTID
WHERE YEAR(a.DATE) = 2017;
```

B:

```
SELECT COUNT(*) AS AccidentCount
FROM CUSTOMER c
JOIN OWNS o ON c.CUSTOMERID = o.CUSTOMERID
JOIN INVOLVEDIN i ON o.CARID = i.CARID
WHERE c.NAME = 'Ahmed Mohamed';
```

C:

```
SELECT c.MODEL, COUNT(*) AS AccidentCount
FROM CAR c
JOIN INVOLVEDIN i ON c.CARID = i.CARID
JOIN ACCIDENT a ON i.ACCIDENTID = a.ACCIDENTID
WHERE YEAR(a.DATE) = 2017
GROUP BY c.MODEL
ORDER BY AccidentCount DESC
LIMIT 1;
```

D:

```
SELECT c.MODEL
FROM CAR c
LEFT JOIN INVOLVEDIN i ON c.CARID = i.CARID
LEFT JOIN ACCIDENT a ON i.ACCIDENTID = a.ACCIDENTID AND YEAR(a.DATE) = 2017
WHERE a.ACCIDENTID IS NULL
GROUP BY c.MODEL;
```

E:

```
SELECT c.*
FROM CUSTOMER c
JOIN OWNS o ON c.CUSTOMERID = o.CUSTOMERID
JOIN INVOLVEDIN i ON o.CARID = i.CARID
JOIN ACCIDENT a ON i.ACCIDENTID = a.ACCIDENTID
WHERE YEAR(a.DATE) = 2017;
```

F:

```
SELECT c.MODEL, COUNT(*) AS AccidentCount
FROM CAR c
JOIN INVOLVEDIN i ON c.CARID = i.CARID
GROUP BY c.MODEL;
```

Main Features of the Car Insurance System

Overview

The Car Insurance System is a desktop application developed using **C#** and the **.NET Framework**, utilizing **Windows Forms** for the user interface. It interacts with a **SQL Server** database (`CarInsuranceDB`) to manage customers, cars, accidents, claims, and related entities, providing a robust platform for car insurance operations.

1. Database Connectivity

- **SQL Server Integration**: Connects to a local SQL Server database (`CarInsuranceDB`) using a connection string with Integrated Security (`Server=localhost;Database=CarInsuranceDB;Integrated Security=True;`).
- **Query Execution**: Implements `ExecuteNonQuery` for INSERT, UPDATE, DELETE operations (returns the last inserted ID via `SCOPE_IDENTITY()`) and `ExecuteSelectQuery` for SELECT operations, supporting parameterized queries to prevent SQL injection.
- **Error Handling**: Catches and displays database errors via `MessageBox`, ensuring users are informed of connectivity or query failures.

2. User Authentication

- **Sign-In System**: Validates customer ID and password against the `CUSTOMER` table. Successful login sets `isLoggedIn` to true and stores the `currentCustomerId`.
- **Sign-Out Functionality**: Resets `isLoggedIn` and `currentCustomerId`, hiding all action buttons except "Sign In."
- **Input Validation**: Ensures customer ID is a positive integer and password is nonempty before querying the database.

3. Dynamic GUI Generation

- **Main Form Layout**: A Windows Forms interface with a fixed size (500x800) containing a sign-in section and dynamically visible buttons for various actions (e.g., Add Customer, Update Car, View Claim).
- **Modal Dialogs**: Each action (e.g., Add Customer, Update Car) opens a modal form with dynamically generated input fields based on the database schema (e.g., textboxes for name, email, license plate).
- **Conditional Visibility**: Action buttons are hidden until the user logs in, ensuring secure access to features.

4. CRUD Operations

- **Customer Management**:
  - **Add**: Inserts a new customer into the `CUSTOMER` table with validated name, email, phone, address, and password. Returns the new `CUSTOMERID`.
  - **Update**: Retrieves current customer data, allows partial updates (uses existing values if fields are empty), and validates email and phone.
  - **Delete**: Deletes the logged-in customer's record with a confirmation prompt, leveraging `ON DELETE CASCADE` for related records.
- **Car Management**:
  - **Add**: Inserts a car into the `CAR` table (license plate, model, year) and creates an `OWNS` relationship with the logged-in customer.
  - **Update**: Lists customer-owned cars in a ComboBox, pre-fills fields, and allows partial updates with year validation (1886–2025). ○ **Delete**: Deletes a selected car with confirmation, cascading to related tables (`OWNS`, `INVOLVEDIN`).
- **Accident Management**:
  - **Add**: Inserts an accident into the `ACCIDENT` table (date, location, description) and links it to a customer-owned car via `INVOLVEDIN`.
  - **Update**: Lists accidents for customer-owned cars, pre-fills fields, and allows partial updates with date validation (YYYY-MM-DD). ○ **Delete**: Deletes a selected accident with confirmation, cascading to related tables (`INVOLVEDIN`, `CLAIM`).
- **Claim Management**:
  - **Add**: Inserts a claim into the `CLAIM` table (accident ID, amount, status) linked to a customer's accident, with status restricted to "Pending," "Approved," or "Rejected."
  - **View**: Displays details of a selected claim (ClaimID, Amount, Status, AccidentID).

5. Input Validation

- **Email Validation**: Uses regex (`^[^@\s]+@[^@\s]+\.[^@\s]+$`) to ensure valid email format.
- **Phone Validation**: Enforces an 11-digit phone number using regex (`^\d{11}$`).
- **Date Validation**: Checks for YYYY-MM-DD format using `DateTime.TryParseExact`.
- **ID Validation**: Ensures IDs are positive integers using `int.TryParse`.
- **Field Constraints**: Enforces non-empty fields for required inputs (e.g., name, license plate) and valid ranges (e.g., car year between 1886 and 2025).
- **Status Enforcement**: Restricts claim status to predefined values ("Pending," "Approved," "Rejected").

6. Data Retrieval and Display

- **ComboBox Population**: Dynamically populates ComboBoxes with customerowned cars (for car and accident operations) and accidents (for claims), displaying human-readable data (e.g., license plate and date).
- **Accident Viewing**: Retrieves and displays all fields of a selected accident in a `MessageBox`.
- **Claim Viewing**: Shows formatted claim details (ClaimID, Amount, Status, AccidentID) for a selected claim.
- **Report Generation**: Generates a report of accidents for a specified year, displayed in a read-only multiline textbox with a scrollable view.

7. Report Generation

- **Annual Accident Report**: Queries the `ACCIDENT` table for records in a userspecified year, displaying all fields in a formatted, scrollable textbox.
- **Year Validation**: Ensures the input year is between 0 and 2025.
- **Empty Result Handling**: Displays a message if no accidents are found for the specified year.

8. Error Handling and User Feedback

- **Database Errors**: Catches exceptions in `ExecuteNonQuery` and `ExecuteSelectQuery`, displaying error messages via `MessageBox`.
- **Validation Feedback**: Provides clear error messages for invalid inputs (e.g., "Phone number must be 11 digits!").
- **Confirmation Prompts**: Uses `MessageBox` with `YesNo` options for destructive actions (e.g., Delete Customer, Delete Car).

- **Success Messages**: Confirms successful operations (e.g., "Customer added successfully! Customer ID: {ID}").

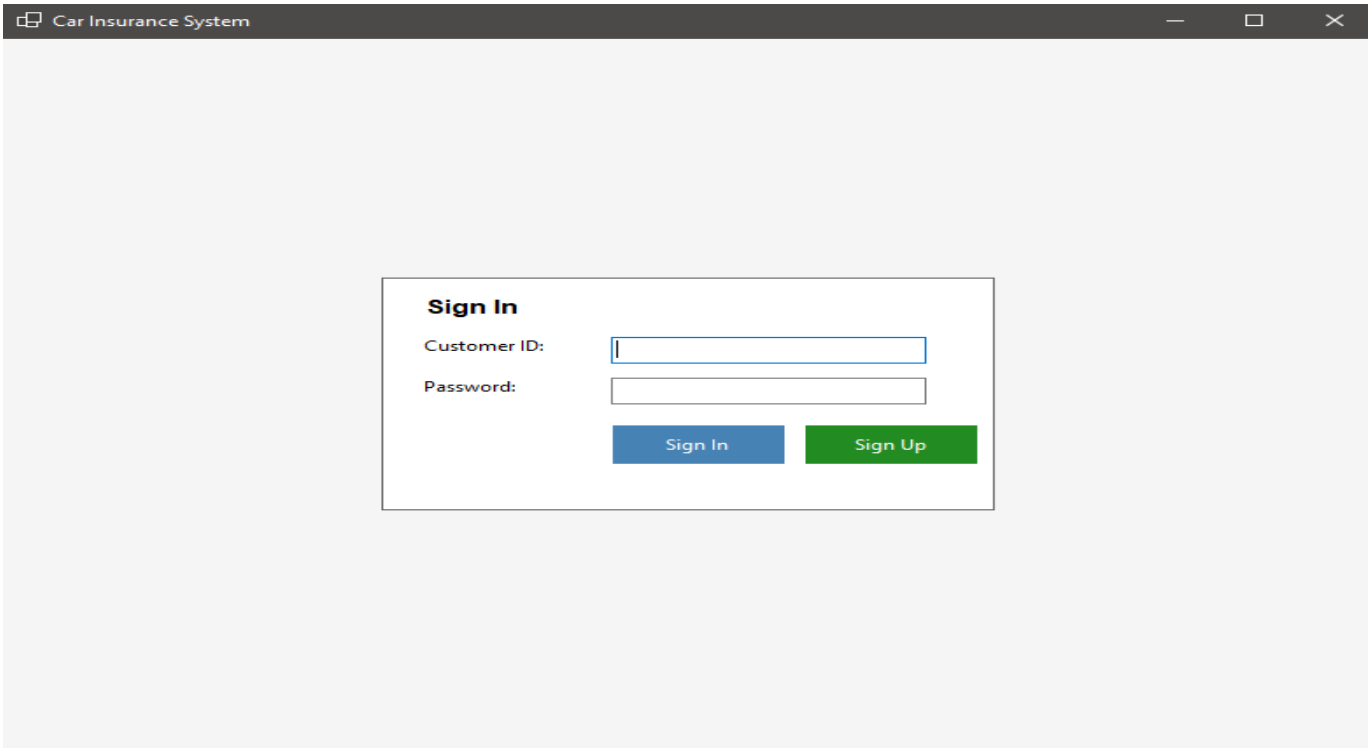
9. Security Considerations

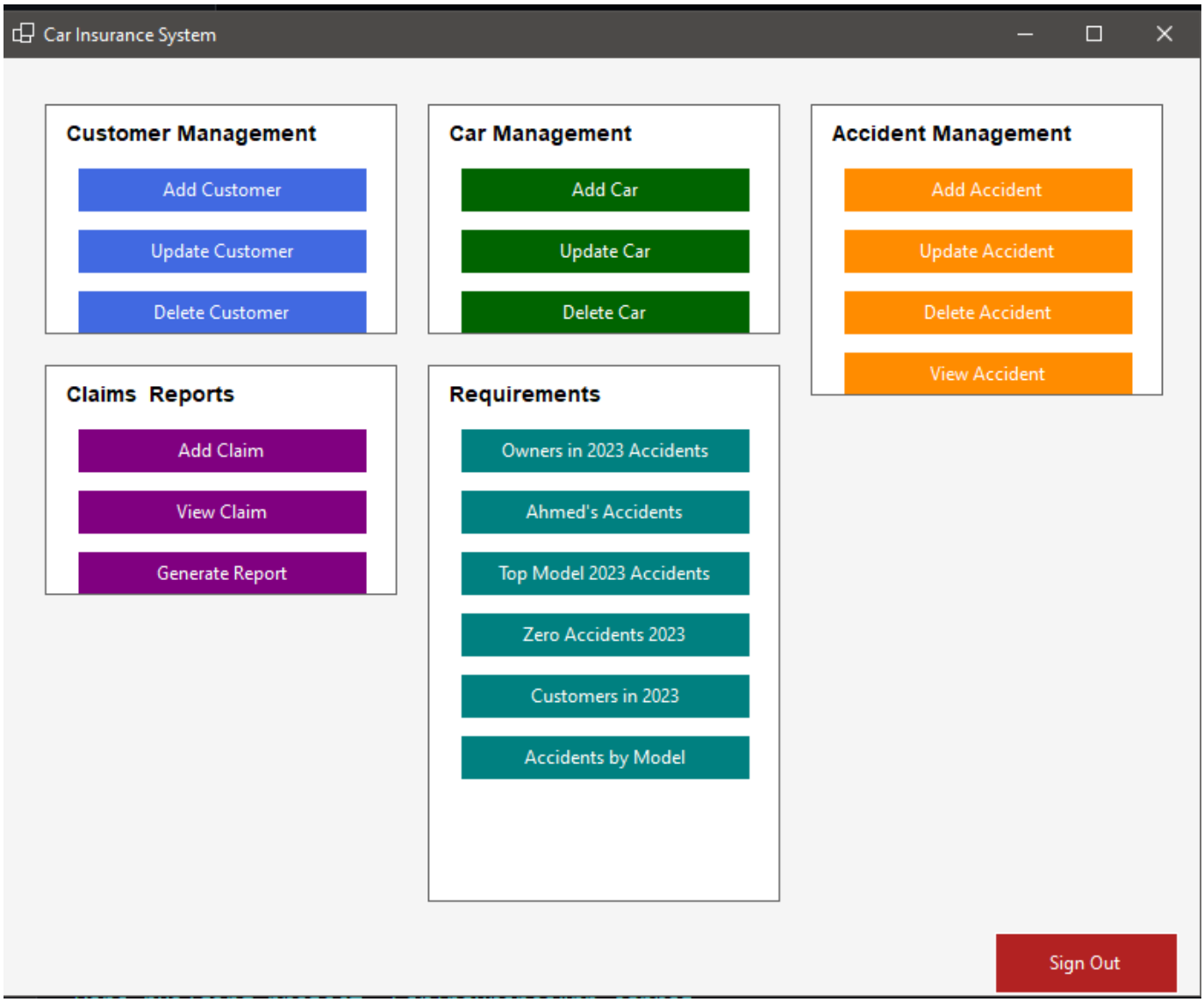
- **Parameterized Queries**: Uses parameterized SQL queries to prevent SQL injection in all database operations.
- **Password Masking**: Displays passwords as asterisks (*) in textboxes.
- **Access Control**: Restricts all CRUD operations to logged-in users by checking `isLoggedIn`.

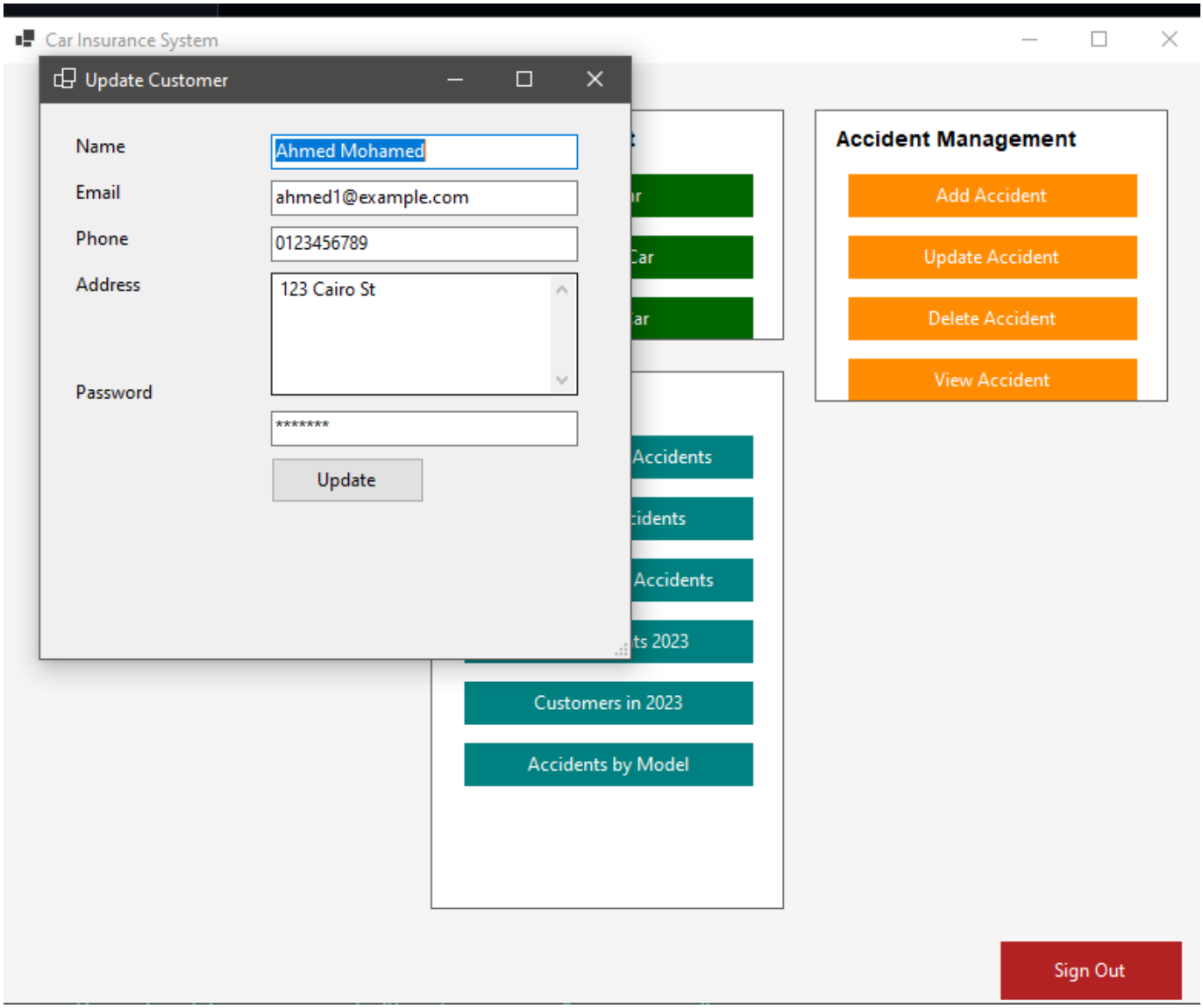
10. Application Lifecycle

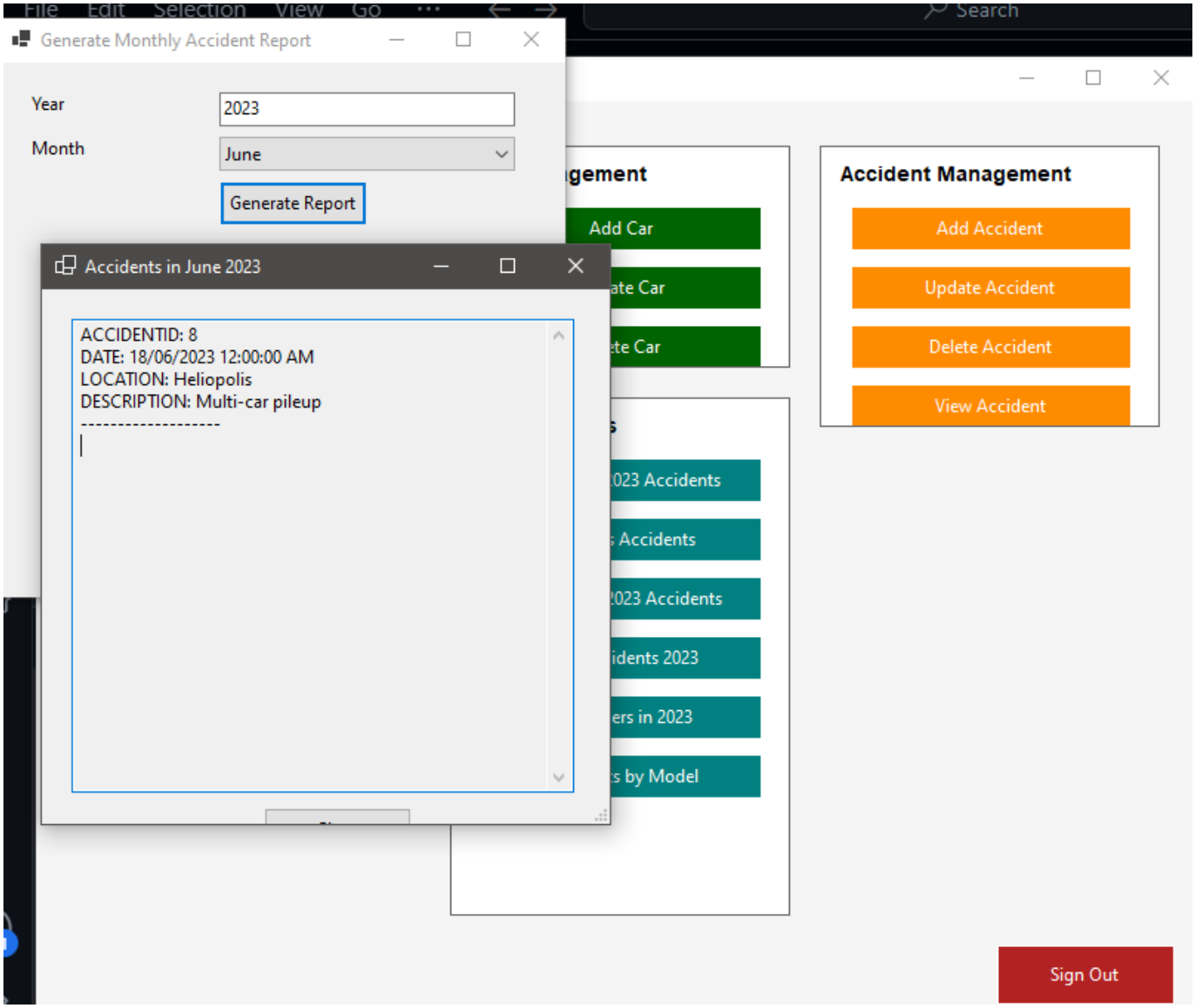
- **Startup**: Initializes the main form, sets up the GUI, and prepares the sign-in interface.
- **Form Management**: Uses modal dialogs for all input forms, ensuring a clean user experience.
- **Shutdown**: Properly disposes of components via the `Dispose` method, inherited from `Form`.

Some GUI Photos

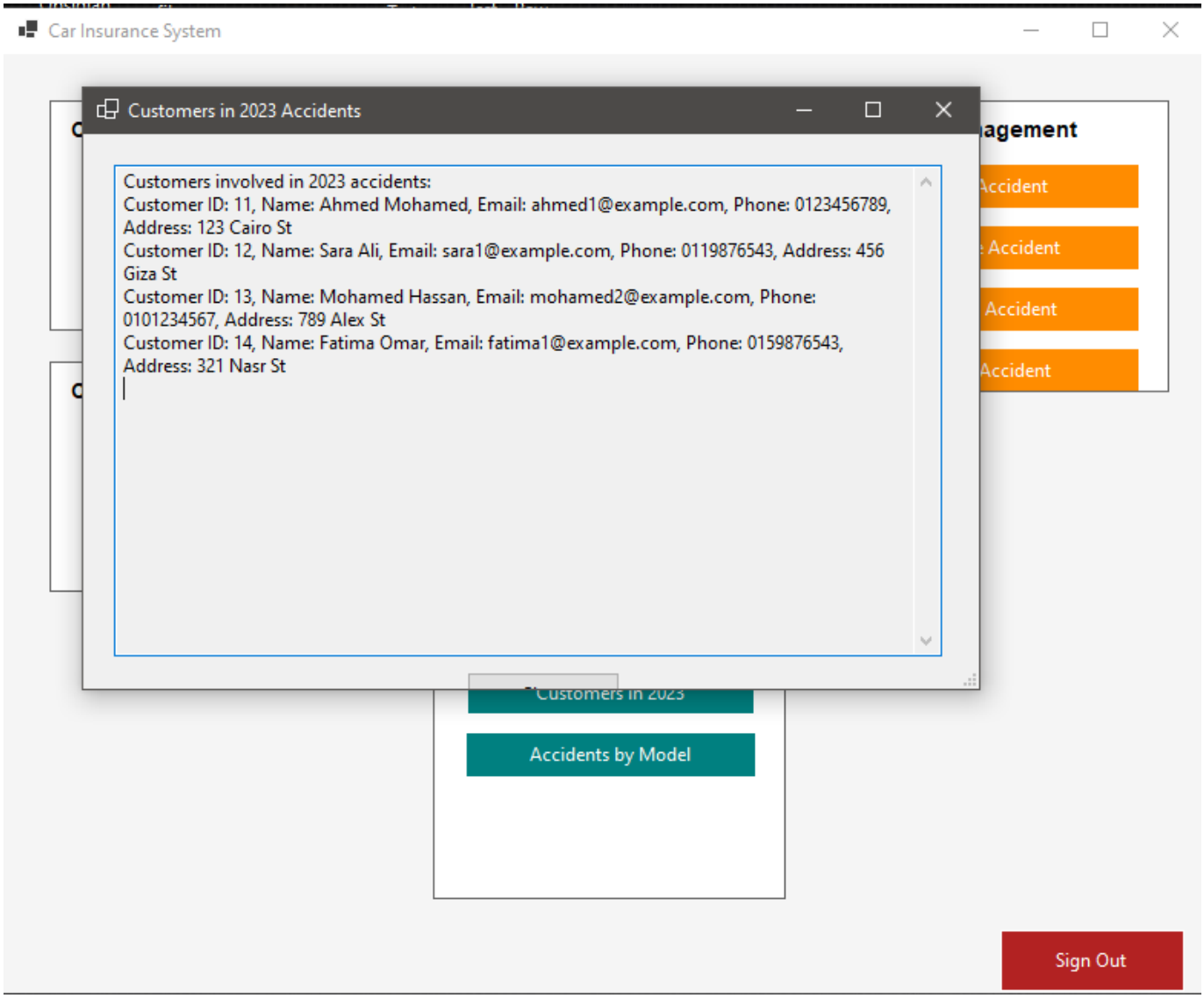



Car Insurance System
Customer Management
Add Customer
Update Customer
Delete Customer
Car Management
Add Car
Update Car
Delete Car
Accident Management
Add Accident
Update Accident
Delete Accident
View Accident
Claims Reports
Add Claim
View Claim
Generate Report
Requirements
Owners in 2023 Accidents
Ahmed's Accidents
Top Model 2023 Accidents
Zero Accidents 2023
Customers in 2023
Accidents by Model
Sign Out


Car Insurance System
Update Customer
Name
Ahmed Mohamed
Email
ahmed1@example.com
Phone
0123456789
Address
123 Cairo St
Password
****
Update
Accident Management
Add Accident
Update Accident
Delete Accident
View Accident
Accidents
Accidents
Customers in 2023
Accidents by Model
Sign Out


Generate Monthly Accident Report
Year
2023
Month
June
Generate Report
Accidents in June 2023
ACCIDENTID: 8
DATE: 18/06/2023 12:00:00 AM
LOCATION: Heliopolis
DESCRIPTION: Multi-car pileup

Add Car
Accident Management
Add Accident
Update Accident
Delete Accident
View Accident
Sign Out


Car Insurance System
Customers in 2023 Accidents
Customers involved in 2023 accidents:
Customer ID: 11, Name: Ahmed Mohamed, Email: ahmed1@example.com, Phone: 0123456789, Address: 123 Cairo St
Customer ID: 12, Name: Sara Ali, Email: sara1@example.com, Phone: 0119876543, Address: 456 Giza St
Customer ID: 13, Name: Mohamed Hassan, Email: mohamed2@example.com, Phone: 0101234567, Address: 789 Alex St
Customer ID: 14, Name: Fatima Omar, Email: fatima1@example.com, Phone: 0159876543, Address: 321 Nasr St
agement
Accident
Accident
Accident
Accident
Customers in 2023
Accidents by Model
Sign Out