Rent car agreement generator - Windows Forms application

## Contents

1. Program description
2. Application UI design
3. Database model
4. Classes description
5. Summary

## Program description

The Agreement Generator is a C# Windows Forms application designed to manage car rental agreements and generate statistical reports. It utilizes a MySQL database to store and manage car data, rental agreements.

Main Features:

1. Edit Database:
   1. Add and remove car brands.
   2. Add and remove car models.
   3. Add and remove individual cars.
2. Generate Agreement:
   1. Create rental agreements using a template.
   2. Find and replace functionality to populate agreements with user input and database data.
3. Statistics
   1. Generate charts on the most frequently rented cars.
   2. Generate chart on the most popular rental periods.
   3. Generate charts on the average rent cost and period.

Techniques Used:

- \*\*File System\*\*: Manage templates and save generated agreements.

- \*\*Regex\*\*: Find and replace functionality in agreement templates.

- \*\*Threads\*\*: Perform background operations such as database queries and report generation.

- \*\*Simple Concurrency\*\*: Ensure thread-safe operations when accessing the database.

- \*\*Inheritance and Polymorphism\*\*: Use object-oriented principles to create a flexible and maintainable codebase.

## Application UI design

Login form

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

**Login Form:**

* **Collects user credentials and database connection details.**
* **Initializes the application and establishes a connection to the database.**

Main Menu Form

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, Prostokąt

Opis wygenerowany automatycznie

**The main menu provides three options:**

* **Edit Database**
* **Generate Agreement**
* **Statistics**

Edit Database Form

Users can manage car data through a user-friendly interface. This section includes the following functionalities:

1. **Add/Remove Car Brands:** 
   * **Add new car brands to the database.**
   * **Remove existing car brands.**

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

1. **Add/Remove Car Models:**
   * **Add new car models associated with a brand.**
   * **Remove existing car models.**

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

1. **Add/Remove Cars:** 
   * **Add individual cars with details such as model, brand, and registration**
   * **Remove cars that are no longer in the fleet.**

**Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie**

Generate Agreement Form

This section allows users to create rental agreements after submitting all required information.

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, numer

Opis wygenerowany automatycznie

Statistics Form

Users can generate and view statistical charts about the rental operations:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, diagram

Opis wygenerowany automatycznie

**Available charts:**

* **Most frequently rented cars.**
* **Most popular rental periods.**
* **Average rent cost and period.**

# Database model

Obraz zawierający tekst, zrzut ekranu, diagram, Równolegle

Opis wygenerowany automatycznie

**Database Description**

The database for the Agreement Generator Application consists of five tables: carbrand, carmodel, car, reservation, and car.

Table Descriptions:

**carbrand**

Fields:

* brandId (INT): Primary key, unique identifier for each car brand.
* brandName (VARCHAR(20)): Name of the car brand.

Relationships:

* One-to-many relationship with carmodel.

**carmodel**

Fields:

* modelId (INT): Primary key, unique identifier for each car model.
* modelName (VARCHAR(30)): Name of the car model.
* brandId (INT): Foreign key, links to brandId in carbrand.

Relationships:

* Many-to-one relationship with carbrand.
* One-to-many relationship with car.

**car**

Fields:

* carId (INT): Primary key, unique identifier for each car.
* brandId (INT): Foreign key, links to brandId in carbrand.
* modelId (INT): Foreign key, links to modelId in carmodel.
* carRegistration (VARCHAR(10)): Registration number of the car.

Relationships:

* Many-to-one relationship with carmodel.
* Many-to-one relationship with carbrand.

**reservation**

Fields:

* reservationId (INT): Primary key, unique identifier for each reservation.
* pickUpDateTime (DATETIME): Date and time when the car is picked up.
* returnDateTime (DATETIME): Date and time when the car is returned.
* reservationCost (FLOAT): Total cost of the reservation.
* carId (INT): Foreign key, links to carId in car.

Relationships:

* Many-to-one relationship with car.

This database schema is designed to effectively manage the data related to car brands, models, individual cars, and their reservations. The relationships ensure data integrity and enable efficient querying and reporting, which are essential for generating car rental agreements and statistical reports in the Agreement Generator Application.

## Classes description

Obraz zawierający tekst, zrzut ekranu, Prostokąt, diagram

Opis wygenerowany automatycznie

Based on the class diagram provided in the image, here are the details of all the classes, including their attributes and methods:

### Class: Info

- \*\*Methods:\*\*

- `EvaluateAttributes()`: bool

- `ShowErrorBox(errorText: string)`

### Class: Client

- \*\*Attributes:\*\*

- `name`: string

- `surname`: string

- `phoneNumber`: string

- `email`: string

- \*\*Methods:\*\*

- `EvaluateAttributes()`: bool

### Class: Location

- \*\*Attributes:\*\*

- `city`: string

- `street`: string

- `postalCode`: string

- \*\*Methods:\*\*

- `EvaluateAttributes()`: bool

### Class: Reservation

- \*\*Attributes:\*\*

- `reservationNumber`: int

- `pickUpDate`: DateTime

- `returnDate`: DateTime

- `price`: float

- \*\*Methods:\*\*

- `GetReservationPeriodInDays()`: int

- `EvaluateAttributes()`: bool

### Class: Car

- \*\*Attributes:\*\*

- `carId`: int

- `carBrand`: string

- `carModel`: string

- `carRegistration`: string

### Class: CarModel

- \*\*Attributes:\*\*

- `modelId`: int

- `modelName`: string

- `brandName`: string

### Class: CarBrand

- \*\*Attributes:\*\*

- `brandId`: int

- `brandName`: string

### Class: Accessories

- \*\*Attributes:\*\*

- `abroad`: bool

- `insurance`: bool

- `kilometerLimit`: bool

- `carSeat`: bool

- `navi`: bool

### Class: User

- \*\*Attributes:\*\*

- `server`: string

- `username`: string

- `password`: string

- `folderPath`: string

### Class: RentPeriods

- \*\*Attributes:\*\*

- `lengthOneToThree`: int

- `lengthThreeToFive`: int

- `lengthFiveToSeven`: int

- `lengthSevenToFourteen`: int

- `lengthFourteenPlus`: int

- \*\*Methods:\*\*

- `SetRentPeriods(timeSpans: List<TimeSpan>)`

### Class: StatisticalData<T>

- \*\*Attributes:\*\*

- `startOfPeriodDate`: DateTime

- `endOfPeriodDate`: DateTime

- `dataBase`: DataBase

- \*\*Methods:\*\*

- `EvaluateDateTimeValues()`: bool

- `GetData()`: T

- `GetDataFromDataBase(reader: MySqlDataReader)`: T

### Class: CarRentFrequency

- \*\*Attributes:\*\*

- `rentedCars`: List<Car>

- \*\*Methods:\*\*

- `GetData()`: List<Car>

- `GetDataFromDataBase(reader: MySqlDataReader)`: List<Car>

### Class: AverageValues

- \*\*Attributes:\*\*

- `data`: List<float>

- \*\*Methods:\*\*

- `GetData()`: List<float>

- `GetDataFromDataBase(reader: MySqlDataReader)`: List<float>

- `GetAverageCost(costDataList: List<float>)`: float

- `GetAverageRentPeriod(periods: List<TimeSpan>)`: float

### Class: RentPeriodFrequency

- \*\*Attributes:\*\*

- `rentPeriods`: List<TimeSpan>

- \*\*Methods:\*\*

- `GetData()`: List<TimeSpan>

- `GetDataFromDataBase(reader: MySqlDataReader)`: List<TimeSpan>

- `GetFrequencies(timeSpans: List<TimeSpan>)`: RentPeriods

# Summary

While working on this project I extended my knowledge about C# object oriented programming. I also gained experience with working with databases like MySQL and writing SQL queries and overall CRUD operations. In addition I have learned to work with windows forms editor to design ui for Windows desktop applications. Finally I have mastered topics like threads, file system library or regex creation.