# Release Information

|  |  |
| --- | --- |
| *Release* | *1* |
| *Leader* | *Michael Serino* |
| *SVN Revision Number* | *63221* |

The Car Rental System now talks to an access database, and can add customer, vehicles, and rentals to the database, each with relationships to one another via primary keys. An admin or user account can be created as well, and all data is saved in the database.

# Progress Summary

|  |  |  |  |
| --- | --- | --- | --- |
| **Stories and spike solutions planned for this release** | | | |
| **Story, Spike** | **Description** | **% Impl.** | **Notes, plan for completion** |
| *Spike 1* | *A database to assist us in designing the system.* | *100%* | *Database was used to implement many of the other stories.* |
| *S.UI.01* | *Allow admins to add cars to the database.* | *100%* |  |
| *S.UI.03* | *Allow the user to specify a pick-up location for the rented car.* | *100%* |  |
| *S.UI.09* | *Admin can verify that the car was returned.* | *100%* |  |
| *S.UI.10* | *Allow the user to specify a drop off location for them to return the car.* | *100%* |  |
| *S.UI.11* | *Prospective customer can create an account in the system.* | *100%* |  |
| *S.UI.12* | *Customer can choose a car from the list of cars.* | *100%* |  |
| *S.SO.06* | *Customer can view the price of a car they wish to rent.* | *99%* | *Works with default car objects, will have to retest when cars become more complicated database entries.* |
| *S.IO.02* | *Allow the user to pay for their car.* | *90%* | *Adding a true payment system seems outside the scope of the project.* |

|  |  |  |
| --- | --- | --- |
| **Accomplishments by engineer** | | |
| **Team Member** | **Total Time (hrs)** | **Stories, Spikes implemented** |
| *Noah* | *35* | *Spike 1: Figure out how to access Database*  *S.1: Add Cars*  *S.4: Specify Pickup* |
| *Mikey* | *22* | *Spike 1: Figure out how to access Database*  *S.1: Add Cars*  *S.15 Return car*  *S.4 Specify Pickup* |
| *Sarah* | *21* | *Spike 1: Figure out how to access Database*  *S.22: Choose a car*  *S.1: Add Cars* |
| *Zack* | *21* | *S.6: Pay for the rental*  *S.18: Specify drop off location*  *S.21: Create a user account*  *S.23: Determine amount owed*  *S.22: Choose a car* |
| *Xinyi* | *21* | *S.4: Specify Pickup*  *S.6: Pay for the rental*  *S.15: Return car*  *S.22: Choose a car* |
| TEAM TOTAL | ***120*** |  |

|  |  |
| --- | --- |
| **Document updates** | |
| **Document** | **Changes** |
| *Status Doc* | *Updated the status doc for each week to represent what our plans were per week.* |
| *Story Book* | *Added the subtasks as well as updated the story numbers to match the category. Gave the implementation percentages for all stories.* |
| *Class Diagram* | *The DBObject was added to the class diagram and some changes were made to the associations of some classes.* |

# Plan for Next Release

Browsing is going to be made more specific and there will be more information for the users to view for cars and rentals.

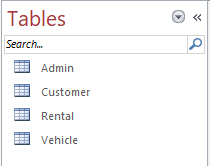
|  |  |  |  |
| --- | --- | --- | --- |
| **Stories and spike solutions planned for upcoming release** | | | |
| **Story, Spike** | **Description** | **Est. Hours** | **Notes, plan for completion** |
| *S.IO.01* | *Admin is able to modify an existing car in the database* | *6* | *There needs to be an edit page. The changes need to be saved in the database.* |
| *S.UI.04* | *User is able to choose car by components* | *9* | *Create a filter method in DBObject.*  *Create a filter method in the vehicle.*  *Add functionality to the GUI.* |
| *S.SO.02* | *User is able to browse for cars* | *12* | *The system is able to list all the available cars* |
| *S.UI.05* | *Admin is able to delete a car from the database* | *9* | *Update the database for the vehicle* |
| *S.UI.08* | *Making other admin accounts* | *3* | *Need to make individual employ ID.* |
| *S.SO.04* | *Track number of rental days* | *3* | *System will have a counter to determine how may days left.* |
| *S.IO.05* | *Search car categories* | *15* | *System will have filter for the kind, year, model, make, for the specific vehicle.* |
| *Spike2* | *This spike is going to assist us in filtering the database for our system* | *3* |  |
|  |  | 60.***0*** | Total |

|  |  |  |  |
| --- | --- | --- | --- |
| **Assignments** | | | |
| **Team Member** | **Stories, Spikes implemented** | | **Tot. Hrs.** |
| *Zackary Neefe* | S.IO.01  S.UI.04  S.UI.08  S.SO.04  S.IO.05 | *Modify Cars (w/ Sarah)*  Choose Car components(w/ Noah)  Create Admin(w/ Sarah)  Track Number of Rental Days(w/ Sarah)  Search Categories | *12.5* |
| *Noah Moss* | *Spike2*  S.UI.04 | *Spike 2(w/ Xinyi)*  *Choose Car Components(w/ Zackary)* | *11* |
| *Michael Serino* | S.SO.02  S.UI.05  S.IO.05 | *Browse Cars(w/ Sarah)*  *Delete Cars(w/ Xinyi)*  Search Categories | *12.5* |
| *Xinyi Lyu* | *Spike2*  S.UI.05 | *Spike 2(w/ Noah)*  *Delete Cars(w/ Michael)* | *11.5* |
| *Sarah Higgens* | S.IO.01  S.SO.02  S.UI.08  S.SO.04  S.IO.05 | *Modify Cars (w/ Zackary)*  *Browse Cars(w/ Michael)*  *Create Admin(w/ Zackary)*  Track Number of Rental Days(w/ Sarah)  Search Categories | *12.5* |
|  |  | **Total** | **060** |

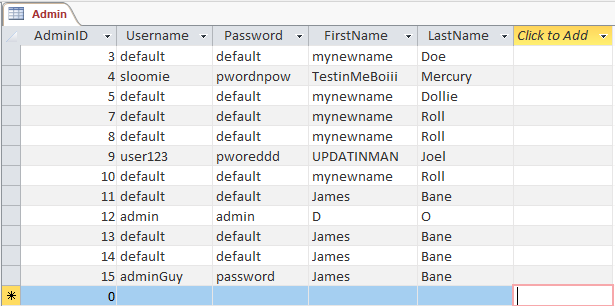
# Issues

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Issue Number** | **Discovery**  **Date** | **Resolution**  **Date (Est.–Act.)** | **Responsible Person** | **Description (Prob / Resolution)** |
| 1 | 3/4 | 3/4 | Entire Team | Some of our implementation was over-done. We slightly implemented some stories for release 2. Our solution is to start checking the story book more often to make certain we will not repeat the same issue. |
| 2 | 3/4 | 3/12 | Michael | The school computer were not running any tests for our software, but the tests worked fine on our personal computers. The solution is to have the either the instructor give feedback on what might be causing the error, or get ITS to fix the issue if possible. |
| 3 | 2/22 | 2/28 | Noah | Noah started major work on the DBObject and needed to refractor a good part of the code to work with the new class. |

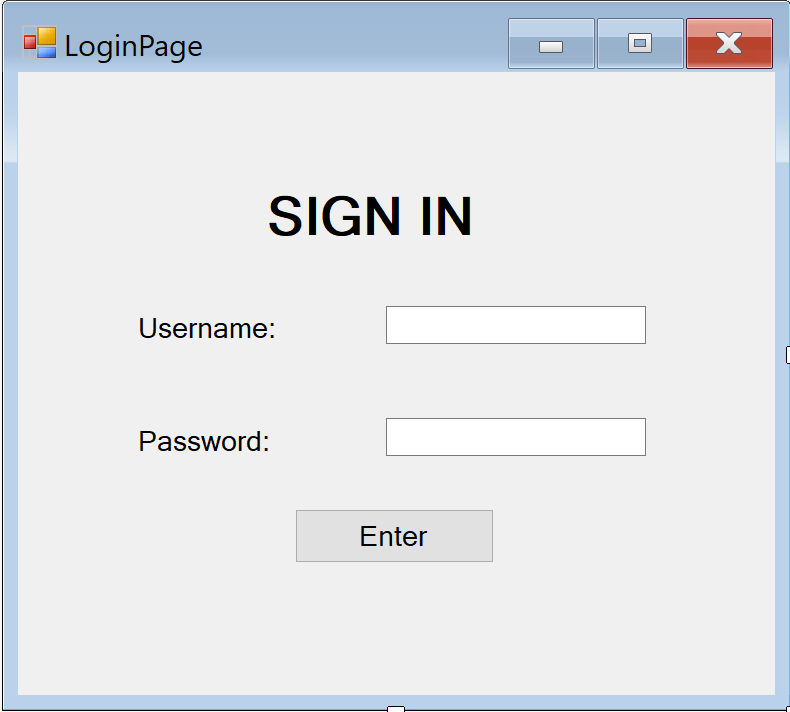
# Screen Shots



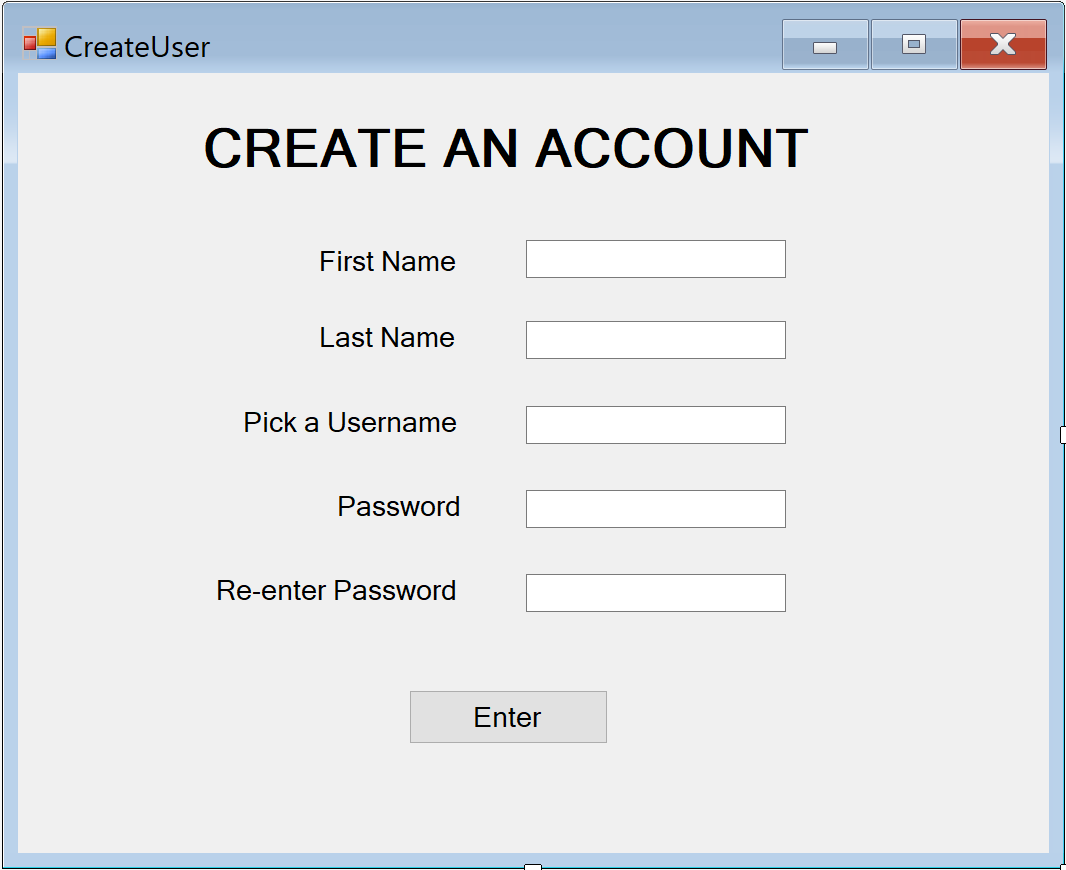
This is a screenshot of the 4 tables that we have nested within our MS-Access database; namely, Admin, Customer, Rental, and Vehicle. Each table has a list of attributes and unique record identifiers. With these tables, we are able to store, retrieve, remove, and update records in the MS-Access.



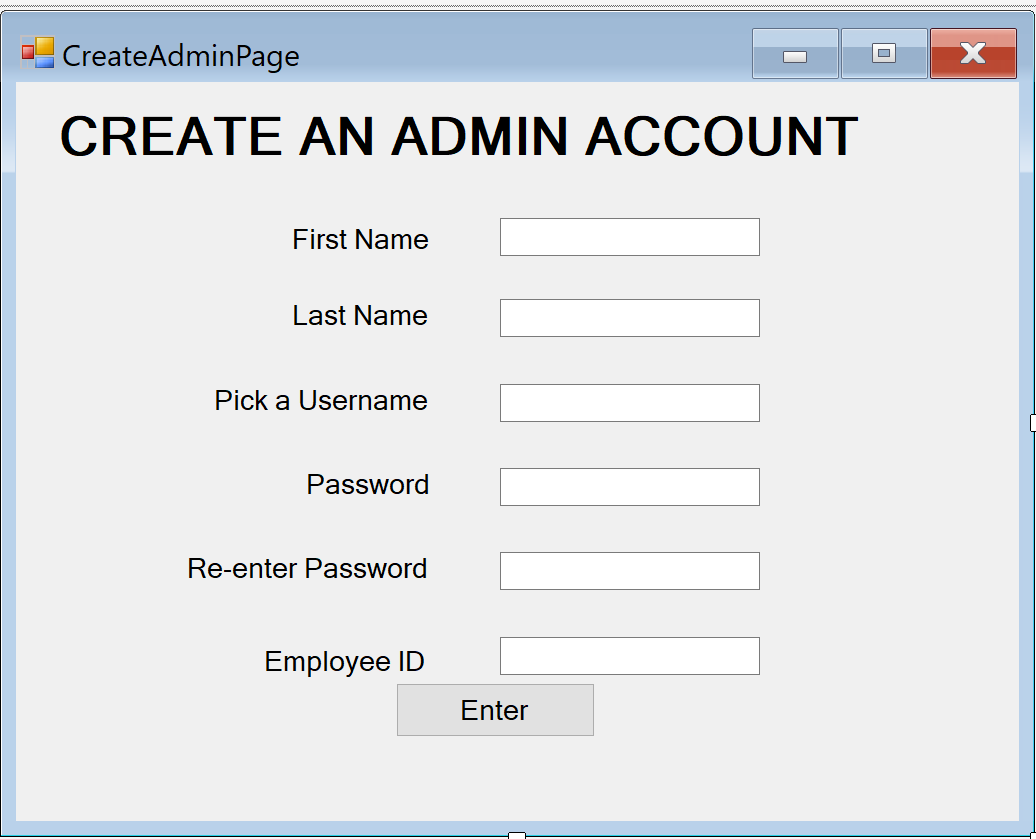
This is a screenshot of the Admin table. The admin table has a list of attributes (AdminID, Username, Password, FirstName, LastName) and a primary key to uniquely identify an admin record (AdminID). Inside of this table is currently dummy data that was used to test the functionality of the database. Each attribute name matches a property name in the associated class using a custom C# attribute called "SchemaField". When we begin using actual data, we will terminate these records and start on a clean slate.



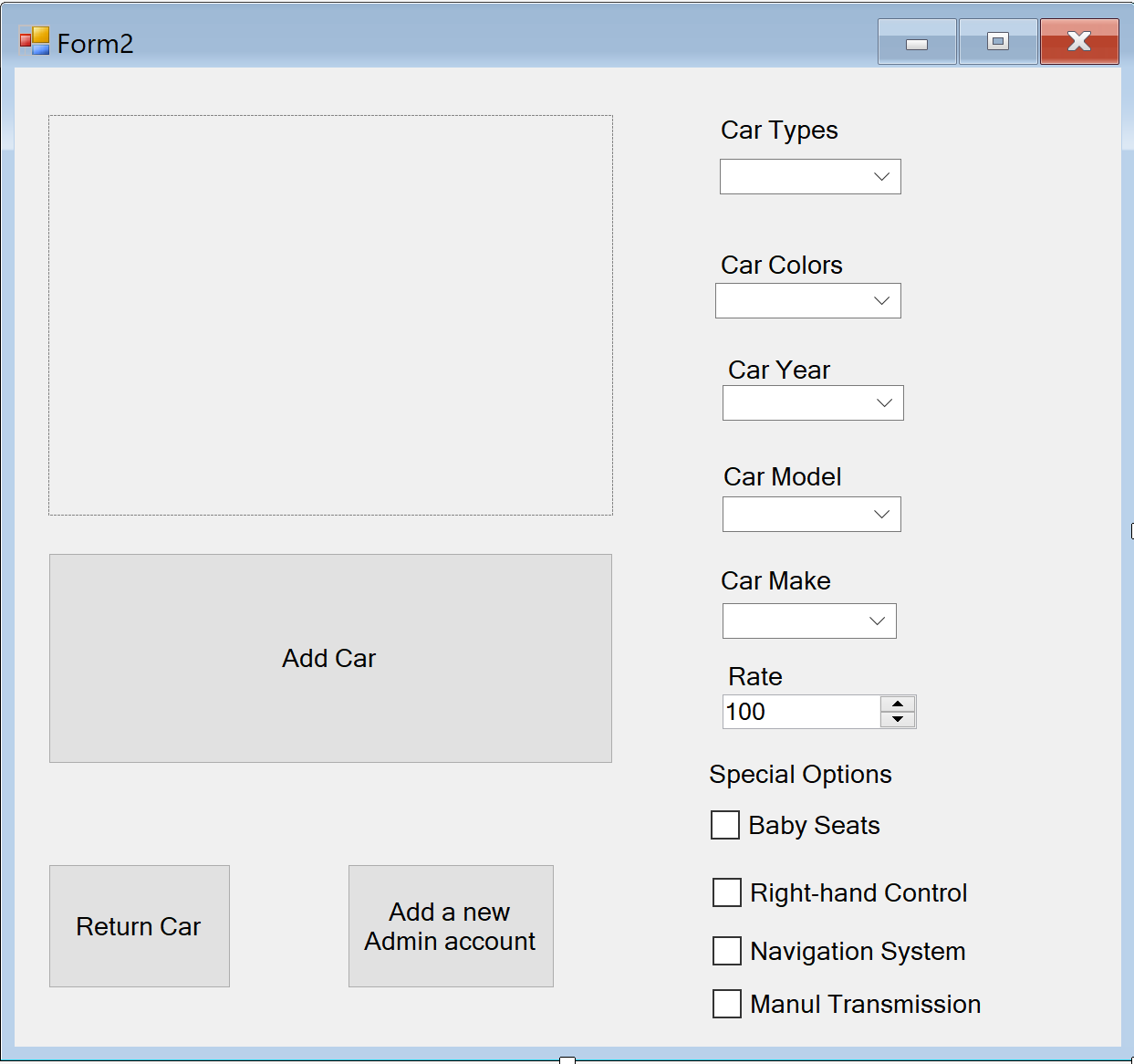
After clicking the "log in" button on the main form, this prompt appears. The entered username is searched for in the database and if the password matches that entry's password the user is logged in.



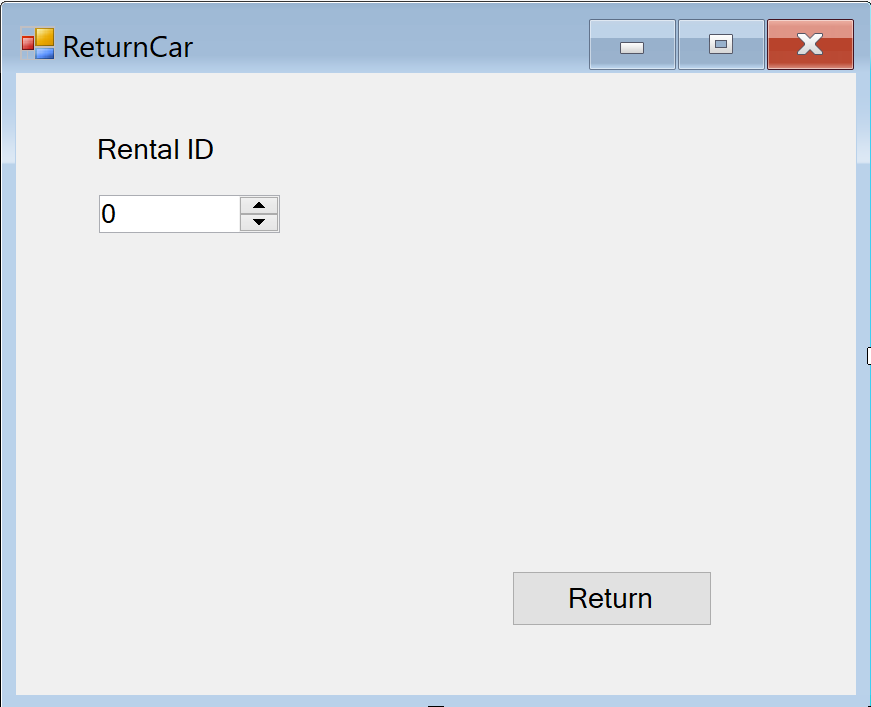
This page appears when the user clicks "create account" on the main page. If all the fields are filled and the password check matches, the account is added to the database.



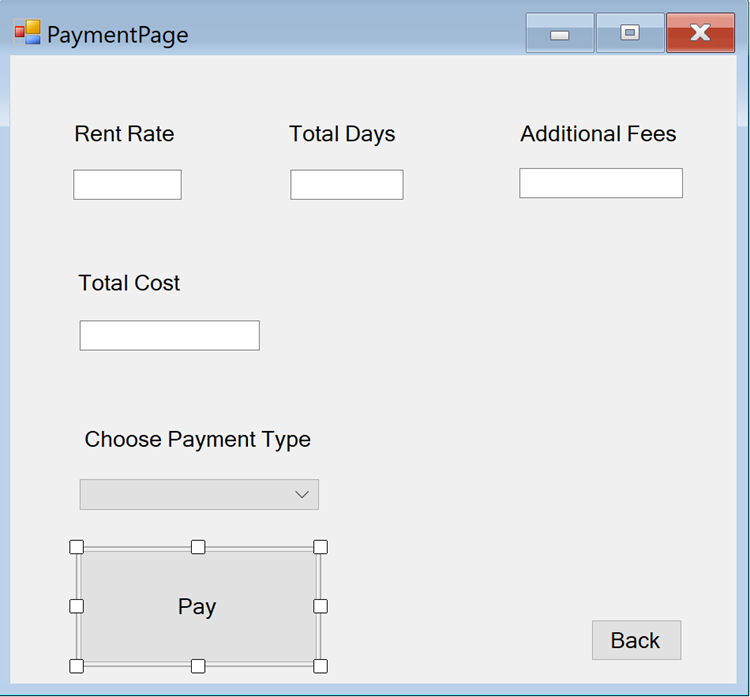
This is the same as creating a normal user, but only accessible by admins that are already logged in.



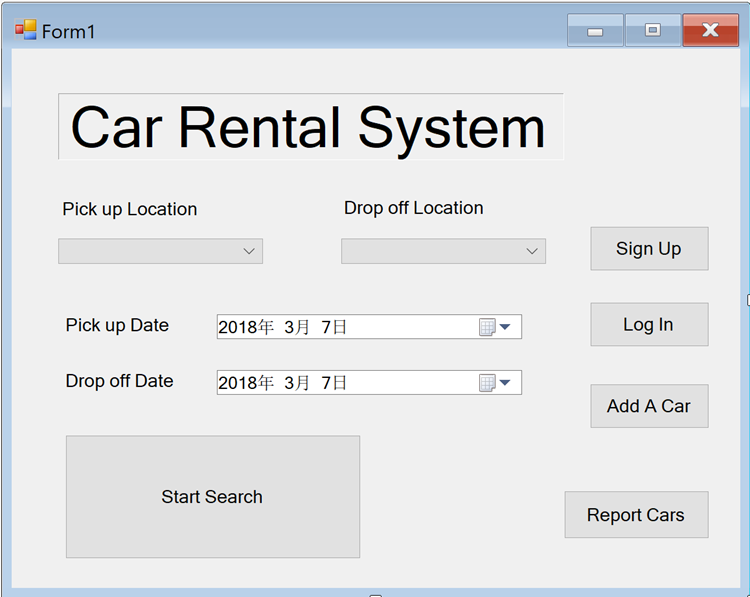
This is the admin-accessible create car page. Rate field was added as well as the options for the admin to create a new admin account and to check that a car was returned.



The admin enters the ID of the vehicle that has been returned and the database marks the vehicle as available again.



This page shows after a customer elects to rent a car they have viewed. The information from the various forms and database objects is displayed. The user can choose to "pay" in either cash or card.



Main form of the system. The user is able to sign up for the system. The user is able to log in the system. The admin is able to add a car. The system is able to display date by year calendar.