**Final FullStack Project**

1. **Installation**
   1. **DB creation**
      1. Files for creating the DB located in the "DB Creation" folder : <https://github.com/oriTid/CS_Search_project/FINAL/tree/master/DB%20Creat>
      2. **You have 2 options for creating the:** 
         1. Restore CS\_Search\_DB.bak
         2. Execute Script CS\_Search\_DB.sql on SQL Server
      3. **Updating the Data-Connection in the program:**After DB is created, you will need to update the Data-Connection, in order for the program to work properly.
         1. **The connection is set in the Web.Config file located in the 04\_UIL:**  
            <connectionStrings>  
            <add name="CarRentEntitiesModel"connectionString="metadata=res://\*/CarRentEntitiesModel.csdl|res://\*/CarRentEntitiesModel.ssdl|res://\*/CarRentEntitiesModel.msl;provider=System.Data.SqlClient;provider connection string=&quot;**data source=devps2010;initial catalog=CarRentComp**;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />  
            </connectionStrings>
         2. Update 'data source' with SQL server name  
            **data source=YOUR SQL SERVER NAME**
         3. Update 'catalog' with the DB name (**if you changed it**)**initial catalog=CarRentComp (or you DB name if you change it)**
         4. Dont forget to update it before running the program.
   2. **Angular environment setup**After the server is up and running you can see the port used for communicating.  
      1. Navigate to "environment.ts" , located in src folder.
      2. Update the endpoint parameter with your local server port  
         etc – 'http:/localhost:XXX**'**
   3. **Don’t forget NPM Install**
2. **DB Structure**  
   Down here is the structure of the App DB tables and fields  
   1. **Users** Table formation types and details:

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Details |
| UserId | Int | No Nulls, PK, Identity=yes, |
| FirstName | nvarchar(50) | No Nulls |
| LastName | nvarchar(50) | No Nulls |
| ID | nvarchar(9) | No Nulls |
| UserName | nvarchar(50) | No Nulls, Uniqe |
| Birthdate | Date | Allow Nulls |
| isMale | bit | No Nulls, default 0 |
| Email | nvarchar(50) | No Nulls |
| Password | nvarchar(50) | No Nulls |
| UserPic | nvarchar(150) | Allow Nulls |
| UserPermission | Int | No Nulls, default 0 |
| IsDeleted | bit | No Nulls, default 0 |

* 1. **Branches** Table formation types and details:

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Details |
| BranchID | Int | No Nulls, PK, Identity=yes, |
| BranchName | nvarchar(50) | No Nulls |
| BranchTel | nvarchar(50) | No Nulls |
| BranchAddress | Float | No Nulls |
| BranchLatitude | Float | No Nulls |
| BarnchLongitude | Float | No Nulls |
| IsDeleted | bit | No Nulls, default 0 |

* 1. **CarTypes** Table formation types and details:

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Details |
| CarTypeID | Int | No Nulls, PK, Identity=yes, |
| Manufacturer | nvarchar(50) | No Nulls |
| Model | nvarchar(50) | No Nulls |
| CarYear | Int | No Nulls |
| DailyRate | Money | No Nulls |
| LateDailyRate | Money | No Nulls |
| IsAutomatic | bit | No Nulls, default 1 |
| IsDeleted | bit | No Nulls, default 0 |

* 1. **Cars** Table formation types and details:

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Details |
| CarID | Int | No Nulls, Identity=yes, |
| CarTypeID | Int | No Nulls, FK (CarTypes.CarTypeID) |
| LicenseNumber | nvarchar(15) | No Nulls, PK, unique |
| CurrentKilometers | Int | No Nulls |
| IsOperative | bit | No Nulls, default 1 |
| BranchLocation | Int | No Nulls, FK (Branches.BranchID) |
| CarPic | nvarchar(150) | Allow Nulls |
| IsDeleted | bit | No Nulls, default 0 |
| IsDeleted | bit | No Nulls, default 0 |

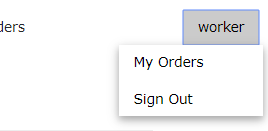
* 1. **Orders** Table formation types and details:

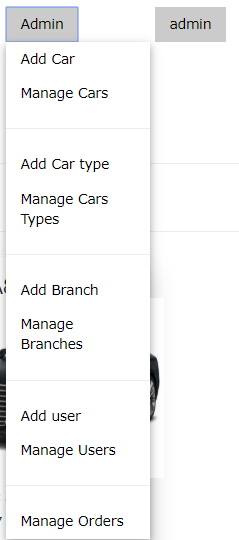
|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Details |
| OrderID | Int | No Nulls, PK, Identity=yes, |
| OrderDate | Date | No Nulls |
| LicenseNumber | nvarchar(15) | No Nulls, FK (Cars.LicenseNumber) |
| UserID | Int | No Nulls, FK (Users.UserID) |
| OrderStart | date | No Nulls |
| PlannedEnd | date | No Nulls |
| ActualEnd | date | Nulls |
| OrigianlCost | money | No Nulls |
| ActualCost | money | Nulls |
| IsActiveOrder | bit | No Nulls, default 1 |

1. **Web-Site Logics**HotWheels is a car rental company.   
   The web site is serving 4 types of users, each with its own operations:  
   1. **Anonymous users** – users that are not registered to the site.  
      Operations available for this type are:
      1. Browsing Home page, Cars page, Branches Page
      2. Signup for the system
   2. **Registered users** – users that are registered to the site.  
      Operations available for this type are:
      1. Browsing Home page, Cars page, Branches Page
      2. Booking a car
      3. See their own orders
      4. Signin to the system
   3. **Worker** – users that work in HotWheels Car rental company.  
      Operations available for this type are:
      1. Browsing Home page, Cars page, Branches Page, Orders Page
      2. Signin to the system
      3. Closing an active Order (current date is greater than the order start date)
      4. Cancel and an active order (current date is less than the order start date)
   4. **Admin** – users that admin HotWheels Car rental company web site.  
      Operations available for this type are:
      1. Browsing Home page, Cars page, Branches Page, Orders Page, Admin
      2. Signin to the system
      3. Closing an active Order (current date is greater than the order start date)
      4. Cancel and an active order (current date is less than the order start date)
      5. **\*Deleting** a non active Order
      6. Add/Edit/Delete Branches
      7. Add/Edit/Delete Car Types
      8. Add/Edit/Delete Cars
      9. Add/Edit/Delete Users

\* **Only Deleting an order** will actual remove it from the DB.  
The rest of deleting operations will mark the relevant param in the DB as "isDeleted" for true, because deleting data from the DB is not recommended because of the relations between items, and the fact that we want to keep history, users, cars etc intact and available if for some reason we will need them in the future.

1. **Web site structure**
   1. **Header** with dynamic navigation menu based on the user role  
      1. After a registered user signs is**, its name appears at the header's end**, and hovering over its name, a menu with some will appear:



* + 1. After an Admin is signed in, an **Admin menu** is showing:  
       
    2. When admin / worker signed in, **manage orders** menu shows,  
       so that the worker / admin can handle the cars orders
  1. Main data area  
     will show all the relevant data routed based on the app-routing.module.ts  
       
     Based on the user's permission, and it is signed in, the buttons will show.  
       
     Admin can **edit/add** Cars/Branches/CarTypes/Users directly from the relevant component page.  
     When admin edits an item (from above) he can also delete it.
  2. Footer

**TBD – WILL UPDATE IN THE FUTURE**