CS340 Introduction to Databases

Group Project
by Jason Rash and Grigori Barbachov

Amateur Hour Racing League (AHRL)

http://web.engr.oregonstate.edu/~rashj/index.html

Project Step 6: Portfolio Assignment

Contents

Executive Summary 3
Project and Database Outlines 4
ER Diagram 6
Schema 7
UI Screenshots
- Main page 8
- Drivers page 8
- Licenses page 10
- Tracks page 11
- Races page 13
- Seasons page 14
- Race Results page 16

Back-end API documentation 18

Executive Summary

The original ER diagram included a many to many relationship between Drivers and Seasons with an extra entity called Teams that had a 1 to many relationship with the intersection table of Drivers and Seasons which we called TeamMembers. This relationship was taken out after step 2 as feedback from our grader helped us realize that it wasn't clear what we were trying to accomplish with the relationship. We decided that Teams don't really play a part in racing at the amateur level and are only really there for sponsorship, so Teams could be cut from the project.

We also originally had a many to many relationship between Licenses and Drivers. Our original idea was for Drivers to have many different Licenses based on the type of race. We changed this to a one to many relationship. Now Drivers can only have one License which is upgradable similar to how CDL licenses work for truck drivers.

Project Outline

Amateur Hour Racing League (AHRL) is a 17 year old amateur racing league for beginner to intermediate level drivers that currently holds 30 races per year at 8 different tracks with between 40 and 60 drivers per race weekend. Last year 158 different drivers participated in one or more races. Our database driven website will allow users to search and display information on who the Drivers are, what License they hold, and Race Results of any Races they were in.

Database Outline

- Drivers: records details of Drivers who race in our racing league
    - driverID: int, auto_increment, unique, not NULL, PK
    - firstName: varchar(255), not NULL
    - lastName: varchar(255), not NULL
    - birthdate: date, not NULL
    - licenseID: int, FK Licenses.licenseID

    - Relationship: M:M relationship between Drivers and Races is implemented with intersection table RaceResults (where Drivers has 1:M relationship with RaceResults with driverID as a FK inside RaceResults).
    - Relationship: 1:M relationship between Drivers and Licenses is implemented with licenseID as a FK inside of Drivers.

- Races: records information on Races at our league Tracks
    - raceID: int, auto_increment, unique, not NULL, PK
    - name: varchar(255), not NULL
    - trackID: int, not NULL, FK Tracks.trackID
    - raceDate: date, not NULL
    - seasonID: int, not NULL, FK Seasons.seasonID

    - Relationship: M:M relationship between Drivers and Races is implemented with intersection table RaceResults (where Races has 1:M relationship with RaceResults with raceID as a FK inside of RaceResults).
    - Relationship: 1:M relationship between Races and Tracks is implemented with trackID as a FK inside of Races.
    - Relationship: 1:M relationship between Races and Seasons is implemented with seasonID as a FK inside of Races.

- Tracks: records Tracks where league Races take place
    - trackID: int, auto_increment, unique, not NULL, PK
    - name: varchar(255), not NULL
    - description: varchar(255)

    - Relationship: 1:M relationship between Races and Tracks is implemented with trackID as a FK inside of Races.

- Seasons: records which Season Race events took place
    - seasonID: int, auto_increment, unique, not NULL, PK
    - name: varchar(255), not NULL
    - description: varchar(255)

    - Relationship: 1:M relationship between Races and Seasons is implemented with seasonID as a FK inside of Races.

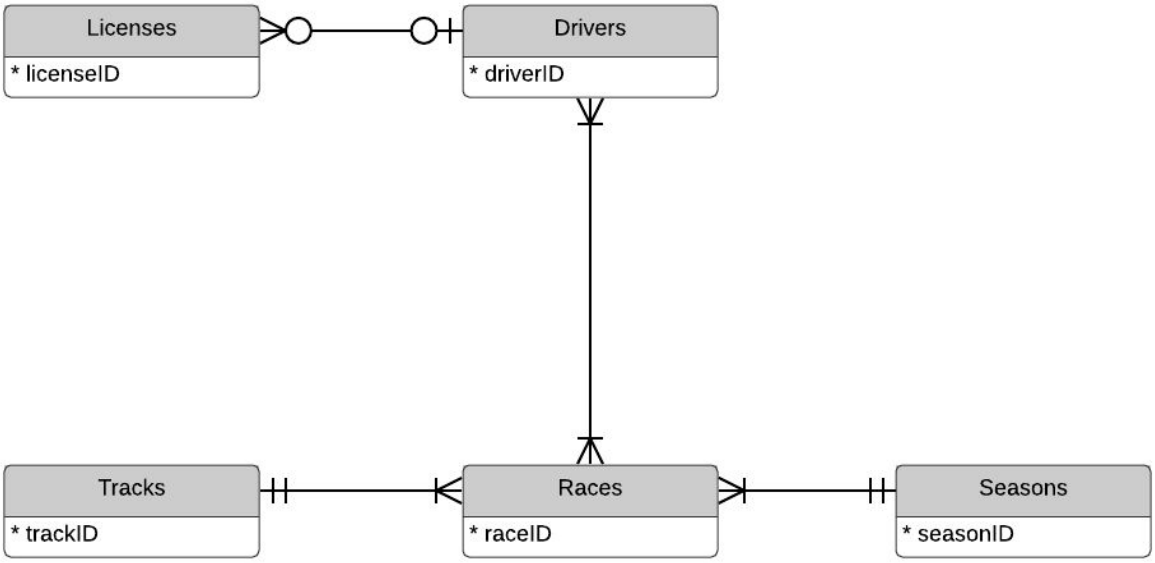
- Licenses: records which Licenses are held by Drivers in the racing league
    - licenseID: int, auto_increment, unique, not NULL, PK
    - name: varchar(255), not NULL
    - description: varchar(255)

    - Relationship: 1:M relationship between Licenses and Drivers is implemented with licenseID as a FK inside of Drivers.

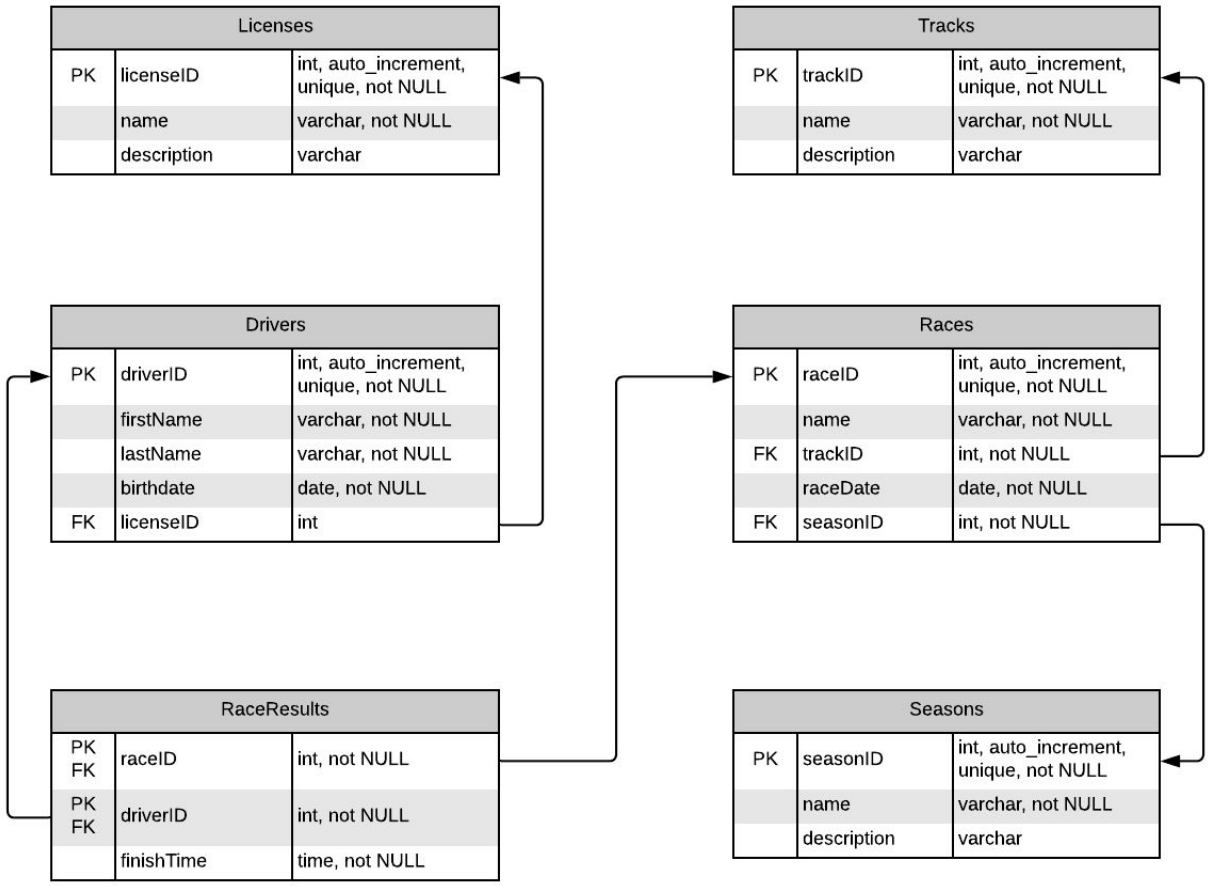
- RaceResults: intersection table that facilitates M:M relationships between Drivers and Races and records finishing time of a particular driver for a particular race.
    - driverID: int, not NULL, PK, FK Drivers.driverID
    - raceID: int, not NULL, PK, FK Races.raceID
    - finishTime: time, not NULL

    - Relationship: 1:M relationship between Drivers and RaceResults is implemented with driverID as FK inside RaceResults.
    - Relationship: 1:M relationship between Races and RaceResults is implemented with raceID as FK inside RaceResults.

Entity - Relationship Diagram

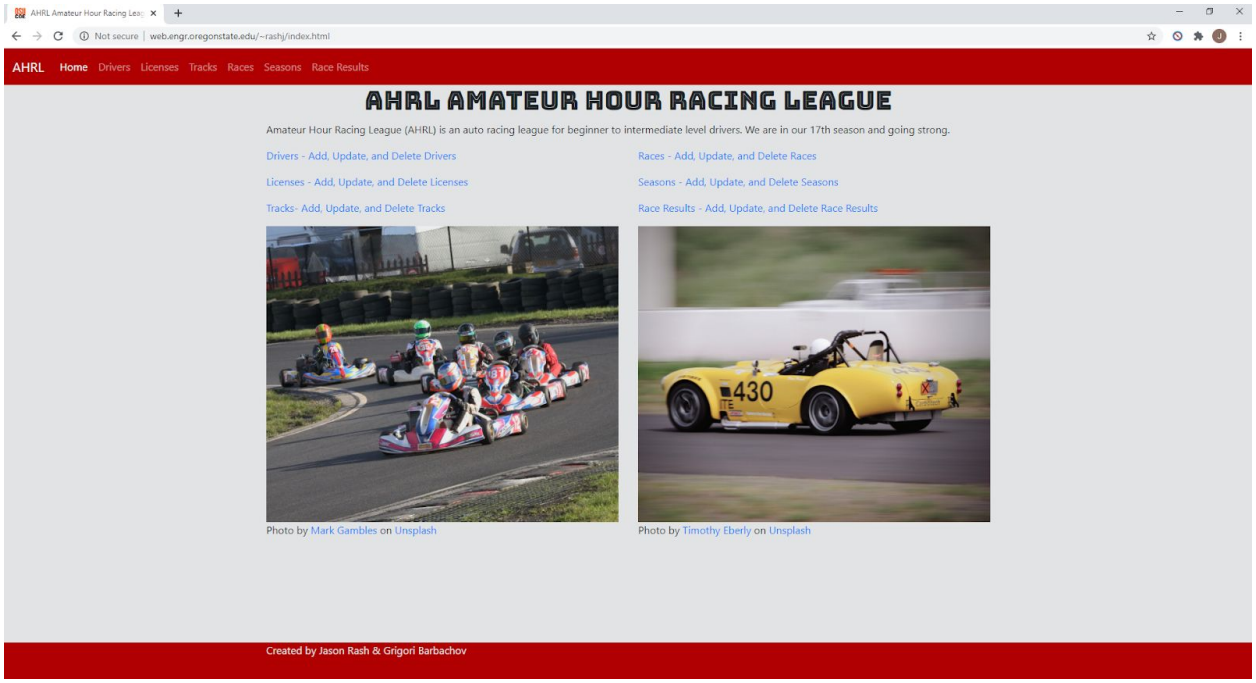


Schema

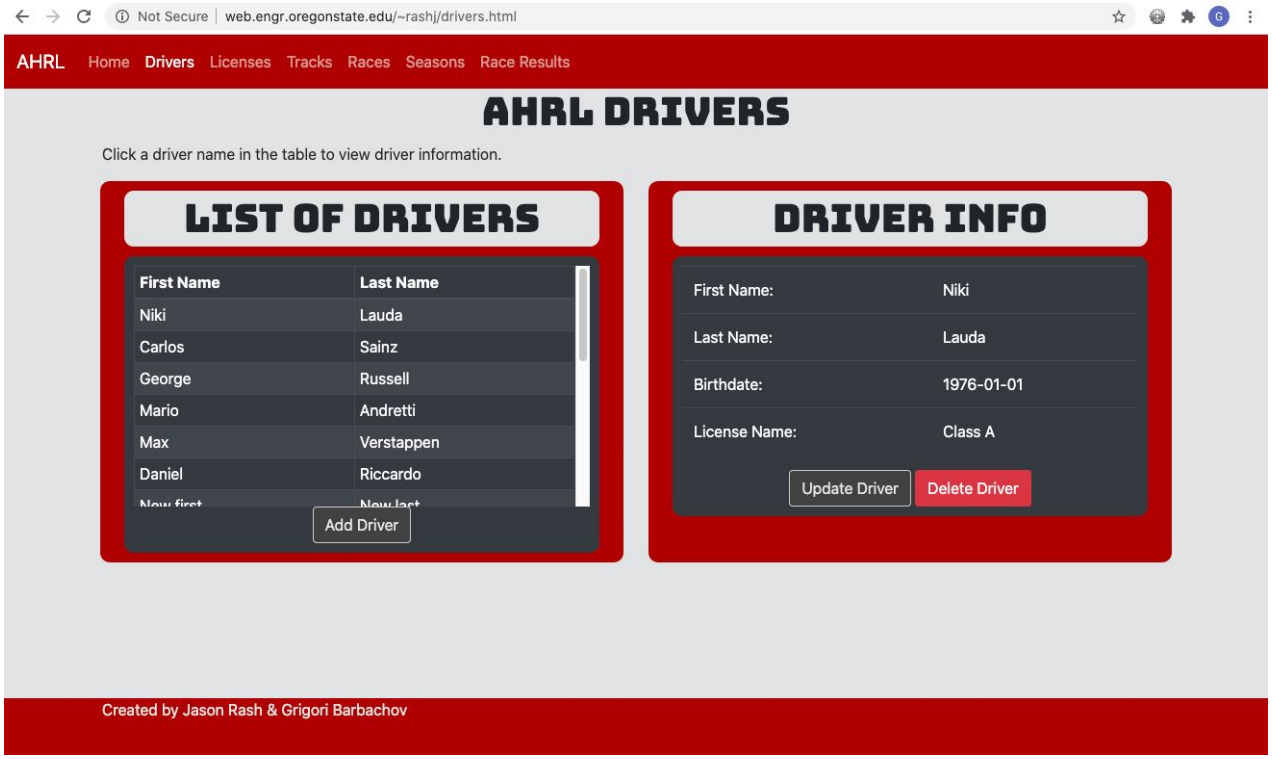

Licenses
PK licenseID int, auto_increment, unique, not NULL
name varchar, not NULL
description varchar
Tracks
PK trackID int, auto_increment, unique, not NULL
name varchar, not NULL
description varchar
Drivers
PK driverID int, auto_increment, unique, not NULL
firstName varchar, not NULL
lastName varchar, not NULL
birthdate date, not NULL
FK licenseID int
Races
PK raceID int, auto_increment, unique, not NULL
name varchar, not NULL
FK trackID int, not NULL
raceDate date, not NULL
FK seasonID int, not NULL
RaceResults
PK FK raceID int, not NULL
PK FK driverID int, not NULL
finishTime time, not NULL
Seasons
PK seasonID int, auto_increment, unique, not NULL
name varchar, not NULL
description varchar

UI Screenshots

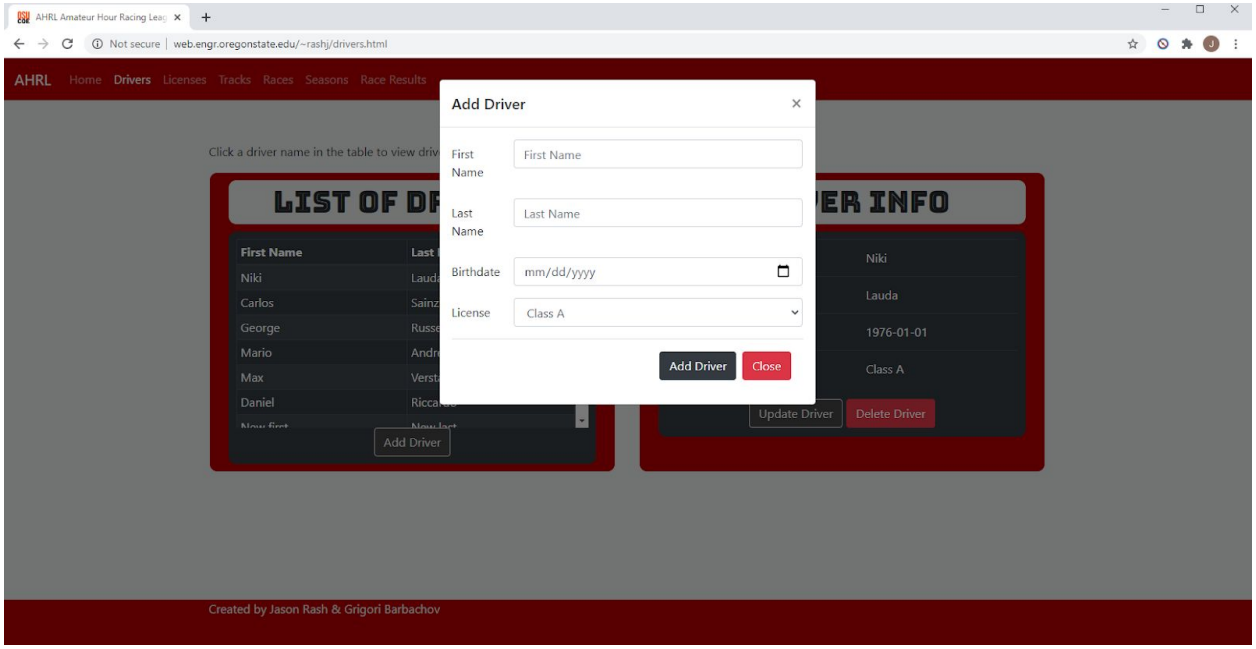
Main Page



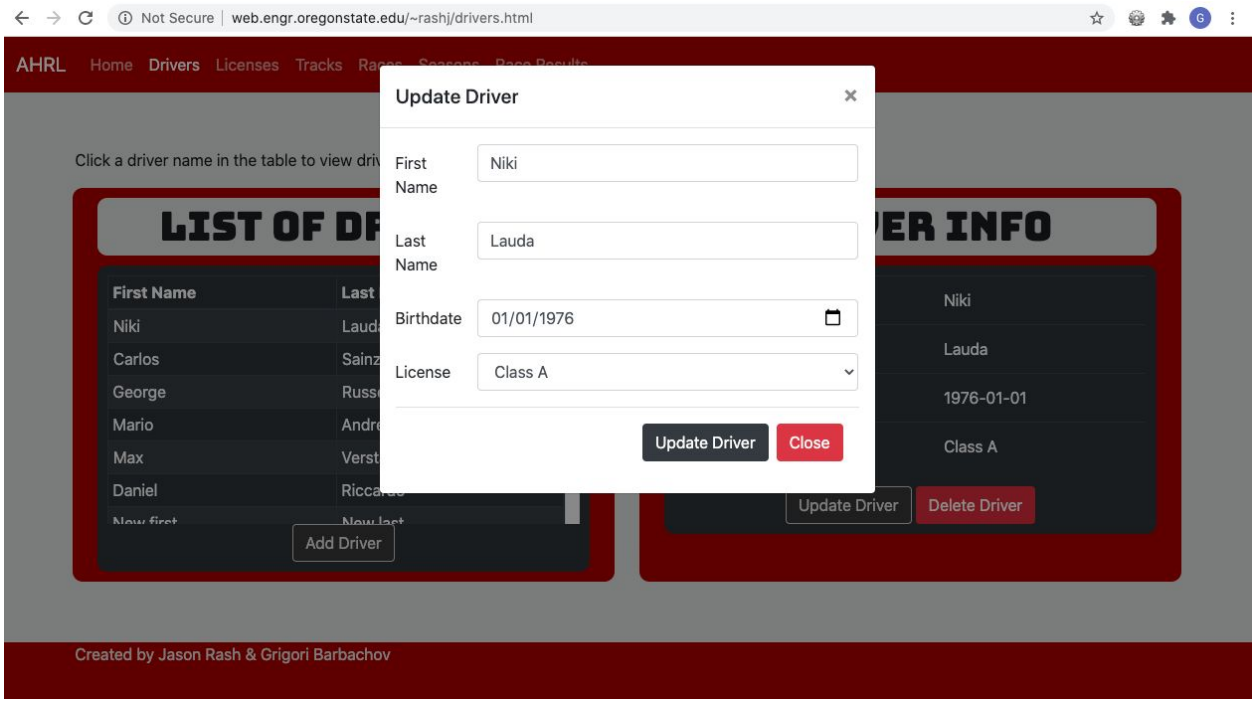
CREATE / READ / UPDATE / DELETE Driver Page



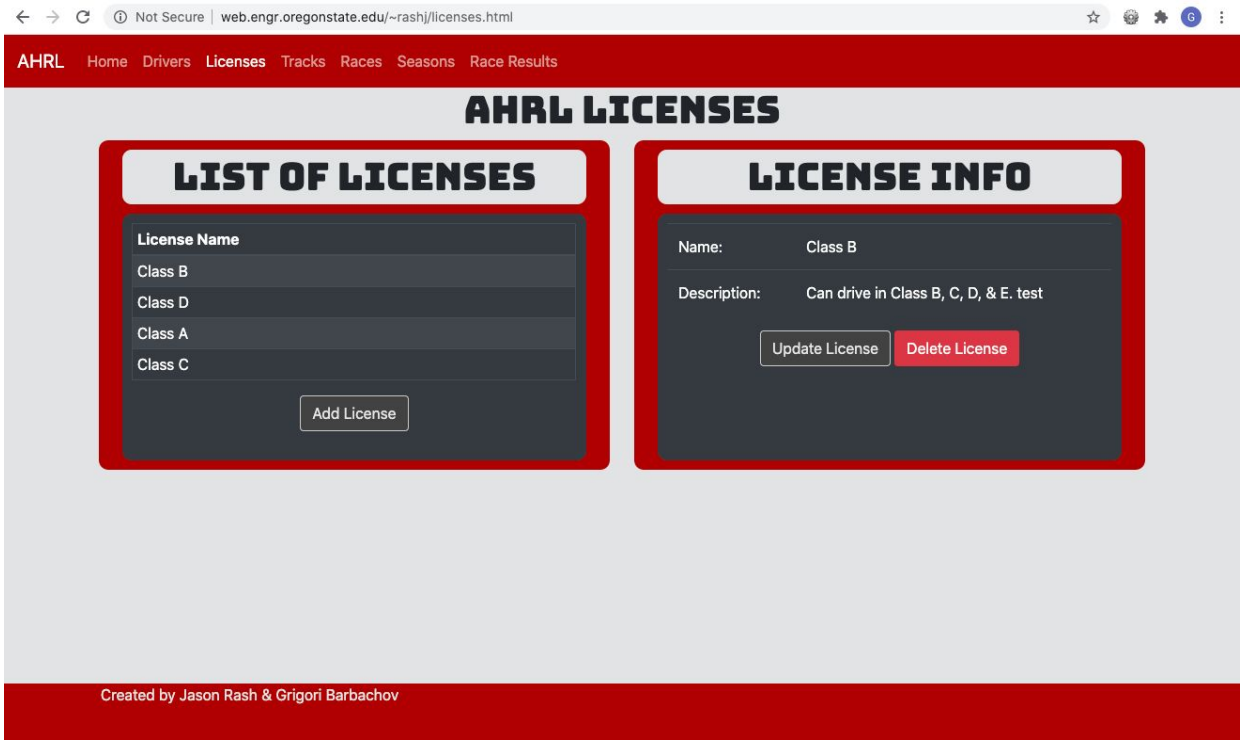
CREATE Drivers Form



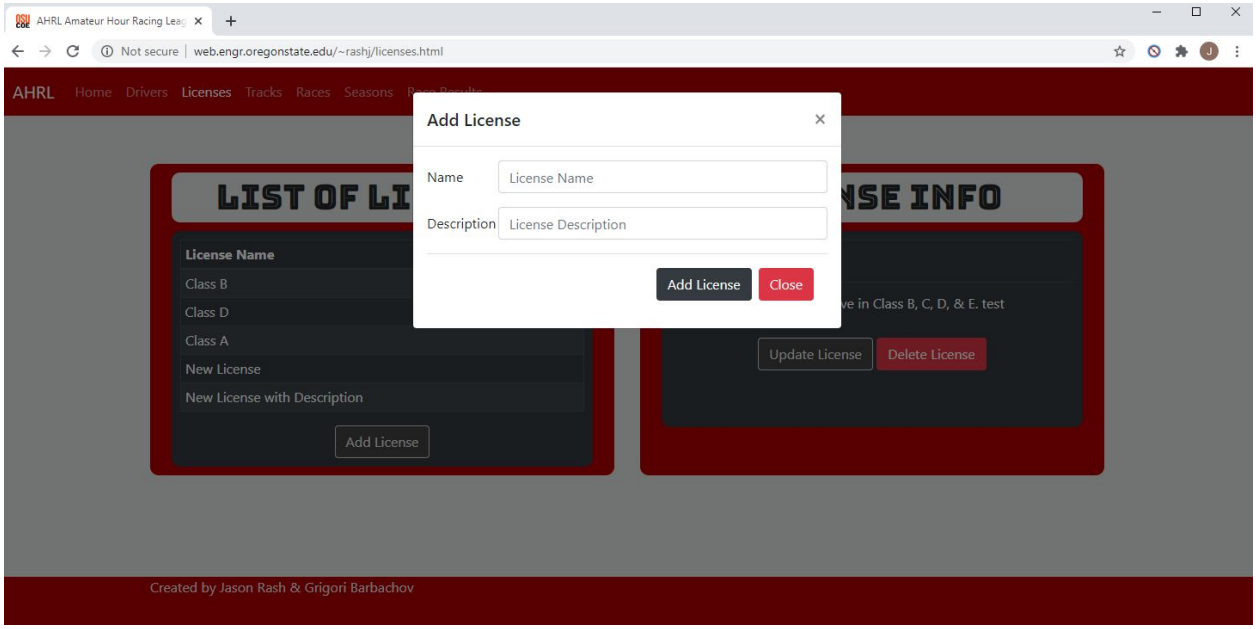
UPDATE Drivers Form



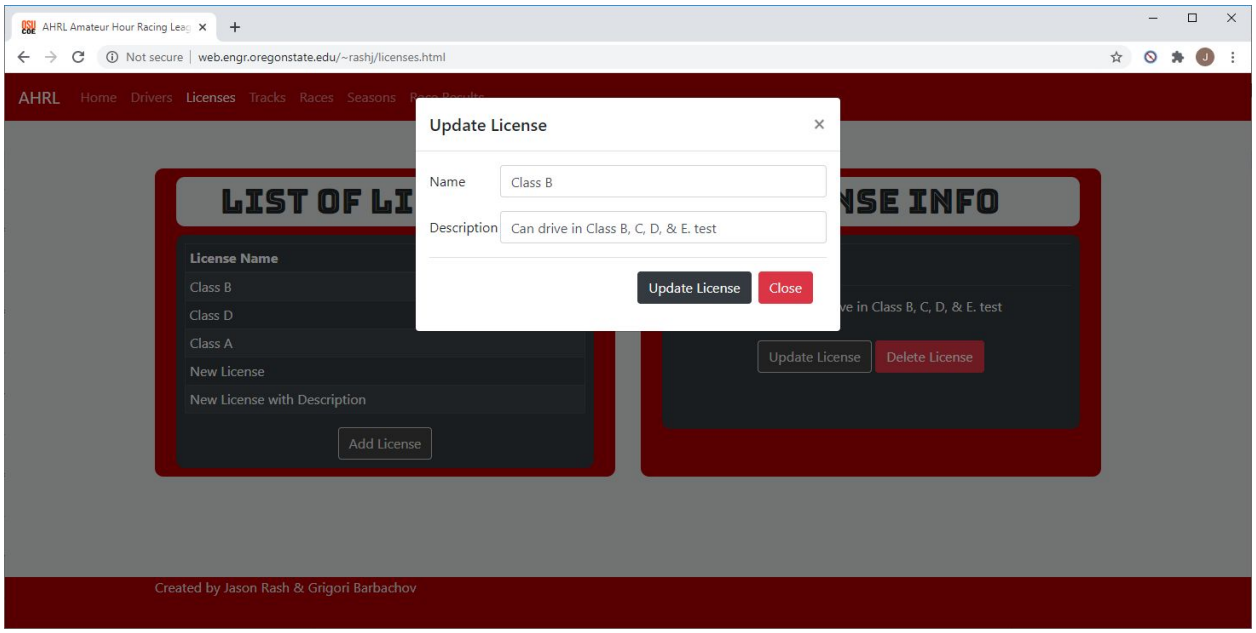
CREATE / READ / UPDATE / DELETE Licenses Page



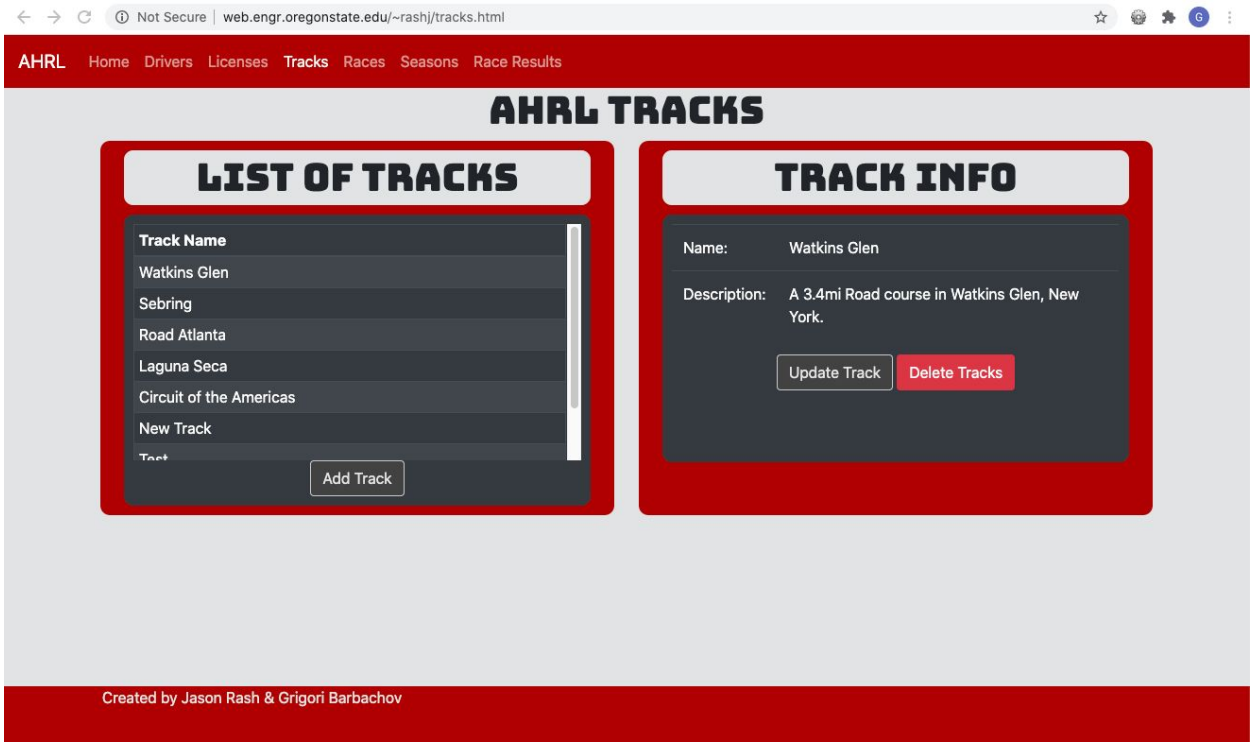
CREATE Licenses Form



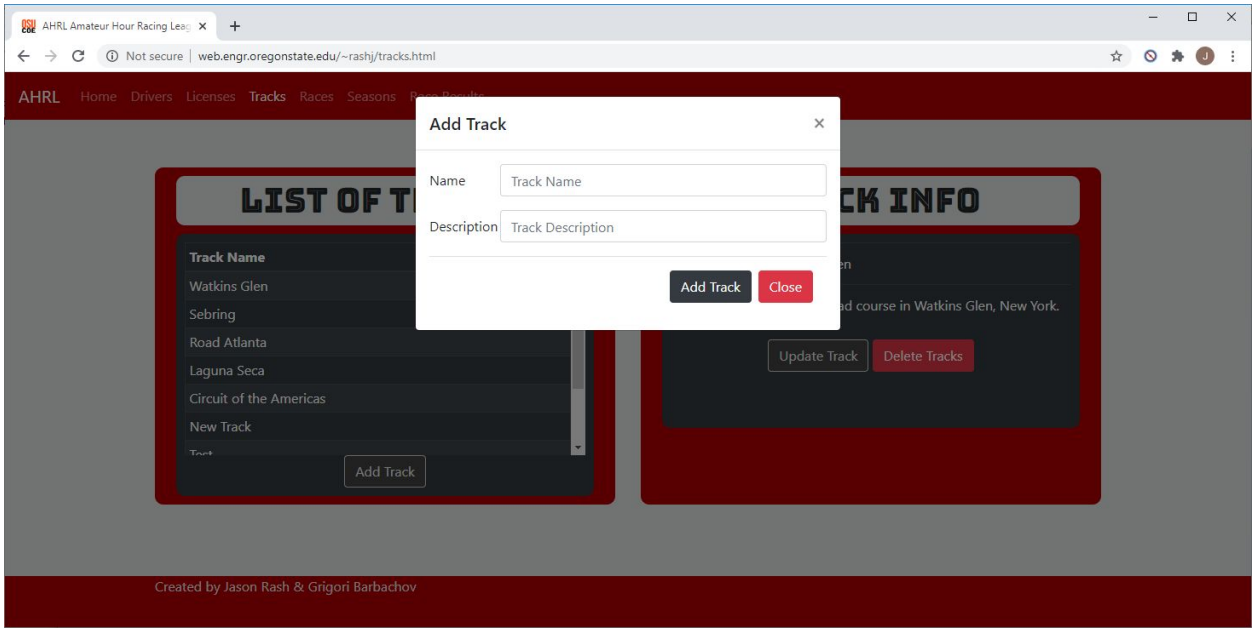
UPDATE Licenses Form



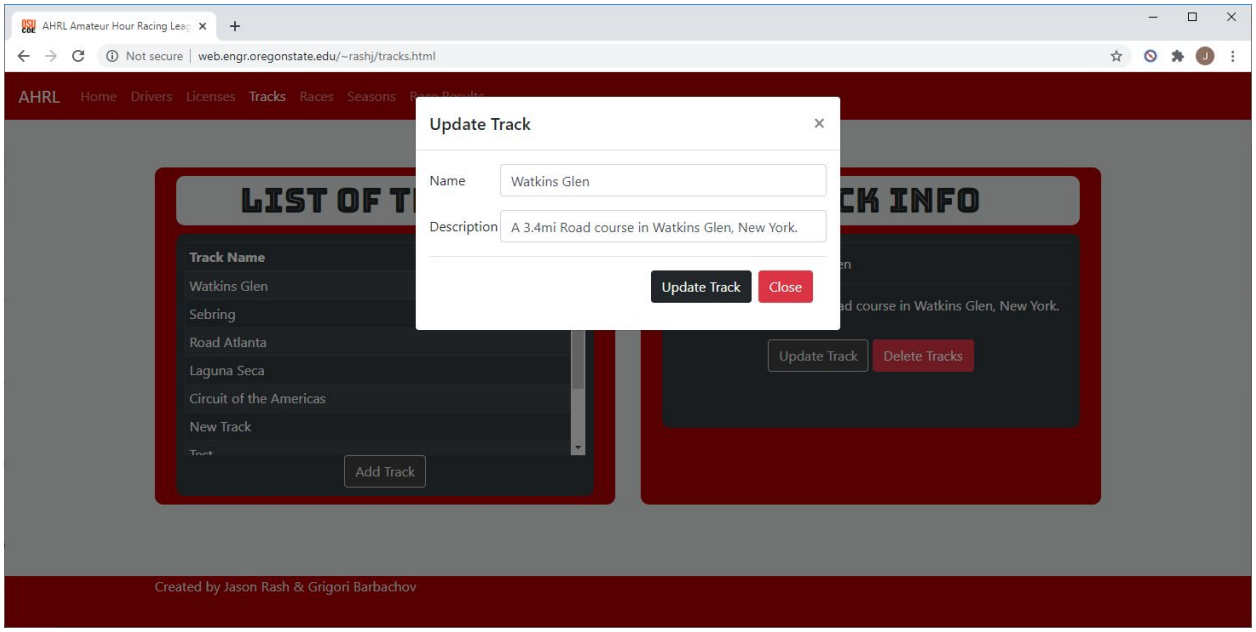
CREATE / READ / UPDATE / DELETE Tracks Page



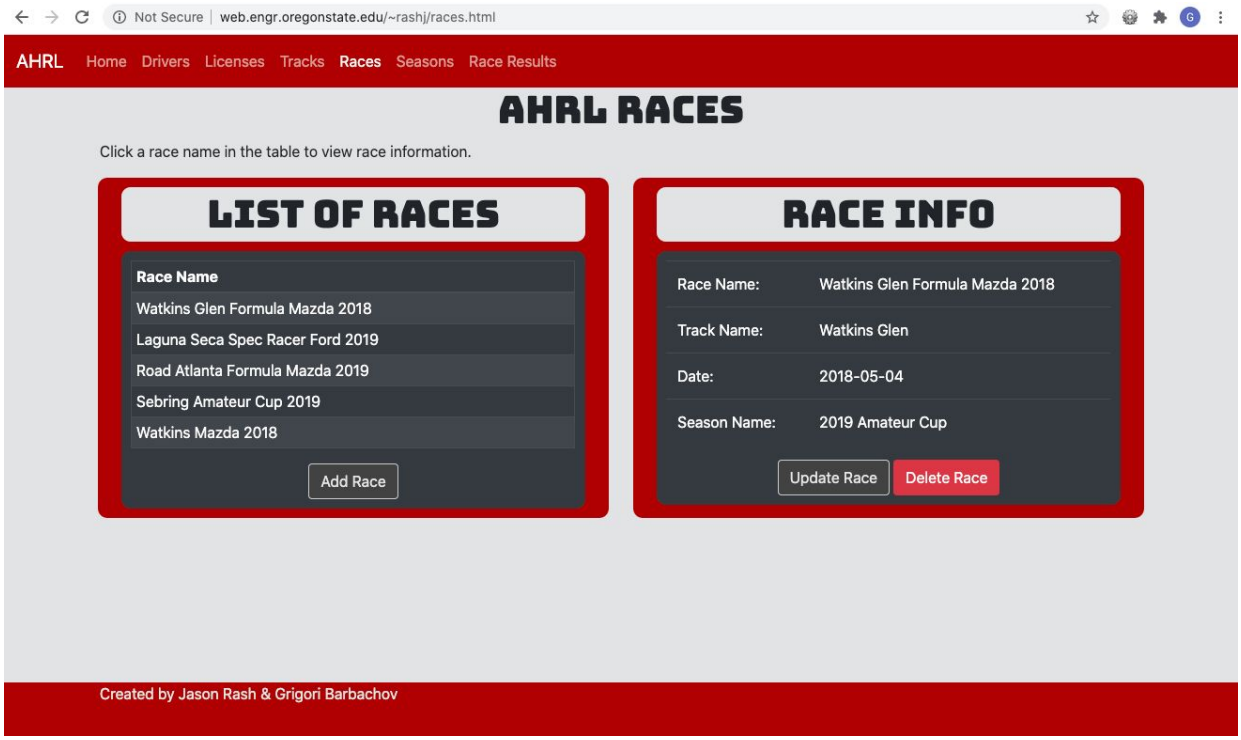
CREATE Tracks Form



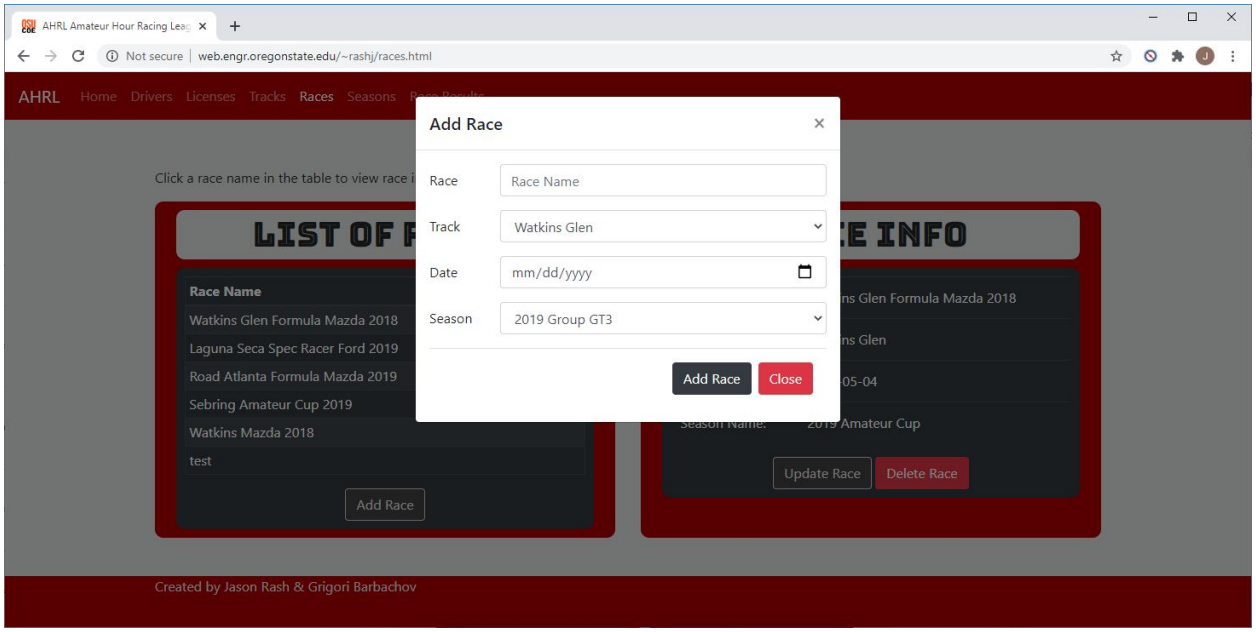
UPDATE Tracks Form



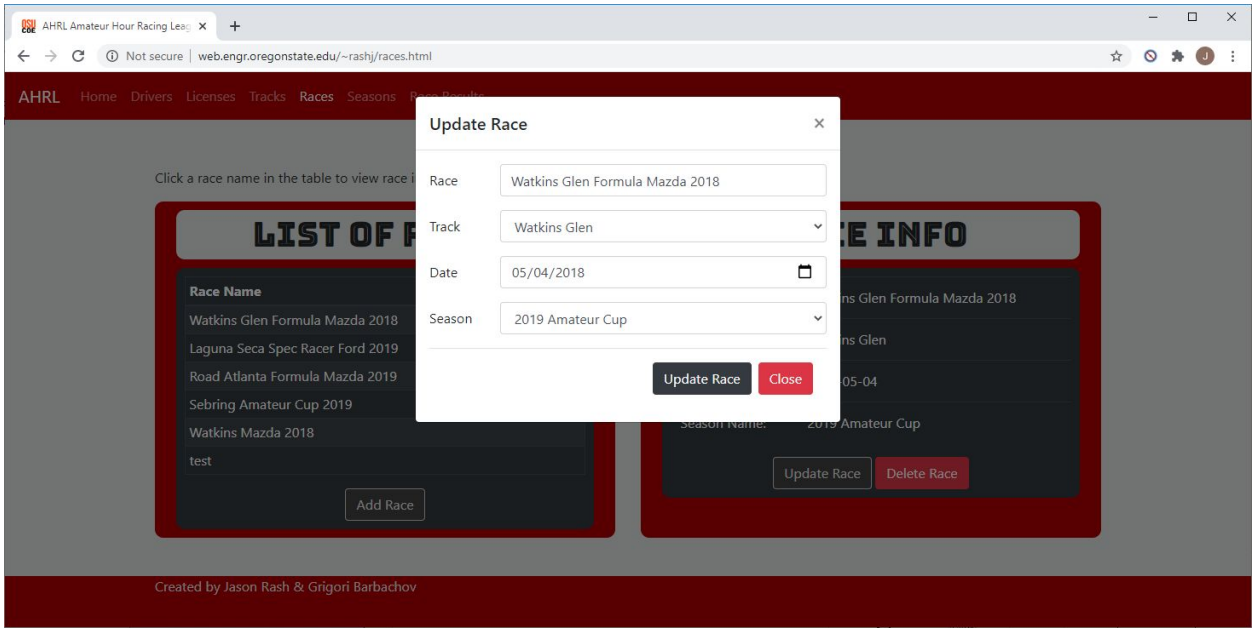
CREATE / READ / UPDATE / DELETE Races Page



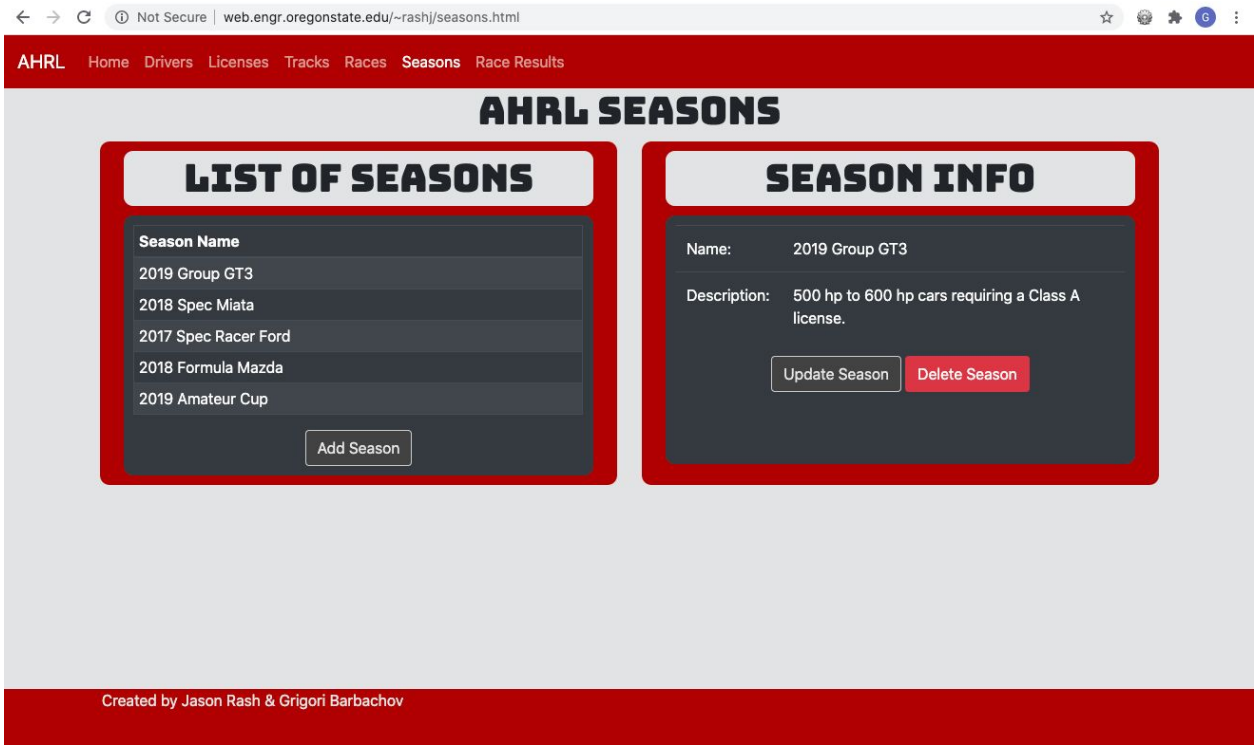
CREATE Races Form



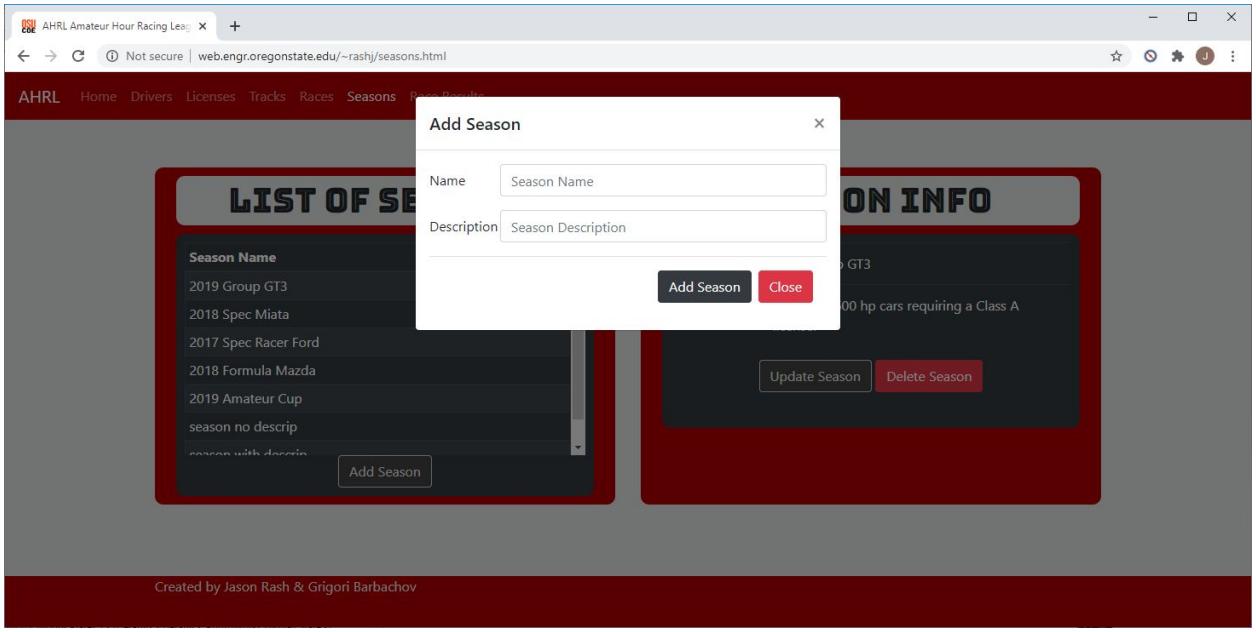
UPDATE Races Form



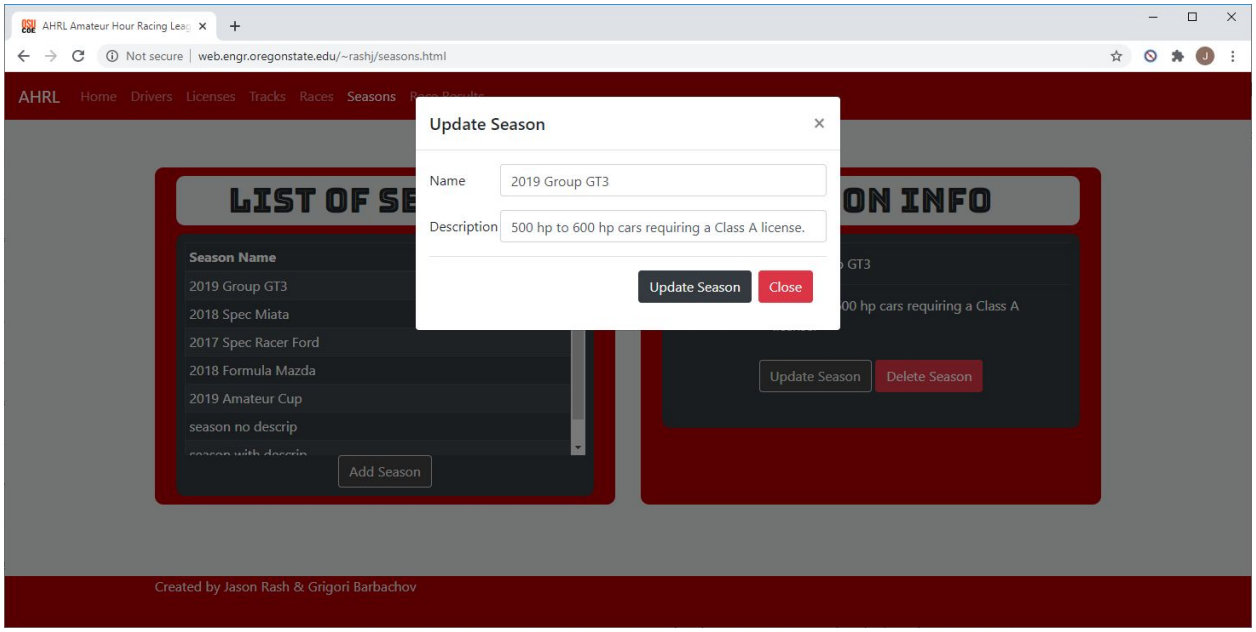
CREATE / READ / UPDATE / DELETE Seasons Page



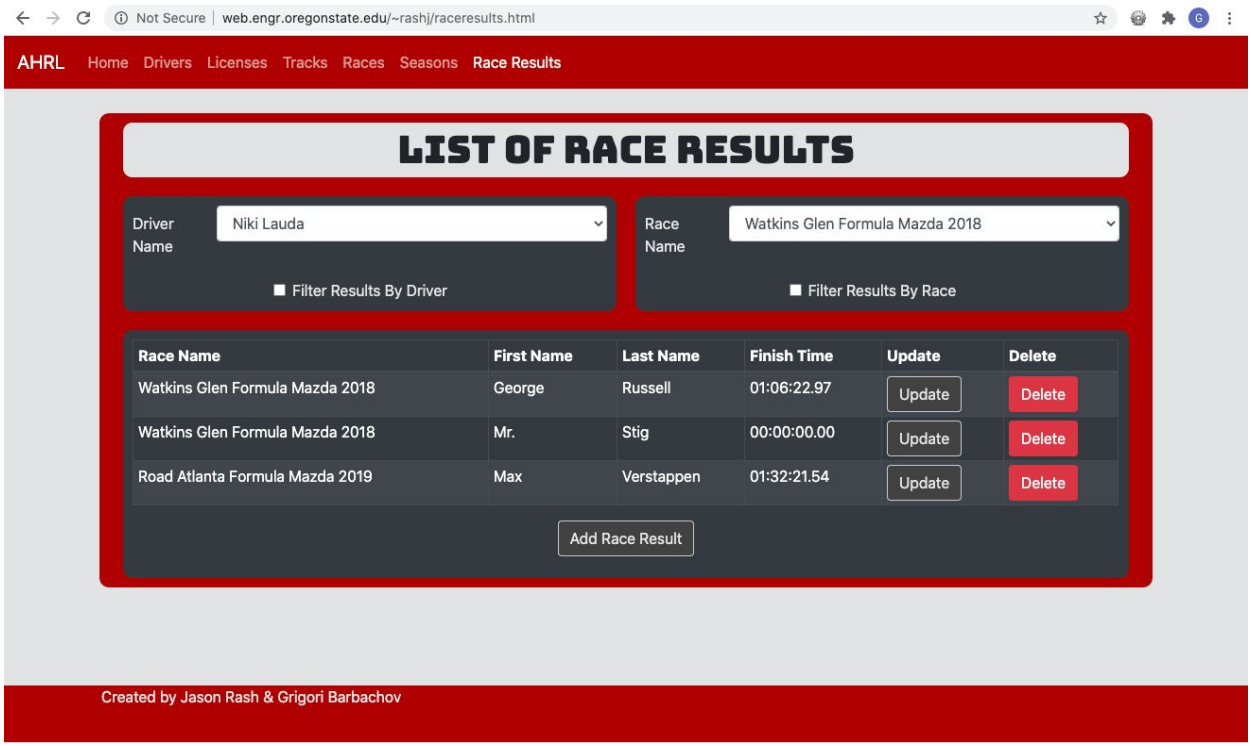
CREATE Seasons Form



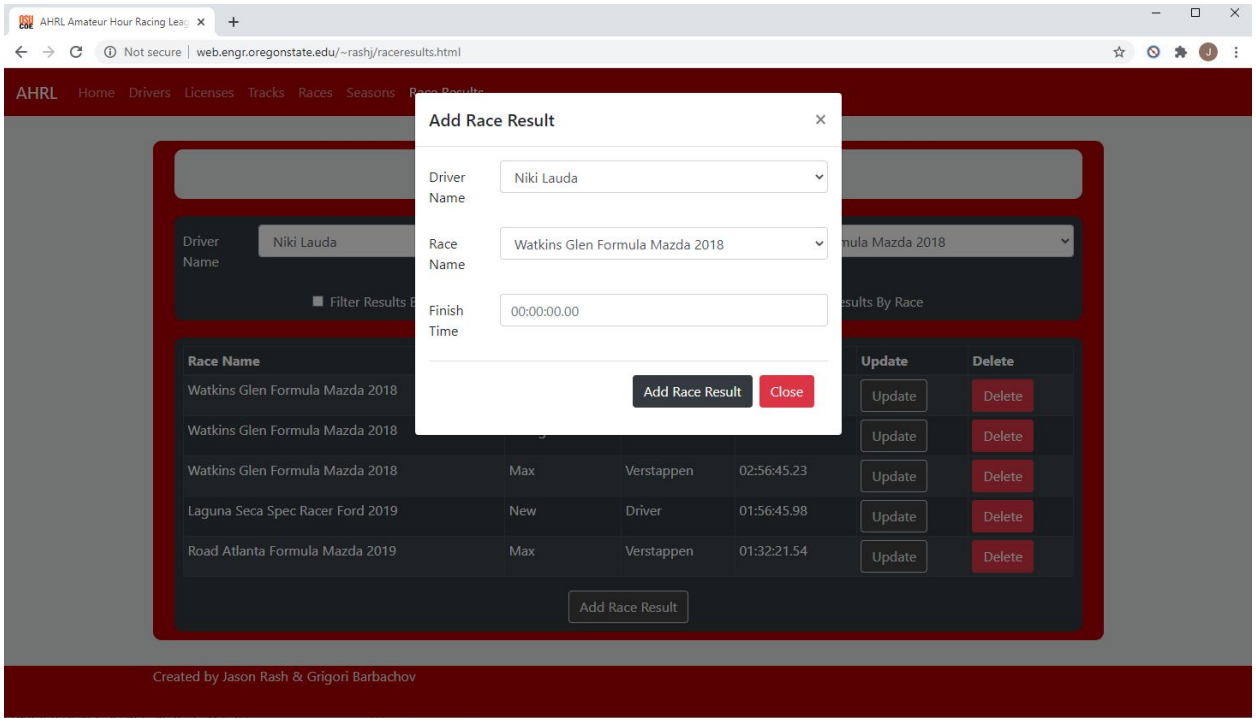
UPDATE Season Form



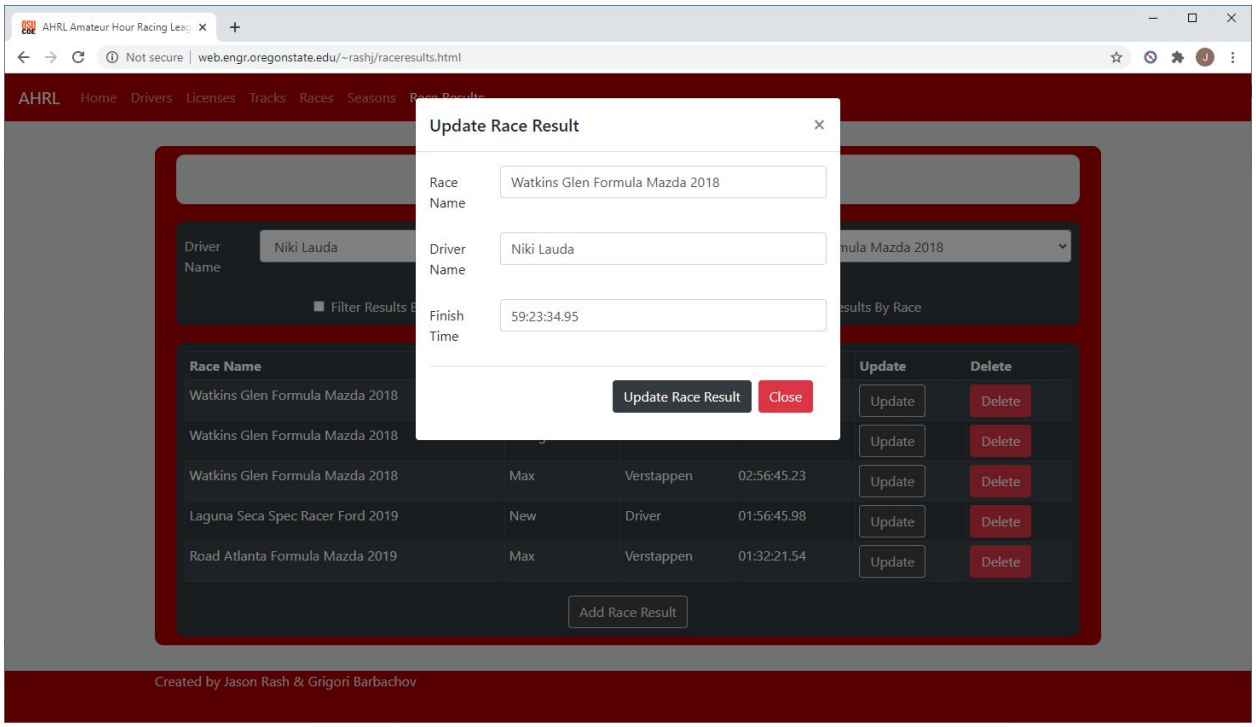
CREATE / READ / UPDATE / DELETE / FILTER Race Results Page



CREATE Race Results Form



UPDATE Race Results Form



Backend API Documentation

All backend API calls are made via a POST request to a single endpoint <api URL>/api. Each call expects a JSON object with three parameters - action, table and params, described below. API returns a JSON object with requested rows from the database.

action	table	params[] * important to be in the the order listed
actionUpdate	Licenses	name, description, *licenseID
actionUpdate	Seasons	name, description, *seasonID
actionUpdate	Tracks	name, description, *trackID
actionUpdate	Drivers	firstName, lastName, birthdate, licenseID, *driverID
actionUpdate	Races	name, trackID, raceDate, seasonID, *raceID
actionUpdate	RaceResults	finishTime, *raceID, *driverID
actionInsert	Licenses	name, description
actionInsert	Seasons	name, description
actionInsert	Tracks	name, description
actionInsert	Drivers	firstName, lastName, birthdate, licenseID
actionInsert	Races	name, trackID, raceDate, seasonID
actionInsert	RaceResults	finishTime, *raceID, *driverID
actionDelete	Licenses	*licenseID
actionDelete	Seasons	*seasonID
actionDelete	Tracks	*trackID
actionDelete	Drivers	*driverID
actionDelete	Races	*raceID
actionDelete	RaceResults	*raceID, *driverID
actionSelect	Licenses	*licenseID, 'APPLY_FILTER'
actionSelect	Seasons	*seasonID, 'APPLY_FILTER'
actionSelect	Tracks	*trackID, 'APPLY_FILTER'
actionSelect	Drivers	*driverID, 'APPLY_FILTER'
actionSelect	Races	*raceID, 'APPLY_FILTER'
actionSelect	RaceResults	*raceID, 'APPLY_FILTER', *driverID, 'APPLY_FILTER'