Project Participants:

Katherine (Kathy) Moss

Title:

Rocket League Database

Executive Summary:

This project is very loosely based on the video game “Rocket League”, where players in cars are pitted against each other in a soccer arena, trying to score goals into the opposing team’s net with a giant soccer ball.

One player can have many different cars with various qualities – this relationship forms the one-to-many table. In the game, a player can have different ranks (like skill levels). However, for the purpose of this DB, the **car** can have different **ranks**, and the car can pick up ranks over time. This creates a many-to-many relationship and a joins table.

Kathy Moss Final Project Proposal 8/24/2023 – *4/9/2024 Edit

Initial Features:

- DB Design contains 4 tables (players, cars, ranks_earned, and car_rank)

  - DB must contain at least 3 tables ✔

- Contains 3 Entities (players, cars, ranks_earned – > Players, Cars, RanksEarned as Entity Classes)

  - DB must contain at least 3 entities ✔

- CRUD Ops:

  Players = C, R

  - R, U (by ID)

  Cars = C, R, U, D

  - C, R (by player ID)

  - R, U, D (by car ID)

  Ranks Earned = C, R* read is planned, but not functional yet, will finish at a later date

Explanation:

- Players – you can add a new player (C) to the DB and look at all players (R), but you can't remove a player's account … once it's there, they are forever in Rocket League
  - Player by ID
    - view a player by specific ID (R)
    - change aspects of the player like banner, border, and anthem (U) given a specific ID
- Cars – can add a new car (C) or view list of cars (R) in player inventory
  - Car by ID
    - view a car by specific ID (R)
    - change aspects of a car, like color and finish (U)
    - remove from inventory (D)
    - Important note: You can not delete all cars from inventory! This is a planned feature.

- Contains all CRUD ops ✔

Kathy Moss Final Project Proposal 8/24/2023 – *4/9/2024 Edit

- 1+ entity has all CRUD op ✔

- RanksEarned – A car can gather ranks (C), but ranks are not removed from a car (unless the car is deleted, then all ranks are deleted), in that regard, ranks are kind of like badges. The rank table exists by itself with all possible ranks. You can view all possible ranks by:
- http://localhost:8080/rocketleagueserver/rank

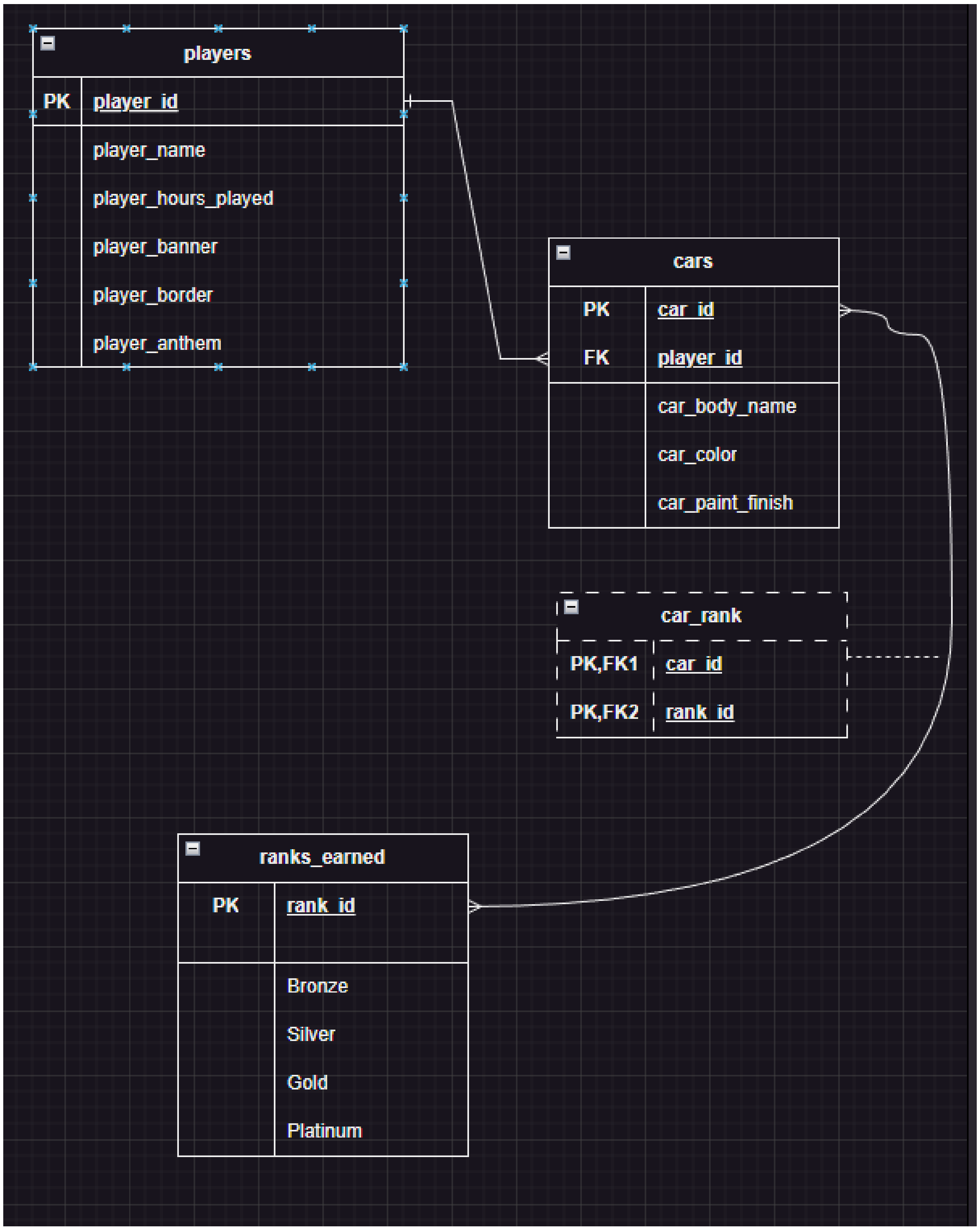
  Many-to-many relationship between cars + ranks:
- - C

- Contains at least 1 many-to-many relationship with 1+ CRUD ops on this relationship ✔

- Each entity should have at least 1 CRUD op ✔

- REST API Server test through: Swagger, Postman, **ARC** ✔

- Database Design –



Planned Features/API Endpoints:

*** one name to tie endpoints together with REST mapping: /rocketleagueserver**

*** all mappings below are a step below under the /rocketleagueserver**

/player ✓

Verb	Path Parameters	Query Parameters	Request Body	Response
POST (C)	None	None	player_name; player_hours_played; player_banner; player_border; player_anthem	201 – A new response has been created with the request body in previous column
GET (R)	None	None	None	200 – shows list of all players

/player/{player_id} ✓

Verb	Path Parameters	Query Parameters	Request Body	Response
GET (R)	player_id	None	None	200 – shows particular player and attributes
PUT (U)	player_id	None	player_name; player_hours_played; player_banner; player_border; player_anthem	200 – shows player with updated attributes

/players/{player_id}/car ✓

Verb	Path Parameters	Query Parameters	Request Body	Response
GET	None	None	None	200 – shows list of all cars for a specific player
POST	None	None	car_body_name; car_color; car_paint_finish	201 – A new response has been created with the request body in

				previous column, added a new car to the garage

/player/{player_id}/car/{car_id}

Verb	Path Parameters	Query Parameters	Request Body	Response
GET	car_id	None	None	200 – show a specific car by player and car ID with specific attributes
PUT	car_id	None	car_body_name; car_color; car_paint_finish	200 – shows car by ID with updated attributes
DELETE	car_id	None	None	204 – No Content, removed from garage

/rank

Verb	Path Parameters	Query Parameters	Request Body	Response
GET	None	None	None	200 – shows all possible ranks and their ID's (the data contained in the RankEarned table)

/car/{carId}/rankEarned – covers join table ops

Verb	Path Parameters	Query Parameters	Request Body	Response
GET (R)	None	None	None	200 – shows list of all ranks earned by a player
POST (C)	car_id	None	rankId	201 – A new response has been created with the request body in previous column

// Note: ddl-auto: create; mode: always --- > done with everything needed for DB? : ddl-auto: update; mode: never

AKA:

- Entities are correct

- Tables are created properly

=> changing the ddl settings makes sure the app will only update tables and not drop them and create them again

Stretch Goals (to be completed if time allows, or after graduation):

- Make a more detailed video showcasing my work
- Do not allow duplicate entries for ranks for player or car
  - Ex: car 1 can not have 2 bronze entries, player 1 can not have 2 bronze entries
- Add a README file
- GitIgnore ?
- Create a frontend for the backend, which requires:
  - getting reacquainted with FE
  - learning more React
  - making sure everything is optimized for different screens
  - knowing how to tie the FE to BE
- Add a READ operation to a car's rank, by car ID