Module 5- Computer Systems (2021-22)
Project

UNIVERSITY OF TWENTE.

Software Design Document Template

Project Name: RC Racing System	**Team ID:** 22
Team Members: Kristiyan Velikov, Alex Petrov, Laurens Neinders, Rick Pluimers, Liran Neta, Kağan Gülsüm	**Mentor(s):** Alex Mo, Mohammad Assad

Introduction

Our Web Application enables management of racing, also among friends. The timings of every lap and sector will be recorded and shown to the users. We aim to give the users data that will drive them even more to play and help them to figure out what are their pros and cons in their racing sessions.

The Web Application can be a great entertainment for individual RC lovers, friends, and even as a family activity.

Functional/Non Functional Requirements

Functional requirements:

*The functional requirements of a **RC Racing System** are:*

- *The user should be able to sign-up.*
- *The user should be able to login.*
- *The user should be able to start a race.*
- *The user should be able to add friends.*
- *The user should be able to remove friends.*
- *The user should be able to challenge a friend.*
- *The user should see his/her lap times.*
- *The user should see his/her times in sectors.*
- *The user should be able to see a global leaderboard.*
- *The user should be able to see a leaderboard of friends.*
- *The user should be able to see the ongoing time, while he/she is racing.*
- *The system should perform user authentication in order to log in **(Security requirement)**.*
- *As a user, I can quickly and securely log in to the system using a 3rd party account (for example Google) **(Security requirement)**.*
- *As a user, I can access the UI I have logged in **(Security requirement)**.*
- *As an admin, I want to check the log files to see what happens if something is out of the ordinary **(Security requirement)**.*
- *In case my files are deleted by hackers, I want to be able to recover the system's data **(Security requirement).***
- *As a user, I don't want my personal information to be known by unknown parties **(Security requirement).***

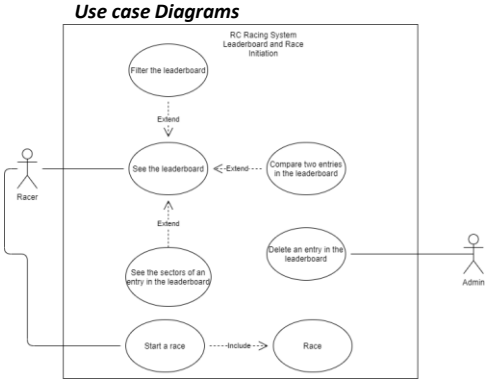
Non functional requirements:

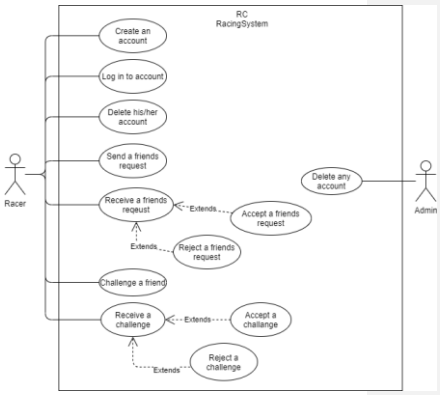
*The non-functional requirements of the **RC Racing System** are:*

- *The delay of time recording should not exceed 0.5 sec.*
- *The systems should be user-friendly.*
- *The system should not limit the number of users who can sign-up.*
- *The system should not limit the number of users who can be online at the same time.*

Architectural Design

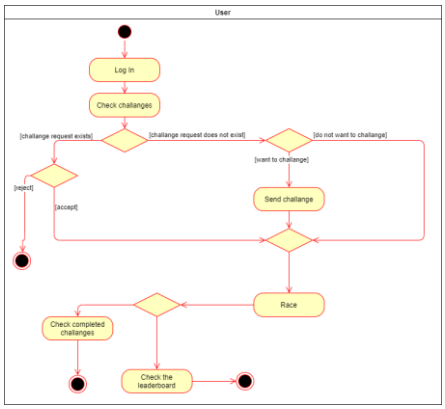
Use case Diagrams

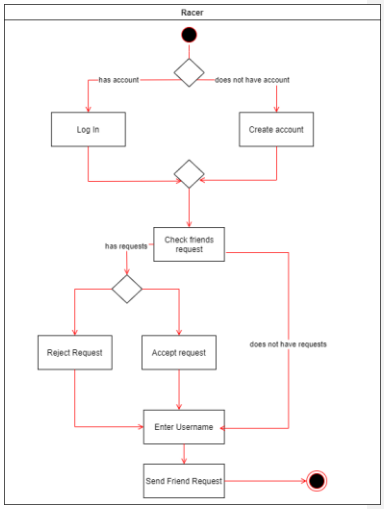




The left diagram shows use cases of how the racer can see, filter and compare times in the leaderboard. Also starting a race is present and the functionality of the admin to delete an entry is present. The right diagram concentrates on the social aspect of the application showing the friends system and also challenges.

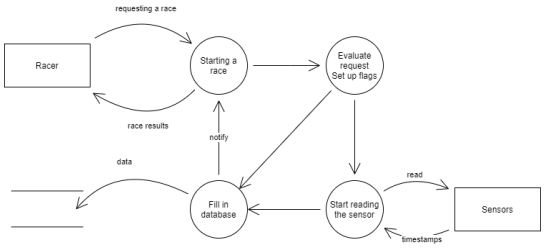
Activity Diagrams





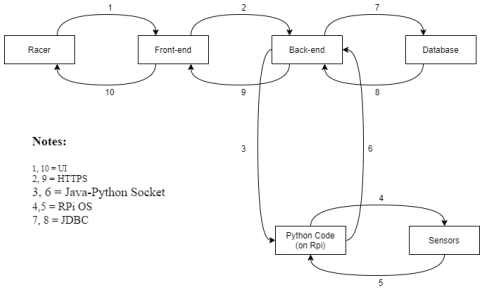
In the left activity diagram, the process of challenging is explained, if you have been challenged, you can accept if and race. if not, you can decide to send a challenge and then race for this challenge. The diagram on the right explains how the friend requests system works. You can send, accept and reject friend requests.

Data Flow Diagrams



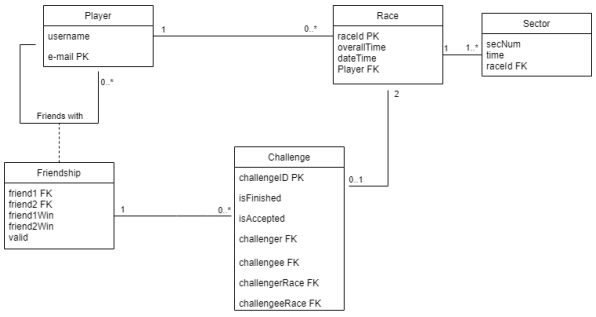
This diagram presents the communication between the front-end, back-end, database and the Raspberry PI. After requesting a race from the UI, the back-end notifies the database which after setting a flag, communicates to the PI the start and the python script starts reading the sensors. Then all the data is propagated back, recording the race time to the database and notifying the user through the UI.

Overall Understanding diagram



This diagram came about when we were scratching components of the system on the paper. Basically every rectangle is a main component. The lines represent the channels of communication and we added a little note explaining where those communications take place. Also we numbered the lines in the order of their execution. The scenario we realized with the numbering was starting a normal race.

Class Diagram



The class diagram represents the main classes of our database. A player can have other players as friends. A player can be in a race which records the current date and time and also the race result. A race will have multiple sectors with their own times. A race and two friends (friendship) can make a challenge which records the two friends' results and also if a challenge is accepted and finished.

Product User Interface

Initial Interface using Figma

We used the platform Figma for designing a prototype version of our planned Web Application.
All the screens of the system were designed:

- *Login Page*
- *Main page (including the "start a Race option")*
- *Ongoing race timing*
- *Leaderboard + Friends leaderboard*
- *Challenges (Done, Waiting, Sent)*
- *Friends (Friends' list, Add friends)*

Hereby attached a link to the Figma prototype showcase (including the interactions between the screens):
https://www.figma.com/proto/sP3cOV8cImU3CoYImclEZB/Mod5-Team-22-team-library?node-id=316%3A4&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=316%3A4

Module 5- Computer Systems (2021-22)
Project

UNIVERSITY OF TWENTE.

Prevention/Mitigation Criteria (Security Controls)

- ***Trying to race while there is an ongoing one.***
  - *Because there can only be one track active at any given moment. We'll need a singleton class with some static flags that we can check when the front-end sends us a request. If there is a race, we inform the user; if there isn't, the python script is notified, and we wait for the user to race.*
- ***Spamming a challenge.***
  - *A test takes place when a challenge is being sent. Regardless of whether the user attempting to send the challenge is a challenger or a challengee, the system checks if there is an unresolved challenge between those two individuals.*
- ***Deceiving the sensors.***
  - *Deceiving of sensors is a problem that is not easily solved, some measures against it are: checking if the order in which the sensors are triggered makes sense; checking if the times that were recorded roughly make sense. The only option that can truly mitigate this risk is to have a referee at the track to check if everything is going according to the rules.*
- ***Forging an illegal request to the back-end.***
  - *OAuth supplies us with tokens which will be checked for every request to be sure that the user is authorized and input sanitization shall also take place. Also the logic behind the request must also be checked. (for example: a race request is sent where the challenge flag is set but the receiver name is null.)*

The cost involved (if any):

The costs involved in this project is the price of the RC Car, sensors, and the track material (possibly bricks for the walls). The time that will be spent is related to the delivery times of the aforementioned elements.

Conclusion:

1. *This document concludes our architectural & visual design of the system, which we are going to implement in the upcoming weeks. The design was done by all the team members, divided into Architecture ("Back-End") team and User Interface ("Front-End") team.*
   We took critical decisions together, as a group, and kept on working in order to finish the design before the deadline.
2. *Sensor decision wasn't made yet, due to lack of knowledge and time.*
3. *Class Diagram: Sensor Class - We contemplated about the design of the system and its DB. We realized that having a separate object which represents a sensor is more wise than having x elements of sectors inside a Race object (based on the OOP practices).*
   The reason for that - scalability & flexibility. For instance, using the Sectors as objects, we can assign the same sector to multiple races more easily.
4. *Challenge for the next phase - implementing the communications between the Sensors, the RPi, the Back-End & the User Interface.*

Commented [1]: To change?

5. *Product User Interface. For the initial user interface we went through different designs and ideas. In the end we decided to keep our interface simple and minimalist.*

*(You should give the **concluding remarks** of your document. You can do this by **highlighting noteworthy design decisions** and **challenges** for the next phase that you recognized.)*

Reference:

- ***Links for diagrams***
  https://www.lucidchart.com/pages/data-flow-diagram
  https://www.lucidchart.com/blog/data-flow-diagram-tutorial
  https://www.lucidchart.com/pages/uml-activity-diagram
  https://en.wikipedia.org/wiki/Activity_diagram
  https://www.uml-diagrams.org/use-case-diagrams.html
  https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case

- ***Firmware software***
  https://www.figma.com/