MTAT.03.295 - Agile Software Development

Regular Exam - 10 January 2023

Part 2. Practice

General notes:

- You are **allowed** to access any resource on the web.
- You are **NOT allowed** to communicate with anyone during the exam in any way (except with the lecturers).
- You **MUST create** a **PRIVATE repository** at https://gitlab.cs.ut.ee, and add the teaching staff as collaborators (with admin rights).
- You should submit a TXT or PDF file with the link to your repository with the solution to the Moodle submission. Please leave your solution in the "main" branch of your repository, and do not make any commits to "main" after the end of the exam.
    - Username: orleny85, email: orlenys.lopez.pintado@ut.ee
    - Username: chapelad, email: david.chapela@ut.ee
- IMPORTANT: You MUST **commit to the repository each time you complete a task** or requirement in the exam. You are welcome to add other commits, for example, if you fix a task completed before. For each missing commit, you will get a penalization of 20% of the points received corresponding to the task. Note that commits whose messages do not correspond to the task will also get a penalization of 20%. For example, if your commit message says, "Task 2, Completed", be sure that the code submitted corresponds to task 2; however, other parts of the implementation may be updated too. Finally, we will discard all the commits submitted after 13:30 (EEST time). These are the minimum required commits and messages (NOT necessarily in the same order). Note that each message must correspond to a different commit:
    - Initial Commit
    - Task 1 Completed
    - Task 2 Completed
    - Task 3 Completed
    - Task 4 Completed
    - Task 5 Completed
    - Task 6 Completed
    - Task 7 Completed

- IMPORTANT: We strongly suggest you **not shareing your code before the end of the exam** to avoid misunderstandings. In the case of projects where it is evident that the students committed fraud, all the involved students will get 0 points in the exam, meaning they all fail it. Also, if we observe suspicious behavior either in the implementation or in the commits, the students involved will be called for an online interview with the teaching staff. Examples of suspicious behavior can be all the commits happening at the end of the exam or with a non-realistic timeline, too many similarities between projects, etc. During the interview, the students will be asked about the code they submitted. Note that, during the interview, we will downgrade the student's mark for each question not answered correctly, meaning that the student may fail the exam in the meeting.

NEED FOR VELOCITY

You have been hired to work in what will be the best driving simulator of all time: **Need for Velocity.** For the first release of the simulator, we ask you to implement a Phoenix application with a basic set of features to handle racers and a race system.

The application must satisfy the following requirements:

R1. A racer is defined with:

- **racer_id:** is a unique identifier code formed by 5 characters, starting from a capital letter A-Z, followed by 4 integer digits 0-9. For example, "A0000" and "B1923" are valid. However, “A00”, “11111,” and “Racer 2” are invalid and should not be allowed,
- **speed:** is an integer number in the range (0, 200], including 200, excluding 0,
- **boost:** is an integer number in the range [0, 100], including 100, including 0,
- **risk:** is a floating number in the range (0.0, 1.0], including 1.0, excluding 0.0,
- **points:** is a non-negative integer number representing the score accumulated in all the races in which the racer participated.
  <u>NOTE</u>: The system must reject any attempt to insert or update any racer violating any of the abovementioned constraints.

R2. The system must allow to **organize a race** as follows:

- To create a race, it must be specified **i)** a float number greater than 250 with the ***distance*** of the circuit, **ii)** a float number in (0.0, 1.0] representing the ***minimum_risk*** participants in the race must take, **iii)** two integers setting the range between a ***minimum*** and ***maximum*** amount of ***points*** (both inclusive) required to participate.
- The system must check that the numbers in the input are valid; otherwise, the error message “***Invalid Race Setup***” must be displayed.
- If the input is valid, the system automatically selects to participate all the racers (in the database) whose points are between the minimum and maximum amount passed as input. Among those candidates, the racers with risk below the minimum risk received as input are *disqualified*. The remaining one continues in the race. However, if the non-disqualified racers are below 3, the system displays an error message "***Not enough racers to compete***", and the race does not occur.
- For each remaining player, the system calculates the time it took them to finish the race by subtracting their “boost” from the distance of the circuit and dividing the result by the speed: ***(distance - racer.boost) / racer.speed***. The top three participants are those with the lowest times among all the non-disqualified participants.
- As the output of the race, the system increases the points of the top 3 racers with 5, 3, and 1 points, respectively. Those values must be updated in the database. Then, the system must display a new template, listing the ranking of all the racers in the database (participating or not in the last race), sorted in descending order by their

points (higher first). This template must display only the racer ID and their points. Additionally, the system must show a message containing the top 3 racers and the number of disqualified ones.

Note: In case of a tie, you can decide the top 3 players in any order.

Tasks to perform:

- **Task 1 (4 points)**: Specify a Gherkin user story describing requirement **R2** (organizing a race) in which the top 3 racers are selected. As a minimum, your feature must include the clauses Given, And, When, and Then. Your user story must specify that 4 racers are already in the database (providing a table with their data). Then, it follows the information required to organize a race (passed from the Gherkin description, i.e., not hard-coded in the white_bread steps) and the action that triggers such creation. Finally, it should check that the corresponding message is displayed to the user (also passed from the Gherkin description).
- **Task 2 (4 points)**: Implement the *white_bread* steps described by the user story in Task 1.
- **Task 3 (2 points)**: Set up the application routes to support organizing a race and display the rankings of the racers in the database.
- **Task 4 (6 points)**: Set up & implement models (via migrations) and seed the database with initial data based on your models. The database seeding must include a set of racers available to compete. Your models must include all the validations described in **R1**.
- **Task 5 (12 points)**: Implementation of controllers. You must implement the operations to render the templates to provide the input data for organizing races and displaying ranking. You must also implement all the operations and validations, showing the messages to notify the success/failure as described in **R2**.
- **Task 6 (4 points)**: Implementation of the view(s) and template(s) to organize races and display the racer's rankings. You do not need to create/render a new template to display the message (you can just write in the *flash*, as we did during the course practical sessions).
- **Task 7 (8 points)**: Unit/Integration test checking the organization of a race (**R2**) ending with exactly one racer disqualified, exactly one racer not participating for not having the required points, and three racers making the top 3. Remember that the test should check that the values in the database were updated as expected (a single test may include several assertions):
  - The winner got their score increased by 5 points.
  - The racer in second place got their score increased by 3 points.
  - The racer in third place got their score increased by 1 point.
  - The non-participating/disqualified racers keep the same score as before the race.
  - The output message contains the required information, i.e., the top 3 and how many disqualified racers.