P3. Sprint 3

ASSIGNED: 5 April 2024
DUE: 11:59PM, Thursday, 2 May 2024

100 points

1. Objectives

- Implement the specific use cases in the frontend using JavaScript and React.
- Implement test cases for the whole application.

The frontend communicates with the backend using http requests that are handled by the RestController implemented in P2. You are free to choose layout, theme, fonts, and color for the frontend. However, you are required to implement the frontend to support the following use cases.

2. Use cases

The use cases are enumerated below.

1. View company qualifications. (3 points)
2. View company employed worker. (3 points)
3. View company projects. (3 points)
4. View qualification details. *(Given a specific qualification, the details include description and workers that have that particular qualification.)* (4 points)
5. View worker details. *(Given a specific worker, the details include name, salary, current workload value, projects they are assigned to, and their qualifications.)* (4 points)
6. View project details. *(Given a specific project, the details include name, size, status, assigned employees, required qualifications, and missing qualifications. Missing qualifications can be visualized by color coding the required qualifications: e.g., red = missing, green = satisfied.)* (4 points)
7. Create new qualification. *(This new qualification should become visible via use case 1.)* (3 points)
8. Create new worker. *(This new worker should become visible via use case 2.)* (5 points)
9. Create new project. *(This new project should become visible via use case 3.)* (5 points)
10. Assign worker. (4 points)
11. Unassign worker. (4 points)
12. Start project. (4 points)
13. Finish project. (4 points)

You should test each of these use cases. Moreover, you should create sample workflows that incorporate these use cases. For example:

Create a new project with a few required qualifications. Assign a few workers. Start the project. Unassign workers. The project should now be in the suspended state. Assign workers. Start the project. Finish the project.

---

3. Analyzing, Coding, and Testing

Describe each use case separately before you implement them. Use a consistent format, which could be a 2-column Actor-Action vs System Response table or a single column table that shows what steps an actor takes and what is the system response. Show alternative courses of action.

From these use cases, derive test scenarios for (1) testing individual use cases, and (2) workflows composed of these use cases. You can use numbered lists with tables or paragraphs showing the test scenarios.

---

4. Grading criteria

Team scores will be based on both the product and the process. Individual adjustments will be made based on quality of contributions.

4.1. Team scores

- Process component (25 points)
  1. Pull requests
  2. Issues
  3. Branches
  4. Commits
  5. Commits to main (Should be none)
  6. How the work is distributed across the sprint (waiting to work until the end is bad)

- Product component (75 points)
  1. Use case report (10 points): Your use cases must be in a document called `P3_UseCases.md`.
  2. Test case report (10 points): You need to describe how you test each use case and document the test scenarios that you created in a document called `P3_TestCases.md`
  3. Quality of your implementation (50 points): We will test your implementation of the frontend using a live demo. We will set up time slots and you will sign up. All (or at least a majority) of the teammembers must participate in the demo. Each use case must appear in the demo, and will be worth 3-5 points depending on the complexity of the use case.
  4. Reflection report (5 points): We will grade your reflection (`P3_reflection.md`)

4.2. Individual Adjustments

Adjustments can be positive up to 20 points or negative up to 100 points (especially when there is no activity) and for reasons such as:

- low individual process score,
- bad peer reviews from teammates,
- gaming the system (e.g., purposefully increasing the number of commits without actually imlementing much functionality).

4.3. GitHub Actions

We use automated Github Actions to compile and tests the code that you push code to Github. You must never push broken code to the main branch, in other words you must never "break main". Github Actions are one of the tools that we use to calculate your grade. You are not allowed to add/modify/delete any GitHub action, workflow, or workflow run.

- If you modify a workflow file, we are going to see it in the repository git history, and your team is going to lose points.
- If you delete a workflow run, we are going to see that there is a missing workflow run for a specific 'git push', and your team is going to lose points.