University of Zurich[UZH]

HASEL

Software Praktikum (SoPra) - FS22

Milestone 2 - Assignment

1 General Information

The focus of Milestone 2 is on designing, specifying, and planning the upcoming implementation phase of your project. The deadline is on Sunday, **27.3.2022 23:59 CET** and includes the deliverables listed below. On the following Monday, **28.3.2022**, you will present a short progress update for Milestones 1 and 2 (combined).

1.1 Deliverables Overview

Report: as PDF with a name of form *FS22-Group-XX-M2-Report.pdf*.
Presentation: as PDF with a name of form *FS22-Group-XX-M2-Slides.pdf*.
Application URLs: Heroku, GitHub and SonarQube URls submitted via OLAT form.
Source Code and Projects Board: hosted on GitHub (public; with git tags "M2").
Emotion Skills Survey: individually completed here[1].
Brownie Points: individually submitted via OLAT.

2 Assignment Description

During Milestone 1, you familiarised yourself with the infrastructure and formulated user stories for your application. Based on feedback for the M1 report and the created artifacts, you will now refine the user stories and decide which ones to include in your first Sprint. Further, you will create UML diagrams, the interface specification, and UI mockups. After this milestone, you should have a clear understanding of how the implementation phase of Milestone 3 should look like.

2.1 Diagrams

Based on the user stories from M1, create a design for the web service using component, class, and activity diagrams with the UML standard.

[1] `https://uzhwwf.qualtrics.com/jfe/form/SV_d5xzUsNQQcvlgpg`

Component Diagram: Define the architecture of your application using a component diagram [2]. These diagrams are used to visualize the organization and relationships among components in a system. There are at least three major components in your application (i.e., client, server, and database). Define these components and how they interact. Specify interconnections among components utilizing provided and required interfaces.

Class Diagram: Domain modeling helps you to identify the relevant concepts (or entities) that describe the domain of your system (e.g., in case of the board game *Rummikub*, "Game" and "Player" are examples of entities). Use a class diagram to define the domain model of the system. A domain model is a set of major classes that represent the application (including an indication of what will be persisted). Notice that you do not have to model the auxiliary classes used in the SpringBoot ecosystem (e.g., repository, controllers, or services). Choose the granularity of the class diagram in a way that the combination of it and the user stories serve as a good basis for the implementation of the system.

Activity Diagram: Model the overall workflow of the system with an activity diagram. Choose the right granularity, such that your diagram reflects the most important activities of the application (e.g., "performing an action"; for *Rummikub*, that could be updating the board or drawing a tile from the pool). An activity diagram is used to represent the flow from one activity to another. Its purpose is to identify which activities (i.e., functions provided by the system) you have to implement and how they are associated with constraints and conditions.

2.2 Specification of the REST Interface

For the communication between the client and the server, a REST interface will be used. You have to create a specification for this REST interface. In particular, describe the entities with their operations, parameters, parameter types, returned values, and HTTP status codes. For an example of a REST specification, please refer to assignment sheet 1. With a precise specification, developers of the client and server should be able to work independently on their tasks.
Advice: Use the methods (HTTP verbs) as they were intended to. Use HTTP status codes to indicate correct or incorrect requests. Optionally, design your interface in a way such that it follows the HATEOAS principle[3] as described in this guide[4].

2.3 User Interface Design – Mockups

For this part of the assignment, you have to create **mockups for the whole user interface** of your future web application. Describe the flow among the different screens of your application. Specifically, describe how your application transitions from one screen to another (e.g., a lobby screen is shown after a successful login).
Advice: You can create the mockups with a dedicated tool if you wish. Examples are: Figma[5], Lucid Chart[6], Adobe XD[7], or Balsamiq Mockup[8] (platform independent). For some tools (e.g., Figma) you should use your UZH email address to register in order to profit from student packages and to use the tool collaboratively.

[2] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/
[3] https://restfulapi.net/hateoas/
[4] https://spring.io/guides/gs/rest-hateoas/
[5] https://figma.com
[6] https://www.lucidchart.com/pages/examples/wireframe
[7] https://www.adobe.com/products/xd.html
[8] http://balsamiq.com/products/mockups/

2.4 Setting Up Development Infrastructure

During this milestone, one group member has to prepare the development infrastructure ahead of the implementation phase in Milestone 3. This task is fulfilled if your group has one commit in your client repository that changes the user interface to display your group name on the deployed client.

GitHub and Heroku
To set up GitHub and Heroku, we will reuse the setup described in the individual assignment of M1, adapting only minor details:

1. First, create a new GitHub organization[9]. Name the organization after your group. Use the following naming convention: *sopra-fs22-group-XX*. Invite your team members and your TA to the organization (as "Owner").
2. Next, create the client and server template repositories as described in assignment sheet 1. Make sure to create the repositories inside the GitHub organization and set them to *public*.
3. For deployment, set up Heroku as described in assignment sheet 1.
4. Finally, check-in the change to display your group name and set git tags named "M2" on the main branches.

SonarQube for Code Quality
Additionally to the basic setup, we will use SonarQube to measure the quality of our code. More precisely, we use the free cloud services from SonarQube for open source projects.

1. One team member has to create an account on `https://sonarcloud.io` by clicking the GitHub icon (i.e., "analyzing your GitHub repo"). Follow the installation guide an set up both the server and client projects.
2. In both GitHub repositories, open `.github/workflows/deploy.yml` and uncomment the environment variables (tokens) and the necessary build step. Notice that `GITHUB_TOKEN` is a default environment variable which is auto-generated (no manual intervention needed).

If your configuration is successful, the next time you push code to your remote repositories on GitHub (main), your SonarCloud dashboard should show a quality assessment of your code.

2.5 Scrum Setup on GitHub

After having established user stories in Milestone 1, we will refine and transfer them to GitHub using *Issues* ([10]) for managing them during Milestones 3 and 4.

1. On GitHub, enter each (revised) user story as a GitHub Issue to the *server* repository. Use the title field for your user story in the role-goal-benefit format.
2. In the description of the user stories, use *Markdown*[11] to create a *Task List* of acceptance criteria. Ensure that your user stories are not too broad, often indicated by very big lists of acceptance criteria.

To keep an overview of all user stories and development tasks, which we will define next, we will set up a GitHub Projects board. [12]

[9] `https://help.github.com/en/github/setting-up-and-managing-organizations-and-teams/creating-a-new-organization-from-scratch`
[10] `https://guides.github.com/features/issues`
[11] `https://guides.github.com/features/mastering-markdown/`
[12] `https://docs.github.com/en/issues/trying-out-the-new-projects-experience`

1. Create a new GitHub Projects board on the organizational level (under the tab "Projects"; select the beta version). Make sure to set its visibility to "public".
2. Select the hidden fields "Repository" and "Milestone" to render them visible.
3. Define new fields named "Type" (single select items: "User Story" and "Task"), "Priority" (single select items: "high", "medium", "low"), "Time Estimate (h)" (number), and "Week" (iteration; set the start date to the first weekly TA meeting date after M2).
4. Finally, add all "user story" Issues from the server repository to the board. Set the "Type" field accordingly.

SoPra follows the Scrum methodology [13]. In Scrum, software development is organized in units called "Sprints". Each Sprint is a time-boxed effort. The duration of a Sprint is usually 2 to 4 weeks. Since you are not working full time on your project, we choose the Milestone meetings as Sprint deadlines. Next, we will set up the Sprint for Milestone 3.

1. Create a GitHub *Milestone* named "Sprint 1", set its end date to the Milestone 3 deadline.
2. Add the GitHub Milestone to user stories you plan to work on during Sprint 1 (Sprint Backlog).
3. Visualize the Sprint Backlog by creating a new GitHub Projects *View* of type "Board" and with filter "Milestone: Sprint 1". Save the modified View.

During a Sprint, we will work on *development tasks*, which are part of a decomposed user story. A development task is a smaller, more manageable entity and should be assigned to a single team member.

1. In the description of all user story you selected for the upcoming Sprint, create a Markdown Task List of development tasks.
2. Each development task item can now be turned into its own GitHub Issue. Hover over items of the Task List and click "Convert to Issue" (you may need to refresh the browser tab). Notice how auto-generated "development task" Issues are tracked by the (parent) "user story" Issue.
3. Transfer "front-end" development tasks to the client repository.
4. Add all "development task" Issues to the GitHub Projects board. Set the "Type" field accordingly.
5. Development tasks require prioritization and a time estimate. Set the fields accordingly. As a rule of thumb, each development task should not take longer than a day (it is better if they are a bit shorter).

Advice: With Scrum, you should have a "shippable product" after each Sprint. In your Milestone 3 presentation, you are expected to show a running application with a basic feature set. Therefore, we suggest doing most implementation work during Milestone 3.

3 Grading and Deliverables

SoPra is a pass/fail course and the grade for M2 will be pass/fail as well. Overall, you have to pass 3 out of 4 milestones, where M1 and M4 have to be passed. You need to hand-in reasonable reports for all the milestones. You will receive feedback on your deliverables including an assessment (either pass, borderline pass, or fail) in the upcoming week after the deadline.

[13] `https://en.wikipedia.org/wiki/Scrum_(software_development)`

Report
The report should be submitted as PDF to OLAT with a name of form *FS22-Group-XX-M2-Report.pdf* by the group leader. It should comprise 3 UML diagrams, a REST specification, and mockups. We will evaluate the quality and completeness of your UML diagrams as well as the clarity of your mockups. Regarding the REST interface, we will evaluate the quality, practical feasibility, and completeness of the proposed APIs. Please make sure the title page contains the group name, group leader, and information about all group members (name and matriculation number).

Presentation Slides
The slides should be submitted as PDF to OLAT with a name of form *FS22-Group-XX-M2-Slides.pdf* by the group leader. The presentation must include results from Milestones 1 (group phase) and 2 and should not take more than 4 minutes (hard cut-off). We suggest focusing on clearly introducing your project. Further, use user stories and mockups to walk your audience through the key functionality of your proposed application. The title slide should consist of the group name and the names of all group members. Underline the name of the presenter(s) on the title slide. The slides and the presentations have to be in English. Please note that each team member has to present at least once (M1+M2, M3, or M4).

Application URLs
The URLs of your application deployments on Heroku are submitted via a form on OLAT by the group leader. In the same OLAT form, you will also be able to enter the URL to your GitHub and SonarQube organizations. The infrastructure setup is complete when the front-end of the deployment displays your group name.

Source Code and Projects Board on GitHub
The source code on GitHub is submitted by setting your repositories to "public" and adding git tags "M2" to commits in the "main" branches that should be taken into consideration for grading.

Similarly, your GitHub Projects board is submitted implicitly and must be set to "public". The board should show your Product Backlog, including user stories and development tasks, and provide a view of your upcoming Sprint Backlog. We will assess your Sprint planning both in terms of the covered development effort, the user stories decomposition into development tasks, and the overall quality of the board. Ensure that each task is derived from a user story, has a priority, and is assigned to team members.

Emotion Skill Survey
The Emotion Skill Survey will be completed in the tutorial session on March 21st. If not completed in class, the survey [14] has to be filled out individually.

Brownie Points
In addition to the report assessment, we will use a "brownie points" system for which you have to distribute brownie points to your team members. The brownie points serve to reflect on how you feel about the extent to which the other team members contributed to your learning, the assignment, and your team's performance. The distribution will be an opportunity to reward the members of your team who worked hard on your behalf. You can split the brownie points equally if you think everyone did the same.

Every student has a total of 40 brownie points to distribute to the 4 other team members (if you have only 3 team members, only distribute 30 brownie points). These brownie points will also allow us to see any concerns in a team early on. The brownie points can decide whether individual group members pass or fail for borderline submissions.

[14] `https://uzhwwf.qualtrics.com/jfe/form/SV_d5xzUsNQQcvlgpg`