University of Zurich[UZH]

s.e.a.l. software evolution & architecture lab

Software Praktikum (SoPra) - FS21

Milestone 2 - Assignment

1 General Information

The Assignment has to be done as a team effort where each member makes continuous progress on the tasks and is able to answer content-related questions during the presentation for milestone 2. The hand-in for milestone 2 consists of a written report, a presentation, and the source code. The report and the presentation must be submitted via OLAT. The source code will be taken from the GitHub project's master branch. For all three deliverables, the deadline is **28.3.2021 23:59 CET**. Milestones 1 and 2 will be presented together on **29.3.2021**.

1.1 Report

The report must be submitted as a single PDF file by the **project leader via OLAT**. The title page consists of the group name, group leader and information about all group members (name, matriculation/student number). Please include the links to your GitHub Repositories, your SonarQube project and your running application (on Heroku). Make sure the report is easily readable in printed form (figures, tables, etc). Furthermore, ensure to use a consistent format (header, footer, font style, font size, page numbers, figure and table titles, etc.), page orientation (portrait) and page size (DIN A4). Use a file name of form **FS21-Group-GROUPNUMBER-M2-Report.pdf**. The report must be written in English.

1.2 Slides

The presentation must be submitted as a single PDF file by the **project leader via OLAT**. The title page consists of the group name and information about all group members (name, matriculation/student number). Use a file name of form **FS21-Group-GROUPNUMBER-M2-Slides.pdf**. The slides and the presentations have to be in English.

Advice: The presentation has to include results from milestones 1 (group phase) and 2. Each team member has to present at least on one occasion (M1+M2, M3, or M4). It is up to the team to decide how to distribute the presentation time over its members. Underline the name of the presenters on the title slide. Presentations will be 6 minutes per team and have a hard cut-off. After the presentation, there will be a 2-4 minutes Q&A session. Make sure to prepare your presentation and prepare slides that are readable by the audience and easy to understand (including

the diagrams). Also, ensure to practice your presentation at least once beforehand, especially to make sure you stay within the given time frame. Some instructors that have not seen the report will sit in the presentation and this will be your moment to shine.

Suggested Structure: We suggest focusing your presentation on the following items:

- Introduce and explain your project.
- Show interesting user stories (without acceptance criteria and time estimates) and connect them to your UI mockups.
- Present your class, component, and activity diagrams. It is okay to present a quick overview before focusing on 1-2 core parts of your diagrams. Again, you can relate them to user stories if it simplifies understanding and reasoning. **Advice**: There is limited time and limited screen resolution to show and explain your complete diagrams. Think of creative solutions to show interesting details and convey the general idea of your diagrams (e.g., use color, zoom-in-effects, overlays, etc.).

1.3 Source Code

The source code submission is done via GitHub. Add a git tag **M2** to the commit in the **master** branch that should be taken into consideration for grading. It is important to push and tag the commit before the deadline **28.3.2021 23:59 CET**. Moreover, make sure the code tagged with M2 contains a stable/running version of your system. If we can not run your system, we can not grade it and therefore will award 0 points. Submissions have to be completed before the deadline. Anything that is submitted later, will not be taken into account.

2 Assignment Description

In Milestone 1 (M1) you familiarised yourself with the infrastructure and formulated the user stories for the development phase. Based on that knowledge, feedback for the M1 report and the created artifacts, you will now have to refine the user stories and decide which stories to include in your first sprint. You are also ready to design the application that you are going to implement. For the design phase, you will create UML diagrams, the interface specification, and UI mockups. After you received feedback regarding the design documents from your TA, you are expected to start implementing the first prototype of the system in Milestone 3.

2.1 Diagrams

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

Component Diagram: Define the architecture of your application by means of component diagrams [1]. Component diagrams are used to visualize the organization and relationships among components in a system. Define the major components of your system and how they interact. Specify interconnections among components by means of 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 *Pictures*, “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

[1] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/

of what is going to be persisted). Notice that you do not have to model the auxiliary classes used in the SpringBoot ecosystem (e.g. repository, controllers or services).

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 *Pictures* that could be either updating the scores or guessing a particular picture representation of a co-player). An activity diagram is used to represent the flow from one activity to another. Its basic 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 (web page) and the server a REST[2] 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. Developers of the client and server can rely on that specification and work independently on their tasks.

An example of such a specification is presented below (from Assignment 1).

Mapping	Method	Parameter	Paremeter Type	Status Code	Returned Value	Description
/users	POST	username <string>, password <string>	Body	201	Location: url<string>	add User
/users	POST	username <string>, password <string>	Body	409	Error: reason<string>	add User failed because username already exists
/users/{userId}	GET	userId<long>	Query	200	User: id<long>, username<string>, creation_date<date>, ...	retrieve user profile with *userId*
/users/{userId}	GET	userId<long>	Query	404	Error: reason<string>	user with *userId* was not found
...	...	...	...	...	...	...

Advice: Use the operations (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 as described here[3] or 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. in case of a successful login, after the login page a lobby screen is shown).

Advice: You can create the mockups with a dedicated tool if you wish. Examples for such tools 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 use the tool collaboratively.

[2] https://en.wikipedia.org/wiki/Representational_state_transfer
[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/ch_it/products/xd.html
[8] http://balsamiq.com/products/mockups/

Let your TA review your mockups before you hand in the report. Together with activity diagrams, mockups are very important to model the behavior of your application.

2.4 Setting Up Development Infrastructure

This part is about configuring the development infrastructure for your team. We will use the same setup you are already familiar with from Assignment 1 (individual phase). Only one person of your team needs to perform the following task. It is fulfilled if you have one commit by a team member in the team's GitHub repository that changes the user interface in a way that your group name/number is displayed. You are not required to give a demo about the running template. However, be sure that you have checked-in the change (including the correct tag) before the deadline **28.3.2021 23:59 CET**.

In the following, you find detailed instructions regarding the GitHub, Heroku and SonarQube setup.

GitHub Setup
We will use the server and client templates you are already familiar with from Assignment 1.

1. Get the server[9] and client[10] templates from the GitHub repositories for this course. You can do this by, for example, using the "git clone" command in a terminal shell.
2. If you haven't done so already in Assignment 1, generate an ssh key for your local machine [11] and add the *id_rsa.pub* key to your GitHub Account[12].
3. Create a new GitHub organization[13]. Name the organization after your group. Use the following naming convention: **sopra-fs21-group-XX** Invite your team members, the GitHub user **sopra-staff**, and your TA to the organization (as "Owner").
4. In the new organization, create two *public* repositories[14], one for the client and one for the server. Unlike in Assignment 1, your repositories need to be public to use SonarQube.
5. Push the server and client templates stored on your local machine to your own repositories on GitHub using the commands in Listing 1 for each of the two, server and client (please note that it is required to execute these commands while being in the root folder of the templates on your local machine).

```
$ cd [local_path_to_template_repo]
$ git remote rm origin # remove existing origin
$ # add your private repository as the new origin
$ git remote add origin git@github.com:[account_name]/[repository_name].git
$ git remote -v # verify the new origin
$ git push -u origin master
```

Listing 1: Github Set-Up

[9] https://github.com/HASE-UZH/sopra-fs21-template-server
[10] https://github.com/HASE-UZH/sopra-fs21-template-client
[11] https://help.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent
[12] https://help.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account
[13] https://help.github.com/en/github/setting-up-and-managing-organizations-and-teams/creating-a-new-organization-from-scratch
[14] https://help.github.com/en/github/getting-started-with-github/create-a-repo

Heroku Setup
Notice that the steps which involve setting up Heroku are identical to the steps from the individual part of Assignment 1. You have to perform the following steps only once per group.

1. Create a free account on `https://www.heroku.com` (one per group).
2. If you have not done so in Assignment 1, install the Heroku CLI tools following this article[15].
3. Set up Heroku by running the commands from Listing 2 in the terminal of your OS. The commands will authenticate your machine to Heroku. **Note:** By now, you should already have an ssh key for your GitHub account. We'll reuse it for Heroku.
4. Set up client and server apps on Heroku by running the commands from Listings 3 and 4 from the command line.
5. For both client and server repositories, go to the *Settings* tab on GitHub, open the *Secrets* tab from the menu on the left and add 3 secrets: `HEROKU_API_KEY`, `HEROKU_APP_NAME` and `HEROKU_EMAIL`. The app names should be equal to the ones you picked in Listings 3 and 4. You can find the Heroku API Key in the Heroku Dashboard when clicking on the avatar (top right) under *Account Settings*.
6. At this point, when you push code to your master branch on GitHub, it will automatically deploy to Heroku. The GitHub action which pushes your code to Heroku is configured in `.github/workflows/deploy.yml`. It should work out-of-the-box. However, if you experience problems you can monitor the deployment in the *Actions* tab on GitHub by selecting the (failed) workflow and then choose the *run* on the left (e.g., build). Now you see the log. On the right, you can rerun all checks. With the default Heroku plan, concurrent builds are not possible. So, if you trigger a build of both the front-end and server (back-end) at the same time, only one will be built and the other will fail.

```
$ cd # change to home directory
$ # only execute ssh-keygen if you don't have a id_rsa key already (for GitHub)
$ # ssh-keygen -t rsa -C "your_email@example.com"
$ heroku login # individual credentials from registration
$ heroku keys:add ~/id_rsa.pub # or heroku keys:add ~/.ssh/id_rsa.pub
```

Listing 2: Heroku Authentication

```
$ cd [path_to_server_repo]
$ heroku create --ssh-git sopra-fs21-group-XX-server --region eu
$ git remote -v # verify the new remote 'heroku'
```

Listing 3: Heroku Setup Server

```
$ cd [path_to_client_repo]
$ heroku create --ssh-git sopra-fs21-group-XX-client --region
eu --buildpack mars/create-react-app
$ git remote -v # verify the new remote 'heroku'
```

Listing 4: Heroku Setup Client

You and your team members should now be able to deploy by simply pushing code to the master branch of the remote GitHub repository. Your application will only be deployed to Heroku if

[15] `https://devcenter.heroku.com/articles/heroku-cli`

the build succeeds (can be monitored in the *Actions* tab of your repository on GitHub).

SonarQube Setup for Code Quality
Additionally to the basic setup, we will use SonarQube to measure the quality of our code. To configure SonarQube you need to follow the steps below (once per group):

1. You'll be using the free cloud services from SonarQube for Open Source projects. Create an account on `sonarcloud.io` by clicking the GitHub icon (i.e. "analyzing your GitHub repo").
2. Follow the installation guide provided by SonarCloud and set up both the server and client projects. In both 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 (master) your SonarCloud dashboard should show you a quality assessment of your code.

2.5 Scrum Setup on GitHub

After having established user stories in Milestone 1 it is now time to refine and transfer them to GitHub. We use *Issues* ([16]) to manage them during Milestones 3 and 4 and as part of the Scrum methodology.

1. On GitHub and in the *Issues* tab, create a *Label* named "user story". You can assign it to every user story you create.
2. Enter your user stories on GitHub by creating a new Issue per user story. Use the title field for your user story in the role-goal-benefit format. Please enter the user stories in both the client and the server repositories.
3. In the description of the user story, use *Markdown* [17] 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.

SoPra follows the Scrum methodology [18]. In Scrum, the software development is organized in development units called "sprints". Each sprint (or iteration) 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. During a sprint, you work on "development tasks" which are part of a decomposed "user story". A development task is a smaller, more manageable entity to implement and should be assigned to a single team member.

1. On GitHub, create a *Label* named "task". Assign this label to all future development tasks.
2. Further, create labels for prioritization (i.e. "high priority", "medium priority" and "low priority"). Assign a priority to each future development task.
3. Decompose your user stories and create an *Issue* for every development task. Select a meaningful and representative title.
4. In the description, write a time estimate (i.e., number of hours/days required to accomplish the task). As a rule of thumb, each development task should not take longer than a day (it's better if they are a bit shorter).

[16] `https://guides.github.com/features/issues`
[17] `https://guides.github.com/features/mastering-markdown/`
[18] `https://en.wikipedia.org/wiki/Scrum_(software_development)`

5. Besides the time estimate, add a reference to the user story the development task originated from. To do so, you can simply type the hashtag character followed by the issue number (e.g. "This task is part of user story #14").

With concrete development tasks and a rough estimate of the effort required to complete them, start planning your first sprint. Pick development tasks from your "product backlog" and add them to your "sprint backlog". To do so, use the GitHub *Milestones* feature, set its end date to the Milestone 3 deadline, and assign the milestone to the tasks you want work on during the sprint.

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 to do the majority of implementation work during Milestone 3 (approximately two thirds of all development tasks).

3 Grading

As SoPra is a pass/fail course, the grade for M2 will be pass or 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.

In M2, you will be evaluated in respect to the content of your report as well as your ability to present it. We will assess the completeness of your sprint planning both in terms of the covered development effort (i.e., 70% of the user stories) and the description of the tasks involved in the sprint. Furthermore, we will evaluate the quality and completeness of your modeling diagrams (i.e., component, class, and activity diagrams) as well as the clarity of your mockups. With regards to the REST interface, we will evaluate the quality and completeness of the proposed APIs. Finally, we will positively evaluate a correct set-up of the development infrastructure. You will receive feedback on your report and presentation including an assessment (either pass, borderline pass, or fail).

Brownie Points

In addition to the report assessment, we will use a “brownie point” 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 members of your team contributed to your learning, the assignment, and your team’s performance. This will be an opportunity to reward the members of your team who worked hard on your behalf. If you think everyone did the same, then you just split the brownie points equally.

Every student has a total of 40 brownie points to distribute on the 4 other team members (if you have 5 team members, you can distribute 50 brownie points). These brownie points will also allow us to see early on if there are any concerns in a team. For borderline submissions, the brownie points can decide whether individual group members pass or fail. Please submit your brownie points via the OLAT form (together with the individual and/or group submission) by **28.3.2021 23:59 CET**.