Software Praktikum (SoPra) - FS21

Milestone 1 - Assignment

1 General Information

Assignment 1 consists of two parts:

(1) An **individual part** in which each student has to individually familiarize themselves with the frameworks and languages used for the course and implement three user stories individually. **NOTE:** The individual part **must be passed successfully** to pass the course.

(2) A **group part** for which students have to form groups of 5 (make sure to join a group on OLAT **before Thursday 25.02.2021 23:59 CET)**, decide on the project/application they want to develop, and come up with the requirements in the form of user stories.

The hand-in and evaluation for Milestone 1 consists of two parts:

Individual Part: Submit one ZIP file *per student* to OLAT, containing the source code implemented for the individual part, including the client and server code as well as the tests implemented for the REST interface. Include a text file called "links.txt" containing the links to the deployed applications on Heroku. Make sure to include descriptive comments in your code (in English). For the ZIP file use a file name of form **FS21-LASTNAME-FIRSTNAME-M1.zip**. The individual part will be **evaluated** in in-person meetings in which each student presents their solution and answers questions asked by the course staff.

Group Part: One PDF report *per team* written in English and submitted by one team member via OLAT. The report must start with a title/cover page that lists the group name and information of each group member (name, UZH email, matriculation/student number). Content-wise, the report has to contain the project title, a short description (150 words) of your project/application and the elicited user stories for the development phase of your project. Make sure that 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-M1-Report.pdf**. The group assignment has to be completed as a team and each member has to be able to answer content-related questions to every part of the report.

Both deliverables must be submitted to OLAT by **Sunday, 14.03.2021 23:59 CET** latest. Milestone 1 (individual phase) has to be presented on **Monday, 15.03.2021**. Milestones 1 (group phase) and 2 have to be presented together on **Monday, 29.03.2021**.

2 Assignment Description

The first assignment is about familiarizing yourself with (a) the development environment, frameworks, languages and deployment platforms used for this course project and (b) to determine the requirements for your group project.

2.1 Individual Phase

In the individual phase, every student is expected to familiarize themselves with the templates and frameworks used. We will use the React framework for the front-end, Spring Boot and Java 15 for the back-end, REST as the interface between front-end and back-end, and JPA/Hibernate for persistence. For getting your application up and running, you are supposed to deploy your front- and back-end to separate instances on Heroku[1]. Do not be discouraged if you are unfamiliar with many of the terms/frameworks, this assignment is designed to familiarize yourself with them. **Note:** each student has to perform this part of the assignment on their own; this will also ensure that everyone will be able to contribute in the group development phase!

Setting Up Repositories Locally and on Github

(1) Create an account on Github[2] if you do not have one yet.
(2) Generate an ssh key for your local machine [3] and add the *id_rsa.pub* key to your GitHub Account[4]. This step will simplify authentication with GitHub.
(3) Get the server[5] and client[6] templates from the GitHub repositories for this course. You can do this by, for example, using the "git clone" command in a terminal shell.
(4) Create two *private* repositories[7], one for the client and one for the server. For this task, private repositories must be used.
(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

[1]https://www.heroku.com
[2]https://github.com/
[3]https://help.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent
[4]https://help.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account
[5]https://github.com/HASE-UZH/sopra-fs21-template-server
[6]https://github.com/HASE-UZH/sopra-fs21-template-client
[7]https://help.github.com/en/github/getting-started-with-github/create-a-repo

Setting up Heroku Deployment

We'll use Github Actions[8] to automatically deploy our application to Heroku, a platform-as-a-service (PaaS) provider. Follow the steps below to get started:

1. Create a free account on `https://www.heroku.com` (one per student).
2. Install the Heroku CLI tools following this article[9].
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.
7. Once the server is deployed successfully make sure to copy the URL of the server application from the Heroku dashboard, add it to `src/helpers/getDomain.js` in the client repository, and re(-deploy) the client. If the server URL is incorrect or missing you might see an alert "The server cannot be reached. Did you start it?" when trying to login.

```
$ 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 Set-Up

```
$ cd [path_to_server_repo]
$ # if your full name is too long for heroku use the uzh shortname instead
$ heroku create --ssh-git sopra-fs21-lastname-firstname-server --region eu
$ git remote -v # verify the new remote 'heroku'
```

Listing 3: Heroku Set-Up Server

```
$ cd [path_to_client_repo]
$ # if your full name is too long for heroku use the uzh shortname instead
```

[8] `https://github.com/features/actions`
[9] `https://devcenter.heroku.com/articles/heroku-cli`

```
$ heroku create --ssh-git sopra-fs21-lastname-firstname-client --region
eu --buildpack mars/create-react-app
$ git remote -v # verify the new remote 'heroku'
```

Listing 4: Heroku Set-Up Client

Individual Implementation of User Stories

To complete the individual part of the assignment, each student has to implement the client and server parts for the *three* user stories listed below and according to the REST specification of Table 2.1. The REST API specifies the communication interface between the client and the server. Please consider that there is already a built-in login function in the provided templates that lets you automatically register a new account. You are expected to modify this function when working on the three user stories. For the login and registration, a good approach could be to create one screen for the login and another one for the registration of new users. The login functionality must allow only registered users to login.

ID: S1 Category: User Management

Story: As an unregistered user, I want to be able to register as a user with my chosen credentials (i.e., username and password that are both not empty words) to leverage/use services and information that are exclusively available to registered users.

Acceptance Criteria:

- Upon successful user registration, the users overview screen (see below) is shown and the user is automatically logged in.
- Upon failure, an error is displayed and the user is redirected (back) to the register screen. A register error can be that a user name is already taken.
- The creation date of a user is saved to the database record.
- Logged-in users can log out and log back in to their registered profile.

Priority: critical
Author: SoPra Assistants
Estimate: 4h

ID: S2 Category: User Management

Story: As a logged-in user, I want to inspect the profile of a registered user by selecting the username in a list of all registered users (users overview).

Acceptance Criteria:

- User can view a list of all registered users (users overview) and select each one for inspection.
- By clicking on a username in the users overview, you are redirected to a profile page of the selected user.
- The profile page contains the following data belonging to the selected user: username, online status, creation date, and birth date.
- The users overview and the profile page is only accessible for logged-in users.
- The birth date is optional and can only be set on the profile page.

Priority: critical
Author: SoPra Assistants
Estimate: 8h

ID: S3 Category: User Management

Story: As a logged-in user, I want to edit my own user profile to set/update my personal information.

Acceptance Criteria:

- By clicking on an edit button in the user-profile view, you are able to change the user's username and birthday (**attention:** make sure to use a separate ID in the database for each user, so that references to a user are not lost after a username change).
- A registered user can only change their own profile and not profiles of other users.
- After changing and saving the data, the user is redirected to the profile page and the new/changed data is displayed.

Priority: critical
Author: SoPra Assistants
Estimate: 8h

Testing the REST interface (individually)

In order to design and reason about the REST endpoints that you need to implement for the user stories, you are also expected to write unit tests. These unit tests are part of the back-end project and have to be implemented using JUnit. The unit tests have to examine the endpoints by passing required data to the endpoint and validating/checking the returned result from the endpoint. Please make sure that:

- you handle data passing properly (i.e. is the data passed as query parameters or as part of the HTTP body)
- the correct HTTP method is used (GET, POST, PUT, or DELETE)
- the response sends the resulting data (if necessary) in the correct format
- the correct HTTP status code is sent (for success or failure)
- the correct HTTP header fields are set (e.g., Accept, Content-Type)

Validity of your request and response must be guaranteed by assertions (`org.junit.Assert.assert*`).

For the three user stories, you are expected to comply with the following REST specification:

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>, logged_in<boolean>, birthday<date>	retrieve user profile with *userId*
/users/{userId}	GET	userId<long>	Query	404	Error: reason<string>	user with *userId* was not found
/users/{userId}	PUT	User	Body	204	-	update user profile
/users/{userId}	PUT	User	Body	404	Error: reason<string>	user with *userId* was not found

Remark: During the individual assessment you have to demonstrate the tests and their successful passing to the examiner. The REST specification does not handle the case, if the user is not registered. For the implementation, you have to come up with a correct response if an unauthenticated requests wants to access `/users/userId`.

Advice: The `User` data passed to update a user profile does not have to contain all fields saved for a user (as by the definition of HTTP put). Only include the data fields necessary to identify the user to be updated and the fields that should be updated.

Assessment

The assessment of the individual phase is done in individual meetings with one or more of the SoPra staff. Please prepare a short (~5 minutes) demo of your running application (make sure you have deployed your front-end and back-end to Heroku, and you have accessed the application a few minutes before your slot, in order for the instances to be running) and open your source code and tests in your editor of choice. The staff will assess the functionality and completeness of the submission, and will ask you questions related to the assignment.

Please make sure to delete the `node_modules` folder of the client repository before compressing to zip. Be sure that you have submitted the zip file containing your source code, tests, and links to the deployed applications to OLAT by **Sunday, 14.03.2021 23:59 CET**. The presentations will take place on **Monday, 15.03.2021**. We will announce the exact times for the assessment of the individual phase separately via OLAT.

2.2 Group Phase

The main goal of the group phase of milestone 1 is to form groups of 5 students each, come up and decide on your group project, and identify/elicit the requirements in form of user stories. The user stories created by you have to be prioritized for M1, which will help in assigning the individual stories to the iterations/sprints later on.

Each group has the choice between (a) your own project idea, or (b) an application to play the game of the year 2020 called *Pictures*. The underlying structure for both projects will be very similar in terms of the architecture, the frameworks used and other aspects. However, when you choose option (a), you will have the opportunity to be more creative in terms of the application which might also lead to more fun and you get a chance to work on something that is interesting and exciting to you.

For fairness reasons and to ensure that the scope of the projects for this course is similar, your project and application has to fulfill a few requirements, regardless of whether you are working on your own idea or on the *Pictures* game your application has to...

- use the same technology stack (React, Java, Spring Boot, GitHub (Actions), Heroku, JPA). If you want/need to deviate from this for justified reasons, please talk to us as soon as possible.
- have a client-server architecture with a web front-end. It cannot just be a command line interface.
- interact with a server with a REST API that you created.
- have some persistence layer, i.e. you need to store something in a database, such as user data and more.
- feature collaboration capabilities where different user profiles work towards a shared goal (e.g. post and collaboratively edit a document or a social network, or play a game in real time).
- perform some useful function and cannot just be a database management app (e.g. simple CRUD apps that do not make sense).
- work with a small user base. It cannot require crowd buy-in to be useful (e.g. apps that would require large numbers of people to contribute content to be viably useful).
- consume at least one external API (e.g. a translation or computer vision service).

Option (a) — Your Own Idea
Develop an application based on your own idea, that is interesting and that excites your team.

To ensure for a similar scope across all proposed applications, discuss your idea **beforehand** with your TAs and think about a 'simple' and an 'extensive' implementation of your idea, so that one can easily adjust (think of it as 'optional' user stories that one could easily add if needed). Your TA *has to* approve your idea, for you to be able to pursue it as this course's project. The deadline for getting approval is Sunday, **07.03.2021 23.59 CET**. Please reach out to the SoPra team via sopra@lists.ifi.uzh.ch with a short (150 words) description of your project idea and name.

Option (b) — *Pictures* Game
If you choose to develop the game of the year 2020, make sure to familiarize yourself with the rules. The game manual can be found here [10] or on OLAT. Use the game instructions to determine the user stories. Try and find interactive and fun digital adaptations to the 5 (physical) sets (e.g.

[10] `https://www.pd-verlag.de/WebRoot/Store/Shops/es719366/MediaGallery/Spielregeln/Pictures_Rules_EN_web_1.1.pdf`

stick and stones, the shoe laces, ...). You can find an image of all 19 icon cards (set 4) on OLAT. Incorporate the digital adaptions into the user story.

Additional to the basic game rules of *Pictures*, we ask you to invent a creative feature (or game modification). It should fulfill the following criteria:

- The feature should share data between game rounds.
- Use the data collected in one game round to modify or augment subsequent game rounds.

For example, your modification could alter the game rules for (individual) players based on their previous performances. Another example would be to modify the behavior of the 5 sets based on historical data.

We expect you to come up with a unique and fun idea. Please make sure to create appropriate **user stories** for it.

User Stories

Determine and specify the requirements for your application in the form of user stories. Each user story should be in the role-goal-benefit format "As a <Role>, I want to <Goal> in order to <Benefit>", with an acceptance criteria that has a number of tests that determine if the use case described by the user story is satisfied. Further, add a rough time estimate to each user story and prioritize it. You can also add a 2-3 sentence description if you want to.

There is no minimum or maximum number of user stories you must have. However, the user stories need to cover all functionality of the user(s) of the app. It would be surprising if it ended up being less than 10. At the same time, you should not have way too many user stories. For the past years, a number of around 15 user stories were often sufficient to describe the game's main requirements. Make sure the set of user stories is sufficient and complete, even if not all of them will be implemented in later stages of this course.

3 Grading

SoPra is a pass/fail course. To pass the course you, as an individual, have to pass the individual part of milestone M1, and as a group, you have to pass 3 out of 4 milestones, but you definitely have to pass milestones M1 and M4. You need to hand-in reasonable reports for all the milestones.

So, in order to pass M1 (and in the end the course), you have to pass *both* the individual and the group phase of M1. For the individual phase, the one-to-one assessments decide about pass/fail, whereas the group phase is assessed based on the completeness of your report. You will receive feedback on your report 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 only 3 team members, only distribute 30 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 **Sunday, 14.03.2021 23:59 CET**.