Project Instructions

Introduction

These instructions will guide you on your quest to create a simple website where a user can play a single player dice game. The game consists of four dice, and the user should guess the sum of the first three dice before throwing them all. If the user's guess turned out to be lower than or equal to the sum of the first three dice, his score should be incremented by the guessed number multiplied by the fourth dice. This should be repeated 10 times, after which the game is over.

To implement the game, you will use:

- HTML to create the graphical components the browser should render.
- CSS to tell the browser how to render the graphical components.
- JavaScript to implement the game logic.
- JavaScript and DOM to catch the user's interaction with the graphical components.
- JavaScript and DOM to update the graphical components to reflect the current state of the game.

Your website should also be connected to a server provided by us. Through it, users should be able to:

- Create a new account.
- Sign in to an existing account.
- Sign out from an account (requires the user to be signed in).
- Add a high score to an account (requires the user to be signed in; the game should be playable for anonymous users).
- Get the top 10 high scores across all accounts (requires the user to be signed in).
- Get all high scores for a specific account (the one the user is signed in to; requires the user to be signed in).

To communicate with the server, you will use:

- JavaScript and AJAX (the XHR object and JSONP) to send and receive HTTP requests/responses.

Figure 1, Figure 2 and Figure 3 on the next page show the three primary use cases for the user.

The project has been divided into four different parts, and you are expected to complete ~one part each week.

You are strongly encouraged to read through this entire document before you start working on the project.

Good luck!

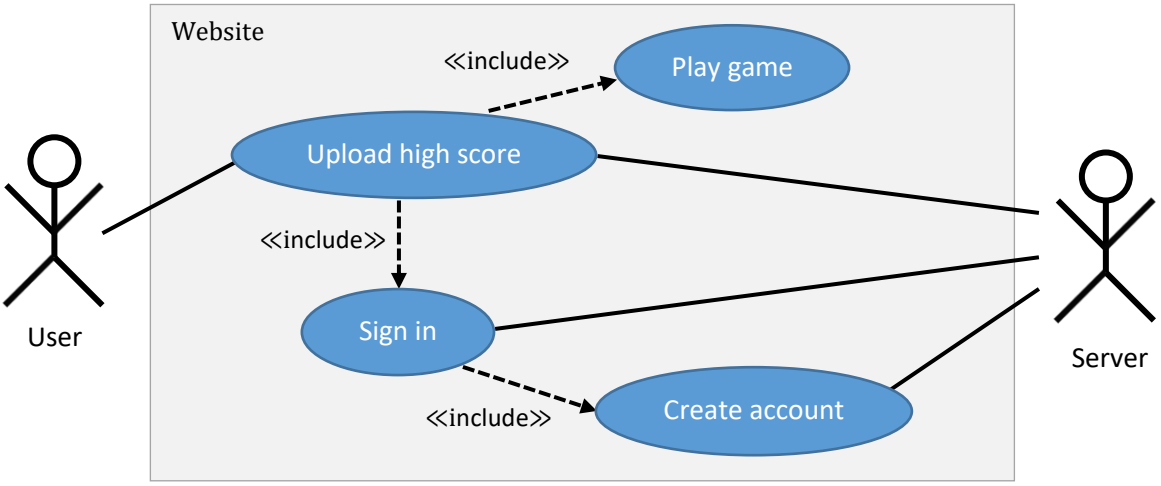

Figure 1, UML use case diagram for uploading new high score.

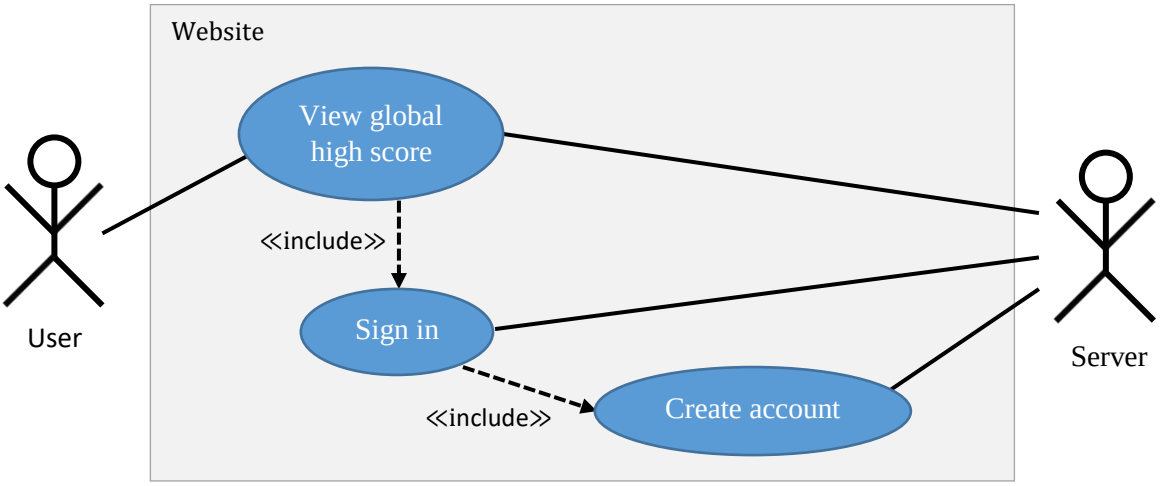

Figure 2, UML use case diagram for viewing global high score.

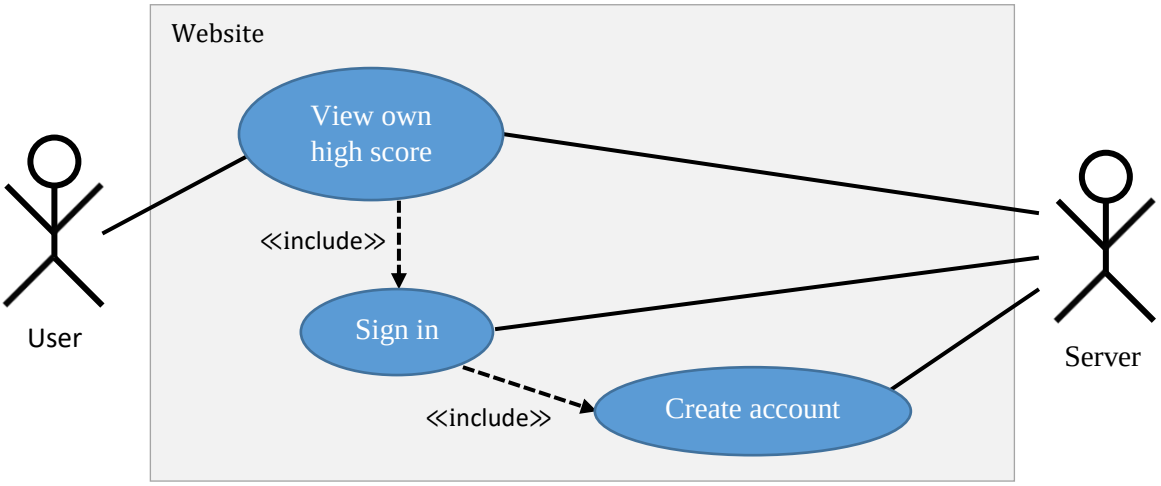

Figure 3, UML use case diagram for viewing own high score.

Part 1: Creating the graphical user interface

The goal with this part of the project is to get familiar with HTML and CSS by creating the graphical user interface for the website.

Although not required, you are strongly encouraged to start by drawing a sketch of what you want your website to look like in the end. Use pen and paper to draw the sketch, and then try to write the HTML and CSS code the browser needs to render your website that way. **You will not get any help with your graphical user interface from any lab assistant unless you show them what you want it to look like on a piece of paper.**

Your website needs to have at least the following graphical components:

- A form for users to sign up (first name, last name, email, username and password).
- A form for users to sign in (email and password).
- A graphical component for signed in users to sign out (e.g. a button).
- A table/list for the top ten high scores (at the moment, use dummy values).
- A table/list for the signed in user's own high scores (now, use dummy values).
- Graphical components for playing the game.

You can put all these graphical components in one and the same HTML file. When you later become familiar with JavaScript and DOM, you can hide/show these components dynamically, e.g. only showing the sign out component when the used is signed in. Also, feel free to simulate different "pages" this way one can navigate between using a menu (e.g. one "page" showing the sign in component, another showing the sign up component, etc.).

It is possible to spread out the components in different HTML files (e.g. to have each HTML file acting as a page), but this will make it harder for you later, when you need to communicate with the server.

In addition to writing your own CSS code, you also need to use an HTML & CSS framework, e.g. bootstrap. It is up to you to decide how you want to make use of the framework, but here are two different ways presented:

- Use bootstrap for the main layout, and style buttons/inputs/texts using your own CSS code.
- Use your own CSS code for the main layout, and use bootstrap to style buttons/inputs/texts.

When you are finish, we strongly recommend you to read through the file `project-grading-guidelines.pdf` on Ping Pong to check if you have done everything expected for the grade you are aiming for.

Part 2: Implementing the game

The goal with this part of the project is to get familiar with JavaScript and the DOM by implementing the game logic and make it playable through the graphical user interface.

Implementing a fully functional game with a graphical user interface is hard. To make it a bit easier, we recommend you to start by ignoring the graphical user interface, and only focus on the game logic. A game perfectly designed should be playable with no graphical user interface at all (which is the case for artificial intelligent players and games played over networks with a hosting server). This essentially simplifies things, since the graphical user interface usually is the hardest part to deal with.

Try to create functions through which one can play the game in the console in a web browser by entering something like the statements below (one after another):

```
var game = createDiceGame(/* Maybe some arguments here? */)
processGuess(game, 10) // Play first round, my guess is 10.
processGuess(game, 13) // Play next round, my guess is 13.
// ...
var myScore = getScore(game) // Get my current score.
```

Use your knowledge from the *Object-Oriented Software Development* course to design your game logic. Although we do not have classes in JavaScript, we can still model the game in a similar way using objects.

Note: JavaScript contains support for constructors and prototypal inheritance, which we will talk more about at one of the last lectures. When you have learned that, feel free to come back here and rewrite your code to make use of these features.

When you have got your game logic to work, all that remains it to make it controllable from your graphical user interface. Connecting the graphical user interface with the game logic is actually not that hard. "*When the user clicks on the graphical component X, call the game logic function Y*". After that you will also need to update the graphical user interface to reflect the new state of the game.

When you are done with this part of the project, the game should be fully playable through the graphical user interface.

Note: If you structure your code well, one should quite easily be able to reuse the game logic on other websites. If you have the time, feel free to exchange your game logic with a friend and see what changes you need to apply to your own code in order to make it work with your friend's game logic.

When you are finish, we strongly recommend you to read through the file `project-grading-guidelines.pdf` on Ping Pong to check if you have done everything expected for the grade you are aiming for.

Part 3: Validating user input

The goal with this part of the project is to get further familiar with JavaScript and the DOM by validating the input from the user.

When the user signs up or in, she enters some input. This input will later (the last part of the project) be sent to a server. Sending/receiving data to/from a server takes time, and devices with a mobile internet connection might only be allowed to send/receive a limited amount of data. Therefore, communication with servers should be kept to a minimum.

One way to minimize this is by validating as much as possible of the input directly on the client. If we on the client can confirm that something is wrong, it does not make sense to send it to the server (which will complain about the data).

The sign up fields should be validated according to the following rules:

- The user's first name: not empty, and only letters (a-Z).
- The user's last name: not empty, and only letters (a-Z).
- The user's e-mail: a valid e-mail address (feel free to search for a regular expression online).
- The user's username: not empty, and only letters (a-Z), digits (0-9) and underscore (_).
- The user's password: at least 3 characters.[1]

The sign in fields should of course be validated using the same rules for the email and password.

Also, to improve the user experience, the user should not be able to play the game if an invalid guess has been entered (a number not between 3 and 18).

Note: If you structure your code well, one should quite easily be able to reuse the validation logic on other websites. If you have the time, feel free to exchange your validation code with a friend and see what changes you need to apply to your own code in order to make it work with your friend's validation code.

Note: In this document, we never instruct you to use a JavaScript library, but to pass the project work, you need to "*Make use of a JavaScript library*". Starting to use a JavaScript library, e.g. jQuery, in this part of the project might be a good idea.

Note: Using HTML5 validation attributes is a great solution (works even if the user has disabled JavaScript), but in this project, you need to validate the input using JavaScript.

When you are finish, we strongly recommend you to read through the file `project-grading-guidelines.pdf` on Ping Pong to check if you have done everything expected for the grade you are aiming for.

[1] A strong password should nowadays be at least ~10 characters long (depending on which characters it consists of), but requiring that will make it a pain to test the website, so we settle with 3 characters. Changing this should only involve changing one number at one place.

Part 4: Communicating with the server

The goal with this part of the project is to get familiar with AJAX using the XHR object in JavaScript and the JSONP workaround to communicate with the server.

It is now time to connect your website to our server. After completing this part of the project, you should be finish with your project work, and your website's visitors should be able to:

- Sign up.
- Sign in.
- Sign out.
- View top 10 high scores.
- View their own high scores (when signed in).
- Play the game.
- Have their score automatically sent to the server when the game is over (when signed in).

You can read more about the server's REST API you can use to communicate with it in the file `project-server-api.pdf` on Ping Pong. There, you will also find instructions for how to upload your files to our server (which you must do in order to pass).

Note: If you structure your code well, one should quite easily be able to reuse your code for server communication on other websites. If you have the time, feel free to exchange your communication code with a friend and see what changes you need to apply to your own code in order to make it work with your friend's communication code.

When you are finish, we strongly recommend you to read through the file `project-grading-guidelines.pdf` on Ping Pong to check if you have done everything expected for the grade you are aiming for.

If you have structured your code well, you should have ended up with a design looking something like the one shown in Figure 4 below. The JavaScript code found in *Server Logic*, *Game Logic* and *Validation Logic* should in the best case be independent of *GUI* and *Main Code* (100% reusable in other projects).

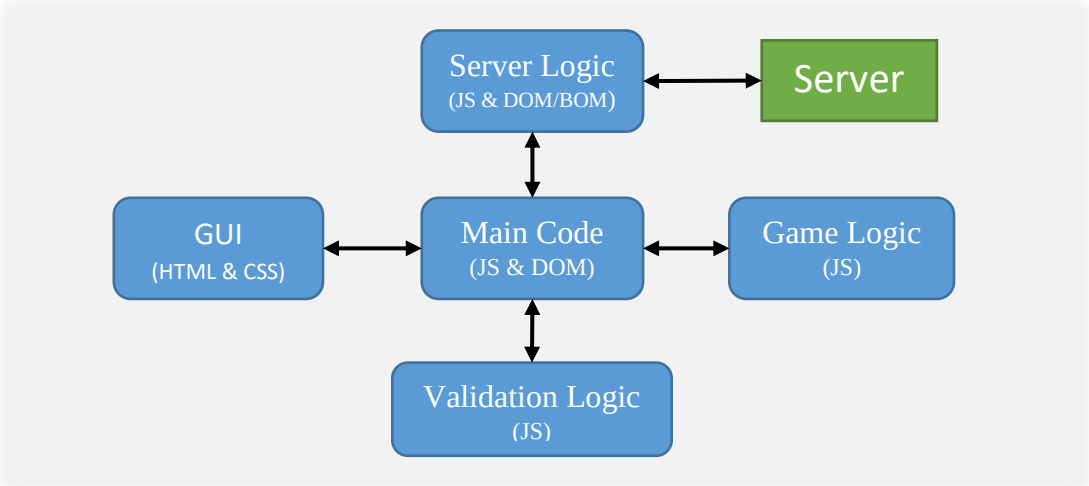

Figure 4, Overall design.

Submitting your work

When you have completed your project work you must submit the *Project Work* assignment on Ping Pong and attach all your project files in a ZIP file. You must also add the full URL to your HTML file on our server as a comment.

Do not submit your work by email; those emails will be ignored.