CO19-320212 Software Engineering
Spring 2019

Sprint 4 Write-up

Technical details

Released

Tuesday, 2.04.2019 at 21:30

Deadline

Wednesday, 24.04.2019 at 21:00

Duration

22 days 23 hours and 30 mins

Objective

This sprint is intended to let you catch up on what you are missing behind. You are should spend some time improving the code base you received. If your code base is missing implementation for objectives from the previous sprints, you will gain credits for doing so.

The objectives for this sprint are as follows:

- Build on the code base you received
- Implement functionalities which require you to work with an API

Deliverable

- Directory containing all the source code required for building the program
- Documentation for the project as specified in the Project Write-up
- README.md file describing all the improvements you made to the code base

Implementation specification for Sprint 4 objective

Welcome screen

Create a welcome screen for your program. Your welcome screen should have the following UI components:

- Start game button
    - Clicking this button will lead you to the gameplay screen (i.e. view you've worked on for the past 2 sprints)
- Load game button
    - Loads a .sav file from the save feature you implemented in Sprint 3
- Highscore board
    - Displays the top 10 scoring players in the class from data on a server
    - Detailed specification for the implementation is included below

Gameplay screen

To make the whole saving and highscoring system work well, your gameplay screen might need to be adjusted. Please ensure that your gameplay screen have the following UI component:

- Save game button
    - Clicking this button would trigger these actions:
        - Create a .sav file
        - Upload your score (the number of pencils produced so far) to the server
- Quit game button
    - Clicking this button would take you back to the welcome screen

Specification for highscore board

In this sprint, we will work with our server API to retrieve and upload data onto a server which stores the highscores. You will be formatting a lot of JSON strings during this sprint, so it is advisable to find tutorials online on how the JSON format works. Please read through this document very careful to prevent unneccessary penalization.

Where the scores are stored

The scores are stored in a database on an AWS server. This server is maintained by your TAs. You do not need to worry about it.

For maintenance reasons, the server does not have a permanent address. The address to the current server can be obtained by performing a HTTP GET request to this following address:

https://matiusulung.com/se.json

The page will return a JSON string with the following format:

{“link”: ”some-url.com”}

The address to the server will be provided as the value in the “link” field. For the purpose of this write-up, we consider this address to be some-url.com. Please make sure your program obtains the correct URL everytime it starts running.

NOTE: The only URL you may hardcode in your code is https://matiusulung.com/se.html. You must not hardcode any other complete URL in your code. You will be penalized for this.

How to get the highscore

To get the top 10 highscore, simply perform a HTTP GET request to the server url and appending “/getscores” to the address (e.g. http://some-url.com/getscores). You will receive a JSON string in the following format:

```
{
	"highscores": [{
		"game-username": "pencilKing",
		"score": "1000",
	}, {
		"game-username": "pencilQueen",
		"score": "900",
	}, {
		"game-username": "pencilJack",
		"score": "100",
	},

	. . .

	{
		"game-username": "pencilAce",
		"score": "300",
	}]
}
```

It is then your task to turn the JSON string into a presentable format in your highscore board. The data you receive are the top 10 high scores but is not necessarily sorted. You must sort the scores in your program.

How to upload your score

To upload your score, you perform a POST request following the format below to the server url appended with “/uploadscore” (e.g. http://some-url.com/uploadscore). Your score should be uploaded every time the game is save. The content of your POST requst must be formatted as following:

- jacobs-id
    - If your email is f.mcfreshface@jacobs-university.de
    - Then your jacobs-id is f.mcfreshface
- se-token
    - This is the token given by your TA via email
- game-username
    - This is the name that will be stored in the server. It can be whatever you want, but let’s keep it tasteful!
    - This is limited to 15 characters
    - May only include alphanumeric values (a – Z and 0 – 9)
- score
    - This is the total number of pencils produced when you save the game
    - The highest score the server will accept (for now) is 25.000.000

Format the fields above into a JSON parseable format. It should look similar to this:

```
{

    "jacobs-id": "f.mcfreshface",

    "se-token": "Pa5Wu7",

    "game-username": "pencilKing",

    "score": "1200"

}
```

If your POST request is valid — i.e. correctly formatted, token is correct, and your input is valid, you will receive a JSON response like this:

```
{"status": "SUCCESS”}
```

Otherwise, you will receive an error message instead.

VERY IMPORTANT: You must not hardcode any personal credentials into your code. This is a common and very important SE / security practice. This includes:

your jacobs-id, se-token, and game-username. You will be penalized for this! Read up on what you should do on the next page.

Store your personal credentials as environment variables

This is how you can store your data as envrionment variables. Run the commands below on a Linux/Mac terminal.

```
export HISTIGNORE=”export*” # This makes sure the export commands that
                            # follow do not get stored in the log
export JACOBS_ID=”your jacobs-id”          # Please use JACOBS_ID
                                           # Change content between the “
export SE_USERNAME=”your se-username”      # Please use SE_USERNAME
                                           # Change content between the “
export GAME_USERNAME=”your game-username”  # Please use GAME_USERNAME
                                           # Change content between the “
```

Your program should obtain your personal credentials at run-time. To ensure your code would run smoothly during the code review, make sure your code reads the variables from your environment as they are given above.

You may delete the variables by using the unset command like this:

```
unset GAME_USERNAME
```

Hacking or tampering with the server

Your interactions with the server are logged. Any attempt to hack or tamper with the server would make your TAs very unhappy and therefore result in a -100% for this sprint.

Some tools you might need

You could use the curl library to help with your GET and POST requests https://curl.haxx.se. Other libraries are also allowed.

You could use the json library by Niels Lohmann to help you format your JSON string https://nlohmann.github.io/json/ — this should be much easier than manually formatting your JSON string. Other libraries are also allowed.

You could use https://jsonlint.com to format and validate your JSON string.

You could use Postman to debug your POST requests / formatting https://www.getpostman.com.