Quality Assurance Document

Nikola (nk465)

Introduction

This document provides an overview of the quality assurance practices that were implemented when developing our project. The aim of these practices was to ensure that in the end our project structure is logical and understandable, all the java files look alike, the code functions as intended, and is maintainable.

Code Reviews

I, Nikola, was tasked with reviewing the entire code for the whole group for every sprint. I did a general code review twice per sprint to ensure the code was of high quality, easy to read, and maintainable. Examples of code reviews conducted during the project are:

* Review of the database schema design to ensure normalization.
* Review of the different GUIs to enforce consistency in the font, background, contrast color, and window size.

Refactoring

Refactoring was used to improve the quality and readability of the code. Examples of refactoring performed during the project include:

* Renaming variables to be more descriptive, such as changing ‘u’ to ‘user’ and ‘p’ to ‘password’.
* Renaming method names to be self-explanatory. This was strongly enforced on public methods, to eliminate the need to jump back and forth between files to find out what a particular method does.
* Extracting duplicate code into functions and classes such as the Info and the General classes from the Session package. One of the methods I extracted to the Session class – the method that sets the look and feel of a GUI – shortened the code with almost 300 lines (this was the 2-nd commit on the 30-th of March, I’ve given detailed description there).
* Breaking down complex functions. I extracted self-sufficient parts of the long methods that I spotted, and separated them into multiple smaller, but easier to understand parts.

Project Structure Maintenance

A package-oriented structure was used to make the project more logical and easier to understand.

TODO:

* Add a brief explanation of the 6 main packages that the project is composed of (and how they link to each other)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Include directory creating, naming, and, often, renaming on a later stage of the project
* Adding some specific word to the end of file names to indicate part of which package they are
* Moving everything into the appropriate packages and then adding the necessary imports

Maybe add a “General section” where you can list the typo fixing, mention the use of IntelliJ’s and VSCode’s suggestions, and other