

**SCHOOL OF ENGINEERING AND BUILT ENVIRONMENT**

**Dept. of Computer, Communications & Interactive Systems**

**Web Platform Development 2**

**M3I322955**

**Session 2020/21**

**Coursework 2 Report**

|  |  |  |  |
| --- | --- | --- | --- |
| **Polina Jakovleva** | **S1718126** | |  | | --- | | **pjakov200@gcu.ac.uk** | |

“This piece of coursework is the work of Polina Jakovleva and has not been submitted elsewhere in fulfilment of the requirement of this or any other award.”

Signature: \_\_\_\_\_\_\_\_\_ Date: 09/05/2021



Table of Contents

[Link Design 3](#_Toc72606275)

[Persistence 4](#_Toc72606276)

[Test Reports 5](#_Toc72606277)

[1. Test Scope 5](#_Toc72606278)

[2. Test Cases 5](#_Toc72606279)

[3. Summary 8](#_Toc72606280)

[References 8](#_Toc72606281)

# 

# Introduction

The following report describes the methods and processes used in the development of the Web Platform Development 2 project following coursework specifications. The report consists of multiple sections, including Link Design, Persistence, and Test Reports.

# Link Design

This section provides an explanation of the link names, as well as the mapping of links to the application’s functionality.

The main thing taken into consideration when designing the links for this project were design principles identified by web developer Kyle Neath in an article that was recommended as additional reading at one point in the module. Specifically, three key points he raised in it. The first was the fact that URLs are meant for humans, not search engines, and, that being so, it is important to remember to use short, unabbreviated words when designing them, and if the link requires a special character, it needs to be simplified. Second was that the URL should be treated as an agreement and must provide the exact information it sets out to provide, and therefore all links should be named after the functionality their pages provide. And third was that portions of the URL that indicate unique content should be segmented from the main part of the link and be universal – easy to remember for continued usage and free of any additional characters [1].

I believe I have managed to adhere quite closely to these principles by keeping the names minimal and prioritising their memorability and recognition. With most of the application’s functionality being tied in with the use of links, each one of which performs minimal tasks and, often, serves one purpose, this was essential. In order to simplify the running of the project even further, each page of the application, apart from the landing page, which can be accessed through the browser using localhost:3000/, can be triggered with the use of built-in buttons. For example – the login and registration pages can be accessed using localhost:3000/login and localhost:3000/register, as this is commonplace across many websites, is easy to remember and can’t be mistaken for anything else, or the user may simply click one of the buttons provided on the landing page. It should be noted, however, that some buttons within the application are used to merely showcase the intended layout and do not perform any functions – such as the ‘delete’, ‘edit, and ‘share’ buttons on /mycalendar. Upon logging in, a registered user is directed to their exercise goal planning calendar, simply called localhost:3000/mycalendar. The calendar page provides access to multiple functionalities – creating new goals, goal editing and goal removal. While the new goal form page can be accessed with localhost:3000/newgoal or with a click on the ‘Create new goals’ button, the edit and delete functions are built into the /mycalendar page and will be called each time the page is loaded. The page also provides navigation to the stopwatch page, which is intended to allow the user to record their exercise durations in real time. In the event that a user inputs an invalid link, the application will respond with a ‘404 Not Found’ error message, and in the event that the server is experiencing performance issues, with an ‘Internal Server Error’ message.

There was also an attempt to implement the option of goal sharing among users. The feature was supposed to allow the user to copy their weekly set of goals as a shareable link for the other user to save for their own use, and vice versa. The attempt did not lead to any usable functionality in time for the submission, but it was recorded nonetheless and can be seen in the form of some comments.

Overall, this aspect of the application provides a range of responses of different kinds – Mustache templates, HTML files, etc. in order to offer a little view into the breadth of possibilities for link design, and complies with the coursework specifications.

# Persistence

The following segment describes the persistence mechanism of the application.

The application was developed using Node.js and Node Express, as specified in the coursework requirements. NeDB was also used to persist application data, the weekly goals and registered users, onto specified files as opposed to an in-memory database, keeping the data safe when the application isn’t running, allowing for new information to be added and information contained within to be updated or deleted with the use of functions.

At the beginning of the project there were issues defining the goal database structure, as it consisted of two nested arrays and ended up complicating the process of rendering Mustache templates and development of functions. The final structure is simple – each weekly goal plan is defined by the author of the entry, the week for which this entry is for, its status (whether it’s completed or not), and consists of each goal’s day of the week, date, intended type of exercise and duration, the actual duration once the goal has been completed, and the entry ID. Below is an example of how the goals are stored from the database:

{"author": "Polina", "week": "Current Week","status":"Incomplete","weekGoals":[{"dayOfWeek":"Wednesday","dateOfMonth":"05/05/2021","typeOfExercise":"Yoga","goalDuration":"1 Hour","actualDuration":"40 Minutes"},{"dayOfWeek":"Thursday","dateOfMonth":"06/05/2021","typeOfExercise":"Running","goalDuration":"45 Minutes","actualDuration":"30 Minutes"},{"dayOfWeek":"Friday","dateOfMonth":"07/05/2021","typeOfExercise":"Boxing","goalDuration":"2 Hours"}],"\_id":"99kpE9vxtJ6tebwo"}

To add a new goal to the database, the user must click ‘Create new goals’ on the calendar page and the system will then render a Mustache template that contains a form for the new goal. The user will be prompted to fill it out and submit, and the details they’ve put in will be used to create the new goal entry and add it to the database by a function. The editing and removal functions also trigger similar processes.

The structure of the user database is even more simplistic, and each user’s record only consists of their username, encrypted password and ID. An example of how the users are being stored can also be found below:

{"user":"Polina","password":"$2b$10$OoSkb/yBcfUM8cAmdCBiL.AUN4Ad17w5U2kQZOUbT4lyvgFCe0WVW","\_id":"LFijQIj3rOub0gKf"}

For a new user to be added to the database, they must register. Upon selecting ‘Register’ on the landing page, the system will also trigger a Mustache template form, which, this time, will require the user’s username and password of choice. Once filled out and submitted, the application will encrypt the password and add the information to the database, making other users unable to use the same username in order to avoid collision in the future.

# Test Reports

This section provides a collection of test reports and summary of the coursework results.

## Test Scope

Functional testing has been carried out on the following modules:

* 1. Landing page.
  2. Registration.
  3. Login.
  4. Goal overview.
  5. Adding new goal.
  6. Updating goals.
  7. Removing goals.
  8. Stopwatch.
  9. Sign out.

## Test Cases

Test Cases for this project can be seen in *Table 1.*

**Table 1.** Test Cases for the application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Action** | **Intended Output** | **Status** | **Evidence** |
|  | Localhost:9000/ | Landing page loads and provides access to registration and login. | OK |  |
|  | Localhost:9000/  register | Registration page loads and allows user to register. | OK |  |
|  | Localhost:9000/  login | Login page loads, allows user to login using their registered details and access functionality. | OK |  |
|  | Localhost:9000/  mycalendar  getGoalsByPolina() | Goal overview loads all goals by specific user and provides navigation to new goal form and stopwatch. | OK |  |
|  | Localhost:9000/  newgoals | New goal input page loads, processes input correctly and posts new goal. | OK. |  |
|  | Localhost:9000/  mycalendar  updateGoal() | Page loads and triggers updateGoal() function. | OK |  |
|  | Localhost:9000/  mycalendar  deleteGoal() | Page loads and triggers deleteGoal() function. | OK |  |
|  | Localhost:9000/  stopwatch | Stopwatch page loads and allows user to start/pause/reset the stopwatch. | OK |  |
|  | Localhost:9000/  logout | Page signs user out of the system and redirects them to the landing page, restricting access to functionality. | OK |  |

## Summary

All functionality of the landing page, login, registration, goal updating and goal removal work as expected.

After adding a new goal, the information is displayed in the goal overview but does not look as other entries do, as was intended. This is due to the variable having been composed incorrectly.

The stopwatch works as intended but lacks a function that would allow to save the stopwatch result and input it as the ‘actualGoalDuration’ of the goal database.

# References

**[1]** K. Neath. ‘URL Design’*.* Warpspire. [Retrieved 8 May, 2021]. Available at: https://warpspire.com/posts/url-design