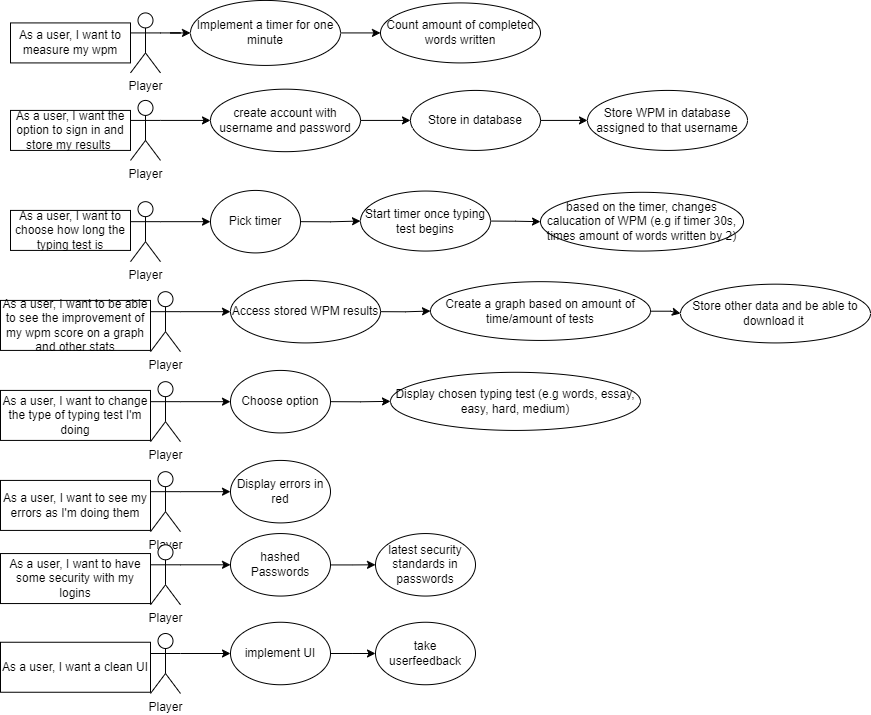
**Introduction**

In this report, I will be describing the methodology of how I created my project, showing my planning, design of my coursework for the COMP1004 module. The project I've chosen is a single page web application of a typing test website, like available resources such as Monkeytype.com or typingtest.com.

Here are my user stories with my intended features:

**Software Development Lifecycle**

During this project, I am using the AGILE methodology, using the SCRUM model to achieve my project, having a scrum meeting at the beginning of every two weeks to outline what my aim is during the 2-week sprint. This ended up totalling up to 8 sprints in total, spanning 16 weeks.

The first week was primarily the planning phase, creating a ULM and flowchart to design the basic logic of the website with a wireframe of what the UI could look like in the end. This is below:

**A diagram of a program

Description automatically generatedA diagram of a program

Description automatically generated**

****

Here you can see (explain the logic and whatnot)

In my second sprint,

Here are my sprints paired against my user stories:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Features | Sprint 1 | Sprint 2 | Sprint 3 | Sprint 4 | Sprint 5 | Sprint 6 | Sprint 7 | Sprint 8 |
| Able to measure WPM with a test |  | WPM/Timer visible in console, array not randomised | WPM/Timer displayed on website; array randomised and tested | Gathered user feedback, attempted word generation, failed. | Tested and implemented user feedback implemented, with positive reception |  |  |  |
| Sign in/Log in |  |  |  |  | Created HTML of login page, investigated JSON | Log in works after fetching data from JSON | Able to create an account which persists. | Properly tested. |
| Store stats and view them |  |  |  |  |  |  | Stats added to JSON, can be viewed with graph included. | Properly tested. |
| Export stats into JSON |  |  |  |  |  |  | Implemented and tested |  |
| Difficulty options |  |  |  | Timer options implemented | Difficulty options implemented | Difficulty options tested (fixed bugs) |  |  |
| Displaying correct/incorrect inputs |  |  |  | Feature implemented and tested. |  |  |  |  |
| Security features |  |  |  |  |  | Passwords hashed,  Passwords standards enforced | Tested properly |  |
| UI | Wireframe created |  | Wireframe implemented | Took user feedback | Wireframe redone, dark mode and light mode implemented. | Wireframe created for login/signup pages and implemented | Create wireframe for stats page and implemented. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Not worked on | Started but limited progress (<50%) | Over half way done (>50%) | Nearly finished, if not finished, testing required. (>90%) | Finished (100%) |

|  |  |  |  |
| --- | --- | --- | --- |
| **SPRINT WEEK 1-2** | **SPRINT WEEK 3-4** | **SPRINT WEEK 4-6** | **SPRINT WEEK 7-8** |
| **Goals**  - Create and display user stories  - Create a flowchart to display the logic of website  (basic logic to be used in prototype)  - Create a wireframe using Figma (UI)  -Create UML for prototype logic | **Goals**  - Create a skeleton of the code  - To amend ULM + Flowchart to match actual logic if any mistakes  - Adhere to wireframe - Create a working prototype   * Can do a 60second typing test for array of words. * Gets WPM (words per minute) | **Goals**  - Create new ULM to show new structure  - Redo UI to fit the wireframe  - Display WPM + timer onto website rather than console  - Make correctly inputted words autodelete on submit  - Creation of proper testing  - randomised input array generation | **Goals**  **- I**mplement difficulty options  - Implement timer options  - Creation of DOM diagram  - Tell user typing test is over  - Take some user feedback  - Display errors on text live  - Change WPM calculation  - Put comments in .js file for better readability  - Attempt to use Word generation instead of using an randomised array |
| **Status**  - Completed  - Goals met | **Status**  - Completed  - Goals not met | **Status**  **-** Completed  - Goals met | **Status**  **-** Completed  - Goals not met |
| **Next steps**  - Using UML and flowchart as base create a working prototype  Features of prototype  -timer  -calculation of wpm  - typing the required words in an input box | **Next steps**  - Using UML and flowchart as base create a working prototype  Features of prototype  -timer  -calculation of wpm  - typing the required words in an input box | **Next Steps**  - Implementation of difficulty/timer options  - Letters go red if incorrect, or green if correct  - Creation of new Ulm to show structure  - Gather user feedback from close friends | **Next Steps**  - Creation of ULM to show new structure  - Show WPM live  - Make login system  - Implement user feedback  - Attempt to do word generation, difficulty options posed problems with an array of words |
| **SPRINT WEEK 9-10** | **SPRINT WEEK 11-12** | **SPRINT WEEK 13-14** | **SPRINT WEEK 15-16** |
| **Goals**  **-** Create ULM to show new structure  - Show live wpm and display it  - Make login system html, allowing user to go back  -Create new wireframe  **-** Clean UI up further (user feedback)  - Once implemented, take more user feedback  - implement a light/dark mode for UI  - Implement difficulty options effectively | **Goals**  - Allow to sign in with their details  - To be able to read from a Json file (holds passwords/users)  - Allow user to create new account and store it in Json file  - Allow creation of new user  - Ensure commenting has continued in new code  - Create wireframe log in/sign in/log out html additions and implement them  - Hash passwords for security  - Fix paragraphs not deleting properly bug | **Goals**  **-** Allow WPM to be stored and assigned to their user details  **-** Put theses stats within a Json that can be exported, using blob  - Create wireframe for page which displays theses stats  - Allow stats to be visible on webpage  - Replace alerts with nice UI implementation  - style log in/sign in/log out html additions according to wireframe  - Allow WPM data to be displayed as a graph  - Retest old features and add new features to testing | **Next Steps -** Final ULM  - Writing Report  - Add any more QOL features if wanted, e.g. more stats in graph, ability to change username/password, ability to reset stats etc  - Do final testing to ensure all features are up to standard  - Get user feedback |
| **Status**  **-** Complete  - Goals not met | **Status**  **-** Complete  - Goals not met, not possible to meet unless using node.js | **Status**  - Complete  - Goals met | **Status**  **-**  incomplete  - Goals not met |
| **Next Steps**  **-** Word/paragraph generation or grabbing it from the internet seems practical for a larger project but not needed for my scope.  - Allow to sign in with their details  - Allow WPM to be stored and assigned to their user details  - Allow it to be displayed as a graph | **Next Steps**  **-** Allow WPM to be stored and assigned to their user details  **-** Put theses stats within a Json that can be exported. Using blob  - Replace alerts with nice UI implementation  - style log in/sign in/log out html additions  - Allow WPM data to be displayed as a graph | **Next Steps -** Final ULM  - Writing Report  - Add any more features if wanted, e.g more stats in graph, ability to change username/password, ability to reset stats etc | **Next Steps**  - Submission |

|  |  |  |  |
| --- | --- | --- | --- |
| TEST | EXPECTED OUTPUT | ACTUAL OUTPUT | PASSED (Y/N) |
| On validate input (after pressing space), accept it and deletes word from display |  | Same as expected | Y |
| Recognises validate input automatically, and submits it |  | Same as expected | Y |
| Randomise word bank |  | Same as expected | Y |
| Display WPM at the end of test, calculated by time and number of characters |  | Same as expected | Y |
| Have a test of 1 minute (by default) |  | Same as expected | Y |
| Display option to change timer |  | Same as expected | Y |
| Have a test of 30s (after using options) |  | Same as expected | Y |
| Have a test of 90s (after using options) |  | Same as expected | Y |
| Have a test of 120s (after using options) |  | Same as expected | Y |
| WPM is updated live, displaying predicted if current speed is maintained |  |  | N (abandoned) |
| UI adheres to wireframe |  | Same as expected | Y |
| Display option to change difficulty |  | Same as expected | Y |
| Implements punctuation |  | Same as expected | Y |
| Implements capitalisation |  | Same as expected | Y |
| Implements paragraphs |  | Same as expected | Y |
| Paragraphs difficulty is exclusive difficulty option to punctuation and capitalization |  | Same as expected | Y |
| Difficulty options can only be changed before/after a test |  | Same as expected | Y |
| Words completed are removed at the end of the first line, allowing all words to move upwards allowing new words to take its place. |  | Same as expected | Y |
| Paragraphs are deleted in a similar fashion, at the end of the line. |  |  | N (Abandoned) |
| Generate words instead of grabbing it from a list |  |  | N (abandoned) |
| On text box, text goes red if error made in input box |  | Same as expected | Y |
| On text box, text goes green if no error is found |  | Same as expected | Y |
| Clicking log shows username and password input boxes |  | Same as expected | Y |
| This also hides typing test page which is shown by default |  |  |  |
| Allows user to create an account |  | Same as expected | Y |
| allows user information to be stored and to be able to login at a later date |  | Same as expected | Y (partial as uses local storage, so based on personal browser cache) |
| User’s wpm is stored |  | Same as expected | Y |
| Graph is made based on user wpm |  | Same as expected | Y |
| Graph is updated with new statistical information |  | Same as expected | Y |
| Stats are stored and tied to the users’ account info |  | Same as expected | Y |
| Stats are updated after every test |  | Same as expected | Y |
| Stats can be visible from a separate page |  | Same as expected | Y |
| Stats can be exported and downloaded into a JSON file. |  | Same as expected | Y |
| Error message if log in information is incorrect |  | Same as expected | Y |
| Error message when signing up if password does not meet security standards |  | Same as expected | Y |
| Hashes passwords, add some security |  | Same as expected | Y |
| Light mode |  | Same as expected | Y |
| Dark mode |  | Same as expected | Y |

User feedback (1.1)

* Found it abrupt that words disappeared after every word, making typing harder as not as natural to do
* UI could be cleaner.

User feedback (1.2)

* This was taken after user feedback initially implemented.
* Like the change that words are deleted at end of line
* User likes it far better, flows far nicer and UI is far cleaner.
* Only required slightly darker shades for background colour
* Suggested that background dimmed if typing test begins.

User feedback (2.1)

* My implemented css does not work on their laptop for some unknown reason.
* Potentially OS issue as they’re using windows 10, not windows 11 but can’t say for sure.

Challenges

* Json files
  + Lack of documentation on how properly fetch data from a json without using node.js (as this is usually the go-to way to do this)
    - Must create a local web server using visual studio preview extension or python -m http.server command in terminal to fetch data from the json file
  + Unable to properly write to a Json file without using node.js, thus to store the new user data, it will be stored in local storage which is persists unless cache is deleted.
  + Lack of documentation of how to export a json file
    - Must use blob and anchor.
* Creation of highlighting green/red letters when writing
  + Required creation of solution of creating a span of each word and styling it individually
* Effective word generation difficult and impractical for purposes.
  + Either uses node.js to grab from internet (I think)
  + Or words created often weren’t real.
* Coding challenges with html, JavaScript and CSS
  + Lack of experience using theses languages
  + Learning the syntax required and new features such as query selector, span of the n-nth child, spans etc
    - In particular, talking about implementing the styling and endOfLine Function
  + How to properly combine the html and javascript successfully
  + How to hide divs in the html using css/javascript