https://dle.plymouth.ac.uk/pluginfile.php/3242871/mod\_resource/content/0/COMP1004%20-%20Assessment.pdf

Backlog

* User stories
  + As a user, I want to be able to measure my typing speed in wpm (word per minute).
  + As a user, I want an option to stay logged in, so that I don’t have to enter my credentials every time.
  + As a user, I want to see an error message if I enter incorrect login.
  + As a user, I want to see my typing speed scores stored and displayed within a graph to be able to track my progress which is linked to my account.
  + As a user, I want to be able to have options of texts to test typing ability against. (e.g word vs paragraphs)
  + As a user, I want to be able to pick if the text I’m choosing to test against has punctuation or capital letters.
  + As a user, I want to be able to pick how long the typing test will be.
  + As a user, I want to be able to see a digital 2d keyboard lighting up and showing me the keys, I inputted.
  + As a user, I want to see which words/letter I’ve made a mistake on and the ability to change my mistakes during the test.

Using SCRUM SDLC – in first sprint, Creation of introduction + sdlc description -> start of initial sprint explanation -> planning phase + start of design phase.

* Make skeleton of code

**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.

**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.

In my first meeting, here is my objectives:

Meeting - 1/11/2023,

* Skeleton outline of webpage
  + A place to write and display words for them to write
  + Put html learning in practice – is current blocker; Mozilla html for help
* Finish introduction
* Use notepad++ and open document with chrome
* Write project vision

Change to from moving difficulty options to “easy,hard , medium” but punctation , capitals or proper texts, allowing to pick one or multiple of them at the same time

|  |  |  |  |
| --- | --- | --- | --- |
| **SPRINT WEEK 1-2** | **SPRINT WEEK 3-4** | **SPRINT WEEK 5-6** | **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 randomised array of words. * Gets WPM (words per minute) | **Goals**  - Create new ULM to show new structure  - Redo UI to fit the wireframe  - Implement the feature to change the timer of typing test  - 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 |
| **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**  **-** Adhering to the wireframe UI design  - Implementing new features  Time options and difficulty options+ pseudo-random expected inputs  - Polish of previous features :  auto deleted in input box, WPM + timer displayed not in console + randomised array | **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** |  |
| **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   * Attempt to use Word generation instead of using an randomised array | **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  - Create new wireframe for light/dark mode.  - 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 | **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  **-** Create new ULM for design  - Retest old features and add new features. | **Goals**  - n/a |
| **Status**  **-** Complete  - Goals not met | **Status**  **-** Complete  - Goals not met, not possible to meet unless using node.js | **Status**  - Incomplete  - Goals not met | **Status**  **-**  incomplete  - Goals 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 -** New ULM |  |

|  |  |  |  |
| --- | --- | --- | --- |
| TEST | EXPECTED OUTPUT | ACTUAL OUTPUT | PASSED (Y/N) |
| On validate input (after pressing space), accept it and deletes word from display |  |  | Y |
| Recognises validate input automatically, and submits it |  |  | Y |
| Randomise word\_bank |  |  | Y |
| Display WPM at the end of test, calculated by time and number of characters |  |  | Y |
| Have a test of 1 minute (by default) |  |  | Y |
| Display option to change timer |  |  | Y |
| Have a test of 30s (after using options) |  |  | Y |
| Have a test of 90s (after using options) |  |  | Y |
| Have a test of 120s (after using options) |  |  | Y |
| WPM is updated live, displaying predicted if current speed is maintained |  |  | N |
| UI adheres to wireframe |  |  | Y |
| Display option to change difficulty |  |  | Y |
| Implements punctuation |  |  | Y |
| Implements capitalisation |  |  | Y |
| Implements paragraphs |  |  | Y |
| Clicking log shows username and password input boxes |  |  | Y |
| Allows user to create an account |  |  | N |
| allows user information to be stored and to be able to login at a later date |  |  | N |
| User’s wpm is stored |  |  | N |
| Graph is made based on user wpm |  |  | N |
| On text box, text goes red if error made in inputbox |  |  | Y |
| On text box, text goes green if no error is found |  |  | Y |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

User feedback (1.1)

* Implement cleaner UI
  + Current is fine but not best looking
  + While similarities are obvious, wireframe is not effectively implemented
* Change how the words are displayed
  + Currently, words are removed from list and its all redisplayed
    - This means that the next word needed to be type gets moved to the start
  + This is abrupt and disrupts the natural flow of the typing test
* Space needs to be included as correct/incorrect stylisation
* Otherwise, rest is well done

User feedback (1.2)

* This was taken after user feedback initially implemented
* 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
  + How to properly combine the html and javascript successfully
  + How to hide divs in the html using css/javascript

Useful for later when doing the statistical analysis of the user.

<https://stackoverflow.com/questions/48199781/without-node-js-or-running-a-server-is-it-possible-to-create-output-a-json-file>

<https://developer.mozilla.org/en-US/docs/Web/API/Blob>

var json\_object = JSON.stringify(json\_data.users);

const blob = new Blob([json\_object], {type: 'application/json'});

var anchor = document.createElement('a');

anchor.download = "data.json";

anchor.href = window.URL.createObjectURL(blob);

anchor.innerHTML = "download"

anchor.click();

console.log(anchor);

blob is used to export json files

fetch is used to get data from json file (import)

localstorage is used to store data which will persist even when browser closed.