Sprint Review

Design:

**Proposed Goals:**

**While creating the necessary assets for the project was a goal, the Design department had more autonomy. This was the result of the continuous improvements undertaken throughout the Design Process in the initial stages, including redesigning the page layout and creating adjustments throughout development.**

**This included:**

* **Creating assets for the page**
  + **Mock profile pictures**
  + **Panels**
  + **Post Boxes**
* **Expanding the design for future sprints**
* **Further details on the exact implementation can be found in the “Sprint A” Documentation**

Process:

**There were few to no difficulties in achieving the Design goals, as the goals were promptly delivered and distributed to the team via GitHub when they were necessary. At worst, a request for a certain asset may have been given somewhat last minute, though this was usually promptly handled. This was likely result of the deliberate choice the team made of assigning users with prior design experience to the design department. The design of the website and its assets was kept in a consistent theme, using similar shades of blue across the webpage, the Menlo font for the header, and variations of the Myriad font for all other content.**

**Regarding the assembly of user feedback, a survey was conducted to access the experience users had with the website, along with interviews that let the testers use the webpage directly and offer feedback after using the website. This was concentrated into a feedback summary to distribute to the team for further discussion. The proposed design of the minimum viable product was also periodically updated as a result of user feedback.**

Outcome:

**For the Design aspect of the Project, the mock-up of the front page was updated to incorporate criticism gathered with the use of surveys, to generate an improved Design for future sprints. Where necessary, the front end was continuously supplied with colour-data and design elements. Every sprint goal was achieved, and coherency amongst the designed assets was assured.**

Front End:

**Proposed Goals:**

**Setting Up the MVP Design of the Website. This consists of the core functionalities of the website, along with the basic style requirements outlined previously in prototypes designed by the Design Department. The MVP is of the front page of the website.**

**This included:**

* **Setting up all stylistic Panels of the Front Page**
  + **Navigation Panel, Including Text and Hyperlinks (Placeholders)**
  + **Trending Panel, with Placeholder text**
  + **Fakebook Header at the top**
* **Implement a scrolling Timeline with user Posts and Facts**
  + **Posts, Facts and Profile Pictures are to be sourced from the database via the back-end**
  + **A box for the user to post something sits at the top of the timeline, with a “Post” button to confirm the Post/Submission in question**
* **Further details on the exact implementation can be found in the “Sprint A” Documentation**

Process:

**The proposed goals have been deemed realistic early on with confidence. The process of learning angular, html, typescript, JavaScript and CSS proved to be very difficult, mainly due to the volume of information available on the given tools. Angular in particular proved to have a high learning curve, however with diligence and dedication the front-end was ultimately realised despite these difficulties.**

**Connecting the front-end to the backend proved to be quite challenging as well. Difficulties lied in both learning about the front-end and back-end while understanding the given departments code. The code was discussed in detail between departments, which was quite time consuming, especially issues surrounding the speed of fetching data from the back-end. Ultimately, HTML rendering was utilized, all development branches were merged and the site worked. Minor glitches remained, however otherwise the front-end goals were successfully achieved.**

Outcome:

**The front-End Page was successfully implemented from a design perspective. The Page looks almost identical to the proposed prototype the team settled on. The functionality has deviated slightly from the original goals. Profile pictures are momentarily replaced by placeholders in the front end, as a working pipeline from the Database to the front-end was ultimately too ambitious for the time scope of a week. The means to retrieve Posts, Facts, Likes and other values from the back-end are in place, however these are yet to be implemented, as a result of discussions amongst team members that functionality between front-end, back-end and the database should be fleshed out before fully committing it.**

**The functionality of more trivial parts of the user interface such as like buttons and comments are momentarily placeholders, to ensure smooth integration once the dataflow from the database to the front-end via the back-end has been achieved. For now the like button changes colour when pressed, without incrementing the posts number of likes.**

**The working front page faces little to no issues during use and has successfully reached most of the goals that were realistically in scope for this sprint. Functionalities that were altered in the interest of stability or time constraints were primed, so that implementation and retrieval of data from the backend would be ready to use.**

Back End:

**Proposed Goals:**

**To set up the retrieval, selection and transfer of data from the database to the front-end. The Pathways to enable the functions of the back-end requires close work with both the front-end and database.**

**This included:**

* **Sending data from the database to the front-end**
  + **This includes profile pictures, likes, posts, and other numeric/string data**
* **Filling the database with randomly generated sentences**
  + **Posts should be randomly generated in the backend and stored to the database for later retrieval**
  + **A sample of 100 sentences should be generated and stores**
* **Pulling Data from the front-end** 
  + **Likes, comments, Posts written by users amongst some of the data to be retrieved**
* **Further details on the exact implementation can be found in the “Sprint A” Documentation**

Process:

**The difficulties encountered during the back-end development mainly surrounded technical details. A working concept was drafted early on, however the implementation and code itself proved to be more difficult. In particular, which files to use and how to write functions were rather challenging. These difficulties were overcome, ultimately setting up the server and api, to allow the MongoDB database to connect to it. To ensure that HTTP requests were working as intended without errors, The application “Postman” was used, while “RoboMongo” was utilised to visualise requests to the database. The entire process involved a lot of research and trial and error.**

**The aims did not have to be adjusted and were all achieved in the sprint’s timeframe.**

Outcome:

**The back-end has successfully created a pipeline from which data is provided to the front end. By closely working together with the Database department. Methods to draw data from the Database via Json files were implemented, with methods to write back to the database being present. While it proved to be difficult at first, methods to successfully pull data from the front end were implemented and tested via black box testing. Afterwards, the backend was successfully merged with and connected to the front-end and database.**

Database:

**Proposed Goals:**

**To set up the retrieval, selection and transfer of data from the database to the front-end. The Pathways to enable the functions of the back-end requires close work with both the front-end and database.**

**This included:**

* **Set up sample data**
  + **Numeric values such as likes should be placeholders**
* **Store Profile pictures provided by the Design department**
* **Nonnumeric data should be plausible**
  + **Usernames for instance should make sense, and not be random text**
* **Work with Back-end to enable storing of data**
* **Further details on the exact implementation can be found in the “Sprint A” Documentation**

Process:

**To initially fill the database with randomly generated sentences, the early prototype of a sentence generator in java was updated and repurposed to create more complex sentences, exporting 100 instances as a json array file to import into the containerized mongoDB instance. This would enable the other sprint goals to be thoroughly tested with sample data, being drawn directly from the database. The numeric values were determined to be placeholders at the moment of writing, so randomly generated numbers were inserted for values such as likes. To ensure user data, such as usernames, were plausible, as outlined in the sprint documentation, a user data generation program was written and used. Similarly to the sentence generator, results were stored as a JSON file and fed into the database. One of the stretch goals, storing facts in the database, was reached early due to the front end developers requiring it.**

**Difficulties were encountered when trying to seed the database automatically when building the docker container, without using “mongoimport” on the command line. An initial attempt at solving this was editing the docker compose file to seed the database via another temporary container, though issues with authentication, connections between containers and the wait-for.sh script caused problems with this approach. Writing another Docker file for the database container proved too difficult. Instead, The best approached was found to be using a node file to seed the database, when spinning up the docker-container. It is called from the db.js file and checks for pre-existing data in the database, in which case it does nothing. Else, some mongoose schemas are utilised and the required content is imported from all the necessary JSON files.**

**Difficulties arose afterwards when trying to connect the database with the front and back-end, as all parts of the given code had to be understood by the team in order to successfully pull data through one end to the other. The storage of likes proved to be outside of the projects scope for this sprint, so the features importance in relation to the aims of the project were noted to be discussed at a later date. All other goals were successfully implemented however, and function well.**

Outcome:

**The given goals of the sprint were achieved, successfully implementing all functions, with the exception of likes. This prompted further discussion as to the necessity of likes in the first place, and whether it would benefit the aims of our project. It was decided to be discussed after the sprint. The database can be successfully seeded and files can successfully be transferred between the front-end, back-end and database.**

**In addition to the Sprint goals, connectivity by usage of json files was stabilised, and the entire project was merged on git, to ensure stability and check whether it functions. The site functioned and further points to improve upon in future sprints were discussed.**