Deliverable 3

Agile Methodology (190007613)

Our team has successfully kept to Agile and Scrum methodology throughout this semester. Our current meeting plan is as follows:

Weekly Meetings:

1. **Tuesday, 3:00 PM:** Meet with supervisor to report current progress, ask questions, and evaluate how we will continue throughout the rest of the semester. Typically begin a sprint here.
2. **Thursday 3:00 PM:** Individual team meeting to discuss progress on what was discussed on Tuesday and to clarify what needs to be done daily throughout the rest of the week.

We have kept much more strictly to the Scrum and Agile development plan this semester than last semester. Specifically, we have made sure that everyone knows when meetings are happening and clearly notifies our supervisor (if required) and the team if they will be missing a meeting. Communication has, on the whole, greatly improved from last semester. Members are more aware of the tasks every individual has been assigned. Roles within Scrum framework haven't changed: with all of us working as developers and collaboratively setting priorities for each week. Jon has maintained his role as Scrum master through note keeping and delegating, but others have taken notes if Jon is not available.

Here is an example of how Agile development has been realised within our team: We are currently developing an extension which allows for inline comments as well as finalising file and reviewer systems. We noticed that our database needed revamping which involved collaboration from all three members of the backend team. This meant that backend shifted priorities from inline comments to database restructuring. Simultaneously, we keep comments in the backlog and assigned to the same people, meaning that once the database is structured correctly, we can seamlessly return to inline comments without needless discussion regarding plans. This is further facilitated by our sprints, which give clear priorities for each member.

All of our notes comprehensively and explicitly state what we have done well in a given Sprint / what needs to be improved when entering a sprint in the future. This has allowed us to stay on track regarding our work and to have Sprints improve as we continue.

Additionally, all weekly tasks are added onto our Trello board to allow for easy tracking of progress of specific tasks. See Appendix 5 for weekly meeting notes that discuss our progress and clarify what we must get done within a given week.

Currently Implemented

Frontend (160005106)

For ease of development, 'Web Component' structure is adopted. This can be found in the components folder of the source code. By componentizing frequently used html tags, we avoid copying the same code. Thanks to this, 190015047 and 160005106, who develop the front-end, were able to efficiently reduce the time required for front-end development. Also, by adopting the component structure, when it is necessary to change the design of one component, we were able to

maintain design consistency by changing the style of the component instead of changing the style of the html tag one by one. As a result, we were able to effectively reduce unnecessary time.

App.js - By using the Header component, even if the user moves to another page, the same header is visible. When a user logs in, a button to go to the profile page and a button to go to the Articles page are created in the header. If the user is not logged in, a button to move to the login page and the registration page is created.

We implemented Single Page Application (SPA) using a library called Router. Thanks to SPA, users can have a positive experience with web pages. When users navigate to another page, they see a page where the current page was created dynamically, rather than a completely new page (not a page that refreshes). Therefore, users will experience that the web page has a fast response speed.

Main Page - The main page is not implemented with big features.

Register Page -To register as a member, the user must enter an email and password. Our web application verifies that the email format entered by the user is valid. If the format of the email entered by the user is not valid, a warning message is displayed above the text field. We also added a Repeat Password text field to check if the user entered the password correctly. If the two passwords are different, a warning message will appear above the text field.

Login Page - It has been implemented so that users can easily log in using the ID of another group using the Dropdown menu.

Profile Page - The Profile page allows users to change their nickname, email address and password.

Articles Page -Instead of navigating to another page to view new Articles, new Articles are added under the existing Articles by clicking the Get More Articles button. Users can check multiple Articles at the same time on one page.

Article Page - We implemented a tab so that users can stay within one page.

Article Page Description - In this tab, users can get basic information about the article.

Article Page Article - In this tab, users can check the actual article.

Article Page Code - In this tab, users can check all files existing in Article. When a user clicks on a file, the user can see the comments on the file. The comment contains the information of the line number, and the code corresponding to the line number is displayed under the comment. Thanks to this, the user does not have to search every single line to see the code corresponding to the line number. Users can comment on files. To write a comment, the user must specify the content and line number of the comment. When the user writes a comment, a modal is created. This allows the user to check the code while writing comments.

Article Page Comments - In this tab, users can write an overall review about the Article. The basic function is the same as the principle of writing comments in the Code tab.

Upload Article Page - Users can upload their own Articles here. To upload an article, the user needs a total of 5 pieces of information: Title, Other Authors (Article uploader is automatically added to Author, so the user needs to enter other Authors here except himself/herself.), License (by clicking the dropdown menu the user can easily select a license), Files (in MVP (Deliverable 2), users can upload multiple files or only one file. Also, the number of files to be uploaded is displayed on the page) and Description. In addition, the user can easily input all information except the file among the

above five information (Title, Other Authors, License, Files, and Description) through the JSON file. When a user imports a JSON file, the application reads the file and fills in all text fields.

Back-end (190012655)

The APIs that are used to communicate between the front-end and back-end currently have two editions. In order to maintain further development by the web dev team, it was deemed necessary for testing to be able to continue while the database all the way up to the APIs was being restructured. This led to the construction of a series of 'testing APIs' which could be used without the back-end getting in the way. Alongside this are a series of APIs which are currently being developed which represent both the requirements of the front-end alongside the new database structure, especially as we continue to implement the editorial process.

A detailed list of what the font-end require through the service controllers, including what will be achieved through database queries, interfacing with the index file system, and modelling through pagination and user stores, has been created which covers the current plans for development. This is an important piece of communication between the front and back -ends and has been extremely useful in development in this area. Currently the functionality supported by database queries, the indexed file system, and pagination work completely, while the user store is being reconnected at this time.

The index file system is a structured series of files used to store those which are uploaded by the user, both as supplementary files as well as actual code. Stored currently in the project files, the system is generated inwards by the back-end, storing all files associated with an article, then inside a particular version, then in the folders they are uploaded in. All files are received from the front end as a zipped file in base64, this is then decoded, placed into the right index location, extracted, and deleted. As the process is carried out, the index grows to suit its needs in a way which is simultaneously logical as well as easy to understand for testing purposes.

Changes Since Last Deliverable

Frontend (160005106)

Security Issue - MVP had an extremely critical flaw in security. The problem was that the user could go to the Articles, Article and Article Upload page without logging in. If the user correctly entered the URL corresponding to the pages, they could go to the pages without credentials. However, the current application checks the user's credentials every time the page is moved and redirects to the login page if the user is not logged in.

Functionality - Comments made by users in MVP cannot be edited once they are written. However, from now on, users can edit their own comments (code tab).

As the method of file hosting has changed, the method of uploading files on the upload page has also changed. Users can upload only one compressed file (.zip file only), not selecting files.

Redundancy Issue - There was a lot of unnecessary redundant code in MVP. Deleted these unnecessary duplicates.

Dead Code Issue - When making MVP, a lot of Mock Data existed because the front end was implemented before the back end. Since these mock data are no different from dead code, it is a

waste of memory and computation time. Therefore, all information that is not shown on the page or unused information has been deleted.

Back-end (190012655)

In moving forward from the MVP, 2 clear areas in terms of back-end functionality were identified. These were the import and export of articles and their content between coding journals and supergroup login. Both of these issues were some of the first to be addressed, and have no problem being carried out from the back-end.

While the structure of the database and the java that supported it worked well for the second deliverable, we wanted to avoid pushing it past its limits by continuing development on what would become shaky foundations. As such, we resolved to start back at our basic understanding of what really is needed to allow the back-end to support the front-end and vice-versa. Through a series of brain-storming sessions at the white board and further modelling we were able to determine what our flow path would look like as well as how the database would change as we continued development.

It was important to choose a design that would support our views for desired functionality by deliverable 4, rather than taking something not adequate and hamming in other pieces. We decided to go with a structure which supports versioning, the editorial process, comment threads, and search functionality. We are also confident that if any further functionality is added, the base structure we have adopted will allow us to stretch to this.

From this planning stage, we created the list of service controller requirements mentioned in currently implemented functionality. The processes that would lie in-between these and the database were modelled and finally developed into real code.

Supergroup Interaction - 190005092

This semester has seen several changes occur in the supergroup protocol, to adapt to the implementations of some groups and to also incorporate new ideas put forward by teams to enhance the federation experience.

Our priority was to make sure implementation of the supergroup MVP protocol was implemented and tested fully as this was something implemented but not yet tested by the end of deliverable 2.

Earlier this week, the supergroup announced two new API implementations for the federation: **/users** to allow for user information to be passed throughout the federation, and **/submission** to streamline our previous approach of downloading then uploading articles from one journal to another. As these have only just been approved by the supergroup, implementation of these is limited to hardcoded 501 “Not Implemented” returns.

Looking at the protocol, we’ve decided that including all the versions and comments of an article while downloading a submission seemed unnecessary – instead treating it like a brand-new uploaded submission, where the details derived from the upload are added to the input boxes of creating a new file. We will include the latest published version of an article with all discussion comments as the article sent when requested as we believed that sending drafts of articles were irrelevant.

As a group, we've also begun the process of getting our own idea for added implementation between the supergroups with a federation wise search of author names, so that one ideally could search for an author's name and be returned articles over several journals. We have submitted a protocol proposal and are waiting for the others to scrutinize and vote on whether they are willing to implement it.

In terms of the supergroup, we've tried to be careful as supergroup co-ordinators not to implement protocols that could lead to problems in the future, taking our time to scrutinize and begin implementing them into our groups before finalising the protocol. However, as the March deadline approaches, I can see that final testing and implementations from some groups may be limited, restricting us in what and who we can test with. This is something I will be keeping an eye on and asking at meetings.

Plans for Rest of Semester (and potential problems) (190015047)

Working backwards from the deadline, we have 6 weeks until the final deliverable is due. Below is a list of tasks that we have yet to do and what week we expect those elements should be completed. At the end is a list of potential issues that we could run into throughout this process.

Week 1

- [Backend] Replace the current dummy API responses that we're using for testing with real data.
- [Backend] Finalize and test the functionality that will allow moderators to view, accept and reject submitted articles.
- [Frontend] Complete the functionality of the view that will allow moderators to view, accept and reject submitted articles.
- [Backend] Complete all initial JUnit tests for all the API calls.

Week 2

- [Frontend] Fetch and display different versions of submitted articles.
- [Frontend] Send files to the backend for storage upon article upload.
- [Frontend] Fetch and display files from the backend when a user opens them in the file selection.
- [Backend] Translate the majority reviewer votes on each article into the corresponding action in the database.
- [Frontend] Complete the functionality of the view that will allow reviewers to view, accept and reject submitted articles.

Week 3

- [Frontend] Implement the view that will allow regular users to apply to become a reviewer.
- [Backend] Implement the logic and API calls that will allow a regular user to be promoted to a reviewer.
- [Frontend] Implement the view that will allow moderators to choose 3 reviewers for a submitted article.

Week 4

- [Backend] Implement the exporting of comments as specified by the new supergroup protocol.
- [Frontend] Add the view that will allow users to search for articles or authors.
- [Backend] Implement the logic of the search function.
- [Frontend] Display replies to discussion comments as nested/indented instead of flat.

Week 5

- Carry out final extensive testing to ensure our code meets deliverable requirements.
- Ensure our implementation follows the supergroup specification and is well-tested against other groups in our supergroup.

Week 6

- Finalize code
- Write the final deliverable report

Problems that could arise

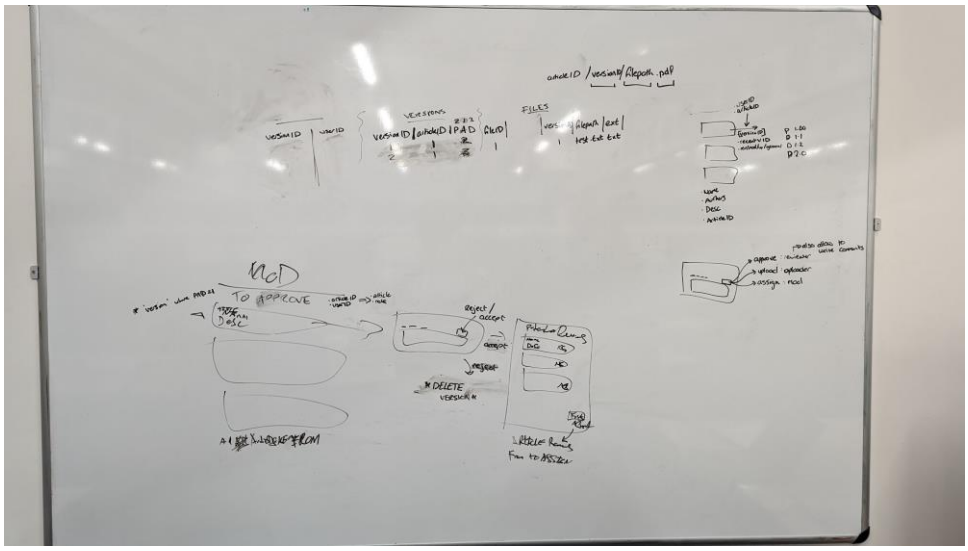
One of the issues that we might expect to run into is that the supergroup may end up changing the protocol near to the deadline and we would need to scramble to remain compatible with the rest of the groups. Next, the specification may be changed at the last second to something unexpected which could cause us to re-evaluate the logic of our submission. It may also be that some bug will unexpectedly take us a longer-than-expected amount of time to fix, which would inevitably move back the progress on the other features. Lastly, factors out of our control such as one of us falling ill or being unable to work for personal reasons as well as the deadline clash that everyone will have during week 5 and 6.

To ensure that we remain on–track and don't fall too far behind schedule in the face of these issues, we have decided to hold in-person meetings every Tuesday and Friday. On these days, we will sit together in the labs and do nothing but work on this project to make sure that everyone is at least putting in the minimal work required to push the project forward as well as to facilitate easy communication between everyone. While we cannot do much about factors outside of our control like people falling ill or a change in specification, we will do everything in our power to ensure that all factors that are in our control are done to the best of our ability.

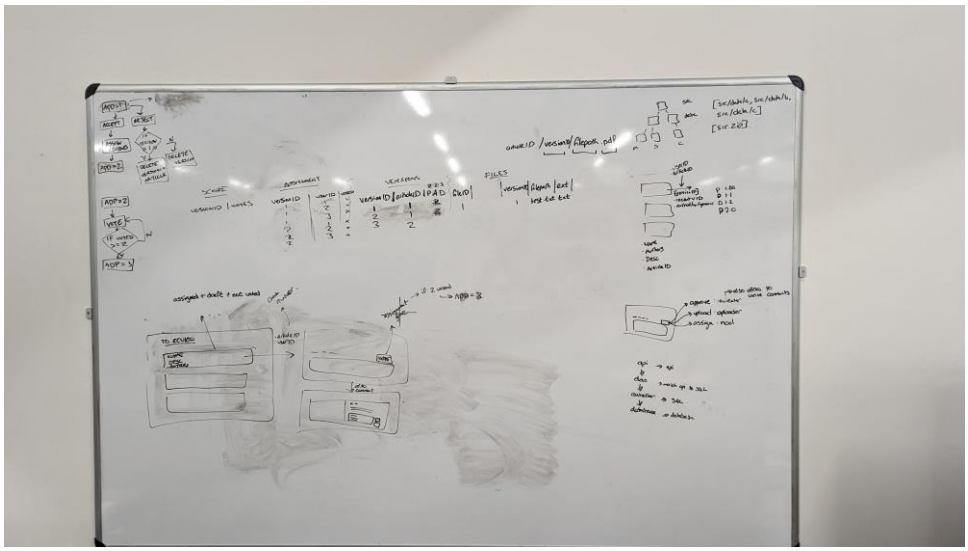
Appendix

Appendix 1 – whiteboard sessions

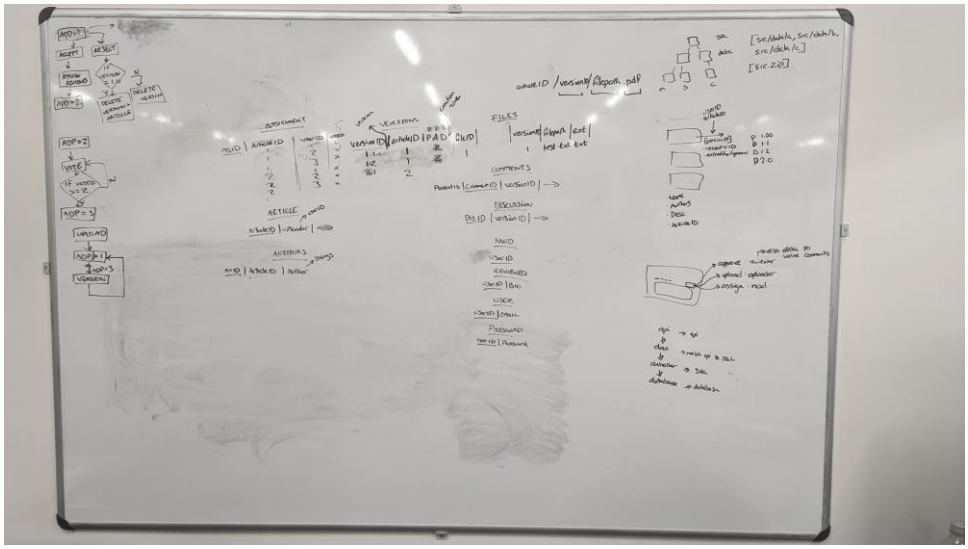
These were taken after a whiteboard brainstorming session in how we wanted the new backend to look and perform. This was then formalised by Jon in a flowchart (see Appendix 2) and API call sheet (see Appendix 3).



Whiteboard 1 – Moderator flow

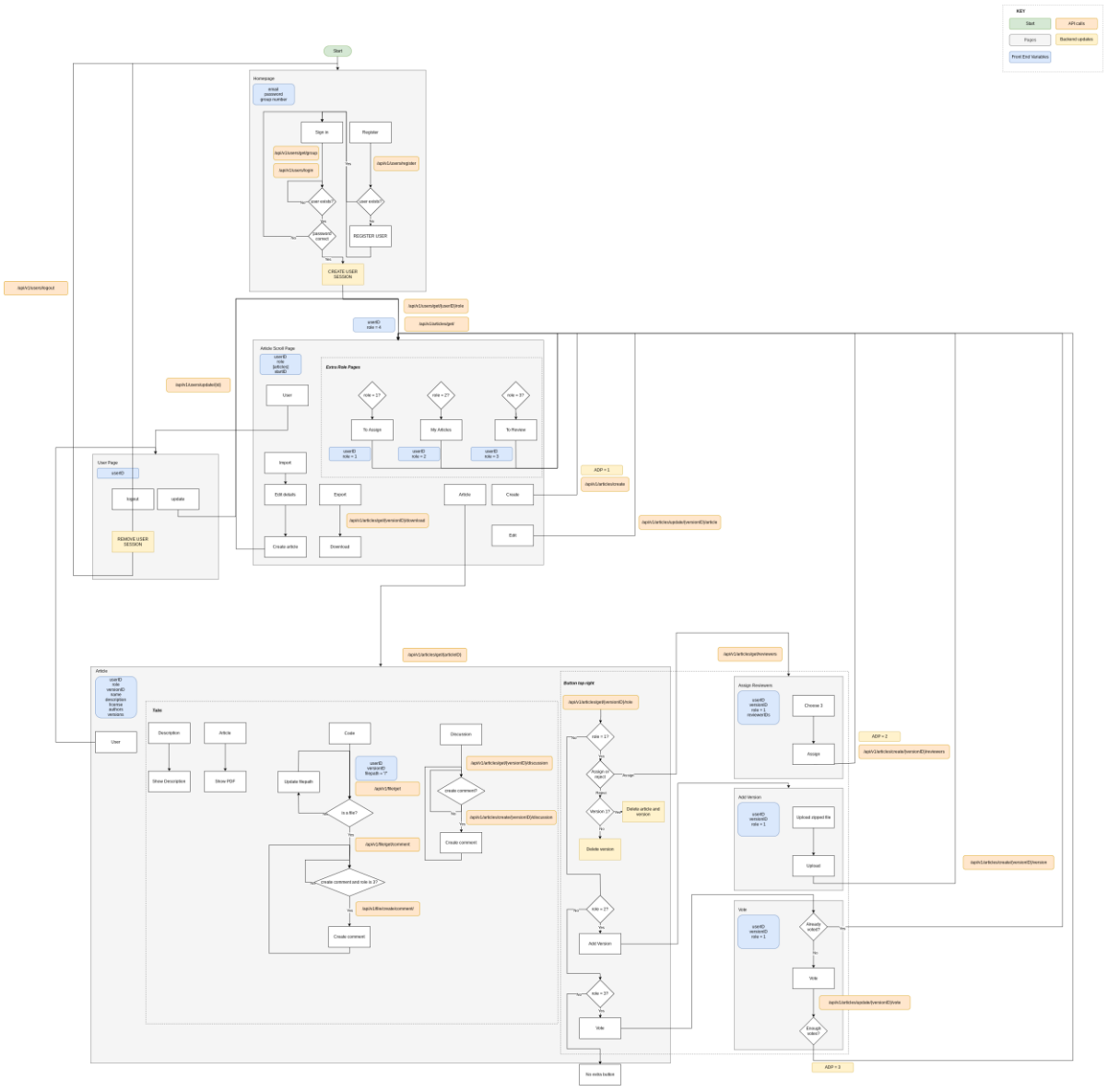


Whiteboard 2 – Reviewer flow



Whiteboard 3 – All flows (top right) plus table relationships (middle)

Appendix 2 – website flowchart



See (https://universityofstandrews907.sharepoint.com/:b:/r/sites/CS3099Team20/Shared%20Documents/General/CS3099%20Group%2020%20Flow%20Diagram.drawio.pdf?csf=1&web=1&e=erBi4p)

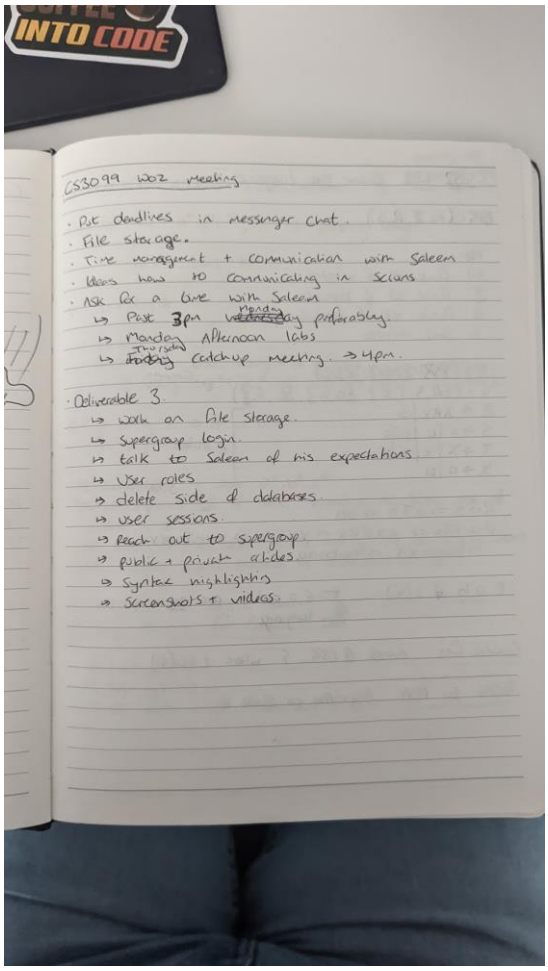
Appendix 3 – API call sheet

See (https://universityofstandrews907.sharepoint.com/:x:/r/sites/CS3099Team20/Shared%20Documents/General/Request%20URLs.xlsx?d=w83c38cd0b0ab4998985585b7f71808e7&csf=1&web=1&e=kiExrp)

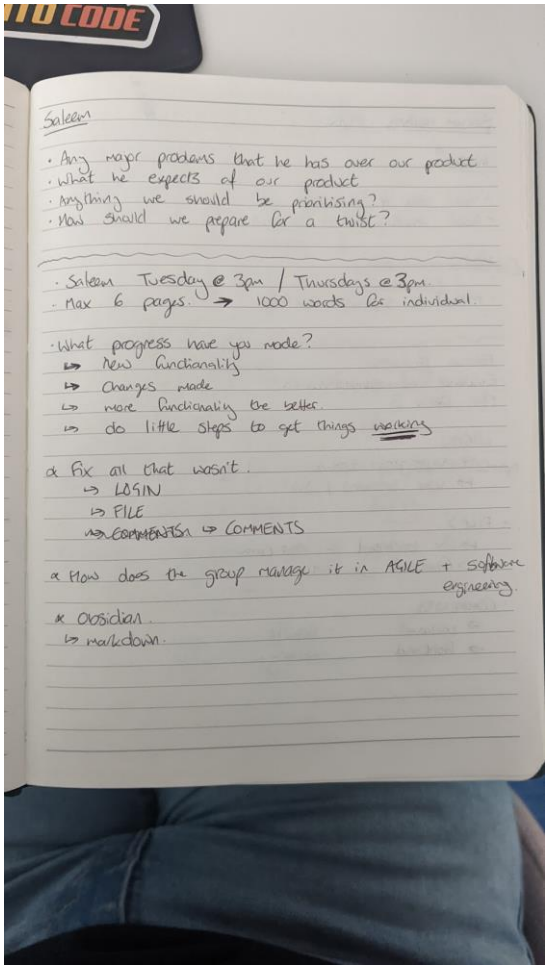
Appendix 4 – Database Query sheet

See (https://universityofstandrews907.sharepoint.com/:x:/r/sites/CS3099Team20/Shared%20Documents/General/CS3099%20DB%20Queries.xlsx?d=wd024f408baa14b5c8ce0b515a62de6f5&csf=1&web=1&e=hnLSGc)

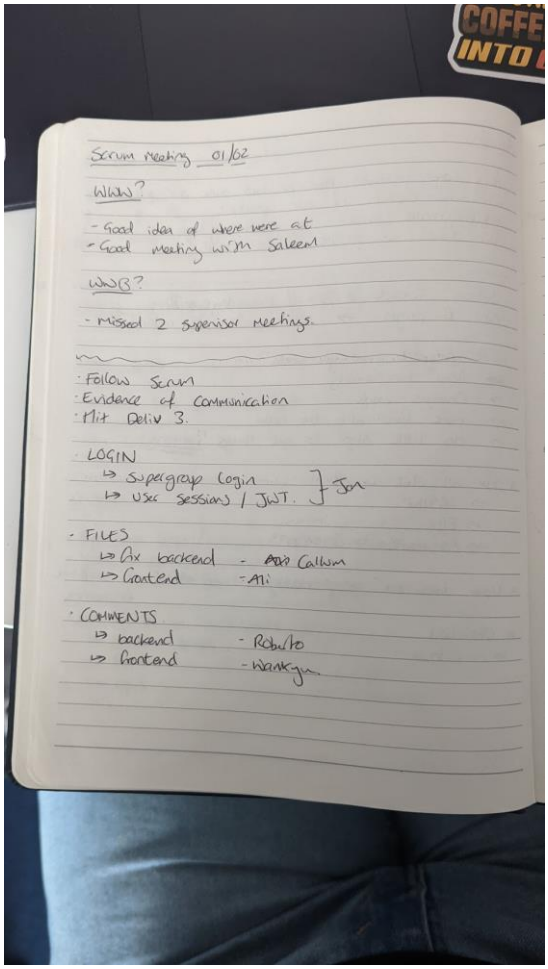
Appendix 5 – Meeting notes



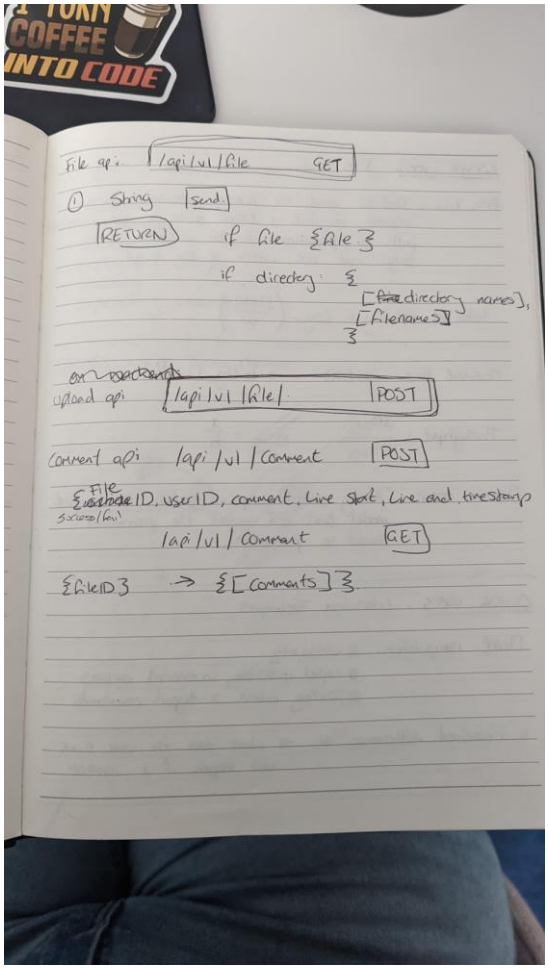
W02 Group Meeting



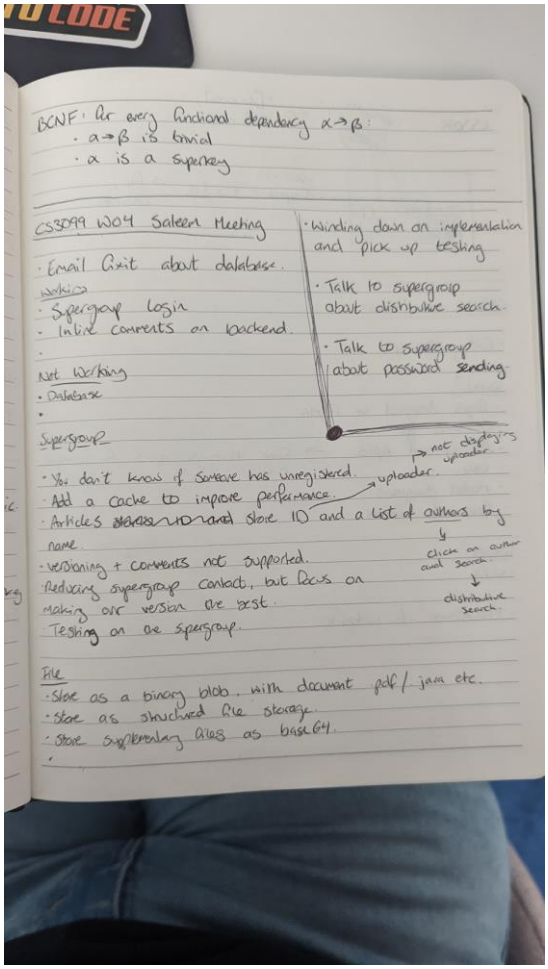
W03 Group Meeting



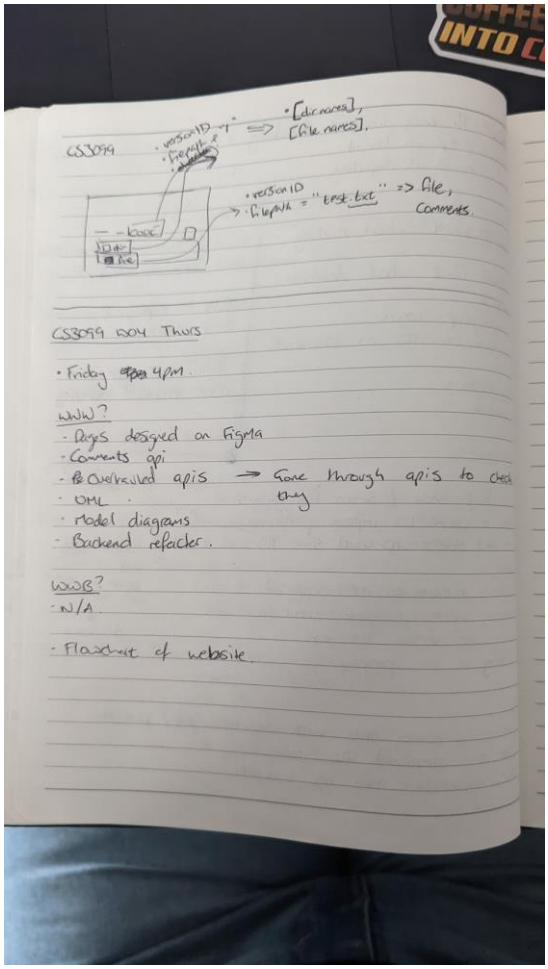
W03 Saleem Meeting



W03 Group meeting – this was mostly noted on whiteboards (see Appendix 1)



W04 Saleem Meeting



W04 Group Meeting

Agenda

Tuesday 15th February 2022

- What's been completed?
  - (From Jon – Updated API request spreadsheet and website flowchart is in teams)
  - Database creation, queries, methods
- What's yet to be completed? - is this before the deadline or before week 7?
  - (From Jon – new implementations of API to database queries)
  - APIs need to be linked to database, currently set up for testing
  - Index file system
  - User Store to be reconnected
  - Auto-reviewer voting
  - Download packaging for files (Index first)
  - API requirements that do not involve queries
- Supergroup new protocol
  - No changes to be adopted on our end
- APIs – do they work?
  - See point 2
- Discussion on what we're going to write in the report
  - Scrum - Roberto
  - Implemented functionality
    - Front-End - Wankyu
    - Back-End - Callum
  - Changes since deliverable 2
    - Front-End - Wankyu
    - Back-End - Callum
  - Supergroup - Jon
  - Outline for rest of semester - Aldiyar
- Queries
  - All bar new versions – need to discuss

The importance of a clear solution to the problem of passing passwords as plain text was emphasised and the supergroup will be pushed to make a decision far sooner than planned.

W05 Saleem Meeting 5