**UX3 – Continuous Improvement**

**PART A**

**1. Associate a README.TXT with the app outlining technical implementation matters**

This file can be found on the Git repository located at <https://github.com/rachelpac/letscollab>

**2. Write HTML help document and associated? icon covering common user interface elements in the UI**

Can be accessed by clicking the help icon or in the README file.

**3. Write a Roadmap going forward of the developments to the interface, including in development repository where code is stored**

Currently the application there are two versions of the application, running on

<https://rachelpac.com/> and <https://rachelpac.com/build/> . The latter is the react implementation of the application. In this implementation the initial build has been refactored to work in react. Functionality that has not yet been implement I would like to include going forward is withdrawing requests and editing collaborations and profiles.

The following business rules are yet to be implemented:

• A user can only have five collaborations started at a time

• A user can only have five pending collaboration requests

**4. Prepare (build) project for production environment**

The app is currently accessible on

<https://rachelpac.com/> and <https://rachelpac.com/build/>

**5. Why is it essential to implement the above (1-4?)**

It is important to outline technical implementation matters in a README file as it provides users with instructions on how to engage with the application you’ve built, as they would otherwise have a difficult time in installing and implementing the application. It is also important if you are collaborating with other developers so everyone is on the same page about how the app should function and how it is set up.

A HTML help document covering common user interface elements is important because it tells users how to successfully engage with your app if there is anything in the user interface that is not immediately clear.

A roadmap of developments to the interface going forward and a development repository where code is stored are important as they provide continuous improvement for the application. The improvements can be tracked using version control in the repository. In order to produce the best app possible, it is important to consider and plan for improvements for versions following the first implementation and to document these.

It is important to prepare a build for the production environment as this ensures your application is accessible to all users on a live server and in the build the files are minified for smoother deployment to production environments.

**6. What other types of documentation may be necessary for this project?**

A final report and a statement of changes could be written for this project that outlines what functionality has been implemented and any changes from the original project proposal, including confirmation that the client has signed off on these changes. This ensure the final project meets the minimum viable product and that future functionality yet to be implemented has been noted.

**PART B – Continuous Improvement**

**7. What portions of the development went particularly well**

I found implementing the User Interface using Materialize, HTML, CSS and JavaScript worked quite well. In this course I have developed my HTML, CSS and JavaScript skills so it was easy to build the interface with the help of Materialize and adding my own code where I needed to. Writing SQL statements for my API also worked quite well as this is another skill I’ve practiced.

**8. What was the most difficult to implement**

The most difficult part of implementation was learning to separate the front end and backend by sending JSON objects between the two, as this was a new skill it took some time to wrap my head around the separating the two and how to access and use the JSON objects I was transferring.

Learning react and how react components worked differently to Vanilla JavaScript was also another difficult part of implementation as this was also a new skill to learn.

**9. If you had the chance to do this again, what would you do differently?**

If I had to do this again, I would simplify the project, so it had less functionality for the first implementation as I learned these new skills. I believe I spent a lot of time refactoring code to provide more functionality, but I was utilising the same skills and not moving forward to other aspects of the project and learning new skills like React sooner.

**10. What parts of the implementation incomplete at this stage of delivery?**

At this stage in the React App the following is incomplete:

* Local storage to store the page the user is on

**11. Write and reflect on "Quality Assurance" how are you practising this?**

Quality assurance is practised by:

Analysing the functional and non-functional requirements of your application, making sure these are clear and concise and documenting these in the project plan to ensure the expected result is delivered and to avoid fixes later.

Undertaking test planning by researching resources such as Postman, Talend and curl test scripts to see how these strategies could be used to test the application

Undertaking test design by designing and developing written test cases using one of the above listed strategies that cover the project requirements. These cases should outline the conditions and steps needed to check that particular features work properly.

Undertaking test execution by executing the created cases and running the automated scripts.

**12. How much of the prototype UX1 remains in the final project?**

A majority of UX1 still remains in the final project, this was mostly due to installing Materialize via npm, importing it into react and initializing the components that included tabs, modals, side menus etc. The styling was accessed via CDN, so this was not an issue.

**13. Where has your project Object-Oriented programming implemented?**

Object-Orientated Programming (OOP) was implemented in the API when creating the database and session objects. It was also used in the React class Components.