IM3080 Design and Innovation Project (AY20xx/xx Semester x)
Individual Report

Name: Aftanza Nadhie Prabowo

Group No: 4

Project Title: Jam Stream

Contributions to the Project (1 page)

Handled most of the backend & technical project management, which includes:

- Ensuring the app runs with all its features.
- Made the proof-of-concept to make sure that app is feasible.
- Designed and implemented the project code & structure.
- Connected Spotify with the app.
    - Handled verification & authentication with Spotify.
        - Includes the use of industry standard OAuth2, with access tokens, refresh tokens, and the decision to use PKCE or not.
    - Handled the deep linking of Spotify authentication.
    - Made most of the API connecting the Spotify API to our app.
    - Handled other Spotify-related codes such as permissions, etc.
- Designed and implemented the database.
    - Implemented Firebase due to its *Real-Time Database* functionality.
    - Implemented database rules in the cloud.
    - Designed the database structure.
    - Made most of the API connecting the database.
    - Implemented a method to easily make more APIs.
- Handling GIT project, which includes task such as:
    - Merging and resolving merge errors.
    - Resolving pull requests.
    - Cleaning up the code.
    - Making sure the GIT repo is clean.
    - Making sure code is in the branch where it should be.
- Handled the production build.
- Assigning technical tasks & groupings to team members
    - Keeping track on who's doing what and how it's being done, along with Rui Han and Gabrielle.
    - Making sure their branch is running correctly.
    - Making sure they know what to do.
    - Give technical help if needed.

Reflection on Learning Outcome Attainment

Reflect on your experience during your project and the achievements you have relating to at least two of the points below:

(a) Engineering knowledge
(b) Problem Analysis
(c) Investigation
(d) Design/development of Solutions
(e) Modern Tool Usage
(f) The Engineer and Society
(g) Environment and Sustainability
(h) Ethics
(i) Individual and Team Work
(j) Communication
(k) Project Management and Finance
(l) Lifelong Learning

Point 1: **React Native & Expo, along with its pros and cons** State the area **(e)**

Our decision to make React Native our primary programming framework was largely due to some team members already having experience with React. We believed that the similarity between React and React Native would leverage the existing expertise of these team members, proving valuable for tasks such as code design, debugging, and teaching – and this proved to be the case. We used Expo in our builds to speed up the development process. However, despite the similarities between React and React Native, there are significant differences.

The most challenging difference lies in debugging. Debugging in React, and most other web development languages/frameworks, is easier because it provides clear indications of the problematic code. However, in React Native (specifically in Expo), the error messages tell us very little. It's usually a general error description, followed by the exact process on which the error occurs, often pointing to internal code that cannot be traced back to the program code. This makes debugging a much more tedious process.

The use of Expo did make development less strenuous, as creating a development build was as easy as running *npm run*. Expo works by running a live server build, and our phones or emulators can connect to the server. The server is typically located on our machine, and users can connect to it through LAN. However, during the earlier stages of development, most of the team couldn't connect to Expo using LAN, posing a problem. This might be caused by Expo using the first network adapter it sees, making people with multiple network adapters unable to connect due to their main network adapter for internet not being the first one. The suggested solution is to change the priority of network adaptors, but this solution proved fruitless. Our solution is to either run Expo in the cloud

and connect through a tunnel or run it locally and connect through an emulator or a physical device. The latter solution proves to be much more time-efficient, so we went with that.

Point 2: **Merging branches, untangling code, and the importance of good practice & modular code**
State the area **(k)**

Our approach to app production is quite straightforward; the weight of it is essentially “who handles what feature”. This localization of work was efficient in the early stages, but issues arose later in development when features became interconnected, requiring changes in other features.

For instance, the music player necessitates changes in the radio room page, but the room details page is concurrently being developed on another branch, utilizing a slightly different version of the radio room page. Moreover, there were instances where some features were using the wrong version of the branch.

These issues accumulated, and as I took on the responsibility of handling most of the merging throughout the development process, resolving merge conflicts became a tedious task of unraveling a code mess. The suboptimal user interface of GitHub's pull request resolver only added to the complexity.

While the ideal solution would be to synchronize our project at stricter intervals, our diverse and tight schedules made this solution less feasible. Instead, we chose to push through. Continuing in this manner allows us to address the high-workload progress as early as possible while implementing necessary features as soon as they're ready.

As the development process went on, I realized how valuable modular code and good practices are, especially for collaborative purposes.

Practices such as commenting on complex code, adhering to common coding conventions, and maintaining consistency across the entire app become increasingly crucial as the project advances, particularly when features begin to intersect. Issues, such as one file using "userID" while another opts for "userId," or choosing lists over objects, which has reliance on ordering, can introduce complexities that require careful attention. Additionally, adhering to consistent naming conventions for variables, branch names, commit messages, and function names contributes significantly to overall code clarity and ease of understanding.

Furthermore, emphasizing modular code proves to be an invaluable asset as the project unfolds. In hindsight, many of code that I've written could have been structured for reusability. This approach not only would have saved development time but also contributed to code clarity and facilitated easier debugging.

Please save the file in PDF and upload to the system.