GROUP 6 SECTION 2

MEMBERS:
Jake McAuley (1060842)
Dawoud Husain (1146816)
Adhyayan Bhandari(1135943)
Naza Anyaegbunam (1158144)
Ike Agbaje (1125093)

As part of the Final Project Sprint (Sprint 4), the team must complete a Project Post-Mortem. This will be similar to the Sprint Retrospectives that the team has been doing at the end of each Sprint, but will also reflect on the whole project It should reflect on the following questions:

What did the team do well?

Our team had several strengths that ultimately allowed us to come through any struggles we encountered and deliver a final working product:

Firstly, our team had excellent communication. Throughout the project, we made sure to be available through our communication platform by consistently checking our messages. Furthermore, we would update each other on our progress even outside of our weekly meetings and eventually developed the habit of creating short screen recordings as an efficient way to show what we were working on. This allowed our team to be more in sync, ensuring that we would have completed our sprint goals by the deadline.

Second was our willingness to help each other complete our issues. There were many instances throughout the project where one of us would be stuck but as a group, we would work through and fix the problem. A great example of this was when we would assign issues, as a group we would help the assignee brainstorm and combine our collective knowledge to determine how to approach the task most optimally. Another instance showing our group's willingness to help each other out was when we helped each other when we encountered a bug or issue while developing code. For instance, when one of our team members had a merge conflict that they were having difficulty resolving as a group during our meeting we worked through that problem so that they could meet their assigned task. In short, our willingness to help each other not only made sure that we were able to complete our individual assigned tasks, it also made sure that we were approaching problems the best way possible when developing code.

Finally, our adherence to standardized coding practices was key to our success in completing our objectives. For instance, when adding functionality to our app we would extensively test our feature using our predetermined definition of done criteria to ensure that added functionality was correct. In addition when we would merge our code, we would also take care that we would not

break previously implemented features through additional testing. Furthermore, we would add comments and follow standard naming conventions for file names, functions and variables, to ensure that our code was readable by other members.

What issues arose during the project? E.g.,

How well the team estimate its Sprint velocity each Sprint? Did this improve over time?
How well did the team coordinate project tasks across the team?
How effective were the team communication tools that were used?

Although in general things went very smoothly, during the development sprints our group encountered several issues, some acted as setbacks while others provided valuable learning experiences:

One issue that we encountered was estimating the issue weights when assigning tasks, especially for the first two development sprints. In one instance we were assigned the issue of having a friend requests feature, however, later in development we realized that there was a large amount of work to be done resulting in it becoming tech debt for the next sprint. This issue however naturally resolved itself over time as we gained a better understanding of our tech stack and were able to better estimate issue weights that would be feasible to accomplish during a two-week sprint.

Another issue that we encountered was our merge conflicts when adding our code to the development branch on GitLab. Since we were working on the same files asynchronously such as the front-end react components about the chat window, Gitlab would reject merge requests and we would manually swift each file and copy/paste our code. This process did not take a significant amount of necessary time but it was not negligible either. In the future, we would need to modularize our app better to prevent the scenario of having to work on the same files.

Finally, our group was not able to complete all user stories within our product backlog which was ultimately due to a lack of understanding of what type of features would be feasible to implement. Even with the investigation done in our project proposal and careful consideration, some of the user stories such as mobile support would require extreme refactoring from the original repository. In other instances such as server-side caching would require a very in-depth knowledge of web technologies to achieve, which we were not able to get to with our time restrictions.

What could be done differently to improve project planning and management in a future project?

Although our team delivered a working product and made significant progress throughout the term, there are several areas where our project planning and management could be improved in future endeavors. One key improvement would be more accurate scoping of features. At the outset, we underestimated the complexity and time requirements of certain user stories, which later proved to be beyond the scope of our current resources and knowledge. Conducting a more thorough technical feasibility assessment during the planning phase would help prevent this in the future.

Additionally, we realized the importance of early modular design. Many of the merge conflicts we faced were due to overlapping work on the same files, particularly front-end components. Had we structured the project in a more modular way from the beginning, it would have reduced the risk of conflict and enabled smoother parallel development. Another improvement would be implementing more frequent checkpoints mid-sprint to assess progress and reprioritize tasks based on current challenges or delays. While our weekly meetings were effective, these additional touchpoints could have improved adaptability.

That said, we made excellent use of GitLab throughout the project. We didn't just use it for issue tracking and merge requests, we also made full use of its advanced features like labels, due dates, burndown charts, and subtasks to organize our backlog, monitor progress, and stay aligned on sprint goals. These tools helped us stay accountable and transparent about who was working on what and how tasks were progressing. Overall, while our planning was effective, incorporating more proactive mid-sprint check-ins to reassess scope and redistribute workload would further enhance our project management moving forward.

How well did the team do at identifying and addressing issues in the Sprint Post-Mortems throughout the project? Give specific examples of the process, communication, and technical implementation improvements that occurred between Sprints during the term.

Throughout the project, our team made consistent use of Sprint Post-Mortems to reflect on our processes, communication, and technical decisions. These retrospectives became increasingly effective as the project progressed, allowing us to identify pain points and make meaningful improvements. Early on, we realized our sprint goals were sometimes too broad or overly ambitious. In response, we began breaking larger tasks into smaller, more manageable issues and clarified our "Definition of Done," leading to more focused and achievable sprints.

Our communication strategy also evolved thanks to insights gained during these reflections. Initially, we relied solely on text updates, but after noticing gaps in understanding, we introduced

short screen recordings to demonstrate progress and explain implementations. This significantly improved team alignment and reduced the need for lengthy debugging sessions during meetings.

From a technical standpoint, we addressed recurring merge conflicts by improving how we coordinated work. After discussing the issue in a post-mortem, we agreed to regularly pull from the main branch and communicate more clearly about which files we were working on. While this didn't eliminate conflicts completely, it reduced their frequency and improved our overall workflow.

A specific example of our learning curve occurred during Sprint 2 when we underestimated the scope of the friend request feature. This was flagged during our retrospective, prompting us to revise our estimation strategy and better account for complexity in Sprint 3. As a result, our sprint planning became more realistic, our workload more balanced, and our delivery more consistent by the end of the project.