🚀 Sprint Retrospective – Sprint 1

Sprint Goal: Fully working student profiles with login + basic availability management.

1. What Went Well 👍

- We were able to quickly establish our user stories and agree on our acceptance criteria during planning, even with vague initial requirements. This allowed us to start our stories and development more efficiently. We were able to finish our core stories ahead of time.

2. What Didn't Go Well 👎

- For this first sprint, we initially had a story dedicated to allowing a user to log in to the app since we were developing features of creating profiles and adding availability. However, the developer assigned to this story quickly raised concerns over this once the sprint started, specifically about how this could be misplaced. We all realized how this story was not important to our app in this stage, and it would actually put us behind on development. So we had to take this story off, and the developer was left with more command-line work that depended on other developers.

3. What We Learned 📚

- We need to divide and plan our tasks better between developers. Since we had ChatGPT generate the user stories, we overlooked this story and its scope, and a developer was left with less work compared to others.

4. Action Items ✅

- Focus on prioritizing core features that are part of explicit requirements to avoid scope creep, adjust backlog to eliminate these stories.
- Establish quicker check-ins to avoid any misunderstandings over each developer's user story.

🚀 Sprint Retrospective – Sprint 2

Sprint Goal: Enable students to create and confirm meetings

1. What Went Well 👍

- Our stories for this sprint were very clear, and each developer efficiently completed their stories without questions. We had active discussions on how to handle certain logic (such as generating student data within the app), which enabled us to use AI to generate our ideas clearly (instead of following whatever ChatGPT said).

2. What Didn't Go Well 👎

- During this sprint, our stories were heavily dependent on our command-line interface. Our main in this sprint became a hindrance since we had a lot of faulty logic we had not noticed before. We had to spend some time going through different scenarios with newly added data to correct this. This did end up taking away some development time, and it also led us to focus less on generating test cases for our specific features.

3. What We Learned 📚

- With each story and new feature, the developer must stay on top of updating the command-line menu to ensure it is easier to build on top of it later. In addition, developers should not leave testing too late into the sprint.

4. Action Items ✅

- Go over our command line interface as a team and note anyone's concerns still remaining after the sprint. Make sure developers work in main as they establish new features, but also generate test cases during development instead of after. Also, adjust the backlog to have developers' next stories build on their previous (reassign).

🚀 Sprint Retrospective – Sprint 3

Sprint Goal: Joining meetings, suggestions and filtering options, and a better menu-driven system.

1. What Went Well 👍

- In this last sprint, all developers completed their stories on time and actively thought through different user needs in their stories. We caught a lot more faulty logic in the main, and even condensed our menu to provide a better user experience.

2. What Didn't Go Well 👎

- During this sprint, there were a lot of different ways to go about some features, e.g where the suggestion feature would be placed (as its own option, or built into other options like browse study sessions). As developers, it makes sense that we will all have different solutions, but this confusion hindered our completion of these stories.

3. What We Learned 📚

- Some stories can be vague, so we need to establish more explicit acceptance criteria to eliminate alternative solutions. As a team, we will continue to have active discussions during standups to ensure every developer is on the same page.

4. Action Items ✅

- Review product backlog on a regular basis, and have each developer point out any stories that are too broad or have inadequate acceptance criteria.