Julia Thompsom
Andre De Santos
Molly Mckenzie

AI Usage Report – StudyBuddy Project (Group B: Agile)

As Group B, we developed the StudyBuddy scheduling app using an Agile (Scrum-inspired) development process. Our team organized the work into short, iterative sprints, which allowed us to adapt quickly to changes and continuously test our app. Throughout the project, we used ChatGPT as a simulated team assistant to support us with tasks such as gathering and clarifying requirements, exploring design options, generating sample code, and even assisting in the testing phase. This integration helped us accelerate our workflow by providing quick insights and reducing time spent on initial drafts or troubleshooting. However, while ChatGPT proved to be a valuable resource, it also required careful oversight and critical evaluation from the team to ensure accuracy, feasibility, and alignment with project goals. By balancing the efficiency offered by AI assistance with our own judgment and collaboration, we were able to make steady progress and complete our StudyBuddy app.

1. Sprint Planning & Backlog Creation

ChatGPT was first used to generate **user stories** and **acceptance criteria** for our backlog. For example:

- *"As a student, I want to add my availability so that classmates can schedule study sessions with me."*

- *Acceptance: Availability is stored with start/end time and visible when searched by classmates.*

AI also suggested how to break down larger features (profile creation, course enrollment, scheduling) into sprint-sized backlog items. We refined these with our own priorities, since the initial list was too ambitious for our sprint timeboxes.

2. Sprint Breakdown with User Stories, Backlogs, and Testing

Sprint 1 – Core Setup and Profiles (Thursday In class

- **User Stories:**
    - As a student, I want to create a profile with my name and student ID so that others can find me.
    - As a student, I want to enroll in courses so I can be matched with classmates.

- **Backlog Items:**
    - Implement StudentDAO and CourseDAO.
    - Build menu-driven input for profile creation.
    - Basic validation for duplicate IDs.
- **Testing:**
    - JUnit tests for StudentDAO (adding students, preventing duplicates).
    - CourseDAO tests for enrolling students and retrieving courses.
- **Outcome:** By the end of Sprint 1, we had a working prototype where students could register and enroll in courses with basic validation confirmed through tests.

Sprint 2 – Availability and Scheduling Foundations (Thursday night)

- **User Stories:**
    - As a student, I want to enter availability slots so classmates know when I am free.
    - As a student, I want to search for other students in my course to set up study sessions.
- **Backlog Items:**
    - Implement AvailabilityDAO with overlap checking.
    - Build CourseDAO query functions (fetch students by course).
    - Begin SessionDAO with basic scheduling.
- **Testing:**
    - AvailabilityDAO tests for inserting valid/invalid availability, overlap detection, and empty queries.
    - CourseDAO search tests to confirm students are correctly retrieved by course.
- **Outcome:** The system supported availability input and course-based search. Testing uncovered initial overlap logic bugs, which were fixed before moving forward.

Sprint 3 – Study Session Management & Full Integration (Friday)

- **User Stories:**
    - As a student, I want to create and cancel study sessions so I can coordinate effectively.
    - As a student, I want to view upcoming sessions so I can plan my week.
- **Backlog Items:**
    - Finalize SessionDAO with add/remove functions.
    - Implement queries for “view all sessions by student.”
    - Expand error handling and menu navigation.
- **Testing:**
    - SessionDAO tests for scheduling, canceling, and retrieving sessions.
    - End-to-end tests simulating a student creating a profile, enrolling, setting availability, and scheduling a session.
- **Outcome:** The app could create, cancel, and view study sessions. Testing identified and fixed bugs in availability conflicts and ID validation, leading to a stable final product.

2. Iterative Coding & Debugging

During implementation, ChatGPT provided code templates for **DAOs** (StudentDAO, CourseDAO, SessionDAO, AvailabilityDAO) and helped structure the **menu-driven main class**. In sprint reviews, we often asked it to suggest **refactorings or defensive programming checks** (e.g., validating student IDs, handling malformed dates).

We quickly realized we could not just copy-paste code AI often introduced methods that didn't exist in our schema. We had to adapt suggestions carefully. This mirrored real Agile practice: we adapted the "requirements" (AI output) as we encountered feedback in implementation.

3. Continuous Testing

Unlike Waterfall (which tests only at the end), we used **JUnit tests throughout each sprint**, many drafted by ChatGPT. These covered:

- CourseDAO: adding courses, fetching courses by student, invalid lookups.
- AvailabilityDAO: inserting valid/invalid availability, overlap scenarios, empty results.
- SessionDAO: scheduling and canceling sessions, retrieving by student.

Some generated tests failed at first because AI assumed database behaviors that were not true (e.g., auto-assigning IDs that weren't implemented). We had to adjust these tests to reflect actual DB logic. Despite that, AI helped us achieve much faster coverage than writing tests entirely from scratch.

4. Stand-ups and Retrospectives

We simulated daily stand-ups by summarizing progress and blockers. ChatGPT sometimes acted as a "rubber duck," suggesting debugging strategies when we hit errors (like missing imports or JUnit not being detected). During retrospectives, we noted that AI sped up brainstorming and boilerplate but introduced overhead in verifying correctness.

Reflection on AI Use

- **Strengths:** AI accelerated backlog creation, generated user stories, provided boilerplate DAO code, suggested error handling, and created test scaffolding. This made sprints more productive.
- **Weaknesses:** It occasionally hallucinated methods, produced overcomplicated code, or assumed DB behaviors. This required us to adapt constantly, but it fit well into Agile's

iterative corrections.

- **Overall:** ChatGPT was most useful as a “junior team member”. Fast at generating drafts, but always needing human review. In Agile, this was manageable because our process expected ongoing refinement.

Reflection:

Throughout working on this agile process development project we used ChatGPT to help us with many parts of the process. When we first started with the project we read through what was required and we decided how to go about building this project. This included what language and structure we wanted to use and what tools we were going to need while making this. We decided to use a SQL database and then use Java for the rest of the project. Thinking through the requirements and deciding how we wanted to build it, made it easier when we went to ask ChatGPT for help with the project. After deciding what we wanted to do, we submitted the project requirements to ChatGPT to give an overview of the project.

We started our first sprint by asking ChatGPT for help with user stories and acceptance criteria for our backlog. This went well and ChatGPT was very helpful, because it generated good responses that we could use in our project, or that we could slightly modify to fit the requirements better. The AI also gave us suggestions to help with our project, which was nice because it was helpful to determine where we should go in our project next and what would be helpful to add. We then moved on to some coding and debugging, during this process, ChatGPT helped provide code templates for us. This was helpful because it helped to give us a base structure for our project, which made it much easier to think things out as we continued with the process. Another helpful process was that we were able to ask AI to help us with suggestions of refactors. This was helpful because we could not just copy and paste code from the AI, but we were still able to use the AI for helpful suggestions. One struggle that we had is sometimes the AI would introduce strange suggestions that did not fully relate to our project. This showed us that we needed to be very specific when asking ChatGPT and sometimes change our original

questions to get a response that was helpful. After this we moved into the testing process, we were able to use JUnit tests that ChatGPT helped to draft through each of our sprints. We also had some trouble here with the tests that the AI drafted, many of them failed at first because of some parts that AI assumed we had but we did not. When this happened we just needed to adjust the code, or just ask AI more specific questions.

Overall our experience with ChatGPT was overall mostly helpful, but we definitely did encounter some struggles. A lot of times when asking ChatGPT about our project the AI would make assumptions and add parts to code that just did not make much sense with our project. What we learned from this is that you need to be very specific when asking AI to help while working on a project. The more specific the better, because what we discovered is when we asked more simple questions, even if it related to something we asked before, AI would tend to get side tracked and add things that messed up our program. So an important part when working with AI is to be very specific.