Bi-Weekly Status Report        Name:\_\_Andrew Swayze\_        Project Title: Community Board  Score:\_100\_

Each week you submit a status report to indicate the progress you're making and list your goals for the upcoming week. Students working in teams must individually submit these status reports to discuss their specific contributions and plans.

The purpose of this report is to help you gauge your own progress and to help the instructor in mentoring.

For status reports, create a template text file with the following sections:

Part one: Project Tasking

* Scrum Status
  + What did you accomplish? (Actually stories with reference numbers from GitHub Projects).
    - 30, 31, 35, 39 (each of these are the overarching epic stories and not the subtasks)
  + What are you planning on doing? (Actually stories with reference numbers from GitHub Projects).
    - We haven’t set the tasks and issues yet, but I’m on the front-end design team
  + What challenges are you facing?
    - Mostly I’m very overwhelmed right now. I got behind because of travelling due to a death in the family and getting caught up has been a lot.

Part two: Project Artifacts

**Meeting Notes**

To-Do’s:

1. Examine the current status of the design document
2. Make sure assignments are clear for development teams
3. Talk about any issues moving forward

Principal Discussion:

1. Break down work into manageable pieces
2. Communication between team members
3. Use UML diagrams for architecture design

Future tasks:

1. Adopt the TDD
2. Further refine the design document
3. Organize subsequent meetings to monitor advancements and tackle any obstacles

Part three: Project Metrics

* Time Tracking
  + This Week
    - Hours worked this week: 4
    - Total number of tasks completed: 0
    - Total number of story points completed: 0
    - Current Story Point Velocity: (Story points/number of tasks): 0
  + Total to Date
    - Total hours on project: 24
    - Total number of story points completed: 34
    - Total number of tasks: 10
    - Story Point Velocity: (Story points/number of tasks) 3.4

**Research Test-Driven Design and Unit Testing**

Test-Driven Development (TDD) seems like kind of a weird thing to me, or it did at first glance. It's all about writing tests before your actual code, which might seem backward at first but has some great benefits. Firstly, it makes your code cleaner and more focused because you're only writing code to pass tests. This means less room for unnecessary features or complexity. It also means you end up with a bunch of tests that document what your code is supposed to do, which is super helpful for anyone (including future you) trying to understand your work later on.

The process is pretty straightforward. You start by writing a test for a new feature, which will fail because the feature isn't there yet. Then, you write the smallest amount of code needed to pass the test. Finally, you clean up your code, making sure it's as good as it can be while keeping all tests passing. This cycle helps keep your codebase healthy and your mind at ease because you know if something breaks, you'll catch it quickly.

For unit testing, which is a big part of TDD, there are tons of tools out there, and the best one depends on what language you're using. Python programmers, including students at BYU-I at least when I was taking early classes, can use pytest, JavaScript developers might pick Jest or something similar. The tool that was mentioned in the video is called Katolon, which I had also used previously, but definitely need to practice. These tools help you write and run your tests, making the TDD process smoother.

In summary, TDD seems to prioritize quality, simplicity, and maintainability from the start. I’m excited to get started in using it on this project.