Michael Gibson Acorn Nursery

Overall Status: Behind Schedule, Back-end team is starting to get a footing with leadership. We should begin working, focusing on the design document the beginning of next week.

Summary of Project Tasking: Update personal development folder with reflections and statuses.

* [GitHub/mikegibson66/cse490R](https://github.com/mikegibson66/CSE490R)

TDD Training Notes:

Four principles associated with TDD and Unit Testing:

* Test-first programming: Tests are written before the code is implemented.
* Red-green-refactor: A cycle of three steps:
  1. Write a failing test case
  2. Write the minimal amount of code to pass the test case
  3. Refactor the code without changing its functionality
* Small, isolated unit tests: Unit tests should test a single unit of functionality in isolation.
* Fast feedback: Tests should be run frequently and automatically to provide fast feedback to the developer.

Advantages of TDD for developers:

* Improved code quality: TDD encourages developers to write clean, well-designed code that is easy to test.
* Reduced bugs: TDD helps to catch bugs early in the development process, before they become more difficult and expensive to fix.
* Increased confidence: TDD gives developers confidence that their code is working as expected.
* Faster development: TDD can help developers to develop code more quickly and efficiently.

Steps to implement TDD:

1. Write a failing test case: This test case should describe the desired behavior of the code that you are going to write.
2. Write the minimal amount of code to pass the test case: This code should not be perfect, but it should be enough to pass the test case.
3. Refactor the code without changing its functionality: This is an opportunity to improve the design and quality of the code.
4. Repeat steps 1-3 until all of the desired functionality has been implemented.

Tools and software for unit testing:

* Python: unittest, PyTest
* Java: JUnit
* JavaScript: Jest, Mocha
* Ruby: RSpec
* C++: Google Test, Catch2

These are just a few examples of popular unit testing tools and software. There are many other options available, so you can choose the one that best suits your needs and preferences.

Additional benefits of TDD:

* Living documentation: Unit tests can serve as living documentation, describing what the code is supposed to do and how it works.
* Improved communication: Unit tests can help developers to communicate their ideas more effectively.
* Increased maintainability: TDD-driven code is typically more maintainable, because it is well-tested and well-designed.

Overall, TDD is a powerful software development practice that can help developers to write better code, faster.

Number of Hours:

* Number of Hours worked this week (expectation: 7-9 hours): 9 hours
* Total Number of Hours this semester (anticipated 120/semester): 126 hours

Accomplishments:

* We began work as a backend team

*Performance*:

* Last Week’s Performance Average (Actual Total Points/ Total Hours): 9
* This Week’s Performance (Actual Points/Hours): 9

*Challenges encountered and resolutions found*:

* I am unavailable to meet with the rest of the team since they meet during the day while I am at work. If I can arrange my schedule, I will attend. Otherwise, the team will record the meeting so I may attend asynchronously.

Plans/Goals/Tasking for the coming week/sprint:

* I plan to complete the two stories that I am currently assigned to by the end of the week.
* I also plan to start working on the story that I will be assigned for the next sprint.

*Reflection goals and status*:

* I am happy with my progress so far, but I want to make sure that I continue to be consistent with my work.
* I am also going to make an effort to communicate more with my team if I have any questions or concerns.

*Follow-up/Reporting*:

* I will ask my team to help me follow through with my goals.
* I will also work with my team on the weekly status reports.