Weekly Progress Report 1

Austin Snyder, Ethan Roppel, Isaac Darlington, Ken Alleyne, Wyatt Wooden

# This Week

## Austin Snyder

I worked on implementing the User, UserStats, and SkillLevels classes this week. These classes will work together to store most of the information about a user. User contains an instance of UserStats. UserStats has an instance of SkillLevels, which holds the levels for the 5 areas of “skills” we have. Made some changes to where some fields belonged, since we used to think we would want just a User/UserStats, but it makes sense to break userStats into smaller parts. It became more modular, and I believe there are potential applications of the SkillLevels class in relation to Tasks, but that will have to be something we do at the end.

I look forward to working on getting data from the database and building the objects with it that will be used during the execution of the program. I also look forward to working on storing our objects in the database. Next week, I will try to finalize the DB schema with everyone and send out a build file so everyone can use the db locally.

## Ethan Roppel

## Isaac Darlington

## Ken Alleyne

My assignment this week was to create the class for tasks. This class encapsulates the following private fields:

* Task ID – a unique identification number for each task.
* Task Category – a category corresponding to user skills.
* Task Name – the name of the task.

Additionally, the class contains a single constructor that takes arguments for each of these fields; and it contains getters, setters, and a toString method.

Furthermore, we determined that the task categories would be best described using an Enum; so I also created an Enum class called TaskCategory which contains the following constants:

* Intelligence (INT)
* Strength (STR)
* Endurance (END)
* Wisdom (WIS)
* Vitality (VIT)

## Wyatt Wooden

# Next Week

We will discuss the implementation of the database connection and functionality within the application. We will meet to discuss how the data should flow from the frontend to the backend, including what each dbModel function should return.