# 

# 

# 

# 

# CENG / ELEC / SENG 499

# **Avatari - FitBit Game**

# Progress Report #2

Denholm Scrimshaw - V00759740

Tyler Potter - V00780941

Charlotte Fedderly - V00729142

Evan Hildebrandt - V00778723

Darren Prince - V00320999

Supervisor: Jens Weber

Group Number: 8

Date: June 30th, 2016

## Updated **Milestones**

|  |  |  |
| --- | --- | --- |
| Date (week of) | Tasks | Progress |
| May 30 | Progress Report 1, Milestones 1, Work Log 1, obtain FitBit device, EC2 instance, preliminary UI mockup, start building the menus in Unity | * In depth discussion to determine gameplay flow as well as what menus/screens the application will need. * Character(Avatar) functionality is worked out. * Application has been separated into 5 areas, 1 for each member to be in charge of. * Menus are still being finalized |
| June 6 | Model database, finalize UI and begin implementing mobile application | * Database has been setup on AWS * A digital mockup of our UI has been created with all screens, menus and buttons. * The application framework has been setup using Unity on AWS. |
| June 13 | Milestones 2, Work Log 2, get information from FitBit device and store it in database | * A simple script has been written to hit the Fitbit API’s and return data. * Fitbit device information is reporting to database in AWS. |
| June 20 | Integrate the FitBit metrics into the mobile application |  |
| June 27 | Progress Report 2, Milestones 3, Work Log 3, find/create artwork, implement levelling up and rewards for achieving goals | * Found art work for inventory items. |
| July 4 | Continue work on the application, get social media integration, implement “battles” between users |  |
| July 11 | Milestones 4, Work Log 4, finish the application, start testing final product, prepare public presentation |  |
| July 18 | Create pamphlets for presentation, public presentation, create project website |  |
| July 25 | Final Report, Milestones 5, Work Log 5, project website completed |  |

# Current Progress

The current progress that has been made on the FitBit Avatari game can be split into four sections: decisions on game aspects, server set-up, authentication, and UI implementation.

## Gameplay

Few changes have been made to the initial gameplay design, but details have been filled in regarding various aspects of the stats and game progression. For example, finalizing character stats as health, agility, strength, and defense. Health was determined to be a viable source of user retention, as it is generally something that has to be replenished, and in our case can force the user to adhere to a regeneration/fight cycle. Boss battles will have probabilities involved, and are likely to damage the user’s character, therefore players will be enticed to return at least a day later to finish what they started.

Defense is a related but unique aspect of a user Tari. It helps to reduce the number of attempts necessary for defeating a boss by providing a secondary health buffer. This means it is technically an optional stat, since a player can still succeed by playing chance. An optional stat was offered on purpose as a means for users to take advantage of the optional FitBit features, namely tracking sleep, water and food consumption. Though optional, it is obviously desired as a means to accelerate gameplay and reduce the regeneration cycle. Therefore it aims to encourage a healthier lifestyle and the full exploration of the Fitbit without forcing a user to perform mandatory logs of food and water.

Next we needed to consider how to accommodate for variations in user activities. For instance if a user was to exclusively walk, they have to achieve progress but shouldn’t be able to do so with ease. To facilitate this we added strength and agility. Strength is primarily gained through walking and the completion of daily goals, and is directly related to the amount of attack damage possible. Agility however is gained through additional challenges and alternative forms of exercise (such as running or stairs). The more agility you have, the better chance you have of hitting (or not missing) enemies. Bosses themselves will have their own stats, and in varying ‘specialities’. In other words, some bosses may be easier to fight if you have a lot of agility (and may not require much strength) while others may require a balanced combination. In any case, this range of stats encourages the full use of the game while still enabling light FitBit users.

Bosses themselves are the primary reward system. Upon defeat, a user has the ability to adopt the boss character as their own Tari. The completion of challenges and the collection of ‘experience’ simply enables you to unlock and survive boss battles. We felt this wasn’t enough as a means of gameplay or encouragement, therefore we also added character items. Character items helps to boost stats and may enable a player to beat a boss with an unfortunate statistical match-up. Items have a random chance to drop while conducting exercise. The chance continues to grow as you reach your average and best workouts. This helps to encourage people going for that half hour run to finish the last five minutes. Alternative backgrounds for your Tari character can also be collected in this way.

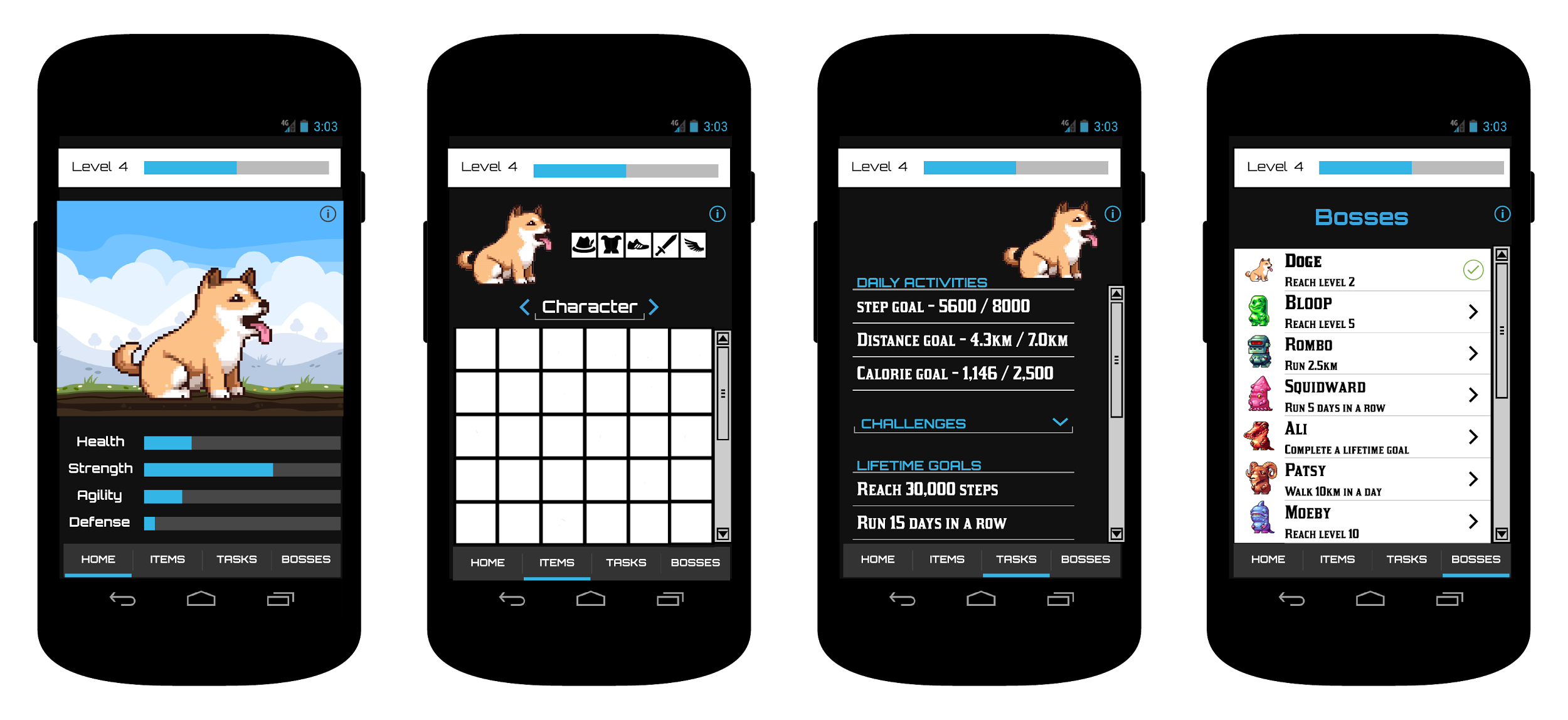


Figure 1: User Interface Prototype

## Server Implementation

There have been a few changes and improvements to the server implementation. Rather than using a mySQL instance spun up on the EC2 instance we have switched to RDS. Amazon RDS is a standalone server for relational databases. This will allow us to scale up more easily by adding additional EC2 instances. Another change came from switching frameworks from Silex to Symfony. Symfony is a much more robust platform with better community and documentation support at a slight hit to performance and code size.

Aside from the platform changes progress was made on the entity relationships as well as parts of the API dealing with user management.

There were a couple of challenges faced both in making these changes and progressing on the project as a whole. Setting up RDS was not a complicated process but as we do not have a lot of experience with RDS, the rule/firewall changes necessary to allow the RDS to talk to EC2 took a while to figure out. Switching to Symfony also caused a problem where installing its requirements would cause our micro EC2 instance to run out of ram and crash. This was because the micro instance does not have a swap file enabled by default and has only 1 gigabyte of ram so enabling the swap file fixed the issue. Finally less progress was made on the API than originally scheduled. This is in part due to the Fitbit Authentication not being complete as well as the changes and challenges outlined above taking away from the time scheduled for API work.

## Authentication

Fitbit uses OAuth 2.0 for user authentication and API authentication, which allows the user to sign into our application without exposing their Fitbit password to our application. This works by redirecting the user to the Fitbit sign-in screen in their Android phone’s default browser. For the Avatari application, the user has to sign into their Fitbit account so that their activity can be tracked.

There were some challenges involved with setting up user authentication due to the technologies we selected. While Unity greatly simplifies the implementation of gameplay and UI elements, there was no Unity-specific way to get the return code from the browser used to complete OAuth sign-in. To further complicate matters, Java is the official language for Android development, whereas Unity’s scripting language is C#. To solve this issue, the code for retrieving the return code for authentication was written in Java, and imported into the Unity project as a plugin in .jar format. This method proved difficult to test, however, as it was often hard to tell whether bugs were issues in the C# scripts or the Java plugin. Another challenge was that there are very few resources online for helping with the specific combination of technologies that we chose for the app, so it was unlikely to find an example to follow.

## User Interface Implementation

For the user interface, the implementation of the UI mockups is undergoing progress. Currently the screens for the bosses and inventory are well under way, along with a toolbar that will be used to navigate scenes, and a progress bar for the user's level and experience. The implementations have been completed using Unity’s UI components and rendering engine, which have been optimized for use on Android mobile devices.

Various challenges were faced while implementing this UI, most of which were a result of exposure and learning Unity’s UI interface. Figuring out how to build UI that scales correctly to Android mobile devices was more difficult than expected. Also challenging was finding the necessary information for our project. Since our project is largely UI based, we had to more heavily utilize the Unity ‘canvas’, and many of the tutorials looked at other tools and aspects of game design. Probing various tutorials for what we needed was time consuming and at times contradictory.