Real Fantasy Adventure (Working Title)
Software Engineering Large Practical 14/15

Paul Scherer (s1206798)

October 15, 2014

Contents

1 Overview

Real Fantasy Adventure is a web application that takes a user's real world activities (exercising, working, studying) and changes his/her *Avatar* inside a fantasy world within the web application. Users perform activities towards goals set in the form of *Quests*, the user will achieve points by completing these quests and can compete against other users who use the web application facilitating competitive behaviour. The categories of activities the user may perform are various: academic, athletic/wellness and professional, therefore allowing users to log their achievements in various ways, and can be ranked individually in each available category as well as overall.

The majority of the web application will be created with *Python 2.7* utilizing the *Django Web Framework 1.7* for the back end handling of data and database management. Development will incremental fashion to build minimum viable product at each stage ready for release, and several possible extensions to the base application are explored in this document.

2 Description of the Application

2.1 Overview for Users: What it does

The web application is a turn-based *game* that allows the *player* to log what he/she has done throughout the day to improve their in-game *avatar*. The purpose is to increase the power of each avatar, much like the early *Tamagotchi* games with some additional features. Each turn is anologous to a day in real life; the player would log his activities and the in-game avatar would reflect this by mimicing the actions of the player in its fantasy world, hopefully improving the stats in relevant properties (this will be handled by luck). Activities the player performs in real life can be logged under 3 major categories: Academic, Athletic/Wellness, and Professional; this allows the player to log information beyond those typically seen in other real-life trackers that tend to focus on athletic deeds such as the *FitBit App* or *Zombies Run!* and allows setting of goals (called *Quests* in the application) in these categories.

Logging of activities in the categories will be rudimentary and intuitive – Studying for 3 hours would increase the stat of a subcategory under Academic, and helping out in a restaurant would boost a Professional stat – rudimentary because the logging would be performed by trusting the player to be honest about the values they input in the name of roleplay[1].

While the exact mechanics behind the changing of the stats and the resulting aesthetic effects happening on the screen are unclear, once the player logs his/her actions several things happen on screen. The in-game

[1]If time permits alternative methods will be explored, outlined in section 3

avatar is an adventurer and travelling across a vast land (possibly with a band of dependent party members, making setting of quests easier with motivation to complete them), and therefore logging a distance traveled by foot input by the player would move the avatar across a map [2]. Additionally the increases in stats are based on probability, thus for example running for an hour would have a 80% chance of increasing the stamina stat under Athletic/Wellness while walking for the same hour would have a 30% chance of increasing the same stat.

Each increase in stat counts as a point both under one of the three main categories and overall. Players can compare their avatars with others to see who is the best adventurer in the land under any of the categories (Academic, Athletic, Professional, Overall) and if time permits further filters will be added to allow cases where the user would like to see who has the best avatar in Scotland.

2.2 Technical Overview

This is concerned with a technical overview of the tools and the process I intend to use to realise the basic requirements set out above. Since this is my first time tackling a project of this size and working in web development I estimate that much time will be spent on learning and understanding these technologies (web development, communications, networks) and expect to greatly exceed the recommended 100 hours, thus to make the task easier I have opted to use a *"heavier"* web framework with a built in webserver and ORM database management system.

List of Languages, Frameworks, and DBMS [3]

- Python 2.7
- Django 1.7
- HTML & CSS (?)
- SQLite3 DBS

2.2.1 Python 2.7

Python was chosen as the language of choice for development as it is a language I am familiar with and want to improve in preparation for next year. Additionally since the main language of development in Django also happens to be in Python, the decision was easier because it would significantly

[2]Ideally across a visual map, however this will only be a feature if time permits

[3]I am still considering other libraries and frameworks but these are the ones I have found most appropiate thus far but would appreciate feedback on my reasoning for choosing each of these, ie. are these appropiate for the task?

decrease integration problems if development stuck to one language. Python 2.7 was chosen over the newer 3 as currently there are still more libraries for 2.7 which might be useful for the application.

2.2.2 Django 1.7

The web framework Django was mainly chosen because I have very little experience in web application creation or web development altogether, it has a very active community, is well documented, and perforated with many tutorials as well. Additionally as the project does not have to be deployed to a live server I find Django's integrated webserver extremely useful in quickly building and seeing the results locally, without having to learn how to manage a full blown webserver such as Apache2 which would require much more research and learning on how communications, protocols, and ports work (this I leave for extra study, and Django Applications work on Apache as well if I decide to pursue this later). Currently as I plan the possibilities of implementing the avatars and metrics I find the concept of Django's ORM database focus very attractive as each avatar can then be treated as full objects with no restrictions on the datatypes of the attributes which will eventually fill the attribute entries in an avatar table. While most of the attributes will likely not involve complex datatypes, the comfort of knowing that I wont have to worry about converting object attribute datatypes for input/extraction in traditional database tables is a large advantage I see in Django over other frameworks, in the context of my application.

2.2.3 HTML & CSS (Maybe some JS?)

HTML and CSS for layout and display of website elements. I don't particularly know much of either and am not particularly gifted aesthetically and as such may rely on HTML and CSS collections (possibly even JS) from front-end frameworks such as *BootStrap*, however I currently find that much of BootStraps elements wouldn't fit the topic of fantasy well. I'd like to recreate the classic *RuneScape 1* look. However the look of the website/application is less of a concern until the back end is completed and I will only use these two languages to format my output text from the backend until analysis of the non-functional requirements has been completed.

2.2.4 SQLite3

The decision to use SQLite3 is currently a very cautious and uninformed one as I have never worked on application that expressly had to change values or receive values from a database, much less make my own decisions on using a particular database management system. Thus I must confess that I don't know what to look out for despite doing some research. The decision to use SQLite3 is that I am currently using it in conjunction with a Django tutorial to understand how web applications talk to DBMSs and I know ORM works with SQLite3 having done this. All the database should do in my imagining of the project is to allow the web application to input and extract attributes of players' avatars for calculation and ranking so I do not see any drawback that could arise from using SQLite3.

3 Development Schedule

Development of the project is intended to proceed in very small increments, however, certain requirements for SELP have to be met and thus milestones were set early to meet these requirements, with each milestone being represented by a stage where it would be a minimum viable project that could be submitted if catastrophe occurs. I intend to use ticketing system (or issue tracker) in order to keep track of bugs and things I need to take care of, might seem overkill for a sole developer project but I would like to keep a backlog of issues that have appeared and replicate these by reading the descriptions from the ticket at a later date.

Some extra stages have been noted to suggest improvements that can be made to the web application, and while each extra stage is a self contained improvement that can be made, I find that this ordering would improve the user experience most significantly if this order is followed (Would this coincide with your opinion?).

3.1 Development Stage 0: Basic Registration & Log-in

Intent Set up basic web page where a user can register, have input user details be placed inside a database table, and log-in as this user.

Requirement Be able to log-in to website with correct credentials (will show confirmation) and raise error otherwise.

Difficulty Learning how to use ORM databases for basic refistration and credential authentication

Questions How will user information be stored?

3.2 Development Stage 1: Basic Access to Avatar Page

Intent After Log-in the player can look at the avatar page which would display the current stats of each attribute

Requirement Create the profile/avatar page, be able to view them.

Difficulty Displaying values and making calls to database for data / see if SQLite is the appropiate if more users

Questions

3.3 Development Stage 2: Game Mechanics

Intent Program the various methods for updating avatar attributes, and allow user to input data. Flesh out and solidify/document game mechanics

Requirement User input correctly updates database entries

Difficulty Writing a lot of code that is similar yet different and seeing if this can be generalized

Questions None

3.4 Development Stage 3: Basic Ranking & Scoreboard

Intent See if point system works and see if another distribution can be made, implement point system, have avatars ranked by these points

Requirement Allow users to see their ranking based on 4 basic categories, and look at scoreboard that is updated every day[4]

Difficulty Preparing a fair point distribution

Questions At this stage are all basic requirements for SELP covered? I will go beyond this but at this stage the web application does have a notion of a user and a competitive way to rank them.

3.5 Development Stage 4 (Extra): Quests

Intent

Requirement

Difficulty

Questions

3.6 Development Stage 5 (Extra): More Advanced Rankings

Intent

Requirement

Difficulty

Questions

[4]not necessary to update so frequently until more people join, and as each turn is equal to a day

3.7 Development Stage 6 (Extra): Non-Functional Requirement Analysis and Redesign

Intent

Requirement

Difficulty

Questions

3.8 Development Stage 7 (Extra): Advanced Metrics and Diagrams

Intent

Requirement

Difficulty

Questions

More could be added but in the interest of keeping options open as requirements and needs change no more extra features added

4 Final Thoughts & Questions

I find my sufficiently difficult to implement given my lack of experience and that it would reach the recommended 100 hours of work easily, yet I find my project interesting enough that this pursuit would be worthwhile. In this section I wanted to focus on some questions explicitly.

Due to the lack of knowledge I have several questions about best practices in keeping a repository or organising my files:

- Would it be ok to leave my database tables in the repository until release or submission?
- I might change some of my requirements drastically for some additional features I would like to add, in this case may I contact you and have this reflect in the final grading instead of being downgraded for not following some of my proposal plans?