Application Development Report

Kieran Warren  
University of Derby  
kieranwarrenuk@yahoo.co.uk

# Installation & Setup

## Server

Requirements:   
Python 3.5  
Internet Connection  
SQLite DB Browser (Optional)

With Python installed you must type pip install apscheduler into a console and press enter. Wait for the package to install and then do the same with pip install webhose. These are packages required for the server to operate correctly.

Providing the software above is installed, setup of the server is simple. First extract all of the files from the .zip file, then navigate to the extracted file and run AppDevServer.py in Python 3.5 (It may work on previous versions but is untested). From this stage, no user input is required. The server should automatically start listening for new connections and will contact the Webhose API servers to retrieve the latest news. Please allow a few seconds for this to complete. Within the console you will receive some notifications to confirm it is updated. The server will continue to listen for new connections and the settings will be updated with the latest news regularly. SQLite DB Browser can also be downloaded to easily view the database named “server.db” inside the server directory.

## Client

Requirements:   
.NET Framework

Once the server is started, you can then open up the client on the same computer. Currently, the server is designed to only run within the local network and has only been tested on the same computer. However multiple instances of the client can still be connected to the server. Upon opening the client, you will be asked to login or register. You can use Username: us and Password: pw if you wish to login straight away, but you can choose to register instead. NOTE: password is not hidden or encrypted, do not use a real password!

Once logged in you will be presented with 3 options. You can either play singleplayer with the servers’ current settings, singleplayer with a previous setting or multiplayer, which also uses the servers’ current settings. Once started, both of the singleplayer options will not be in contact with the server. Multiplayer will of course be kept in regular contact with the server. The server will handle all of the positions received every tick by each of the players, compile them and send all of them back to each client to update.

# Technologies & Development

I originally specified that I would use Javascript to develop the client side application, however with little experience in either Javascript, or my chosen game engine Cocos2D I felt it necessary to move to an alternative solution that I would feel more comfortable with as I felt that Javascript, a new game engine and everything else would be too much to pick up within a short time frame. Instead I chose C# which I am comfortable with the language syntax but would still get the opportunity to learn a number of new things including WPF, Sockets in C#, multiplayer server development, using a web API and much more.

However, there was a big downside to using C#. The lack of any game frameworks or engines made it difficult to produce detailed games. Instead the games were kept simple. This was generally the plan, as I wanted the final result to be more of an example of what could potentially be extended from the API and the news instead of just high quality games. I already have a good amount of experience in game development and so I wanted to concentrate on other areas that I am new to or have little experience in.

My choices between APIs were quite restricted. Very few individual news outlets release their own APIs for public use and the only one I was able to find was from The Guardian. I later found Webhose, which offers much more flexibility and allowed me to add a little bit more variety into what the results may be by randomizing which website was used for every new request. The main drawback to using Webhose, is the limit of API calls. The limitation of 1000 per month means that I’m restricted in the number of keywords I can use in the game so variety in the game is rather limited. But I think it is a worthwhile trade-off for the extra benefits that Webhose offers over The Guardian such as access to 1000’s of websites and the option to include much more detail in the search query.

I think Python was a good choice for the server side development. It has good support for sockets, networking and also threading. The simplicity of the language was one of the best reasons to use it, but also one of the biggest downsides to it. Having no specified data types can lead to some data occasionally causing issues in the code where it is different to the expected data type. Having SQLite built into the Python language was also a great advantage and removed the requirement for separate SQL script files, instead they can be created inside the Python file used for database communication. Table creation commands are commented out, but can be used if you wish to recreate the database.