ASSIGNMENT: BASIC UI

Technology Used and Architecture

In order to develop the JSON API, a combination of Node.js with Express.js acting as the web framework, and Mongoose/MongoDB (running locally) as the database was chosen.

The whole application was deployed with the directory layouts shown below:

/home

---------/Ubuntu

----------/app

----------/node_modules

----------/package.json

----------/models

----------/jsonitem.js

----------/server.js

"server.js" contains the main code used to deploy the server and listen on the various routes required to respond the various HTTP verbs required, with "jsonitem.js" used to specify the schema for Mongoose to create/modify objects to load into MongoDB. "package.json" contains all the dependencies that need to be used in the application.

Deployment Instructions

A build script ("**nodeRunner.sh**") has been provided which will install all the required pre-requisites, construct the directory structure (as shown above) and deploy the application in the background using forever. The build script is intended to be run with Ubuntu 14.04, and may not function correctly with other versions (for Ubuntu 12.04, there are some modifications that have to made in the script, which are mentioned in the comments).

To run the script, simply transfer the package.json, server.js, and jsonitem.js files into the same directory as the runner script and execute. Please make sure that there is no folder titled 'app' in the same directory as the runner script, as the directory creation will fail.

An AWS Instance of the server with the application running has been deployed in the us-east-1 under the following DNS: **ec2-54-85-6-185.compute-1.amazonaws.com** on port 8080. Given that the server is only for testing, it is open to All Traffic.

Other Information

For the POST/PUT requests, the server only accepts data with the content-type set to application/json or application/x-www-url-formencoded formats. Anything else will be rejected by the server.

All testing during the development process was done using **Postman**, which allows generation of all the required HTTP requests easily.

Enhancing the API

I thought it would've been interesting to use some sort of simple authentication against an existing database of users that we would've had to either create/use and load into the database or data structure that we were working with.