Extra Documentation - Reservoir Planning Tool

**Revised installation for a local testing environment:**

Ubuntu 16x/Windows 10 Ubuntu Bash:

1. sudo apt-get update
2. sudo apt-get install nodejs
3. sudo apt-get install npm
4. sudo apt-get install python-pip
5. sudo apt-get install mysql-server
6. sudo apt-get install libmysqlclient-dev
7. sudo apt install nodejs-legacy
8. sudo apt-get install redis-server
9. git clone <https://github.com/conley31/reservoir-planning-tool>
10. cd reservoir-planning-tool
11. npm install
12. cp config/config.json config/config.json
13. change password in config/config.json to your mysql password
14. cd ~/
15. scp [invent@128.210.17.162:TDPDump.09102017.sql](mailto:invent@128.210.17.162:TDPDump.09102017.sql) . (don’t forget the dot!)
16. For step 15, make sure you have Cisco AnyConnect installed, if not, go to <https://webvpn.purdue.edu/+CSCOE+/logon.html#form_title_text> and login with your Purdue login information. Download the client and run it; connect to webvpn.purdue.edu and enter your Purdue login information. Try step 15 again.
17. Enter password for the server
18. The server should begin to copy the database to your machine, it may take a while.
19. sudo service mysql start
20. mysql -u root -p
21. CREATE DATABASE TDP; -mysql command
22. exit mysql
23. mysql -u root -p -h localhost TDP < TDPDump.09102017.sql
24. Step 23 will take a very long time, with no updates whatsoever, so be patient
25. You now have the database setup in mysql!

MacOS:

* Installation works the same way as windows/linux, but without the nice ‘apt-get install’ commands
* I recommend downloading Homebrew which works in the same way as ‘apt-get install,’ some of the names of packages are different for Homebrew, but they have the same overall naming idea

1. /usr/bin/ruby -e “$(curl – fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install>)”
2. brew install node
3. brew install python
4. brew install mysql
5. brew install redis
6. Note: to start mysql: brew services start mysql
7. Follow steps 9-18 from the Ubuntu/Windows 10 setup
8. mysql -uroot
9. CREATE DATABASE TDP;
10. exit mysql
11. sudo mysql -p TDP < TDPDump.09102017.sql
12. Note: you may need to change your mysql password before doing step 11, use:

$(brew –prefix mysql)/bin/mysqladmin -u root password NEWPASSWORD (where NEWPASSWORD = type in your new password)

1. Again, step 11 will take a very long time with no updates whatsoever, so be patient!
2. Your database is now setup in mysql!

Windows 7:

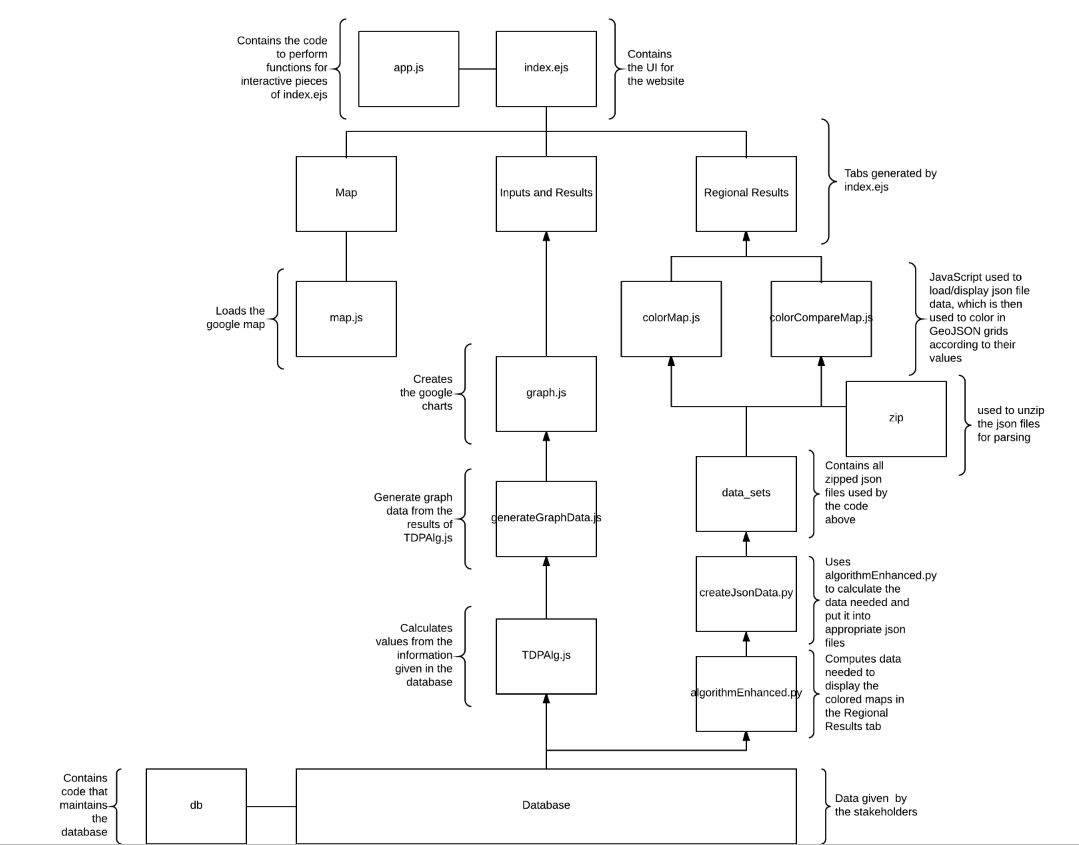
* Run a VM of Ubuntu and follow the instructions for Ubuntu above

To run a local instance of the code:

1. sudo service mysql start
2. run an instance of the redis server in another terminal: redis-server
3. cd to reservoir-planning-tool
4. npm start
5. Open browser of choice and go to whatever local port the terminal is listening on

**Important files and what they do:**

* /views/index.ejs – contains the main HTML code for the entire website
* /util/TDPAlg.js – uses algorithm to calculate results from the information in the database, is responsible for the information displayed in the resulting graphs:
  + Change in bypass volume and storage deficit across pond sizes
  + Pond water depth and storage effects of a single pond size across years
  + Pond water depth and storage effect of a single pond and single year
* /public/data\_sets – contains all json/txt/zip files of the data files and their statistics
  + The data files are named according to their pond size and presets (low/medium/high) – this will be discussed later
* /public/zip – contains code used to zip/unzip the json files
* /public/app.js – contains code for every event listener in index.ejs
* /public/colorCompareMap.js – code used to color in each grid of the GeoJSON overlay based on that grid’s calculated data – for the comparison map
* /public/colorMap.js – code used to identify which json file to load/download, and also to color in each grid of the GeoJSON overlay based on that grid’s calculated data – for the top map on “Regional Results”
* /public/generateGraphData.js – code used to generate the graph data given the results from TDPAlg.js
* /public/graph.js – code used to create the actual google graph given the final results from generateGraphData.js
* /public/map.js – code that loads the actual google map onto the page
* /newFeatures/algorithmEnhanced.py – code that is used to compute the json files which are used to color the top map on “Regional Results”
* /newFeatures/algorithm.js – (old code) – code that was used to compute the json files – replaced with algorithmEnhanced.py
* /newFeatures/calculate-map-data.js – constants that are used as the low/medium/high values for computation in algorithmEnhanced.py
* /newFeatures/createJsonData.py – code that calls algorithmEnhanced.py’s function to compute the data for the json files and save said data to json files
* /db/ - contains python code used to setup/destroy/update the database

**Overview of interactions:**