This is the quick and dirty installation instructions for the TSSG schedule project. You can call it anything you want to but that's it for now.

First of all, be aware that the readme.md files in the two projects are very much out of date. There may be some helpful information in them but most likely there is some very inaccurate information in them. If something conflicts with the following instructions, believe these first. If you find problems in the following instructions, please let me know. I will keep the master for now as later on these will be incorporated into new readme.md file(s.)

For now, clone or download from https://github.com/pscheid1/ngTSSG.git. Use the master repo only. Currently all branches have been merged into master and I will keep it that way for the time being.

When you pull or download the project you should end up with a root directory of ngTSSG. The following instructions will assume that you keep it that way.

Under ngTSSG you will find two very important folders, api and src. Api is what I refer to as the backend code. It was developed in node/js and express. It provides the interface to the database. I will supply an additional breakdown of the api/folders and files later on.

Under the src folder you will find the angular implementation of the front-end code. Basically, everything that runs on the client side (the browser.) A breakdown of the contents will appear later on.

Before you can do anything, you must install node which will also install npm (node package manager.) How you do this depends on the platform and OS so you will have to do this yourself. I'm sure everyone can figure this out.

Next, open a command prompt and navigate to ngTSSG/

Execute the following command: npm install

Npm will read the package.json file in the ngTSSG directory and install the required packages.

Next, cd api, (you should now be in the ngTSSG/api directory).

Execute the following command: npm install

Npm will read package.json file in this folder and install the required packages.

As a result of the above install you should now have a node_modules folder in each of the above two directories (ngTSSG and ngTSSG/api)

In the ngTSSG/api directory execute the following command: npm run dev. If you're successful a server should be running on port 7010.

Next cd back one directory to ngTSSG. Execute the following command: ng serve -o. If successful, it will successfully compile all the angular modules, and start a server on default port 4200. The -o or –open is optional. When present it automatically opens in your default browser.

The ng serve and npm run dev commands start the servers in the development mode. In this mode, changes to files are monitored and when detected angular will recompile the appropriate components and restart the server and express will restart its server (it has nothing to compile.)

Both servers support port modification via optional declarations:

ng serve [-o | --open] [--port ####]

npm run dev [-p #### | --port ####]

For production builds you use ng build and npm run prod. Don't worry about these for now. We'll get into these builds later on.

As for structure, here is a quick description:

As previously mentioned, api contains what I call the backend code.

The major folders are as follow:

Controllers (one each for meetings, venues, and users)

The controllers contain the code that accesses the database. The CRUD functions

Models (one each for meetings, venues, and users)

The models are basically the schemes

Routes (one each for meetings, venues, and users)

Each route module maps the command (get, post, etc) and urls to the appropriate controller function.

Some of the other folders and files are mainly there to support my version of the classic web site that you saw today. It's just for testing.

Under the src directory is all the angular code.

The major modules are under src/app

The generated angular components are in the meeting, user, venue (add, edit, get, admin, create, login, logout sub folders.)

Each folder then contains four files. A list of one is as follows:

src/app/meeting-add

Meeting-add.component.css

Meeting-add.component.html

Meeting-add.component.spec.tx

Meeting-add.component.ts

This model repeats for each component.

Directly under src/app is the main angular root component. Again, the same basic files as above:

app.component.css

app.component.html

app.component.spec.tx

app.component.ts

There are two additional app modules:

app.routing.module.ts (This specifies the angular intra module routing.)

app.module.ts (This pulls all the desired modules, services and components into a single class that can be imported by other classes.)

The src/app/service folder contains interfaces between the components and the back end. They are similar to controllers in the backend.

The src/app/models folder contains what else, models. They are not scheme's but they basically define the objects for meetings, venues and users.

The src/app/guards and src/app/helpers contain miscellaneous modules. The are basically services but are not component specific so they are grouped
outside of the services folder. These folders all begin with an underscore which is not significant. It just a way of making them easily identifiable.

I haven't addressed any database issues here. I have a couple of ideas of dealing with that. Some are temporary implementations and some are more permanent. I don't want to describe anything here until I decide on a few things. I'll try and get something to you in the next few days.

After reading this and/or actually attempting to use it, if you have any questions or have any issues, please let me know. It is very possible that have I forgotten something and a question or description of some issue might be quickly answered.

Paul