# Project Structure

.  
├── app.js  
├── modelStuff.json  
├── package.json  
├── static  
│   ├── bower.json  
│   ├── button.css  
│   ├── face.png  
│   ├── home.css  
│   ├── main.js  
│   ├── templateSpecific  
│   │   ├── ChicagoBooth  
│   │   │   ├── booth.css  
│   │   │   └── print.css  
│   │   ├── Ivey  
│   │   │   ├── Ivey.css  
│   │   │   └── print.css  
│   │   ├── LBS  
│   │   │   ├── LBS.css  
│   │   │   ├── bold.ttf  
│   │   │   ├── normal.ttf  
│   │   │   └── print.css  
│   │   ├── StGallen  
│   │   │   ├── StGallen.css  
│   │   │   └── print.css  
│   │   └── core.js  
│   └── toolTip.json  
├── templates  
│   ├── home.jade  
│   └── uniTemplates  
│   ├── ChicagoBooth  
│   │   └── index.jade  
│   ├── Ivey  
│   │   └── index.jade  
│   ├── KellogAlumni  
│   │   └── style.css  
│   ├── LBS  
│   │   └── index.jade  
│   └── StGallen  
│   └── index.jade  
├── uni2Category.json  
└── utils.js

Important files:- - app.js - Main file that starts the server. Takes port number from process.env or sets to 3000 by default. - package.json - Lists out all the dependencies etc for the project. If you've ever worked with node, you know what this is about. npm start will start the server(note that it runs with nodemon so any changes to files cause a rebuild(yay!) , npm test runs nodemon with --debug-brk, allowing you to use node-inspector(in a separate terminal window, of course). - static/bower.json - Lists out all our frontend dependencies. Due to various difficulties, some of the frontend libs are also installed in node\_modules, so don't expect consistency here. - static/button.css - Extra CSS for the button styles(governs the hover and active states seen in the templates). Set to grey/black by default. If you wish to do custom button styling for a template like in St. Gallen, add custom styling in the template itself. - static/home.css - CSS for the homepage. TODO: add Vmock custom branding. - static/main.js - JS file for the homepage. All that it does right now is generate the link for the submit button on the homepage. - static/templateSpecific - Contains folders with *styling* specific to each template. Things like column widths, capitals/small letters, bold/italics, fonts, colors etc covered here. Please refrain from making changes that may break layout. Also, only flexbox based layout is allowed( no CSS floats/other hacks that can potentially break things). Within each folder, there's 2 files(other than ttfs for fonts) - a <insert university name>.css and a print.css. The first is responsible for web based styling. The second gets activated when a user tries printing the web page. - static/templatespecific/core.js All of the JS common across templates. Contains Angular code along with code to check email ID format etc. - tooltip.json - no longer necessary as there is a tooltip object in core.js. - templates/ - contains the templates themselves. They're written in [Jade](http://jade-lang.com/) (now called pug). - templates/home.jade - Template for the homepage. - templates/<uniName>/index.jade - Actual template. Includes several classes and may have template specific JS. Make sure you know what you're doing before you touch this.

# Block Diagram

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  
| | | | | | | |  
| JSON | -> | Template | -> | Angular variables | -> | Webpage |  
| | | | | with 2 way binding | | |  
|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_\_|

# Technology Stack

Express.js, Jade on the backend. Jade directly inserts the resumé object as a string into the webpage. Angular.js reads the variable and inserts objects into the template(lists, bullets, headings etc). Each of the editable elements is contenteditable. When there's a blur/change event, Angular reads the element text and propagates the changes backwards into the scope variable where the resumé data is stored.

# Modules

1. In app.js, the mba object contains all the static data that we're displaying. This will be replaced by a REST API call, probably.
2. When the user selects a template and user\_ID, the relevant data is passed onto the template(refer to the res.render function in app.js). This data is passed as a plain text string to the web page.
3. The frontend reads this data( stored as resumeDataOld ), and passes it over to the $scope variable in angular.
4. The angularjs code in core.js reads all the data and binds it to the template (read up on ng-repeat, ng-model etc before editing angular code).
5. Once the data is bound to the DOM, where it can be edited. Edits are automatically back-propagated to the angular scope variable