**File: example\_BackboneLayouts.jpg**

This file is a screenshot of our login page. In the center of the page, there is a region populated with one view, either the Login view or Sign Up a new user view (placed on the right). For the Login view, there are two fields that must be filled out in order to login in, Username and Password. For the Sign Up view, the fields listed are First Name, Last Name, Username, Email, Password, and Password Again.

**File: Tutorial\_for\_Application.md**

**Back-End**

**Main**

- Flask

- PostgreSQL

**Addons**

- flask

- sqlalchemy

- psycopg2

- flask-sqlalchemy

- flask-login

- flask-cors

- flask-mail

- flask-bcrypt

- redis

- gevent

- gunicorn

**Front-End**

**Main**

-Backbone

-Marionette Framework: Framework on top of Backbone to provide application structure

-Foundation Framework: Framework that handles styling for the website (responsiveness,mobile, etc)

**Addons**

-Handlebars: Templating Engine

-RequireJS: Handles Javascript file dependencies

**Breakdown of Backbone/Marionette**

**Folder Structure**

-static: Folder is in backend folder now

--CSS: Contains website's CSS files

--Img: Contains website's IMG files

--JS: Contains website's JS files

---Lib: Contains the JS files required to run the website (foundation, backbone, require,marionette, etc)

---App: Contains the backbone/marionette files for the website

----Collections: Will contain our Backbone model collections

----Config: Contains the requireJS configurations

----Controllers: Contains the main controller to handle routing and transitions betweenthe pages of the websit

----Layout: Contains the Backbone/Marionettte Layouts (Home and Main so far)

----Models: Contains the Backbone/Marionette Models (Sessions and Users so far)

----Routers: Contains the Backbone/Marionette Routers (AppRouter so far)

----Templates: Contains the template files

----Views: Contains all our views

----App.js: Instantiates the Marionette application and defines the mainRegion in which our layouts are put into

**Application Hierarchy**

\* This explains the Application at a pretty high level, things may change as our application gets sophisticated. As of right now, every view is an ItemView. Later on, we may start using CollectionViews, CompositeViews, etc.

1. Application starts in index.html. "div#mainRegion" is the "main" region in which the home page and main page are rendered into. Remember that a single application (which are website is) does NOT re-render the whole page, instead it only renders parts of the page when needed. Thus, essentially a single page application has only one "page" and different "sections" in the page. Our website is a multi- single page application. The home page is one of the single pages and the main page (page after user logs in) is another page. They both have different layouts and sections on the page which is why the website is a "multi-" single page application.

2. Index.html calls two js files in the line "<script type="text/javascript" src="js/libs/require.js" data-main="js/app/config/Init.js"></script>". Src references the requireJS source file. Data-main calls our require config file in which we establish our JS paths and dependencies.

3. In the Init.js we instantiate our AppRouter, our controller (in the AppRouter), and start the Marionette application itself. AppRouter defines the url routes and Controller contains the functions that the AppRouter routes to.

4. Depending on the function that is called via AppRouter/Controller, the home or main layout will be rendered (HomeLayout.js or MainLayout.js).

5. Layouts establish the "structure" of the single page. The "sections" are defined as regions in layouts. The layout uses a template which is a html file that contains the shell of the website. It uses html ids to determine which parts of the template are the regions.

6. In these layouts, we insert Backbone views into the regions. A region can only have one view at a time. A view is what handles user interactions with the page. Each view also has a template which is the html file that will be inserted into the "section"/region of the page which was defined in their respective Layout. (Templates are what the user actually "sees", it is the html itself which is rendered onto the website)

7. An example of this is we currently have a HomeLayout.js which has regions for the login/signup area and about information. Our views are Home\_LoginView.js, Home\_SignupView.js, and Home\_AboutView.js which have home\_loginBox.html, home\_signupBox.html, and home\_about.html templates respectively. We can switch these views into the regions defined in the Layout. This is what gives the website its single page feel.

8. Backbone provides a level of abstraction in regards to how DOM manipulation and javascript/jQuery is handled. It has an event function (example in Home\_LoginView.js) that handles javascript events. Thus, instead of having to make another JS file and defining jQuery event bindings and such on elements of the DOM, we can just do it directly through Backbone. An example would be "click #loginButton": "login" which means call the login function when the element with id of "#loginButton" is clicked.