****

|  |
| --- |
| **Developer Documentation**  **Automation Dashboard** |
|  |

\***A** = Added / **M** = Modified / **D** = Deleted

# Revision History

| **Version** | **Date** | **Author** | **A/M/D\*** | **Comments / Title / Brief Description** |
| --- | --- | --- | --- | --- |
| .1 | Aug 21, 2017 | Joshua Sun | A | First version |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

\***A** = Added / **M** = Modified / **D** = Deleted

# Contributors

| **Name** |
| --- |
| Joshua Sun |

Contents

[Revision History 2](#_Toc491087582)

[Contributors 2](#_Toc491087583)

[Table of Contents 3](#_Toc491087584)

[1 Purpose 4](#_Toc491087585)

[2 Technologies used 4](#_Toc491087586)

[3 Installation 4](#_Toc491087587)

[3.1 Prerequisites 4](#_Toc491087588)

[3.1.1 Version Check 4](#_Toc491087589)

[3.1.2 ‘Command Not Found’ 4](#_Toc491087590)

[3.2 Cloning From Git 5](#_Toc491087591)

[3.3 Installing dependencies 5](#_Toc491087592)

[3.3.1 Global dependencies 5](#_Toc491087593)

[3.3.2 Project dependencies 5](#_Toc491087594)

[3.3.3 Confirm that dependencies are installed correctly 5](#_Toc491087595)

[3.3.4 Adding more dependencies to this project and installing them on Jenkins 5](#_Toc491087596)

[3.3.5 Essential dependencies explained: 6](#_Toc491087597)

[4 Runing the server 6](#_Toc491087598)

[5 Developing 7](#_Toc491087599)

[5.1 Files and File structure 7](#_Toc491087600)

[5.2 The workflow 7](#_Toc491087601)

[5.3 Backend development 7](#_Toc491087602)

[5.3.1 Server 7](#_Toc491087603)

[5.3.2 Database 7](#_Toc491087604)

[5.4 Frontend development 8](#_Toc491087605)

[5.4.1 Wrting JS - React JS 8](#_Toc491087606)

[5.4.2 Writing HTML 8](#_Toc491087607)

[5.4.3 Writing CSS 8](#_Toc491087608)

[6 Version Control and Deployment to Jenkins 8](#_Toc491087609)

[6.1 Commiting to Git 8](#_Toc491087610)

[6.2 Deploying on Jenkins 9](#_Toc491087611)

[6.2.1 MongoDB on Jenkins 9](#_Toc491087612)

[6.2.2 Dev deploy 9](#_Toc491087613)

[6.2.3 Production deploy 9](#_Toc491087614)

[7 Running Tests 9](#_Toc491087615)

[8 Useful Links 9](#_Toc491087616)

[9 Acknowledgments 9](#_Toc491087617)

[10 Licence 10](#_Toc491087618)

# Purpose

This is the official documentation of the Automation Dashboard MERN stack web application. The aim of this documentation is to give you a detailed look into the tools and technology used to build this tool, and ways to maintain it.

# Technologies used

Frontend

* React
* JQuery (for Ajax requests only)
* HTML 5, SASS
* Modernizr, Normalize CSS, Semantic Ui, React-Table

Backend

* Node.js And Express
* MongoDB

Build tools

* Yarn, Gulp, Babel, Node-ruby-sass, Browserify, Browser-sync

Version control

* TFS Git

# Installation

## Prerequisites

Make sure you have Node.js, NPM, Yarn and MongoDB installed. These can be obtained from their official websites. Continue once you have these installed.

Note: after you install mongodb, create the directory C:/data/db

### Version Check

On Windows, open Git Bash

On MacOS or Linux, open Terminal

Confirm the following:

$ node -v

7.9.0 (or greater)

$ npm -v

4.2.0 (or greater)

$ yarn --version

0.27.5 (or greater)

$ mongod --version

V3.4.6 (or greater)

### ‘Command Not Found’

If any of the above commands give ‘command not found’ and you have installed the tool, it means that you have not set the PATH variable correctly. To configure you system environment variables, see <https://www.java.com/en/download/help/path.xml>

## Cloning From Git

To clone this project ot your local, run

$ git clone <http://10.115.17.17:8080/job/Leap_DataCreation_Event/>

$ cd Dashboard

You are now at the root folder of the Dashboard project

## Installing dependencies

Confirm that you are at the root folder of the dashboard project, if not, see 3.2

### Global dependencies

These are command line build tools needed to compile our project, on bash, run:

$ yarn global add gulpjs/gulp.git#4.0 gulpjs/gulp-cli nodemon browserify

### Project dependencies

These are project specific dependencies, run:

$ yarn install

the Yarn package manager will install all dependencies automatically. Unlike global dependencies, project dependencies are listed under package.json

### Confirm that dependencies are installed correctly

Run:

$ node app.js

If you get ‘server started on port 3000’, then you are all good to go. Otherwise you have not setup the project correctly.

### Adding more dependencies to this project and installing them on Jenkins

Since our Jenkins server does not have internet access, Yarn install will not work properly. To solve this issue, we must install dependencies in offline mode. This setting is written in .yarnrc file in the project’s root folder. This file tells Yarn that whenever you run $ yarn add to add dependencies, save a compressed version of the dependency in a folder called npm-packages-offline-cache. This folder is uploaded to git along with the project so the dependencies can be installed correctly on Jenkins. For this project, this has already been set up so you don’t need to worry about it.

### Essential dependencies explained:

Build dependencies

These are listed under ‘dev-dependencies’ in package.json and will not be installed on our production environment on Jenkins

* Gulp: a build tool that compiles SASS into CSS, transforms React JSX into JavaScript, transpiles ES6 syntax into ES5, monitors changes as you develop the project and reloads the browser automatically, and starts our server
* Nodemon: watches for changes in our project and restarts the server automatically
* Eslint: a linter that checks for JavaScript syntax and style errors
* Babel: transpiles ES6 into ES5
* Browserify: bundles your JavaScript files into a single bundle.js and enables CommonJS modules on the frontend
* SASS: an improved version of CSS that addes features such as variables and inheritance to writing CSS, must be comiled to CSS
* Forever: restarts server if it crashes
* Mocha: runs unit tests

Server and Frontend dependencies:

These are dependencies needed for the server and frontend to run. These must be installed on Jenkins.

* Express: powerful framework for Node that abstracts away many low level server operation
* EJS: HTML templating engine, compiles .ejs files into HTML files
* Body-parser: parses the request body into a JS object in req.body
* Moment: Date & time formatting library
* React: frontend library for build declarative modern web apps
* Mongoose: MongoDB driver, allows for running queries in Node
* JQuery: library for manipulating the DOM, attaching event listeners and making AJAX requests

# Runing the server

## Vanilla Server

There are a number of ways to start the server, at the most basically level, you can run:

$ node app.js

In the project’s root folder.

## Development Tools when Running Server

4.1 starts server in ‘barebone’ meaning you don’t get things such as logger or automatic restart. Instead, we use Gulp (see 3.3.5) to add some build tools and start our server.

Run:

$ gulp

Once you see ‘server started on port 3000’, you are good to go.

## Passing in Environment Variables

It is easy to pass in environment variables. For example, to change port:

$ PORT=1234 node app.js

With gulp:

$ PORT=1234 gulp

# Developing

## Files and File structure

* Node\_modules – where dependencies are saved locally, not uploaded to git
* Doc – project planning and documentation files
* Model – Mongoose database schema
* Public – static assets such as Javascript, images and CSS
* Routes – route handler functions
* Src – precompiled SASS and React components written in JSX
* Views – EJS files (precompiled HTML files)
* .babelrc – babel setup for transpiling ES6 to ES5
* .eslintrc – linter configurations
* .gitignore – tells git to not upload certain files
* App.js – entry point of our server
* Gulpfile.babel.js – entry point for gulp when you run $ gulp, written is ES6 hence the .babel
* LICENSE – this project uses GPL 3.0 freeware license
* Package.json – lists all denpendencies and project meta information
* README.md – markup files of the documentation
* Yarn.lock – auto generated by yarn

## The workflow

This section will walk you through how this project is compiled when you type $ gulp

1. Goes to ./src/scss and compiles .scss files into css files and save them in ./public/stylesheet
2. Goes to ./src/components, Babel transpiles JSX (React) syntax into ES6 Javascript, then transpiles ES6 into ES5, then Browserify relaces all ‘require’ statements with the actual Javascript code, then concatenate to reduce file size, and finally saves bundle.js into ./public/javascript
3. By using Nodemon, calls start server
4. Goes to ./app.js, load all middlewares, load route handlers, then start server on port 3000
5. Browsersync detects file changes, reloads your browser window automatically

## Backend development

### Server

The Node.js server is responsible for routing and database queries. The server setup file is ./app.js, the route handlers and database queries are written in files in ./routes, and the database Schema are in ./model

For documentation on the Express framework, see <https://expressjs.com/>

### Database

This app uses MongoDB as its database. To connect and perform CRUD operation on MongoDB in Node.js, we use the Mongoose plug in.

For documentation on Mongoose, see <http://mongoosejs.com/>

## Frontend development

### Wrting JS - React JS

Unlike the traditional DOM manipulation approach with jQuery used in 90% of the websites you see, this app utilizes the React JS frontend framework. React is developed by Facebook and allows you to write declarative, component-based frontend code that is clean and highly maintainable. This new approach is quickly being adopted by tech companies in Silicon Valley and around the globe.

For this project, the React components are in ./src/components with app.js being the top level of the app. You should not need to direct write Javascript code on the frontend or do any DOM manipulation.

For documentation on how to use React, see: <https://facebook.github.io/react/>

### Writing HTML

Although most of the frontend code in written in React, if you do need to write html, all html (.ejs) files are under ./views. EJS support all html syntax so you can treat all .ejs files as .html files. There are many cool features that EJS provides.

To see documentation on EJS, go to: <http://ejs.co/>

### Writing CSS

This project uses SASS for preprocessing CSS. All SASS files are under ./src/scss and will compile to bundle.css in ./public/stylesheet

## Posting new data to MongoDB

The server has serveral REST API endpoints for performing CRUD operations

### Endpoints

GET – get existing data by id - hostname:port/TC/:id

GET – get exisiting data by name - hostname:port/TCByName/:name

POST – New Data - hostname:port/TC

PUT – update data by id - hostname:port/TC/:id

DELETE – delete data by id - hostname:port/TC/:id

### Naming Convention

### Test cases

Database does not run any validation on test case naming convention, but please follow this pattern: word1\_word2\_word3\_word4 etc. example: manage\_client\_info\_test

### BP

Database runs the following validation using regular expressions on BP names:

(^bp[0-9]+(?:\_[a-z0-9]+)+$)

Explanation (variables placed in [ ]): name must be of the following format:

bp[number]\_[word]\_[word].

There is no limit on how many words consist the name.

Example: bp012\_maintain\_client\_services

### BPG

Databse runs the following validation on BPG names:

BPG name must be one of:  
'enablement\_framework',

'client\_maintain\_client\_services\_org\_maintain',

'accounting\_financial\_data\_service\_purchases',

'benefit\_processing\_disability',

'avc\_pension\_management',

'annual\_processing'

# Version Control and Deployment to Jenkins

## Commiting to Git

Make sure all new features are commited on the dev branch and thoroughly tested before merging to the master branch. Once you are finished making changes, issue the following commands

$ git add .

$ git commit -m “commit message”

$ git fetch

$ git rebase

$ git push

After $ git rebase, you should get fast-forward. Merge conflicts are rare and if they do come up, you are probably not commiting your changes right.

## Deploying on Jenkins

### MongoDB on Jenkins

Before starting your server, make sure MongoDB is running:

<http://10.115.17.17:9090/view/Automation%20Tools/job/MongoDB/>

Note that the dev and master branches each use a separate database, so chanes in dev won’t affect master

### Dev deploy

After pushing your changes to the dev branch go to <http://10.115.17.17:8080/view/Automation%20Tools/job/dashboard_dev/>

And run the job, this starts the dev server on port 4001

### Production deploy

After pushing your changes to the dev branch go to <http://10.115.17.17:8080/view/Automation%20Tools/job/dashboard_master/>

And run the job, this starts the dev server on port 4000

# Running Tests

Units tests for this project are currently unavailable. If you would like to write unit tests, I recommend using the Mocha library:

<https://mochajs.org/>

# Useful Links

* Git repository URL: <http://tfs.omers.com/tfs/Production/LEAP/_git/EnrollmentDataCreation>
* Node.js documentation: <https://nodejs.org/en/docs/>
* Express.js documentation: <https://expressjs.com/>
* React js documentation: <https://facebook.github.io/react/>
* SASS documentation: <http://sass-lang.com/>
* MongoDB documentation: <https://docs.mongodb.com/?_ga=2.191918509.667046423.1503338162-1605050878.1503338162>
* Mongoose documentation: <http://mongoosejs.com/>
* Yarn documentation: <https://yarnpkg.com/lang/en/>
* EJS documentation: <http://ejs.co/>
* Javascript ES6: <http://dev.venntro.com/2013/09/es6-part-1/>

# Acknowledgments

* Parasar Saha for coming up with the idea
* Jannie Zheng for writing the Selenium job to dump data

# Licence

This project is licensed under the GNU Public General License Version 3, see LICENSE.md for details