Hughes Fieldhouse Developers Manual

Developers: Matt Woolery, Dattu B Medarametla
Clients: Greg Hansen, Brooke Byland
GDP 1 Instructor: Dr. Denise Case
GDP 2 Instructor: Dr. Ajay Bandi

Preface

This document is to be used by developers that wish to set up the Hughes Fieldhouse app on their local machine and instructions on how to work on the project. Steps on setup will be highlighted for the project as well as setup steps for changing to different service accounts to be used for the project.

Table of Contents

Getting Setup Locally

The project source code can be found in the project repository at https://github.com/mwoolery/project-charter-template . Make sure that you have downloaded git, Node.js 10.15.3, and a code editor of your choice to code with.

1. After you have familiarized yourself with the project a bit, feel free to download it to your local machine using the clone or download option in the repo using either git clone or the download zip file, which ever you are comfortable with. Alternatively, you could fork this repo into your own github account if you decide that you will want to set up your own Travis CI and Heroku Deployment with your own Github repository for the connection to these services
2. After you have done this you will need to navigate to the src folder where the project source code is stored and run npm install , to install all of the node.js module dependencies needed to run the project.
3. Next you will need to create a file called config.json . In this file you will be inserting api keys that your project needs to run with your individual Mailgun and Atlas MongoDB connections, refer to the Mailgun and MongoDB setup for steps to get your api keys. An example of what this will look like can be seen below.

4. After you create your config.json file with your information, you will be able to run the command node app.js to start the app up. After the app is started, you can visit the index of the site by visiting http://localhost:3000/ in your browser of choice. You have the app up and running with your information on your local machine now.

Mailgun Setup

Mailgun is the service that the app uses to send emails to the Hughes Fieldhouse Team with information users enter into the Contact form found on http://localhost:3000/contact. The Mailgun service is free to use, easy to set up, and allows up to 10,000 free emails to be sent from using one account api key. There are paid options that give you better tracking options and allow for more emails to be sent. Instructions for setup will be provided below.

1. Sign up for a mailgun account at https://signup.mailgun.com/new/signup, you will just need to do the basic account set up and choose the free plan which provides 10,000 emails
2. After logining in you will be able to see a dashboard which you may find useful for you to figure out some insights from the messages that have been sent.
3. Next you will click on the domains tab, here you can create a new domain, if you want your own domain, you will need to pay for one at this time, otherwise you can have a sandbox domain created by Mailgun. Take note of your domain name, as you will need to have it for your config file in the setup for running the app locally and also to have in the deployment's config variables.
4. After your domain has been setup, you can click on the domain from the domain tab and it will bring you to a screen with information about the domain, here you will be getting the API key for use in the config file and in the config variables in the deployment.
5. Next you will be going to the app.js file in the application source code and go to the mailgun options section, example code to demonstrate what it looks like can be seen below. You will be modifying the from, to, and subject as needed here, the text field comes from the contact form's body, the code where it is gathering information from the contact form is above the mail options. In from, change what is in the <> with either your email, or you can use the mailgun default smtp login like I did in the example code below. Next you will change the to: to whichever user at the fieldhouse would be receiving the email. You can change the subject as well which changes the subject of the email that is sent to the user specified in the to: section.

```
const mailOptions = {
  from: 'Hughes FieldHouse Contact Form <postmaster@sandboxc3954874d6c14d68a692fc29a2900ae9.mailgun.org>', // sender address
  to: 'mwoolery@nwmissouri.edu', // list of receivers
  subject: 'Message from Resume Website Contact page', // Subject line
  text: str


}
```

After you completed the above steps, your mailgun account will be used by the application to send emails to a designated user or users when someone fills out the contact form.

Atlas MongoDB Setup

You may have noticed that this node.js application uses the Mongoose Node Module, meaning that it is setup to communicate with a MongoDB database. The cloud MongoDB service that we used to store the data entered in the BannerItem editor is Atlas. Atlas is another easy service to set up so follow the below steps to get the app to use your Atlas MongoDB instance.

1. Create an account at https://cloud.mongodb.com/user#/atlas/register/accountProfile
2. After Creating an account, you will need to create a new Cluster. Choose one of the Free options in the region that you are in so that it does not cost you anything and data operations can be done quickly because of the networking time will be lower for clusters closer to you. The free option is limited in how much data storage, but the amount of storage needed for this app is low.
3. Next, go to the Security section of the dashboard, and click add new user, here you will set up a dbuser name and dbuser password. Give the user atlas admin privileges on your account. Remember your dbuser name and password as it will be used in your connection string
4. After you have your cluster created and you have a user set up, you can go to the clusters dashboard and click the connect button to get your connection string needed for the app config file and the heroku deployment config variables. You will click on “Connect Your Application” and set the connection driver options to Node.js and version to 2.2.12 or later. Once you copy your connection string, add it to your config file that you created in the Getting Setup Locally section and also in your Heroku Deployment Config Variables, remember to use the dbuser name and password in your connection string for it to work.

Heroku Deployment

The app is currently hosted on Heroku at https://hughesfieldhouse.herokuapp.com/ . You may decide that you would like to change the application url by setting up your own Heroku Deployment. Below is the steps we took to deploy the application for free using Heroku. Visit https://devcenter.heroku.com/ to learn more about Heroku other than the steps to free deployment that we provide.

1. Create an account with Heroku. After you have your account set up and you are logged in to your personal dashboard, you will need to click “New” in the top right and name your app what you want. What you name your app is what will appear in your url followed by herokuapp.com.
2. After the Heroku app is created, navigate to the deploy tab. Next you will be connecting your app to github. Click on github in the deployment method section. Set your github up in Heroku and then select the repository that you have the Hughes Fieldhouse app in. After this you will click on enable automatic deploys. If you decided to use a CI tool like Travis CI, which instructions will be provided for in another section, go ahead and click wait for CI to pass.
3. Next, you will go to the Settings tab. Here you will be setting your Config variables to the same variables that you have in your config.json file that you created in the Getting Setup Locally Section. Click Reveal config variables, then start filling in the keys and values with your variable from the config file for the key and the variable’s value as the value.
4. After your config values are added, go back to the deploy tab and click manual deploy to start the first build of the project from Github. Once it is successful, you can click open app and see your deployment. Subsequent commits to your github will then automatically be deployed to Heroku.

Travis CI

The CI tool that we set up to deploy to Heroku on successful builds is Travis CI, it is easy to set up and code for the deployment is already provided in the github repository. Follow the steps below to set up your Travis CI account.

1. Sign up with Travis CI using your Github account that way Travis will instruct you to give it permission to access your repos so you can set up a repository in the CI.
2. Under your account repository, navigate to the repository that you are using for the Heroku deployment and select the toggle to get started.
3. Navigate to the repo's Travis CI page and click on the settings. Here you will add in your HEROKU_API_KEY and value so that Travis can build it and know where to deploy it when it is successful. Get the api key from API key section of your Heroku Account's profile.
4. Next you will need to go into the .travis.yml file of the repo and change the app field to whatever you named your app and the repo field to whatever repo you have it stored in on Github.
5. After you completed the above steps, you can trigger a build manually by clicking "More Options" and then "Trigger build", or you it will trigger on each deploy. Alternatively, you may decide to go into the settings and set up Cron jobs to trigger builds at specified intervals as it is commonly done in DevOps.

Social Media

It is important for the client to be able to engage their users via social media, they would also like to see that what they are doing on social media appears on their website. The clients have yet to decide on a profile for Twitter or Facebook that they would like to embed, so currently Northwest's feeds are included. Follow the steps below to set up the feeds on the homepage of the app.

Twitter

1. Go to https://publish.twitter.com/ . Next you will need to enter in the Twitter profile that you would like to embed. Select Embeded Timeline and then copy the code provided
2. Go to the index.ejs file for the main app and scroll to the code documentation for the social media sections. Paste the provided code over the tags for twitter-timeline to replace the feed.

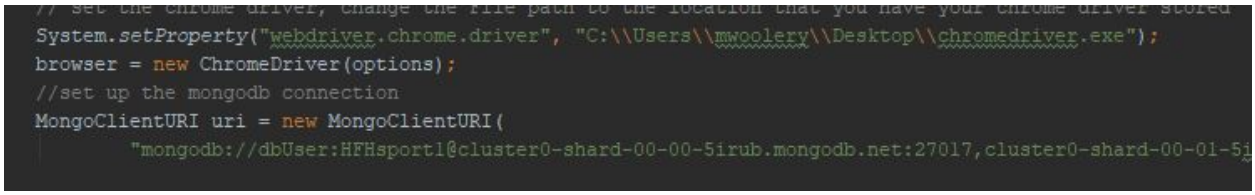
Facebook

1. Go to https://developers.facebook.com/docs/plugins/page-plugin
2. Enter in the url of the page that the client decides to use for sharing posts about the Fieldhouse. Enter the width and height as desired for sizing, we used 450px width and 430px for the height, but you could decide to change these at another time as you see fit.
3. Click Get Code, then select IFrame. Copy the code and navigate to the index.ejs file for the main app and scroll to the code documentation for the social media sections. Paste the provided code over the facebook iframe to replace the current feed with the one that the client wants to be shown.

Selenium Tests

For testing, we chose to use Selenium automation testing to test how the different pages integrated together. Since there is not any algorithms to be individually tested with Unit Testing, this is the best way to ensure that our site is functioning as expected. The tests check to see if the controllers are returning the expected pages, checking to see if the database is handling data correctly, and that various other website functionality performs as expected such as the resizing to a mobile screen. Make sure to download Selenium and get ChromeDriver.

1. Get our test code from https://github.com/mwoolery/HughesFieldhouseTests . It is a Java project that we created in IntelliJ. Depending on which IDE that you used and what JAR files that you downloaded, you may need to download some JAR files to run the tests. The JAR files that we had to download to run it are listed below
    a. mongo-java-driver-3.9.1.jar - https://github.com/mongodb/mongo-java-driver/releases/tag/r3.9.1
    b. selenium-server-standalone-3.141.59.jar - https://www.seleniumhq.org/download/
    c. Junit4.12
2. After you get the project cloned and have the necessary jar files, the next thing is to change the MongoClientURI uri to the one that the app uses. Also you will need to change the location of the Chromedriver.exe that downloaded from https://sites.google.com/a/chromium.org/chromedriver/ to where ever you have it located on your local machine. The below code has the 2 lines the you will need to change for your reference



3. You may choose to change the login keys to whatever ends up being the finalized login for the app. It is currently set up as “admin” “admin” for username and password. It will likely be changed where a Fieldhouse Team member’s login is used instead.
4. After you have downloaded all of these things and changed the code as indicated you need to start up the project locally, referring to the Getting Started Locally instructions may be helpful. Once the project is running locally, you are ready to run the tests. The tests try to load the pages for the controller sections and gets the title of the tab that is set after a successful load. It then tries to login to the banner item editor and tries to perform the various data modifications on a test item that it creates. After all of this is tested it checks to see if it resized for mobile then exits out.

Site Assets

Site assets were added to src/public folder. Here is where the scripts, css, and images are stored. The area that may be of most interest to be changed is the images, as it was often mentioned that the backgrounds of certain things would need changed frequently. You can change the files to new images directly, or you could upload new files to this folder and change the code to have the new image file.