#bootcamp/Project2/README#

The Rogue Scholar

This application is a forum for the exchange of ideas. It encourages its users to explore and experience

Links

Heroku: Rogue Scholar

Motivation

Problem:

The prevalence of the "bubble" in social media and social interactions in general. The unwillingness of the of the audience to travel outside of their comfort zone. We are inclined (or is it persuaded? maybe, even encouraged?) to remain in our bubbles with no motivation to explore the many trains of thought on any subject.

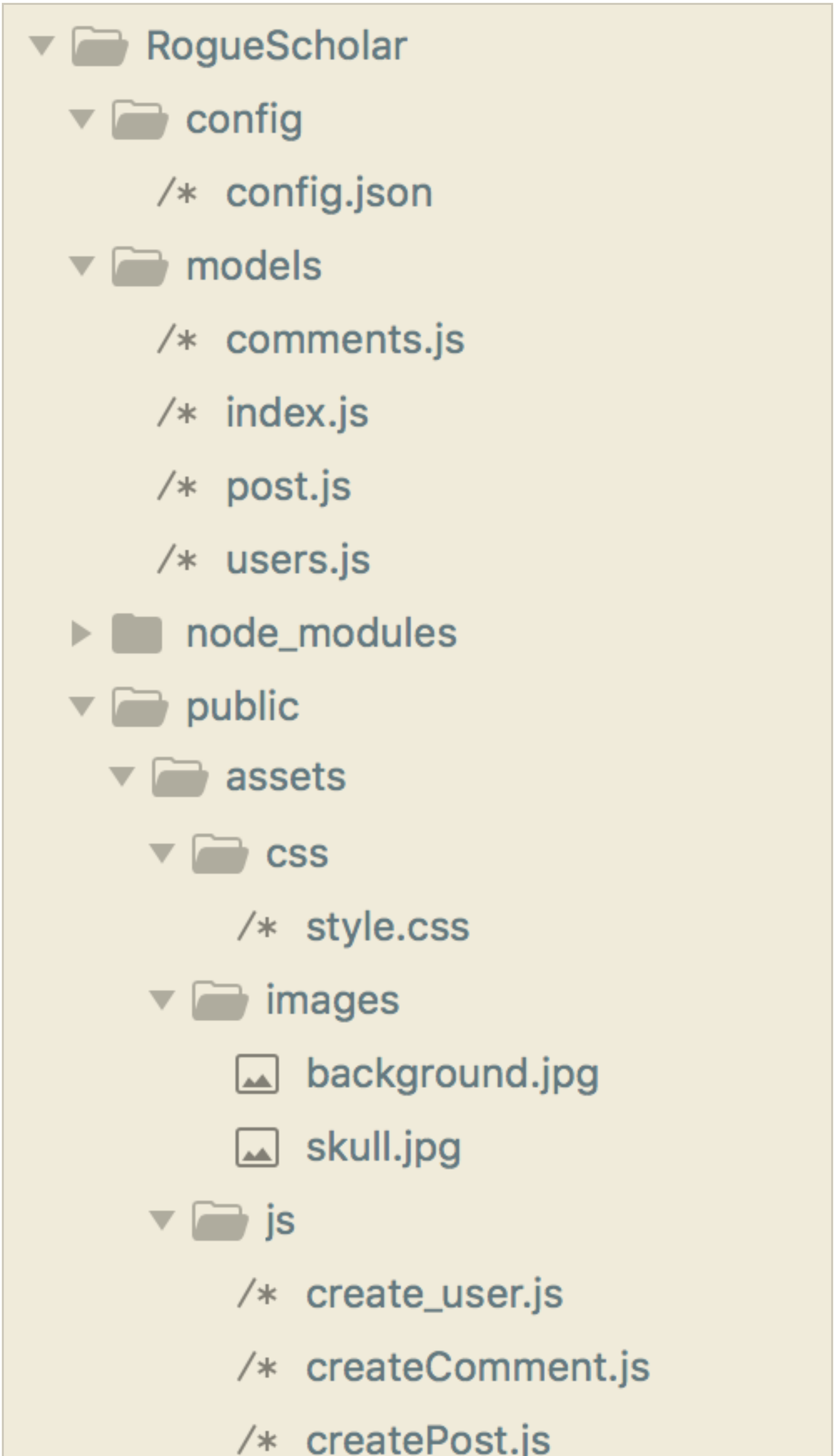
Solution:

"Rogue Scholar" is a social experiment meant to challenge the user to "burst the bubble". It is intended to encourage the user to travel off of the beaten path that we subconsciously create and challenge them to interact and comment on another user's thoughts on a subject that they did not actively choose. It also provides "stimulus" in the form of random news articles to provoke the user to share their thoughts.

File Overview – what does what?

This is a complex project. A short overview of the function and structure of each file will be helpful. We'll look first at the file organization, then at the driver code, and then at how the MVC design pattern is implemented.

Project Directory/File Structure



```
RogueScholar
  config
    config.json
  models
    comments.js
    index.js
    post.js
    users.js
  node_modules
  public
    assets
      css
        style.css
      images
        background.jpg
        skull.jpg
      js
        create_user.js
        createComment.js
        createPost.js
```

Driver Code

- server.js
  - Requires the npm modules express, body-parser, method-override.
  - Requires routes
    - html-routes.js
    - post-api-routes.js
    - user-api-routes.js
  - Requires the models that are used by Sequelize for synching and communicating with the database.
  - server.js contains the code that
    - Sets up the parameters for the Express server.
    - Establishes for the Express server the /public directory as the source for static information (e.g., images and css).
    - Initializes Handlebars.
    - Starts the server listening on the default port (usually 8080).
    - Renders the home page.

MVC Design Pattern Implementation

Files that constitute the "Model"

These are the files that interact with the database.

"Model" files

"View" files

"Controller" files

Build status

This is version 1.0 of the application. Version 2 will include the replacement of orm.js with Sequelize, as well as refactoring and clean up of all the code.

Screenshots

Tech/framework used

- MVC design pattern
- Materialize css framework
- Handlebars
- Node
- Express
- Sequelize
- JAWS_DB (MySQL)

Features

This project is a complete, end-to-end implementation of a web app.

- Front End/Browser
  - Handlebars
  - CSS
  - Dynamic web page rendering
- Middleware
  - Express routing
  - Controller utilizes Sequelize for object - relational modeling to interface to the database.
- Server
  - Database

Code Example

Installation

1. git clone the repository.
2. From the command line,
   1. $ npm install package.json
      1. This will install node and the following required packages in the node_modules directory:
         - "body-parser": "^1.18.2"

        - "express": "^4.16.2",
        - "express-handlebars": "^3.0.0",
    - "mysql": "^2.15.0”

3. Set up posts_db
4. From the command line, cd to the RogueScholar directory. Start the server with the following command:
    - $ node server
5. Point the browser to localhost:8080.

License

Copyright 2018 Joshua Gomez, Logan Sawyer, Steve Hulme, Michael Ashe.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MIT © Joshua Gomez, Logan Sawyer, Steve Hulme, Michael Ashe (2018)