Mini-Project: gBlog

You and a partner will build a Full Stack app that allows users to CRUD blog posts.

Wait!!!!

Before you rush in, scan this whole document. Soak it in. Take your time.

Think Things Through Thoroughly

These instructions leave a lot of room for interpretation, and may not list exactly everything you need to do to build this CRUD app.

Be smart, be pragmatic, be excellent to each other.

---

gBlog Relationships

gBlog consists of the following relationships:

- A website has many blog posts.
- A blog post has a Title (Required), Body (Required), Author Name (Required), and Creation time (Required).
- A blog post can have many comments.
- A comment has a Body (Required), and Author Name (Required) and Creation time (Required).

> Do not worry about authentication and authorization right now. Auth is a stretch goal. Get your blog post CRUD working and UI created FIRST.

Steps for Success

1. Think and Diagram FIRST

Work with your partner to create an entity relationship diagram. Do not move on until you have agreed on a database structure.

2. Project Setup

- One team member should get the project directory setup (using the express generator, or from scratch using npm init, npm install etc.).
- The other team member should create a new repo on github and prepare it for the initial check in.
- Check in your code

3. Create the Database and Tables

- Work together to create the database and tables using the entity relationship diagram you drew as a reference.
- You may use your preferred method for creating the database and tables (psql, pg or knex migrations).
- Check in your code

4. CRUD Blog posts

- Create the CRUD routes for blog posts.
- Your routes should only return json.
- Take turns sage/scribe each route.
- Make sure to test a route using postman or ajax requests before moving on to the next route.

- Check in your code

5. CRUD Blog comments

- Create the CRUD routes for blog comments.
- Your routes should only return json.
- AGAIN, take turns sage/scribe each route.
- Make sure to test a route using postman or ajax requests before moving on to the next route.
- Check in your code

6. Mock Up

- Draw/Mockup each of the following pages before moving on. Be sure to take pictures of your mockups and store them in your github repo.
- You should have an exact idea of the interaction and navigation that will happen in your app before you write a single line of code.
- The website will consist of the following pages:
- Home page:
    - A list of all blog posts sorted by creation time and a 3 sentence excerpt from each.
    - A link next to each blog post that will take you to the blog post's page.
    - A link to create a new blog post
- Blog page:
    - Display a single blog post. Should show Title, Author, Body and creation time.
    - Display all comments for the given blog post. Next to each comment, display a link to delete the comment.
    - A link to edit the blog post.

    - A link to delete the blog post.
    - A link to add a comment.
- Create/Edit Page:
    - Display a form with input boxes for Title, Author, Body
    - A button to submit the creation/edit of the blog post

Take pictures of your mockups and check them into your repo.

7. Front-End

- NOTE: Your api should only be returning JSON, so you will need to make ajax requests to get the page content.
- Use bootstrap or another style framework of your choice. Plan ahead and use a grid system for your layout.
- Use gulp to automate your build tasks.
- As a team, focus on creating one page at a time, one team member should focus on layout/styles and the other team member should focus on ajax requests/DOM manipulation.
- After you have finished a page, switch focus areas for the creation of the next page.
- Communicate frequently and work efficiently.
- Commit your code often.

Stretch

1. User Authentication

- Re-factor your application to use user authentication.
- Create a users table to store your users.

- Change the author_name columns on the blog posts table and blog comments table to author_id and add a relationship to your users table.
- Restrict blog update/deletion to the author that created the blog post.
- Restrict comment deletion to the author of the post and the comment author.
- You may use passport local strategy or role your own local strategy.

2. Use a postgres DB hosted on heroku

- Familiarize yourself with how heroku works
- Follow the guide here to create a DB on heroku.
- Update your DB connection string to point to the heroku DB.

3. Deploy to Heroku

- Follow this guide to familiarize yourself with the heroku toolbelt.
- Deploy your site to heroku.