


 [SEI-CC](#) / [SEI-6-5](#) Public


<> Code

 Issues Pull requests 1 Projects Wiki Security Insig main ▾

...

[SEI-6-5](#) / [Unit_2](#) / [project-2](#) / project-2.md

JimHaff Update project-2.md

 History 1 contributor 185 lines (111 sloc) | 9.05 KB

...



GENERAL ASSEMBLY

Project 2

Node/Express/MongoDB

Full-stack CRUD Application

Overview

This second project will be your first adventure into **building a full-stack application**. You'll be **building a Node/Express/MongoDB app** from the ground up yourself.

This is exciting and by the end of this unit we will have given you the all of the tools needed to build your app.

You get to decide what you want to build - as long as it meets the technical requirements outlined below.

You will be working individually for this project. You'll be designing and coding the app yourself. However, you will have access to up to [insert number] fifteen-minute one-on-one sessions with your instructors.

Additionally as part of the project's requirements

Planning & Presentation Requirements

Planning - Due by _____:

- A [Trello](#) board with:
 - ☐ **User Stories**, each moving from left to right in the following three lists in your board:
 - Ice Box
 - Current/MVP
 - Completed

User Stories must follow the following template:

As a <user role>, I want <feature>, because <reason>.

The *reason* is optional if it's blatantly obvious.

Prioritize your user stories within the Ice Box with your "wish list" stories at the bottom.

Then move the stories that make up the MVP to the Current/MVP list (in the order that you want to implement them) leaving behind the "real" optional/ice box features.

When all of the completed stories have been moved to the Completed list, move the highest-priority Ice Box story to the Current/MVP list and implement it, etc.

- ☐ A **Wireframes** list containing wireframes for the app's main pages of functionality, e.g. Landing Page, Posts Index Page, Favorite Posts Page, Add Post Page, etc.

- ❑ An **ERD** list containing an ERD identifying the attributes of each Data Entity (one for each Model and embedded schema). The ERD also needs to diagram relationships between the Entities (1:1, 1:M or M:M). Here's a [YouTube video to show you how](#).

Project Presentations - _____:

You will have a maximum of 10 minutes to present your project following these guidelines:

1. Introduce the Project:

- ❑ Intro your project by paraphrasing the README.

2. Demonstrate the Project:

- ❑ Launch the deployed app by clicking the link in the README.
- ❑ Log out and back in to demonstrate that OAuth is working.
- ❑ Demonstrate the features of the app, including full-CRUD data operations.

3. Show/discuss your code:

- ❑ Show the "main" Mongoose model.
- ❑ Show your favorite EJS template.
- ❑ Show the controller for the main model.

4. Share the experience:

- ❑ What was your biggest challenge?
- ❑ What are your key learnings/takeaways?

5. Q & A + Feedback

Technical Requirements

Your App Must:

- ❑ **Have at least 2 data entities (data resources) in addition to the User Model** - one entity that represents the main functional idea for your app and another with a **One:Many** or **Many:Many** relationship with that main entity (embedded or referenced).
- ❑ **Use OAuth authentication.**
- ❑ Implement basic **authorization** that restricts access to features that need a logged in user in order to work (typically CUD data operations) by "protecting" those routes from anonymous users using the `ensureLoggedIn` middleware from the OAuth lesson. In addition, ensure that editing and deletion of a data resource can only be done by the user that created that data (this is done in the controller - refer to the Guide to User-Centric CRUD).
- ❑ Have **full-CRUD data operations** somewhere within the app's features. For example, you can have functionality that **Creates & Updates** a *post* and satisfy **Delete** functionality by implementing the ability to delete *comments*.
- ❑ Be styled such that the app looks and feels similar to apps we use on a daily basis - in other words, **it should have a consistent and polished user interface.**
- ❑ Be **deployed online** (Heroku).

Optionally, Your App May:

- ❑ Consume a third-party API. If you choose to implement this option, it's likely that the data from the API will be a key data resource in your app, therefore it's important to consider how to implement whatever CRUD data operations will apply. For example, how will data from the API find its way into your database? Be sure to discuss with an instructor when planning your app's features.
- ❑ Expose its own API where it returns data resources as JSON.

Necessary Deliverables

- ❑ **A working full-stack app that meets or exceeds the above technical requirements, built by you, and hosted on Heroku.**
 - **A README.md file** with these sections (here's a [basic template](#)):
 - ❑ **App Title:** Contains a description of what the app does and optional background info.

- ❑ **Screenshot(s):** A screenshot of your app's landing page and any other screenshots of interest.
- ❑ **Technologies Used:** List of the technologies used.
- ❑ **Getting Started:** Include a link to the deployed app and your Trello board with the project's planning.
- ❑ **Next Steps:** Planned future enhancements (icebox items).

Note: Don't underestimate the value of a well crafted `README.md`. The `README.md` introduces your project to prospective employers and forms their first impression of your work!

- ❑ **Daily commits (the more the better) dating back to the very beginning of the project.** Commit messages should be in the present tense, e.g., "Style landing page" instead of "Styled landing page". **Do not "start over" with a new repo.**

Getting Started

- Discuss your app idea with an instructor to get their feedback before you dive too deep into user stories and wireframes.
- Because your app's functionality revolves around the logged in user, **implement authentication and basic navigation first!**
- **Prioritize and implement the user stories one at a time** by following the [Guide to Add a Feature to a Web App](#).
- Follow the guidance and concepts in the [Guide to User-Centric CRUD](#).
- Remember to keep things small and focus on the MVP – feature creep can doom a project!

Project Idea Guidance

Lots of the web applications you interact with on a daily basis can provide inspiration for this project as most are full-stack CRUD apps. That is, they manipulate and display data.

DO NOT Choose Non-CRUD Applications Such As:

- Games
- Portfolio, or presentational pages

- Marketing or content-oriented websites

Good Examples

Some of the best apps are apps that track or manage things of **personal interest to you**:

- Music lesson tracking
- Soccer team tracker
- Rock climbing planner

So much of the Internet is CRUD apps!

- Social media:
 - Twitter
 - Instagram
 - Reddit
- Marketplaces:
 - Craigslist
 - Etsy
- Organizational or Business apps:
 - Home Inventory planner
 - Personal planner
 - Customer management
 - Payroll/Accounting

Many simple apps can have their functionality enhanced by implementing the ability of users to comment on, and/or like/favorite items.

Another piece of advice: If you choose to develop an app that has the concept of a shopping cart (eCommerce app), do not attempt to implement the actual payment functionality. Plus, here's a hint in regards to the data model: a "cart" is simply an "order" that has yet to be paid - in other words, you would only need an `Order` model vs. both `Order` & `Cart` models.

Actual Recent Student Projects

- [Insert Updated Sample projects here](#)
-
-

Project Feedback + Evaluation

- Your instructors will be using the [Project 2 Code Review](#) form to determine whether or not the project passes all of the minimum requirements.
- Your instructors will endeavor to deliver your code review to you ASAP following presentations.
- If your instructors determine that your project would pass with minor fixes, you will be required to address the minor deficiencies by 9 am the following day. Please be sure to inform your local instructor when the fixes are complete. FYI, "minor fixes" are minor items that can be fixed very quickly, like code formatting, correcting the README, etc.
- If your instructors determine that the project does not meet the minimum requirements you may request to address the deficiencies identified and resubmit the project. However, be aware that **there is only a single opportunity to resubmit a project or project assessment during the course.**
- Immediately after your presentation, your instructor and/or outcomes may provide you with feedback that will benefit your project and perhaps the projects of other students as well.
- If there is a specific section of code that you would like an instructor to provide additional feedback, please ask!