PER SCHOLAS

CP 325.1:
Planning a Project

Version 1.0, 08/14/23
Click here to open in a separate window.

Introduction

This document will introduce you to the requirements of the course Capstone Project (CP), and set you on the path toward its beginning.

Starting to think about the *what* before you know the *how* may seem daunting, but it is important to build this project just like you are going to learn the technologies required for it: from the ground up, and in small steps. The ground floor of this project is simply an idea, and you will climb the steps as you go.

We will give you the tools to build what you want, and you will decide how to build it. This level of creative freedom is intended to allow you to showcase not only the skills you develop throughout the class, but also your own unique perspectives. Have fun with it!

This is a strictly individual activity, but you may consult with others along the way. You are encouraged to ask instructors, peers, and the internet for help along the way. However, it is ***not acceptable*** to copy another person's code and submit it as your own. Own your work, and avoid plagiarism.

Capstone Objective

- Create a full-stack web application using MongoDB, Express, React, and Node (MERN).

Timeline

As the course progresses, you will be given incremental assignments that challenge you to integrate the content that you have learned into your capstone project, pushing it closer to completion over time. It is **incredibly important** to spend time on your capstone throughout the course! If you save everything until the end, you will not be able to accomplish what you otherwise could.

At each stage of your project, do not hesitate to ask your instructors or peers for input. It is very likely that your plans for the application will change over the course of time, and feedback can help validate the choices that you wish to make, or open your mind to other possibilities that you may not have considered.

While you will be given the entirety of the course to work on the project in stages, there will be **eight days** of dedicated project time at the end of the course for final tasks. Plan for this!

The complete application is due by 11:59pm on submission day, following the class time zone.

Submission

These submission instructions will only be included in this document and the final assignment document. Keep this document somewhere convenient for future reference.

When your project is finally complete, submit the link to your completed assessment using the **Start Assignment** button on the assignment page in Canvas.

Your submission should include:

- A GitHub link to the repository for your completed project.

You will also be given the opportunity to present your project to guests, instructors, and/or a Talent Advocate Manager (not the class) upon completion. Lean into this opportunity to become accustomed with speaking about your work, both from a technical and non-technical perspective.

Your presentation should include:

- A demonstration of the application.
- An overview of the challenges you endured, and how you handled them.
- A short question-and-answer period.

Your instructor will schedule practice presentations throughout the development phase.

Instructions

This first assignment charges you with planning your application from a non-technical perspective. This can include sketches of the kind of interfaces you want to build, lists of the types of information you might need to keep track of, details about the functionality or purpose of the application, etc.

It is extremely likely that your project vision will change over the course of time, as will its implementation details. Whenever you make new decisions, make sure that they adhere to the requirements outlined below.

Getting Started

Create a "Project Plan" document for your own reference. Within this document, plan what kind of application you would like to create and begin documenting the details of your vision.

Here are some suggestions for starting the planning process:

- Break the project into different components (data, presentation, views, style, and server-side work), and brainstorm for each component individually. You can tackle these components as you are introduced to them; not everything needs to be planned from the start!

- Write pseudocode. Start by stating the problems in plain text, and write down how you could solve them. This will help guide your process and help you understand the problem better once you have the coding tools to implement the solution.
- Begin with the end in mind. Know where you want to go by planning with sketches and functionality ideas so you do not waste time building things that you do not need. Make a todo list!
- Do not hesitate to write throwaway code to solve short-term problems.
- Read the documentation for whatever technologies you use. Most of the time, there is a tutorial that you can follow, but not always. Learning to read documentation is crucial to your success as a developer.
- Commit early and commit often to your git repository. Do not be afraid to break something; you can always return to a previous version.

Once you have the basics of your project planned out, get the project plan approved by your instructors. They can provide valuable feedback on scope and functionality from the perspective of somebody that is already familiar with all of the technologies involved.

As you continue building your application, ask yourself the following questions for guidance:

- Did you deliver a project that met all of the technical requirements?
- Given what the class has covered, did you build something reasonably complex?
- Did you add a personal touch or a creative element into your project submission?
- Did you deliver something of value to the end-user (not just a login button and an index page)?
- Did you follow the code style guidance, and exercise best practices?
- Did you provide an appropriate level of comments?
- Did you try to deploy your application to a public URL as a personal stretch goal?

Requirements and Rubrics

Many of these requirements will seem like gibberish at this stage; that's okay! Your goal right now is not to plan the perfect project, it is to plan what you would like to do.

As you learn technologies, reference the requirements below to ensure that you are headed in the correct direction with their implementations.

These requirements are intended to give you as much creative freedom as possible while still showcasing the skills gained during your technical learning journey.

Requirements

Your project folder should be named as follows for submission:

- LastName_FirstName_ProjectName_Capstone

(20%) Project Structure, Standardization, and Convention	**Weight**
Project is organized into appropriate files and directories, following best practices.	**2%**
Project contains an appropriate level of comments.	**2%**
Project is pushed to GitHub, and contains a README file that documents the project, including an overall description of the project.	**5%**
Standard naming conventions are used throughout the project.	**2%**
The program runs without errors (comment out things that do not work, and explain your blockers - you can still receive partial credit).	**4%**
Level of effort displayed in creativity, presentation, and user experience.	**5%**

(12%) Core JavaScript	**Weight**
Demonstrate proper usage of ES6 syntax and tools.	**2%**
Use functions and classes to adhere to the DRY principle.	**2%**
Use Promises and async/await, where appropriate.	**2%**
Use Axios or fetch to retrieve data from an API.	**2%**
Use sound programming logic throughout the application.	**2%**
Use appropriate exception handling.	**2%**

(9%) Database	**Weight**
Use MongoDB to create a database for your application.	**5%**
Apply appropriate indexes to your database collections.	**2%**
Create reasonable schemas for your data by following data modeling best practices.	**2%**

(19%) Server	**Weight**
Create a RESTful API using Node and Express. * For the purposes of this project, you may forgo the HATEOAS aspect of REST APIs.	**7%**
Include API routes for all four CRUD operations.	**5%**
Utilize the native MongoDB driver or Mongoose to interface with your database.	**5%**
Include at least one form of user authentication/authorization within the application.	**2%**

(35%) Front-End Development	**Weight**
Use React to create the application's front-end.	**10%**
Use CSS to style the application.	**5%**
Create at least four different views or pages for the application.	**5%**
Create some form of navigation that is included across the application's pages, utilizing React Router for page rendering.	**5%**
Use React Hooks or Redux for application state management.	**5%**
Interface directly with the server and API that you created.	**5%**

(5%) Presentation	**Weight**
Create a short overview of your application.	**1%**
Highlight the use cases of your application.	**1%**
Highlight the technical functionality of the application from a high-level perspective.	**1%**
Discuss what you have learned through the development of the application.	**1%**
Discuss additional features that could be added to the application in the future.	**1%**

The following section is NOT included in the requirements for this project. Completing this section is NOT required. This section will NOT negatively impact your grade if left unfinished.

This section is intended for learners looking to go the extra mile by showcasing additional skills such as project management, and optional technologies like TypeScript.

You must complete ALL other requirements to receive credit for this section; however, this extra credit will not be included if you have already received the maximum 100% grade. The extra credit can only offset points lost elsewhere.

(5%) Extra Credit	**Weight**
Adhere to Agile principles and the Scrum framework. Perform stand-up sessions (with an instructor) when possible.	**1%**
Successfully track your project using a software similar to Jira.	**1%**
Build your application primarily with TypeScript.	**3%**