TASK

Capstone Project Part I: Define Your Product

Visit our website

Introduction

WELCOME TO THE CAPSTONE PROJECT PART I!

Congratulations on making it to the first part of your final Capstone project of this Bootcamp! Think back to your first HTML websites – what progress you have made! Great job on making it to this point.

This final Capstone project will require that you completely showcase the skills that you have acquired as a full-stack web developer. You will be required to completely conceive, design, and build a full-stack web application using the MERN stack.

TWO PARTS TO THIS PROJECT

This final Capstone project is divided into two parts. In part 1 of the Capstone Project, you will plan and design your application. In part 2 you will implement, test, and deploy your application.

This first part of your Capstone Project will allow you to get valuable feedback from a reviewer. Typically, you would have various stakeholders involved in the design and planning part of the software development life cycle. However, for this project, your reviewer may be your only stakeholder. Use this part of your project to get feedback about considerations such as:

- The scope of your project (are you aiming to do too much or not enough?).
- The planned UI (what are the strengths of your design and where can you improve?).
- The planned architecture of your project. Can you foresee problems with the libraries, frameworks deployment methods, etc., that you plan to use?

This planning and feedback can help you save valuable time in the second part of this Capstone Project.

PROJECT REQUIREMENTS

You can choose to design any web application that you like as long as it meets the following requirements:

- It is built using Express, React, and MongoDB (the MERN stack).
- It creates, reads, updates, and deletes (CRUD) information from MongoDB.

- It has a custom server built using Express.

- It authenticates users using at least three passport strategies (e.g. using Google, Facebook, or a normal username and password).

- The front-end is built using React. You can use a React framework (e.g. Create React App or Next.js) of your choice.

- The application allows for normal end-user access and admin access. An administrator should be able to monitor and make changes to users' behaviour.

Need some inspiration? Here are some examples of projects:

- An application that allows doctors to track information about patients and appointments. Here normal end-users may only be able to view appointments for a certain period of time (e.g. for a day or week), whereas an administrator can make, cancel, or edit appointments and patient information.

- An application that a store owner can use to keep track of inventory. A normal end-user may be able to see when certain stock is running low and order new stock. An administrator would be able to add an item to the inventory, set the thresholds for stock (e.g. how much of a certain item to order at a time, when the new stock should be ordered, etc.), decide to no longer stock certain items, and so on.

- An application that a conference centre could use to advertise upcoming events. Normal end-users might be able to see a list of upcoming events. In contrast, an administrator might be able to add information about new events, cancel events, edit information about events, etc.

- Your own implementation of an existing app, e.g. write your own web version of WhatsApp, Twitter, or Instagram.

A note from our coding lecturer

Nizaam

Still struggling for ideas? Why not use this Capstone Project as an opportunity for making some money and getting a good reference! Can you find a real client — a friend, family member or local business that would benefit from a web application? If so, maybe you could earn some money and get your name out there. Read some tips for making money while finishing off your Bootcamp ***here****.*

SUBMISSION CRITERIA FOR PART 1

In part 2 of this Capstone Project, you will be required to write the code to implement the project that you designed in this first part.

For now, you will be required to submit the following that demonstrates your planning for this project:

1. Software requirements documentation
2. Wireframes

1. Software requirements documentation

You are required to submit some software requirements documentation. Create a file called 'readme.md' in which you document the software requirements. As shown in the image below, your documentation should describe the *system architecture* and the *system requirements specification*.

When describing the system architecture, describe the web stack that you plan to use for developing the application. Motivate your choice of architecture. E.g. How will you deploy the app? Why? Will you be using Create React App (CRA) or Next.js to create the front-end of the app. What tools have you used for styling your app? What motivated this choice?

When describing the system requirements specification, be sure to describe exactly how your application will work, who will use your application, and how they will benefit from using it. Include a number of *user stories* for your application. Also explain what other software there is that currently does something similar to what you are planning and how your software will be different (So, will it be simpler to use? Cheaper? Improve certain functionality, etc.). In addition, be sure to list all the functional and non-functional requirements.

System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.
System requirements specification	This should describe the functional and non-functional requirements in more detail. If necessary, further detail may also be added to the non-functional requirements. Interfaces to other systems may be defined.

Structure of a Requirements Document (Sommerville, 2016)

2. Wireframes

Create wireframes that demonstrate how the front-end (user interface) of your application will work. In part two of this Capstone project, you will implement the functionality of the user interface shown in the wireframes you design here.

Create a wireframe using **draw.io** or similar software.

When designing the UI, keep the following best practice guidelines in mind:

- **Keep the interface simple**: Interfaces that are simple are the best interfaces. They avoid unnecessary elements and are clear in the language they use on labels and in messaging.

- **Create consistency and use common UI elements**: Users feel more comfortable and are able to do things more quickly when you use common elements in your UI. You should also aim to create patterns in language, layout and design throughout the site to help facilitate efficiency. Once a user learns how to do something, they should be able to transfer that skill to other parts of the site.

- **Be purposeful in page layout**: You should consider the spatial relationships between items on the page and structure the page based on importance. You can help draw attention to the most important pieces of information and can aid scanning and readability by careful placement of

items. UI design should promote user efficiency. The user should also always know where they are and how to go back to the previous page to retrace their steps.

- **Strategically use colour and texture**: You can direct attention toward or redirect attention away from items by using colour, light, contrast and texture to your advantage.

- **Use typography to create hierarchy and clarity**: How you use typeface should be carefully considered. Use different sizes, fonts, and arrangement of the text to help increase scannability, legibility, and readability.

- **Make sure that the system communicates what's happening**: You should always inform your users of location, actions, changes in state, or errors. The use of various UI elements to communicate status can reduce frustration for your users. For example, a simple progress bar can give the user an indication of how long an action will take. It can also show that the system is processing so that the user doesn't repeatedly press a button because they think your system isn't responding.

- **Think about the defaults**: You can create defaults that reduce the burden on the user by carefully thinking about and anticipating the goals of the people using your system. This is particularly important when it comes to form design where you might have an opportunity to have some fields already filled out.

Compulsory Task 1

Carefully follow the instructions in this document. Create and submit:

- A file called 'readme.md' that contains the software requirements documentation as described in this task.
- Wireframes created using **draw.io** or similar software that show how the front-end (user interface) of your application will work.

Completed the task(s)?

Ask an expert to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.

REFERENCES

Sommerville, I. (2016). *Software Engineering* (10th ed., pp. 101-137). Essex: Pearson Education Limited.