A screenshot of a computer

Description automatically generated with low confidence

**Project Introduction**

**Project Summary**

This project aims to give you the opportunity to put all of the skills you have learned into one project to build your own custom travel app. Due to the nature of this course, it is very JavaScript heavy, but it is still expected you create clean and appealing HTML/CSS. You will also be targeting the DOM, working with objects, and retrieving data from 3 APIs in which one of those is reliant on another to work. Finally, this is all going to be done in a Webpack environment, using an express server, and wrapped up with service workers.

For this project, refactor and test as much as possible while you are building. You should figure for every piece of functionality you add, you will likely spend just as much time testing and refactoring your code. If it takes you 5 hours to figure out the logic, it should likely take you another 5 hours determining that you wrote the best code possible. As your skills improve, this process will feel more natural. Make sure to remove any debugging code from your final submission.

The minimum requirements ask a fair amount from you, but the final app is quite simple. A roadmap to expand on the application and make it uniquely your own is provided.

**What You Will Build:**

You will be building a travel application. It’s common to pull basic data from an API, but many applications don’t just pull the weather, they pull in multiple types of data, from different sources and occasionally one API will be required to get data from another API.

The project will include a simple form where you enter the location you are traveling to and the date you are leaving. If the trip is within a week, you will get the current weather forecast. If the trip is in the future, you will get a predicted forecast. The OpenWeather API is fantastic but it doesn’t let you get future data for free and it’s not that flexible with what information you enter; we are going to use the Weatherbit API for you to see how another API accomplishes the same goals. Weatherbit API has one problem, it only takes in coordinates for weather data -- it’s that specific. So, we’ll need to get those coordinates from the Geonames API. Once we have all of this data, we’ll want to display an image of the location entered; for this, we will be using the Pixabay API.

This may not sound like a lot, but there is a fair amount of moving pieces that rely on each other to work. You’ll need to plan out the logic of what you are trying to accomplish before you begin developing. There are a lot of paths you can take, and what you choose to display and how you display it is somewhat flexible. It is highly recommended that after you meet the minimum requirements in the rubric, you continue debugging the UX and improve the project.

**What will I learn?**

At this point, you have learned all of the technical skills necessary to complete this project. You will likely stumble along the way and find that there are some pieces you didn’t encounter in the past projects. Remember, if you get stuck, you should always look things up. Sometimes just articulating the problem renders a solution.

The following are just some of the questions that you’ll experience along the way:

* What’s the ideal workflow?
* How many files do I need?
* How do I convert one project into another?
* Should I redo the HTML/CSS first or go straight to the javascript?
* How many JavaScript functions do I need?
* Should my function be this many lines of code?
* Is readability or performance more important?

There’s no single correct answer to each question. While building this project, working with peers, and getting feedback from the project reviewer -- you will naturally develop your own answers to these questions!

**Getting Started**

**Introduction**

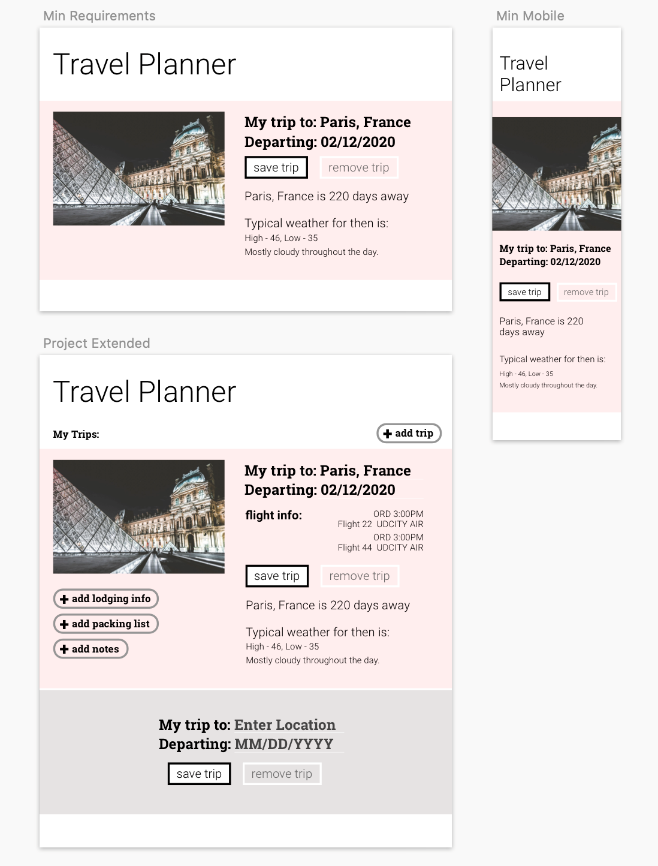
This project requires you to build out a travel app that, at a minimum, obtains a desired trip location & date from the user, and displays weather and an image of the location using information obtained from external APIs. Given that this is the Capstone project, it's highly encouraged for you to go above and beyond, adding additional functionality and customization to truly stand out with a project you are proud to have at the top of your portfolio!

**Project Rubric**

Your project will be evaluated by a Udacity code reviewer according to the Travel Planner App project rubric. Please review for detailed project requirements.

**Get the Starter Code**

If you'd like to start from scratch without any files, you are encouraged to do so! You learn the most by developing on your own! But, it can be a bit challenging having to start from scratch. There’s not the standard starter code you’re used to. If you would like to start with starter code, duplicate your code from project 3 and start there. Here is just one possibility for the visual of this project. Of course, you can style it how you choose, and extending the project will dictate the final appearance.



**Before you begin...**

Before moving forward, reacquaint yourself with project 3 & 4. After, ask yourself:

* What is the Document?
* What are events?
  + How do we listen for them?
* How can we access elements within the DOM?
* We can also access elements with getElementsByClassName(). What does this method return, and how do you use it?
* Every element has an innerHTML property that represents the markup of the element's content. How can you leverage this property to get and set content?
* What is the Fetch API and how can we use it to get data?
* What are callback functions and how do we use them appropriately?
* What is asynchronous javascript?
* How/Why do we use Express?
* How/Why do we use Webpack?
* How/Why do we use service workers?
* How to set up Unit Testing using Jest framework? You can review P4 ("Evaluate a News Article with NLP) project instructions Stage 4 to install Jest and create unit tests.

**Development Strategy**

It's very important that you plan your project before you start writing any code! Break your project down into small pieces of work and strategize your approach to each one. With these bite-sized amounts, it'll be easier to debug and fix any issues that appear.

Feel free to implement your own design workflow, but if you get stuck -- here is a walkthrough to get you up and running!

1. Start by duplicating your project 3 weather app. Once duplicated, change the new project’s name to make certain you're not overwriting your old project. We are going to build off this project as a foundation.
2. Get webpack set up to work with this project. Use the skills you learned in project 4 to get your development environment going.

* Create your src folder first. The src folder should contain a client folder and a server folder.
* Your server folder should contain your server.js content.
* Your client folder should contain a js folder, media folder, styles folder, and views folder, as well as an index.js file.
* Your application js should go into the js file, your css into styles, and your index.html into views.
* Convert your stylesheet from a .css file to a .scss file
* Remember that webpack builds a dist file. You’ll need to update your server js to access the dist folder. (Hint: app.use(express.what goes here?))
* Your index.js file inside the client folder should import the main function of your application javascript, it should import your scss, and it should export your main function from your application javascript. But in order to import, where will you need to export it?
* In project 4, you may have added your event listeners to the buttons themselves. For this project, you should be using .addEventListener(); If we are exporting functions from our application.js file, our event listeners can’t go there. Where can we put them? To call that exported function?
* Now that your src folder is set up, it’s time to get webpack going. You should already have a few dependencies installed from project 3. We need to add babel, babel loader, css loader, file loader, html loader, html webpack plugin, node sass, sass loader, style loader, webpack, webpack cli, and webpack dev server. Refer to your project 4 to see what’s there, most of these should have been in use there.
* Next, update the scripts in package.json. You will want to have test, dev, start, and build. NOTE: Start will be for your express server, dev will be so that you can take advantage of web dev server. It is possible depending on your setup to run both of these with one command.
* Get your webpack config set up. Should be fairly similar to your language processing app webpack config. If you did not use webpack dev server in your language processing app, you will want to do so here. Additionally, using source maps will help you debug your css.
* To get webpack running, you’ll want to first run npm run dev, then npm build to get your dist folder created. Once that is created you can run npm run dev and npm start simultaneously to have hot loading of your project as well as a working express environment. NOTE: If needed, reference the stripped-down version of project 3 with webpack in the starter documentation.

1. Create an account with Geonames.
2. Replace the openweather api with geonames api. You already have one working api. What information needs to get adjusted so that instead of entering a zip code, you enter a city? We want to get the latitude, longitude, country, instead of getting the temperature, feeling, and date.

* The weather data array was named differently, what do we need to change the name to?
* The weather data only had 1 object in the array, the geoname api outputs multiple objects. How do we call the first object?

1. Introduce a countdown. You’ll need to add a text field to your project to get the date.

* What type of input should it be? What about cross browser rendering?
* We’re looking to see how soon the trip is, how can you get the information from the DOM and see how soon that date is?
* Where should you be storing that data once you have it?

1. Create an account with Weatherbit.
2. Integrate the Weatherbit API similarly to how you integrated the geoname api. What information needs to get adjusted for you to pull in the future weather? Getting a CORS error? Check out this article for some options. NOTE: If you see that your app is working, but it takes several clicks to get all of the data, think of why this could be. This is possibly the most challenging part of the project. There is a major hint located in the Before you Begin section. If you’re unable to figure it out, and your app still works with a few clicks, continue working on it, it may come to you later, or you’ll get guidance from your reviewer when you submit the app.

* How does the Weatherbit API distinguish from the current forecast and future forecasts? Does the API change in any way? Hint: please refer to Weatherbit API documentation - Current Weather API and Forecast API
* How will we include the date? What format does it need to be in? How can we change it to the appropriate format?

1. Create an account with Pixabay.
2. Integrate the Pixabay API similarly to how you integrated the Geoname/Weatherbit APIs. What information are you going to submit to the API to achieve an appropriate image? What if there are no results?

* What Parameters will you want to set to pull in images?
* How will you submit your data from the location field to a Pixabay URL parameter without having spaces in the url?

1. Choose one of the items from the suggested list to add in. The items vary in complexity, but you must choose at least 1, all others are optional.
2. REFACTOR. At this point, your code should be working properly. Ideally, refactoring happens while you are developing, but as a new developer, you often don’t have the whole picture in your head to be able to do so properly. Let’s clean the project up.

* Have you run your code through a linter? We request you still follow Udacity standards when the code is complete, but running it through an eslinter is going to help you get started in refactoring.
* Are you using ES6 const and let?
* Are all your functions using ES6 arrow functions?
* Is your code DRY? Are there any pieces that would be better served as a helper function to avoid duplication?
* How is your code structured? Have you created functions for the main functionality with properly scoped variables? Starting out it’s likely that you will have a globally scoped variables on occasion until you learn more about closures and design patterns. But placing your code into functions is a great way to make your code more readable and a way to avoid globally scoped variables.
* Are your project files named in a way that makes sense?

1. Add in services workers. Refer to project 4 for guidance.

**Extend your Project Further - Roadmap/Strategy**

You'l need to implement at least one of the below in the project. If you’re going to do any of the suggested tasks, it’s recommended that you hold off on service workers until you are closer to submitting. This is a good use for comments.

* Add end date and display length of trip.
* Pull in an image for the country from Pixabay API when the entered location brings up no results (good for obscure localities).
* Allow user to add multiple destinations on the same trip.
* Pull in weather for additional locations.
* Allow the user to add hotel and/or flight data.
* Multiple places to stay? Multiple flights?
* Integrate the REST Countries API to pull in data for the country being visited.
* Allow the user to remove the trip.
* Use Local Storage to save the data so that when they close, then revisit the page, their information is still there.
* Instead of just pulling a single day forecast, pull the forecast for multiple days.
* Incorporate icons into forecast.
* Allow user to Print their trip and/or export to PDF.
* Allow the user to add a todo list and/or packing list for their trip.
* Allow the user to add additional trips (this may take some heavy reworking, but is worth the challenge).
  + Automatically sort additional trips by countdown.
  + Move expired trips to bottom/have their style change so it’s clear it’s expired.

**Version Control**

Although not a requirement, we recommend using Git from the very beginning. Make sure to commit often and to use well-formatted commit messages that conform to our Git Style Guide.

**Udacity Style Guides**

You should write your code and markup to meet the specifications provided in these style guides:

* CSS Style Guide
* HTML Style Guide
* JavaScript Style Guide
* Git Style Guide

**Still Not Sure How to Begin?**

To reiterate, be sure that you are comfortable with the content from all previous coursework, especially your weather and language processing apps; you should keep those handy to reference through the project.

A note on plagiarism: Viewing someone else’s code to get a general idea of implementation, then putting it away and starting to write your own code from scratch is okay. Please do not copy someone's code, in whole or in part. For further details, check out this guide regarding plagiarism.

**Project: FEND Capstone - Travel App – Rubric**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a project

Description automatically generated with low confidence**