PROJECT PROPOSAL

Exploring National Parks:
A web-based service educational and planning assistant for US National Parks

Sarina Curtis, Daniel Eap, Jason Hankins, Erin Ross

Adding 4 new functionalities:
Hirab Abdourazak, Noel Chacko, Shayna Odle, Jennifer Vaughn

GitHub repository: https://github.com/cis3296f24/Exploring-National-Parks-Noel-Hirab-Shayna-Jennifer
Project board: https://github.com/orgs/cis3296f24/projects/85/views/1

Table of Contents

Project Proposal

Project Abstract

This document proposes Exploring National Parks, a web application devoted to both assisting users in learning about US National Parks and in planning a trip to a national park, through three services: a park search service that allows users to input activities and get a list of parks back that offer that activity, a park planning service that allows users to input their desired activity list, their desired park, and their dates visiting in order to receive back a recommended schedule for their trip, and a park info service that provides more information on a specific park (e.g. hours, contact, location, etc.).

High Level Requirement

Describe the requirements – i.e., what the product does and how it does it from a user point of view – at a high level. (You can include screenshot mockup of the interface)

This document proposes a novel web application to revolutionize how users interact with information from the National Parks Service, with a combined goal of increasing awareness of activities offered in the parks and helping individuals plan park visits. When the application is visited, users have the option of either pre-selecting a National Park or selecting a variety of search terms. If the first option is selected, the user will be fed a suggested list of activities at that park, catered based on weather conditions, favorited activities, upcoming events, and so on. If the second option is chosen, the service will suggest National Parks based on the user's desired activities (chosen through a 'tag' system, with options including "hiking", "astronomy", "living history", etc.).

Conceptual Design

Describe with text (and maybe UML diagrams) the initial design concept: Hardware/software architecture, programming language, operating system, etc.

No additional hardware is required for this application beyond a computer and access to the internet, and the project should work on any operating system on which normal browsers, such as Google Chrome or Microsoft Edge, can be run. The project depends on calls to two external APIs: the National Parks Service API and the National Weather Service API. The former (NPS API) offers access to event, facility, news, activity, and alert data from the National Park Service on locations that are under their jurisdiction. The National Weather Service (NWS)'s API can be used to access weather data such as forecasts, observations, and alerts for given locations. The software architecture will depend primarily on ReactJS for handling API calls and allowing for a dynamic user interface (this is overall a JavaScript project, so React will be the main technology used). HTML and CSS will be used for building the basic structure of the website and adding design elements to ensure navigating through the website is both straightforward and aesthetically pleasing for the user. If there is extra time to implement additional features (such as a log-in system that allows a user to create an account to store information about their planned trip), a Java back-end may also be used.

In terms of design, the project will get the outputs of these two APIs (retrieved through JavaScript API calls) which will be parsed, combined, and utilized by the app's software logic to make recommendations to the user of events, locations, or activities in the National Park of their choosing or one suggested by the app. For example, if the weather is supposed to be mostly cloudy one night, the user would not be suggested "astronomy" but rather an activity like "Shopping" or "Wildlife Watching" that does not require clear skies. Alternatively, if the weather is supposed to be below freezing, the user would be suggested activities that fit with experiences such as "Skiing" or "Snowshoeing," rather than "Swimming" or "Golfing"

Proof of Concept

Link to a GitHub repository (created with the github classroom link in the Canvas assignment). In this repository include code using the same programming language and operating system and APIs needed to demonstrate the tools are going to work together. The code does not need to be extensive, but it needs to compile and run. It is ok if the code is straight from a tutorial if it is compiling and running. Include a README.MD in the main directory with instructions on how to run and compile the code.
If your project is as contribution to an open source, you need to show you can modify, compile and run the source code of the project. Simply create a README.MD with instructions on how to do it. Specify the operating system used, the compiler used and a code modification you have done.

Include a link to a public git repository. This repository include code using the same programming language, framework, libraries, and APIs needed to demonstrate the tools are going to work together. The code does not need to be extensive, but it needs to compile and run. It is ok if the code is straight from a tutorial as long it is compiling and running.

In this document or a readme file in the repository, include instructions on how to compile and run the code. Specify the operating system version and the version of the compiler to used build the project.

If your project is as contribution to an open source, you need to show you can modify, compile and run the source code of the project.

Link to initial proof of concept public git repository:
https://github.com/eross04/SoftwareDesign_02_Ross_Proposal

Link to new proof of concept for Scrum Pregame:
https://github.com/cis3296f23/Exploring-National-Parks

Link to another public git repository that demonstrates use of the NPS API, CSS, JavaScript, and HTML:
DC-Practice/FrontEndProject at main · Lasseignejk/DC-Practice · GitHub

Instructions to how to compile and run the new proof of concept:
In order to launch a local version of the website, a user must download all of the files off of the public GitHub repository/clone the GitHub repository.

Then, they must download Node.js for their system. The different installers can be found here: https://nodejs.org/en/download/current.

The user should ensure that both **npm** and **Node.js** are now on their local machine using the following commands in the terminal command line:

1.) **npm -v**
   a.) Checks the installed version of **npm**
2.) node -v
   a.) Checks the installed version of **Node.js**

Next, the user should run **cd exploring-national-parks** (lowercase) in order to set the current directory as the one with the necessary ReactJS files.

Lastly, the user should run **npm install** to install necessary packages and dependencies, followed by **npm start** in order to open the app in a new tab with a localhost server.

Furthermore, if a user plans to run the website locally, they should insert their own individual NPS API Key in place of the default key currently being used in the API call to the /Activities endpoint located in the NPSAPI.js file:
(`https://developer.nps.gov/api/v1/activities?api_key=PUT_KEY_HERE`).

The ReactJS code has been run on a Windows 11 machine with **npm** version 10.2.0 and **Node.js** version 21.1.0.

Alternatively, a running version of the proof of concept can be seen on this deployed GitHub pages: https://cis3296f23.github.io/Exploring-National-Parks/

Background

The background will contain a more detailed description of the product and a comparison to existing similar projects/products. A literature search should be conducted and the results listed. Proper citation of sources is required. If there are similar open-source products, you should state whether existing source will be used and to what extent. If there are similar closed-source/proprietary products, you should state how the proposed product will be similar and different.

The goal of this project is to make the US National Parks and their activities more accessible to those who do not have a lot of outdoor or traveling experience. By having options available to both find parks that fit the criteria of a desired trip and suggest activities for a single chosen park, the app serves both as a learning and planning service, giving inexperienced travelers tools to find the activities, events, and locations desirable to them and feasible given a day's conditions (e.g. closures and weather events). However, it does this in a more controlled environment than other outdoor activity apps like AllTrails, which novice hikers, bikers, etc. may find overwhelming at first. AllTrails also allows users to find trails on which to hike, bird watch, bike, ski, or snowshoe, but the majority of the onus falls on the user to know what they are looking for in terms of activities and location, sift through other users' comments to find closure or weather issue notices, or commit to paying a regular subscription fee to access specific guides to National Parks. Like AllTrails, the app from this project will introduce users to available trails and outdoor experiences. However, it does so without a paywall and through a more focused lens: by narrowing in on the National Parks, users do not have to search through other less traveled parks, areas, or hikes in order to find a location to complete their desired activities. Also, it provides the user information on closures or other alerts automatically without them needing to rely on other individual users to gather this information.

The app's feature allowing users to find a park based on a specific list of criteria they provide (through a search or search term "tag" function), is similar to a webpage on the NPS website that allows a user to input the state in which they want to search, the activities they wish to complete, and different topics (like the "American Revolution" or "animals") they are interested in. Parks fitting those criteria are returned and then can be clicked on to get more information about that specific national park. The Exploring National Parks project will differ from this preexisting service, as the NPS application is just one page of their diverse and varied website where this project acts as a single centralized location to gather that same information, in addition to providing the service to plan their trip activities for a specific park as well. This helps novice explorers complete their National Park journey from beginning (discovering what is available) to end (receiving activity suggestions based on inputted criteria and outside factors) while avoiding the potential overwhelming nature of having to search through the entire NPS website to find the service for which the user is looking.

There is also a pre-existing public git repository for a web site project called Park-a-pedia that has some similar functionality to what this project will offer, including using the NPS API to find parks and share their information (e.g. activities available, alerts, phone number, etc.). However, as the plan for this project is to create a new application rather than build upon an existing one, the Park-a-pedia repository only serves to prove that the technologies intended for use on this project are able to be combined together into a functioning system. This web site project could serve as a resource for JavaScript and possible NPS API offerings as well, but no exact code should be taken and used for this project from the repository. Also, while Park-a-pedia offers a similar search function for National Parks as this project will, it does not offer the same planning functionality to take the gathered information and make suggestions to the user of potential events and activities to take part in at a specific park.

Required Resources

Discuss what you need to develop this project. This includes background information you will need to acquire, hardware resources, and software resources. If these are not part of the standard Computer Science Department lab resources, these must be identified early and discussed with the instructor.

This project does not require much in terms of required resources or knowledge outside of access to and comfortability with JavaScript (specifically, ReactJS) and HTML/CSS (for structure/styling). A NPS API access token can be requested by each user if they wish to run the website locally, but requesting this token is a free, instantaneous process. All that must be done on the user's part is navigating to the NPS page to "Get Started with the NPS API" and filling out a form with their first name, last name, email, and how the API will be used (this final field is optional) in order to receive their key.

Given time or desire, an additional resource that may be used for this project is Google Maps, as integration with Google Maps would allow for additional filtering of parks based on location
(and distance from the user) as well as offering a method to "pin out" locations to be visited in a park on a map to provide an overview for navigation. One possible method for implementing this functionality would be utilizing the Google Maps Platform, an offering from Google Cloud, which has API endpoints for directions, maps, and routing. However, this may require transitioning the project architecture onto Google Cloud and/or for participants to create a Google Cloud account and project with a payment method enabled (as although Google advertises that 28,500 maps can be loaded a month free of charge, creating a project on Google Cloud still requires a billing method to be selected). An alternative platform for creating custom maps for the website would be by exploring the use of the ArcGIS Maps SDK for JavaScript; this service, while being free, still requires the user to make an account and is less ubiquitous than Google Maps, meaning a larger potential learning curve for both the developers integrating the map into the web service and for users interacting with the map on the website.
Also, if it is decided that a method of creating an account is to be implemented (again, given time and desire), a Java back-end, potentially that can connect to a database, may be needed.

Project Design

Vision

FOR inexperienced outdoor travelers WHO wish to explore the opportunities of and plan a trip to a US National Park, THE Exploring National Parks is a web-based service THAT provides suggestions of activities, park locations, and events for visitors based on their inputted interests and outside factors (weather, closures, etc.).

UNLIKE other outdoor activity planning applications, such as those offered by the National Park Service or apps like AllTrails, the Exploring National Parks app has the singular focus of helping users plan their trips from beginning to end without needing prior knowledge of where to look on a website or what factors to look out for to ensure a satisfactory National Parks experience. Our product streamlines the process of learning about potential National Parks travel locations and offering planning suggestions of what activities to do at these parks on a given day, so no prior knowledge of possibilities or activity blockers (e.g. bad weather) is expected of the user.

Added Features Vision:

For those who wanted to keep track of their planned trips, the website now includes a registration page, which allows the user to add their email and password. Later on, this will allow the user to access their previously saved planned trips as well as parks they have saved/recently searched. For inexperienced travelers or any traveler that has not visited a specific park, the Park Planner page will now give you the location of the visitor centers in that park, along with the address, and also display events that will occur during planned trip.

Persona Sally, a concert violinist

Sally, age 38, is a violinist in the New York Philharmonic. Currently living in Tribeca with her husband Sam (an accountant) and their two daughters, age 10 and age 6, Sally has lived in New York City her whole life: she grew up in Brooklyn with her father (a journalist), attended Juilliard in Manhattan to receive her Bachelor of Music and Master of Music in Violin, and performed in various New York City-based orchestral ensembles before joining the Philharmonic.

Sally does not consider herself to be technically savvy, as she always preferred to hand-write assignments when she was in school, and she prefers to use hard copies when reading books or music. She uses social media to stay up-to-date on current events and stay in contact with family/friends but that is the extent of her technical know-how. She is also not knowledgeable of many outdoor activities. After her older daughter expressed a desire to go to a National Park on vacation after learning about them in school, Sally would like to see what park options are out there and which ones would fit her family's interests. Therefore, she is particularly interested in using Exploring National Parks to learn which parks would have opportunities for her family to do activities like swimming, hiking, and going to a museum, and she would like to do so without needing to figure out how to search through more complicated, information diverse sites or downloading apps onto her phone.

Persona Alex, a park ranger

Alex, 26, is a Park Ranger at Grand Teton National Park in Wyoming. She was born in California and spent most of her life there. She went to school for graphic design and earned a degree. Alex worked in the city of Los Angeles for a couple years before moving out to Wyoming to escape the city. She was able to continue her digital work online, but found it unfulfilling. Alex decided to become a park ranger because it felt more rewarding than her current job. Alex specializes in giving wildlife tours and rock climbing, and would like a way to know when specific events are occurring at her park. She is excited to use the scheduler feature of Exploring National Parks, as it will allow her to see a quick overview of what events are going on in her park in the near future and if any closures/weather are likely to affect her schedule.

Persona Tae, a travel agent

Tae, age 30, is someone who has always had a passion for exploring new places, which led them to wanting to pursue a degree in tourism. After completing their Tourism and Hospitality BS and their Travel and Tourism MS, they were instantly hit with offers from travel agencies. While only two years of education is required to become a travel agent, Tae was just extremely excited about learning about the travel business. After finishing up their masters, they accepted a job offer from the top travel agency. As a travel agent, they specialize in curating personalized trips for their clients, which requires many different tools and platforms. In their spare time, Tae enjoys messing around with old tech and buying the latest tech. Tae hasn't gone to one travel agency conference without discussing their new gadget. Although Tae is well versed in technology, their knowledge of national parks is limited to the top five national parks to visit, even as a travel agent with 6 years of experience, and they would love for a more efficient way to create itineraries for their clients that are interested in exploring what national parks have to offer. Tae truly loves being able to create personalized travel experiences that are tailored to their clients preferences, so Tae is highly likely to enjoy the scheduler feature of Exploring National Parks, while having limited national park knowledge. Tae also looks forward to being able to cut down on research and planning time with a streamlined scheduling process for clients wanting to visit a national park.

Persona Felix, a sixth grade student

Felix is a 12 year old boy from Delaware. He often spends his time on the computer playing video games and watching YouTube videos, and has been doing so for quite a while. His parents allow him to do this because he gets passing grades in school, so he always did what was required in order to do these activities he enjoys. He isn't the most studious, but does the work nonetheless. Recently, he just started sixth grade and is starting to get homework that requires a lot more time than elementary school required. Realizing this, he notices he might have to work a little harder to get a passing grade. He was tasked by his environmental science teacher to write a short paper about national parks. Felix doesn't like to read from physical books, so he decided to use the internet to do his research. However, he has never done academic research on the internet before, all he knows is that he wants to not spend too much time going to different websites to try and find information. He wants to get the necessary information that is easy to understand to help him write his paper and be able to find it without going through a lot of trouble. Although, he admits that websites with good visual design help him focus a lot better when reading and consuming information on the website. He is interested in using the Exploring National Parks as a "one stop shop" to see clear information about different national parks, including their activities available, their open hours, their locations, their events, etc.

Risk Table

Categories -
PS - Product Size
BI - Business Impact
CC - Customer Characteristics
PD - Process Definition
DE - Development Environment
TR - Technology to be built
ST - Staff

Risk	Category	Probability	Impact	RMMM
Size estimate may be significantly low	PS	70%	2	**Mitigation -** Ensure we have a clearly designed project scope from the outset, so we know exactly what we want to implement, can discuss if the size of the project is feasible within our timeline, and can prioritize project functionality we deem necessary **Monitoring -** Keep track of product backlog size and number of user stories that are completed each week by team members to see if completion levels are not where they need

				to be in order to finish all tasks by the project deadline **Management** - Contingency plan includes relying on prioritization conversations from the beginning of the project to decide what functionalities should be kept/focused on versus what can be treated as "nice to have" and only implemented if there is time
Larger Number of users than planned	PS	5%	3	**Mitigation** - As a given key for the NPS API has a certain call limit (1000 requests per hour), make sure to avoid publishing/advertising the website to large groups that would exceed that number of requests rapidly **Monitoring** - Utilize a service such as Google Analytics that can be embedded within

				our GitHub Pages site (to which our web service is deployed) in order to track website traffic information as an estimate to the number of users **Management** - Contingency plan includes requesting a new API key with a different email (to start API count limit over from the beginning) and updating the deployed website quickly
Less reuse than planned	PS	30%	2	**Mitigation** - Work into the design plan where reuse is expected ahead of time to ensure that development by each team member remains collaborative and does not occur independently, with each member “reinventing the wheel” for every feature **Monitoring** - Use PR code reviews to

				ensure that reuse is occurring where expected so that if it is not, the team can have conversations about why that is the case or why reuse was not possible where it was expected **Management** - Contingency plan includes relying on prioritization conversations from the beginning of the project to decide what functionalities should be kept/focused on versus what can be treated as “nice to have” and implemented if there is time; also, contingency plan includes being willing to refactor code to allow for more reuse elsewhere or to implement reuse where it was missing (especially if future implementations depend on the reuse to utilize a

				common feature) in order to save time on later tasks
Hit limit on allowed number of API calls	DE	45%	3	**Mitigation** - Do not make unnecessary API calls, only make call to update the webpage once the user has indicated they are ready (i.e. they hit "Search" button after selecting their desired options) **Monitoring** - Keep track where in the architecture API calls are being made, and carefully watch any locations that have the possibility for frequent, repetitive API calls (such as where API calls are dependent on user input); also, utilize the X-RateLimit-Limit and X-RateLimit-Remaining HTTP headers that are returned in every NPS API response to check that usage is not exceeding

				what we have available **Management** - Contingency plan includes requesting a new API key with a different email (to start API count limit over from the beginning) and deploying the change quickly
Lack of training on tools	DE	30%	2	**Mitigation** - Establish expectation early on into the project what tools/framework s will be used so that team members can start to grow their comfort level with them before needing to use the tools for more complicated tasks as the project progresses; also, utilize knowledge of team member(s) that are more familiar with certain tools than others so they can help train/teach others the specific aspect of the

				tools that will be needed for this project **Monitoring** - Ensure all the team members stay in contact so that if someone is lost on how to use a given tool, it is known before a deadline **Management** - Delegate team member(s) time for a given sprint to helping the team member that is lacking training or knowledge on a tool (lost time in one sprint may save time in future sprints)
Final project requirements are different than expected	PS	5%	3	**Mitigation** - Conduct regular meetings with the project team to discuss and clarify requirements. Utilize change control processes to manage any modifications to requirements **Monitoring** - Implement version control for project documentation to track changes.

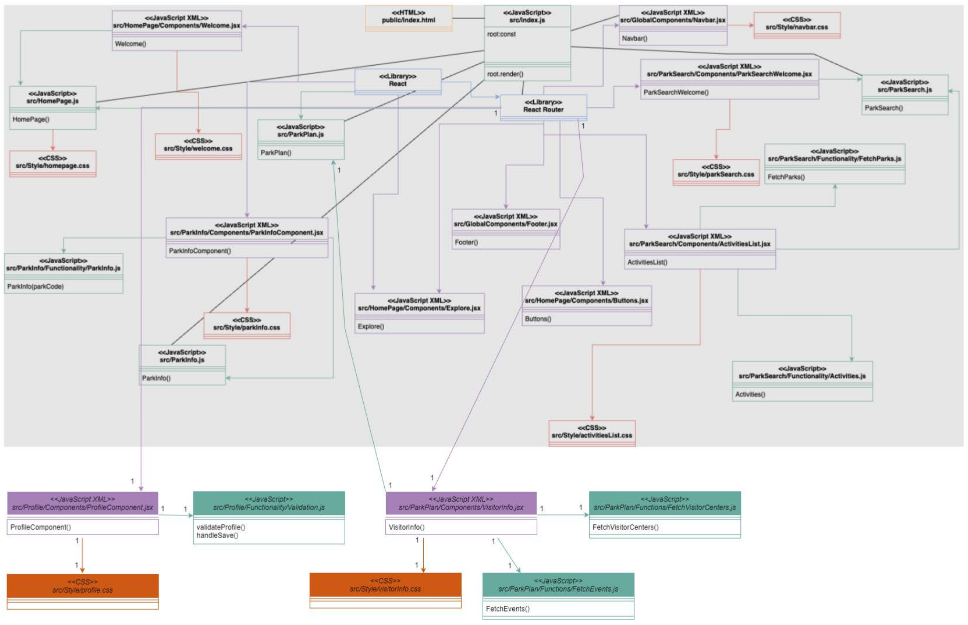
				Management - Establish a protocol for immediate communication and resolution if discrepancies are identified.
End users resist system	BI	2%	4	**Mitigation -** Implement a user-friendly interface based on feedback to enhance user acceptance **Monitoring -** Utilize surveys and feedback loops to gauge end-user satisfaction. **Management -** Establish a communication plan to keep end users informed throughout the implementation.
Delivery deadline will be tightened	BI	5%	1	**Mitigation -** adjust project scope as needed to meet tighter deadlines. **Monitoring -** Regularly review project timelines against milestones. **Management -** Establish clear communication channels for

				real-time updates on project progress
Technology will not meet expectations	TR	40%	3	**Mitigation -** Implement a pilot phase to test and validate the chosen technology. **Monitoring -** Regularly assess technology performance against predefined benchmarks. **Management -** Regularly update technology components to align with industry standards.
Funding will be lost	CC	0%	N/A	**Mitigation** - We have no funding to lose **Monitoring** - We have no funding to lose **Management** - We have no funding to lose
Staff turnover will be high	ST	5%	1	**Mitigation -** Conduct regular surveys to gauge morale. **Monitoring -** Track satisfaction metrics and turnover rates. **Management -**

				Foster a positive work culture
Customer will change requirements	PS	30%	2	**Mitigation -** Implement a change control process to manage and document alterations. **Monitoring -** Regularly review project documentation to identify and address any discrepancies. **Management -** Establish a communication plan to keep customers informed of project progress and changes.
Staff inexperienced	ST	10%	2	**Mitigation** - Establish expectation early on into the project what tools/framework/ languages will be used so that team members can start to grow their comfort level with them before needing to use the tools for more complicated tasks as the project progresses; also,

				utilize knowledge of team member(s) that are more familiar with certain tools than others so they can help train/teach others the specific aspect of the tools that will be needed for this project **Monitoring -** Ensure all the team members stay in contact so that if someone is lost on how to use a given tool, it is known before a deadline **Management -** Delegate team member(s) time for a given sprint to helping the team member that is lacking training or knowledge on a tool (lost time in one sprint may save time in future sprints)

Class Diagram


<<JavaScript XML>> src/HomePage/Components/Welcome.jsx
Welcome()
<<HTML>> public/index.html
<<JavaScript>> src/index.js
root:const
root.render()
<<JavaScript XML>> src/GlobalComponents/Navbar.jsx
Navbar()
<<CSS>> src/Style/navbar.css
<<JavaScript XML>> src/ParkSearch/Components/ParkSearchWelcome.jsx
ParkSearchWelcome()
<<JavaScript>> src/ParkSearch.js
ParkSearch()
<<Library>> React
<<Library>> React Router
<<JavaScript>> src/HomePage.js
HomePage()
<<CSS>> src/Style/homepage.css
<<CSS>> src/Style/welcome.css
<<JavaScript>> src/ParkPlan.js
ParkPlan()
<<CSS>> src/Style/parkSearch.css
<<JavaScript>> src/ParkSearch/Functionality/FetchParks.js
FetchParks()
<<JavaScript XML>> src/ParkInfo/Components/ParkInfoComponent.jsx
ParkInfoComponent()
<<JavaScript XML>> src/GlobalComponents/Footer.jsx
Footer()
<<JavaScript XML>> src/ParkSearch/Components/ActivitiesList.jsx
ActivitiesList()
<<JavaScript>> src/ParkInfo/Functionality/ParkInfo.js
ParkInfo(parkCode)
<<JavaScript XML>> src/HomePage/Components/Explore.jsx
Explore()
<<JavaScript XML>> src/HomePage/Components/Buttons.jsx
Buttons()
<<CSS>> src/Style/parkInfo.css
<<JavaScript>> src/ParkInfo.js
ParkInfo()
<<JavaScript>> src/ParkSearch/Functionality/Activities.js
Activities()
<<CSS>> src/Style/activitiesList.css
<<JavaScript XML>> src/Profile/Components/ProfileComponent.jsx
ProfileComponent()
<<JavaScript>> src/Profile/Functionality/Validation.js
validateProfile()
handleSave()
<<JavaScript XML>> src/ParkPlan/Components/VisitorInfo.jsx
VisitorInfo()
<<JavaScript>> src/ParkPlan/Functions/FetchVisitorCenters.js
FetchVisitorCenters()
<<CSS>> src/Style/profile.css
<<CSS>> src/Style/visitorInfo.css
<<JavaScript>> src/ParkPlan/Functions/FetchEvents.js
FetchEvents()
1

The class diagram above provides a view of the relationships between the key components in the *Exploring National Parks* website. The class diagram reflects the website's user interface through a collection of components, created using JavaScript XML files. These components, including ParkInfoComponent, NavBar, Footer, Buttons, Explore, Welcome, ParkSearchWelcome, and ActivitiesList, are each associated with specific functionalities.

Moving beyond the user interface, the class diagram also represents the functional components responsible for implementing the website's logic. These components, HomePage.js, ParkInfo.js, ParkPlan.js, ParkSearch.js, Index.js, Activities.js, and FetchParks.js, are linked to their corresponding user interface components.

In addition to the user interface and functionality components, the class diagram incorporates styling files, such as HomePage.css, NavBar.css, ParkInfo.css, ParkSearch.css, Welcome.css, ActivitiesList.css, and visitorInfo.css. These styling files are associated with their respective user interface components. Keeping the styling separate from the user interface and functionality components allows for easy modification of the appearance without cluttering or impacting the logic.

As for the specifics of each file, Index.js serves as the entry point for the web application, which utilizes the javascript library, React. Index includes the setup for the website and imports the components that make up the user interface. Homepage.js holds the code for the main landing page of the website. ParkInfo.js holds the logic for displaying detailed park information through an API call to the National Parks Services 'parks' API.
ParkInfoComponent.jsx, a React component, renders the detailed information to the user interface. ParkPlan.js, while not fully implemented during this sprint, is intended for creating detailed plans after a user selects a park. ParkSearch.js is dedicated to handling searching functionality, allowing users to input preferred activities and receive a list of matching parks.

ParkSearchWelcome.jsx, a react component, renders a welcome message and short description on the park search page. Activities.js handles the logic associated with fetching the list of available activities, through a call to the National Parks Services 'activities' API. ActivitiesList.jsx, a react component, is responsible for rendering the list of activities a user can choose from when searching for parks. FetchParks.js handles the logic related to fetching parks for ParkSearch. Footer.jsx, NavBar.jsx, Buttons.jsx are responsible for rending the footer, navigation bar, and the homepage buttons to either go to park search or plan a trip, based on a selected park. Welcome.jsx and Explore.jsx are responsible for rendering welcome information for the user on the homepage.

Sequence Diagram

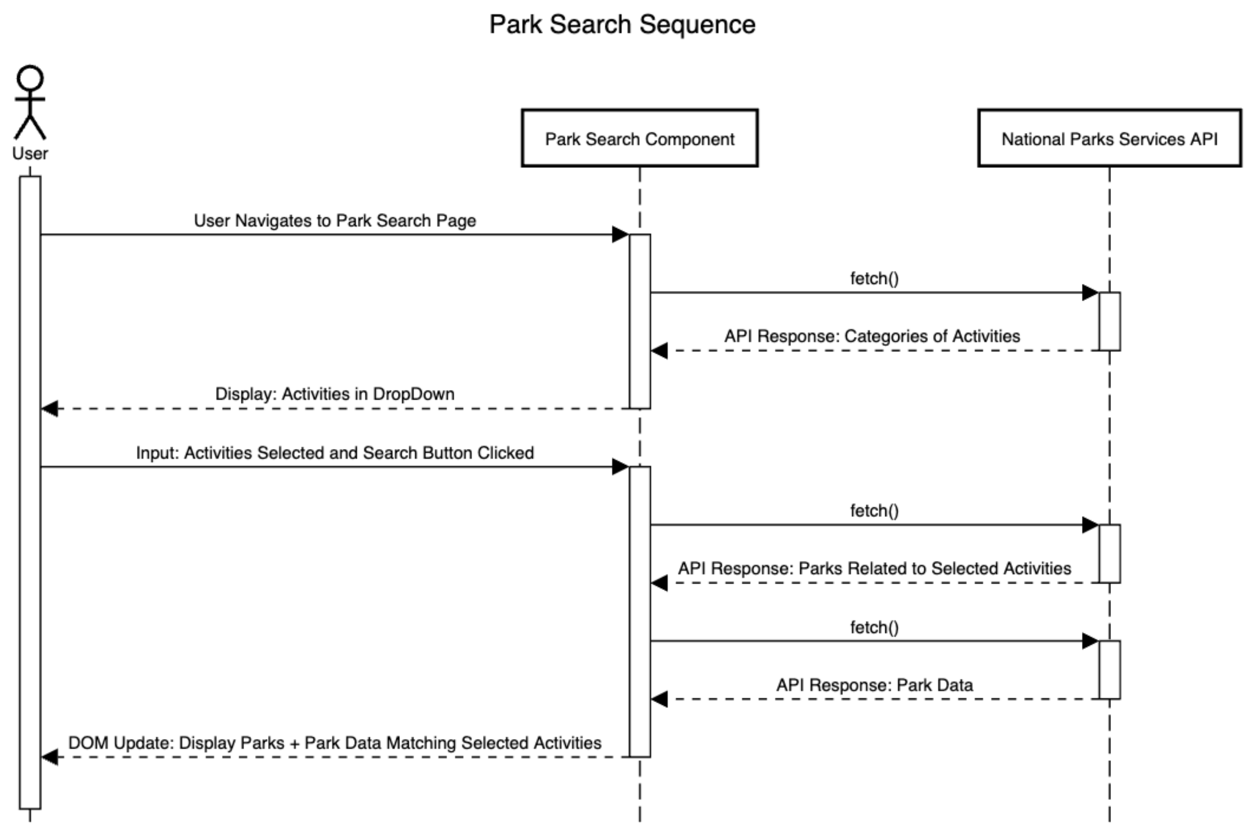

Park Search Sequence
User
Park Search Component
National Parks Services API
User Navigates to Park Search Page
fetch()
API Response: Categories of Activities
Display: Activities in DropDown
Input: Activities Selected and Search Button Clicked
fetch()
API Response: Parks Related to Selected Activities
fetch()
API Response: Park Data
DOM Update: Display Parks + Park Data Matching Selected Activities

Figure S.1

Figure S.1: The Park Search Sequence Diagram above illustrates the interactions between a user, the park search component, and the National Parks Service's API. The diagram begins with the actor/user navigating to the park search page on the website, which activates/renders the park search component. The component then initiates a call to the National Parks Service's API by calling the fetch() method, resulting in the activation of the API. The API responds with all the categories of activities, which are then displayed to the user in a dropdown menu by the park search component.

From here, the user is able to provide input by selecting activities and clicking a search button. The park search component is activated and makes two calls to the NPS API, the first fetching all parks related to the selected activities, and then second fetching the park data for each park returned. The API responds with park data and the park search component updates the DOM to display the parks and their data matching the user's selected activities. (The UML website does not have an option to show an asynchronous call with the arrowhead, but all three fetch() calls are asynchronous)

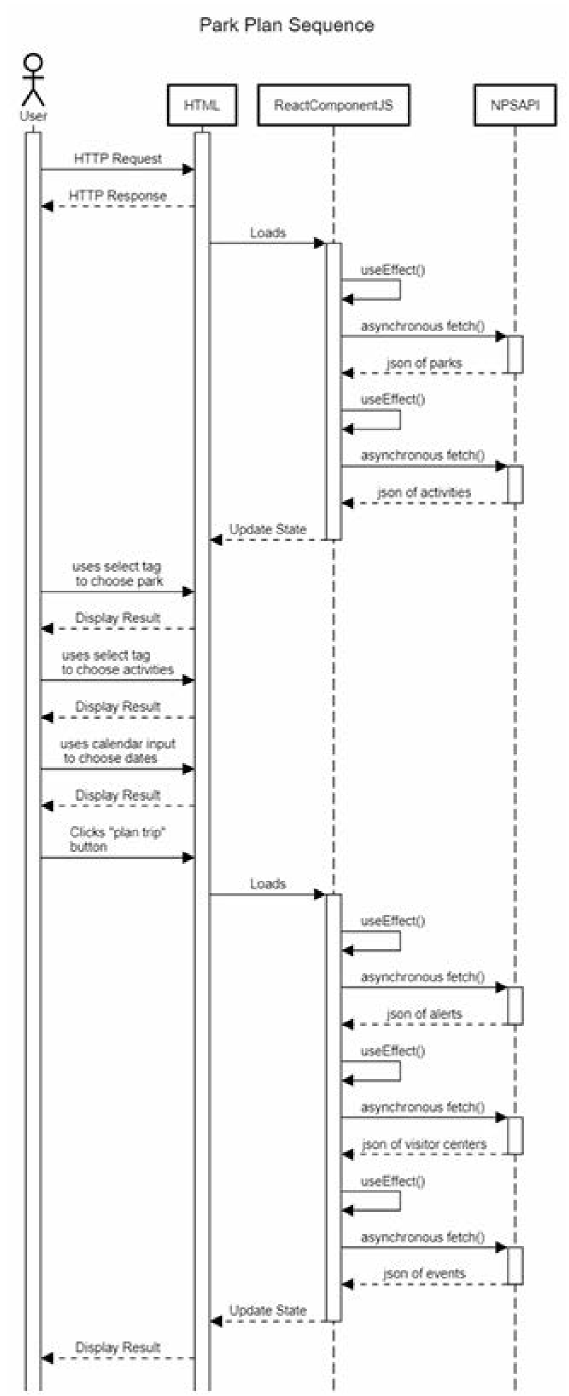


Figure S.2

Figure S.2: This sequence diagram represents a user interacting with the website and how the website interacts with its frontend components and connecting with the related API. The user navigating to the page calls an HTTP request to the server that hosts the HTML page and gets a response. The HTML page begins to render by loading the graphic related components from

React. React then calls its javascript rendering function useEffect to make asynchronous calls to the NPS API which returns the relevant JSONs from the fetch. Since it is asynchronous, React updates the HTML for the user. Now that the HTML is updated to display the relevant graphics, the user interacts with them and the HTML updates the display to represent their actions. The user can also click a button that will reactivate React to make additional fetch calls to the API and return the JSON of alerts, visitor centers, and events, update the HTML, and update the display for the user. (The UML website does not have an option to show an asynchronous call with the arrowhead, so it is denoted as asynchronous)

Project Progress

Week 2 Progress

Sprint Goal: The goal of this sprint was to modify the website to include a homepage and a search page, where users can find national parks based on the activities they'd like to do while there.

Sprint Problems: Although many tasks were completed, there were some that were incomplete. Though these tasks were mostly of a larger size. One of the biggest hurdles was with the "Park Search - Use output of search bar to return park list" task. This became a problem as after trying to develop the code, we found that the API endpoint did not output the data that we had expected it to. Due to this, lots of time was spent trying to debug and understand the API and led to a delay in the completion of this task. Having other tasks that are not started or that are in progress is due to potentially having more tasks in the sprint than time allowed for each developer, or being blocked by requiring a task that was also not completed in order to continue your task.

Backlog Features focused on in this Sprint

- Homepage with information about the site
- Homepage button to navigate to park search page
- Homepage button to navigate to park planning page
- Park search page with information, detailing the purpose of the page
- Selecting park activities with a search bar for park search
- Selecting park activities with a drop down for park
- Selecting park activities shows activities in the search bar for park search
- Viewing selected park activity tags in search bar
- Clicking the search button to see park options matching park activities selected
- Park info page with longer description of a specific park

Tasks in Sprint	Task Status at end of Sprint	Assigned To
Homepage - Make landing page	Done	Daniel

Homepage-Creating routing/page for park planning	Done	Daniel
Homepage - Create routing/page for park search	Done	Daniel
Homepage - Add main structure	Done	Sarina
Homepage - Connect button to park search	Done	Sarina
Homepage - Connect button to park planning	Done	Sarina
Homepage - Make top menu bar	Done	Sarina
Homepage - Make footer	Done	Jason
Homepage - Overall page styling mockup	Done	Daniel
Park Search - Park List Styling	In Progress	Daniel
Park Search - Search bar for activities	Done	Erin
Park Search - Search button to signal done selecting activities	Done	Erin
Park Search - Use output of search bar to return park list	In Progress	Erin
Park Search - Overall page styling mockup	In Progress	Sarina
Park Search - About section	Done	Jason
Park Search - Background Page Styling	Done	Jason
Park Search - Click on returned park, get short description	Not Started	Jason
Park Search - Button within short description	Not Started	Jason

reroutes for park with more info		
Park Info - Retrieve park website	Done (pulled into sprint)	Jason
Park Info - Retrieve basic park info	Done	Jason
Park Info - Retrieve photos for park	Done (pulled into sprint)	Jason
Park Planning - Overall Page Styling Mockup	Done	Erin

Calculations for Sprint Velocity

Github projects do not use a built-in story point method to keep track of task size on the project board, instead relying on a pseudo-"t-shirt" size system with labels that include "Tiny", "Small", "Medium", "Large", and "Extra Large".

For our purposes, the following conversion was used to transfer from the Github projects' sizing method to a more traditional story point approach (ex. 1, 3, 5, 8, and 11):

Tiny → **1** story points
Small → **3** story points
Medium → **5** story points
Large → **8** story points
Extra Large → **13** story points

Estimated Velocity (At beginning of Sprint)	106
Calculated Velocity (At end of Sprint)	74

Week 3 Progress (OFF WEEK- SPRINT WAS COMBINED WITH WEEK 4)

Sprint Goal: SEE WEEK 4

Backlog Features SEE WEEK 4

Tasks in Sprint	Size	Task Status at end of Sprint	Assigned To

Estimated Velocity (At beginning of Sprint)	0
Calculated Velocity (At end of Sprint)	0

Week 4 Progress

Sprint Goal: The goal of this sprint was to complete the park search page (where users can find national parks based on the activities they'd like to do while there), connect the park search page to the park info page (so users can navigate to a page with more information on a specific park from the curated list returned from their selected activities), and to begin the park planning page (where users will be able to enter their desired activities, a desired park, and a timeframe to generate a individualized park visit schedule).

Backlog Features

- Clicking the search button to see park options matching park activities selected
- Clicking one of the parks that populates from the activity search gives a short description of the park
- Clicking on a "More Information" button within the shorter park description navigates to a page with a longer park description
- Selecting park from search bar
- Selecting park from dropdown
- Selecting park visit date from calendar
- Selecting park visit date from search/input bar
- Selecting park activities from dropdown for park planning
- Selecting park activities from search bar for park planning
- Selecting park activities shows activities in the search bar for park planning
- Clicking on the "Generate Schedule" button populates a schedule fitting with the selected park, activities, and days
- Park info page with longer description of a specific park

Tasks in Sprint	Size	Task Status at end of Sprint	Assigned To
Park Info - Retrieve video to imbed	Medium (5)	Completed	Sarina
Park Planning - Use dates from calendar to set up schedule structure (with morning, afternoon, evening)	Large (8)	Completed	Daniel
Park Planning - Search bar for activities with input and dropdown (for search)	Medium (5)	Completed	Sarina
Park Planning - Report relevant park closures/alerts	Medium (5)	Completed	Daniel

Park Planning - Search bar for park with input and dropdown (for search)	Medium (5)	Completed	Daniel
Park Search - Overall page styling mockup	Large (8)	Completed	Sarina
Park Search - Click on returned park, get short description (for Park info)	Medium (5)	Completed	Jason
Park Search - Button within short description reroutes to page for park with more info (for Park Info)	Small (3)	In Progress	Jason
Park Search Fix Footer so that it doesn't overlap other components	Tiny (1)	Completed (pulled in part way through sprint)	Jason
Park Planning - Create search button that indicates to use the results from activities, park, and date selection	Medium (5)	Completed	Erin
Park Search - Use output of search bar to return park list	Large (8)	Completed	Erin
Park Search - Park list styling	Large (8)	Completed	Erin
Park Planning - Go through activity list and define what is done indoors/outdoors, what can be done in cold/warm weather, etc. (investigation task)	Large (8)	Completed	Sarina
Park Info - Overall styling mockup (in Figma)	Large (8)	Completed	Jason
Park Planning - Define logic for park scheduling (investigation task)	Large (8)	Completed	Erin
Park info - Retrieve park visitation info (hours, fee, activities list)	Medium (5)	Completed	Jason

Documentation - List out sprint goal, sprint tasks, and sprint velocity	Small (3)	Completed	Erin
Documentation - First UML Sequence Diagram	Medium (5)	Completed	Sarina
Documentation - Second UML Sequence Diagram	Medium (5)	Completed	Daniel
Park Planning - Alerts styling	Medium (5)	Completed	Daniel
Park Planning - Date selection feature and return date (for search)	Medium (5)	Completed	Daniel

Calculations for Sprint Velocity

Github projects do not use a built-in story point method to keep track of task size on the project board, instead relying on a pseudo-"t-shirt" size system with labels that include "Tiny", "Small", "Medium", "Large", and "Extra Large".

For our purposes, the following conversion was used to transfer from the Github projects' sizing method to a more traditional story point approach (ex. 1, 3, 5, 8, and 11):

Tiny → **1** story points
Small → **3** story points
Medium → **5** story points
Large → **8** story points
Extra Large → **13** story points

Estimated Velocity (At beginning of Sprint)	117
Calculated Velocity (At end of Sprint)	114

Week 5 Progress

Sprint Goal: The goal of this sprint was to complete the park planning page (where users will be able to enter their desired activities, a desired park, and a timeframe to generate an individualized park visit schedule), to complete the park info page (all functionality and styling), to add a "featured parks" list on the homepage of the application, and to complete general bug fixes and improvements.

Backlog (User) Features for this sprint

- Homepage with information about the site
- Clicking the search button to see park options matching park activities selected
- Clicking on the "Generate Schedule" button populates a schedule fitting with the selected park, activities, and days
- Park info page with longer description of a specific park

Complete List of Backlog (User) Features

- Homepage with information about the site
- Homepage button to navigate to park search page
- Homepage button to navigate to park planning page
- Park search page with information, detailing the purpose of the page
- Selecting park activities with a search bar for park search
- Selecting park activities with a drop down for park search
- Selecting park activities shows activities in the search bar for park search
- Viewing selected park activity tags in search bar
- Clicking the search button to see park options matching park activities selected
- Clicking one of the parks that populates from the activity search gives a short description of the park
- Clicking on a "More Information" button within the shorter park description navigates to a page with a longer park description
- Clicking on a "Make Schedule" button within the longer park descriptions navigates to the planning page for that park
- Park planning page with information, detailing the purpose of the page
- Selecting park from search bar
- Selecting park from dropdown
- Selecting park visit date from calendar
- Selecting park visit date from search/input bar
- Selecting park activities from dropdown for park planning
- Selecting park activities from search bar for park planning
- Selecting park activities shows activities in the search bar for park planning
- Clicking on the "Generate Schedule" button populates a schedule fitting with the selected park, activities, and days
- Park info page with longer description of a specific park

Tasks in Sprint	Size	Task Status at end of Sprint	Assigned To
Park Search - Button within short description reroutes to page for park with more info (for Park Info)	Small (3)	Completed	Jason
Homepage - Create "Highlighted Parks" section	Large (8)	Completed	Sarina
General - Create Favicon for site	Small (3)	Completed	Jason
Park Info - Styling for basic park info	Medium (5)	Completed	Jason
General - Fix Refresh Issue	Medium (5)	Completed	Erin
Miscellaneous Clean up	Large (8)	Completed	Daniel
Park Planning - Call to NWS API to return weather for given location/time	Large (8)	Completed	Daniel
Park Planning - Fill in schedule with activities based on search logic	Large (8)	Completed	Erin
General - API Load Issues	Medium (5)	Completed	Daniel
Park Planning - Report weather alerts for given days if available	Medium (5)	Completed	Daniel
Park Planning - Schedule Styling	Large (8)	Completed	Erin
Park Info - Fix Zero applicable parks from return all parks	Small (3)	Completed	Sarina
Park Info - Create button to make schedule for selected park (for Park Planning)	Tiny (1)	Completed	Jason
Park Info - List pages to have all Parks	Medium (5)	Completed	Jason

Park info - Styling for park visitation information and website	Medium (5)	Completed	Jason
Park Info - Styling for park media (videos and pictures)	Large (8)	Completed	Jason
Park Info - Connect button for schedule to Park Planning page (for Park Planning)	Small (3)	Completed	Jason
Fix issues with GitHub pages/Explore other hosting options	Large (8)	Completed	Erin
Replace Footer	Tiny(1)	Completed	Daniel
Save API calls to a cache	Medium (5)	Completed	Daniel
Park Search - Add state selection capability (or distance away)	Medium (5)	Completed	Sarina
General - Refactor API Calls	Small (3)	Completed	Daniel
General - Miscellaneous Clean up	Medium (5)	Completed	Erin
Create JSDoc	Large (8)	Completed	Daniel
Individual Part of Presentation	Large x4 (8)	Completed	Daniel, Sarina, Jason, Erin

Calculations for Sprint Velocity

Github projects do not use a built-in story point method to keep track of task size on the project board, instead relying on a pseudo-"t-shirt" size system with labels that include "Tiny", "Small", "Medium", "Large", and "Extra Large".

For our purposes, the following conversion was used to transfer from the Github projects' sizing method to a more traditional story point approach (ex. 1, 3, 5, 8, and 11):

Tiny → **1** story points

Small → **3** story points

Medium → **5** story points

Large → **8** story points

Extra Large → **13** story points

Estimated Velocity (At beginning of Sprint)	158
Calculated Velocity (At end of Sprint)	158

Complete List of Backlog Features

- Park Search - Button within short description reroutes to page for park with more info (for Park Info)
- Homepage - Create “Highlighted Parks” section
- General - Create Favicon for site
- Park Info - Styling for basic park info
- General - Fix Refresh Issue
- Miscellaneous Clean up
- Park Planning - Call to NWS API to return weather for given location/time
- Park Planning - Fill in schedule with activities based on search logic
- General - API Load Issues
- Park Planning - Report weather alerts for given days if available
- Park Planning - Schedule Styling
- Park Info - Fix Zero applicable parks from return all parks
- Park Info - Create button to make schedule for selected park (for Park Planning)
- Park Info - List pages to have all Parks
- Park info - Styling for park visitation information and website
- Park Info - Styling for park media (videos and pictures)
- Park Info - Connect button for schedule to Park Planning page (for Park Planning)
- Fix issues with GitHub pages/Explore other hosting options
- Replace Footer
- Save API calls to a cache
- Park Search - Add state selection capability (or distance away)
- General - Refactor API Calls
- General - Miscellaneous Clean up
- Create JSDoc
- Individual Part of Presentation

List of Additional Backlog Features

- Park Planning - Call to NWS API to return visitor centers for a given park
- Park Planning - Call to NWS API to return events for a given park
- Park Planning - Styling for visitor centers & events
- Profile - Direct users to input email and password
- Profile - Validate user input
- News - Call to Twitter API to return the Temple News timeline