Pokédex

Team members: Jason Hunter, Duncan Britt, Simon Wickersham, Ayla Tabi, Jacey Fischer

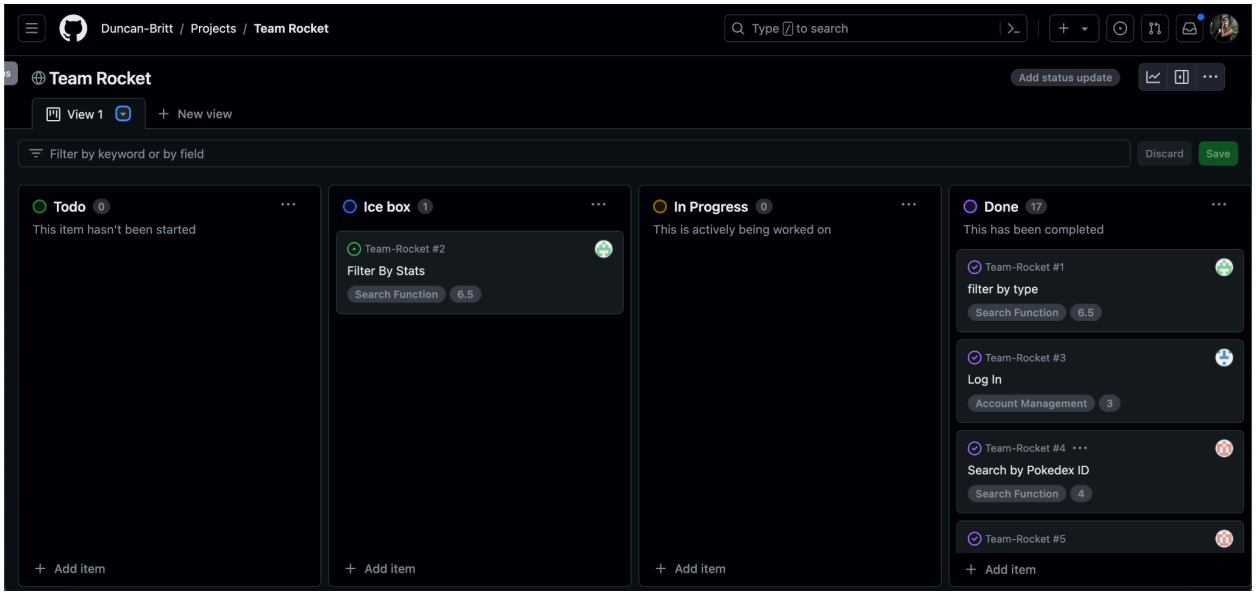
Project Description:

Pokédex is an interactive database of Pokémon with their corresponding stats and information, where users can track and maintain a personal collection and trainer profile. The application's purpose is to act as an online tracker/simulation of a real world Pokémon collection that captures the magical and fun-loving nature of the card game. Pokédex distinguishes itself from other Pokémon databases by integrating user functionality through trainer profiles as well as prioritizing stylization of the website to enhance the user's enjoyment. All viewers of the website have access to the search function, where you can look up and pull certain data about Pokémon such as type and stats. Once a user is registered and logged in, they can add Pokémon to their collection through the search function and trade cards with other users. The future scope of the project, if given more time and resources, would be to polish existing features of the website and to implement more ways for the user to interact with their online collection that stand separate from a real world collection. For example, we would hope to polish the search function by adding fuzzy search and filter by type/stats. Additionally, we would dedicate a page to tracking or simulating game play online with other users live.

Video Link: https://drive.google.com/file/d/174anFaVSNxp06gT4Kx22Q8aLxWW4M9ti/view?usp=sharing

VCS: https://github.com/Duncan-Britt/Team-Rocket

Project Tracker: https://github.com/users/Duncan-Britt/projects/1



Contributions:

Jason Hunter: I helped get some of the core skeleton of our application running with the PokéAPI via pair programming with Duncan. I worked on getting a functional navigation bar set up and implementing the page routes properly within it, including conditional displays of certain routes. I implemented the collections page where users can look at the cards they have added to their collection, as well as an interior collection's page with more detailed cards. Fixed various bugs related to card displays, partials, and added polish to the application with lots of CSS.

Duncan Britt: I designed and implemented the database schema necessary for our application's core functions, including user accounts and trading mechanics. I developed the account registration and login functionalities to manage user access. I also developed the code necessary to query and extract Pokemon information from an external API while pair programming with Jason. Additionally, I created comprehensive trading features: I implemented

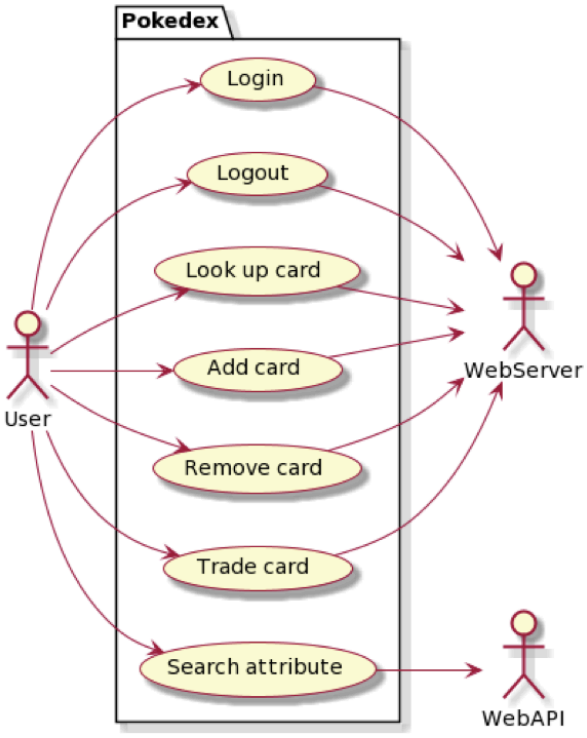
pages to browse users' collections, request trades, and view as well as respond to trade requests. I also set up the backend routes to support these trading activities.

Ayla Tabi: I designed the pokemon cards based on type. In Search.js, I isolated the first pokemon type that was returned and then created a css class (pokemon-card) based on the type. In css I stylized the card with multiple classes. I imported images for each pokemon type that will be displayed as the background of the card. Each card has 4 unique classes to make sure that they all look different. There are 7 classes that make the general template for each card as well as of this is in syles.css

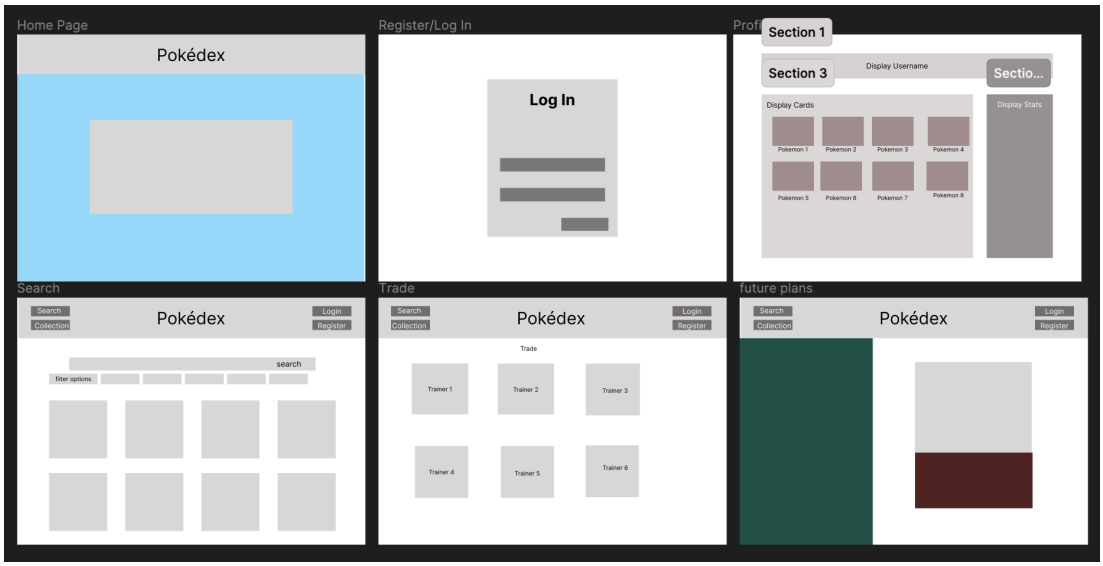
Simon Wickersham: I designed several of the initial wireframes and set up the core file structure of the site (including prototype page routes). I also helped troubleshoot the Javascript portion of the responsive card design. Additionally, I began the implementation of a more comprehensive search feature that would allow users to search Pokemon by type or stats.

Jacey Fischer: I collaborated in the initial design of the website through wireframes and mockups. I helped lay the framework for the basic styling of the website as well as designed the framework/format for the dynamically generated pokemon cards in the search and collections pages. I pair programmed with Ayla Tabi in troubleshooting in the beginning of the Pokémon cards creation as well as pair programmed with Duncan and Jason to finish functionality on the collections and search page near the end of the project. Additionally, I helped manage team meetings outside of class as well as scrum calls.

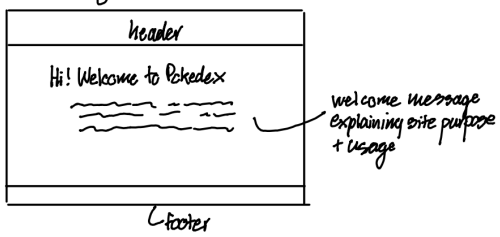
Use Case Diagram:


Pokedex
Login
Logout
Look up card
Add card
Remove card
Trade card
Search attribute
User
WebServer
WebAPI

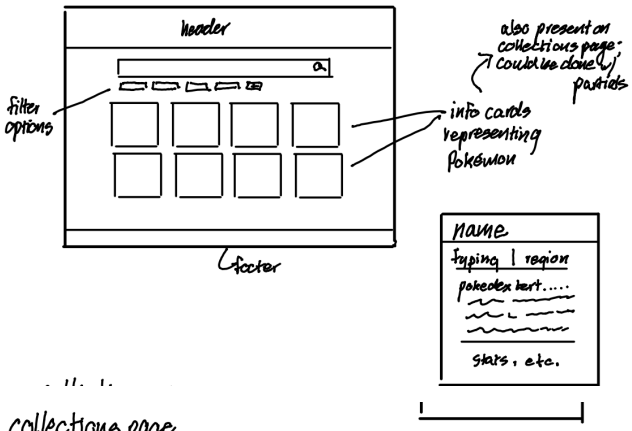
Wireframes:


Home Page
Pokédex
Register/Log In
Log In
Section 1
Section 3
Display Username
Display Cards
Display Stats
Search
Pokédex
Login
Register
search
Trade
Pokédex
Trainer 1
Trainer 2
Trainer 3
Trainer 4
Trainer 5
Trainer 6
future plans
Pokédex

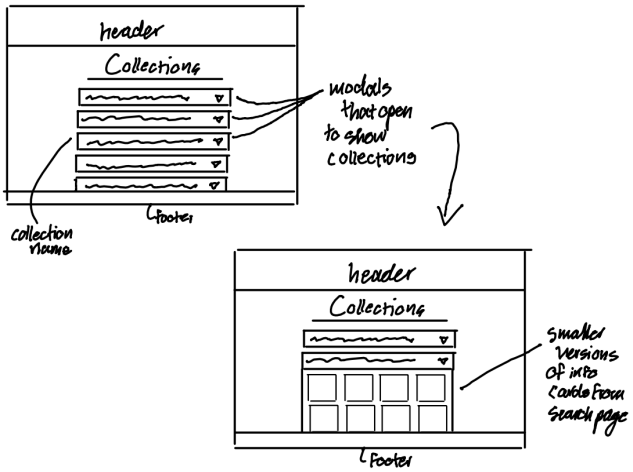
main page

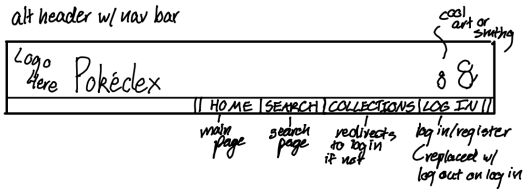


search page

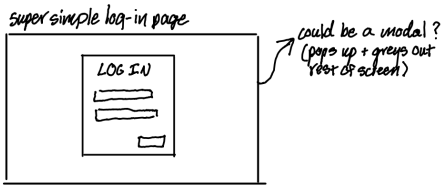


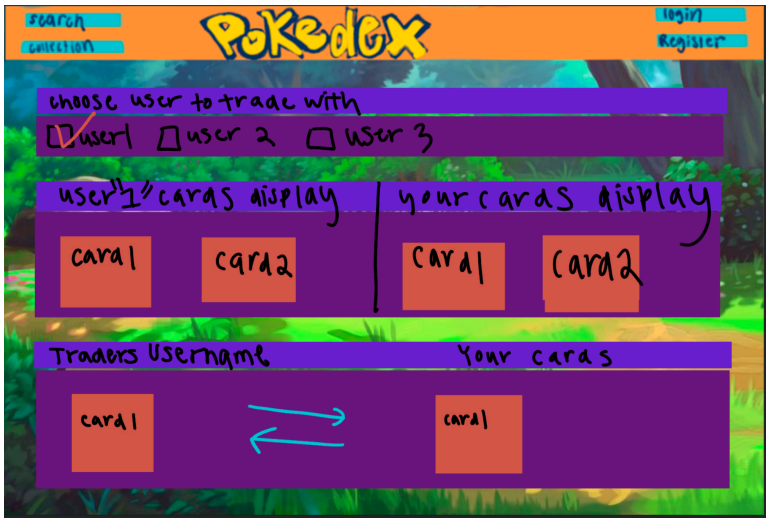
collections page

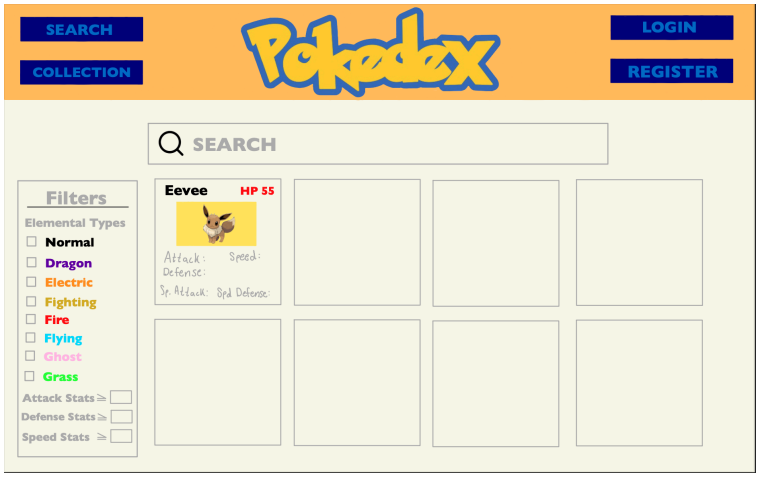



alt header w/ nav bar
cool art or something
Logo Here
Pokédex
HOME
SEARCH
COLLECTIONS
LOG IN
main page
search page
redirects to log in if not
log in/register (replaced w/ log out on log in


another alt header w/ nav bar
profile button
Pokédex
log in/out
HOME
COLLECTIONS
Search...
REGISTER
not a button, just looks neat
could be in any order?


super simple log-in page
could be a modal? (pops up + greys out rest of screen)
LOG IN


search
collection
Pokedex
login
Register
choose user to trade with
user1
user 2
user 3
user 1's cards display
your cards display
card1
card2
card1
card2
Traders Username
Your cards
card1
card1


SEARCH
COLLECTION
Pokedex
LOGIN
REGISTER
SEARCH
Filters
Elemental Types
Normal
Dragon
Electric
Fighting
Fire
Flying
Ghost
Grass
Attack Stats
Defense Stats
Speed Stats
Eevee
HP 55
Attack:
Speed:
Defense:
Sp. Attack:
Spd Defense:

Test Cases:
https://github.com/Duncan-Britt/Team-Rocket/blob/main/TeamMeetingLogs/User%20Acceptance%20Testing%20Test%20Plans.pdf

Note: These tests were conducted by users outside of the class and computer science department.

- **Test Case 1**: User looks up "eevee" in the search bar.

  Acceptance criteria: Correct pokemon is returned based on name searched

  Test Data: Pokemon within database

  Test Environment: Localhost

  Results: Eevee pokemon card is returned with all correct information.

During this test, the user started at the home page and scanned the navigation bar, finding the search page. After clicking on the search page button, they quickly recognized the search bar and typed in eevee. Once done they pressed enter on the keyboard and the eevee card was returned. This test passed because it met the acceptance criteria once the "eevee" card was returned. Notably, if the "enter" button on the keyboard had not been made to return search results the test wouldn't have gone as smoothly.

- **Test Case 2**: A logged-in user should be able to add Pokemon to their collection

  Acceptance criteria: If a user tries to add a Pokemon to their collection, the application should first check and see if the Pokemon exists in their collection, and if not, the Pokemon should be added to their collection

  Data: User credentials, Pokemon information, user collection information

  Environment: Localhost

  Results: If the Pokemon is already in the user's collection, display an error

message. If the Pokemon is not already in the user's collection, add it to their collection and display a success message

This test case utilizes multiple features of the website, including the registration, login, and search page. The user started at the home page of the website and was instructed to create a profile and then add a pokemon to their collection. The user quickly navigated to the registration page, created a profile and then was instructed to login by the application. After a few moments to register what to do next, they navigated to the login page via the nav bar. Once logged in, the user took a moment looking at the different pages of the website (via the nav bar) before coming to the search page and looking up the pokemon "eevee". Once the pokemon card was returned, the user scrolled down and clicked the add to collection button. The user was then instructed to check to see if the card had been added to their collection. The user clicked on the collection button on the navigation bar and found that Eevee had been added to their collection.

This test case was passed as the acceptance criteria was met successfully. Overall takeaways of this test case was that the application was simple enough for a user to effectively add a Pokémon to their collection, but not user intuitive at certain points. It was somewhat unclear to the user where Pokemon could be added to the collection and had to explore each page on the website to figure it out.

- **Test Case 3**: A logged-in user should be able to view the Pokémon in their collection

  Acceptance criteria: Application queries the database for Pokemon associated with a user and correctly returns the Pokemon in their collection

  Data: User credentials, associated Pokemon in database

  Environment: Localhost

Results: Correctly displays all Pokemon in a user's collection

This test case was achieved in the previous test run (test case 2) when confirming that the Eevee card had been added to their collection. Overall this was an exceptionally easy task to complete on the users end and did not highlight points of improvement for the application.

- **Test Case 4**: Trading Pokemon with another user should swap the Pokemon in both users' collections correctly

  Acceptance criteria: When two users trade Pokemon, ownership of the Pokemon the first user is trading should be transferred to the second user and vice versa. If the trade fails for whatever reason, both users' collections should remain unchanged.

  Data: Both users' credentials, the Pokemon in both users' collections

  Environment: localhost

  Results: After the trade is complete, if successful, both users' collections should be updated correctly. If unsuccessful, the application should display an error message and leave both collections unchanged.

  Feature: Search

After test case 2 and 3 had been achieved, the user was instructed to trade with another user in the database. The user then took a moment to examine the navigation bar to find the correct page for this task, finding the trade button. Once on the trade page the user took a moment to examine the features of the page and looked at multiple other profiles collections. The user then choose the player "Jess" to trade with, and selected the "trade with partner" button. Then, the user

selected cards from both hands (Wigglytuff from Jess's hand and Eevee from her own) to trade and selected "Request Trade". This concluded the test case on the first user's part, the rest of the trade was conducted by another user, logged into account "Jess". The second user successfully completed the trade by selecting the "accept trade" button on the pending trades tab.

This test case passed because the acceptance criteria was met, and the trade updated each user's collections once completed. Overall takeaway of this test case is that the trade function was relatively easy for the user to navigate through, but could be slightly improved by adding more instructions/descriptions on the page to help the user when first learning the interface.

Deployment: The application can be deployed on any web browser through localhost:3000. To access the website you must use a containerizing application and an IDE application, our team used Docker and Visual Studio Code. First, you must open the project code folder in VS code and then (with Docker open) run the command "docker compose up" in the terminal. This will make the website accessible in your web browser. You can then type localhost:3000 in your browser and the website will be fully functional.