Auto Parts Finder

Group

- Nanda Min-Fink
- Eva Pavlik
- Sabir Saklayen
- Rey Stone
- Sami-ul Ahmed

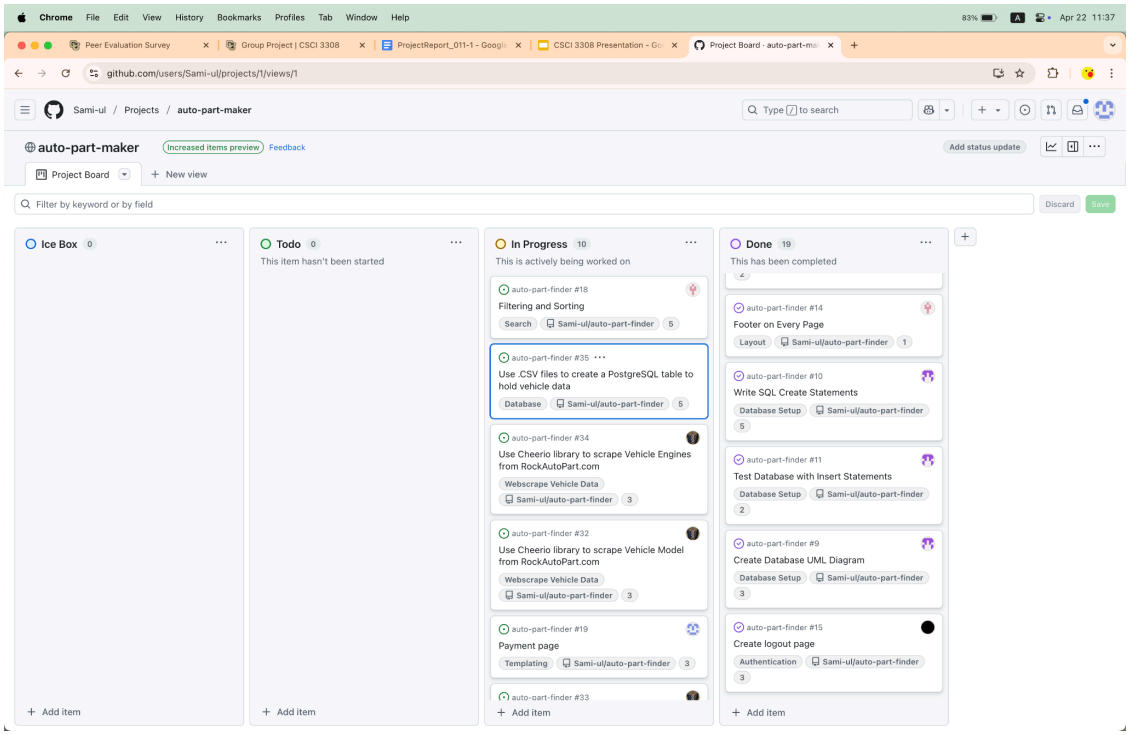
Description

Pocket Mechanics' Auto Part Finder is a feature rich, easy-to-use website for all your auto part needs. By creating an account on our website, you have the ability to save your vehicles in your garage and search for parts that are guaranteed to be compatible with your vehicle. Our website allows you to save addresses of local stores as well as your home address, making it seamless to get parts shipped and delivered to you.

This project was a conglomeration of multiple layers of software development coming together to create one working, cohesive application. It demonstrated the amount of skill and dedication needed to create a fully functioning website. The different components used in this website included a working user interface, a database to store user information as well as the auto parts that people intend to purchase, and a server to connect both together.

Our application connects the user to the server which displays data about auto parts available online. If the user makes an account, they are able to purchase parts, save vehicle information, list addresses of stores and their residence, as well as update any of these settings on a needs-basis. The UI makes this all available to the user, while the server helps connect the UI to the database which stores all of the necessary information.

Project Tracker



Video

This should work if you are signed into Microsoft Office.

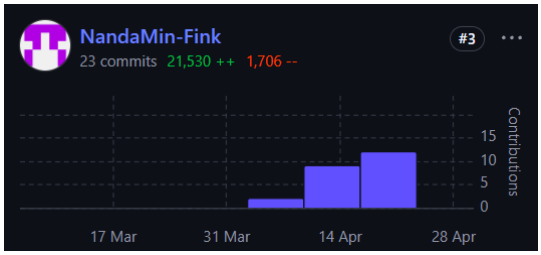
Git Repository

Contributions

Nanda

I worked primarily on backend and database development. I designed the database schema that connected users, cars, vehicles, parts, and orders. Once we obtained the rockauto scraping data. I created the system that inserted it properly into the database with all the necessary relationships. I developed the address functionality that leveraged apis for address autocomplete and nearby auto shops. I also created the checkout page and used stripe's api to create an auto validating payment method. I spent a lot of time working with forms on the front end and ensuring that the server communicated properly with these forms and enacted queries to the database to improve form

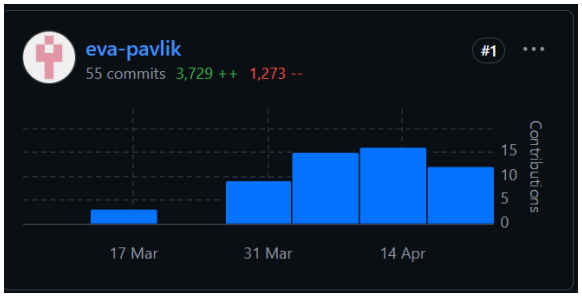
validation. I worked on the MyCars page to ensure it worked with our database schema properly and created the search by car dropdown on the discover page.



*Note: my account was not properly linked in github so some earlier commits are missing.

Eva

One of my contributions to this project was creating the drop down buttons for the discover page and my cars page. I had the drop down filter by previous make, model, etc. choices so you could only choose what was appropriate. I also implemented the “Search for Parts” button that autofilled the “Choose Vehicle” button on the discover page with the vehicle from the user’s garage. My main focus was the frontend development of our website as well as the overall consistency in styling throughout the website with CSS. When anyone would add in a new part, I would edit the styling to ensure overall appearance consistency.



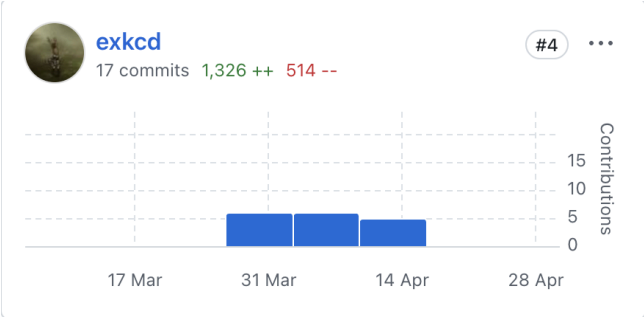
Sabir

My main focus for this project was data acquisition. Using Cheerio and Nodejs, I was successfully able to navigate the HTMl structure of www.rockauto.com to scrape the relevant necessary data. After scraping, this data was structured into a .CSV file from which another team member was able to take a sample set that fit the criteria of our storage limitations as well as the database schema that was developed for this project. Once this database was built, I aided in adding functionality to the Discover page. Specifically, adding onto the initial search route built by another team member, I implemented robustness into the search, added pricing filters, aided in the design of paginating the results from a query, implemented vehicle filtering via document.cookie if the end-user selected vehicle filtering options, developed a “Vehicle compatibility” filter for individual parts (this was different from the vehicle filtering) and added simulated markups to the pricing relational table to simulate other prices from vendors.

(Note: Cannot locate contribution graph, however my commit history should be well-logged.

Rey

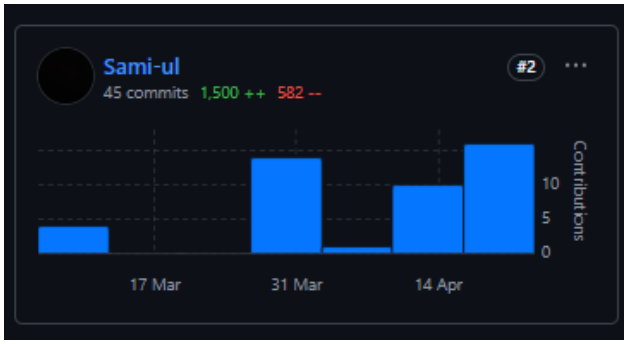
I worked primarily on the user interface of the application. I designed the wireframes in Figma which were used to model all the pages that are seen on our website. I then implemented all the designs into the templater that we used—Handlebars—to provide a consistent layout and look for the website. For example, I designed the navigation bar to be styled consistently throughout the pages you can navigate, as well as designed the accounts, discover, and login/register page to display relevant and correct information no matter who accesses the website. Implementing the UI early on in the project was integral for when the data was then scraped and populated into the database. This ensured that all relevant data and information was correctly displayed in the application.



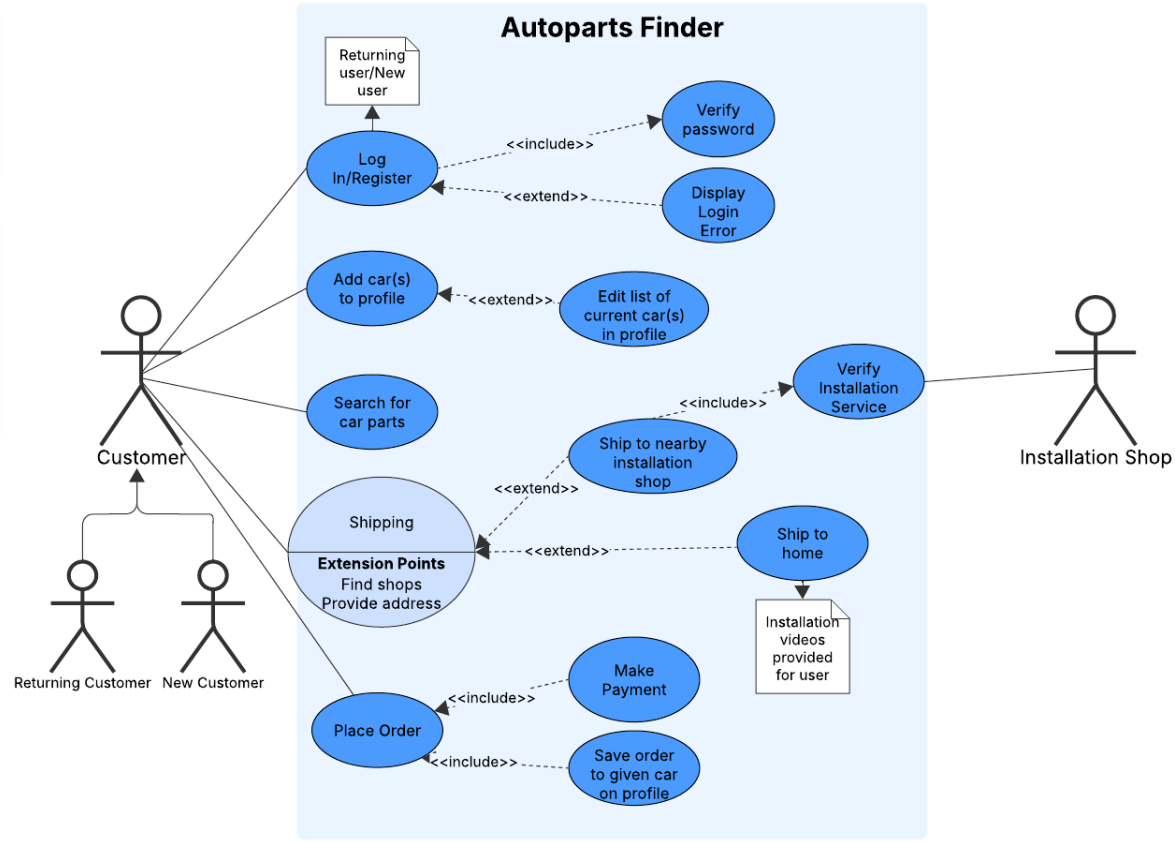
I had two GitHub accounts contributing to this project (rast4675, exkcd) because I couldn't figure out how to only use one.

Sami

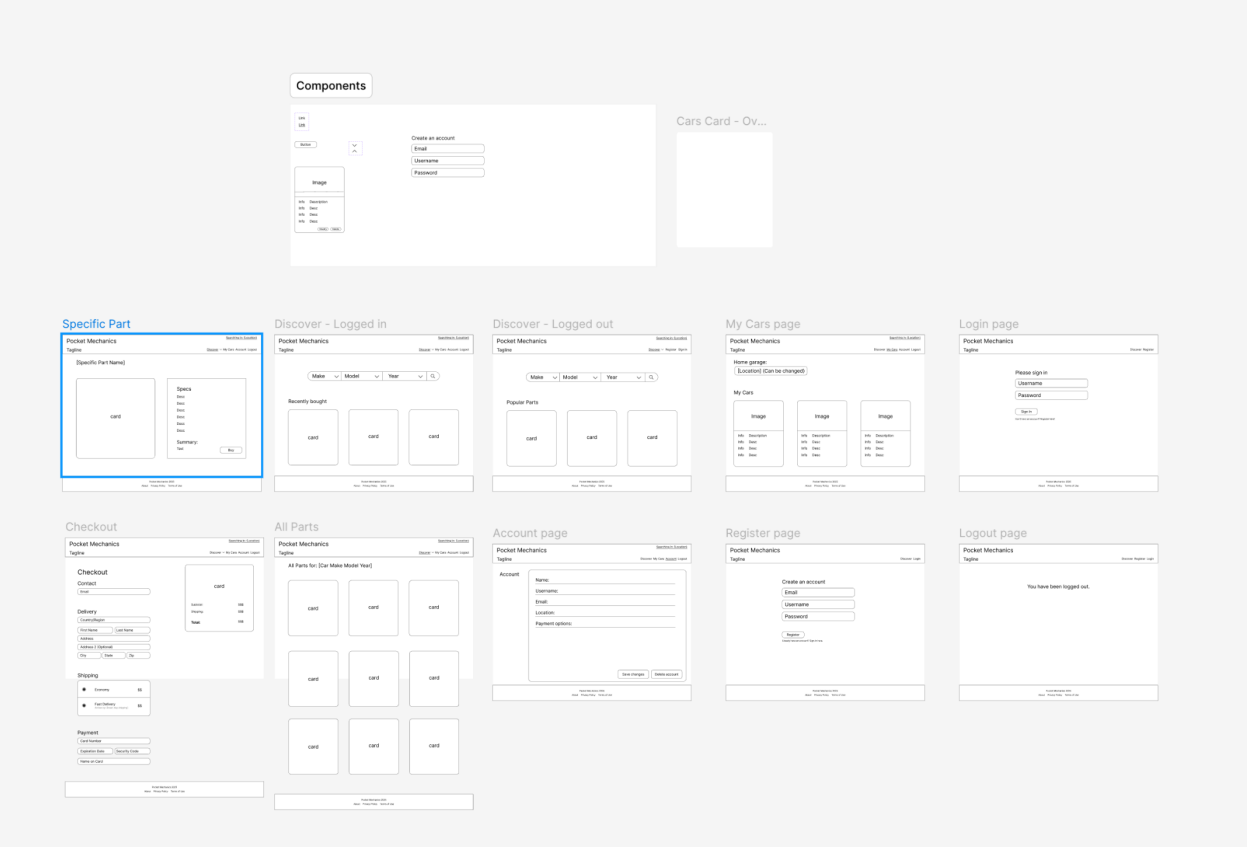
I worked on the DevOps and CI/CD side of the project. I wrote and maintained a suite of automated tests to guarantee reliable commits to GitHub, ensuring that changes in one module couldn’t inadvertently break another. I also had to continually update the tests to be up to date with the codebase to prevent false negative test failures. I set up Github Actions which automatically ran the tests I wrote whenever the remote repo had a change, catching regressions early. I was in charge of deploying our database and app on Render, securing sensitive data and ensuring seamless online functionality.



Use-Case Diagram



Wireframe



Test Results

Test subject who had not used the application previously was selected, minimizing bias and giving a real sense of what users would experience and how they would interact with the application.

Account Registration

- Test subject instructed to create an account. They started on the discover page and correctly navigated to the register page using the button in the navigation bar. They entered their email, username, and password, and then submitted the form. No issues, such as invalid emails, usernames, or passwords, were encountered. The user behaved as expected.
- Test subject instructed to log into their newly created account. They entered the username and password they selected and once they submitted it they were correctly redirected to the discover page.

Feature 1: Adding Cars To Profile

- Test subject instructed to add a car to their profile. They correctly found their way to the My Cars page from the navigation bar. They pressed the Add Vehicle button and filled out the dropdowns, with the intention of selecting a 2018 Toyota Camry. Once they filled out this popup form, their *My Garage* page was populated with the corresponding vehicle information. The user was not allowed to submit the form without filling out all the mandatory fields. The user was able to see their vehicle in the garage. The user interacted with the application as expected.
- Test subject was instructed to add a car without a make. They were unable to fill out any other information as the application requires a make before continuing to further drop downs. The application blocked form submission, behaving as expected.

Feature 2: Search for Car Parts

- The user was instructed to search for a wiper for the car they added to their garage. They correctly used the “Search for Parts” button in their garage to take them to a filtered search for their 2018 Toyota Camry. Once on the discover page, they searched for a wiper and were able to find some options. The user interacted with the application as expected.
- The user was asked to filter by wipers between 2 and 3 dollars. They correctly used the pricing filter to do so. The user interacted with the application as expected.
- The user was asked to add a part to their cart. They correctly used the add to cart button for a chosen wiper blade to do so. The application showed a confirmation popup and added the item to their cart. The user interacted with the application as expected.

Feature 3: Checkout Flow

- The user was instructed to purchase a part.
    - The user was instructed to add a second part to their cart as they did before. The user interacted with the application as expected.
    - With the newly added part, the user was asked to navigate to their cart. The user navigated to their cart through the popup following the *add to cart* button, which asks “Continue to cart?”--another path to reach the cart is through clicking *cart* in the navigation bar. The cart properly had both parts that the user had selected. The user interacts with the application as expected.
    - The user was prompted to remove a part from the cart. The user interacted with the application as expected, and the part was removed from the cart, updating instantly.
    - The user was instructed to check out their parts. They correctly navigated from the cart to the checkout page. They filled out the form for contact information as expected. They then filled out their address, which took them to the address page.
        - In the address page their address was autofilled. The user was instructed to select a nearby auto shop. The icons on the interactive map made this easy.
    - Once they had saved their address details, the user went back to the cart page, and then the checkout page, where they filled out the details and selected their saved address.

        - They then selected their shipping option, and then pressed *Continue to Payment*.
        - The user filled out the Stripe payment form with the default Stripe testing card. Once they hit pay, their cart was cleared and they were shown a *Success* screen.
    - The user was then asked to check out another part without selecting an address.
        - The address page did not allow them to enter an empty address; the checkout page did not allow them to select no address. The user interacted with the application as expected.

The User Acceptance Test was completed. The user commented that the user interface was intuitive. Akshay Patnaik was a great tester!

Deployment

- The deployment will take ~50 seconds to become online, due to Render's restrictions.