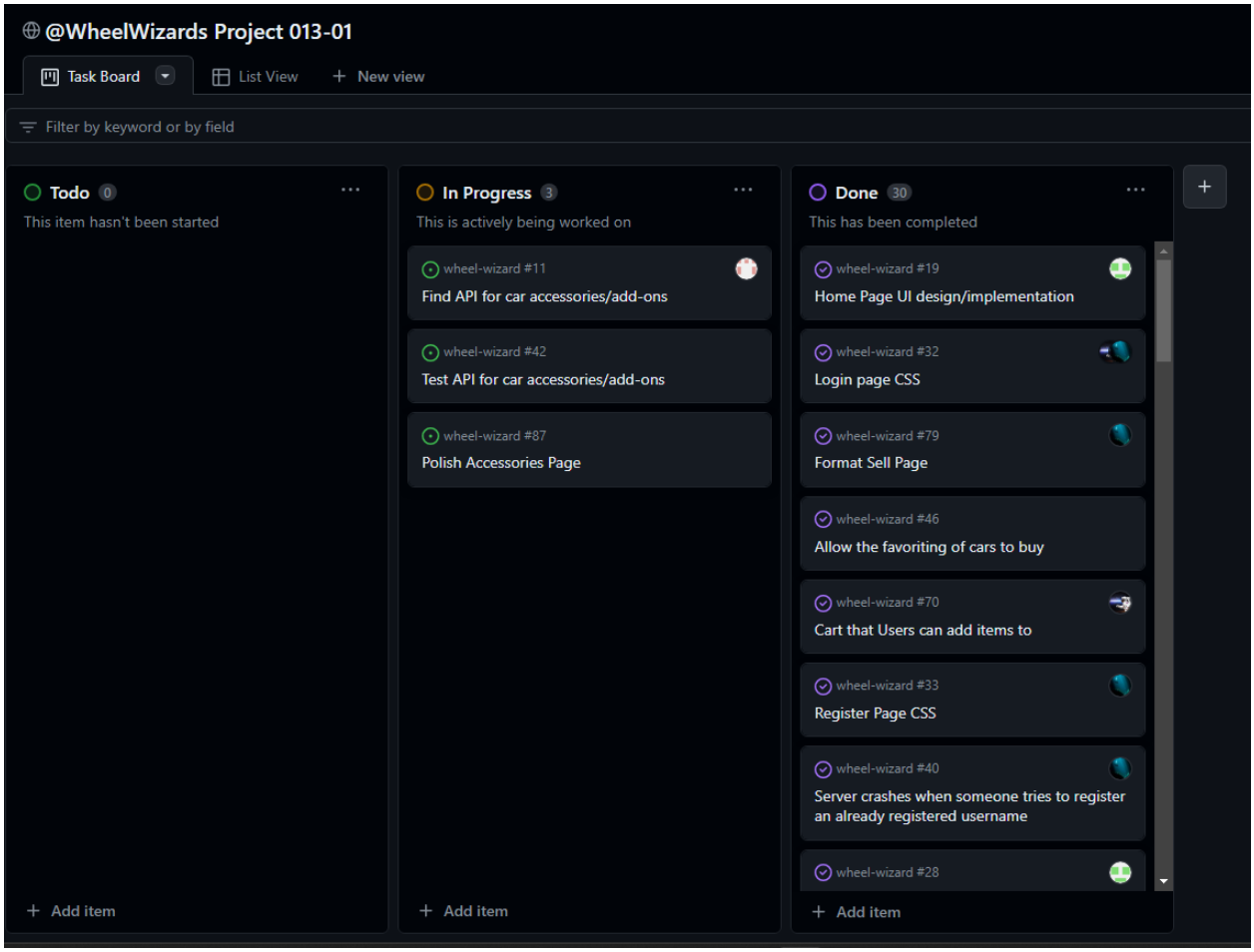
Wheel Wizard Project Report

Cobhan Kale, Brock Hoos, Ananya Tandel, Michael Giesecke, Michael Stich

Project Description:

Our website, Wheel Wizard, on the most basic level is a used car marketplace. Once a user is on our website they will not have access to any of the pages besides the homepage until they login or register. In the buy page they can search for any car from our external API based on make, model, color, price, mileage, or engine type. After finding a car they like they can then add the car to their 'cart', which serves as a favorites category that connects to the user's profile. Within the sell page a user can list a car they want to sell by providing the make, model, mileage, price, and a quick description. After submitting their car it gets dynamically added to the buy page where other users can see it and add it to their cart in real time. The user that posted it can see it under their listings on the sell page where they can delete it at any time. The accessories page serves the purpose of allowing users to see possible accessories they can add to their future car. On the profile page, the user can see their username and has the option to change their password along with deleting their profile altogether. In the cart page, the user can see all the cars they have added and search through all the cars and remove them from their cart.

Project Tracker: https://github.com/users/stichmc/projects/4



DEMO VIDEO LINK:

https://drive.google.com/file/d/1npyPooVjETu_QWTy4paswaLGNnYs8ACg/view?usp=sharing

VCS: https://github.com/stichmc/wheel-wizard

Final Report:

https://github.com/stichmc/wheel-wizard/tree/main/milestone_submissions/Final_Report

Contributions:

Michael Stich:

I created the GitHub repo, and protocols to push, pull, and merge which led to minimal merge conflicts and rebasing. I set up the root folder structure. I wrote all the chai/mocha tests in the server.spec.js file. I created the docker-compose file and got all dependencies working and the environment running for the website. I co-created the database schema, conceptually and code-wise. I assisted in login/register APIs and created the login/register EJS pages, created POST profile delete API, created GET buy API, created POST add to cart API, created GET cart API, created POST remove from cart API, created GET accessory API, created GET welcome API, created POST delete profile test API. I created an Axios API call to sort through external car website API data and insert that data into the database. I also performed various bug fixes.

Cobhan Kale:

I co-developed the database schema, made the login and register GET and POST APIs, the logout page and related APIs, sell EJS pages and sell_new page, all the sell page and the related APIs (GET /sell, GET/sell/new, POST /sell/new, GET /sell/remove-listing, POST /sell/remove-listing). I made all the dynamic EJS messages on the login/register/logout page, the home page, and the sell new car pages. I configured the Azure VM and hosted the live website. Also I did much of the organization / documentation in index.js and in the repo.

Brock Hoos:

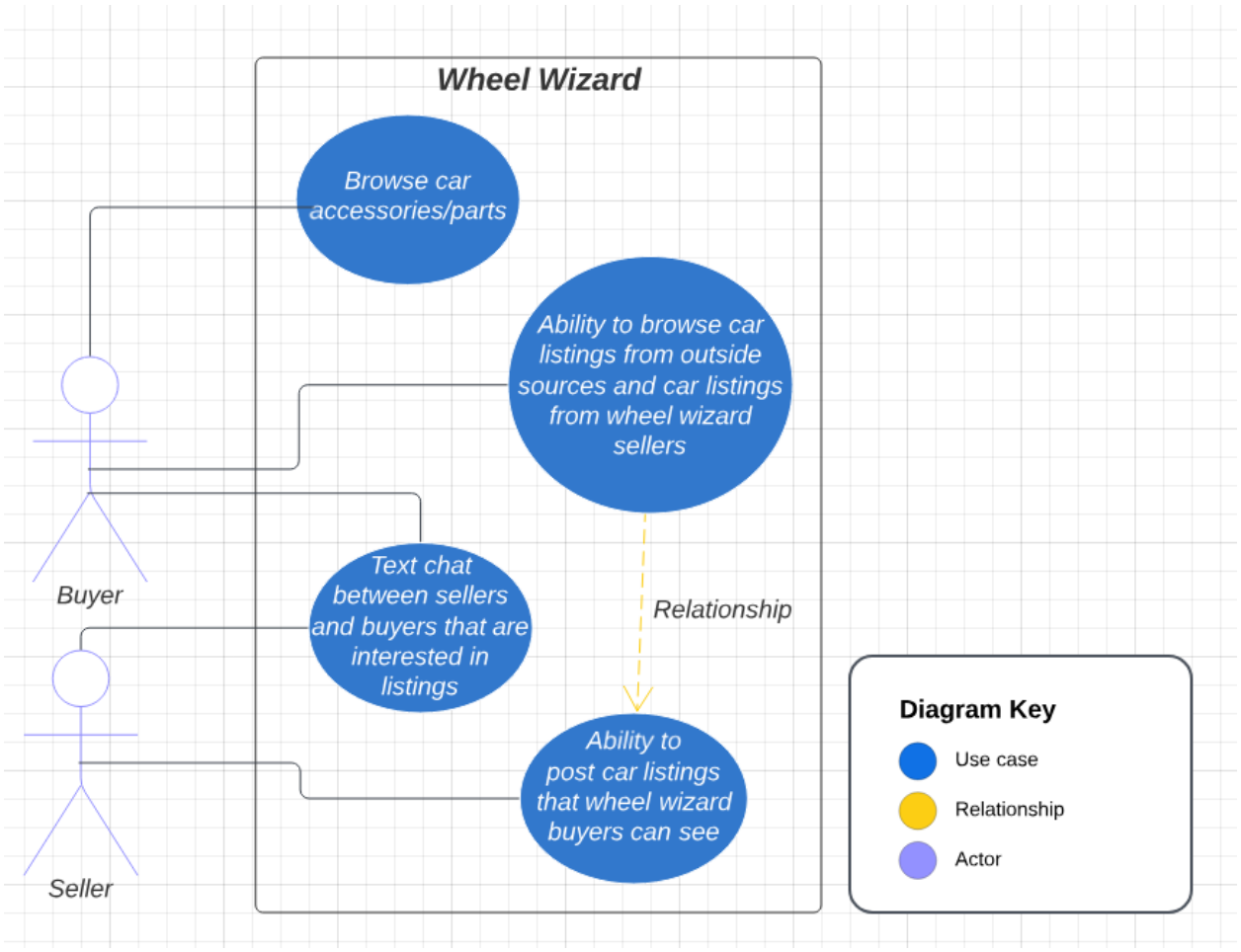
I designed and created the UI interfaces through wireframes. Along with this completed the partial pages which involved the NavBar and Footer along with the Wheel Wizard logo in the Navbar. I completed the .ejs files for the Home page and Profile page and got the functionality of all the redirects working to the desired pages. Along with Michael S. we got the ‘Delete Profile’ and ‘Change Password’ API calls to work and be fully

functional within the Profile page. Also, I was able to formulate a plan to be able to implement images into our website through raw github photos rather than using Cloudinary.

Ananya Tandel:
For my part, I was in charge of the accessories page with the grid styling and adding the title, description, and price. I used Bootstrap and EJS to complete the accessories page. Instead of pulling from the Cloudinary API for images, I used raw github photo links with the help of Brock. I also had made an accessories array/database so that I could get the route for adding or removing accessories to work, but I was unfortunately unable to make it functional. I also made the entire slide deck for our project.

Michael Giesecke:
I found the API that we were able to use for our website and went through the documentation to help Cobhan and Michael S. understand how to implement it. I also added an external IT/Help Chatbot where users can submit questions and someone is able to respond to that question. Once Michael S. got the backend database working for displaying car listings, I worked on the UI side of the page by adding a search/filter script written by myself to filter cars for any text inputted and created a bootstrap table to improve aesthetics. This code Cobhan used to do the UI for the Sell page to make it more functional and keep with the theme. I then created the custom colors and stylings for others to use. I also worked on some of the ejs. pages including adjusting the footer and header.

Use Case Diagram:



Test Results:

Feature 1: User Authentication

Test Case 1: User should be able to log in with correct credentials.
Test Case 2: User authentication fails when the user provides invalid credentials.
Test Case 3: The form provides the user with specific feedback about the error.

Test Data:
* Valid credentials for successful login.
* Invalid credentials for unsuccessful login.

Test Environment:
* Docker
* Visual Studio Code
* HTML
* Bootstrap
* Chrome browser

Test Results:
* Record success/failure of login attempts
* Display error message to users with invalid credentials.

User Acceptance Testers:
* Buyers and sellers using our website.
* Classmates / Peers of team members when gathering feedback.
* Team members for initial testing and verification.
* A user cannot login successfully until they fill out the mandatory fields: username and password.

Acceptance Criteria:
* The login page includes mandatory input fields for username and password.

* Users can successfully log in with correct credentials.
* Users are directed to the homepage after logging in successfully.
* User authentication fails if invalid credentials are entered.
* Users receive error messages for invalid login credentials.

Feature 2: Accessories Page Functionality

Test Case 1: Users should be able to view a list of accessories on the Accessories page.
Test Case 2: Clicking on an accessory should display detailed information about that accessory.
Test Case 3: Users should be able to add accessories to their cart.

Test Data:
* List of accessories for testing the display.
* Individual accessory details for testing viewing detailed information.
* Test user's cart with added accessories.

Test Environment:
* Docker
* Visual Studio Code
* HTML
* Bootstrap
* Chrome browser

Test Results:
* Verify that the accessories are displayed correctly.
* Verify that clicking on an accessory shows accurate details.
* Verify that adding accessories to the cart updates the cart correctly.

User Acceptance Testers:
* Buyers and sellers using our website.
* Team members for initial testing and verification.
* Classmates / Peers of team members when gathering feedback.
Acceptance Criteria:
* The Accessories page displays a list of accessories.
* Clicking on an accessory shows additional information, like name, description, and price.
* "Add to Cart" button is shown for each accessory.
* Clicking "Add to Cart" updates the user's cart with the selected accessory.
* The cart reflects the correct number and details of added accessories.

Feature 3: Profile Page Functionality

Test Case 1: Users should be able to view and edit their profile information.
Test Case 2: Changes made to the profile information should be saved and reflected in the database.

Test Data:
* Test user's existing profile information.
* Modified profile information for testing edits.

Test Environment:
* Docker
* Visual Studio Code
* HTML
* Bootstrap
* Chrome browser

Test Results:

* Verify that the "Profile" page is accessible.
* Verify that the current profile information is displayed accurately.
* Verify that users can edit and save changes to their profile information.
* Verify that changes made are correctly stored in the database.

User Acceptance Testers:
* Buyers and sellers using our website.
* Team members for initial testing and verification.
* Classmates / Peers of team members when gathering feedback.

Acceptance Criteria:
* The "Profile" page is accessible from the homepage AND navbar.
* Users can view their current profile information.
* Users can edit and save changes to their profile information.
* Changes made to the profile information are correctly stored in the database.

Deployment: http://wheelwizards.eastus.cloudapp.azure.com:3000/home

In this project, we successfully deployed the website using a combination of Azure and Docker technologies. Leveraging Docker for containerization allowed for seamless and consistent deployment across various stages. Azure was chosen as the cloud platform for its robust hosting capabilities and free usage for CU students, and we utilized Azure Container Instances for efficient container deployment.