Karv

Parker Allen, Xan Gardner, Jack Savio, Conrad Barron, Luke Gosnell, Andrew Lan

Project Description:

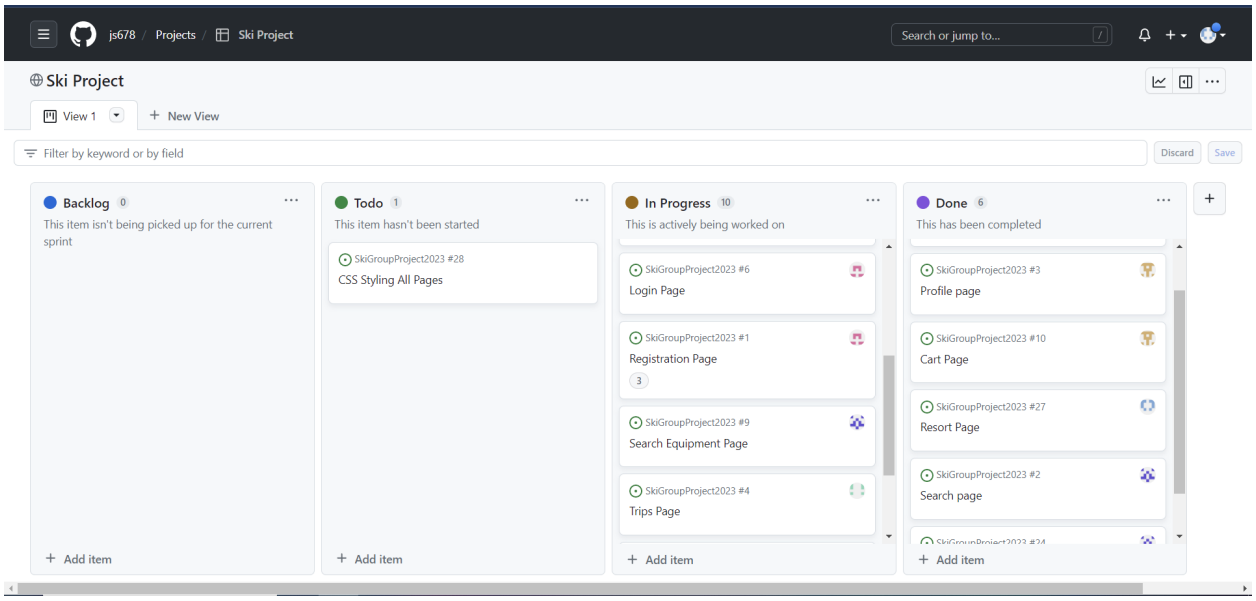
Karv is an all-in-one ski marketplace software that guides users to discover their new life within skiing, with both shopping and planning functionalities. The home page is the hub of our website. There, users can navigate to all the different pages, view our mission statement, and watch a skiing video. To access the other website features, however, users must create an account so that their information, such as past purchases, can be safely stored and looked at later. Users can do this at our user-friendly login and registration pages. Once logged in, users can look at various ski equipment within our software, from skis to gloves to boots. Many different brands and prices are available, giving the user many options for their ideal ski trip. Users can add items they are interested in into their cart, where they can be removed or purchased later. We also offer information on trips to over 100 different ski resorts and locations. The user can even log which ski resorts they went on a trip to and for how long. Both the past trips and the past purchase information is stored on the profile page, where users can view and change their username and password as well.

Project Tracker:

We used github for our project tracker

Link: https://github.com/users/js678/projects/1/views/1

Screenshot:



Video:
https://drive.google.com/file/d/1ml_IISrbuwuqka9zWZtIcZ9SZLMPs0pq/view?usp=sharing
VCS:
Github Link: https://github.com/js678/SkiGroupProject2023

Contributions:
Parker Allen: I helped with the presentation and report. I implemented the individual trip page that displays a table of all the trip information, including the website and the resort's picture. I also implemented the trips database with all the data and inserted all the data into the table to be used in the trips page as long as other pages. I used GitHub to upload code and keep track of the project's progress, VSCode for writing the code, EJS to display the trip page, and NodeJS to get the behavior of the website.

Xan Gardner: I implemented the individual resort page which displays the resort information, has a Google Maps API, and has the add trips feature. I helped where needed, including searching for and fixing bugs in the code. I helped contribute to the project report and presentation. I used GitHub for uploading code and keeping track of the project, VSCode for writing code, the Google Maps API, EJS for UI, and NodeJS for the server side.

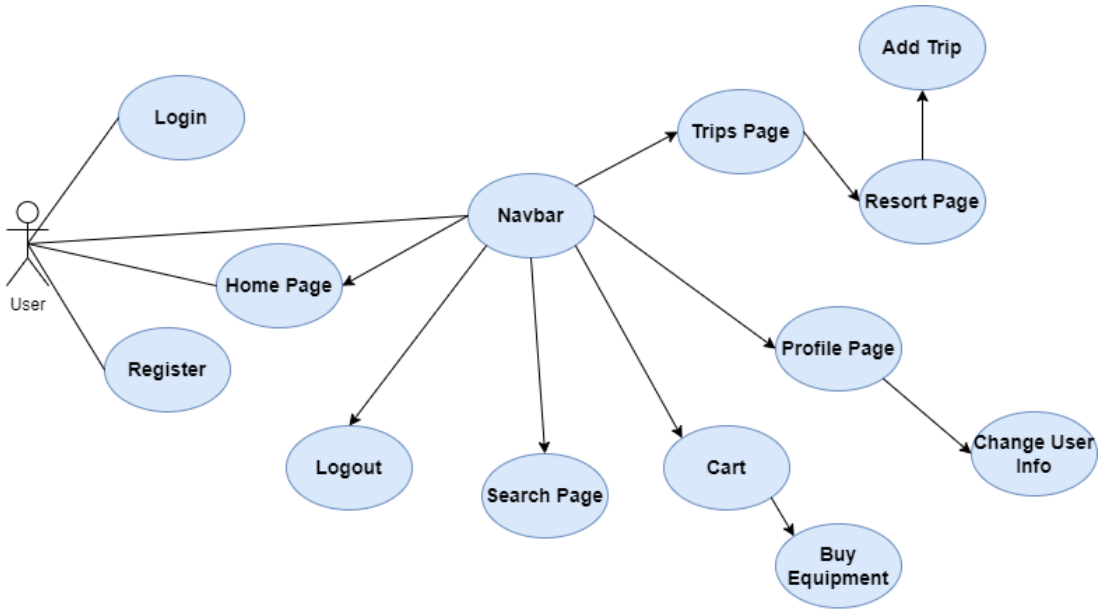
Jack Savio: I fully implemented the navbar, footer, home page and database for the shopping marketplace. This included all the partials for ejs, as well as the main page for the website with links related to each web page. I also assisted wherever needed, as well as contributing to parts of the project report and presentation. I used the project board on GitHub, coded on VSCode, implemented parts of nodejs, as well as used EJS to implement UI tools. Tested vigorously, trying to find errors within the site.

Conrad Barron: I fully implemented the profile page and the cart page as well as doing the test cases. For the profile page I read from the database and added some javascript to help update the user information. For the cart page I read from the database and added some javascript to help add the item to the past items table or remove it from the cart items table. Other than this I helped on some other little things with other people.

Luke Gosnell: I implemented the login and registration pages This included writing the basic structure of the pages in EJS, the styling for the pages in CSS, and the backend APIs for the pages in NodeJS. I also helped populate the database for skiing products.

Andrew Lan: I implemented the search and product page that shows all the accessories for skiing and snowboarding, and the functionality to add item to the user's cart. For the product page, we made our own database and read the information from the database, I was trying to help with the page style and other stuff. I used Github to keep track of the code and changes, VSCode for modify local changes, EJS for pages, and NodeJS for api.

Use Case Diagram:



Test Results:

When testing the registration page, the user first tried a normal input for the required fields but then tried to use other strange inputs to see if they could break the page. They put in an emoji for their username and the application allowed this input. The application didn't break and the user was able to enter the same information on the login page to login. This behavior deviates from the expected behavior and interactions expected because we did not account for users trying to input emojis and other strange characters into the registration form. If given more time, it would be good to sanitize the inputs for the login and registration forms so that users cannot input characters other than regular letters, numbers, and a few allowed characters.

When testing the login page, the user tried to login with an account that didn't exist and was redirected to the registration page. There they made a new account and logged in with the new account information. The user's behavior is consistent with the use case and they didn't do anything extreme besides what they did in the registration page, where they used an emoji as a part of their username. Looking at the registration test, it would be good to sanitize the input for this form so that only allowed characters can be put into the fields.

When testing the profile page, the user tried to change their username to something extremely long. They wanted to see if they could break the page. The UI allowed the user to change their username to something absurdly long but the server

threw an error because the string was too long. If given more time, it would be a good idea to sanitize the inputs for those forms and to put in a character limit.

When testing the search page, the user tried inputting characters such as emojis and numbers to see if the page would break. The page didn't display any ski gear and had no strange outputs. The user then tried to add over 100 of the same item to their cart to see if there was a limit to what they could buy and if the cart would break. The cart page was fine and did not break, displaying all of the items normally. The user's behavior was not consistent with the use case because we didn't expect user's to try and put in emojis for the search input or to try and add over 100 of the same item to their cart. If given more time, it would be a good idea to sanitize the input for the search bar.

Deployments:

For the most part, the lab was deployed locally via VSCode, Docker, and a web browser. To access/run the app locally, you would need to clone the project to your local machine, navigate to the repository in VSCode, start up docker, and go to http://localhost:3000/ to access the application.

Following lab 13, we also deployed the lab with Microsoft Azure at the following link: http://recitation-012-team-1.eastus.cloudapp.azure.com:3000/home. However, since we are only allowed a certain number of free credits, we had the application running there for only a short period of time before we shut it down.