Title

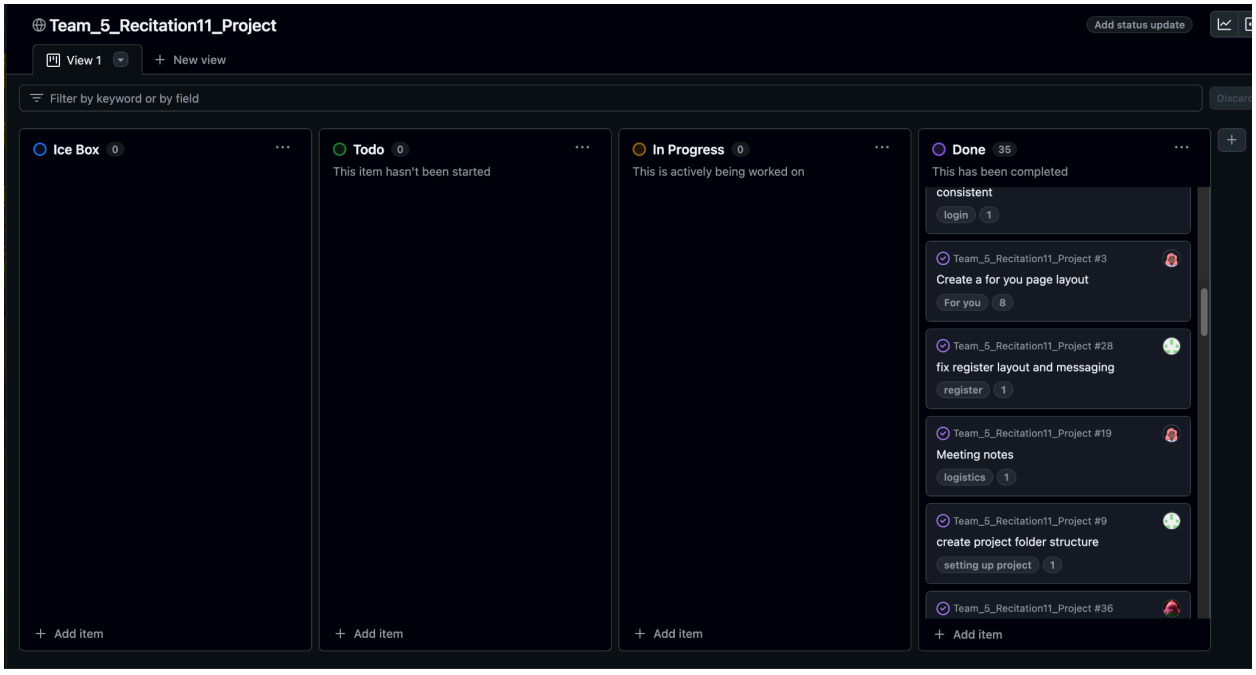
Style Swipe - Mary Kodenkandath, Matis Uhl de Morais, Nandini Nema, Soumya Devallapuli

Project Description

Style Swipe is a Tinder-style web application designed for fashion enthusiasts to discover, organize, and draw inspiration for their outfits. The platform features a simple interface with images of clothes, and an engaging experience where users can swipe left (❌) or right (❤️) on pictures of clothing to express their preferences. To use the website you are required to register an account and/or login. Once logged in, you are redirected to the profile page where you have 2 options: go to your closet or discover new items. The discover page is the main feature of our application, showcasing a collection of outfits where, again, users can choose if they would like to wear or not. The my closet page saves all of the outfits which you swiped right on, for future inspiration. Additionally, the my closet page also highlights the upload images feature, where users can upload their past outfits for even more inspiration! Whether you're looking to try out new trends or compile your favorite looks, Style Swipe makes fashion exploration effortless and fun.

Project Tracker- Github Project Board

Link: https://github.com/users/nnema05/projects/1/views/1



Link to Github repository

https://github.com/nnema05/Team_5_Recitation11_Project

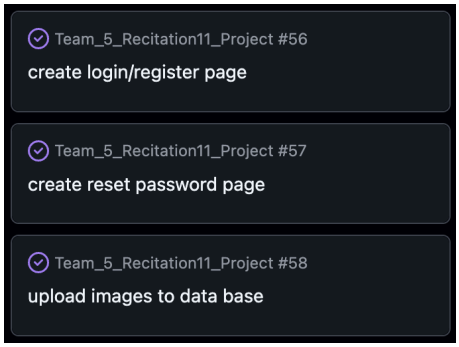
Video

StyleSwipe.mp4

Contributions

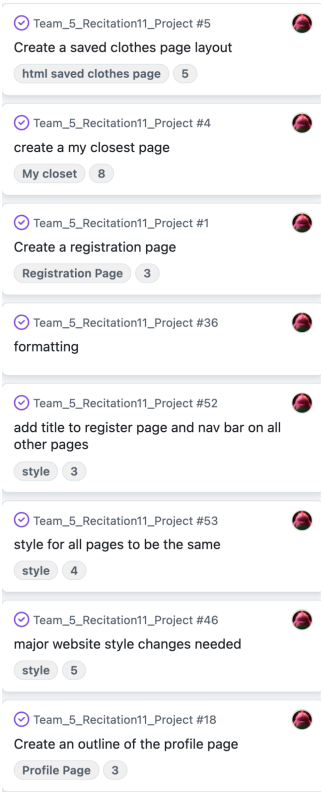
Mary Kodenkandath

Mary's contributions included the setup of the login, registration, reset password page, the uploading of images for the feed, and styling changes. Setting up the login and registration page was similar to those in other labs so she used those for reference and edited them to meet the requirements for this platform. The reset password had additional logic, including confirming the new password, and resetting it for the user's account. For the feed, she uploaded images from Pinterest, and followed a series of steps to convert and integrate them into the database. Her styling changes ensured that all the pages she managed followed the platform's overall theme.



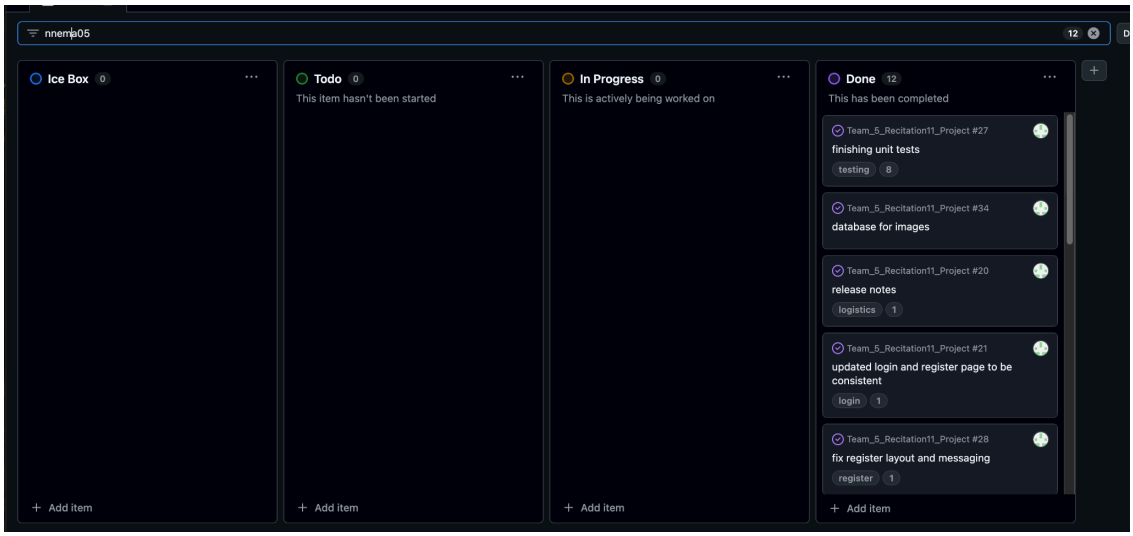
Matis

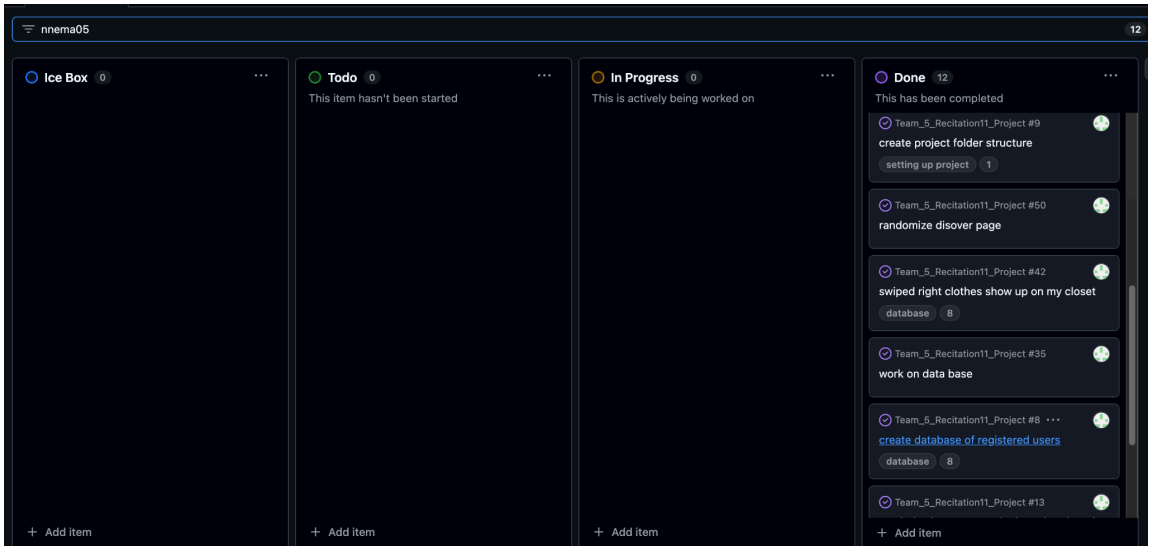
Matis' largest contributions were the style changes including a dynamic animated background as well as aesthetic gradient buttons and other layout choices. Additionally, Matis worked on the Pinterest API early on which ended up being changed later on, started the upload image feature (finished by Soumya) and fixed additional functionalities for that feature, and created the profile page personalized to each user. Created the first version of the discover page implementation for choosing between closed but was eventually modified for efficient use with the database. Created consistent logical formatting for the navigation bar on all pages.

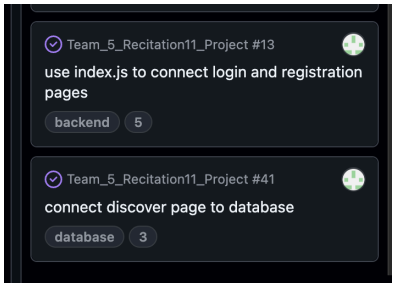


Nandini Nema:

Nandini worked mostly on the backend of the application. She created a database, the users and outfits table and wrote scripts to populate them. Then she wrote the route to connect the discover page to the outfits table and saved what image the user was viewing at the moment. She also worked on the backend of swiping right features by creating a saved clothes database, writing a route and making sure they appeared on the user's My Closet page. Finally, Nandini contributed to the testing lab and to the final deployment of our website on Render.

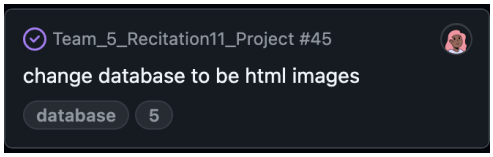


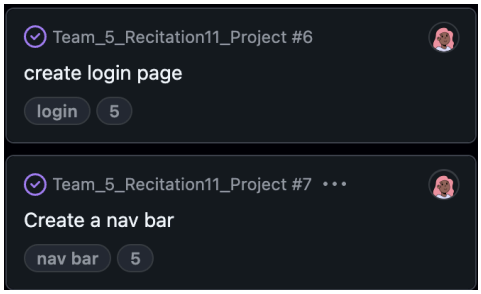


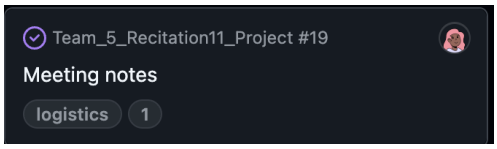


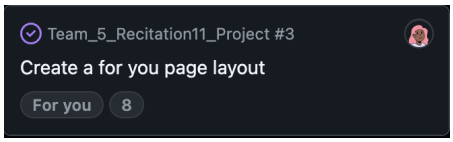
Soumya Devallapuli

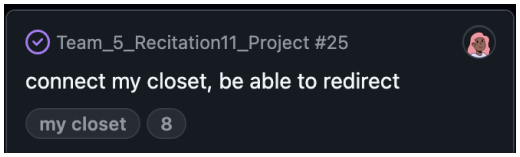
Soumya worked mostly on the saved clothes and discover pages. She worked on both the front end and the back end of both of these pages. She worked on setting up the cards and the base blueprint of the discover page. This involved using HTML, CSS and JavaScript. Soumya also worked on the upload clothes feature, which was first started by Matis. She worked on making sure that the upload button would properly upload the clothes, save them to the my clothes database and finally made sure that they printed out onto the my clothes page. She finally also started the implementation of the login page and the nav bar.

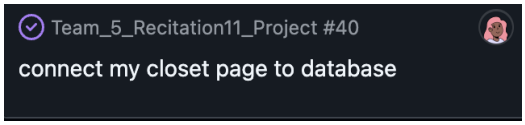






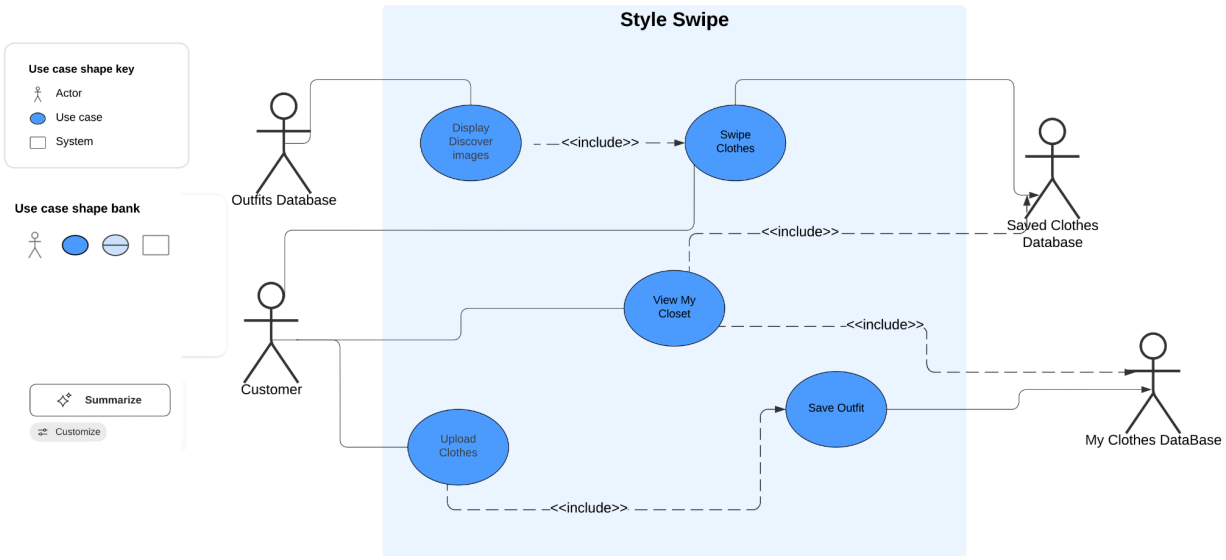




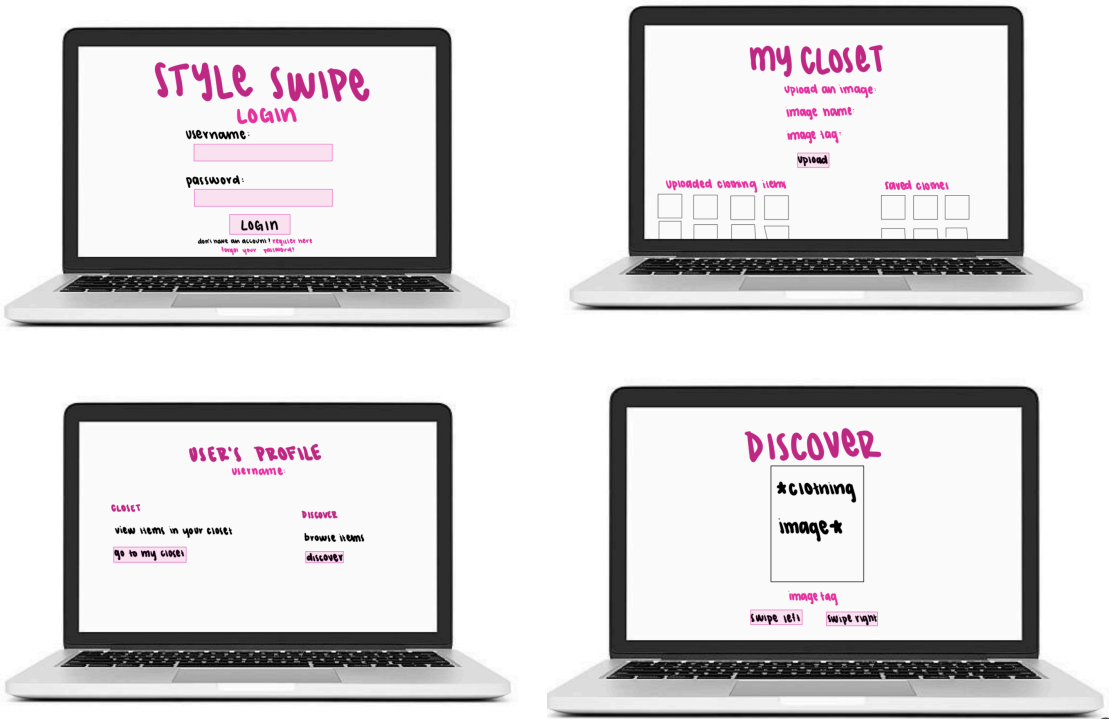


Use Case Diagram

A use case diagram is a visual representation of the interactions between the system and the user. The primary actor is the customer here, and they are interacting with the use cases of swipe clothes, view my closet and upload clothes. There are secondary actors which are the outfit database, the saved clothes database, and the my clothes database. The outfits database connects to the display discover images use case, as this is where all the outfits that are displayed on the discover page are stored. This is then included in the swipe clothes, as the images that are swiped on are directly from the outfits database. The clothes that are swiped right on are included in the saved clothes database, and the clothes that a customer uploads are saved into the my clothes database. The clothes from these two databases are included in the view of my closet use case. The content from these two databases are displayed there.



Wireframes



Testing Report

As part of Lab 11, we developed and completed a test plan for observing user interactions with our application. This report describes our test cases. In addition, we ran our test plan with

peers outside of our class and documented their user interactions and the changes made based on the findings.

Test Cases

We tested four use cases for users with positive and negative test cases:

1. **Registering a New User**

   Positive case: A valid username and password creates a new user account successfully.

   Negative case: Some sort of invalid input into the register pages results in an error. For example, if the user tries to register an empty username or password name, the invalid input message is sent.

2. **Logging In**

   Positive case: A user that is already successfully registered is able to log in

   Negative cases: If the user puts in an incorrect password or tries to log in with a user that has not been registered and does not exist in our database then the corresponding error message is displayed.

3. **Redirect Behavior**

   We made a sample /test endpoint to test if our application is able to redirect a user that is not registered or authenticated to the login page as designed.

4. **Rendering Pages**

   We tested that specific routes and endpoints could render an HTML page correctly and meet our user interface requirements!

   We also developed tests that weren't tested by Mocha and Chai in our test plan and just tested by our user. These included tests like:

5. **Profile Page**

   Our criteria for this test is to make sure every user is asked for a name, username, email, phone number, password, profile picture, and they are all displayed on the profile page.

6. **For You/Discover Page**

   This test checks that the For You Page/Discover Feed keeps going until all the data from the API key has all been displayed. This test then displays a message that "You are all caught up" when there is no data left to display.

7. **My Closet Page**

   The test checks if the user uploads a picture of their clothing, make sure they have info about the clothing piece, a picture, date, where it is from, and how much it was.

Findings

We ran these tests in two ways. We used Mocha and Chai technologies where Mocha was used to actually run the runner and Chai was used as the assertion library to check for expected outcomes. We also ran this applications with peers outside of this class to run our 4 test cases:

1. Registering a New User

Observations: Our test cases run successfully as our positive test case returns 200 and a success message and negative test case returns 400 and an invalid input message. All of our peer users, when they first used the website, had to register to get access to the details of our application. They were able to register successfully when they entered a valid username and password. If they entered an empty username and/or password, they got a notification to fill out the empty field. If they entered a username that was already registered and in our database, our page redirects them to re-register.

Changes: As we tested with different types of users, we realized that when users tried to register with an already registered user, it was not exactly clear why the page was redirecting them to re-register, especially if they knew nothing about the application. As a result, we added an error message to display on the page that told the user that this "Username already exists. Please choose a new one."

2. Logging in

Observations: Our test cases run successfully as our positive test case returns 200 and a success message and negative test case returns 401 and an either incorrect password or username not found message. All of our peer users, after they registered, went to login. Most of them were able to login successfully but one of them mistyped their password and were redirected to the login page to login again. One of them tried to login with a mistyped username that was not registered and another with a mistyped password. Both of these redirected the page to re-login.

Changes: When the cases of the mistyped username and mistyped password occurred, the redirect to login was not entirely clear what the user did wrong. As a result, on the login page we added error messages too. If the user mistyped their username and tried to login with an unregistered username, they got the error that "Username not found. Please register." If they mistyped their password, they got an error that said "Incorrect username or password." These errors gave our users more actionable feedback.

3. Redirect Behavior

Observations: Our test cases run successfully as our test case returns a success status code when our user is able to successfully redirect to the login page. This worked also for all users. Every time they visited the website, the user was automatically redirected to the login page which showed our redirect was working. Because this worked the way we wanted to the first time, there were 0 changes needed.

4. Rendering Pages

Observations: Our test cases run successfully and return a 200 success status when it successfully renders an HTML content page. This worked also for all test users. Every time they open the website, it successfully renders and shows the HTML of the login page. Because this worked the way we wanted to the first time, there were 0 changes needed.

Overall, the testing phase highlighted both our strengths and the areas that needed to improve and change. This made sure our application functions well and is user-friendly.

5. Profile Page

Observations: These tests were not run using Mocha and Chai and just tested with our user. This worked because everytime we rendered the profile page, the user was able to see their name and information about the user.

Changes: We changed what is displayed in the profile page because we wanted to start with the minimum viable project which was showing the username first. Then we also made the profile page mostly show links to other places in the website so that they could visit their closet and discover because we found that more important to user functionality. For future scope, we would have taken in and implemented their email, phone number and password. However, after we made this change, the test ran successfully!

6. For You/Discover Page

Observations: These tests were not run using Mocha and Chai and just tested with our user. This worked because everytime the Discover page continuously shows clothes until all the images are shown. After the user is done swiping through the images, they get a message that all the outfits to them are shown.

Changes: We changed where the images shown on the discover page are coming from. Although we originally tried to do an external API key to Pinterest to get the images we would show, this proved to be difficult because Pinterest wanted a deployed application (which we didn't have at the beginning of the project) and would take weeks to approve us. We were on a time limit so instead we created a database of images of outfits and displayed those. However, after we made this change, the test ran successfully.

7. My Closet Page

Observations: These tests were not run using Mocha and Chai and just tested with our user. This worked because everytime the user uploaded a picture of their own clothing, they were required to upload both the image and information about the image.

Changes: We changed what information was displayed for the image for our minimum viable project such as the image name and tags about the image. In the future, we could implement more information about the image, like the link to where the image is so another person could purchase it. However, after we made this change, the test ran successfully.

Deployment

Link to Deployment: https://team-5-recitation11-project.onrender.com

Our app was deployed using Render. Render is used here to deploy a web application that interacts with a PostgreSQL database. This cloud deployment allows for another platform to initialize and maintain our database and our underlying infrastructure and has features like automatic deployments with our git pushes, managed databases, securely managing environment variables, and scalability if needed. To access and run our app, all someone has to do is click the link above.