NXTFolio

Final Report

Team Members

- Balmaseda del Campo, Vicente – Scrum Master
- Fletcher, Lance
- Jain, Ayushri – Product Owner
- Li, Baichuan
- Li, Wenyuan
- Price, Niah

Links

GitHub: https://github.com/vibalcam/match-my-fashion-public-CodeCreators

Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2630237

New Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2635646

Heroku Deployment: https://nxtfolio.herokuapp.com/

Presentation Video: https://youtu.be/4Ix2khDeu_o

Demo Video: https://youtu.be/4Ix2khDeu_o

Summary

NXTFolio is a web application aimed to become a networking platform for the creative industry and help professionals to find other professionals based on their requirements, saving them from paying talent agencies. It aims to help someone find professionals based on custom requirements like 'looking for graphic designer for pottery' or 'who is good at Runway photography in Dallas' along with establishing their own portfolio. The web application is expected to be used by three main categories of professionals - Creators, Services, and Makers. The stakeholder for this project is Prasenjit Tito Chowdhury, who is the Chief Executive Officer and Executive Producer of FashioNXT, a marketing agency in Portland, OR. The primary customer need was to improve the user interface and enhance the search engine, but new features were identified on the go and added to the project along with fixing some existing issues.

The first step that we took was to upgrade the rails and ruby version without which Heroku deployment was not possible. Along with that, we created Docker file for the project so that it can be deployed within minutes from scratch. As we familiarized ourselves with the legacy code, we identified broken features like sign up, adding and removing images from gallery, non-functional buttons on homepage and deletion of images in S3 bucket. Along with fixing these issues, we added new features to the application - following other users, adding other users as collaborators, adding travel details, following other users, and inviting other users to application, posting jobs, searching jobs and API for customer relationship management. We significantly improved the search algorithm by incorporating word matching, showing similar results in case of no matching results for a search, using travel information in search results and recommending projects on homepage. Another significant step was to transfer the ownership to client's Heroku and GitHub account so that the next teams can pick up the project from where we left easily. Finally, we also added automated GitHub actions to run cucumber and RSpec tests while creating a pull request. Other details like user stories, tests, setting up the project, issues and future tasks are described in the rest of the document.

User Story Descriptions

1. Project Ruby & Dependency Update

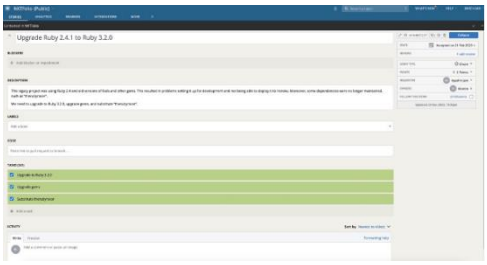
Points: 2

Implementation status: Accepted

Changes:

Upgrading the versions of Ruby and other gems, despite breaking some functionality, was deemed necessary for easier deployment and to create new features, and allowed for a more gradual upgrade of dependencies, while also fixing non-functioning features and tests, and substituting "therubyracer" for "nodejs".

Storyboards/Screenshot:



2. Migration and Installation Fixes

Points: N/A

Implementation status: Accepted

Changes:

Migration issues were resolved by isolating the migration that created duplicate columns and removing it, while database issues related to Postgres and unclear installation instructions were addressed by creating a Dockerfile and a script to automate the installation process, simplifying the setup for future groups and allowing for faster deployment of the app.

Storyboards/Screenshot: N/A

3. Gallery Fix(1)

Points: 1

Implementation status: Accepted

Changes:

Users can add their own reviews and multiple images as a gallery on both the profile and gallery pages.

Storyboards/Screenshot:

4. Database Seeding

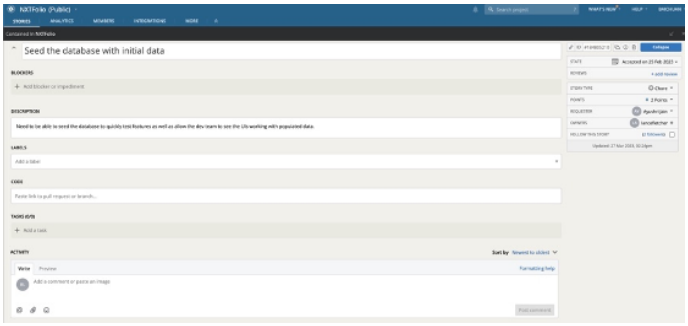
Points: 2

Implementation status: Accepted

Changes:

Populating the database with seeded data has been done to facilitate application development and acceptance testing, currently containing data for LoginInfo, GeneralInfo, and Gallery tables, with plans to include data for ratings, comments, and other tables in the future,

Storyboards/Screenshot:



5. Heroku Deployment

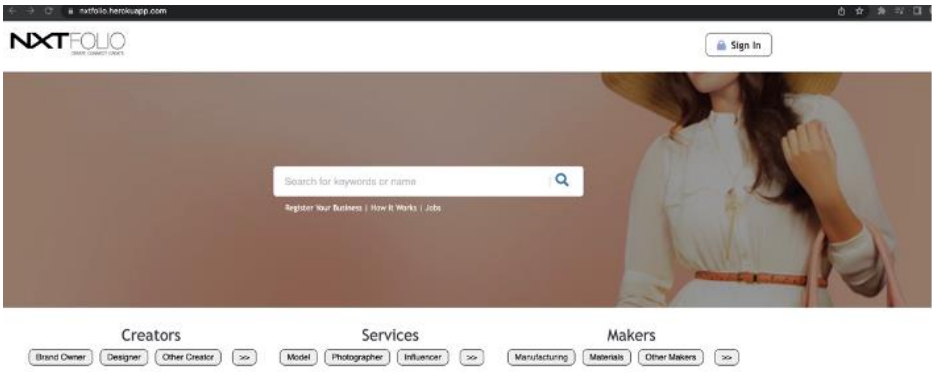
Points: 1

Implementation status: Accepted

Changes:

Deploy the application, utilizing Heroku for its easy deployment of ruby applications, with delays in the process caused by upgrading the ruby version and its libraries, and initially failing due to a missing storage.yml configuration file, but ultimately succeeding once added and all fixed features being deployed and working as intended.

Storyboards/Screenshot:



6. Fix sign up page

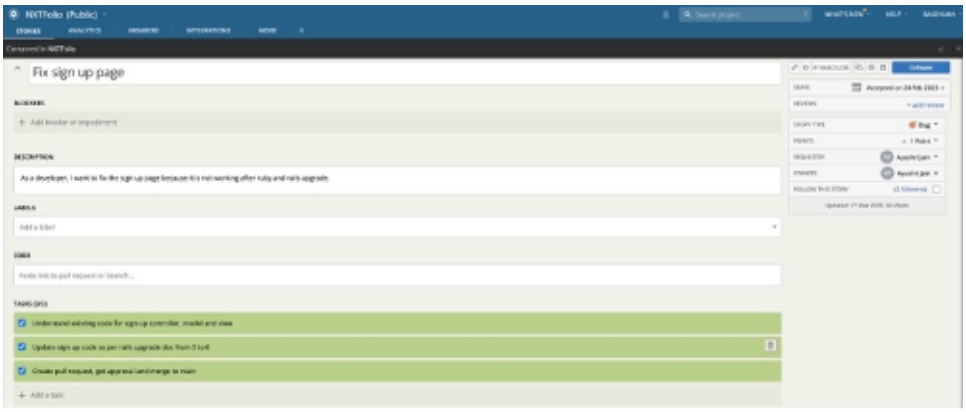
Points: 1

Implementation status: Accepted

Changes:

Solved the issues caused by upgrading to Ruby 3.2.0 and Rails 6.1.4.2 caused issues with the sign up and login pages.

Storyboards/Screenshot:



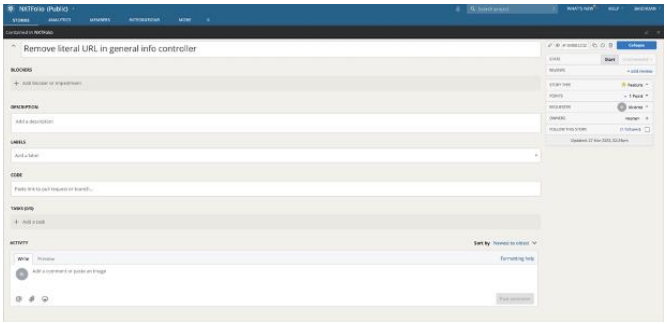
7. Remove literal URL in general info controller

Points: 1

Implementation status: Unscheduled

Changes: Remove literal URL in general info controller

Storyboards/Screenshot:



8. Legacy cucumber tests use selenium_chrome webdriver which does not work on docker nor E2C

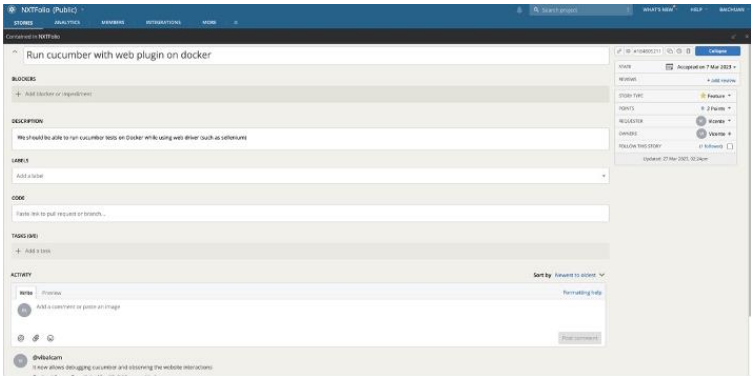
Points: 1

Implementation status: Accepted

Changes:

We added a configuration variable to enable/disable the remote webdriver and included the postgres dataset setting in the app's Dockerfile.

Storyboards/Screenshot:



9. S3 Bucket Integration

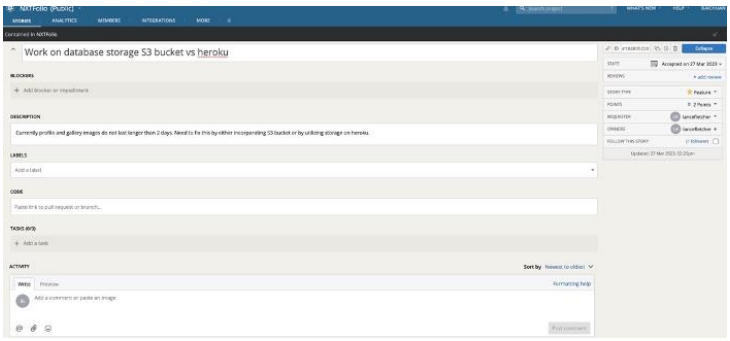
Points: 2

Implementation status: Accepted

Changes:

The previous team had issues with setting up Amazon S3 bucket, and the app would crash after a day or two due to the images being stored locally in the tmp folder on Heroku. We integrated S3 bucket successfully, and images can now be uploaded and retrieved from the bucket with proper security permissions.

Storyboards/Screenshot:



10. Refactoring To Do List

Points: 1

Implementation status: Accepted

Changes:

The list makes the code easier to understand and thus allow for quicker debugging and development. More details are contained within the “NXTFolio Refactor Doc”.

Storyboards/Screenshot: N/A

11. Form should keep its input when submit fails

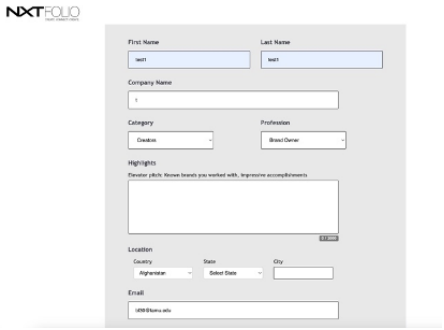
Points: 1

Implementation status: Accepted

Changes:

We improved the user experience during sign up by modifying the code to keep the information filled in by the user even if submission fails, allowing for easier correction and resubmission.

Storyboards/Screenshot:



12. Testing sign up and log in

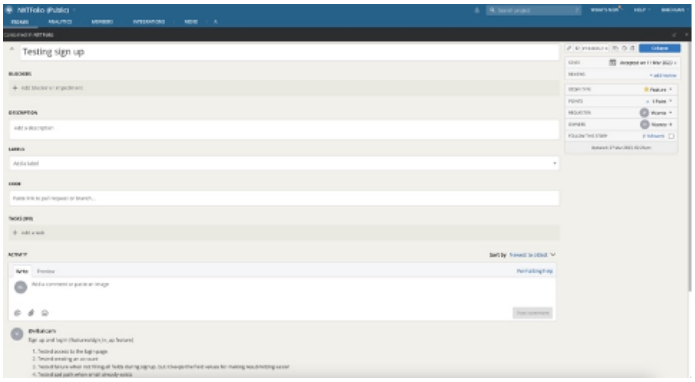
Points: 1

Implementation status: Accepted

Changes:

We have included 5 completely new tests for both happy and sad paths for the sign up and log in features.

Storyboards/Screenshot:



13. Travel Feature

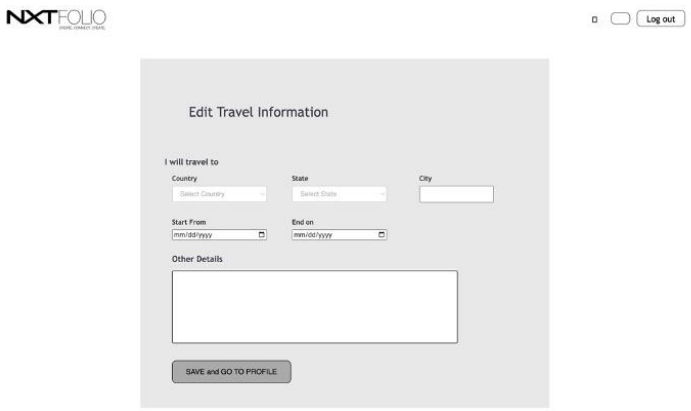
Points: 2

Implementation status: Accepted

Changes:

A new feature for displaying travel details has been added as requested by the client. Professionals can now add their upcoming travel details under Edit Travel Info in the dropdown menu of Edit Profile. The information is visible on their profile page and will be used in the future to update the search engine to display professionals who are traveling to a city within 30 days in search results.

Storyboards/Screenshot:



14. Follow Feature

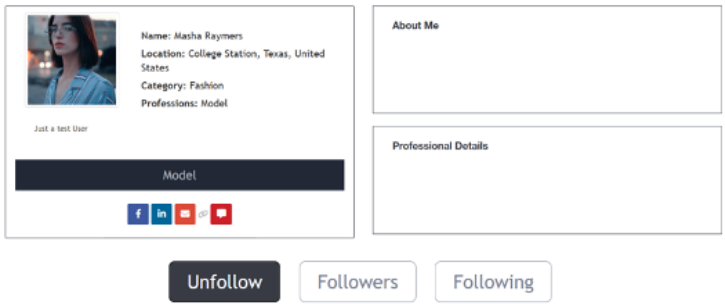
Points: 2

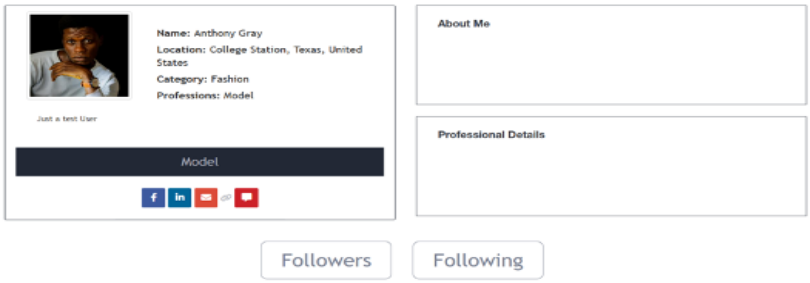
Implementation status: Accepted

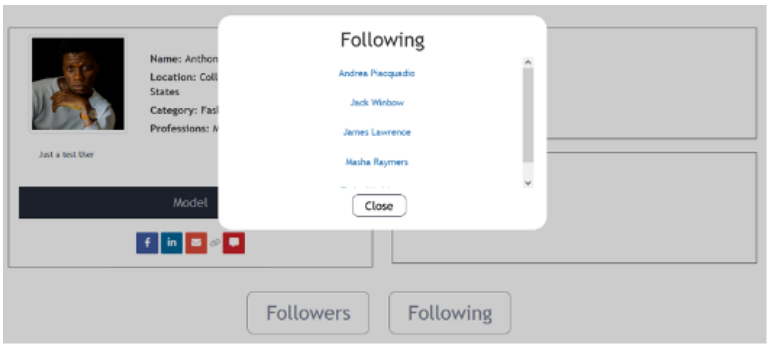
Changes:

The follow feature has been added to allow users to keep track of other users they are interested in. This is a uni-directional feature and is utilized in the new search algorithm to provide search results. A joining table has been created to support the many-to-many relationship between users, and the show_profile_controller and show_profile view have been modified to support the new feature.

Storyboards/Screenshot:







15. Home Button

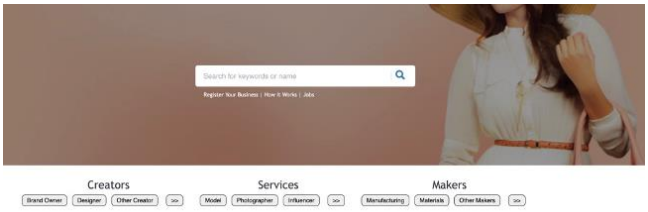
Points: 2

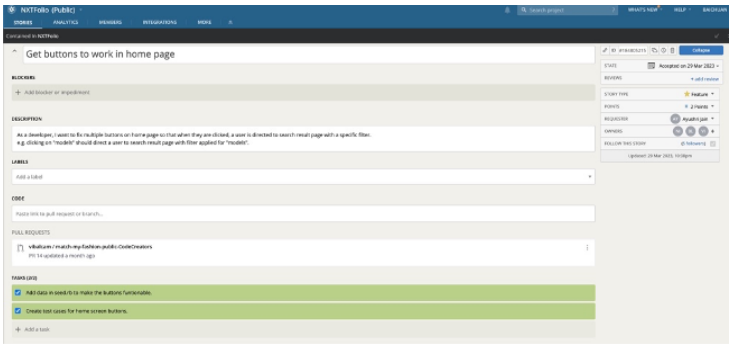
Implementation status: Accepted

Changes:

The buttons in each category at the bottom of the page act as filter buttons to quickly navigate to a page with a list of their respected category. Previously, the buttons fire, but did not navigate to a new page.

Storyboards/Screenshot:





16. Chat Feature Tests

Points: 1

Implementation status: Accepted

Changes:

We continued testing non-tested features from the legacy code. In this iteration we tested the chat feature.

Storyboards/Screenshot: N/A

17. Mobile UI (redesign pages)

Points: 3

Implementation status: Unstarted

Changes:

Buttons will be placed at the bottom of the screen, inside of a navigation bar to create a more user friendly environment.

Storyboards/Screenshot:

18. Tag Feature

Points: 2

Implementation status: Accepted

Changes:

We added a basic tagging feature for collaborators in projects as requested by the client. Users can tag others and send invitation emails if they are not part of the application. The feature will be

updated in the next iteration to show collaborated projects on collaborator's profile page and to tag using name instead of email.

Storyboards/Screenshot:

19. Recommend projects in homepage

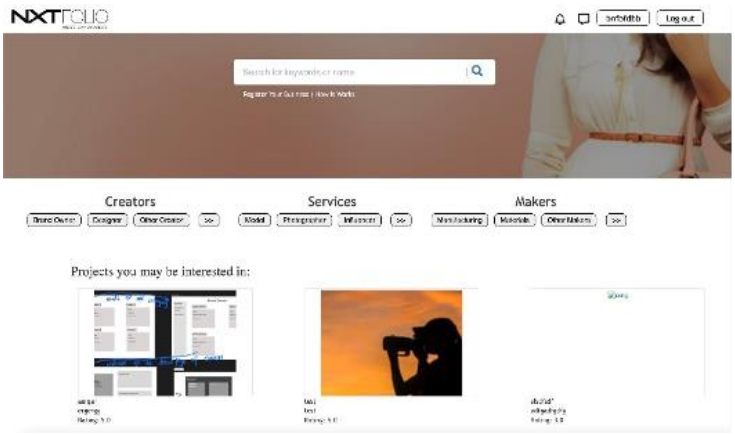
Points: 3

Implementation status: Accepted

Changes:

When users stay on the homepage, there will be a list of recommended projects with average rating >= 3.0 and listed with decreasing average rating order.

Storyboards/Screenshot:



20. Recommend profiles in no-result searching

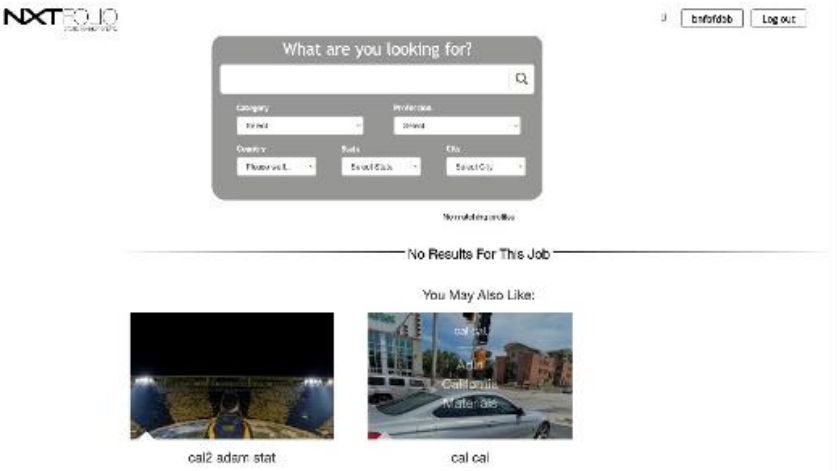
Points: 3

Implementation status: Accepted

Changes:

When there are no matching profiles in using the search engine, the page will show some recommending profiles based on login user's State and City. The profiles that match login user's city will be shown first, and the profiles match login user's state will be shown next.

Storyboards/Screenshot:



21. Country/state/city database

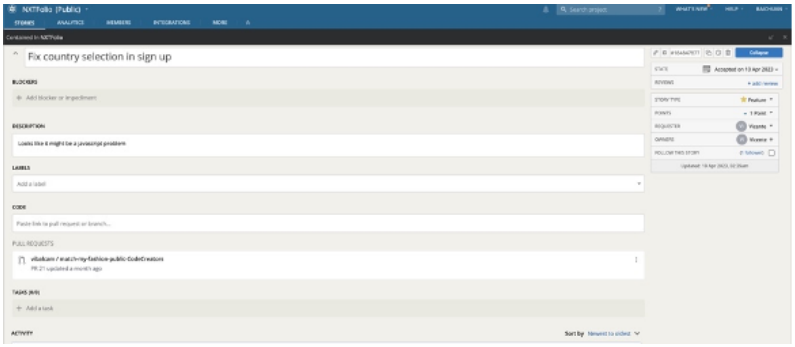
Points: 1

Implementation status: Accepted

Changes:

The app's external API for dropdown selectors stopped working, which would have affected new profile creation and other features. So, the team downloaded an exhaustive database of countries, states, and cities, and hosted it locally. They created models for the database, added seeding code, and created a controller and routes for state and city. Jquery code was written to make ajax calls to the local database when a user changes their selection.

Storyboards/Screenshot:



22. Update tagging feature

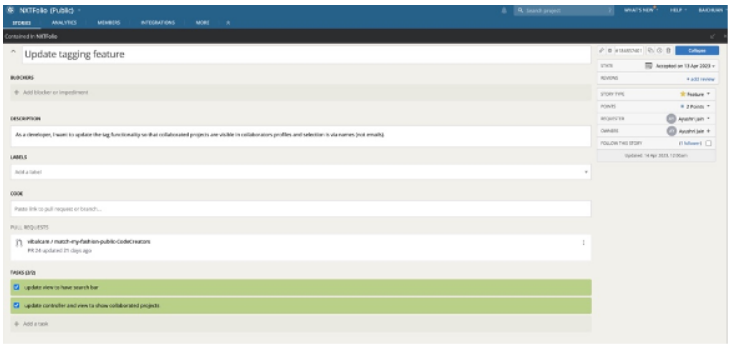
Points: 2

Implementation status: Accepted

Changes:

Tagging functionality was available via a dropdown (with email) in the last iteration. However, it had to be updated so that users could be tagged via their names using a search bar, not dropdown. Also, collaborated projects needed to be visible on collaborator's profile page.

Storyboards/Screenshot:



23. Mobile UI – Navigation Bar - Homepage / Profile Page

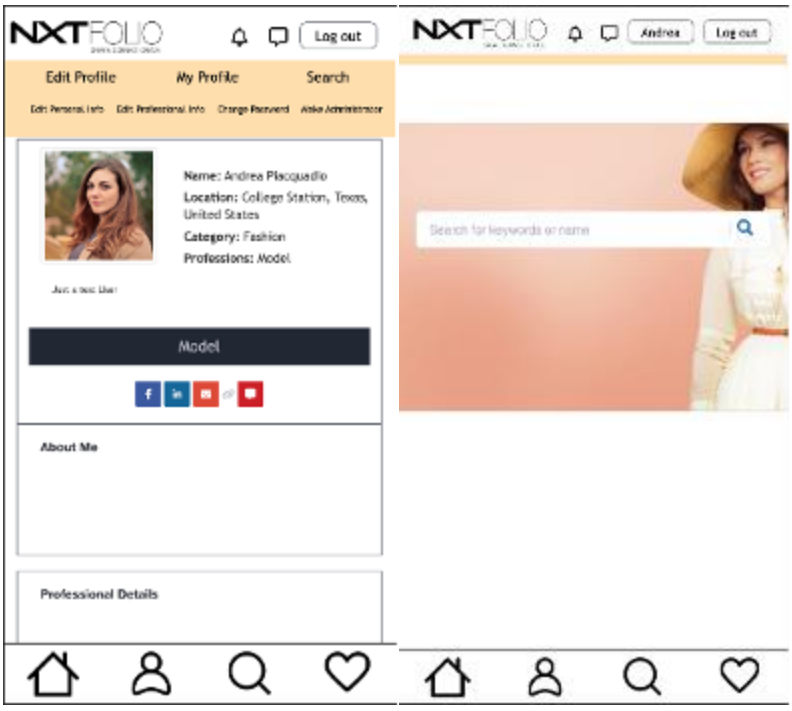
Points: 3

Implementation status: Accepted

Changes:

The pages were poorly formatted and required scrolling left to right to view everything on the screen. The pages have been reformatted to allow for up and down scrolling, making it easier for users to view all content. Additionally, a navigation bar has been added at the bottom of the screen to allow for easier navigation between the home page, profile page, search page, and review page in mobile view.

Storyboards/Screenshot:



24. GitHub actions and failing tests cleanup.

Points: N/A

Implementation status: todo

Changes:

Legacy tests are currently failing, making it difficult to ensure new features don't break existing tests. To address this, failing tests need to be refractored.

Storyboards/Screenshot: N/A

25. Moving to a unified GitHub and Heroku together with other NXT apps (Transfer Heroku and GitHub ownership)

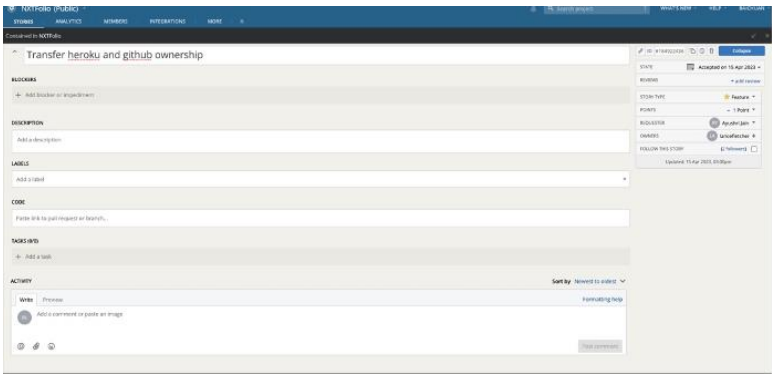
Points: 1

Implementation status: Accepted

Changes:

Ownership of the app on Heroku has been transferred to the customer's account and ownership of the Github repo has been transferred to the FashioNXT account. The Amazon S3Bucket, which stores pictures, will be transferred in the next iteration to eliminate the dependency on personal accounts.

Storyboards/Screenshot:



26. Job-related task

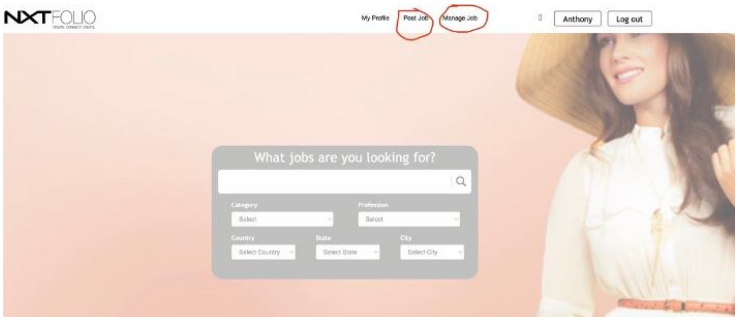
Points: 3

Implementation status: Accepted

Changes:

The latest iteration of the app includes new pages for job-related user stories. Users can now access the job page by clicking on a button on the home page. On the job page, users can perform three main functions: job-posting, job-management, and job-searching.

Storyboards/Screenshot:



27. Fix gallery (2)

Points: 2

Implementation status: Accepted

Changes:

We improved the gallery page by limiting the number of images to 5 per gallery and displaying a flash notice if a user tries to upload more than 5 images. We implemented the function that users can delete images individually. If the gallery has only 1 image, the users can only delete this gallery. We also created the page and functions that can edit each gallery, including editing gallery's title, description, and adding pictures to a gallery. Also, there can be no more than 5 images in a gallery, if the user adds more, there will be a flash notice.

Storyboards/Screenshot:

28. Country/state/city database rollback

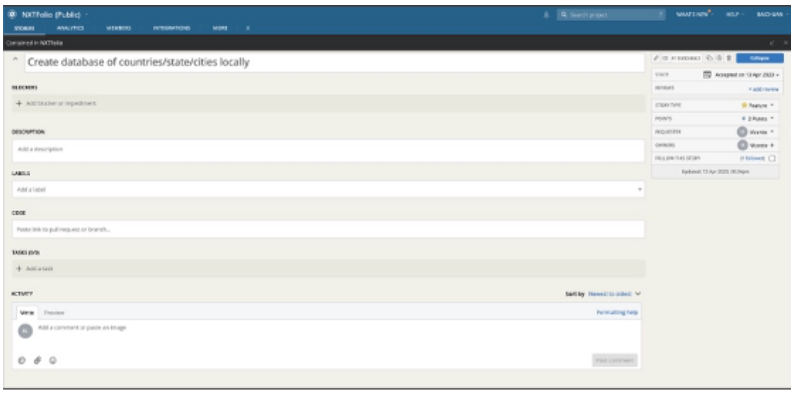
Points: 1

Implementation status: Accepted

Changes:

The country/state/city database exceeded Heroku's capacity, so we reverted back to the previous system after discussing with the client. Nonetheless, we kept the code and structures for the database for future use.

Storyboards/Screenshot:



29. Automatic cucumber and rspec testing on push

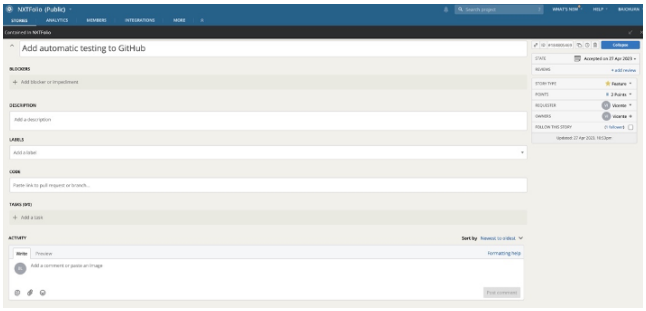
Points: 2

Implementation status: Accepted

Changes:

We created two GitHub actions to run cucumber and rspec tests whenever a push is made, or a pull request is created. This makes it easier to ensure that tests pass and to catch potential issues with new changes.

Storyboards/Screenshot:



30. Automate country/state/city update

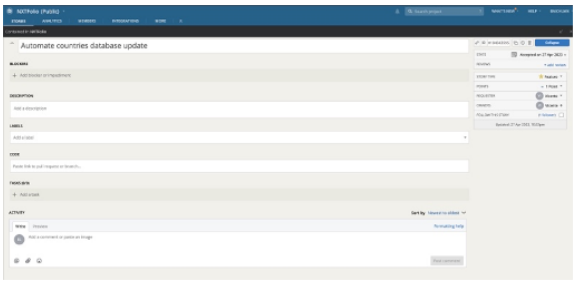
Points: 1

Implementation status: Accepted

Changes:

The task clears the database and reseeds it with the updated json file, which is obtained from a GitHub repo. The repo provides the information in json format and is updated every few months.

Storyboards/Screenshot:



31. Improved search algorithm

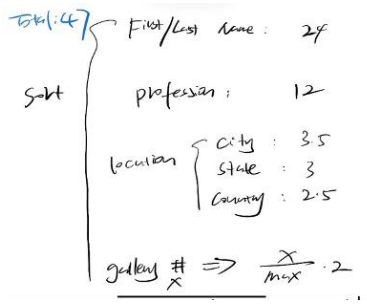
Points: 3

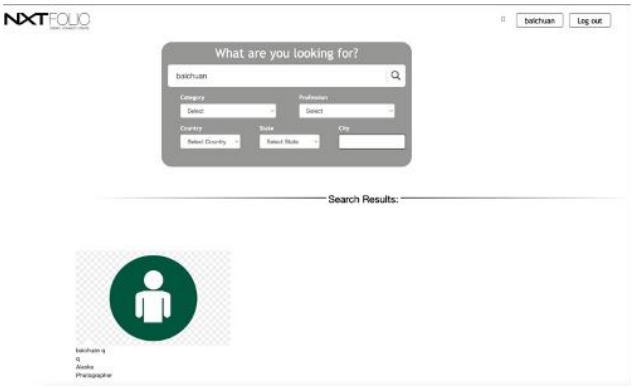
Implementation status: Accepted

Changes:

Improved search engine now allows keyword search in profile's highlights, specialization, prof details, compensation, and bio, and blocks "stop words". Empty search will not show all profiles. Results are rated and sorted by profession, location, name, and rating.

Storyboards/Screenshot:





32. Extra seeding

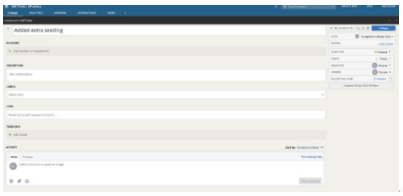
Points: 1

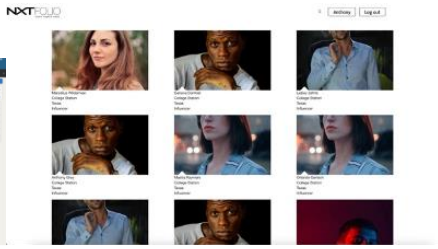
Implementation status: Accepted

Changes:

Added 200 fake users with random jobs and automated the process of creating fake users and galleries for testing purposes.

Storyboards/Screenshot:





33. Mobile UI

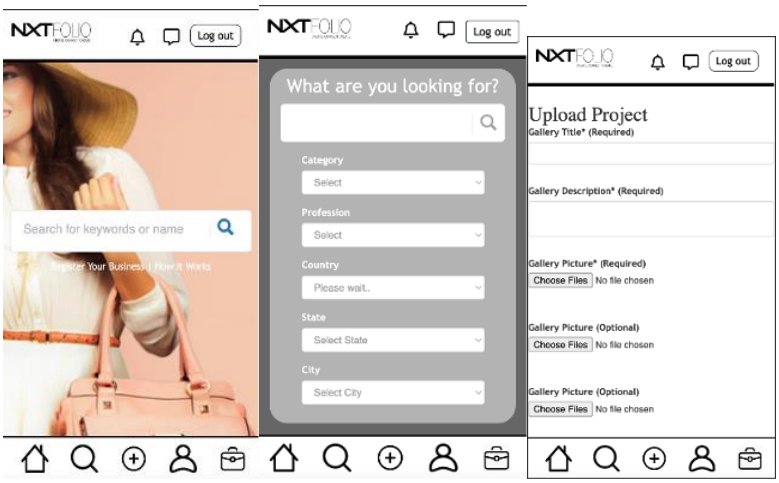
Points: 2

Implementation status: Accepted

Changes:

The mobile navigation bar has been updated to include a "Create Project" and "Job" button, with a layout similar to other creative sites like "Creatively". Additionally, the profile button has been moved from the header to the footer for easier navigation between the home page, search page, create page, profile page, and job page.

Storyboards/Screenshot:



34. CRM

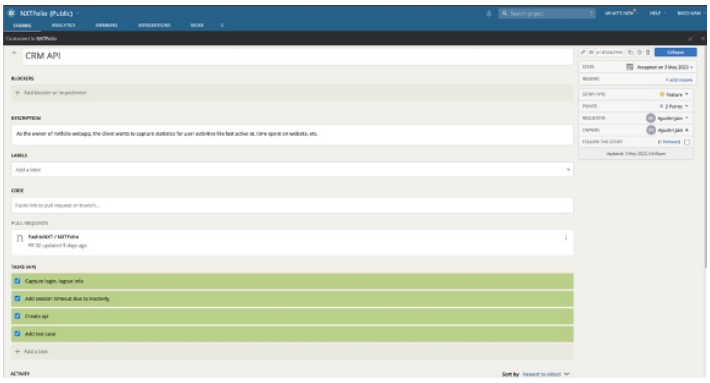
Points: 2

Implementation status: Accepted

Changes:

Application changes were made to track user activities and pull requests, but the pull request was not accepted due to comments from the professor about correct CRM implementation. The activity is on hold until further guidance from the CRMNXT team.

Storyboards/Screenshot:



35. Minor Fixes

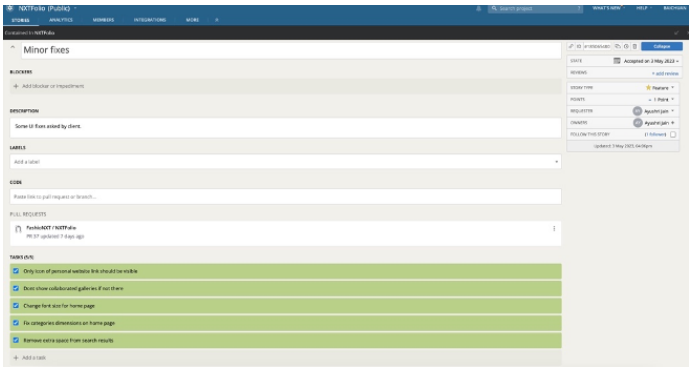
Points: 1

Implementation status: Accepted

Changes:

Client requested UI fixes including removing unnecessary space before search results, decreasing font size of "projects you might be interested in", not showing collaborated galleries if not present, making personal website link a hyperlink only, and making three subcategories visible on large screens on the home page. No test cases were provided.

Storyboards/Screenshot:



36. Full GitHub/Heroku Migration

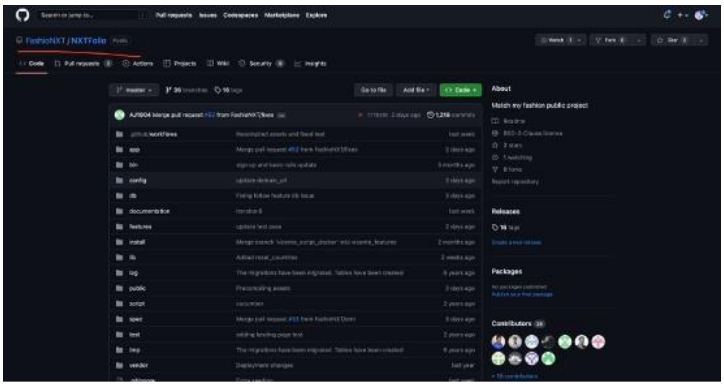
Points: 1

Implementation status: Accepted

Changes:

The application was migrated to the owner's GitHub account, and we ensured that Heroku was pulling from the new repo. We also coordinated with other FashioNXT groups to make sure their repositories and Heroku deployments were set up correctly.

Storyboards/Screenshot:



37. Work on database storage S3 bucket vs heroku

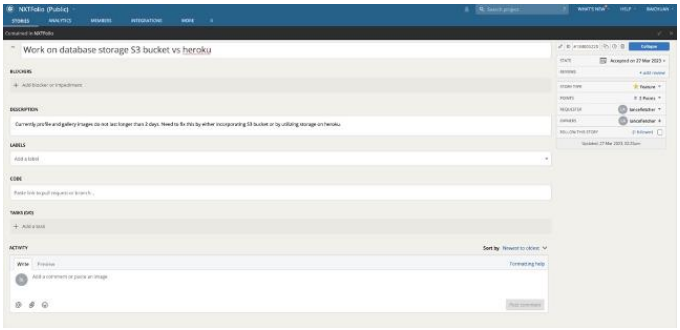
Points: 2

Implementation status: Accepted

Changes:

Profile and gallery images do not last longer than 2 days has been fix this by either incorporating S3 bucket or by utilizing storage on heroku.

Storyboards/Screenshot:



38. Job Management System

Points: 3

Implementation status: Accepted

Changes:

We finished the job management system. We can add a new job, show all jobs you created, edit a specific job, and delete the jobs.

Storyboards/Screenshot:

39. Job Search

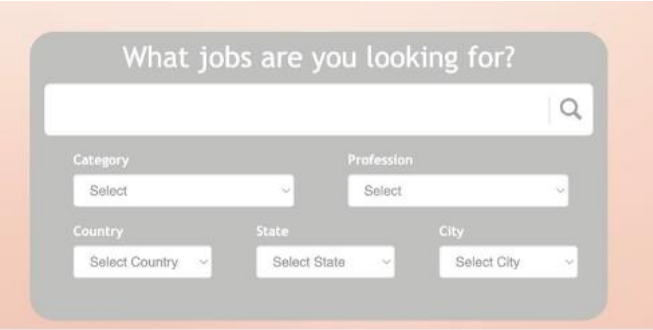
Points: 1

Implementation status: Accepted

Changes:

We finish the basics of the job search system. And we are working to improve the algorithms and the UIs.

Storyboards/Screenshot:



40. Low-fi mockups for job posting and travel feature

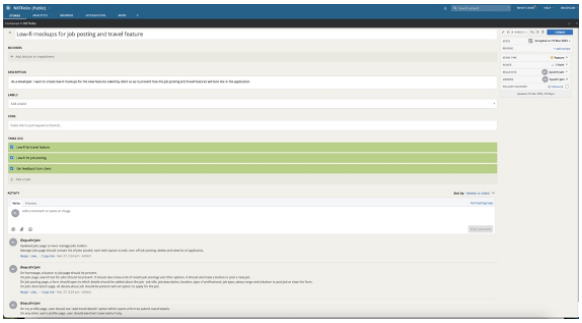
Points: 1

Implementation status: Accepted

Changes:

Create low-fi mockups for the new features asked by client to present how the job posting and travel features will look like in the application.

Storyboards/Screenshot:



41. Fix search UI

Points: 1

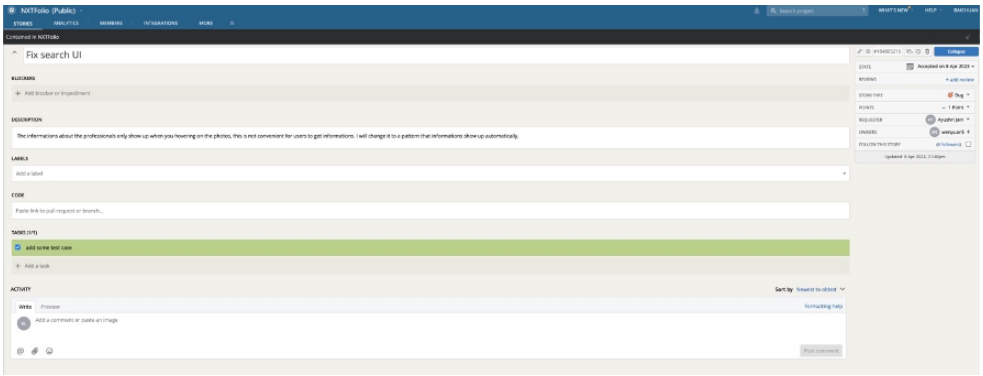
Implementation status: Accepted

Changes:

The information about the professionals only shows up when you are hovering on the photos, this is not convenient for users to get information. I will change it to a pattern that information shows up automatically.

Storyboards/Screenshot:

Alissa Fields
College Station
Texas
Designer



42. Fix country selection in sign up

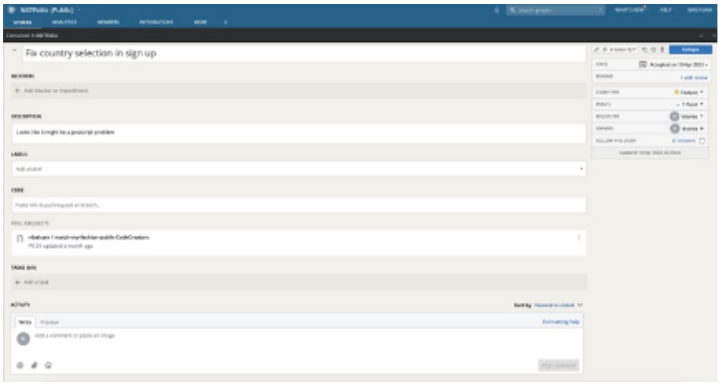
Points: 1

Implementation status: Accepted

Changes:

Fix country selection in sign up caused by JavaScript issue.

Storyboards/Screenshot:



43. Add time variable for search

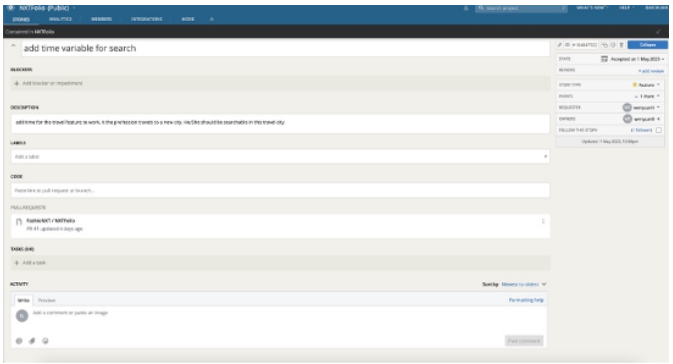
Points: 1

Implementation status: Accepted

Changes:

Add time for the travel feature to work, it the profession travels to a new city. He/She should be searchable in this travel city.

Storyboards/Screenshot:



44. Follow Feature Testing

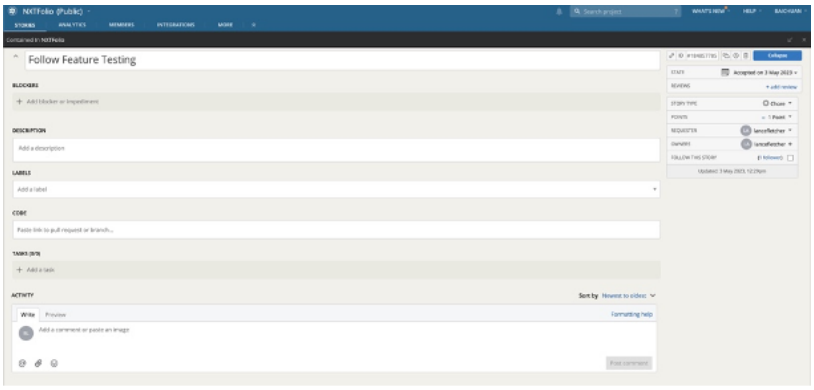
Points: 1

Implementation status: Accepted

Changes:

Tests for the follow features are added.

Storyboards/Screenshot:



45. Profile page details fix

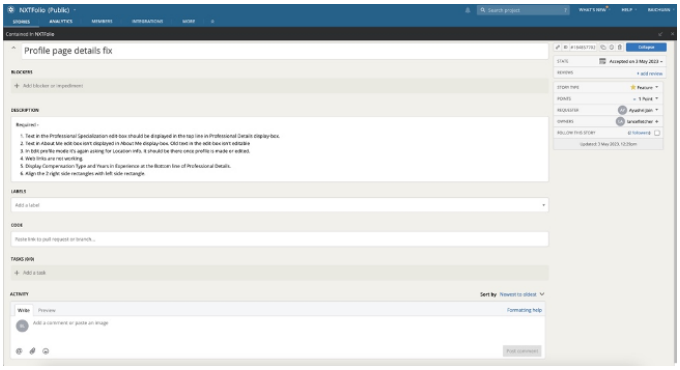
Points: 1

Implementation status: Accepted

Changes:

Fixes were made to the application including displaying text from the Professional Specialization edit-box in the top line of the Professional Details display-box, fixing the issue with text not displaying in the About Me display-box, correcting the Location info prompt in Edit profile mode, fixing broken web links, displaying Compensation Type and Years in Experience at the bottom line of Professional Details, and aligning two right side rectangles with the left side rectangle.

Storyboards/Screenshot:



46. Fix Search Page and Create Page Mobile View

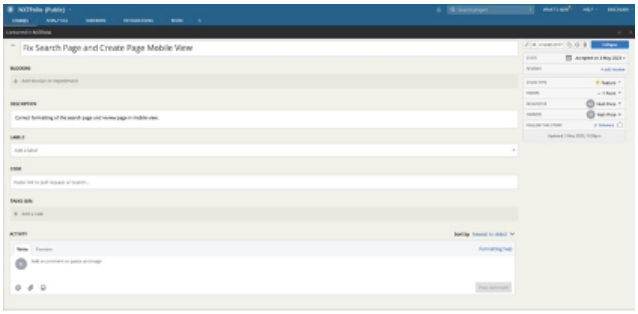
Points: 1

Implementation status: Accepted

Changes:

Correct formatting of the search page and review page in mobile view.

Storyboards/Screenshot:



47. Job search page

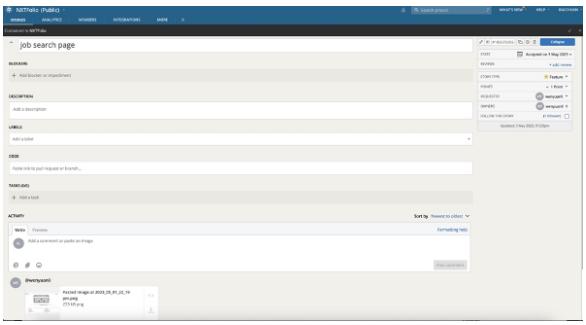
Points: 1

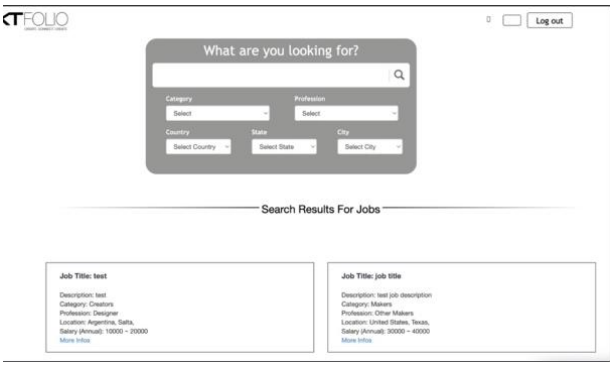
Implementation status: Accepted

Changes:

Design search pages for job search engine and result pages.

Storyboards/Screenshot:





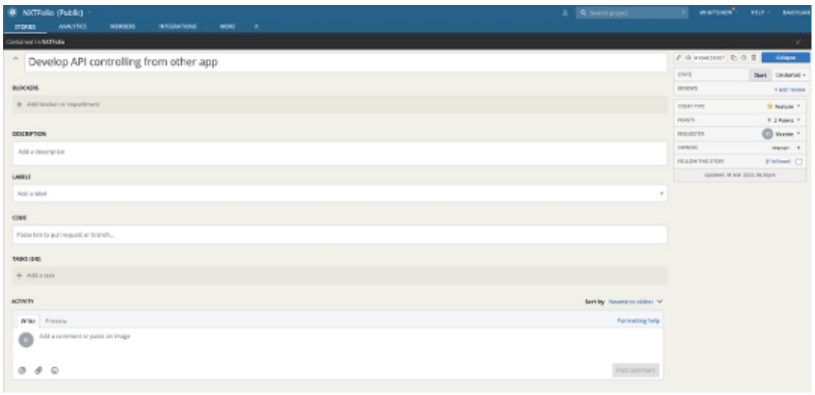
48. Develop API controlling from other app

Points: 2

Implementation status: Unstartled

Changes: N/A

Storyboards/Screenshot:



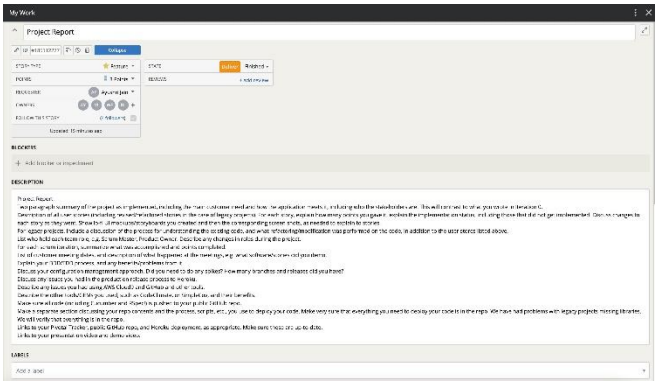
49. Final Report

Points: 3

Implementation status: Accepted

Changes: Final Report for the project.

Storyboards/Screenshot:



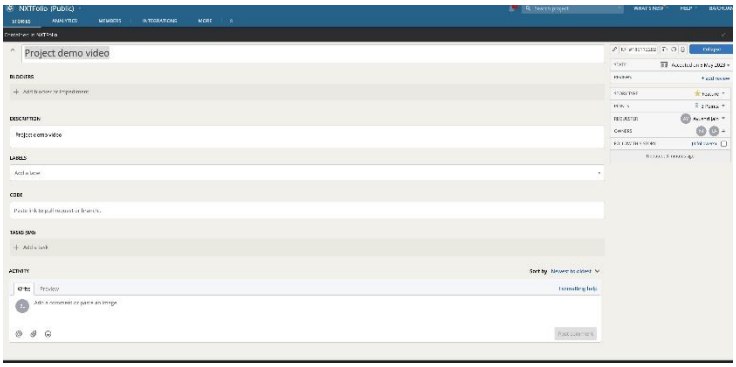
50. Project demo video

Points: 3

Implementation status: Accepted

Changes: Final demo video for the project.

Storyboards/Screenshot:



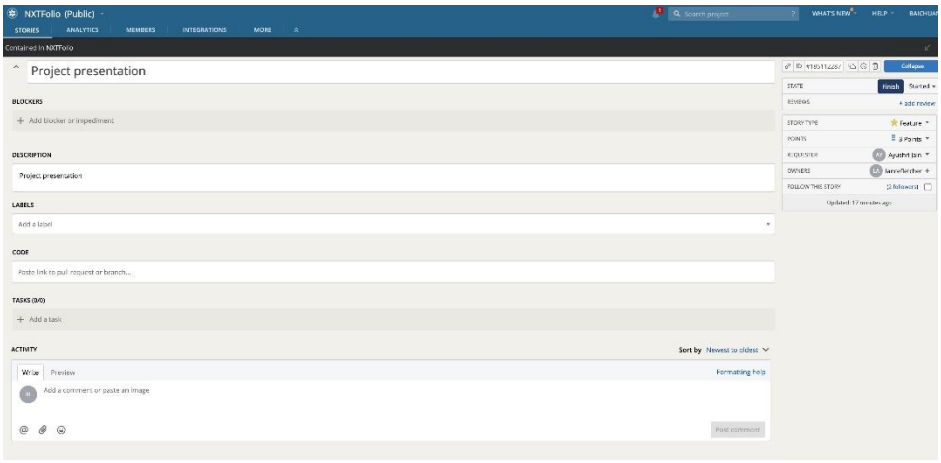
51. Project presentation

Points: 3

Implementation status: Accepted

Changes: Final presentation for the project.

Storyboards/Screenshot:



Understanding Legacy Code

The first step is to get the project to work locally. For that we recommend looking at the README and following the steps to set it up with docker compose. This will make it easier to develop and run cucumber tests (the tests use a webdriver which might be hard to set up in E2C or a single docker). We recommend taking advantage of the GitHub action which automatically runs cucumber and Rspec tests for pushes to master and pull requests. It will save time reviewing pull requests and ensure that new features do not break anything. Due to several issues which we will mention later, there are currently many tests which we did not have time to fix. We recommend taking the first week or two to fix these tests. This will help familiarize yourselves with the code and the app, plus having all tests passing allows checking only

for a tick or a red x when the GitHub tests run instead of having to check whether the failing tests were already failing. It might take some time, but it will really pay off later. These automatic tests were introduced in the final weeks of our team's work, but we introduced them since we believe they will be extremely helpful for any team that comes behind. We also recommend running 'bundle update' often to ensure the gems are up to date which will make it easier to maintain the project. To familiarize yourself with the general functioning of the project. look at the models, which handle fetching and storing data and define the objects used by the app. The routes file is also especially useful to know what pages correspond to what controllers and will help you debug and understand the relation between different pages. For cucumber testing, we recommend using the "I debug" step which allows pausing the test at a certain point, making it easier to develop the tests and check what the app is doing.

This app has been worked on for several teams now. When we first started working on the project, it had several bugs and issues. We have focuses part of our time to refractor the app and to make it easier for the next teams to work on it:

- The project was using an old ruby version (2.4) and obsolete dependencies (therubyracer). This not only gave problems setting up production on Heroku, but also the development environment. We updated the ruby version to 3.2, updated the gems, and substituted 'therubyracer' for 'Nodejs'.
- Issues setting up development environment due to migration errors, setting up Postgres database, instructions with missing information, and not a pre-established environment. We decided to use docker for development, which would allow everyone to have a free and known environment. We also created a script that downloads all necessary dependencies and helps set up the database, ruby, gems, etc. We also updated the README with up-to-date instructions. We have the new instructions, docker, and the script makes it possible to have the app up and running in just a few minutes.
- The project uses a webdriver to run cucumber tests. Installing and setting up a webdriver on E2C or docker is not straightforward, which would require doing the testing locally, which is something we wanted to avoid having the same development environment. The previous instructions did not mention anything about this, which made it harder to solve. We set up a docker compose script to make it easy to run at the same time the development docker and a docker webdriver that will work together to run the cucumber tests. Additionally made, the webdriver docker can be used to check in the browser what the tests are doing, making them easier to debug. We included an additional section in the README with the steps to get this set up. The use of docker compose makes it so it is as easy to set up as the previous docker mentioned, making it the recommended environment to use.
- The previous report mentioned that they were having problems with Heroku, with the project crashing every few days and having to be redeployed. Additionally, the images were not being saved for long due to Heroku deleting them. We fixed the issues related to Heroku and set up an S3 bucket where the images are being saved so they do not get deleted.
- Several features implemented were not working, such as the home buttons. We fixed these features, improved others (e.g., sign up not saving progress when some information was not entered correctly), and added new tests (e.g., login and sign-up tests or home buttons tests).
- There was no seeding for the database to test the app. We added a seeding file and, with the help of the 'faker' gem, allowed to create an infinite number of users to test the app. In the last week we added 200 users to make it easier for the next team to develop and test the improved search and other features that may be desired.

- Several already implemented features had their tests failing or did not have tests. Moreover, the cucumber steps were repeated in different files with very specialized steps. Apart from writing tests for our own features, we added tests for some of these features increasing the coverage. We also added more general steps that can be reused reducing the work to create new cucumber tests.
- The country, state, and city lists were obtained through ajax calls to an external API. This API stopped working suddenly. After consulting with the client, we incorporated this information locally to reduce the dependency on the third-party API.

Production Issues

Heroku was a very valuable tool and significantly simplified the deployment process. However, there were a few difficulties that occurred. At one point when trying to migrate the database an error stated that a specific column already existed in the database table. It seems as if the application did not know the migration had already been applied, which caused it to throw the error. We're not sure if this was a Heroku issue or GitHub issue, but it was quite difficult to debug at first. The biggest issue we encountered was some of our views being rendered differently than how they looked in our development environments. We had to precompile our assets folder and push it to Heroku to fix the issue. This issue is not unheard of when using Heroku. Another issue we had was storing images on Heroku since the images got deleted every few days. To solve this, we set up an S3 bucket and save the images there instead. Overall using Heroku was very beneficial to the project but when an issue did arise, it was not always the easiest to debug.

E2C and GitHub issues

While E2C is very useful to have a standardized development environment, it became difficult to use for cucumber testing with a webdriver. Moreover, not having full access to this system can sometimes make it harder to debug. We chose to instead use a standardized Ubuntu docker image which allowed scripting the installation process. While this script normally works without issues, we sometimes had issues with installing postgres and authorization issues for the database in Ubuntu. We included a section in the README that goes over some issues we came across so other teams can check how we fixed them. We encourage other teams to keep adding to this list if issues arise. We also set up a docker compose script to make it easy to run the development docker and a docker webdriver together to run the cucumber tests. On some occasions working with a webdriver for cucumber tests can sometimes be a bit stressful due to the webdriver docker freezing, and thus having to be restarted, or being very slow.

Configuration Management Approach

Our team utilized GitHub as our version control system, following a feature-branching approach with a code review system. Instead of working directly on the master branch, we created branches for individual features, and each pull request included the corresponding test command to facilitate the reviewer's job. Whenever issues arose during the review process, we resolved them through peer programming, with occasional guidance from the TA and Professor. This approach allowed us to effectively manage changes to our codebase, ensuring high quality and collaboration throughout the project's lifecycle.

Currently, we have 36 branches in git repository out of which 24 have been merged into the master branch and the remaining are stale branches. Around 50 pull requests were created and reviewed.

For each scrum iteration, summarize what was accomplished and points completed.

Iteration 0

Our first step is to gain an understanding of this application. We watched the demo video and tried out the features in the application. We arranged our first meeting with the client and explained why we chose this project. The client gave us some demands regarding the features that they wanted us to design.

After that, we spent a lot of time deploying the application. We encountered many difficulties during the deployment process, so we wrote brand new instructions in the readme to help others with the deployment. We also tested the deployment on different platforms such as AWS Cloud9, EC2, Ubuntu 18, Mac, and Windows. Unfortunately, the application was too outdated, and some of its features had broken. We identified these broken features and decided to begin development by fixing them. This approach would also help us get familiarized with the code.

Using our own ideas and following the client's requests, we designed several user stories to be completed during the semester. These user stories include autofilling emails, adding dimensions to the rating system, creating pages that show all creators, specifying proper error messages for unsuccessful logins, autofilling locations on the edit pages, rating every project, and implementing better searching algorithms.

Points completed: 0

Iteration 1

The main task for this iteration was to fix the bugs and broken features in the legacy code. Our most significant accomplishment was upgrading the ruby version from 2.4 (which is quite old) to the latest version, 3.2. Due to the significant gap between these two versions, some of the gems, dependencies, and functionalities broke during the upgrade. We had to identify new dependencies and functionalities that would work for the new ruby version. We also addressed some problems related to the database migration issues. To simplify the installation process for future teams, we recommend using Docker, as it isolates everything.

Another important achievement of this iteration was deploying the application to Heroku. We made the necessary updates and fixes and successfully deployed the application on the Heroku platform.

Due to the upgrade and the remaining bugs from the previous team, we spent time fixing broken features. The main features we checked and fixed were the gallery issue, sign up page, RSpec tests, and search function. We also seeded the database for development and testing purposes.

Points completed: 4

Iteration 2

During this iteration, our first task was to find a way to get the selenium chrome web driver to work, which is needed to run the cucumber test. We used docker-compose to combine our app container with a selenium docker container. We also modified the configuration variables to enable the remote web driver and, in the end, successfully ran the test.

We did not have a place to store the images. The images will be deleted after 2~3 days. So we integrated the AWS S3 bucket with the app. This enabled the images to be uploaded, stored, and retrieved from the bucket.

After examining the app, we created a 'NXTFolio Refactor Doc' that addressed bugs or areas that needed to be improved.

During this iteration, we fixed several broken features, such as the form deleting inputs when submission failed instead of keeping them. We also added new tests for the login and sign-up features, designed low-fi mockups for the job posting and travel features. We finished the travel feature, which allowed professionals to edit and display their upcoming travel details. We also wrote tests for every new feature added. According to the RSpec test coverage report, there was a significant improvement in coverage (9% to 14%).

Points completed: 4

Iteration 3

In this iteration, we completed the Follow feature, which allows users to follow or unfollow others, and view information about their followers and who they are following. When someone follows the user, a notification will be sent to inform the user, and testing for the Follow feature is in progress. We also made previously non-functional home buttons into filter buttons that allow users to quickly navigate to pages within their respective categories. Tests were written for this feature. Additionally, we added tests for the chat (direct message) feature, which had not been tested by the previous team. As per the client's request, we began making the app more mobile-friendly and compatible. We added a tag functionality, which allows users to tag other users as collaborators on their projects. Collaborated projects will show up on users' profiles, and email notifications will be sent. Recommended projects will now appear on the home page and in no-result searches. Finally, we adjusted the search range to incorporate the travel feature, allowing

users to search for professionals in the city they are traveling to. As I mentioned before, every new feature has tests before they are merged to the master branch. We also added the new features to the design diagram.

Points completed: 9

Iteration 4

In this iteration, we encountered an issue that the previous selector for country, state, and city, which relied on an external API, suddenly broke, affecting all location selections in our app. To prevent this problem from occurring again and provide a solution, we downloaded a database of countries, states, and cities, and added controllers, models, and seed files to Get the data. We also started exploring options for automatically updating the location database, as per the client's request.

We made some modifications to the tagging feature, making it easier to find collaborators and display collaborated projects on the collaborator's profile page. We continued working on improving the mobile UI and fixed navigation, home, and profile pages. We also commented out some failed legacy tests and worked on adding automatic testing functionality using GitHub.

We spent time transferring ownership of the app on Heroku, the repo in GitHub, and the AWS S3 Bucket to the FashioNXT account. This was done to eliminate dependencies and charges on our personal accounts, and to make it easier for future teams to work on the app.

Since all the app's functionalities were for professionals, we started building pages for job-related user stories, including job-posting, job-management, and job-searching. We also fixed several gallery issues, such as broken creating-gallery, image number limit, deleting individual images, and editing each image's information.

Points completed: 17

Iteration 5

Due to the large number of cities, Heroku does not support so many lines of data. To address this issue, we changed the city selection to an input field. We provided instructions on how to obtain and update the latest country/state/city database. Additionally, we set up automatic testing for each pull request on GitHub using Cucumber and RSpec.

We also improved the search algorithm to make all information on a professional's profile searchable. The search engine filters out irrelevant keywords like "he", "she", "able", and "like", and displays the results sorted based on a matching algorithm.

To facilitate testing, we seeded the database with additional data. We continued to work on the mobile UI. We started to work on unifying multiple FashioNXT apps into a single login system and database. We resolved any remaining issues related to the GitHub/Heroku/S3 Bucket migration.

The job search and management systems are now functional. Users can create, edit, delete, and view all their posted jobs, as well as search for jobs. We also made some minor UI fixes based on the client's feedback.

Points completed: 12

Points completed after iteration 5: 23

Client meetings

We met the client weekly on zoom and discussed about new features, got feedback for implemented features and discussed about issues. Below is the summary –

Date	Description
02/03/2023	Introduction meeting, expectations, and functionality of legacy application, emphasized on using AI to improve the app
02/17/2023	Discussed about problems in deploying app due to older version of ruby and rails, client shared PowerPoint presentation on the expected functionalities
02/24/2023	Showed deployed app to the client, showed dummy profiles (created from seed files), informed about broken features
03/02/2023	Demo on fixing sign up, client asked to add travel feature
03/09/2023	Demo on fixing home page buttons, discussed problem of images disappearing from S3 bucket
03/23/2023	Demo on image persistence in S3 bucket, client asked for recommended projects on home page, mobile view like LinkedIn app, follow feature, job posting, tagging, time parameter in search
03/30/2023	Demo on tagging feature, client emphasized for search engine improvement
04/06/2023	Demo on follow feature and extra seeding, discussed about problem with country, state, city due to 3rd party api, problem with gallery, ownership transfer
04/13/2023	Demo on removing 3rd party api for country, state, city problem, demo on improved search algorithm, updated tagging feature and fixed gallery, client asked for crm api, discussed about GitHub automated actions
04/20/2023	Discussed about rolling back database due to country, state, city problem, demo on travel feature and improved search algorithm, discussed about minor fixes
04/27/2023	Demo on improved search, crm api, job posting/search/management and minor fixes, discussed about last minute improvements and future scope
05/04/2023	Final demo of deployed app with all features

Explain your BDD/TDD process, and any benefits/problems from it.

BDD means behavioral driven development. It means we should start by thinking about the user stories. We should consider ourselves as customers or clients who use this application. In addition, we should think about what we need and what we want to obtain from this app. We must collect user demands and then write down the user stories at the beginning. Then, based on the user stories, we should create some lo-fi and plots to illustrate the features. We should also talk and communicate with the clients to make sure that everything is clear. After that, we start software development.

We used Behavioral Driven Development throughout the whole project. We met with the client every week. We communicated a lot to ensure that we were clear about what the client wanted. We had some conflicts when our opinions about how to design the features diverged. Also, sometimes what the client wants is technically hard to achieve. But we solved these problems smoothly through peaceful communications and discussion. We benefited a lot from BDD. We were clear about what we needed to build, and we didn't have to waste time wondering if we should do it this way or another way during development. Although we spent some time designing the user stories, we saved a lot of time in total. We can incorporate every client's need into our development.

TDD means Test Driven Development. After we figure out what we need to build, instead of beginning development, we should start by writing tests. The tests are a good way to illustrate how we want the features to be and to understand the clients' demands and the functionality that the new features should have. By developing according to the tests, there is a lower chance of making mistakes, and the possibility of diverging from the goal and user stories becomes much smaller with the tests already written.

At the beginning of our development, we found that the test coverage for the previous features was extremely low (around 10%). So for some selected important features, we made up tests for them. Although we did not start by writing tests (as the features were already built), writing tests for existing features was also meaningful. This way, we can ensure that the current features work well and do exactly what we want, and that there are no bugs in the code.

We made a strict rule that if there are no complete and comprehensive tests or if the tests fail, the pull request must not be approved and merged. Every member in our team sticks to this rule. When we build new features, we begin by writing tests (cucumber or rspec). Writing tests do take a lot of time, sometimes even more than the development time of the features. However, it is beneficial because we have a clear goal in mind, and passing the tests means the new features are working satisfactorily. In this way, we believed that we saved a lot of time in total.

Future Scope

This section summarizes the future tasks that the client suggested along with some features that we identified for this application to be its better version.

- Add Contracts and Disclaimer Page – A page for users to click a box to Agree on the terms of using this app while signing up.
- Reduce footprint – Since the application deals with images, it is necessary to incorporate code that reduces file-size of uploaded images automatically.
- Improved notifications – Add notifications for different activities: Tagged as collaborator and Project got rated. Make notifications more prominent like existing social media apps.
- Improved DM – Add group chat feature. Add search bar on DM page to search for user(s) you want to send text message.
- Improved gallery –
  i) Give the user an option to select the cover image of each gallery.
  ii) Give word limits in Gallery Title (40 char), and Gallery Details (200 char)
  iii) Mobile view for Gallery (currently, picture overlaps with texts below)
  iv) Better Viewing of Gallery: Not needed to open a new page to see the gallery and description, rather, pop up
- Improved collaboration –
  i) The Invite Collaborator field needs to have separate field for Collaborator's name (currently it only has email field), and the name need to be shown as a Collaborator. When invited collaborator creates a profile, the name should automatically embed the Collaborator's profile link to the name, and as usual show this project to the Collaborator's page.
  ii) Collaborated user must approve the Tag he/she's collaborated on, and can break Tag (option under the gallery in his/her own page)
- Image proximity search – Since the application deals with images, it makes sense to add an image search option (find professionals whose projects contain images like this image).
- Improved job – Give user option to apply for the job (either a form or a DM), look at the job owner's profile.
- Add AI chatbot – The client is seeking to utilize the capabilities of artificial intelligence to generate content for the "About Me" and "Highlights" sections of their personal website and has expressed interest in using an API such as ChatGPT to achieve this goal. Also use AI in search page to help find better match by being a helping hand for the user.
- Mobile view – The current user interface does not work well in mobile devices. Significant efforts are required to make that happen.
- Improved search – While the current search is quite good, it can be improved further using machine learning as more data is gathered when users start using the applications. Suggesting profiles or projects based on user's search history can be the first step.
- Refactor code – This project started in 2017 and has been worked upon by multiple teams with different coding styles. Since it is a Ruby on Rails project, efforts can be made to make it more Ruby-ish.

- Improved rating – Currently, there is only 1 dimension rating for a project. Multiple dimensions like "how good the theme is", "how good the images look", "overall rating", etc. can be introduced along with the option to add comments while creating a rating.
- Adding more tests – Add more cucumber and rspec tests to improve the coverage and get rid of bugs.

Tools/GEMs we newly added or updated

1.Update ruby 2.7.5 to ruby 3.2.0

Benefits: Updating from Ruby 2.7.5 to 3.2.0 improves performance, new features and enhancements, better error handling, and increased security

2. Update rails 5.0.7.2 to 6.1.4.2

Benefits: Updating from Rails 5.0.7.2 to 6.1.4.2 would provide benefits such as improved security, better performance, and access to new features. The latest version also has better support for modern web development practices and easier maintenance due to improve code organization.

3. Added aws-sdk-s3

Benefits: AWS SDK for S3 provides benefits such as simplifying S3 API calls, improving security with IAM roles, and enabling easy uploading and downloading of files to/from S3 buckets. It also offers improved performance and scalability, and compatibility with other AWS services.

4. Added faker

Benefits: This gem provides fake data for testing and development purposes. It can generate various types of data like names, addresses, phone numbers, etc. automatically, which helps in creating test data quickly and efficiently.

5. Added Rubyzip

Benefits: This one allows us to work with ZIP files in Ruby. It provides an interface for reading and writing ZIP files, as well as manipulating the files within them. Its benefits include the ability to compress and decompress files, create and extract archives, and add or remove files from an archive. Also, we used this to avoid update ruby to 3.0.

6. Delete therubyracer and mini_racer

Benefits: Because we updated the ruby version to 3.2.0, these 2 gems are not needed any more, also therubyracer can not normally works in Mac with M chips.

7. Added imagemagick

Benefits: It can be used to resize, crop, and convert images to different formats. Its benefits include a wide range of image editing options, fast processing times, and compatibility with a variety of programming languages and platforms.

Repo contents and the process

How to start a local server:

1. git clone the repository to your local machine

https://github.com/FashioNXT/NXTFolio.git

2. Once the files are downloaded, follow the standard procedure of installing ruby on rails. Click here

3. It is recommended to use rvm over rbenv.

4. Once Ruby on Rails is installed, you have to install the dependencies available in the gem file. Important to note that the dependencies and other rail commands will only work, when installed rails version and ruby version matches the versions mentioned in Gemfile.

5. To ensure the same versions are installed, open Gemfile, and check the ruby version. Do the following:

rvm install ruby-3.2.0
bundle install

6. Install the bundler ‘gem install bundler’

7. Bundle install to install all the dependencies in Gemfile bundle install

8. Start the rails server rails server

9. The local server will start on the following link.

http://localhost:3000/

If there is any problem with the postgres database. Try the steps below:

1. Install postgres brew update brew install postgres

2. open another terminal to run PG server postgres -D /usr/local/var/postgres

3. Open PG console (Open one more terminal to run the code below) psql postgres

4. Create admin user (more details in /config/database.yml) create user beaverthing; create password beaverthing; type in password(ubuntu) twice

5. Create database follow the previous team’s setting create database workspace_development; create database workspace_test; create database workspace_production; PG console setting completed. You can close the console here.

6. Generate table with database

bin/rails db:migrate RAILS_ENV=development

7. Start the rails server rails server

8. The local server will start on the following link

http://localhost:3000

What will you see on the web app?

1. Home page with recommending projects based on current user's location

2. Login/Signup

3. My Profile with galleries, and followers and following users.

4. Rating/ Review feature on application

5. Admin user has been added, with privilege for using the admin APIs. This is only available to the single admin user, added by the developer. Contact admin for help

6. Direct Messaging (DM) and Notification feature on Profile Details page.

7. Forgot Password option in the Sign in Page.

8. Search with keywords matching for profile's name, bio, compensation type, and highlights. And filtering by country, state, city, and professions.

9. Search results will be sorted by a logic based on rating, gallery numbers, names and location.

10. Add/delete individual images in the gallery, also tag collaborate contributors to a gallery.

11. Add travel information to users.

12. Mobile view of each page.

To Get Started

1. Click on sign up and create your profile in 3 quick steps.

2. Once your profile is created, you will be shown logged in to the account

3. Click on my profile page, to see your profile information.

4. If you want to edit your profile, you can click on the three main links available on the edit profile page. You can find two options, "Edit Personal Info" and "Edit Professional Info". You can edit your profile according to your preferences

5. Social Media handles are shown in the profile details page,

6. You can add projects and upload up to 5 photos to the project. You can view the average project rating of your project. If you are the creator of the project, you will be able to find a link to creating the project.

Home Page Features:

1.Home page will show the list of the projects based on current user's city, state, and country.

2. Buttons of the Home page are updated, and professionals can see the projects filtered by specific professions.

3. The professionals can see both the profile details and project details page link from the home page.

4. The users can search directly from home page now.

Search Features:

1. The user can search from the home page directly.

2. The user can search with keywords matching for profile's name, bio, compensation type, and highlights. And filtering by country, state, city, and professions.

3. The search results will be sorted by logic based on rating, gallery numbers, names and location.

4. The users can also find travel profiles during their travelling time.

5. If there are no results for a search, the recommending profiles based on user's location will be listed.

Rating Features:

1. The user can add star ratings from the profile page now in (1 to 5 scale).

2. To rate another professional, the user has to be signed in.

3. One can add only one rating to a project and edit their rating after that. A professional cannot rate their own project.

Forget Password Features:

1. The user can request a password reset in their registered email address.

2. A temporary link (validity: 15 minutes) has been generated for the resetting password

3. After resetting, they will be redirected to the sign in page again.

Job features:

1.The user can go to the pages to post jobs, edit jobs, and delete jobs.

2. There is a separate search engine in which users can search for matching jobs.

Tag feature:

1.User can tag users in project to show that they work together to do a project.

2. User can invite users to collaborate by sending email to others, and then the invitor can edit it directly from the link in project.

Travel feature:

1.User can add travel info in their profile.

2. Travelling users will show up in search results.

Gallery feature:

1.The user can add and delete gallery in their profile page.

2. The user can add/delete individual images in their gallery. (no more than 5)

3. The user can edit their text info in a gallery.