NXTFolio - Final Report

Team Members

Guang Hui Liew - Scrum Master

Piyush Sharan - Scrum Master

Quanqi Hu - Scrum Master

Vishnuvasan Raghuraman - Product Owner

Links

GitHub repo: https://github.com/FashioNXT/NXTFolio

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

Deploy: https://nxtfolio.herokuapp.com

Presentation: Presentation Link

Presentation and demo video: https://vimeo.com/890464198?share=copy

Summary

NXTFolio is a social software platform designed to address the specific needs of the creative industry, encompassing fields such as fashion and lifestyle. The website aims to connect creative professionals with complementary skill sets, allowing them to collaborate on projects seamlessly. The existing legacy project has the fundamental structure in place, including user interfaces, login functionality, profile creation, keyword search, and job posting capabilities. It also includes necessary user agreements and terms of service for users.

As the client requested, the development plan for Fall 2023 is to identify and address the issues in the existing codebase and meanwhile implement some of the new features he suggested. Eventually, we identified and addressed the following issues in the legacy codebase:

1. The job posting feature does not function properly on the Heroku application.
2. The mobile view feature was not merged into the master branch.
3. Cucumber tests did not function properly on local deployment.
4. The job search feature was incomplete. The core search algorithm is missing.
5. Most of the legacy tests, including Cucumber and Rspec tests, failed, and the test coverage was only 27%.

The stakeholder of features 1, 2, and 4 is the application user. For features 3 and 5, the stakeholder is the application developer.

Meanwhile, we implemented some new features that the client requested:

1. Collaborator comment access feature.
2. AI chatbot.

The stakeholder of the above two features is the application user.

User Stories

Feature: Job Posting, Mobile view (Incompletely Developed last semester)

As a Developer,
So that I can complete legacy features,
I want to complete legacy features which are not available on the master branch due to some issues and lacking areas which need to be fixed.

Number of points: 6
Implementation status: finished

Feature: Legacy Code Understanding and Automation Testing

As a Developer,
So that I can test features,
I want to set up automation and test framework so that I am successfully able to Test the cucumber tests, and specs and verify the code coverage.

Number of points: 6
Implementation status: finished

Feature: Job Posting

As a user
So that I can post a job
I want to be able to post jobs and manage the posted jobs in the job management system.

Number of points: 6
Implementation status: finished

Feature: Enabling Comments on Galleries

As a User,
So that I can comment on my posts,
I want to be able to comment on my posts, both created and collaborated. I also want my collaborators to be able to comment on my post.

Number of points: 6
Implementation status: finished

Feature: Job searching

As a professional,

I want to search for jobs based on keywords, profession, and location information,

So that relevant jobs show up and irrelevant ones do not.

Number of points: 6

Implementation status: finished

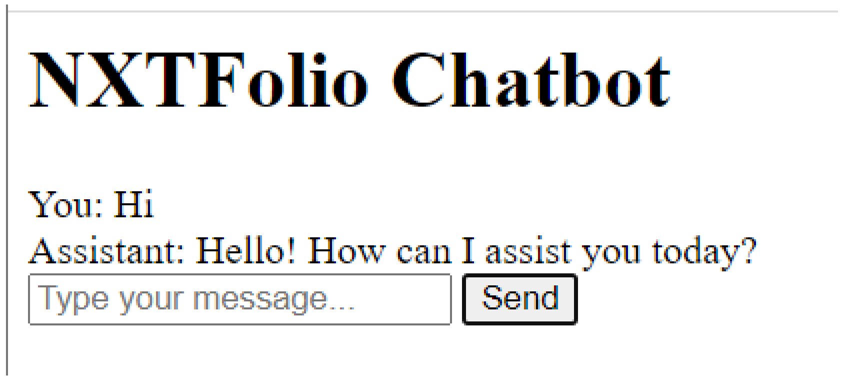
Feature: AI Chatbot

As a fashion content creator

I want to get suggestions on how to set up a nice portfolio profile for NXTFolio

Number of points: 6

Implementation status: finished



NXTFolio Chatbot

You: Hi

Assistant: Hello! How can I assist you today?

Feature: Collaborator Comment Access

As an Active and Engaging User,

So that I can get commenting access,

If I have collaborated with over 3 different users, then I get the comment access and I am able to comment on any gallery on the website.

Number of points: 6

Implementation status: finished

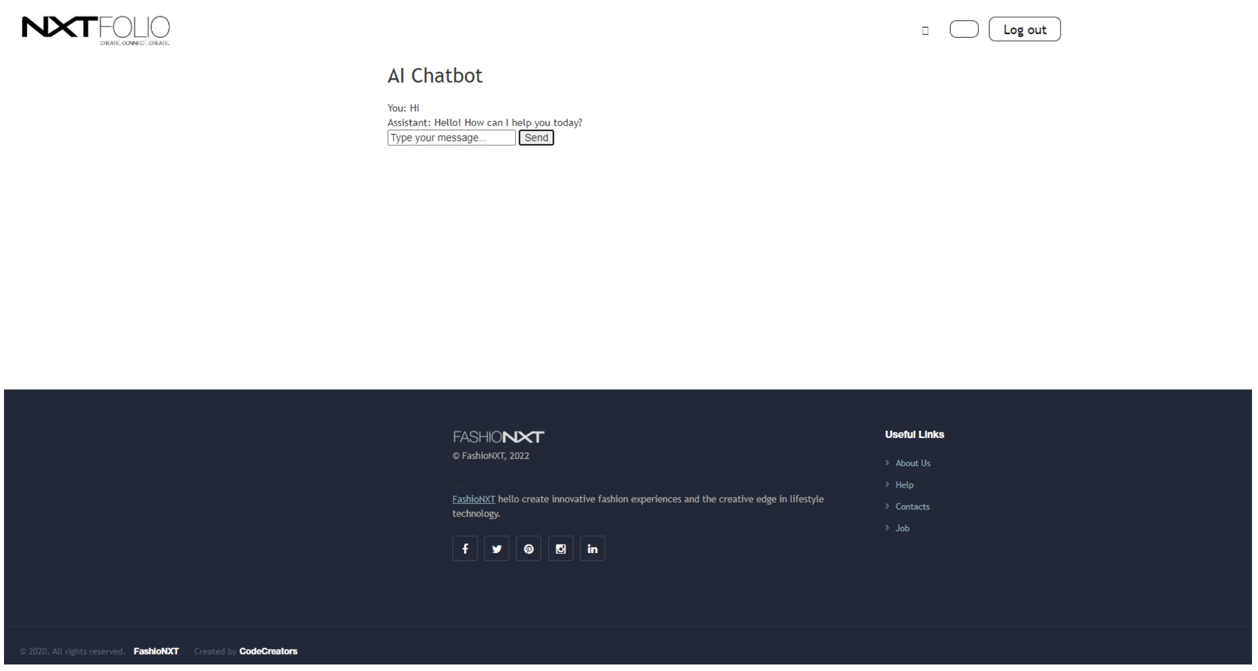
Feature: AI Chatbot Frontend Design

As a fashion content creator

I want to get suggestions on how to set up a nice portfolio profile for NXTFolio

Number of points: 6

Implementation status: finished



AI Chatbot

You: Hi

Assistant: Hello! How can I help you today?

Type your message...

© FashioNXT, 2022

FashioNXT hello create innovative fashion experiences and the creative edge in lifestyle technology.

© 2020. All rights reserved. FashioNXT Created by CodeCreators

Feature: Legacy Test Development and Cleanup

As a developer

I want to see a test coverage of more than 80% when I run Cucumber and Rspec tests.

Number of points: 12

Implementation status: finished

Modifications on Legacy Code

The existing legacy project has the fundamental structure in place, including user interfaces, login functionality, profile creation, keyword search, and job posting capabilities. The codebase follows a standard structure of Ruby on Rails application. The functionality-related files are located in ./app. The database seeding and migrating files are in. /db. The installation and environment configuration files are in ./install. The Cucumber tests and Rspec tests files are in ./features and ./spec respectively.

During the development in fall 2023, we made the following changes to the legacy code.

1. Added solutions to possible issues in executing Cucumber tests to the README file.
2. Fixed issues in existing features, including job posting and mobile view.
3. Completed features initiated by previous teams, including job search.
4. Implemented two new features, collaborator comment access and an AI chatbot.
5. Managed to increase the test coverage from 27% to 84% by adding new Cucumber and Rspec tests and cleaning up irrelevant code.

Team Roles

	Scrum Master	Product Owner
Iteration 0	Guang Hui Liew	Vishnuvasan Raghuraman
Iteration 1	Piyush Sharan	Vishnuvasan Raghuraman
Iteration 2	Guang Hui Liew	Vishnuvasan Raghuraman
Iteration 3	Quanqi Hu	Vishnuvasan Raghuraman
Iteration 4	Piyush Sharan	Vishnuvasan Raghuraman
Iteration 5	Quanqi Hu	Vishnuvasan Raghuraman

Team Member Effort Chart

Member ——------------- Iteration #	Guang Hui Liew	Piyush Sharan	Quanqi Hu	Vishnuvasan Raghuraman
0	0	0	0	0
1	3	3	3	3
2	3	3	3	3
3	3	3	3	3
4	3	3	3	3
5	3	3	3	3
Final week	2	2	2	2
Total	17	17	17	17

Iteration Summaries

Iteration 0: During the initial iteration, we focused on getting familiar with the legacy codebase and the existing functionalities of the deployed application. We ensured that the application could be deployed locally and on Heroku. We discovered that there were a couple of legacy features missing from the website, and there were lingering issues related to the Heroku deployment from the previous team's efforts. Subsequently, we have been actively attempting to establish communication with the previous team to follow up. At the same time, we had a discussion with the client about potential new features to implement this semester.

Iteration 1: During iteration 1, we finished two user stories. First of all, the job posting and mobile view features were partially developed by the previous team last semester. The job posting feature was merged with the master branch last semester but it is not working. The tests were failing and we were unable to create job listings manually. Similarly, the Mobile View has a lot of missing functionality in key areas. We started with adding cucumber tests. We located the issues of these features and aimed to address them in the following iterations. The other user story is to understand the legacy code and conduct testing on local deployment. After addressing several minor issues, the Cucumber tests on local deployment were still failing across nearly all scenarios, yielding a coverage of only 3.37%. Ultimately, we utilized the Github Action devised by the prior team, achieving a coverage of 27.32%, aligning with the previous team's reported results. We give 6 points to each user story and thus 12 points are completed in total.

Iteration 2: During iteration 2, we focused on fixing the job posting feature on the master branch, implementing the feature of enabling comments on galleries, and fixing the local testing issue. We managed to locate and fix the issues found previously in the job posting, and currently, the job posting function works as expected on the Heroku application. The feature of enabling comments on galleries is also completed. After a discussion with the previous team, we fixed the issue in local testing by doing docker-compose down the docker and docker-compose up again. We give 6 points to the gallery commenting feature, 3 points to job posting, and 3 points to local testing. Thus, 12 points are completed in total.

Iteration 3: During iteration 3, we focused on completing the job-searching feature and implementing a simple AI chatbot. The previous team has finished the front end of the job searching feature. However, the search algorithm was not implemented. The searching function always returns all existing jobs. We added a search algorithm, and now the job search function is complete. On the other hand, we started working on the AI chatbot feature. We built up a simple UI and used the API of ChatGPT to achieve the AI chat function. We give 6 points to each user story and thus 12 points are completed in total.

Iteration 4: During iteration 4, we focused on implementing the collaborator comment access feature, further improving the front-end design of the AI chatbot, and seeding fake jobs to the database for testing. Now the collaborator comment access feature is finished so that a user can get the comment access only if the user has collaborated with over 3 different users. The access button for AI chatbot is moved from the homepage to the profile page as the client suggested. Moreover, the UI for AI chatbot has been modified with footer and header to align with the general design of the website. To better test the job search feature implemented in the previous iteration, we seed 100 fake jobs to the database. We give 6 points to the collaborator comment access feature, 3 points to the front-end improvement of AI chatbot, and 3 points to the fake job seeding. Thus, 12 points are completed in total.

Iteration 5: As we came to an agreement with Professor and TA during the last iteration, no new feature development will be progressed and our team's full focus will be on legacy code's coverage. Significant progress has been made in terms of test coverage, and we have successfully increased the coverage from 27% to 84% across more than 2000 lines of code. Several bugs were also addressed in this development cycle. Notable fixes include resolving issues related to "Make User Admin" and "Searching" features. These bug fixes contribute to the overall stability and functionality of the system. We give 12 points to this testing coverage improvement.

Customer Meetings

9/08/2023

During this initial meeting with the client, we had a discussion about the current status of this project and some potential plans for this semester. We discovered that there were a couple of legacy features missing from the website, and there were lingering issues related to the Heroku deployment from the previous team's efforts. Subsequently, we have been actively attempting to establish communication with the previous team to follow up and obtain guidance and instructions from them. Moreover, the client introduced several new features that he expected us to implement, including

1. completing the job posting and the mobile view feature (partially developed last semester);
2. AI based ChatBot;
3. reverse Image Search;
4. collaborator only comment;
5. CRM Statistics (Customer Relation Management).

9/22/2023

We engaged in a comprehensive discussion with the client concerning our development priorities for the upcoming weeks. These priorities encompass the implementation of new features, including the integration of an AI Chatbot for user inquiries, the introduction of Collaborator-only comments to enhance comment quality, and the incorporation of a Reverse Image search feature to extract keywords and display relevant posts. During the meeting, we gained a deeper understanding of these features, which will significantly contribute to our project's progress. Additionally, we addressed critical issues related to legacy features and tests. Our Heroku deployment revealed data inconsistencies from the previous semester, prompting us to promptly notify the Spring '23 team via email. Furthermore, our legacy tests, utilizing Cucumber, were encountering challenges, with 76 out of 78 scenarios failing. In consultation with the client, we collectively decided to prioritize resolving these issues to establish a robust foundation for our upcoming development efforts.

10/12/2023

During this week’s customer meeting, several crucial points were discussed to enhance NXTFolio. Emphasizing the need for data preservation, it was highlighted that data losses from previous semesters must be prevented, prompting a focus on robust data-saving mechanisms moving forward. With the job posting feature issue resolved, customer suggested further improvement of the search algorithm, a recommendation that will be integrated into ongoing development efforts. A strategic decision was made to postpone the development of the mobile view feature, enabling the incorporation of new features for a more comprehensive user experience upon its eventual implementation. Key features currently in active development include collaborator-only comments and an AI chatbot. Lastly, the client provided design references, suggesting Behance.net and Creatively.life for the platform's aesthetics and Upwork.com for job posting and smart contract functionalities. Overall, the project is progressing with a strong emphasis on data security, user experience enhancement, and the integration of innovative features, aligning the platform with industry standards and client expectations for the creation of NXTFolio, a cutting-edge fashion portfolio website.

10/27/2023

During the client meeting, various key points were addressed. We initiated the session with a demonstration of the recently fixed legacy Job Search functionality, which the client found to be satisfied with. Suggestions were made to improve the UI, along with a request to incorporate a "travel" parameter in the search algorithm to align with requirements. We also presented the collaborator-only comment feature, which garnered feedback and change requests. The client requested modifications to the way this feature operates. Additionally, we showcased the Chatbot feature and received input on its placement on the website and how the target audience can make use of it. Finally, there was a mutual agreement to enhance the application's UI/UX, aligning it with the standards set by popular websites like Indeed and Upwork. We assured the client that our primary focus would be on developing fully functional features with an appropriate UI, with plans to address the aforementioned UX aspect at a later point in time.

11/10/2023

A successful demonstration of the collaborator-only comment feature was presented to the client, who expressed satisfaction with the implementation. Additionally, a demonstration of the AI chatbot feature, incorporating minor modifications from the previous meeting, was well-received. Following Professor Ritchey's recommendations to enhance legacy test case coverage, a consensus was reached with the client to prioritize test writing and coverage improvement in the upcoming iteration. The client also suggested addressing UI bugs after the coverage has been enhanced.

11/22/2023

Unfortunately, the client did not attend this week's arranged meeting. We have reached out to the client via email and presented a comprehensive assessment, covering the current status of test coverage and feature implementation. As we came to an agreement with Professor and TA during the last iteration, no new feature development will be progressed and our team's full focus will be on legacy code's coverage. Significant progress has been made in terms of test coverage, and we have successfully increased the coverage from 27% to 84% across more than 2000 lines of code. Several bugs were also addressed in this development cycle. Notable fixes include resolving issues related to "Make User Admin" and "Searching" features. These bug fixes contribute to the overall stability and functionality of the system.

BDD/TDD Process:

In our development process for implementing new features this semester, we adhered to a structured Behavior-Driven Development (BDD) and Test-Driven Development (TDD) approach. For this, we used Cucumber and Rspec tests, ensuring the reliability and functionality of our Ruby on Rails web application.

Steps Followed:

Test Creation:

- We initiated the process by creating a comprehensive suite of tests using Cucumber and Rspec. These tests were meticulously designed to cover diverse scenarios based on the specified feature requirements.

Code Implementation:

- Following the test creation, our focus shifted to writing the minimum needed code necessary to fulfill the above written scenarios. This step-by-step implementation allowed us to iteratively build and refine the functionality of the new features.

<u>Client Requirement Modifications:</u>

- Modifications in client requirements prompted updates in Cucumber and Rspec tests before modifying the code. This sequential approach minimized risks and guarded against regressions.

Benefits:

<u>Robust Test Coverage:</u>

- BDD/TDD ensured a comprehensive test suite, enhancing application reliability and facilitating maintenance.

<u>Early Issue Detection:</u>

- TDD facilitated early issue identification, preventing complexities in later stages.

<u>Client Collaboration:</u>

- Transparent BDD practices, with clear Cucumber scenarios, encouraged client participation in validation.

Challenges and Adaptations:

While effective, rigid adherence to initial scenarios may pose challenges in dynamic projects. We had to ensure we had an agile mindset for adapting to evolving requirements without compromising code integrity. This was key in us delivering a successful project.

Configuration Management

Establishing a clear branching strategy is important to manage different stages of development, such as feature development, bug fixes, and releases. Our configuration management approach includes making separate branches for each feature and making pushing commits as we make progress along the way, and finally having a thorough code review by the team members before merging each branch to master for deployment. We also make sure to double-check check there is no issue with the deployed app. We did not need to do any spikes. In total, we had about 18 branches and 18 merge releases.

Issues on Heroku Deployment

Initially, our attempts to deploy on individual Heroku accounts encountered issues related to AWS and Google Geocoding API credentials. Recognizing the need for a more efficient solution, we proposed obtaining direct access to these credentials in a meeting with the client.

However, the client, foreseeing the importance of facilitating future teams, strongly recommended utilizing their established Heroku account. This account had the credentials preconfigured by a previous team, ensuring a smoother deployment process.

Subsequently, we've consistently deployed the website on the client's dedicated Heroku account. This decision not only addressed the initial credential-related challenges but also aligned with the client's desire to provide easy access for future teams. To streamline future deployments, we advise contacting the client for credentials or reaching out to our team for guidance, ensuring a seamless continuation of the production release process.

Issues on Other Tools:

AWS and GeoCoding API Credentials:

During the web app deployment, we consistently faced challenges related to missing AWS and GeoCoding API credentials. This issue was effectively resolved by transitioning to the client's Heroku account, where these credentials were already configured. This adjustment successfully resolved the deployment issues.

Cucumber Tests and Capybara Timeout Error:

Another noteworthy challenge emerged during the execution of Cucumber tests, manifesting as a persistent Capybara timeout error. To address this, we implemented a three-step solution:

- Execute "docker-compose down" to stop running containers.
- Delete existing Docker images.
- Re-run the setup steps.

This approach successfully mitigated the Capybara timeout error, ensuring the reliable execution of Cucumber tests.

For any lingering issues or further assistance, we encourage future teams to reach out to our team.

Other Tools/Gems

ruby-openai gem: The ‘ruby-openai’ gem is a Ruby client library for integrating with OpenAI’s API, which includes services like the GPT models, including ChatGPT. This gem simplifies the process of connecting to and utilizing OpenAI’s powerful AI models from Ruby applications, and we utilize it for our AI chatbot feature.

SimpleCov: Is a code coverage analysis tool for Ruby. It’s a widely used gem in the Ruby community, offering the ability to track how much of our Ruby codebase is covered by tests. In this semester, we successfully increased the test coverage from 27% to 84%.

Deployment Instructions (Also available on GitHub)

Setup and Testing
Go to `config/environments/test.rb` and set `config.use_remote_webdriver` to true if using docker-compose, and to false otherwise.
Recommended: Using docker-compose (Ubuntu 18.04 and selenium firefox)
This is the fastest way to get started.
It uses docker-compose to set up two containers
- ruby: container with the app and in which you will be developing
- browser: container with selenium web driver needed for testing with cucumber

The files to setup docker-compose can be found in the `install` folder.

0. If you are using M1 chip, change in `docker-compose.yml` the `browser` image to

   `image: seleniarm/standalone-firefox`
1. Clone the repository
   ```bash
   git clone https://github.com/vibalcam/match-my-fashion-public-CodeCreators --config core.autocrlf=input
   ```
2. Go into the `install` folder
   ```bash
   cd install
   ```
3. Build and start the containers for docker-compose (-d used to do start in detach mode)
   ```bash
   docker-compose up -d
   ```
   The project folder has been binded to `/home/match-my-fashion-public-CodeCreators`.
   In other words, it is shared by the container and host so it you can continue developing in your host machine.
4. Run a bash terminal on the ruby container
   ```bash
   docker-compose exec ruby bash
   ```
5. Copy the setup script and cd to home
   ```bash
   cd /home
   cp match-my-fashion-public-CodeCreators/install/setup.sh .
   sudo chmod +x setup.sh
   ```
6. Run `setup.sh` script and follow the instructions (more details about what it does can be found in `install/setup.sh`)
   ```bash
   source setup.sh
   ```
   If you run into trouble, try looking the script since the comments might help.
7. You should be set. Whenever you want to run the server just run
   ```ruby
   rails server -b 0.0.0.0 -p 3000
   ```

8. To connect to the website go to:

http://localhost:8080
When you are done developing, you can stop the docker-compose
```bash
cd install
docker-compose stop
```
The next time you want to connect you just have to run the following:
```bash
cd install
docker-compose start
```
or start the docker-compose from the desktop app.
Running Tests
In the root directory, run "rails cucumber"
When running cucumber tests (`rails cucumber`), you can access the following website to see what the tests are doing
http://localhost:7900/?autoconnect=1&resize=scale&password=secret
This combined with the cucumber step `When I debug` is a powerful tool to debug the cucumber tests.

Possible issues ->
1. Capybara Timeout issue :
Go to /install ->
docker-compose down ->
Delete docker container images ->
docker-compose up -d ->
run setup steps from step 4 above.

Using Dockerfile (Ubuntu 18.04)
This is the fastest way to get started.
The Dockerfile and script to automatically setup app can be found in the `install` folder.
1. Clone the repository
git clone https://github.com/vibalcam/match-my-fashion-public-CodeCreators --config core.autocrlf=input
2. Build the Dockerfile using
docker build -t ruby-ssh install/
3. Run docker container
If using Windows:
docker run -d -p 8080:3000 --mount type=bind,src="%cd%",target=/home/match-my-fashion-public-CodeCreators --name test_container ruby-ssh
If using Linux:
docker run -d -p 8080:3000 --mount type=bind,src="$(pwd)",target=/home/match-my-fashion-public-CodeCreators --name test_container ruby-ssh
We are mapping port 3000 (used by rails server) to 8080 in the host.

We are also binding the project folder to `/home/match-my-fashion-public-CodeCreators` so it is shared by the container and host.
4. Connect to terminal in container
docker exec -it test_container bash
5. Copy the setup script and cd to home
cd /home
cp match-my-fashion-public-CodeCreators/install/setup.sh setup.sh
sudo chmod +x setup.sh
6. Run `setup.sh` script and follow the instructions (more details about what it does can be found in `install/setup.sh`)
source setup.sh

If you run into trouble, try looking the script since the comments might help
7. You should be set. Whenever you want to run the server just run
    rails server -b 0.0.0.0 -p 3000

9. To connect to the website go to:
    http://localhost:8080
Run on Linux without docker (tested on AWS E2C, Ubuntu 18.04)
1. Copy `install/setup.sh` to folder where you want to download the app
2. Give permission to run the script
    sudo chmod +x setup.sh
3. Run `setup.sh` script (more details about what it does can be found in `install/setup.sh`)
    If you have already cloned the repo:
    ./setup.sh
    If you have just copied the script:
    ./setup.sh https://github.com/vibalcam/match-my-fashion-public-CodeCreators
    If you run into trouble, try looking the script since the comments might help
4. You should be set. Whenever you want to run the server just run
    rails server -b 0.0.0.0 -p 3000
5. To connect to the website go to:
    http://localhost:8080

How to Deploy
The website is deployed using AWS credentials and additional buckets. Client has requested to keep everything deployed on Client's Heroku.
Contact the previous team for Credentials.

Setup and Testing

Go to `config/environments/test.rb` and set `config.use_remote_webdriver` to true if using docker-compose, and to false otherwise.

Recommended: Using docker-compose (Ubuntu 18.04 and selenium firefox)

This is the fastest way to get started. It uses docker-compose to set up two containers

- ruby: container with the app and in which you will be developing
- browser: container with selenium web driver needed for testing with cucumber

The files to setup docker-compose can be found in the `install` folder.

1. If you are using M1 chip, change in `docker-compose.yml` the `browser` image to
   ```
   image: seleniarm/standalone-firefox
   ```
2. Clone the repository
   ```
   git clone
   https://github.com/vibalcam/match-my-fashion-public-CodeCreators --config
   core.autocrlf=input
   ```
3. Go into the `install` folder
   ```
   cd install
   ```

4. Build and start the containers for docker-compose (-d used to do start in detach mode)
   `docker-compose up -d`
5. The project folder has been binded to `/home/match-my-fashion-public-CodeCreators`. In other words, it is shared by the container and host so it you can continue developing in your host machine.
6. Run a bash terminal on the ruby container
   `docker-compose exec ruby bash`

Copy the setup script and cd to home

```
cd /home
cp match-my-fashion-public-CodeCreators/install/setup.sh .
```

7. `sudo chmod +x setup.sh`
8. Run `setup.sh` script and follow the instructions (more details about what it does can be found in `install/setup.sh`)
   `source setup.sh`
9. If you run into trouble, try looking the script since the comments might help.
10. You should be set. Whenever you want to run the server just run
    `rails server -b 0.0.0.0 -p 3000`
11. To connect to the website go to:
    http://localhost:8080

When you are done developing, you can stop the docker-compose

```
cd install
docker-compose stop
```

The next time you want to connect you just have to run the following:

```
cd install
docker-compose start
```

or start the docker-compose from the desktop app.

Running Tests

In the root directory, run "rails cucumber" When running cucumber tests (`rails cucumber`), you can access the following website to see what the tests are doing

http://localhost:7900/?autoconnect=1&resize=scale&password=secret

This combined with the cucumber step `When I debug` is a powerful tool to debug the cucumber tests.

Possible issues ->

1. Capybara Timeout issue :

   -> Go to /install

   -> docker-compose down

   -> delete docker container images

   -> docker-compose up -d

   -> run setup steps from step 4 above.

Helpful fixes

- Changes to css or javascript not reflecting on test
  Try recompiling the assets by running `rake assets:clobber` and `rake assets:precompile`.
- env: ruby\r: No such file or directory
  Windows uses a different line ending to linux, which is makes linux not read correctly windows files. Use `git clone url --config core.autocrlf=input` or `dos2unix` linux package to fix.

error: Homebrew on Linux is not supported on ARM processors
Problem with M1 chip macOS - brew not available for arm64 processor. Use platform option in docker build and run commands.

```
  docker build --platform linux/amd64 -t ruby-ssh install/
  docker run -d -p 8080:3000 --platform linux/amd64 --mount
type=bind,src="$(pwd)",target=/home/match-my-fashion-public-CodeCreators --name
nxtfolio ruby-ssh
```

error: brew not found
Make sure to run the steps which are displayed in console after installing brew.

```
  (echo; echo 'eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"') >>
/root/.profile
  eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"
```

Error: Is the server running on host "127.0.0.1" and accepting TCP/IP connections on port 5432?
First go to the setup script and try to run all the postgres related commands

```
  sudo apt install -y postgresql postgresql-contrib libpq-dev -y
```

If an error occurs involving the "dpkg is locked" then there are other processes trying to use apt. You need to kill these processes
To get a list of processes using apt run the follow command

```
  ps aux | grep -i apt
```

Hopefully it will show that some other processes are trying to use apt, kill these processes by running

```
  sudo kill <process_id>
```

- Once the processes are killed try installing postgres again using the lines specified in the setup script.

How to Deploy

The website is deployed using AWS credentials and additional buckets. The client has requested to keep everything deployed on the Client's Heroku. Contact the previous team for Credentials.