Dog Dates Release Summary

Team members

Name and email	GitHub id	Role of each member and tasks done by each member (brief description)
Sammul To duy5@myumanitoba.ca	sammulto	Role: Developer Tasks: Frontend and Backend development, API design and documentation, DB, dockerlize, code analysis, application deployment
Shawn Lanting umlantin@myumanitoba.ca	Aiphox	Role: Developer Tasks: Frontend development, Acceptance Testing, Documentation, Presentations
Yelizaveta Yashin yashiny@myumanitoba.ca	LizaYa	Role: Developer Tasks: Frontend and Backend development, and documentation.
Zhijie Zheng zhengz1@myumanitoba.ca	ZhijieZheng-UM	Role: Developer Tasks: CI/CD pipeline, Unit test, Integration test, Load testing, Bug reporting, API documentation, backend development

Project summary

This project is similar to a dating app but is for making playdates for dogs. Our target users are dog owners who want to find playdates for their dogs. A user can create an account, add information and pictures and then view other users' accounts and either ‘like’ or ‘dislike’ them. When two users like each other they have ‘matched’ and they are given new permissions such as the ability to see each other’s email address so that they can communicate with one another.

Our final project covered most of what we laid out in our proposal. All the core features were implemented but we did not get the chance to add some of the additional features such as being able to upload video and social media links to an account as well as receiving a notification when a user matched with you.

GitHub repository Link

https://github.com/DogDatesComp4350/DogDates

DockerHub repository

DockerHub link:

https://hub.docker.com/r/sammulto/dogdates

Instructions to run docker images:

1. Pull frontend and backend images from docker hub:
   - docker pull sammulto/dogdates:backend-git
   - docker pull sammulto/dogdates:frontend-git
2. Open a new terminal, run the frontend docker image:
   - docker run -it --rm -v ${PWD}:/app -v /app/node_modules -p 3000:3000 sammulto/dogdates:frontend-git
3. Open a new terminal, run the backend docker image:
   - docker run -it --rm -v ${PWD}:/app -v /app/node_modules -p 5000:5000 sammulto/dogdates:backend-git
4. In the browser, use the following URL to access to the application:
   - localhost:3000

List of user stories for each sprint

Note: Sprint 1 was focusing on management and creating the user stories for the project. Sprint 4 was focusing on tests, presentation, security, and project summary.

Sprint 2

US #1: Be able to create an account [Status: Done]

US #2: Be able to login an account [Status: Done]

US #3: Be able to update account [Status: Done]

Sprint 3

US #1: Be able to see other dog owners and their dogs [Status: Done]

US #2: Be able to like or dislike other users' profiles [Status: Done]

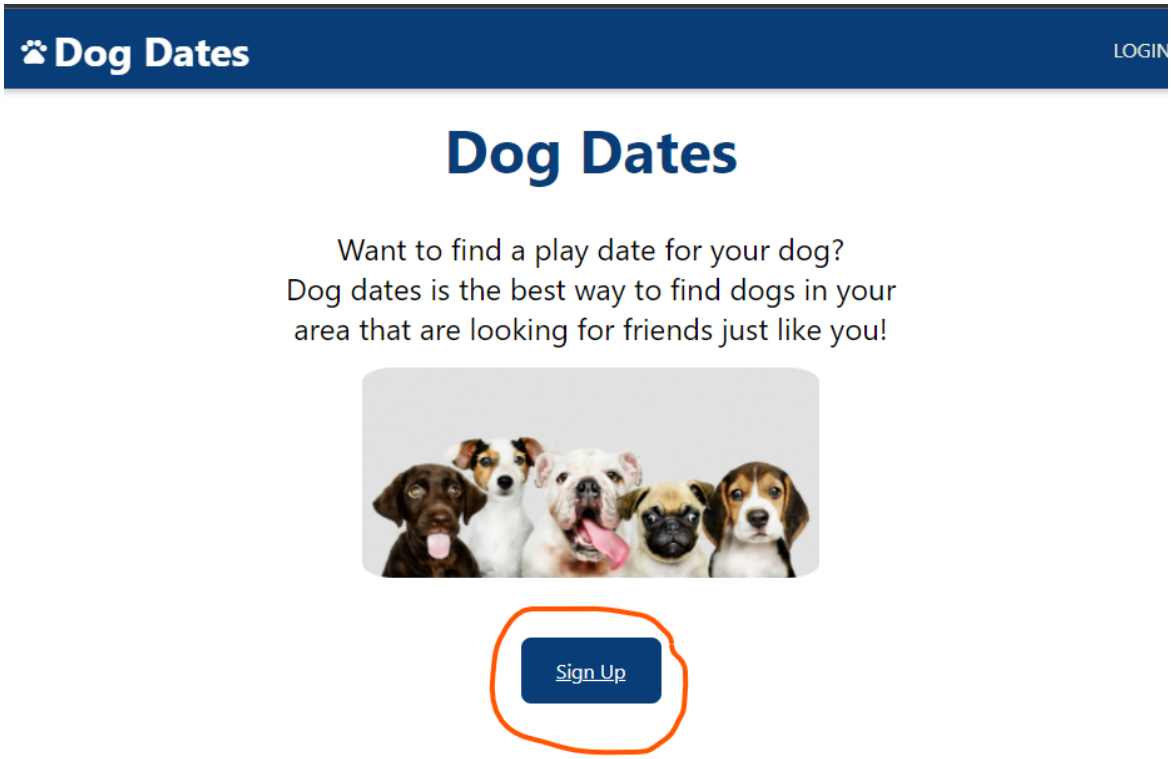
US #3: Be able to match with other dog owners [Status: Done]

US #4: Be able to see who I've matched with [Status: Done]

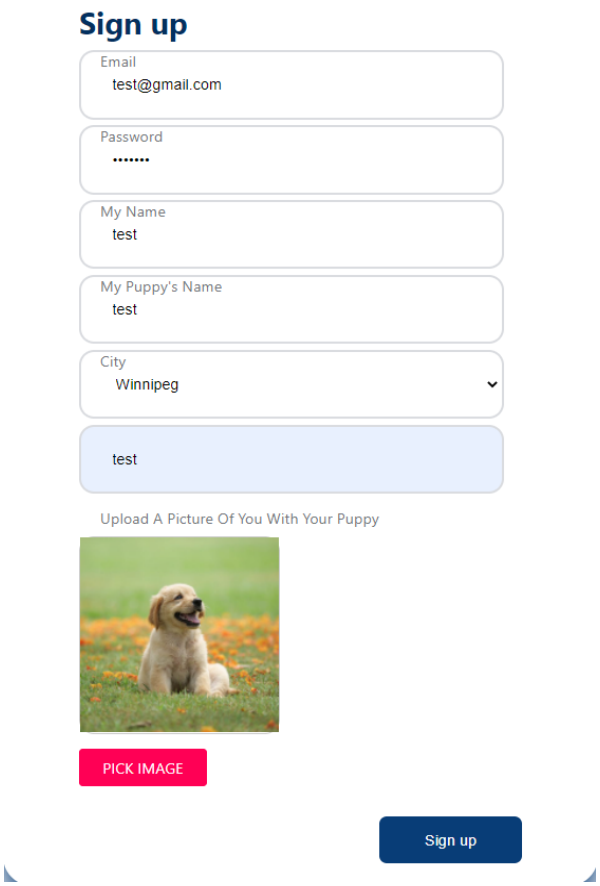
User manual

Create an account:

- Click on “Sign up” in the home page:

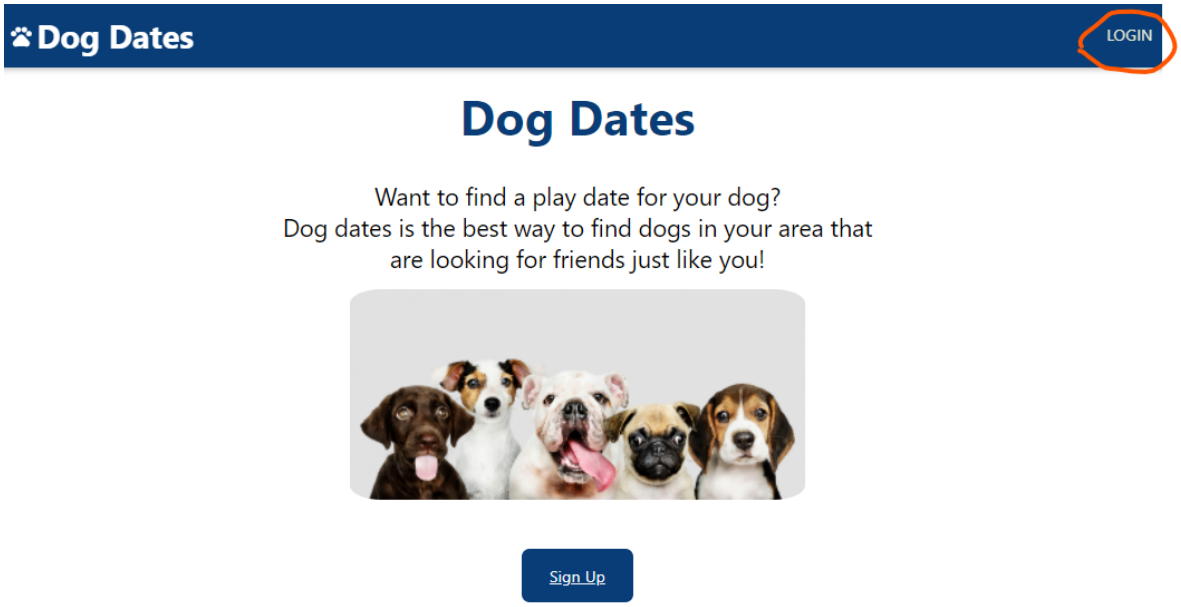


- Filling the account information (all fields are required):
    1) Fill in your email address. The email address must be a unique username to login later. Make sure the email address is valid.
    2) Fill your desired password. Remember your password as you will need it to login.
    3) Fill your name and your puppy’s name.
    4) Choose the city you are located in (options: Winnipeg or Toronto).
    5) Fille a description about your and/or your puppy.
    6) Click on the “Pick Image” button and upload an image from your local machine.
    7) Click on the “Sign up” button on the bottom right side (see screenshot on next page).

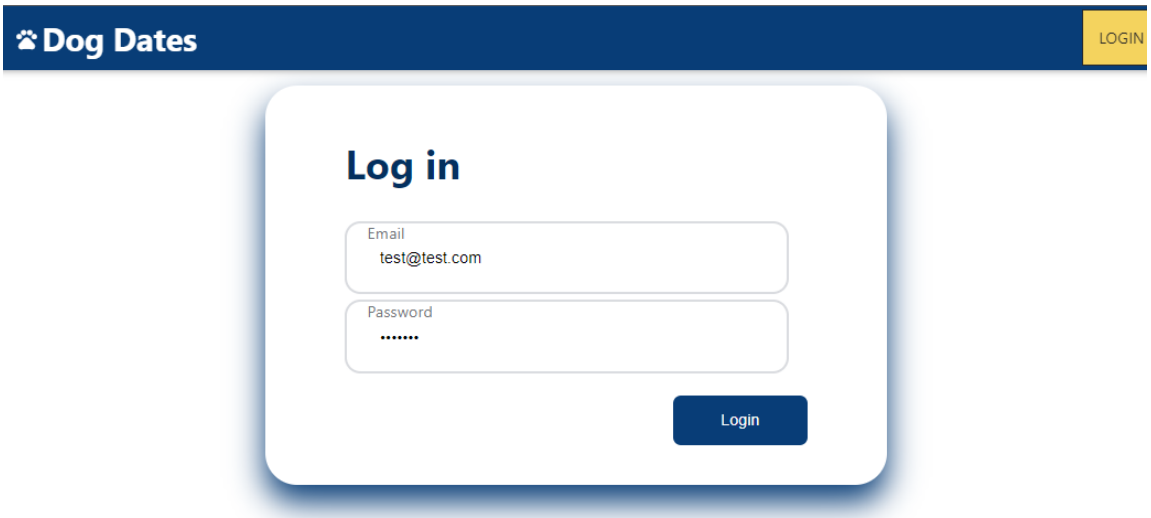


Login to account:

- Click on the “Login” button in the home page on the right top side:

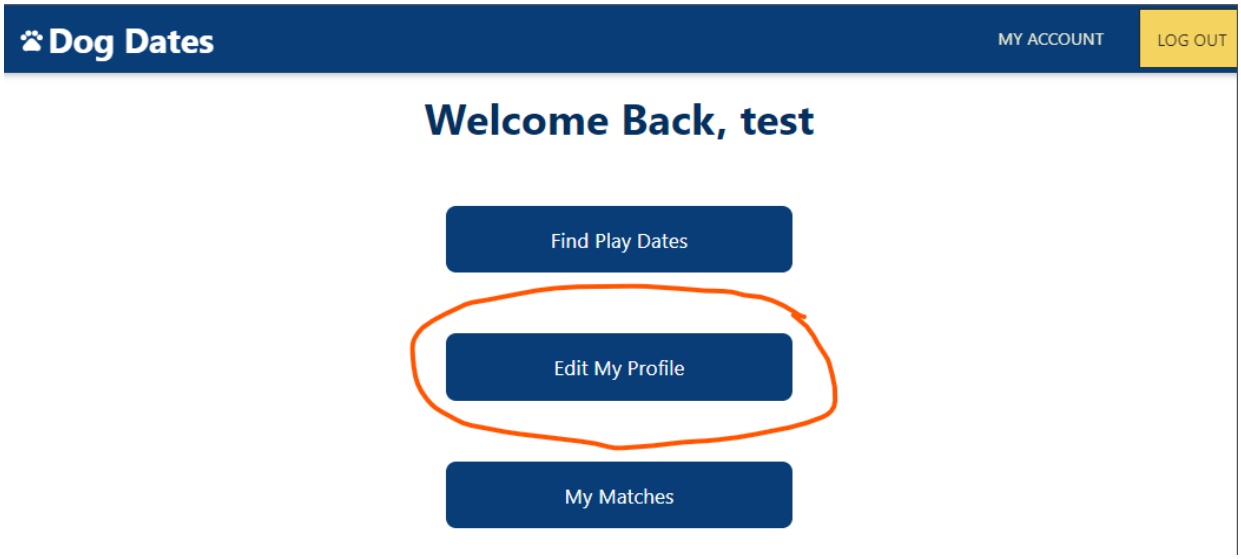


- Fill you email address and password you used in the account creation process and click on the “Login” button:

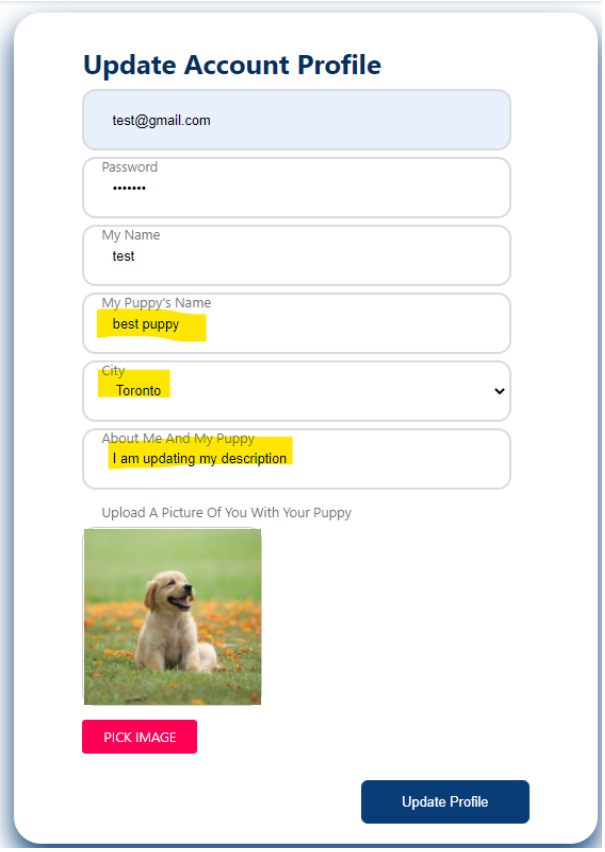


Update account details:

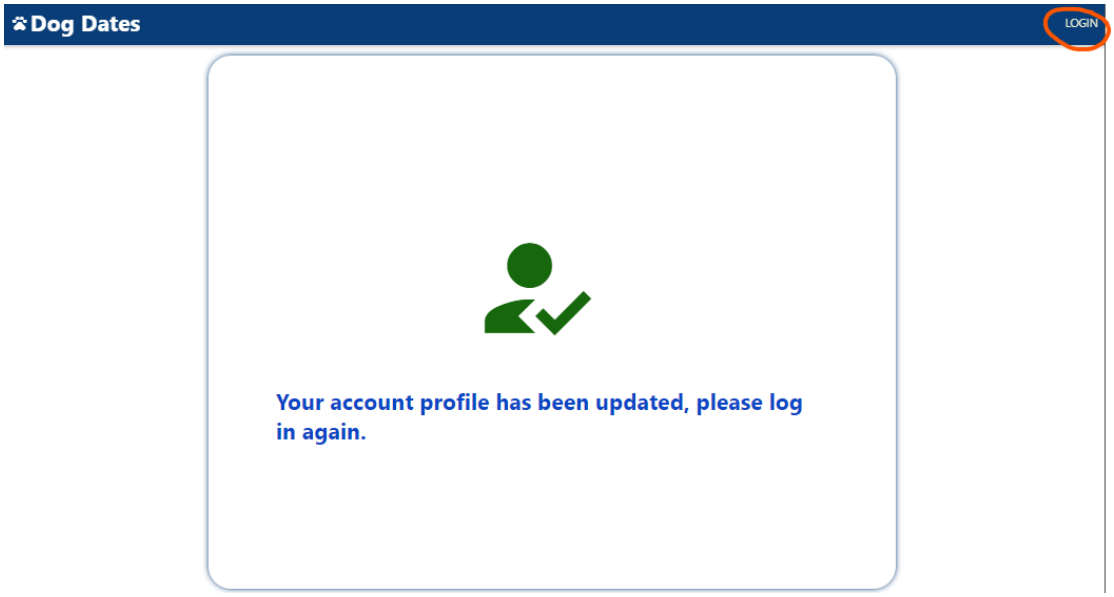
- After you logged in your account, click on the “Edit My Profile” button:



- Fill out your profile information and change the fields you wish to change. In the example below, the fields that will be changed are highlighted. Click “Update profile”:

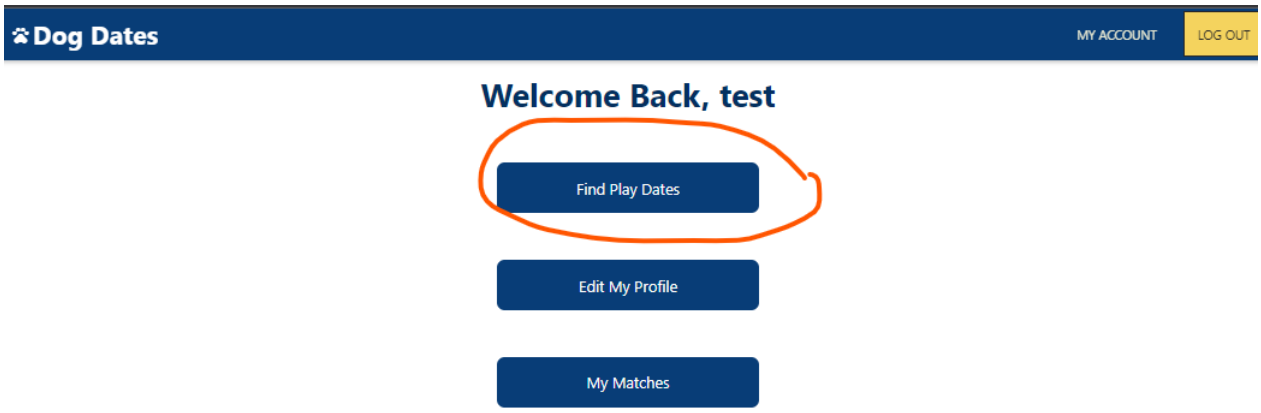


- After you click on “Update Profile”, you will be automatically logged out and will be prompted to login back again:

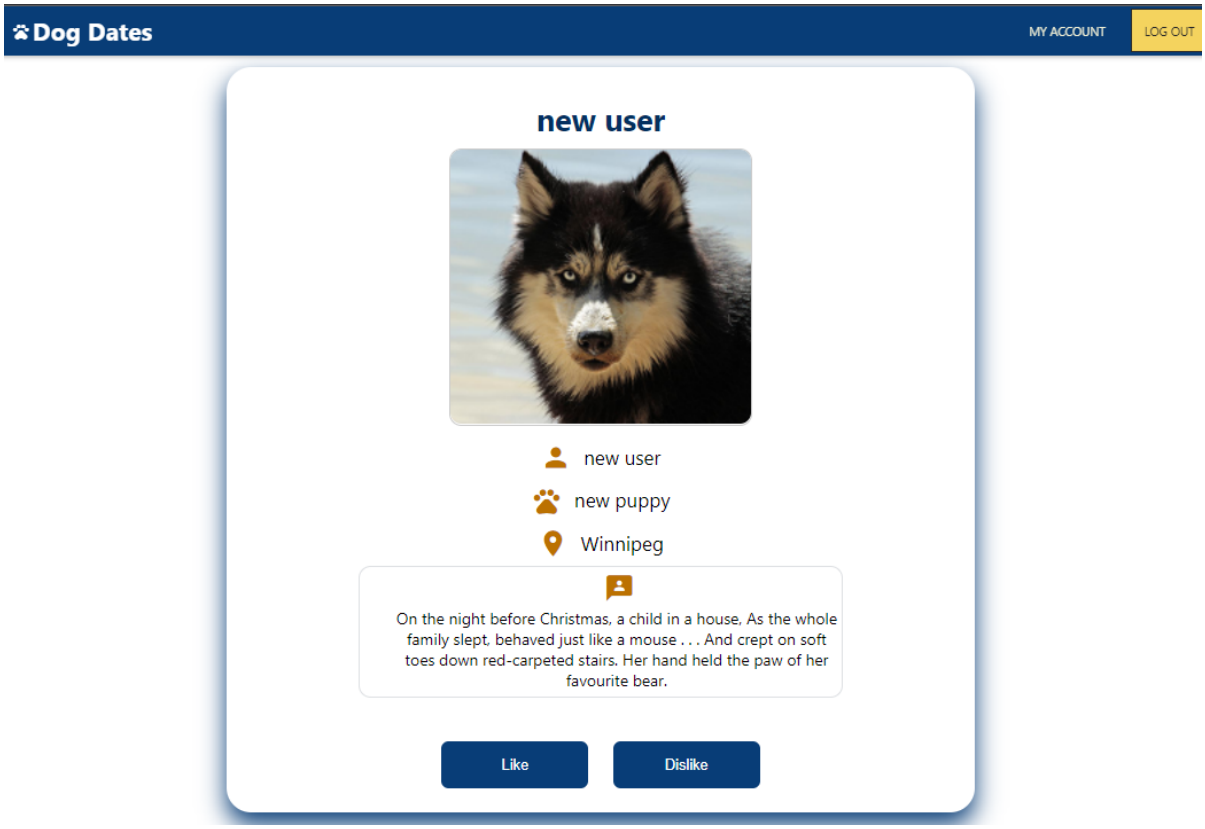


See other dog owners and their dogs

- After you logged in, click on “Find Playdates”:

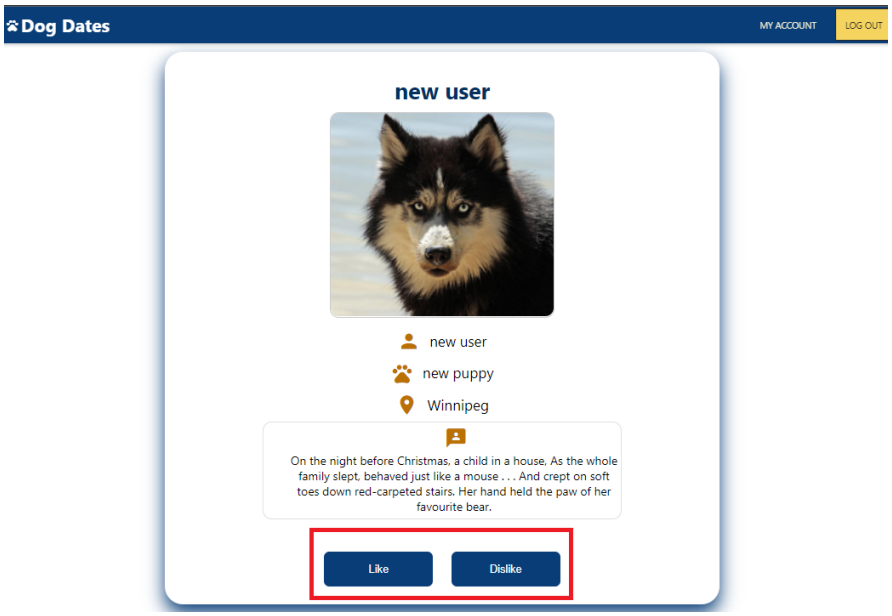


- Here you will be able to see a list of other users and their dogs from your city



Like/dislike other dogs

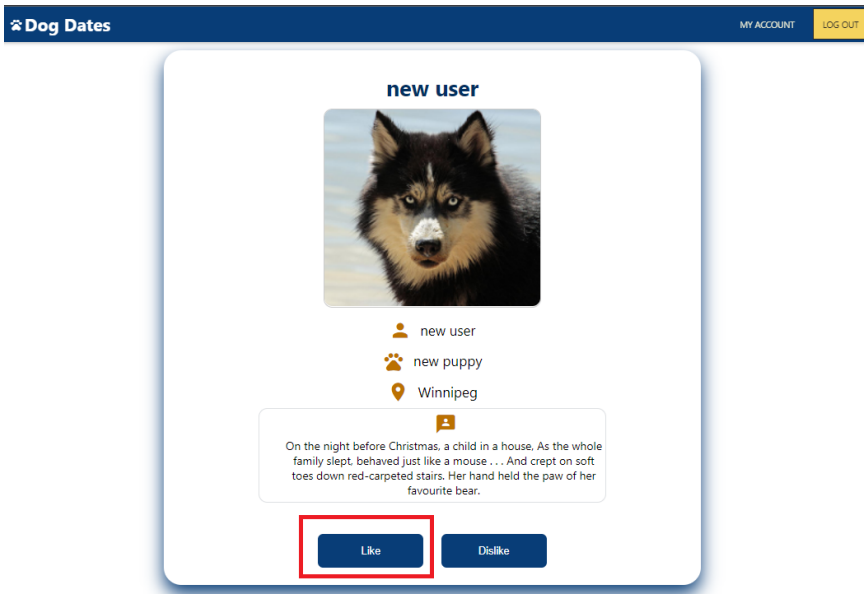
- After you clicked on “Find Play Dates”, if you see other dogs, you can click either the “Like” or “Dislike” buttons:



- To see other potential dogs, you must click like/dislike on the curent dog.

Match with other dogs

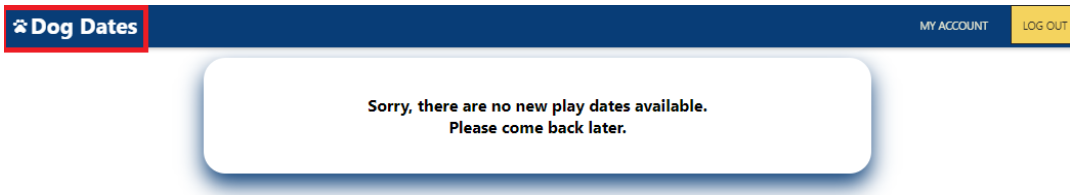
- Suppose you like the current dog. For the purpose if demonstration, assume that we clicked “Like” on this dog:



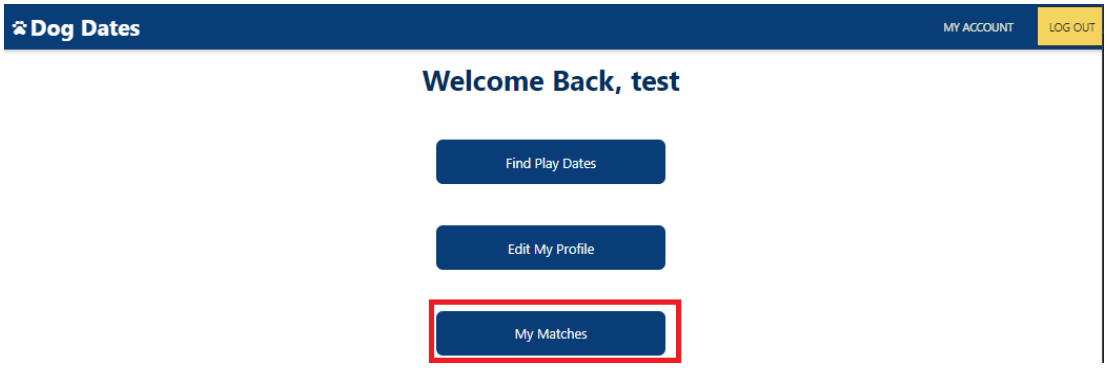
- Suppose “New User” also liked our profile. This means that we should have matched with “New User”.

View matches

- After you liked/ disliked all potential dogs, navigate to home page:



- Click on “My Matches”:



- Finally, you should see the dogs that you have matched with (you liked them and they liked you back). For all the matched dogs, you will be given their email address as a way to communicate:

Overall Arch and Design

Links to the overall arch, class diagram:

DogDates/System Architecture.PNG at main · DogDatesComp4350/DogDates (github.com)

DogDates/Class Diagram S3.PNG at main · DogDatesComp4350/DogDates (github.com)

Infrastructure

React https://reactjs.org/
React is a Javascript library used for building UI components. It has become very popular in the last handful of years and is used by large companies such as Facebook, Netflix, Uber, and AirBnB. It has a large community of support and many tutorial videos on sites like YouTube and Udemy. We chose it instead of other frameworks such as Angular and Vue not only for it's power and flexibility, such as the ability to reuse components, but also because none of us had ever used it before and we wanted to take the opportunity of this course to learn a new language that could open doors for us in our future's as software developers.

NodeJS https://nodejs.org/en/about/
NodeJS is a free, open-source server environment. We chose it because it runs on Javascript and passes JSON objects asynchronously between the front and back end. It has built in utilities that make reading and writing JS object fast and easy. Since it is asynchronous it is non-blocking and event driven. It is the default back end to use with React and has a lot of online support. It wouldn't have made much sense to use anything else.

Express https://expressjs.com/
Express is used to simplify node's API and also add additional features. It made it easier to organize our application's functionality with middleware and routing. It also added helpful utilities to Node.js's HTTP objects and facilitated the rendering of dynamic HTTP objects.

MongoDB https://www.mongodb.com/
MongoDB is a noSQL database used for scaling up large amounts of unstructured data. I chose it because we did not have a complicated schema but instead needed a DB that could theoretically handle a large amount of users. It also provided the added benefit of not needing to implement SQL-injection validation on our forms.

Name Conventions

https://github.com/DogDatesComp4350/DogDates/blob/main/CodeStyle.md

Code

Top 5 most important files (full path):

File path with a clickable GitHub link	Purpose
https://github.com/DogDatesComp4350/DogDates/blob/main/backend/controllers/signup-controller.js	Handle the user sign up event
https://github.com/DogDatesComp4350/DogDates/blob/main/backend/controllers/match-controller.js	Handle the user matching event
https://github.com/DogDatesComp4350/DogDates/blob/main/backend/controllers/users-controller.js	Handle view user account, update user account, delete user account events
https://github.com/DogDatesComp4350/DogDates/blob/main/backend/controllers/auth-controller.js	Handle user login event
https://github.com/DogDatesComp4350/DogDates/blob/main/backend/persistence/db-schema.js	Database Schema

Continuous Integration and deployment (CI/CD)

Workflow:

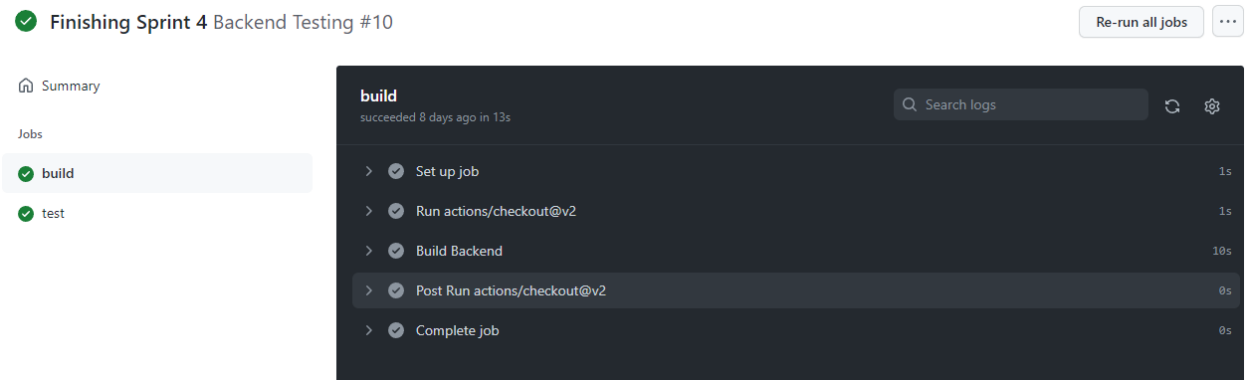
https://github.com/DogDatesComp4350/DogDates/actions

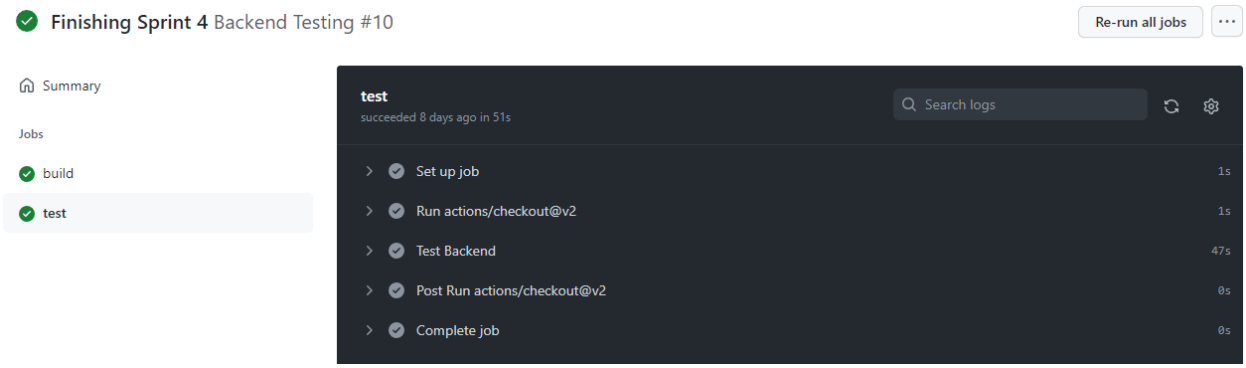
https://github.com/DogDatesComp4350/DogDates/tree/main/.github/workflows

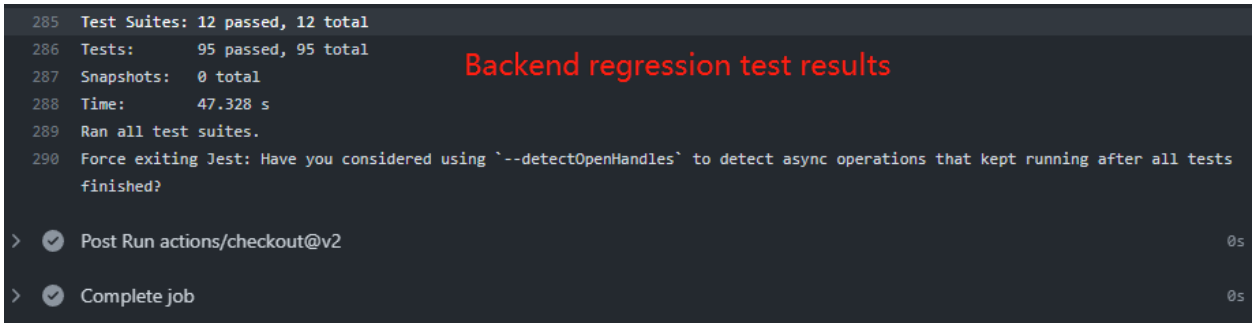
We used Github Action for CI/CD. With:

- Frontend regression testing (FrontendTest.yml)
- Backend regression testing (BackendTest.yml)
- Dockerlize Frontend and Backend and push to Dockerhub (BuildDockerImage.yml)
- CodeQL Security analysis (SecurityAnalysis.yml)

CI Snapshot (Backend regression testing):







CD Snapshot (Dockerize):

Finishing Sprint 4 Build And Push Docker Images #28

Re-run all jobs

Summary

Jobs

Build-And-Push-Docker-Images

Build-And-Push-Docker-Images

succeeded 8 days ago in 1m 43s

Search logs

Set up job	3s
Build SpicyPizza/create-envfile@v1.3	6s
Checkout	1s
Docker Buildx Set Up	7s
Login to DockerHub	0s
Frontend - Build And Push Docker Images	1m 1s
Create env file for backend	0s
Backend - Build And Push Docker Images	17s
Post Backend - Build And Push Docker Images	0s
Post Frontend - Build And Push Docker Images	0s
Post Login to DockerHub	0s
Post Docker Buildx Set Up	2s
Post Checkout	0s
Complete job	0s

Build-And-Push-Docker-Images

succeeded 5 days ago in 1m 32s

Search logs

```
#8 10.24 npm notice Changelog: <https://github.com/npm/cli/releases/tag/v8.7.0>
#8 10.24 npm notice Run `npm install -g npm@8.7.0` to update!
#8 10.24 npm notice
#8 DONE 10.5s

#9 [5/5] COPY . .
#9 DONE 0.1s

#10 exporting to image
#10 exporting layers
#10 exporting layers 2.7s done
#10 exporting manifest sha256:470c264cb964832383dae6d3ea25391686b42c6c4a913cc94f360190c4b16bac done
#10 exporting config sha256:94487a23070cedd47b09b9eaf82ecc6f4da70901024d9d31e224dbfb67395894 done
#10 pushing layers
#10 pushing layers 2.1s done
#10 pushing manifest for docker.io/***/dogdates:backend-
git@sha256:470c264cb964832383dae6d3ea25391686b42c6c4a913cc94f360190c4b16bac
#10 pushing manifest for docker.io/***/dogdates:backend-
git@sha256:470c264cb964832383dae6d3ea25391686b42c6c4a913cc94f360190c4b16bac 0.5s done
#10 DONE 5.2s
▼ ImageID
  sha256:94487a23070cedd47b09b9eaf82ecc6f4da70901024d9d31e224dbfb67395894
▼ Digest
  sha256:470c264cb964832383dae6d3ea25391686b42c6c4a913cc94f360190c4b16bac
▶ Metadata
```

Build and push docker image to Dockerhub

Post Backend - Build And Push Docker Images

Post Frontend - Build And Push Docker Images

Testing

Link to testing plan

https://github.com/DogDatesComp4350/DogDates/blob/main/Dog_Dates_Testing_Plan.pdf

Unit/integration/acceptance test

Note: We adopted the practice of functional components instead of class components. Due to this, components need to work with others to be functional. Therefore, some components can't be tested solely and testing them acts not just as a unit test, but also integration. Also, some statements are for error handling such as internet disconnected between our backend and MongoDB. These statements are untestable since we can't introduce such errors in our testing.

10 most important unit tests with links below:

Each of our test files contain multiple tests, the link will point to the first line of the test.

Test File path with clickable GitHub link	What is it testing
https://github.com/DogDatesComp4350/DogDates/blob/565ae918598b52aaa5e6cc6a191c764184fdecb3/backend/__tests__/unit_test/signup.js#L60	Password length verification at signup (ensuring the password length is not less than 6 chars)
https://github.com/DogDatesComp4350/DogDates/blob/565ae918598b52aaa5e6cc6a191c764184fdecb3/backend/__tests__/unit_test/signup.js#L151	Duplicate account registration verification (avoiding create a created account)
https://github.com/DogDatesComp4350/DogDates/blob/565ae918598b52aaa5e6cc6a191c764184fdecb3/backend/__tests__/unit_test/login.js#L99	Email verification during login (preventing users whose emails are not in the database from logging in)
https://github.com/DogDatesComp4350/DogDates/blob/565ae918598b52aaa5e6cc6a191c764184fdecb3/backend/__tests__/unit_test/login.js#L71	Password verification during login (ensuring the user input the correct password)
https://github.com/DogDatesComp4350/DogDates/blob/69f453f9138bb25ea32afe5642d271f3f2a1e049/backend/__tests__/unit_test/update.js#L110	Security verification when update profile by matching login token and uid (ensuring the user changes his/her profile, not someone else's)
https://github.com/DogDatesComp4350/DogDates/blob/69f453f9138bb25ea32afe5642d271f3f2a1e049/backend/__tests__/unit_test/like.js#L85	Ensure users can't like their own profiles

https://github.com/DogDatesComp4350/DogDates/blob/69f453f9138bb25ea32afe5642d271f3f2a1e049/backend/__tests__/unit_test/dislike.js#L85	Ensure users can't dislike their own profiles
https://github.com/DogDatesComp4350/DogDates/blob/69f453f9138bb25ea32afe5642d271f3f2a1e049/backend/__tests__/unit_test/match.js#L111	Ensure users match each other once they like each other
https://github.com/DogDatesComp4350/DogDates/blob/69f453f9138bb25ea32afe5642d271f3f2a1e049/backend/__tests__/unit_test/match.js#L125	Ensure users don't match with users they didn't like
https://github.com/DogDatesComp4350/DogDates/blob/69f453f9138bb25ea32afe5642d271f3f2a1e049/backend/__tests__/unit_test/delete.js#L80	Security verification on delete account by matching login token and uid (ensuring the user delete his/her account, not someone else's)

5 most important integration tests with links below:

Each of our test files contain multiple tests, the link will point to the first line of the test.

Test File path with clickable GitHub link	What is it testing
https://github.com/DogDatesComp4350/DogDates/blob/69f453f9138bb25ea32afe5642d271f3f2a1e049/backend/__tests__/integration_test/SignupLoginTests.js#L53	User signup with valid user inputs
https://github.com/DogDatesComp4350/DogDates/blob/69f453f9138bb25ea32afe5642d271f3f2a1e049/backend/__tests__/integration_test/SignupLoginTests.js#L93	User login with valid credential
https://github.com/DogDatesComp4350/DogDates/blob/69f453f9138bb25ea32afe5642d271f3f2a1e049/backend/__tests__/integration_test/UserAccountTests.js#L65	User profile valid update and improper update (update profile for a deleted user)
https://github.com/DogDatesComp4350/DogDates/blob/d9a12bc27eb21c4275a5e6fce400538a45ac368f/backend/__tests__/integration_test/ViewTests.js#L82	Users can see next user waiting to be paired and return blank when there are no other users left
https://github.com/DogDatesComp4350/DogDates/blob/d9a12bc27eb21c4275a5e6fce400538a45ac368f/backend/__tests__/integration_test/MatchUidTest.js#L126	Matched user's profile returned with email address

5 most important acceptance tests with links below:

Acceptance tests are done manually, steps are in our test plan
https://github.com/DogDatesComp4350/DogDates/blob/main/Dog_Dates_Testing_Plan.pdf

Test steps location	Which user story is it testing
Registration Test (Page 2 in the test plan)	Be able to create an account https://github.com/DogDatesComp4350/DogDates/issues/2
Log In Test (Page 2 in the test plan)	Be able to login an account https://github.com/DogDatesComp4350/DogDates/issues/3
Find Play Dates Test (Page 3 in the test plan)	Be able to like or dislike other users' profiles https://github.com/DogDatesComp4350/DogDates/issues/24
View Matches Test (Page 3 in the test plan)	Be able to see who I've matched with https://github.com/DogDatesComp4350/DogDates/issues/22
Update Profile Test (Page 3 in the test plan)	Be able to create and update profile https://github.com/DogDatesComp4350/DogDates/issues/23

Regression testing

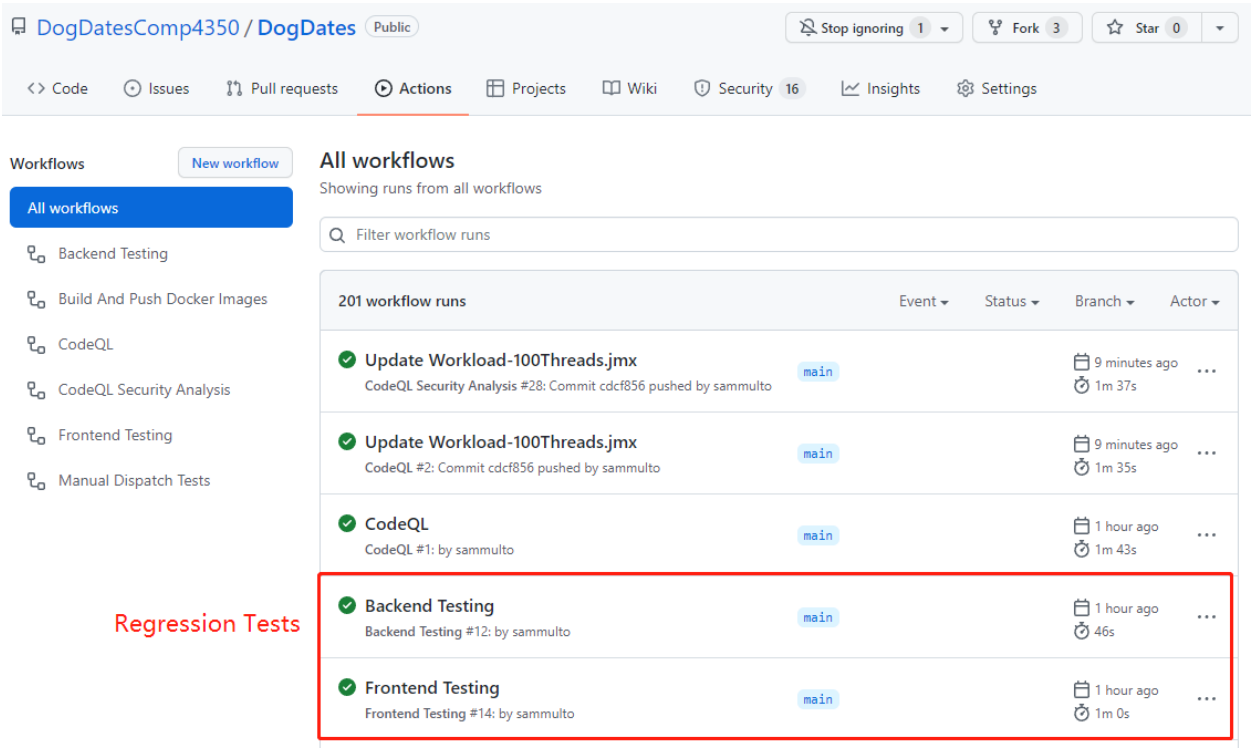
We used Github Action for regression testing. Unit tests and integration tests are executed for regression testing. Tests are run automatically by Github Actions when new code is committed to the repository or a pull request is created.

Regression testing scripts:

Frontend: https://github.com/DogDatesComp4350/DogDates/tree/main/frontend/src/__tests__

Backend: https://github.com/DogDatesComp4350/DogDates/tree/main/backend/__tests__

Regression tests execution snapshot:



Regression tests result snapshots:

Backend regression test results

```
Test Suites: 12 passed, 12 total
Tests:       95 passed, 95 total
Snapshots:   0 total
Time:        47.328 s
Ran all test suites.
Force exiting Jest: Have you considered using `--detectOpenHandles` to detect async operations that kept running after all tests finished?
```

Post Run actions/checkout@v2 0s

Complete job 0s

Frontend regression test results

```
> dogdates@0.8.0 test
> react-scripts test --watchAll=false

PASS src/__tests__/user/updateAccount.test.js
PASS src/__tests__/signup/signup.test.js
PASS src/__tests__/match/matchItem.test.js
PASS src/__tests__/login/login.test.js
PASS src/__tests__/user/deletedWarning.test.js
PASS src/__tests__/user/accountDeleted.test.js
PASS src/__tests__/home/home.test.js
PASS src/__tests__/home/findPlayDatesButton.test.js
PASS src/__tests__/home/editprofileButton.test.js
PASS src/__tests__/user/infoUpdated.test.js
PASS src/__tests__/home/myMatchButton.test.js
PASS src/__tests__/user/account.test.js
PASS src/__tests__/home/registerButton.test.js

Test Suites: 13 passed, 13 total
Tests:       48 passed, 48 total
Snapshots:   0 total
Time:        5.705 s
Ran all test suites.
```

Post Run actions/checkout@v2

Complete job

Load testing

https://github.com/DogDatesComp4350/DogDates/wiki/Load-Testing

Security analysis

We used CodeQL for security analysis. The analysis is run automatically by Github Action when new codes are committed to the repository or a pull request is created.

Security analysis report

Code scanning

Add scanning tool

Latest scan	Branch	Workflow	Lines scanned	Duration	Result
37 minutes ago	main	CodeQL Security Analysis	3.4k / 3.39k	1m 17s	17 alerts

is:open branch:main

9 Open ✓ 8 Closed

Tool ▾ Branch ▾ Rule ▾ Severity ▾ Sort ▾

- Uncontrolled data used in path expression High — 1 — main
  #1 opened 12 days ago • Detected by CodeQL in backend/app.js:75
- Missing rate limiting High — main
  #23 opened 2 days ago • Detected by CodeQL in backend/routes/dislike-route.js:14
- Missing rate limiting High — main
  #22 opened 2 days ago • Detected by CodeQL in backend/app.js:73
- Missing rate limiting High — main
  #21 opened 2 days ago • Detected by CodeQL in backend/app.js:50
- Database query built from user-controlled sources High — main
  #20 opened 2 days ago • Detected by CodeQL in backend/controllers/like-controller.js:40
- Database query built from user-controlled sources High — main
  #18 opened 2 days ago • Detected by CodeQL in backend/controllers/auth-controller.js:96
- Database query built from user-controlled sources High — main
  #7 opened 12 days ago • Detected by CodeQL in backend/controllers/users-controller.js:169
- Database query built from user-controlled sources High — main
  #6 opened 12 days ago • Detected by CodeQL in backend/controllers/users-controller.js:110
- Database query built from user-controlled sources High — main
  #5 opened 12 days ago • Detected by CodeQL in backend/controllers/signup-controller.js:117

ProTip! The libraries and queries that power CodeQL are open-source. Learn more

Detected problem 1:

Uncontrolled data used in path expression

Open in main 14 seconds ago Tracked in #108

backend/app.js:75

```
72  //other errors
73  app.use((err, req, res, next) => {
74    if (req.file) {
75      fs.unlink(req.file.path, (err) => {
```

This path depends on a user-provided value.

CodeQL Show paths

The file path is provided by the API user which allows the attacker to access sensitive server resources. However, this path is validated before the file system operation, hence, this warning is a false positive.

Detected problem 2:

Database query built from user-controlled sources

Open in main 33 minutes ago

backend/controllers/like-controller.js:40

```
37    try {
38      //get the liked user's View list
39      let targetUserViewList = await viewListModel
40        .findOne({ uid: targetUid })
```

This query depends on a user-provided value.

CodeQL Show paths

The targetUid is provided by the user and is not sanitized before using it in DB query. A malicious DB query may be run by the user.

Detected problem 3:

Database query built from user-controlled sources

Open in `main` 34 minutes ago

backend/controllers/users-controller.js:169

```
166     let user = false;
167
168     try {
169       let result = await UserModel.find({ email: email }).exec();
```

This query depends on a user-provided value.

CodeQL Show paths

The variable ‘email’ is provided by the user and is not sanitized before using it in DB query. A malicious DB query may be run by the user.

Detected problem 4:

Missing rate limiting

Open in `main` 1 minute ago

backend/routes/dislike-route.js:14

```
11  router.patch(
12    "/:uid",
13    [check("uid").not().isEmpty()],
14    controller.addUidToDislikeList
```

This route handler performs a database access, but is not rate-limited.
This route handler performs a database access, but is not rate-limited.

CodeQL

This API handler performs two DB operations per API call without a rate limit. This makes the application vulnerable to DDoS attacks.

Detected problem 5:

Missing rate limiting

Open in `main` 1 minute ago

backend/**app.js**:50

```
47  app.use("/api/auth", authRoutes);
48
49  //validate user's token before proceding to the protected API routes
50  app.use(authenticator);
```

This route handler performs authorization, but is not rate-limited.

CodeQL

Authentication operations without a rate limit make the application vulnerable to user account brute-force attacks.