

Affordable Admin Capstone Summer 2020

Andy Zawada, Jacob Iannarino, Yunseong Lee, Konrad Kappel

Table Of Contents

Setup

Implementation

Testing / CI

Process

Technical Dictionary

Conclusion

Introduction

Konrad

The Affordable application allows regular users in need of funding for healthcare to send in an application to Affordable and explain their needs and provide documentation. If accepted, the recipient can accept the funds into a stripe connected account and register their bank account using Plaid, which allows them to withdraw the funds directly into their bank account. The Affordable application also has an Admin user type. The Admin user type exists to bring more security and control to the Affordable application. The Admin is able (based on privileges set for that specific admin) to approve/reject admin requests, set admin privileges, deactivate/activate users, reset user MFA and password, email users, and revoking admin access.

Setup

TODO - Add linux documentation and test Windows setup.

Windows

The following steps have been tested against a clean version of Windows 10 Enterprise version 2004 (Build 19041).

In an admin PowerShell prompt go through the following steps. The prompt may need to be restarted several times throughout to pick up new packages.

1. Install Cocolatey, a package manager for Windows. Follow the instructions at <https://chocolatey.org/install>
2. Install git using chocolatey: `choco install git`
3. Install node 12 LTS: `choco install nodejs-lts`
4. Install Python 2: `choco install python2`
5. Install pip, the Python package manager: `choco install pip2`
6. Install the selenium package: `pip install selenium`
7. In your workspace folder clone the repo: `git clone https://www.x3xoj4kpf0ql1dwzact9uu8v01ic81pd.net:5443/sean/AffordableServerOSU`
8. `cd` into the new directory that has been created.
9. Install other necessary node modules:
10. Build the whole application. From the root of the repo run `yarn build`
11. Add the appropriate `.env` files listed below
12. In one prompt `cd` into the app directory and run `yarn start`
13. In another prompt `cd` into the server directory and run `yarn start`

Linux

The following steps have been tested against a clean version of Ubuntu 18.04.3 LTS.

In a bash prompt run execute the following steps:

1. Become root:
`sudo su`
2. Install git:
`apt-get install git`
3. Install the snap package manager:
`apt-get install snapd`
4. Install node 12 LTS:
`snap install --channel=12/stable --classic`
5. Install Python 2:
`apt-get install python-minimal`

6. Install pip, the Python package manager:
`apt-get install python-pip`
7. Install the selenium package:
`pip install selenium`
8. In your workspace folder clone the repo:
`git clone --single-branch --branch OSU-develop
https://www.x3xoj4kpf0ql1dwzact9uu8v01ic81pd.net:5443/sean/AffordableServerOSU.git`
9. cd into the new directory that has been created.
10. Install other necessary node modules:
`sudo apt install yarn nodejs python2`
11. Install mariadb, using ppa to get newest version (old version in the Ubuntu repo has incompatible syntax):
<https://computingforgeeks.com/install-mariadb-10-on-ubuntu-18-04-and-centos-7/>
First, uninstall any current version present:
`sudo apt-get remove mariadb-server`
Then run this whole string:
`sudo apt-get install software-properties-common ; sudo
apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80
0xF1656F24C74CD1D8 ; sudo add-apt-repository "deb
[arch=amd64,arm64,ppc64el]
http://mariadb.mirror.liquidtelecom.com/repo/10.4/ubuntu
$(lsb_release -cs) main" ; sudo apt update ; sudo apt -y install
mariadb-server mariadb-client`
Enter a password when prompted to complete installation.
Then start it, entering your password:
`mysql -u root -p`
Verify the version:
`SELECT VERSION();`
Make sure it's at least 10.4.6.
12. Build the whole application. From the root of the repo run
`yarn build`
13. Add the appropriate .env files listed below
14. Like with the .env SES keys, ask Sean and Warren for the SSL certificates to put in the server directory. Then specify the path of the certificate files in app/package.json, "scripts" section, in the "prod" command. (This command is used to start the frontend using HTTPS on a server.) Also specify these files in

server/src/init.ts, in the section of code labeled with the comment “// Load HTTPS certs”.

15. Also, set up certbot for certificate generation:

https://lightsail.aws.amazon.com/ls/docs/en_us/articles/amazon-lightsail-using-lets-encrypt-certificates-with-wordpress

Do 2.{5,6,7} on that page:

```
sudo apt-add-repository ppa:certbot/certbot -y ; sudo
apt-get update -y ; sudo apt-get install certbot -y
```

And then run:

```
sudo certbot -d demo.affordhealth.org --manual
--preferred-challenges dns certonly
```

Use the email: sean.pannella@affordhealth.org

Then ask Sean to complete the steps on his end.

16. Add inotify fix before starting frontend:

Check limit with

```
cat /etc/sysctl.d/50-max_user_watches.conf
```

Change /etc/sysctl.d/50-max_user_watches.conf to 524288

Then do

```
echo fs.inotify.max_user_watches=524288 | sudo tee
/etc/sysctl.d/50-max-user-watches.conf && sudo sysctl --system
```

Check it again to see if it changed:

```
cat /etc/sysctl.d/50-max_user_watches.conf
```

17. In one prompt cd into the app directory and run

```
yarn start
```

18. In another prompt cd into the server directory and run

```
yarn start
```

.env Templates

server/.env:

```
AFFORDABLE_DB_HOST= 127.0.0.1
AFFORDABLE_DB_USER=root
AFFORDABLE_DB_PASSWORD=password
AFFORDABLE_DB_NAME=Affordable

AFFORDABLE_SES_KEY_ID=<foo>
AFFORDABLE_SES_KEY_SECRET=<foo>
AFFORDABLE_EMAIL_ENABLED=false
AFFORDABLE_SES_REGION=us-east-1
```

```
AWS_ACCESS_KEY_ID=<foo>
AWS_SECRET_KEY_SECRET=<foo>
AWS_BUCKET_NAME=ou.messenger.test

AFFORDABLE_FRONTEND_URL=http://localhost:3000
AFFORDABLE_BACKEND_URL=http://localhost:4000
AFFORDABLE_ADMIN_USER=admin
AFFORDABLE_ADMIN_PASSWORD=password
AFFORDABLE_ADMIN_EMAIL=admin@affordhealth.org
AFFORDABLE_ADMIN_ID=1
AFFORDABLE_TOKEN_SIGNING_KEY=secret
AFFORDABLE_PAYOUT_ACCOUNT=<foo>

STRIPE_SECRET_KEY=<foo>
PLAID_CLIENT_ID=<foo>
PLAID_SECRET=<foo>
PLAID_PUBLIC_KEY=<foo>

MAX_FILE_SIZE=25000000
UPLOAD_FOLDER=testFolder // Folder within S3 bucket
REQUIRE_JWT_VERIFICATION=false
```

app/.env

```
REACT_APP_STRIPE_PUBLISHABLE_KEY=<foo>
REACT_APP_PLAID_CLIENT_ID=<foo>
REACT_APP_PLAID_SECRET=<foo>
REACT_APP_PLAID_PUBLIC_KEY=<foo>
REACT_APP_AF_BACKEND_URL=http://localhost:4000
```

client/.env -- may not be necessary

```
REACT_APP_AF_BACKEND_URL=http://localhost:4000
```

Implementation

TODO - Add emailing documentation


1. Two Factor

- a. Two factor code has been reimplemented with new features.

Descriptions and example screenshots

Upon registration, users can request two-factor authentication by clicking the checkbox under the email form field.

AFFORDABLE



Open an account

This username is available

(Must contain at least 1 number or special character.)

Password Match

☐ Request 2-Factor Authentication (Requires a Mobile Device)

The user will be presented with the two-factor form options. Here they enter a name for their device, scan the QR-code with the google authenticator app, input the 6-digit code the app gives them.


(Must contain at least 1 number or special character.)

Password Match

☒ Request 2-Factor Authentication (Requires a Mobile Device)

Option provides additional security to your information on login

1. Download the "Google Authenticator" app
2. Use your camera to scan the QR code below
3. Enter your device name
4. Input the 6-digit code shown on your mobile device



We also recommend securing your device with a password to prevent fraud

After filling out the form, the user presses the 'Authenticate' button to apply two factor authentication to their account.

(Must contain at least 1 number or special character.)

Password Match

testuser@gmail.com

☒ Request 2-Factor Authentication (Requires a Mobile Device)

Option provides additional security to your information on login

1. Download the "Google Authenticator" app
2. Use your camera to scan the QR code below
3. Enter your device name
4. Input the 6-digit code shown on your mobile device

my_phone

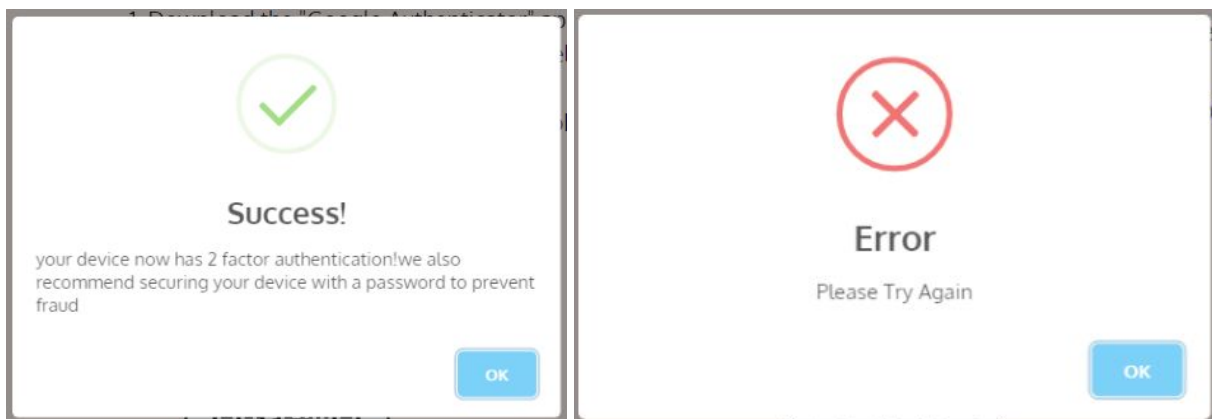
625424

Authenticate

We also recommend securing your device with a password to prevent fraud

Apply Now

The user is then presented with a success dialog if their code was valid or error dialog box if their code was invalid.



Two-factor authentication settings have been added to the settings page. If the user currently has two-factor enabled on his or her account they are presented with a form to remove it.

Please click the link in your confirmation email to confirm your email address.

testuser ▾

Settings

Modify MFA

Account Banking Information **Modify MFA**

Enter Code to Remove two Factor Authentication

Enter Code

Remove Two Factor

Otherwise, if the user does not have two-factor registration, they are able to enable two-factor here.

Please click the link in your confirmation email to confirm your email address.

testuser ▾

Settings

Modify MFA

Account Banking Information **Modify MFA**

Add Two Factor Authentication Here

1. Download the "Google Authenticator" app
2. Use your camera to scan the QR code below
3. Enter your device name
4. Input the 6-digit code shown on your mobile device

Enter Device Name

Enter Code

Authenticate

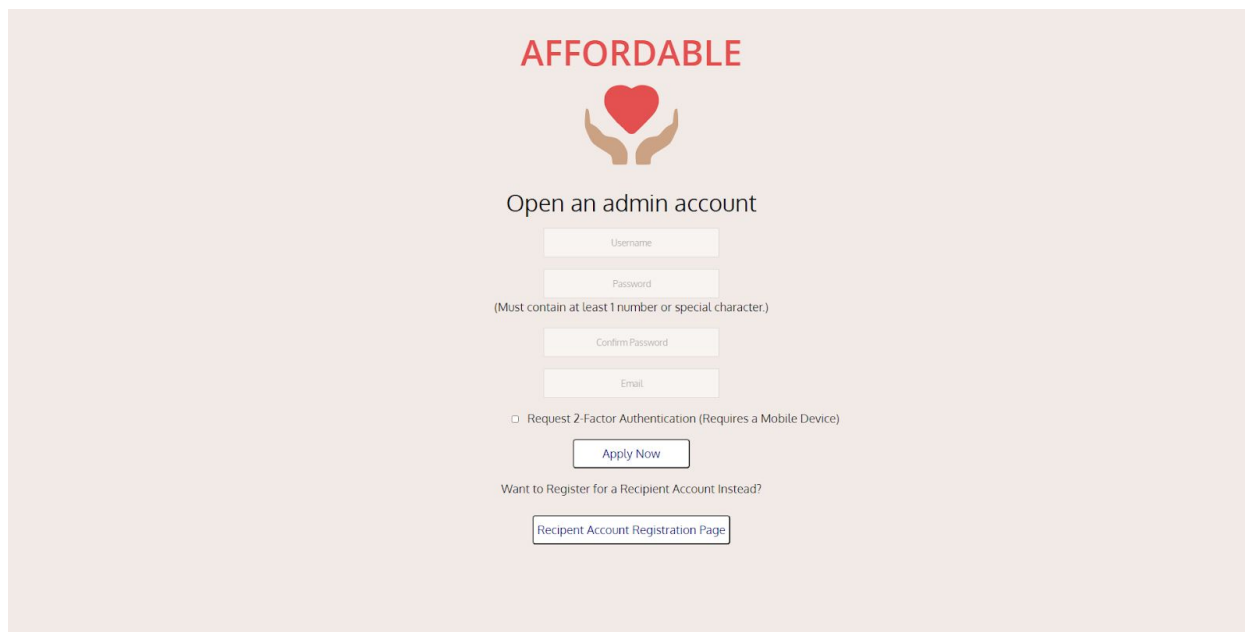
2. Admin Registration Page

- When a user registers for an admin account, they are not immediately logged into the application. Their registration must first

be approved by an existing admin with the AllowRejectAdminRegistration privilege via the Approve Admin Registration Page.

- b. The admin registration form can be accessed at the /admin-register endpoint

Descriptions and Example screenshots

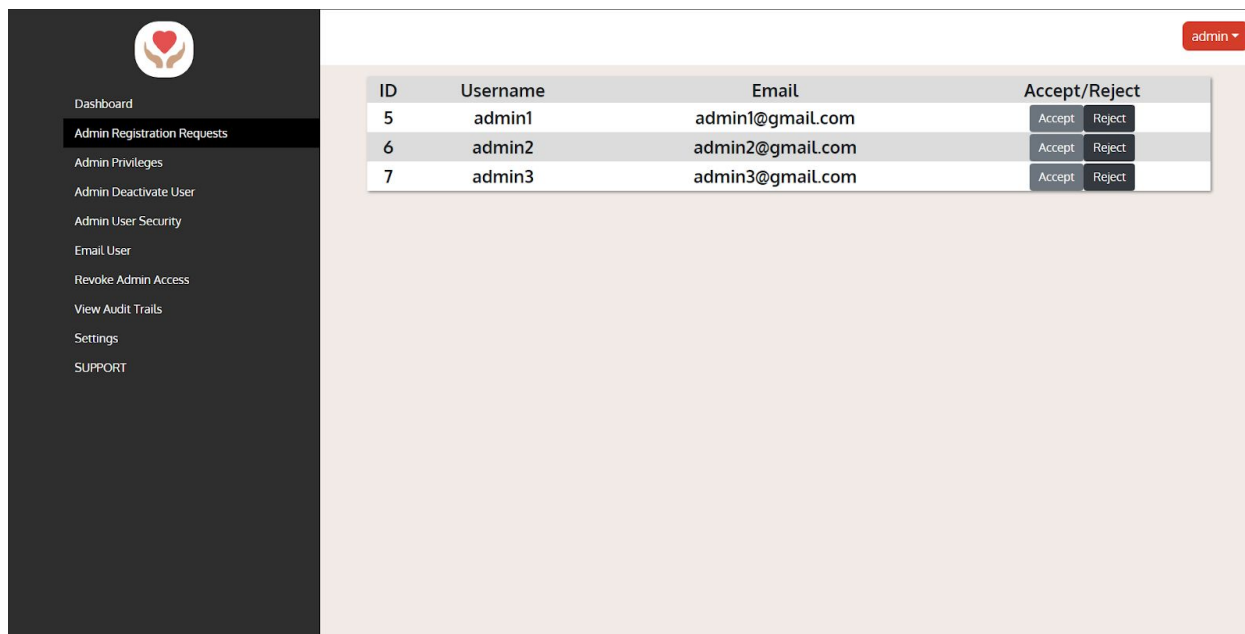


The screenshot shows the 'AFFORDABLE' Admin Registration Page. At the top, the word 'AFFORDABLE' is in red, followed by a logo of two hands holding a red heart. Below the logo, the text 'Open an admin account' is centered. The form contains four input fields: 'Username', 'Password', 'Confirm Password', and 'Email'. A note below the Password field states '(Must contain at least 1 number or special character.)'. Below the Confirm Password field, there is a checkbox labeled 'Request 2-Factor Authentication (Requires a Mobile Device)'. An 'Apply Now' button is positioned below the checkbox. At the bottom, the text 'Want to Register for a Recipient Account Instead?' is followed by a button labeled 'Recipient Account Registration Page'.

3. Admin Registration Requests Page

- a. **Required Admin Privilege: AllowRejectAdminRegistration**
- b. View incoming admin registration requests. Allow or reject requests

Descriptions and example screenshots




ID	Username	Email	Accept/Reject
5	admin1	admin1@gmail.com	Accept Reject
6	admin2	admin2@gmail.com	Accept Reject
7	admin3	admin3@gmail.com	Accept Reject

(Overall view of Admin Registration Requests page)

Admin Registration Requests

“Admin Registration Requests” button on the side panel will navigate admin with “AllowRejectAdminRegistration” privileges into the “/requests” page.

 localhost:3000/requests

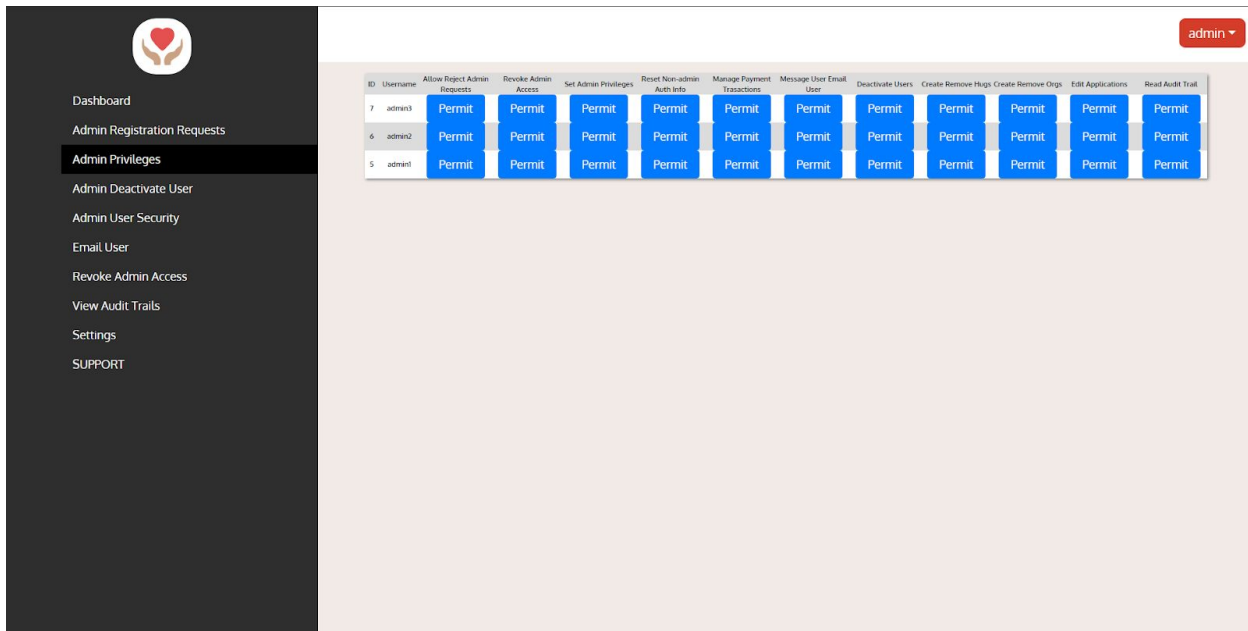
Admin Can view incoming admin registration requests and accept and reject them using the buttons in the rightmost column of the table.

ID	Username	Email	Accept/Reject	
5	admin1	admin1@gmail.com	Accept	Reject
6	admin2	admin2@gmail.com	Accept	Reject
7	admin3	admin3@gmail.com	Accept	Reject

4. Admin Privileges Page

- Required Admin Privilege: SetPrivileges**
- View admin privileges. Permit or revoke privileges for each admin.

Descriptions and example screenshots




The screenshot shows the Admin Privileges page. On the left is a sidebar menu with the following items: Dashboard, Admin Registration Requests, Admin Privileges (highlighted), Admin Deactivate User, Admin User Security, Email User, Revoke Admin Access, View Audit Trails, Settings, and SUPPORT. The main content area displays a table with 12 columns representing different privileges and 3 rows of admin users. Each cell in the table contains a blue 'Permit' button. A red 'admin' dropdown menu is visible in the top right corner.

ID	Username	Allow Reject Admin Requests	Revoke Admin Access	Set Admin Privileges	Reset Non-admin Auth Info	Manage Payment Transactions	Message User Email User	Deactivate Users	Create Remove Hugs	Create Remove Orgs	Edit Applications	Read Audit Trail
7	admin3	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit
6	admin2	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit
5	admin1	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit

(Overall view of Admin Privileges Page)

Admin Privileges

“Admin Privileges” button on the side panel will navigate admin with “SetPrivileges” privileges into the “/privileges” page.

 localhost:3000/privileges

Admin can view privileges set for each admin on the sight in the privileges table. By default, new admins have no privileges.

ID	Username	Allow Reject Admin Requests	Revoke Admin Access	Set Admin Privileges	Reset Non-admin Auth Info	Manage Payment Transactions	Message User Email User	Deactivate Users	Create Remove Hugs	Create Remove Orgs	Edit Applications	Read Audit Trail
7	admin3	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit
6	admin2	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit
5	admin1	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit	Permit

From this table, the admin can permit a privilege to a specific admin.

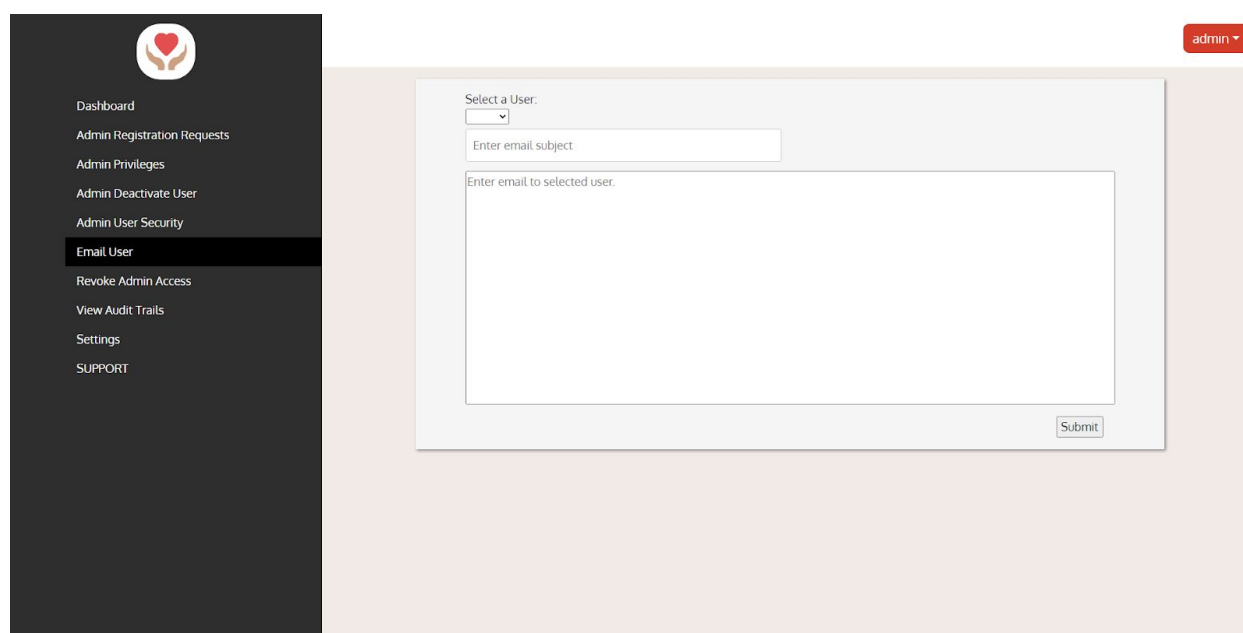
ID	Username	Allow Reject Admin Requests	Revoke Admin Access	Set Admin Privileges	Reset Non-admin Auth Info	Manage Payment Transactions	Message User Email User	Deactivate Users	Create Remove Hugs	Create Remove Orgs	Edit Applications	Read Audit Trail
7	admin3	Revoke	Revoke	Revoke	Revoke	Permit	Revoke	Permit	Permit	Permit	Permit	Revoke
6	admin2	Permit	Permit	Permit	Permit	Permit	Revoke	Permit	Permit	Permit	Permit	Permit
5	admin1	Permit	Permit	Permit	Revoke	Permit	Revoke	Permit	Permit	Permit	Permit	Permit

Once an admin is given a specific privilege, the 'permit' button changes to 'revoke' allowing the admin to revoke the granted privilege if necessary.

5. Admin User Email Page

- Required Admin Privilege: MessageUserEmailUser**
- View list of all active users. Send email to selected users based on information passed to the email header and email body text fields.

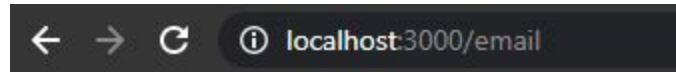
Descriptions and example screenshots



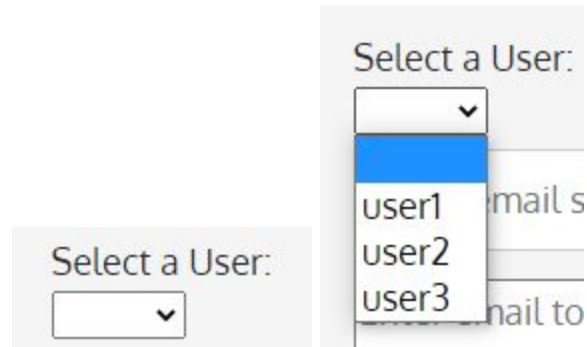
(Overall view of Admin User Email page)

Email User

“Email User” button on the side panel will navigate admin with “MessageUserEmailUser” privileges into the “/email” page.



Admin can select a user to email from a dropdown list of all users.



The Admin can then fill out the Email Subject and Email Body fields and submit the form so send the email to the selected user.

A screenshot of a web form for sending an email. It features a 'Select a User:' dropdown menu with 'user1' selected. Below this is a text input field labeled 'Email Subject'. Underneath is a large text area labeled 'Email Body.'. At the bottom right of the form is a 'Submit' button.

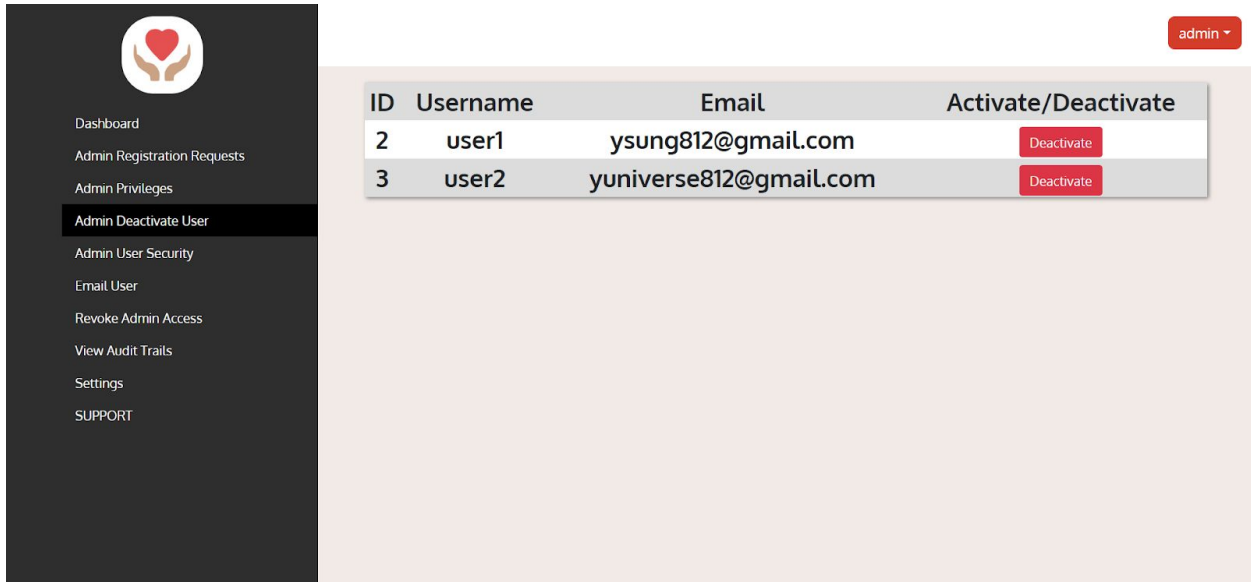
After Submission, the page is refreshed and the email is sent.

6. Admin Deactivate User Page

a. Required Admin Privilege: DeactivateUsers

- b. View list of all users. Deactivate/Activate users and sends email to users about their account status

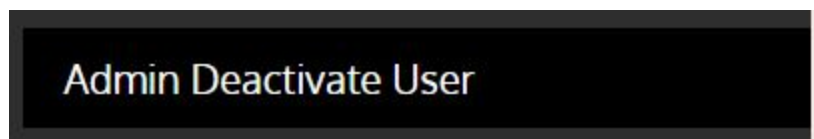
Descriptions and example screenshots



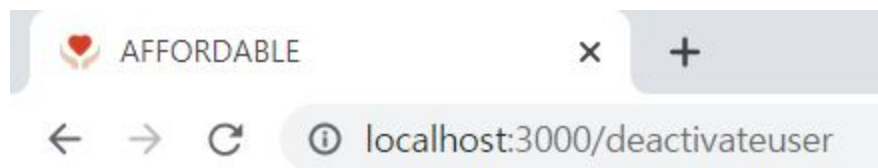
The screenshot displays the 'Admin Deactivate User' page. On the left is a dark sidebar with a logo at the top and a list of navigation items: Dashboard, Admin Registration Requests, Admin Privileges, Admin Deactivate User (highlighted), Admin User Security, Email User, Revoke Admin Access, View Audit Trails, Settings, and SUPPORT. The main content area has a light beige background. In the top right corner, there is a red 'admin' button. Below it is a table with the following data:

ID	Username	Email	Activate/Deactivate
2	user1	ysung812@gmail.com	Deactivate
3	user2	yuniverse812@gmail.com	Deactivate

(Overview of the Admin Deactivate User Page)



The “Admin Deactivate User Page” will navigate admin user to the “/deactivateuser” page and admin will be able to see the list of all active or deactivated users and their informations in this page along with the “Activate” or “Deactivate” button.



ID	Username	Email	Activate/Deactivate
2	user1	ysung812@gmail.com	<button>Deactivate</button>
3	user2	yuniverse812@gmail.com	<button>Deactivate</button>

If the user is active, there will be a “Deactivate” button and if the user is deactivated, there will be a “Activate” button. The button will switch as the admin presses the button and the status of the user account changes.

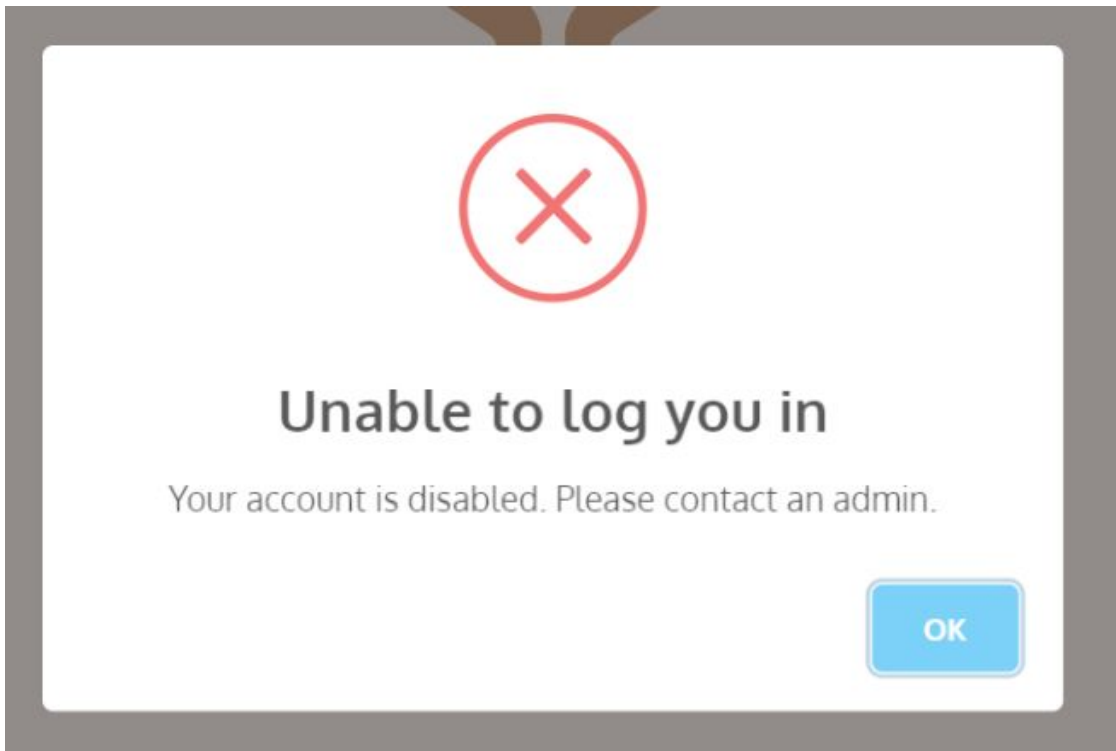


Once the admin deactivates or activates the user, that user will receive a notification email about their account status.

When the user is deactivated, the application will block them out from logging in. For example, if the admin clicked “Deactivate” button for the “user1” in the above figure.

The image shows a login form on a light beige background. At the top, there is a small illustration of a person's head and shoulders. Below this, the form has two input fields. The first field is labeled 'USERNAME' and contains the text 'user1'. The second field is labeled 'PASSWORD' and contains seven dots. Below the password field is a white button with the word 'Login' in blue text.

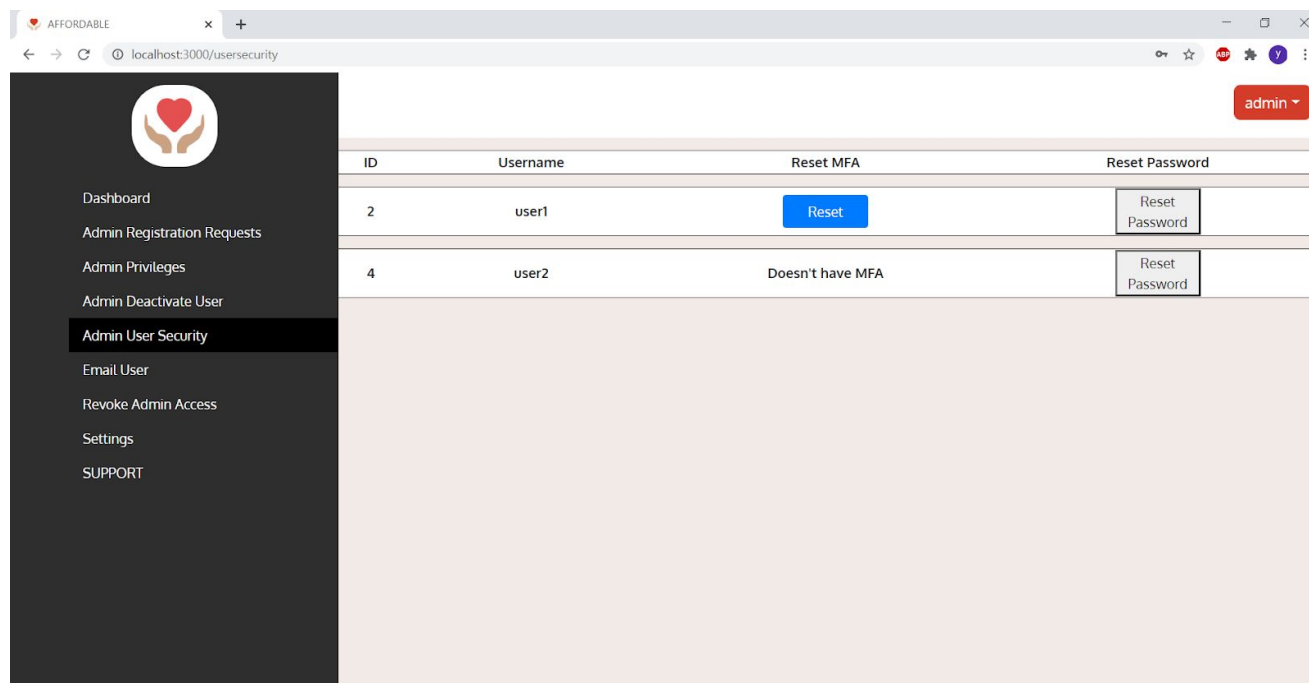
Next time user1 tries to log-in, the application will throw a popup, letting them know their account has been disabled.



7. Admin User Security Page

- a. **Required Admin Privilege: ResetAuthInfoNonAdmin**
- b. View list of all users. When resetting a user's MFA, it sends the notification email and when resetting the user's password, it sends the email with a link to the front end page for the user to change their password.

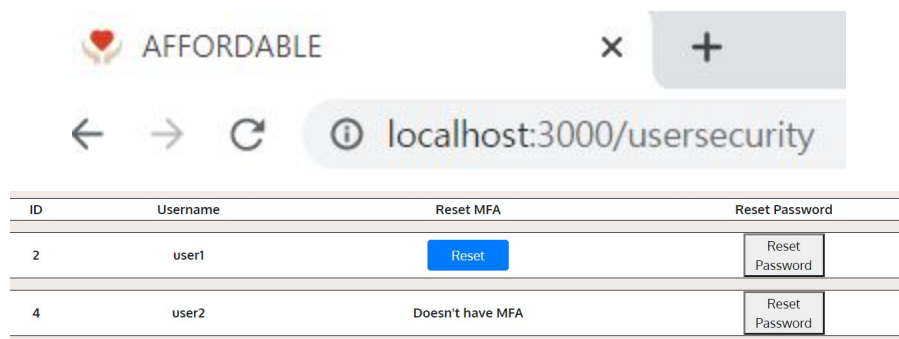
Descriptions and example screenshots



(Overall view of Admin User Security page)



“Admin User Security” button on the side panel will navigate admin with “DeactivateUsers” privileges into “/usersecurity” page.



(example figure of “/usersecurity” page)

Doesn't have MFA

When a user doesn't have any MFA registered, there won't be any button to reset their MFA.



When a user has MFA registered, there will be a button to reset their MFA. Resetting a user's MFA will send out a notification email to that user with a text saying their MFA has been reset. Once the admin clicks the "Reset" button, the button will disappear and change to "Doesn't have MFA" text.

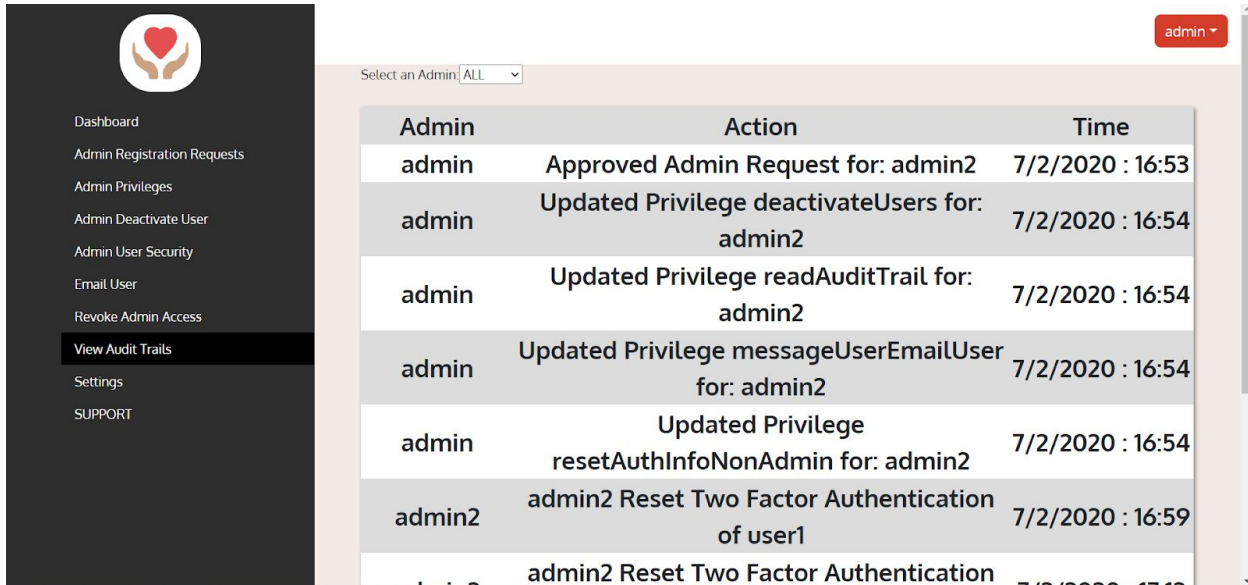


"Reset Password" button will reset the password of any user on the list and sends the email that has a link to the front end page where users can change their password without knowing their previous password.

8. Admin View Audit Trail Page

- a. **Required Admin Privilege: ReadAuditTrail**
- b. View list of audit trails by all admin users.

Descriptions and example screenshots



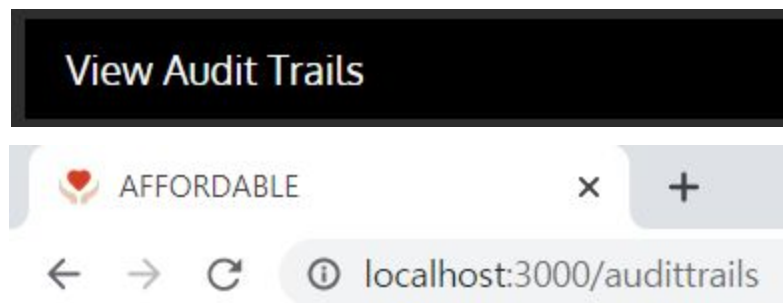
The screenshot shows a web application interface. On the left is a dark sidebar with a logo at the top (a heart with hands) and a list of navigation items: Dashboard, Admin Registration Requests, Admin Privileges, Admin Deactivate User, Admin User Security, Email User, Revoke Admin Access, View Audit Trails (highlighted), Settings, and SUPPORT. The main content area has a light background. At the top right of this area is a red button labeled 'admin'. Below it is a dropdown menu labeled 'Select an Admin' with 'ALL' selected. The main part of the content area is a table with three columns: Admin, Action, and Time. The table contains several rows of audit logs.

Admin	Action	Time
admin	Approved Admin Request for: admin2	7/2/2020 : 16:53
admin	Updated Privilege deactivateUsers for: admin2	7/2/2020 : 16:54
admin	Updated Privilege readAuditTrail for: admin2	7/2/2020 : 16:54
admin	Updated Privilege messageUserEmailUser for: admin2	7/2/2020 : 16:54
admin	Updated Privilege resetAuthInfoNonAdmin for: admin2	7/2/2020 : 16:54
admin2	admin2 Reset Two Factor Authentication of user1	7/2/2020 : 16:59
admin2	admin2 Reset Two Factor Authentication	7/2/2020 : 17:12

(Overview of the Audit Trail Page)

View Audit Trails

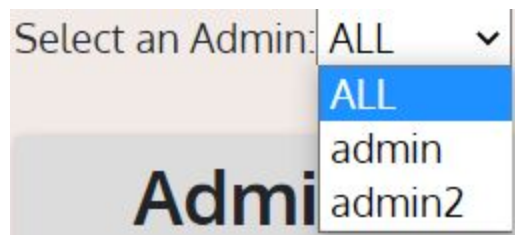
The “View Audit Trails” tab on the sidebar will navigate admin user to “/audittrails” or the audit trail page.



Admin can select which admin’s trails to look at and view a list of all the audit trails done by that selected admin. The selected admin will be “ALL” by default, which will show a list of audit trails done by all admin.

Select an Admin: ALL ▾		
Admin	Action	Time
admin	Approved Admin Request for: admin2	7/2/2020 : 16:53
admin	Updated Privilege deactivateUsers for: admin2	7/2/2020 : 16:54
admin	Updated Privilege readAuditTrail for: admin2	7/2/2020 : 16:54
admin	Updated Privilege messageUserEmailUser for: admin2	7/2/2020 : 16:54
admin	Updated Privilege resetAuthInfoNonAdmin for: admin2	7/2/2020 : 16:54
admin2	admin2 Reset Two Factor Authentication of user1	7/2/2020 : 16:59
admin2 Reset Two Factor Authentication		

Admin will be able to view the list of all active admins on this drop down. Once the admin selects which admin's trail they want to view, the view of the list will change and show the trails of the selected admin.



For example, if admin were to select admin user named "user2" on the drop down, the page will show the list of audit trails by "admin2".

Select an Admin: admin2 ▼

Admin	Action	Time
admin2	admin2 Reset Two Factor Authentication of user1	7/2/2020 : 16:59
admin2	admin2 Reset Two Factor Authentication of user1	7/2/2020 : 17:12
admin2	admin2 Reset Password of user1	7/2/2020 : 17:13
admin2	admin2 Deactivated User user1	7/2/2020 : 17:13
admin2	admin2 Emailed User with id 2	7/2/2020 : 17:15

9. Emailing - **Andy**

- a. **Blah Blah Blah**
- b. Blah Blah Blah

Descriptions and example screenshots

Blah Blah Blah

Other Improvements

Login

The team spent time cleaning up the authentication logic between the frontend and the backend. We added an `InvalidLoginResponse` type to be returned by the backend when the backend deems the login response to be invalid. Currently this happens in four conditions:

- The user has entered an invalid username.
- The user has entered an invalid password.
- The user is an admin that has not been approved yet.
- The account in question has been deactivated.

In the frontend we have added switching between the different return types the login method can have, and we have added the appropriate alerts to the user explaining what the issue is.

JWT Verification

While implementing the two factor TS conversion the team discovered that the authentication logic for the REST calls to the backend did not respect the results of the JWT verification logic. This meant that the entire backend was unauthenticated and could be called by anyone with a REST testing client such as Postman. To fix this we implemented strict checking on the validity of the incoming JWT tokens and added the flag `REQUIRE_JWT_VERIFICATION` in the server `.env` file to make this strict checking optional if it interferes with the functionality of the app. The team also performed a quick audit and modified the Express routing to use JWT verification as needed, as before the method for JWT verification was called globally, even though it wasn't enforced.

Fetch API Conversion

After the JWT Verification effort the team noticed several pages were throwing errors that we had not experienced before. The team researched and realized that several frontend pages are not utilizing the `AffordableClient` and are still using the `Fetch` API to make calls to the backend. These old calls do not contain any authentication, so they will fail when they attempt to hit the backend. The team corrected the calls to use the clients in the components we touched outside of the normal admin pages, such as the login and settings components.

Testing / CI

Integration Tests: Selenium tests

TODO - Add CI. look over intro

Intro

The test suite was built on Selenium tests. These tests serve as integration tests that give feedback to developers on the code they produced. These tests can be either run locally or through the Gitlab CI pipeline. The Gitlab CI pipeline runs every time any developer would push code. This CI pipeline would give feedback to each developer after they pushed to see if any errors were caused by the code they pushed.

Setup

1. **Install selenium:** "pip install selenium"
2. **Chromium Webdriver:** Already in the repo. Check if it is compatible with your version of chrome.
3. **Run a specific test:** python test-name.py
4. **To run the test suite locally:** From the selenium folder run `./run-integration.sh` in Git Bash

Implementation

1. Used selenium tests to make integration tests
2. **Test Readme:** each test near the top of the file has a readme with what the test is supposed to do.
3. **Test Dependency list:** Each test below the readme has a test dependency list. This list states any tests that need to be run before that test and other dependency notes.
4. **Test Format:** Each test is formatted in a try catch statement: This starts the test with the phrase "Running testname.py". Then if it passes without any failures or tripping any assert statement it will go to the block that will display "testname.py was successful". If the test fails or trips an assert it will go to the block that states "testname.py has failed"
5. **Test Flags:** Every test has flags near the beginning of the test to catch whether the test is running on a local windows machine or on the gitlab linux CI. This will direct the test to the correct chromium webdriver for the system. It will also make the CI version run headless (No GUI).
6. **Script File:** Run-integration.sh will run all of the tests up to this point in the correct order test dependency wise. The script will also run both on a local windows machine and in the gitlab CI.
7. **Legacy Tests:** Applicable legacy tests to our version of the code were reformatted to our standard in terms of (functionality, formatting, and flags). The legacy tests that were not applicable to our version did not get that overhaul and were put into "AffordableServerOSU/selenium/legacy_tests".

Improvements:

1. **CI time savings (Caching):** When we started the project the tests took about 45-60 minutes to run in the CI pipeline. When we added caching the pipeline took 15 minutes. This was a massive time saver. Currently the pipeline aggressively caches all of the node_modules directories. If there is an issue with the node_modules this can be resolved by running the pipeline in gitlab CI without cache.
2. **Test formatting:** We added the try/catch blocks into the code to intelligently fail/succeed the tests, and to gracefully stop the chromium driver when tests fail.
3. **Portability of tests:** We added flags so that way the tests can pull the correct chromium webdriver depending on whether you are running locally on a Windows machine or on the Linux Gitlab CI.

Process

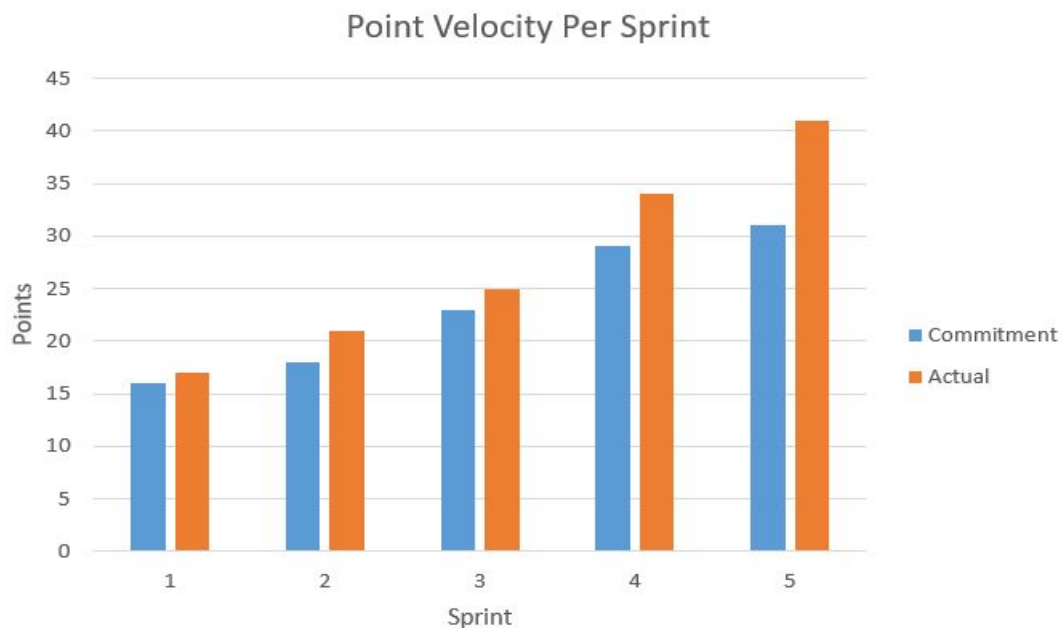
Description

The team adopted a scrum process to use for the semester. The team chose to run two-week sprints, and plan for five sprints throughout the semester, from week three to week twelve of the semester. The team used story points to estimate our work, adopting a one point per hour model of scoring cards. The team used Jira to manage our sprints and project backlog, and utilized an integration with Slack so the team could get notifications when work was being done by team members.

Team Roles

Name	Phone	Email	Role
Andy Zawada	(614)940-3005	zawada.11@osu.edu	Scrum Master co-PM
Jacob Innarino	(614)286-3641	iannarino.22@osu.edu	Architect co-PM
Yunseong Lee	(419)905-5502	lee.7478@osu.edu	Requirements Lead
Konrad Kappel	(614)390-0319	kappel.12@osu.edu	Test Lead

Burndown chart



Project Schedule

Project Task	Planned Start Date	Planned End Date	Actual End State
Milestone 1 Objectives			
MFA Code conversion	5/25	6/8	6/15
Learning App Architecture	5/11	6/8	6/8
Learning Testing Framework	5/11	6/8	6/9
Learning Development Tools	5/11	6/8	6/9
Milestone 2 Objectives			
Admin Registration Page	6/8	6/22	6/23
Admin Access Page	6/8	6/22	6/23
Various Email Notifications	6/22	7/6	7/7
Admin Privileges Page	6/22	7/6	7/7
Milestone 3 Objectives			
Admin Block/Deactivate User Page	7/6	7/20	7/21
Admin Email Page	7/6	7/20	7/21
Admin Audit Trail Page	7/20	8/3	
Admin Security Page	7/20	8/3	
Stretch Goals			
User Application Editing Page	Potentially Milestone 3		

Project Goals

Requirement	Milestone Group	Status
-------------	-----------------	--------

Create user types in the database to handle an admin user and a special admin which is the root user who has all privileges	Milestone 2	Complete
Create a separate admin registration page with the same information as the regular registration just linked at a different page	Milestone 2	Complete
Convert existing Javascript MFA and activity manager code into the new typescript codebase	Milestone 1	Complete
Create an admin privileges page where an admin can set other admin privileges	Milestone 2	Complete
Create a page for admins to manage admin account access and shows admin registration requests to manage	Milestone 2	Complete
Create a page for admins to change user's passwords, security questions or MFA devices	Milestone 3	Complete
Create a page for admins to block / deactivate user accounts	Milestone 3	Complete
Create a page for editing user's applications	Stretch Goal	Not Implemented
Create a page where an admin could email any user with a search on users	Milestone 3	Complete
Create an audit trail page which will include all actions done by any admin	Milestone 3	Complete
Create an email for when an admin has been allowed access to notify the new user	Milestone 2	Complete
Create an email for when an admin has been allowed access to notify current admins with revoke privileges, and list which admin did the action	Milestone 2	Complete
Create an email for when an admin has been rejected or revoked access to be sent to the admin that lost access	Milestone 2	Complete
Create an email for when an admin's privileges have been changed that is sent to the changed admin	Milestone 2	Complete
Create an email for when an admin's privileges have been changed that is sent to all admins with privilege changing privileges, and note which admin made the change	Milestone 2	Complete

Create an email to notify user when his account password or other information has been changed by an admin	Milestone 3	Complete
Create an email as a receipt to the admin that changed user's account information	Milestone 3	Not Implemented
Create an email for when an admin changes another admin's user account information and send it to the changed user and all admin's with this privilege (for admins note which admin made the change)	Milestone 3	Not Implemented
Create an email template for emailing users from the email communication page for admins, also email the admin a copy of this email	Milestone 3	Complete
Create an email for when an admin deactivates or blocks a user that notifies that user	Milestone 3	Complete
Create an email to notify all admins with deactivate user privilege that a user was blocked by a specific admin	Milestone 3	Complete
Create an email for when an admin edits a user's application	Stretch Goal	Not Implemented
Create an email to notify all admins with edits a user application privilege of the action and which admin did it	Stretch Goal	Not Implemented

Obstacles

1. Unfamiliarity with tools and framework - The team struggled to get familiar with the many tools, languages, and frameworks. Through pair and mob programming the team was able to upskill the less experienced members of the team.
2. Dependency issues. The team experienced multiple dependency issues throughout the project. We were able to solve this by removing all of the node_modules, yarn.lock files, and the yarn global cache, which can be located by running `yarn cache dir`.
3. No in-person meetings - Due to the Coronavirus pandemic the team was not able to hold in-person meetings with the team or the sponsors. To overcome this the team regularly met on Zoom throughout the semester to accomplish our work. The team also met with the sponsors weekly to conduct a status review and show off what the team had accomplished.
4. Summer work - 3 of 4 members of the team had outside work commitments of 40hrs/week or more. This made it difficult to find time to meet together. The team overcame this by meeting late in the evening as well as on the weekends when all members are free.

Technical Dictionary

- JWT - JSON Web Token - A technology-agnostic standard for authentication between REST services.
- REST - Representational State Transfer - Stateless web service protocol using the HTTP standard. In this architectural pattern the caller of the REST methods is responsible for maintaining state, supplying all of the information the backend needs to operate.
- Selenium - Multi-language framework for running automated tests against a website in a browser. Can be used for web-scraping, integration testing, and regression testing.

Helpful Documentation

- Node.js docs - <https://nodejs.org/docs/latest-v12.x/api/>
- Express.js docs - <https://expressjs.com/en/4x/api.html>
- Jest.js docs - <https://jestjs.io/docs/en/getting-started>
- React.js docs - <https://reactjs.org/docs/react-api.html>
- Axios docs - <https://github.com/axios/axios>
- TypeORM docs - <https://typeorm.io/#/>
- SQL intro - <https://www.w3schools.com/sql/default.asp>
- MySQL docs - <https://dev.mysql.com/doc/refman/8.0/en/>
- Selenium Python API docs - <https://www.selenium.dev/selenium/docs/api/py/api.html>
- REST - https://en.wikipedia.org/wiki/Representational_state_transfer
- HTTP Code Explanations - https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
- Basic React Tutorial - <https://www.youtube.com/watch?v=sBws8MSXN7A>

Conclusion

Future work

1. User Application Editing Page

- a. Page for admins to edit user's applications
- b. Email Functionality:
 - 1. Email when admin edits a user's application.
 - 2. Email notifying all admins with "edit user application" privilege which admin user edited a user application.

2. More User Security Page Functionality

- a. The description of this page is in the "Admin User Security Page" section under "Implementation".
- b. Following email functions need to be implemented:
 - 1. When an admin with a certain privilege makes changes, Email all admins with the same privilege.
 - 2. ex) RootAdmin, Admin1, Admin2 has deactivate user privilege. When Admin1 deactivates "User1", RootAdmin, Admin2 should get the email informing the Admin1 has deactivated User1.
 - 3. This email functionality is implemented on all of the other admin pages except the "User Security Page".

3. Other Email Functionality

- a. Email to all Admins when an Admin changes a user's account information

4. Front End Design

- a. The front end stylings are not consistent
- b. The view of the page will be different over different display screens
- c. Admin portal has one stylesheet and uses that for all admin pages

5. Upgrade Testing Capabilities

- a. The unit testing for the application needs cleaning up. The test pass rate is roughly 50% for both the backend and the frontend.
- b. For Selenium testing the user portion of the website needs to be examined. The team discovered many outdated and failing selenium tests pertaining to legacy code.

6. Convert Old JS code to the TS Architecture

- a. Several frontend and backend components still use old jsx code that needs to be converted to Typescript.

- b. The architecture of these components also needs to be updated so that they match with the most up to date conventions for the application

7. Convert legacy Fetch API calls to use the client package

- a. Old frontend code still uses the Fetch API that is unauthenticated against the backend. These calls will fail if REQUIRE_JWT_VERIFICATION is set to true in the server .env file.
- b. These files can be located by running: `grep -rnwl --exclude-dir="node_modules" --exclude-dir="build" --include="*.js" --include="*.ts" --include="*.tsx" --include="*.jsx" -e 'fetch'`
- c. The list of these files is located in the repo at docs\2020-summer\legacy-fetch-calls.txt

Known Bugs

1. Authentication errors (JWT Verification)

Unhandled Rejection (Error): Request failed with status code 401: undefined

×

(anonymous function)

C:/Users/ggams/gitlab/AffordableServerOSU/client/dist/AffordableClient.js:87

```

84 |     return axios_1.default.post(endpoint, body, Object.assign(Object.assign({}, axiosParams), { headers: (_a = axiosParams === null || axiosParams === void 0 ? void 0 : axiosParams.headers) !== null && _a !== void 0 ? _a : this.getHeaders() }))).then((response) => {
85 |         return response.data;
86 |     }).catch((error) => {
> 87 |         throw new AffordableHttpError(error);
88 |     });
89 | }
90 | doPut(endpoint, body, axiosParams) {

```

View compiled

This screen is visible only in development. It will not appear if the app crashes in production.

Open your browser's developer console to further inspect this error. Click the 'X' or hit ESC to dismiss this message.

- a. JWT verification is buggy. This error will not happen if you put “REQUIRE_JWT_VERIFICATION=false” in the .env file under server. More details on “Future Work #7”.
- b. This error will show up randomly when the user is trying to execute some function that is related to the Affordable Client while the application is running.
- c. This is not likely to happen most of the time, but when it happens press “x” on the right top corner or refresh the page and try again. If refreshing doesn’t work, restart the app.

- d. Most of the functions used in the admin portal use the Affordable Admin Client, so very unlikely this will happen, but keep in mind, this can happen anytime.

2. Two Factor functions

admin2	admin2 Reset Two Factor Authentication of user1	7/2/2020 : 16:59
admin2	admin2 Reset Two Factor Authentication of user1	7/2/2020 : 17:12

- a. The methods related to Two Factor Authentication sometimes get run twice. This does not cause the application to crash or any errors like 401 error. The method just gets executed twice.
- b. The User Security Page in the admin portal uses the resetTwoFactor method from the Affordable Client. This method has an emailing service that sends the user a link to the front end page for the user to reset their password. Another method. This method also has a recordAuditTrail method to record audit trails.
- c. Above Screenshot shows that somehow resetTwoFactor method getting called again after about 12 mins when all admin did was to click the "Reset" button once to reset the user's MFA.
- d. resetTwoFactor method is in the Affordable Client. Similar to authentication error above, this might be a problem with how an Affordable Client is implemented.