Tyler Hayes

IS601

Spring 2024 (Online)

Final Project

# **Foreword**

* My repository is available at the following link: [Github](https://github.com/TyHys/user_management)
* My Dockerhub image is available at the following link: [Dockerhub](https://hub.docker.com/repository/docker/tewq01/wis_club_api/general)

**Introduction**

Throughout this project I learned a ton about debugging and troubleshooting applications. More importantly, I feel like I learned a lot more about test driven development. I'm still in the habit of writing something and getting far too from writing tests before I finally realize I have to go back and write them. During this project I tried to keep tests relatively close instead of leaving them until the end.

I learned a lot about careful reading of documentation as well as the resilience needed to see something complex through to its end. I was unfamiliar with APIs before the Homework 9 a few weeks ago, so I've been doing everything I can to familiarize myself with how they work via this project. While I feel like I did struggle a lot in some areas since I lack a computer science background, I do feel like I learned a great deal not only from this project but the class.

## **Issues**

* [/myaccount Endpoint returns null values for populated database info #8](https://github.com/TyHys/user_management/issues/8)
* [/register/ Endpoint presents incomplete data #6](https://github.com/TyHys/user_management/issues/6)
* [Admin User Erroneously Converts to "Authenticated" #4](https://github.com/TyHys/user_management/issues/4)
* [Verification Link Error #2](https://github.com/TyHys/user_management/issues/2)
* [LibC Vulnerability - Preventing Docker Deployment #1](https://github.com/TyHys/user_management/issues/1)

## **Tests**

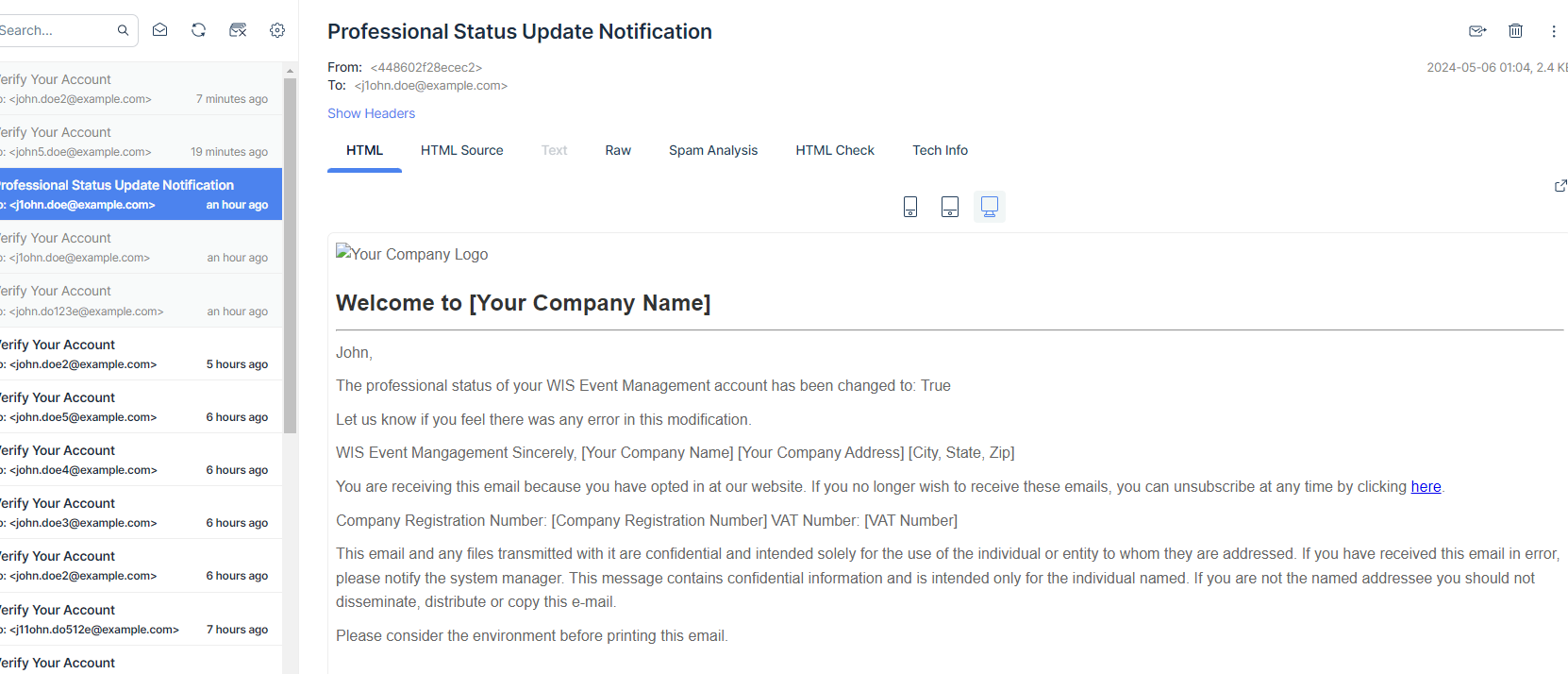
* [Two tests were developed for the event schemas to test some edge cases.](https://github.com/TyHys/user_management/commit/a0142b00ea526b903b5040dd7526e634178b6397#diff-290282138412a840643c110987874d7b4c32259148ea54e8ca419ccacbcb6527)
* [Three more tests were developed for testing the change is-professional endpoint.](https://github.com/TyHys/user_management/commit/9111cc08c4f6f98a8a476016c59d39b9ffc245a6#diff-1d7b620623536f4fb4d120c0d59e8a3ad15f248d9f7af6aa9aa8965cfe923ef9)
* [Five more tests were later written in the event schema and user service tests.](https://github.com/TyHys/user_management/commit/98bf6e8d049ab4c09b2a2f5d21527a541ab6452b)
* (There are other tests, such as for the Self-Profile Update, I just listed ten above since that was the project guidelines.)

## **Feature**

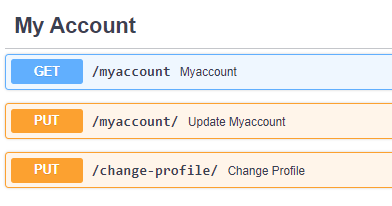
For the additional feature, I chose User Profile Management:

Description: Enhance the user profile management functionality to allow users to update their profile fields and enable managers and admins to upgrade users to professional status.

I successfully added a new endpoint so that admins and managers can update the professional status of any account that has been registered. Upon modification, the user will receive an email notification to alert them of the change. Here you can see where a user was successfully notified via email of their email, received after the status was updated (Link to the commit):



I also added an endpoint so that regular users can update their profile. This will allow self-service so that an individual can update their profile directly through a singular endpoint designed for the purpose.



## **Developer Notebook**

* Composed the docker image and navigated to PGAdmin, got Postgres setup per instructions. No issues.
* Ran the Alembic migration script to load the tables/schemas in Postgres, no issues.
* Navigated to http://localhost/docs & registered a new user successfully.
  + It took me a long time to realize the admin authentication method from Homework 10 no longer worked and I burned a few days here before asking others in the Discord.
  + After this I realized that I needed to begin by registering a user when I have no authentication.
* Authenticated the user via the email being the 'user id' (I'm not proud of how long it took me to realize that the user ID was the email and not the UUID).
* Executed the Get User endpoint successfully, returned the correct data.
* Spent several days (intermittently) trying to figure out why an email was not received in MailTrap.
  + Realized that an email isn't sent for the initial admin user. (ouch)
* Registered a second user, successfully received email into MailTrap.
* Link didn't appear to be valid. Modified the user\_service.py to correct for invalid token in link.
* Clicked the link within the email, and logged in successfully.
* Queried the users table & saw that the email\_verified bool was flipped to TRUE for the user.
* Admin user was changed to authenticated in Postgres
* Added check to see that user is ANONYMOUS in user\_service.py as well.
* Noticed that registering a user doesn't actually show you the full details in the response body.
* Modified the response model in user\_management/app/routers/user\_routes.py to correct for this.
* Noticed a similar issue in the response model for the /myaccount endpoint.
  + Corrected this, similar to the previous issue.
* Added a route to change the professional status of an account.
* Added an email template and successfully tested the automatic sending.
* Wrote new tests for the additional endpoint
* Made a new endpoint so that users can update their own profile based on the existing ones for updating information.
* Wrote additional tests to satisfy new feature boundaries.
* Published completed work with notebook.