Swamp Services: Group 11

University of Florida Volunteer Service Website

https://github.com/zachj112/CEN3031-Project
https://youtu.be/Bg9BTL_2iPg?feature=shared

Mihir Patel, Robert Culley, Kathy Matos, and Zach Jasko

Table of Contents

Section 1

Project Description

Group 11's project was to create a student volunteer web application for the students of the University of Florida. Specifically, this project titled "Swamp Services" helps address the myriad of struggles that college students face during their college journey. The service was intended to be hosted by the University of Florida student government or other administrative organization as a way to provide a central place for students to earn service hours, engage in social activities, and help their community while earning gator points. By restricting account registration to only those who have a UF email, this service ensures safety and accountability for those who make use of it. This product differs from other service board websites like Craigslist by providing free help and a safer volunteer base.

Challenge Statement and Solution

Challenge

One common problem for college students, especially first year and first generation, is managing new challenges and environments. This can be stressful for many and harm their ability to learn and get the most out of their time at college. Some common challenges that college students might face include food insecurity, financial troubles, loneliness, time management, and dealing with illness/disability.

Solution

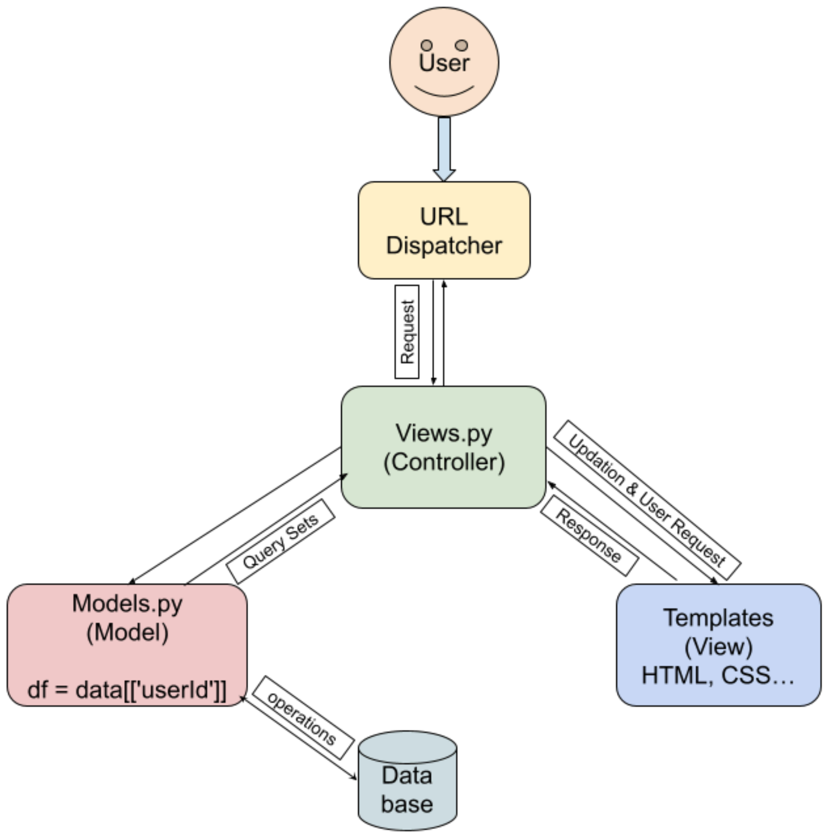
To help mitigate this problem, group 11 has created a student volunteer community website centered around helping their peers. College students are best able to help because they may have gone through similar problems and be able to emphasize and lend their experience. To create a realistic scope for our project, group 11 focused on the University of Florida community and tailored our website accordingly. Students will be able to earn volunteer hours and help their community by providing quick services such as creating meal plans, looking over resumes, taking students to the campus food bank, and teaching time management skills.

Features and Functionality

- Registration and Login Functionality
  - Registration requires a UF email and username.
  - Passwords are required to be at least 8 characters and not include letters from the user's email.
- Profile Settings Features
  - Changing email address.
  - Setting a profile picture.
  - Short biography section.
- Gator Point Leader Board Page
  - Provides an incentive for students to help others.
  - Shows the top 10 students who have helped the most people.
- Forum Home Page
  - Shows various forum topics including the latest.
  - Has a separate table to show the latest forum topic and post.
  - Search functionality to find a specific post by account, category, or title.
- Topic Discussion Page
  - Allows for other students to comment on a post to show interest in providing volunteer help or advice.
  - Student Government admin can hide/close issues once a volunteer has been found.
- Topic Creation Page
  - Allows for selection of the category of help needed. Some categories include food donation and pet services.
  - Assignment of gator points based on the amount of time the service should take.
  - Inclusion of a service start and end date.
  - A title and description field to give details on what is being requested.
- Database Separation for better security and risk management
- Moderation and admin tools to ensure the website is a healthy and productive community space.
  - The admin account can also be used to remove old data and accounts after a certain period of time for better risk management.

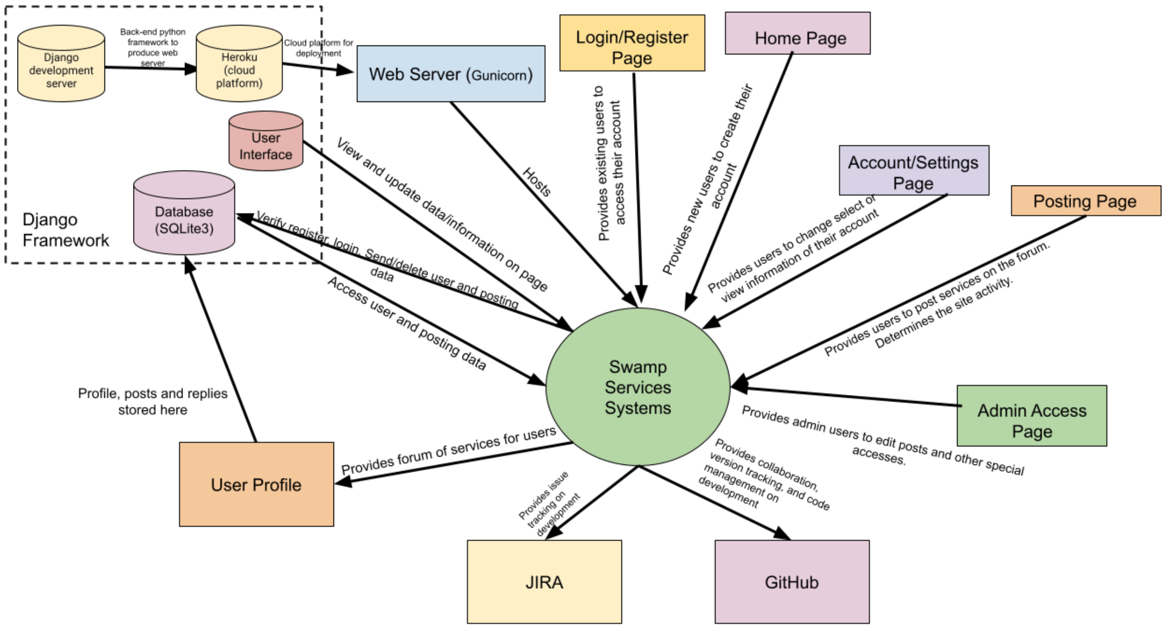
System Models

Architectural Pattern



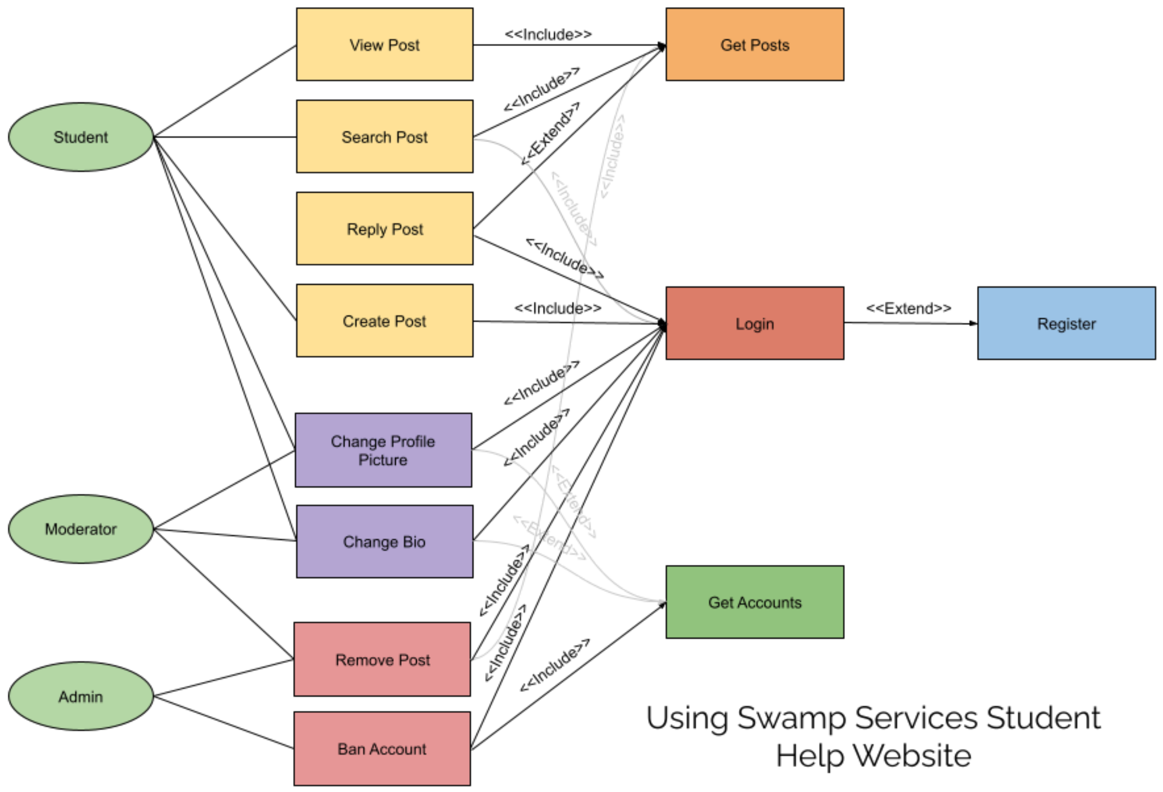
The pattern group 11 decided on was the Model-View-Controller (MVC) architecture. The reasoning for this choice comes from the fact that this pattern follows how the actual World Wide Web works today (Sheldon, Robert). For example, consider a user of an e-commerce website who wants to browse the e-market and add an item to their cart at the click of a button. The MVC architecture easily models this using its three components. The clicked button is sent as an input event to the controller component. The controller then sends the added item data to the model component to add to the list of items in the user's cart. The model reports all the items in the cart back to the controller, which then sends the updated cart data to the view component to be rendered. In this case, the view will update the image showing the current contents of the user's cart. Finally, the rendered web page is sent back to the controller to be presented to the user.

System Context Model


Django development server
Back-end python framework to produce web server
Heroku (cloud platform)
Cloud platform for deployment
Web Server (Gunicorn)
Login/Register Page
Home Page
User Interface
Database (SQLite3)
Django Framework
Hosts
View and update data/information on page
Verify register, login. Send/delete user and posting data
Access user and posting data
Provides existing users to access their account
Provides new users to create their account
Account/Settings Page
Provides users to change select or view information of their account
Posting Page
Provides users to post services on the forum. Determines the site activity.
Swamp Services Systems
Admin Access Page
Provides admin users to edit posts and other special accesses.
Profile, posts and replies stored here
User Profile
Provides forum of services for users
Provides issue tracking on development
Provides collaboration, version tracking, and code management on development
JIRA
GitHub

A systems context module is "a diagram that defines the boundary between the system, or part of a system and its environment, showing the entities that interact with it" (Sanethia, Thomas (a)). One important concept to note here is that the Swamp Services Systems component acts as the control center or the brain of our application. Inputs consist of user actions that the system then processes and sends out instructions to the other components of what needs to be done to complete this action. The system always returns a message back to whichever component the action came from to tell if it was successful or not.

Use Case Model



A use case model is a "list of actions or event steps typically defining the interactions between a role as an actor and a system to achieve a goal" (Sanethia, Thomas (b)). In the diagram above, the vertical lines represent a page in our website and the horizontal lines represent methods that the user calls (typically by hitting a button or entering text into a textbox and the resulting outcome. The extends relationship is conditional and only triggers on certain input. The includes relationship is something that always needs to be done for another given action.

Section 2

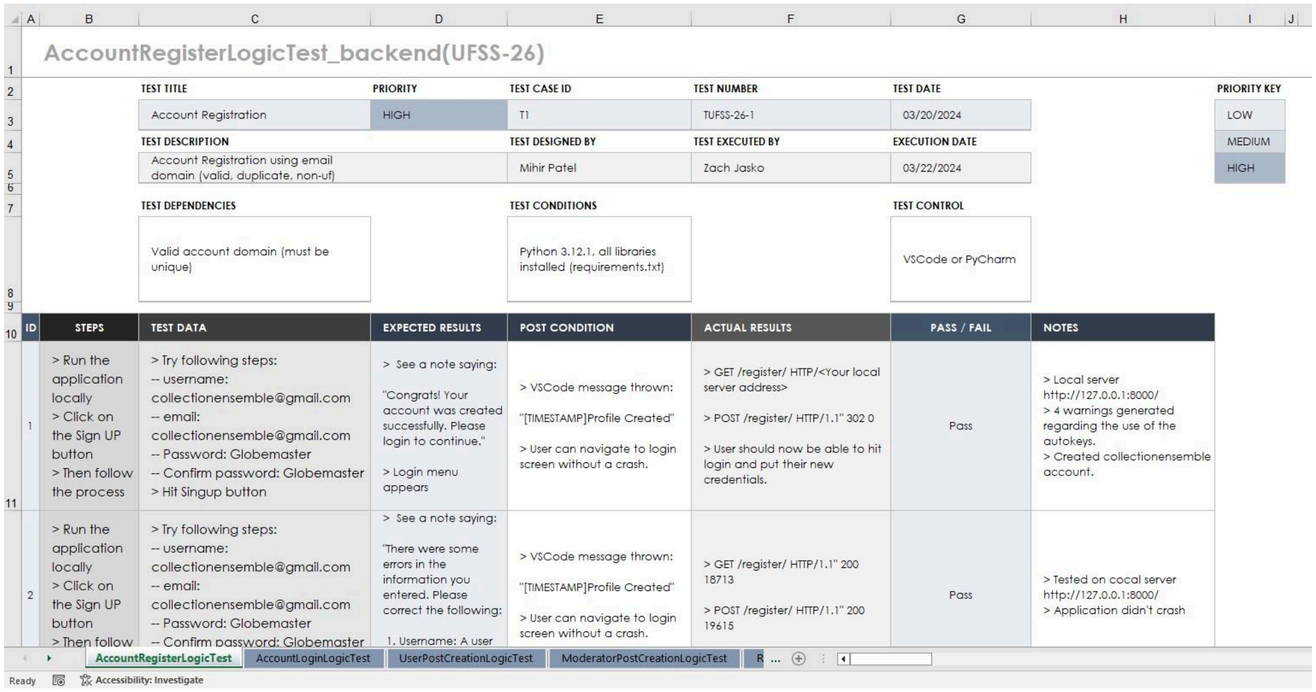
Code Management

The main tools that were used to manage this project's workload were Github, Slack, Zoom and Jira. Group 11 used Github as version control as new features were added to preserve prior semi-functional products and find when bugs were introduced. The implementation of branch rules was used to ensure the main branch always contained a semi-functional product that could not be pushed to before a pull request was reviewed. Additionally, tagging and the inclusion of quality software attributes like separation of responsibility and testability were checked before merging. Pull request rules were also set up that required code review, tagging and ensured the code is of good quality before merging to the main branch. Slack was used to coordinate changes with others and ask development questions. Slack was also used to schedule zoom meetings for more difficult problems and to review submissions and PRs. Lastly, Jira helped us split large issues into smaller user stories that could then be assigned to different team members. This helped create a clear separation of responsibilities and track whether the group was ahead of or behind schedule.

Test Plan

For assurance of the good software qualities like testability, functionality and reliability of the Swamp Services application, the group developed a comprehensive testing template using Excel for individual component testing. Even though simple, this template provides a structured approach to testing each component, from user registration, post creation, commenting functionality, forum management, etc. Following this template allowed for the testing of edge cases of each component thoroughly, identifying bugs, and correcting them before integration. To partially automate testing and save time, Django's inbuilt testing functionality was used. The group decided against using CircleCI for testing as Django had a better integrated alternative and the time researching and setting up CircleCI would have taken away from development time. Below, the recorded output for one such test is given as an example

Test Example



AccountRegisterLogicTest_backend(UFSS-26)

TEST TITLE	PRIORITY	TEST CASE ID	TEST NUMBER	TEST DATE
Account Registration	HIGH	T1	TUFSS-26-1	03/20/2024

TEST DESCRIPTION	TEST DESIGNED BY	TEST EXECUTED BY	EXECUTION DATE
Account Registration using email domain (valid, duplicate, non-uf)	Mihir Patel	Zach Jasko	03/22/2024

PRIORITY KEY
LOW
MEDIUM
HIGH

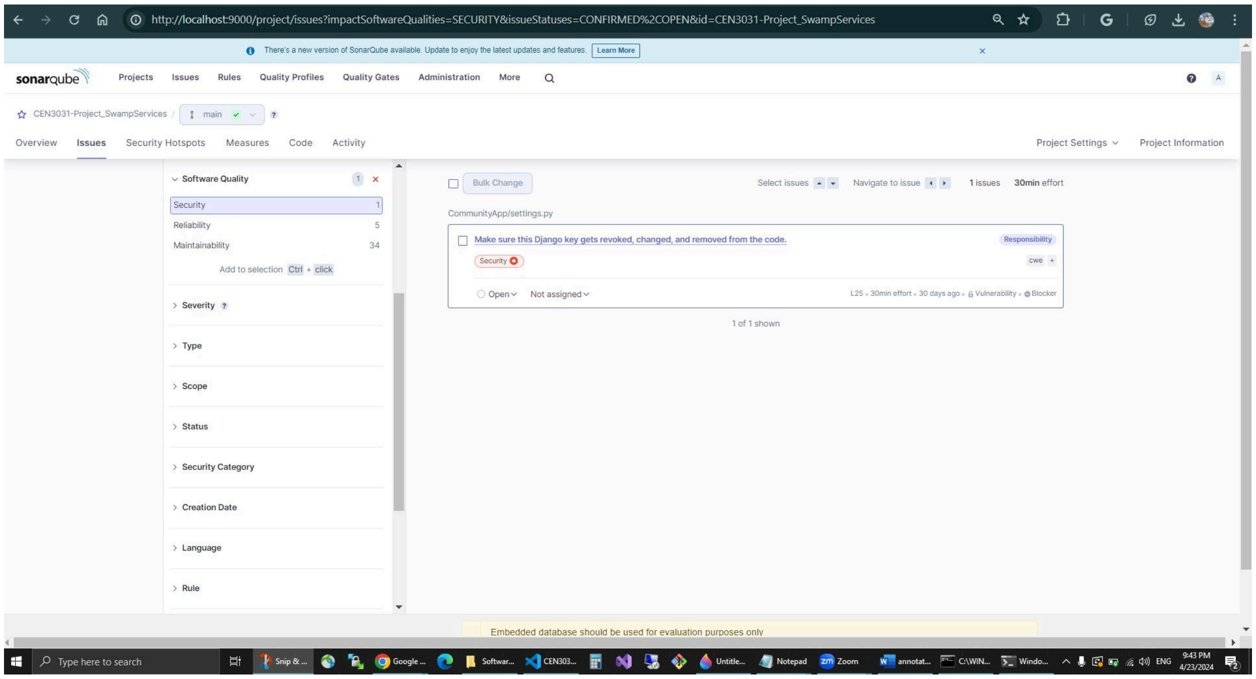
TEST DEPENDENCIES	TEST CONDITIONS	TEST CONTROL
Valid account domain (must be unique)	Python 3.12.1, all libraries installed (requirements.txt)	VSCode or PyCharm

ID	STEPS	TEST DATA	EXPECTED RESULTS	POST CONDITION	ACTUAL RESULTS	PASS / FAIL	NOTES
1	> Run the application locally > Click on the Sign UP button > Then follow the process	> Try following steps: -- username: collectionensemble@gmail.com -- email: collectionensemble@gmail.com -- Password: Globemaster -- Confirm password: Globemaster > Hit Singup button	> See a note saying: "Congrats! Your account was created successfully. Please login to continue." > Login menu appears	> VSCode message thrown: "[TIMESTAMP]Profile Created" > User can navigate to login screen without a crash.	> GET /register/ HTTP/<Your local server address> > POST /register/ HTTP/1.1" 302 0 > User should now be able to hit login and put their new credentials.	Pass	> Local server http://127.0.0.1:8000/ > 4 warnings generated regarding the use of the autokeys. > Created collectionensemble account.
2	> Run the application locally > Click on the Sign UP button > Then follow	> Try following steps: -- username: collectionensemble@gmail.com -- email: collectionensemble@gmail.com -- Password: Globemaster -- Confirm password: Globemaster	> See a note saying: "There were some errors in the information you entered. Please correct the following: 1. Username: A user	> VSCode message thrown: "[TIMESTAMP]Profile Created" > User can navigate to login screen without a crash.	> GET /register/ HTTP/1.1" 200 18713 > POST /register/ HTTP/1.1" 200 19615	Pass	> Tested on cocal server http://127.0.0.1:8000/ > Application didn't crash

AccountRegisterLogicTest
AccountLoginLogicTest
UserPostCreationLogicTest
ModeratorPostCreationLogicTest
R ...
Ready
Accessibility: Investigate

Static Code Analysis Report

Group 11 used SonarQube to analyze the Swamp Services project for the code quality and catching the bugs, code smells, and vulnerabilities. The analysis revealed that for the breadth of our project, the number of bugs, code smells, and vulnerabilities detected is on the lower side, indicating a relatively clean and maintainable codebase. SonarQube tagged some issues as false positives, which group 11 carefully reviewed and addressed as necessary. All in all, the SonarQube analysis has given valuable insights to our code's quality and helped us further improve the robustness and reliability of the project.



There's a new version of SonarQube available. Update to enjoy the latest updates and features. Learn More

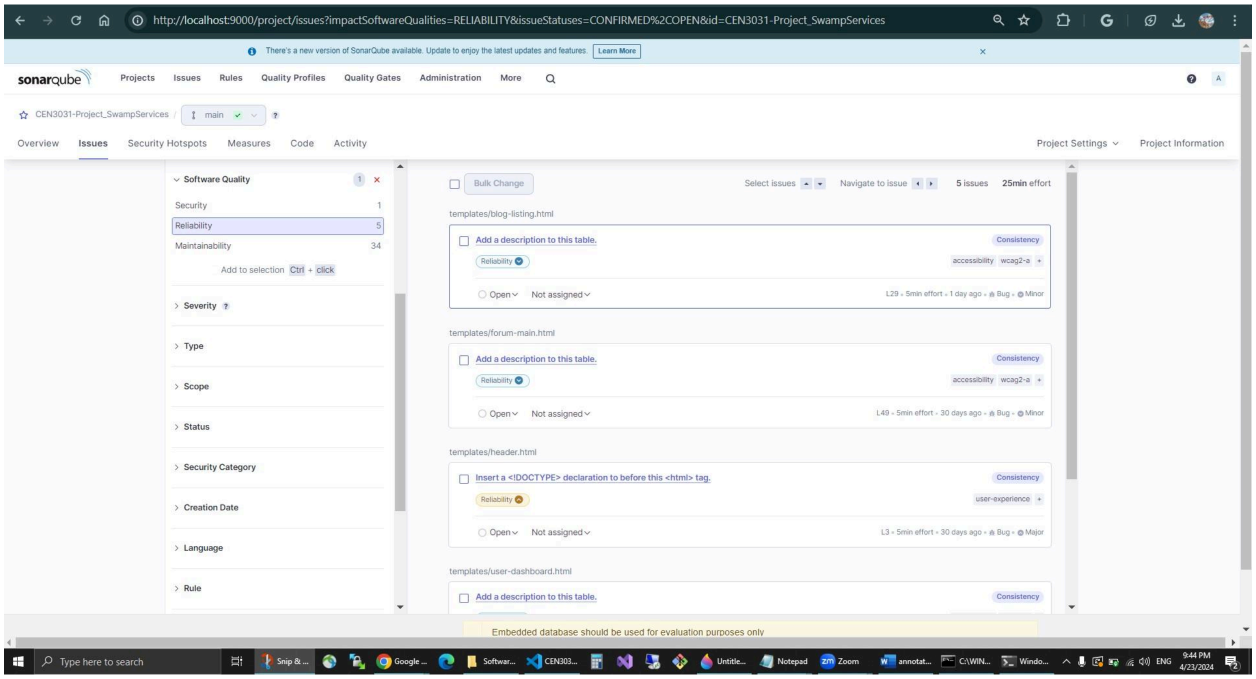
CommunityApp/settings.py

Make sure this Django key gets revoked, changed, and removed from the code.

L25 · 30min effort · 30 days ago · Vulnerability · Blocker

Embedded database should be used for evaluation purposes only

Security in software quality is protection from unauthorized access, use, or destruction of software. It ensures that the software is protected from threats. We have made a dummy hashed Postgre key. This allows new users to start the application without the need to set up a separate PostgreSQL server. Given the scope of this project, the team believes they can safely ignore the security flag for now.


http://localhost:9000/project/issues?impactSoftwareQualities=RELIABILITY&issueStatuses=CONFIRMED%2COPEN&id=CEN3031-Project_SwampServices

There's a new version of SonarQube available. Update to enjoy the latest updates and features. Learn More

CEN3031-Project_SwampServices / main

Software Quality: Security 1, Reliability 5, Maintainability 34

5 issues 25min effort

templates/blog-listing.html
Add a description to this table. — Consistency — Reliability — accessibility, wcag2-a — Open, Not assigned — L29 · 5min effort · 1 day ago · Bug · Minor

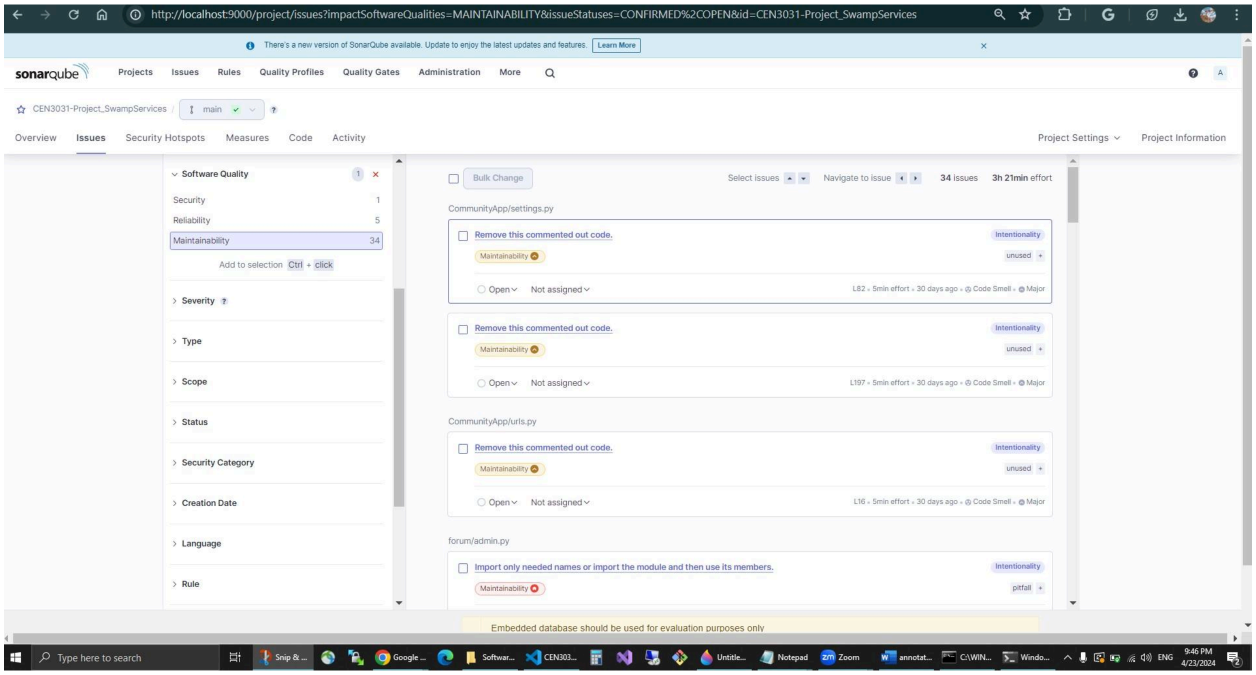
templates/forum-main.html
Add a description to this table. — Consistency — Reliability — accessibility, wcag2-a — Open, Not assigned — L49 · 5min effort · 30 days ago · Bug · Minor

templates/header.html
Insert a <!DOCTYPE> declaration to before this <html> tag. — Consistency — Reliability — user-experience — Open, Not assigned — L3 · 5min effort · 30 days ago · Bug · Major

templates/user-dashboard.html
Add a description to this table. — Consistency

Embedded database should be used for evaluation purposes only

Reliability measures how well the software can maintain its performance under deployment. SonarQube reported consistency errors in our HTML document revolving around naming tables and other elements. This ideally should be added in when the code is refactored but the names should be descriptive enough for both what each table does for most purposes.


http://localhost:9000/project/issues?impactSoftwareQualities=MAINTAINABILITY&issueStatuses=CONFIRMED%2COPEN&id=CEN3031-Project_SwampServices
There's a new version of SonarQube available. Update to enjoy the latest updates and features. Learn More
sonarqube
Projects Issues Rules Quality Profiles Quality Gates Administration More
CEN3031-Project_SwampServices / main
Overview Issues Security Hotspots Measures Code Activity
Project Settings Project Information
Software Quality 1
Security 1
Reliability 5
Maintainability 34
Add to selection Ctrl + click
Severity
Type
Scope
Status
Security Category
Creation Date
Language
Rule
Bulk Change
Select issues
Navigate to issue
34 issues 3h 21min effort

CommunityApp/settings.py

Remove this commented out code. — Intentionality — Maintainability — unused — Open — Not assigned — L82 · 5min effort · 30 days ago · Code Smell · Major

Remove this commented out code. — Intentionality — Maintainability — unused — Open — Not assigned — L197 · 5min effort · 30 days ago · Code Smell · Major

CommunityApp/urls.py

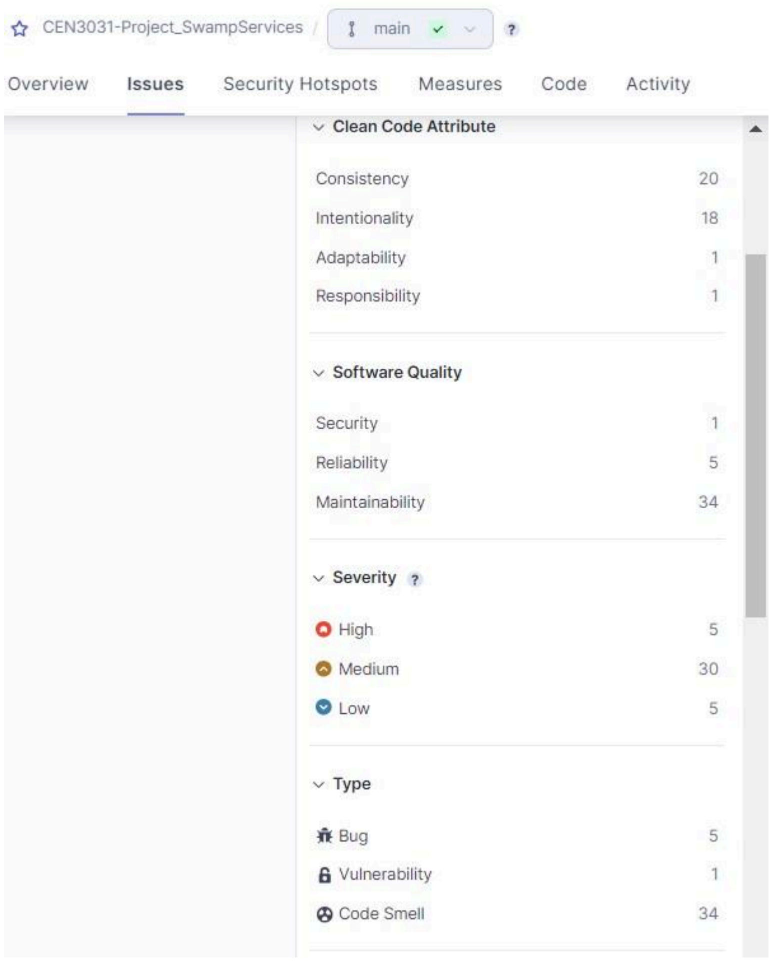
Remove this commented out code. — Intentionality — Maintainability — unused — Open — Not assigned — L16 · 5min effort · 30 days ago · Code Smell · Major

forum/admin.py

Import only needed names or import the module and then use its members. — Intentionality — Maintainability — pitfall

Embedded database should be used for evaluation purposes only

Maintainability describes how easy it is to fix, improve, or understand software code. Some people have commented out minor functionality of the code, which can be used for implementing new features, which is why it was left in. Import statements for this code were also left in, even if they aren't being actively used right now. On our final merge, we are going to take care of these issues.


CEN3031-Project_SwampServices / main

Clean Code Attribute

Consistency	20
Intentionality	18
Adaptability	1
Responsibility	1

Software Quality

Security	1
Reliability	5
Maintainability	34

Severity

High	5
Medium	30
Low	5

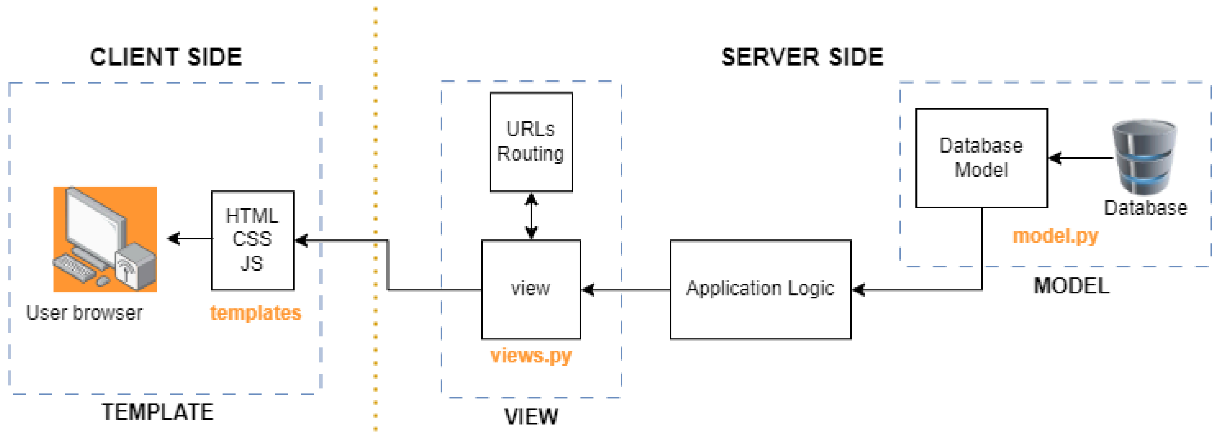
Type

Bug	5
Vulnerability	1
Code Smell	34

While there were some problems, the vast majority of these can be fixed when refactoring. Everything of major consequence was fixed in the project and the rest were left for refactoring while the project report and presentation were finished.

Section 3

Technical Details



Project Overview

- Objective: Create a community service website for UF students to help each other with various challenges.
- Target Users: UF students in need of assistance and those willing to volunteer.
- Service Offered: Meal planning, cooking lessons, pet walking, emotional support, resume/job search help.
- Unique Feature:
    - Gator Point rewards for volunteers.
    - Free service unlike platforms like Craigslist.
    - Focus on building a strong community within UF.

Technical Details

- Framework: Django (version 3.2.10)
- Database: PostgreSQL (suggested 15 or up) or SQLite (default)
- Dependencies: Several Python libraries like Django-baton, gunicorn, psycopg2, etc. (refer to requirements.txt)
- Project Setup:
    - Requires virtual environment creation and installation of dependencies.
    - Database configuration and migrations are necessary.
    - Instructions for both Windows and Mac are provided in README.md.

- Branch Protection Rules: The branch protection rules on the main branch enforces code quality and collaborative development practices.

Frontend & Features

- Templates: base.html, forum-main.html, search-result.html, blog-detail.html, user-dashboard.html, user-post.html, blog-listing.html, topic-detail.html, accept_job_posting.html, header.html, footer.html, info.txt, profile.html, login.html, register.html
- Search functionality: Users can search for topics by keywords.
- User Dashboard: Volunteers can track their posted topics, answers, and earn Gator Points.
- Leaderboard: Shows top 10 students with the most Gator Points.
- Posting Questions/Answers:
    - Users can create new topics specifying category, Gator Point reward, and service dates.
    - Users can answer existing topics with detailed responses.
- Moderation: Student Government members have additional privileges to open/close posts.
- Accepting Answers: Topic creators can accept an answer as the solution.
- Authentication: Registration, login, logout, password reset functionalities are implemented.

Backend & Features

- Admin:
    - Creation and deletion of posts.
    - Creation and deletion of users.
    - Accessible by adding '/admin' at the end of the local server url

- registration/*:
    - forms.py, models.py, urls.py, views.py components handle new account creation and credential verification respectively. Messages are returned specifying whether registration or login was successful or not.
- forums/*:
    - forms.py, models.py, urls.py, views.py components display all posts currently in the website's database while post allows for adding new entries into the database. The Swamp Service System is used to get and add posts every time these components are loaded or refreshed on the website so they are always up to date.
- forums/models.py:
    - The profile system allows for changing account settings. The Swamp Service System gets this input, updates the accounts database and then gives instructions for the profile page to refresh and display the new changes.
- forums/models.py and registration/models.py:

    - To protect the data needed by the website to operate, only the Admin and Swamp Service System component can get, set, and delete from this component. The Swamp Service System is limited to only allowing for specific changes to be made depending on the component that requested the change.
    - Moderator and Admin:
        - These are pages that accounts with the required permissions are able to access. They are used to directly interact with a database UI to moderate user accounts and make website changes respectively.

Installation Instructions

These instructions are shown for Windows, however, they should be the same across other operating systems:

Prerequisites

- Python installed on your system (we are using 3.12).
- PostgreSQL installed and running on your system (ideally Postgre 15 or above).
- Make sure python and postgres are added to PATH.

Steps

- Open the terminal, clone the repository using git clone. Then, change to the project directory.

- Install a virtual environment
  ```
  > pip install virtualenv
  ```

- Setup a virtual environment
  ```
  > python -m venv myenv
  ```

- Activate the Virtual Environment
  ```
  > myenv\Scripts\activate
  ```

- Install Django
  ```
  > pip install django
  ```

- Install psycopg2 (PostgreSQL Adapter)
  ```
  > pip install psycopg2-binary
  ```

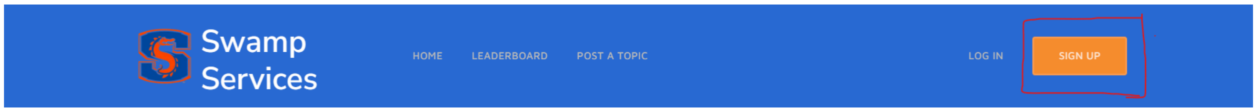
- Install other packages to run the project
  ```
  > pip install -r requirements.txt
  ```

- Initialize Django Database
  - ❖ Create migrations:
    ```
    > python manage.py makemigrations
    ```
  - ❖ Run migrations:
    ```
    > python manage.py migrate
    ```
- Run the Django Development Server
  ```
  > python manage.py runserver
  ```
- You can access it at http://127.0.0.1:8000/. (link is subject to port availability, so it may change on your terminal).
- Exit the server by hitting ctrl+c. And type deactivate to exit the environment.

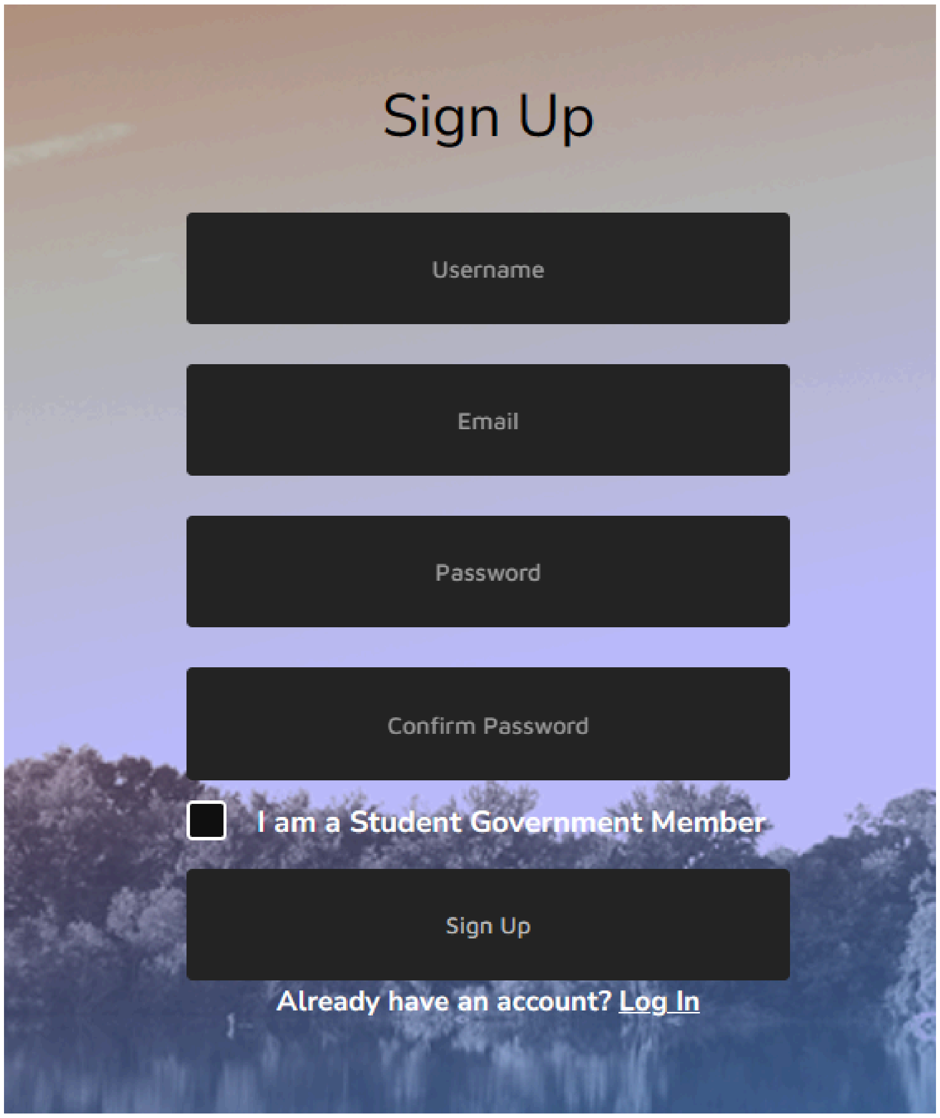
❖ Run following commands in the Mac:
  - ➢ pip3 install virtualenv
  - ➢ python3 -m venv <your_env_name>
  - ➢ source <your_env_name>/bin/activate
  - ➢ pip3 install -r requirements.txt
  - ➢ python3 manage.py makemigrations
  - ➢ python3 manage.py migrate
  - ➢ python3 manage.py runserver
  - ➢ Copy the *http://...* link in your browser and run.
  - ➢ When done running. Close the environment: deactivate

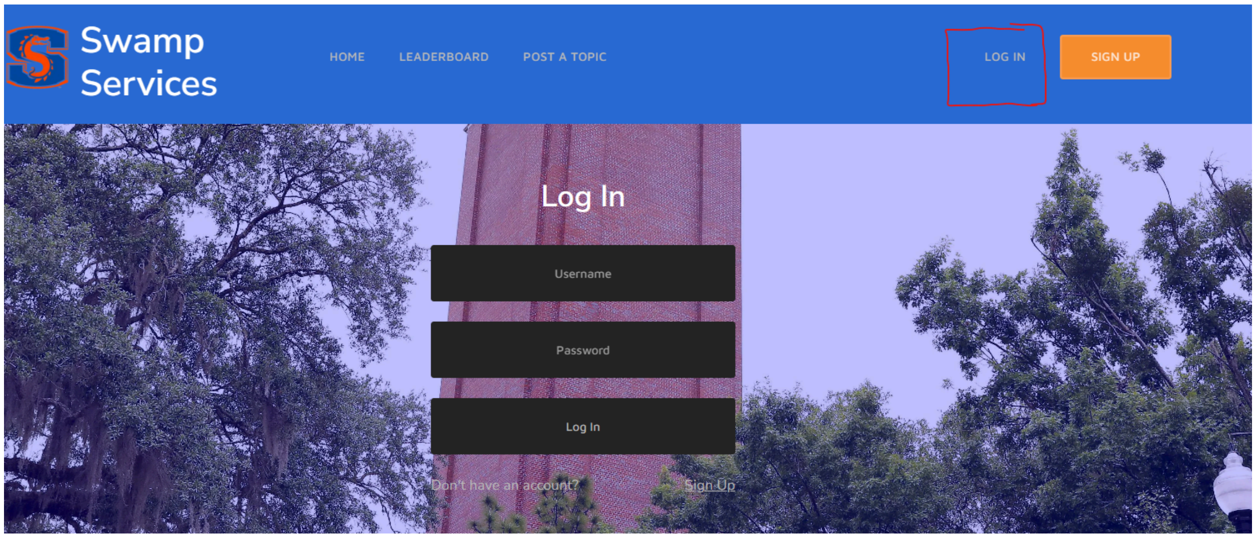
Login and Access Credentials/API Keys

We don't have any backend APIs or login access made specifically for the project. Users can create a regular user account after they have started the application on a local server. Please follow these steps:



*** **Note** *** Check the “I am a Student Government Member” box to get moderator access.


Sign Up
Username
Email
Password
Confirm Password
I am a Student Government Member
Sign Up
Already have an account? Log In


Swamp Services
HOME
LEADERBOARD
POST A TOPIC
LOG IN
SIGN UP
Log In
Username
Password
Log In
Don't have an account?
Sign Up

- If need to create an admin account, they can do that by:
    - Open the powershell/terminal on your machine.
    - Navigate to your Django project directory where the manage.py file is located.
    - Run the following command to create a superuser:
      > python manage.py createsuperuser

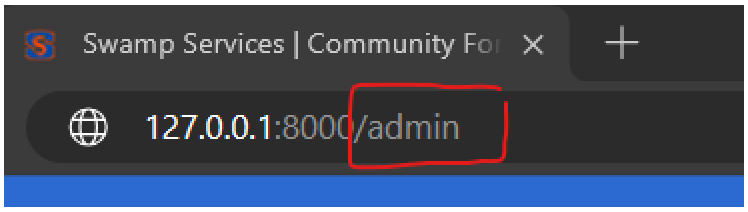
- The command will prompt you to enter the following information:
    - Username: Enter a username of your choice.
    - Email address: Enter a valid email address.
    - Password: Enter a secure password. You will be prompted to enter the password again for confirmation.

- Once you have entered all the required information correctly, the superuser will be created successfully.

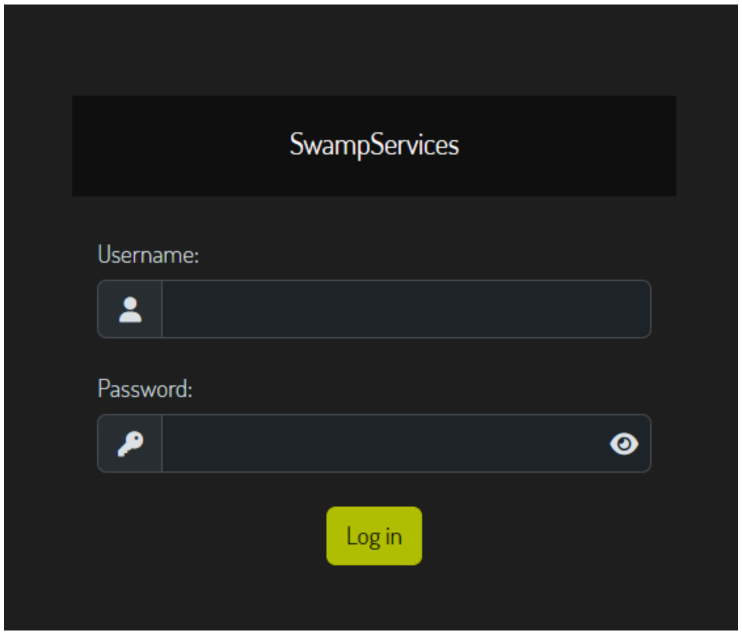
- To verify that the superuser was created, you can start the development server by running:

  > python manage.py runserver

- Open a web browser and go to the Django admin site, typically located at http://127.0.0.1:8000/admin/.



- Use the superuser credentials you just created to log in to the admin interface



Section 4

Risk Management Plan

Because this web application is used to connect people and allows for users to give out personal information like contact info to get in touch with other students for volunteer work, there are a few safety concerns that need to be addressed. One is bad actors getting access to the website and trying to harvest data or cause harm in other more serious ways. One strategy that we use to reduce the chance of this happening is by requiring a valid UFL email domain to make an account. Without this, you are unable to interact with the website. Another way we try to reduce the risk of this happening is by giving moderator accounts to trusted students (like student government or people who have a history of doing good volunteer service work) to report and remove accidently posted personal info. Additionally, because UF emails are associated with the website's account, this safe and secure service can be leveraged to share contact information and other needed personal info. The final way that we manage risk is by allowing the admin account to remove personal info periodically from the website after a volunteer opportunity has concluded so if a data breach does happen, the damage it can do is mitigated.

Software Quality Attributes

When designing and implementing our project, group 11 tried to follow best software practices for better software quality attributes. One of the primary attributes that was followed was that of reusability. By separating and organizing code by their function, it is easy to reuse one of the components of this project in a future project. For example, the models.py file contains all information about data that the web application uses and parts like the account model can be reused in future websites. Furthermore, by separating our code based on its function all in one place, it is now much easier to test specific functions which give our project good testability. Another software attribute that was closely followed was that of compatibility. By using Python, our project is platform-independent and can be used on many of the most popular operating systems including macOS, Windows, and Linux (Python Foundation). Additionally, python can be built on any platform which makes it easy to run and get started. In group 11's project, the use of a requirements text document was used to make it easy for those wishing to modify the project to quickly install all needed dependencies. Some other attributes that were followed was that of maintainability and functionality. The use of Django, allows for the Swamp Services web application to run even when certain features go down. For instance, if the ability to change your profile picture on the website encounters a bug, it won't crash the entire application but rather just display an error. These error messages are also very helpful in maintaining the website because they provide granular detail into what exactly went wrong with a stack trace and broken url.

References

Python Foundation. “What’s New in Python.” *Python Documentation*, Python Foundation, docs.python.org/3/whatsnew/index.html. Accessed 23 Apr. 2024.

Thomas, Sanethia (a). “System Models.” *Module 6 Week 1 CEN3031 Slide Deck*, University of Florida, ufl.instructure.com/courses/498788/pages/module-6?module_item_id=10778737.

Thomas, Sanethia (b). “Use Case Models.” *Module 13 Week 2 CEN3031 Slide Deck*, University of Florida ufl.instructure.com/courses/498788/pages/module-13?module_item_id=10778777.

Sheldon, Robert. “What Is Model-View-Controller (MVC)?: Definition from TechTarget.” *WhatIs*, TechTarget, 12 Sep. 2023, www.techtarget.com/whatis/definition/model-view-controller-MVC.