CS437 ASSIGNMENT-1 REPORT

Group: #22
Project Number: 5

Barış Ulaş Çukur 29461
İbrahim Ege Oral 29299
Ömer Can Öztürk 29248

Video Link:
https://drive.google.com/file/d/1GxYf_E4EPWslWk1dEaOA6lE3Ofp38hm9/view?usp=sharing

Task
In this assignment, we have created a honeypot server running on Flask and python for an Azerbaijani news agency. In this honeypot, our aim is to track and log all user activities through kernel-level input tracking and auditing of the visited pages. The application will log a console output whenever a specific endpoint present in our project is called.

Before we dive into the explanation of the code, let's talk about the task and how our thought process was before starting.

Our Decoy:
We were asked to craft an authentic-looking Azerbaijan news website that not only showcases the recent news sourced from a legit news platform but also incorporates dynamic content mechanisms to baffle the potential attackers. While leveraging the RSS feeds from the Azerbaijani news agency, we made sure that our website offered a neat experience for users, allowing them to explore the diverse news topic, hence not an obvious website regarding knowing whether it's fake or not. The dynamism in our content delivery was to ensure that the attackers can't easily distinguish our honeypot from static content.

The Vulnerabilities:

The vulnerabilities of this system occur from server-side component issues. As stated in the homework document, the task was to find outdated components such that they can be exploited to gain access to the system. The server has three vulnerabilities from the following packages: Copyparty, fonttools, and xml2xlss. The older version of the copyparty component introduces an arbitrary file access vulnerability while fonttools and xml2xss introduce XXE vulnerabilities. A detailed version of the vulnerabilities will be explained in the vulnerable components section of the report at the end.

Responsibilities:

Tasks	Ibrahim Ege Oral	Ömer Can Öztürk	Barış Ulaş Çukur
Project Report	X	X	X
Backend	X	X	X
Frontend	X	X	X
Honeypot Implementation			X
Decoy Creation			X
Vulnerability Research & Implementation			X
HTML Webpage Creation			X
Vulnerable Component Exploitation			X
Static Code Analysis			X
Credential Recovery Workflow			X
Server Monitoring Implementation	X	X	
Authentication Workflow	X	X	
Commenting & Liking	X	X	
News Search	X	X	
Database Implementation	X	X	

Let’s have a look at the existing endpoints on our honeypot server:

Backend:

`/login(ömer):`

Purpose:

Handles the process of user login, allowing users to authenticate themselves.

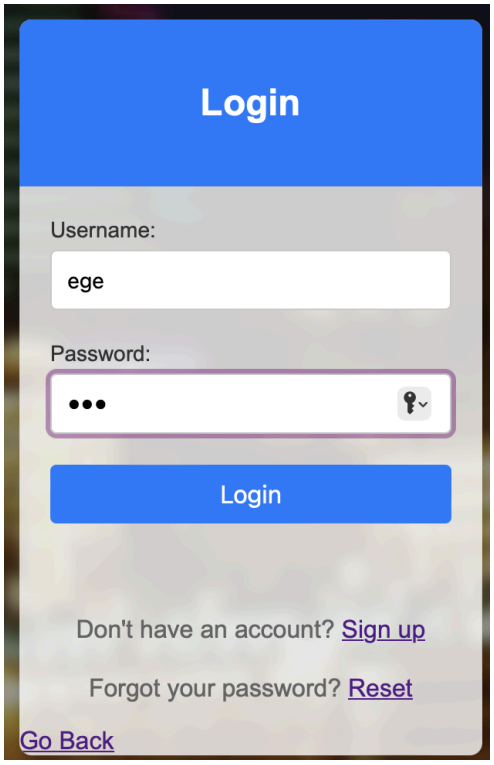
Method:

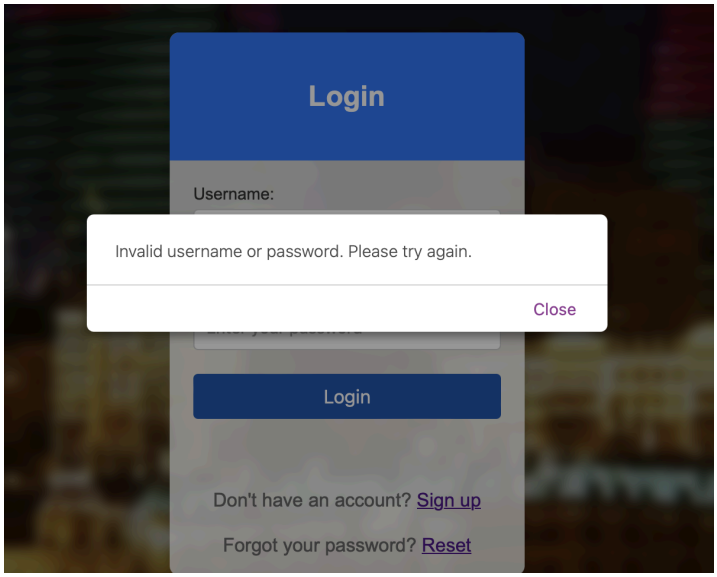
- **POST**: This endpoint expects data to be sent in the form of a JSON payload, containing the user's `username` and `password`.

Functionality:

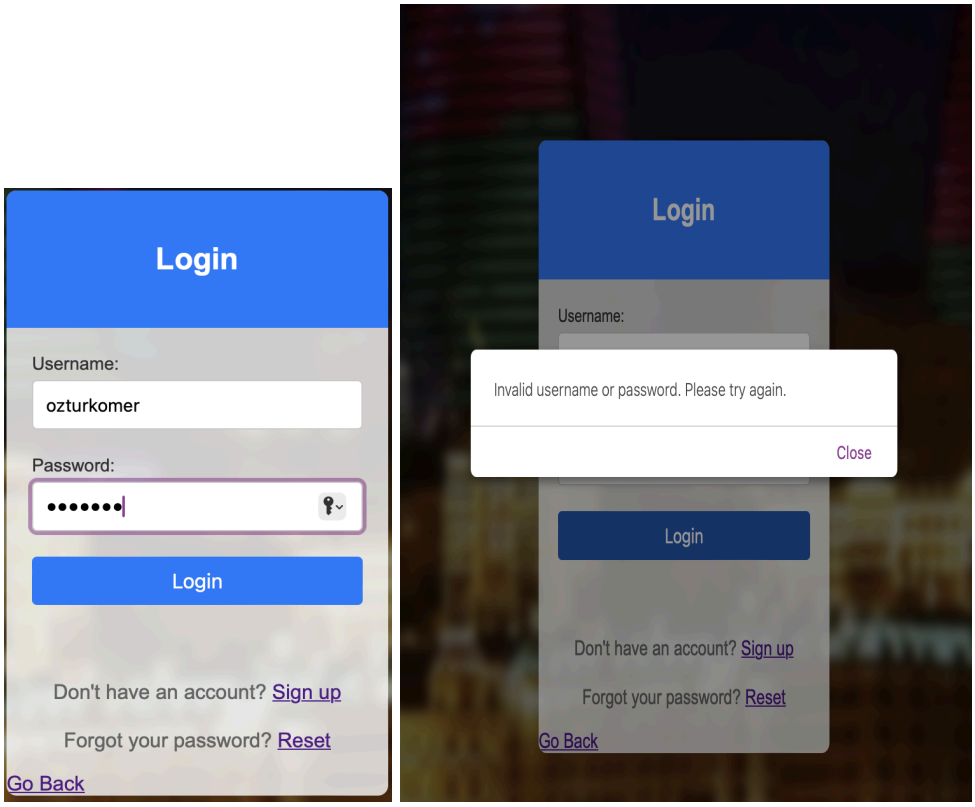
Accepts User Credentials: The endpoint accepts a JSON payload containing the `username` and `password` of the user attempting to log in.
Verification Against Database: The received credentials are then verified against the `User` table in the PostgreSQL database. It checks whether a user with the provided username and password exists.
Redirect on Success: If the credentials are valid, the user is redirected to the index page (`redirect(url_for('index'))`).
Error Handling: If the credentials are invalid or any other error occurs during the process, the endpoint returns an appropriate error response, indicating the nature of the issue.

This is the endpoint which allows the login process to succeed. Whilst retrieving the username and password values from the user, the program first checks whether the user entered both the necessary credentials, if he/she did not, The program warns the user:

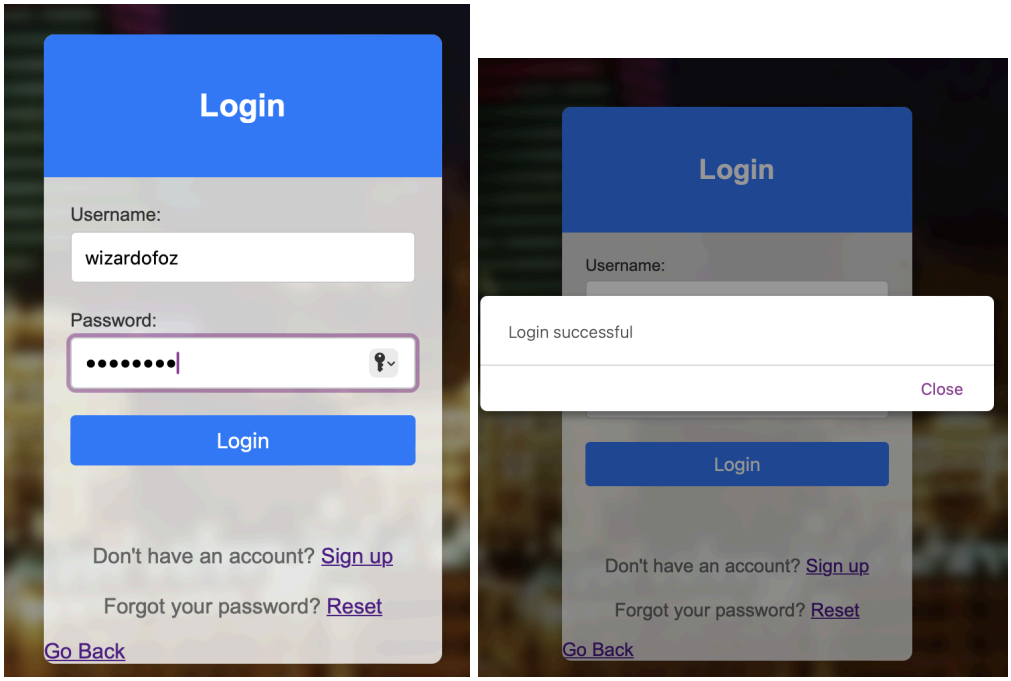




The same message also pops up, whenever the user enters one or both of his/hers credentials incorrectly:



If both credentials are non-empty and correct, the user is logged in successfully to the system:



All of these actions are logged into the application console:

```
127.0.0.1 - - [07/Jan/2024 22:40:21] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [07/Jan/2024 22:40:33] "POST /login HTTP/1.1" 400 -
127.0.0.1 - - [07/Jan/2024 22:40:56] "POST /login HTTP/1.1" 400 -
127.0.0.1 - - [07/Jan/2024 22:41:38] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [07/Jan/2024 22:41:45] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [07/Jan/2024 22:42:07] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [07/Jan/2024 22:42:40] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [07/Jan/2024 22:43:08] "GET /registerPage HTTP/1.1" 200 -
127.0.0.1 - - [07/Jan/2024 22:43:23] "POST /register HTTP/1.1" 201 -
127.0.0.1 - - [07/Jan/2024 22:43:25] "GET /loginPage HTTP/1.1" 200 -
```

The red logs show that the user's requests failed with 400 status codes, whereas the normal colored requests represent the successful requests with 200 status codes.

`/loginPage(ömer):`

Purpose:

Renders the login page, providing a user interface for users to enter their login credentials.

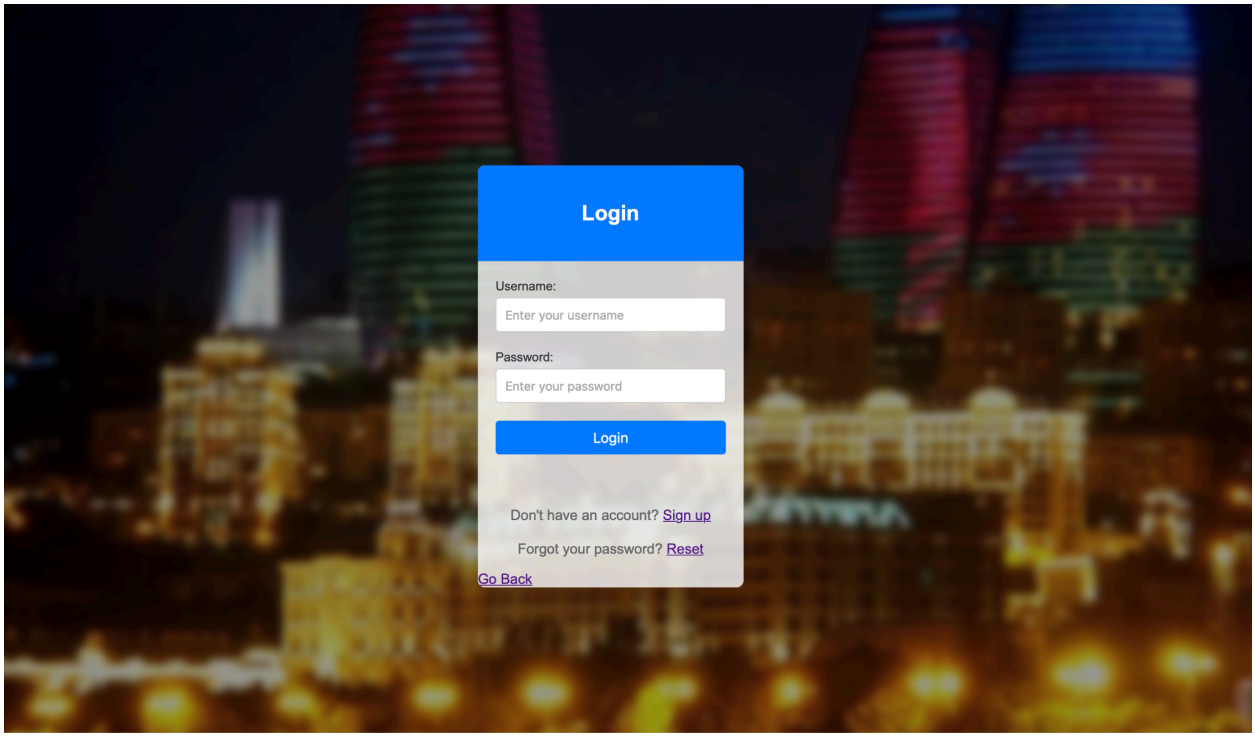
Method:

- **GET**: This endpoint responds to HTTP GET requests.

Functionality:

Rendering Login Page: When accessed via a web browser, the endpoint uses the `render_template` function to render and display the login page. This page typically includes input fields for username and password, as well as a form for

submitting login information.



Here, you can see the application's user login interface. It contains two fields for the user to input: Their username and password. There are also two other choice at the bottom of the login block for the user to click on: One redirecting the user to register page, and the other to reset their password if they have forgotten it. The usage of this endpoint is again logged:

```
127.0.0.1 - - [07/Jan/2024 22:36:36] "GET /loginPage HTTP/1.1" 200 -
```

`/register(ömer):`

Purpose:

Handles the user registration process, allowing new users to create an account.

Method:

- **POST**: Expects data in the form of a JSON payload, including the desired `username` and `password` for the new user.

Functionality:

Data Validation: The endpoint checks whether both `username` and `password` are provided in the JSON payload. If not, it returns an error message.
User Creation: Attempts to create a new user in the `User` table of the PostgreSQL database with the provided `username` and `password`.
Success Response: If the registration is successful, the endpoint returns a success message with an HTTP status code of 201.
Error Handling: If the provided username already exists (IntegrityError), it returns a conflict error (HTTP status code 409). For other errors, it returns a generic error message with an HTTP status code of 500.

`/registerPage(ömer):`

Purpose:

Renders the registration page, providing a user interface for users to sign up for a new account.

Method:

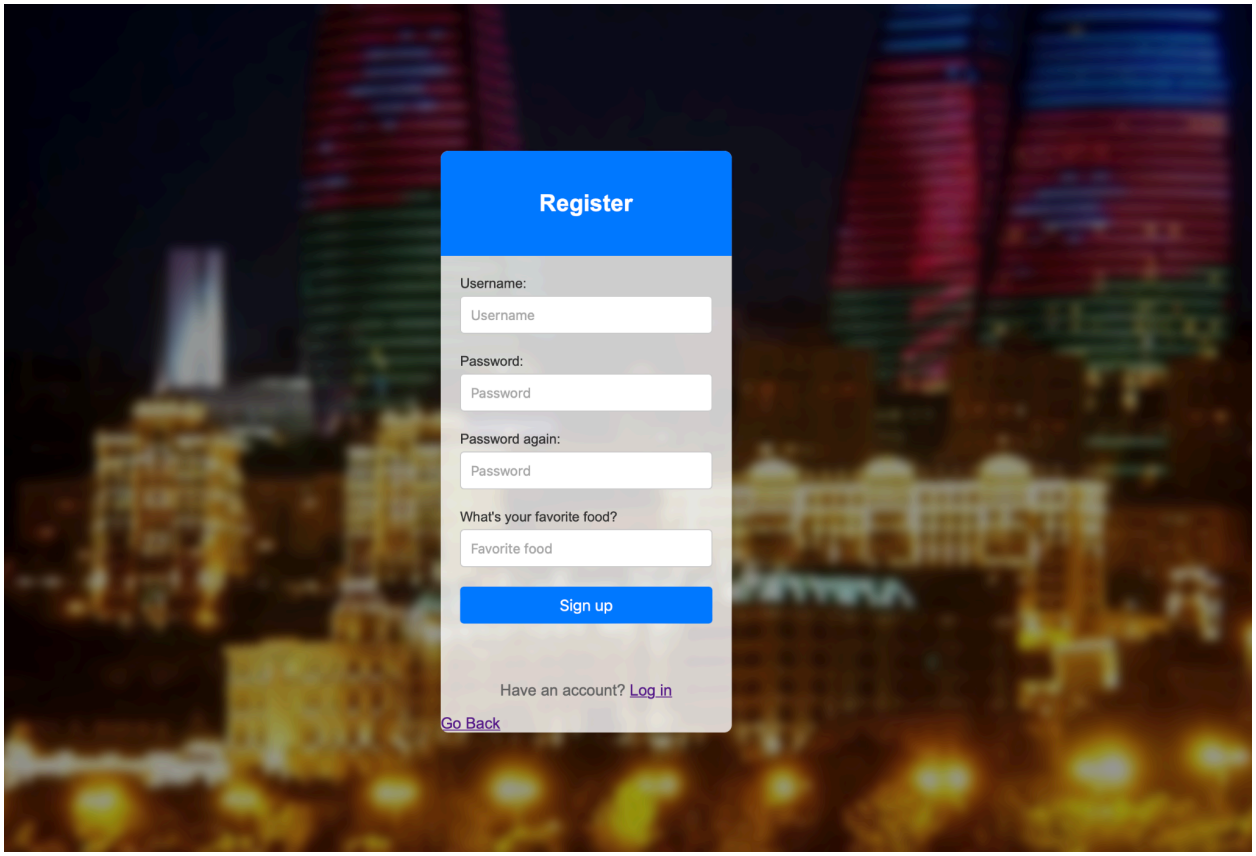
- **GET**: This endpoint responds to HTTP GET requests.

Functionality:

Rendering Registration Page: When accessed via a web browser, the endpoint uses the `render_template` function to render and display the registration page. This page typically includes input fields for choosing a username and password, as well as a form for submitting registration information.

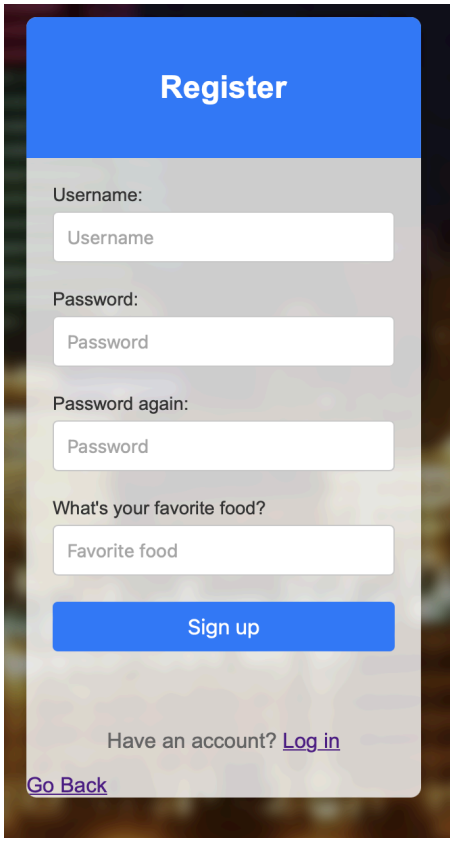
These endpoints collectively form the user authentication and registration system of the Flask application. Users can log in, register for new accounts, and access dedicated pages designed for these purposes. The detailed error handling ensures a smooth user experience and secure data handling.

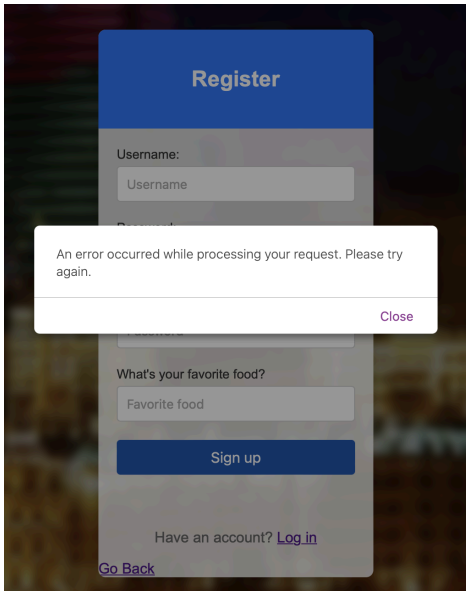
Here is our registration page on the “/registerPage” endpoint. It is quite similar to the login page we have discussed earlier. However, since this is a registration page and for security purposes, we have one additional field that we ask the user only once during the registration process: Their personal favorite food:



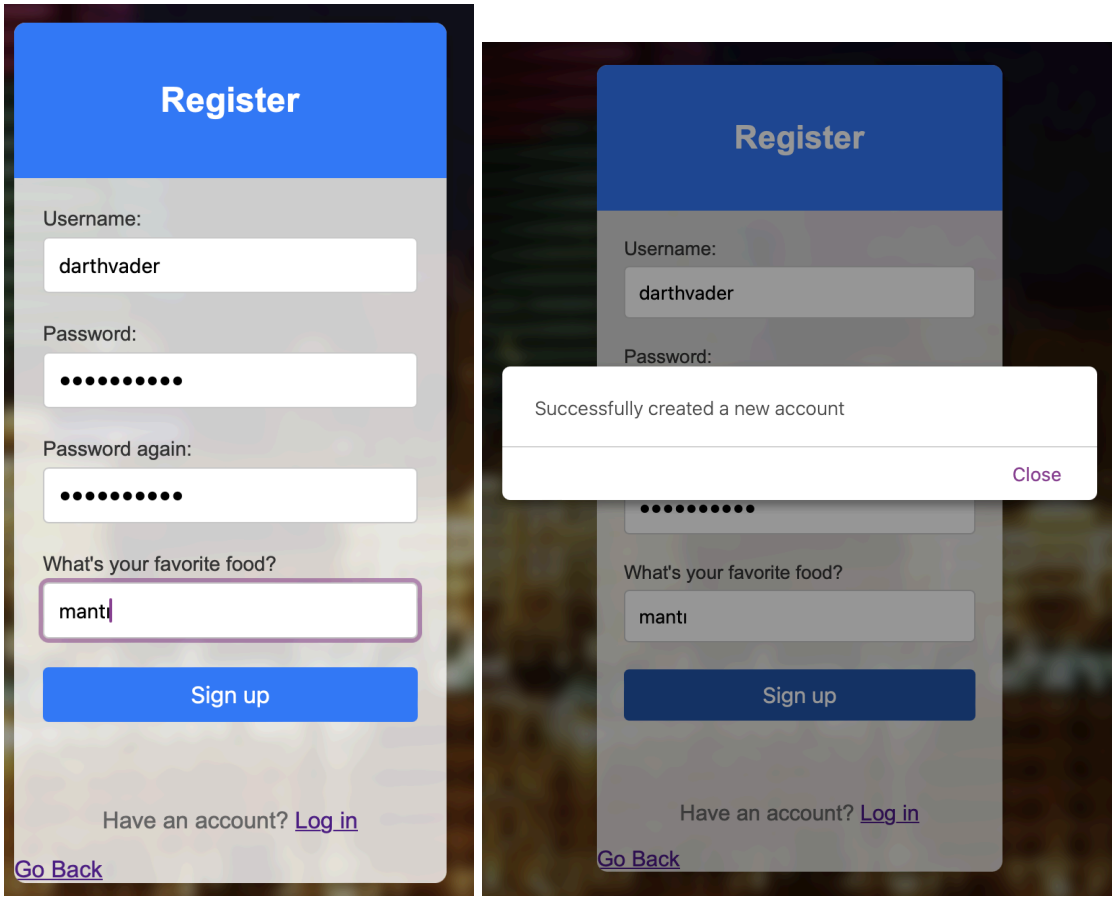
As you can see, the necessary fields for registration, such as username, password, check password and user's security question fields are all present.

If the user leaves one or more of these fields blank, the system will warn the user:





If the user enters all the fields properly, His/her registration is deemed to be complete:



Once again, all of the user's actions are logged onto the console for the administrator to observe:

```
127.0.0.1 - - [07/Jan/2024 22:49:55] "GET /registerPage HTTP/1.1" 200 -
127.0.0.1 - - [07/Jan/2024 22:51:34] "POST /register HTTP/1.1" 400 -
127.0.0.1 - - [07/Jan/2024 22:52:37] "POST /register HTTP/1.1" 201 -
```

/resetPassword(ege & ömer):

Purpose:

The resetPassword endpoints serves as the web page if the user decides to reset their password either due to security reasons or simply because they forgot it.

Functionality:

Rendering The Page:

- When accessed via a web browser or a GET request, the endpoint uses the `render_template` function to render and display the homepage (`resetpass.html'`).

Application Structure:

- The page may include navigation links, input fields, or any other information that the application deems important for users to see immediately.

Connection to Other Pages:

- The page might contain links or navigation elements that guide users to other pages such as login and register pages.

Customization:

- Developers often customize the homepage to make it visually appealing and to provide a user-friendly experience.

Potential Redirects:

- In some cases, applications may choose to redirect users to the homepage after successful login or registration. This can be done using the `redirect(url_for('index'))` method.

This page contains three input fields for the user to enter: username and password tuple to verify that this is indeed an existing user, and the favorite food input. This way, the user will be able to reset their password.

`/` (Index Endpoint)(Ege):

Purpose:

The index endpoint serves as the homepage of the application. When users access the root URL of the application, they are directed to this endpoint.

Method:

- **GET**: Responds to HTTP GET requests.

Functionality:

Rendering Homepage:

- When accessed via a web browser or a GET request, the endpoint uses the `render_template` function to render and display the homepage (`'homepage.html'`).

Template Rendering:

- The rendered template typically includes the main content and features of the application that users should see when they first visit the site.

Application Structure:

- The homepage may include navigation links, featured content, or any other information that the application deems important for users to see immediately.

Connection to Other Pages:

- The homepage might contain links or navigation elements that guide users to other pages within the application, such as login, registration, or specific content pages.

Customization:

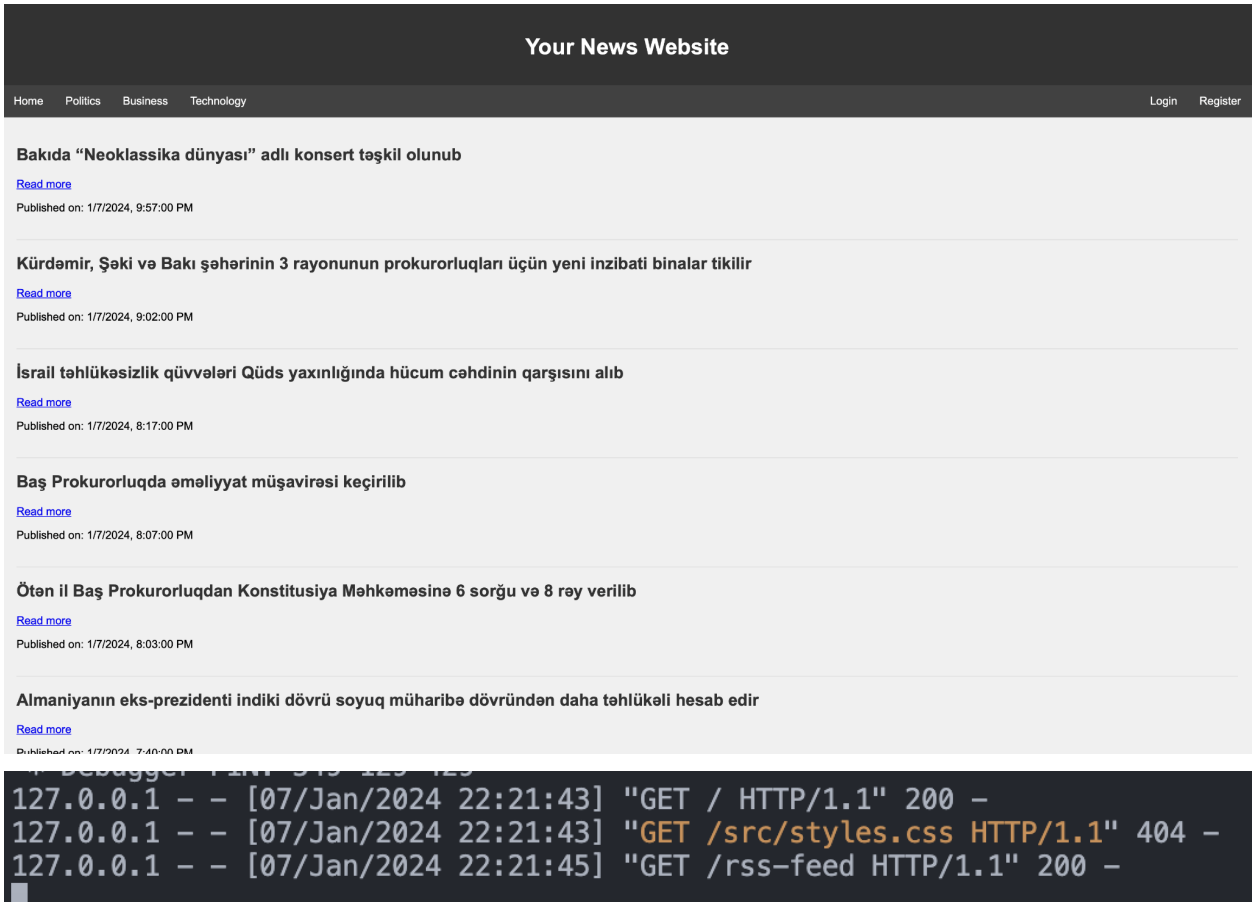
- Developers often customize the homepage to make it visually appealing and to provide a user-friendly experience.

Potential Redirects:

- In some cases, applications may choose to redirect users to the homepage after successful login or registration. This can be done using the `redirect(url_for('index'))` method.

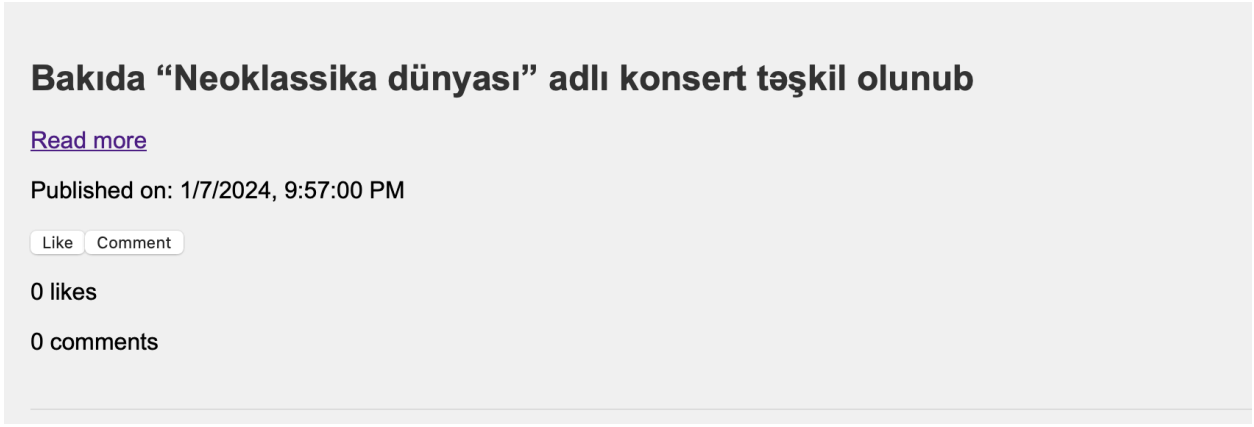
By serving as the starting point for users, the `/` endpoint sets the tone for the application and provides a central hub for navigation. It plays a crucial role in creating a cohesive and engaging user experience. The specific content and features displayed on the homepage can vary based on the goals and design choices of the application developers.

Here, you can see the index page of our application. It lists all the retrieved news from the rss-feed, contains several navigation bar components to display certain categories and has a login/register bar at the top right corner.



As you can see, our HTTP requests to get to the page are also being logged into the console, giving us access to the information about the HTTP request. The index endpoint also makes use of the /rss-feed endpoint, which retrieves all the news from the predefined rss-feed link.

Once the user clicks on “Read More” in one of the news blocks, the user will be redirected to the actual news page:



Here, you can see the news block on our web server. The Read More link at the bottom of the header of the news, will direct the user to the specified link in the news block.

Database Initialization (Ege):

- A PostgreSQL database is configured using SQLAlchemy with the URI specified in the `app.config['SQLALCHEMY_DATABASE_URI']`.
- Two models are defined:
  - **`User`**: Represents user data with fields `id`, `username`, and `password`.
  - **`Comment`**: Represents comments with fields `id`, `text`, `user_id`, and a relationship with the `User` model.
- The application runs in debug mode (`app.run(debug=True)`) and creates all database tables during initialization (`db.create_all()`).

Notes:

- The application includes routes for handling RSS feeds, user registration and login, comment management, and rendering HTML templates.
- The `/rss-feed` and `/rss-feed-more/<path:rss_feed_url>` routes demonstrate fetching and parsing RSS feeds.
- The `/register` and `/login` routes handle user registration and login, respectively.
- The application supports adding, deleting, and retrieving comments via the `/add-comment`, `/delete-comment/<int:comment_id>`, `/comments`, and `/comments/<int:comment_id>` routes.
- The homepage is rendered at the `/` endpoint, and specific pages are rendered for login, registration, and news.
- Error handling is implemented for various scenarios, and the application provides appropriate status codes and messages.

`/delete-comment/<int:comment_id>:(ege)`

- **Purpose**: The delete-comment endpoint serves the purpose of allowing the removal of a comment from the database based on the specified `comment_id`.
- **Functionality:**
  Comment Deletion:
  - When a DELETE request is sent to this endpoint with a valid `comment_id`, the server attempts to delete the comment associated with that specific identifier from the database.

  Database Interaction:
  - The server checks if a comment with the specified `comment_id` exists in the database.
  - If the comment exists, it is deleted from the database using SQLAlchemy.
  - If the comment does not exist, the endpoint responds with a JSON message: "Comment not found" and a status code of 404 (Not Found).

  Response:
  - If the comment is successfully deleted, the endpoint returns a JSON response with a message: "Comment deleted successfully" and a status code of 200 (OK).

- If the comment does not exist, the endpoint responds with a JSON message: "Comment not found" and a status code of 404.

Error Handling:

- The endpoint includes error-handling mechanisms to handle cases where the specified `comment_id` is not found in the database.

Use Cases:

Content Moderation:

- Allows administrators or authorized users to remove inappropriate or irrelevant comments from the system.

User Management:

- Provides a mechanism to manage and control the content displayed on the platform by allowing the removal of specific comments.

Database Integrity:

- Ensures the integrity of the database by removing any instances of comments with the specified `comment_id`.

- 
DELETE
http://127.0.0.1:5000/delete-cor ...
Send
Body
raw
JSON
Beau
1 {
2 "text":"Hello",
3 "user_id": 1
4 }
Body
200 OK
638 ms
245 B
Save as example
Pretty
JSON
1 {
2 "message": "Comment deleted successfully"
3 }

127.0.0.1 - - [07/Jan/2024 23:03:05] "DELETE /delete-comment/1 HTTP/1.1" 200 -

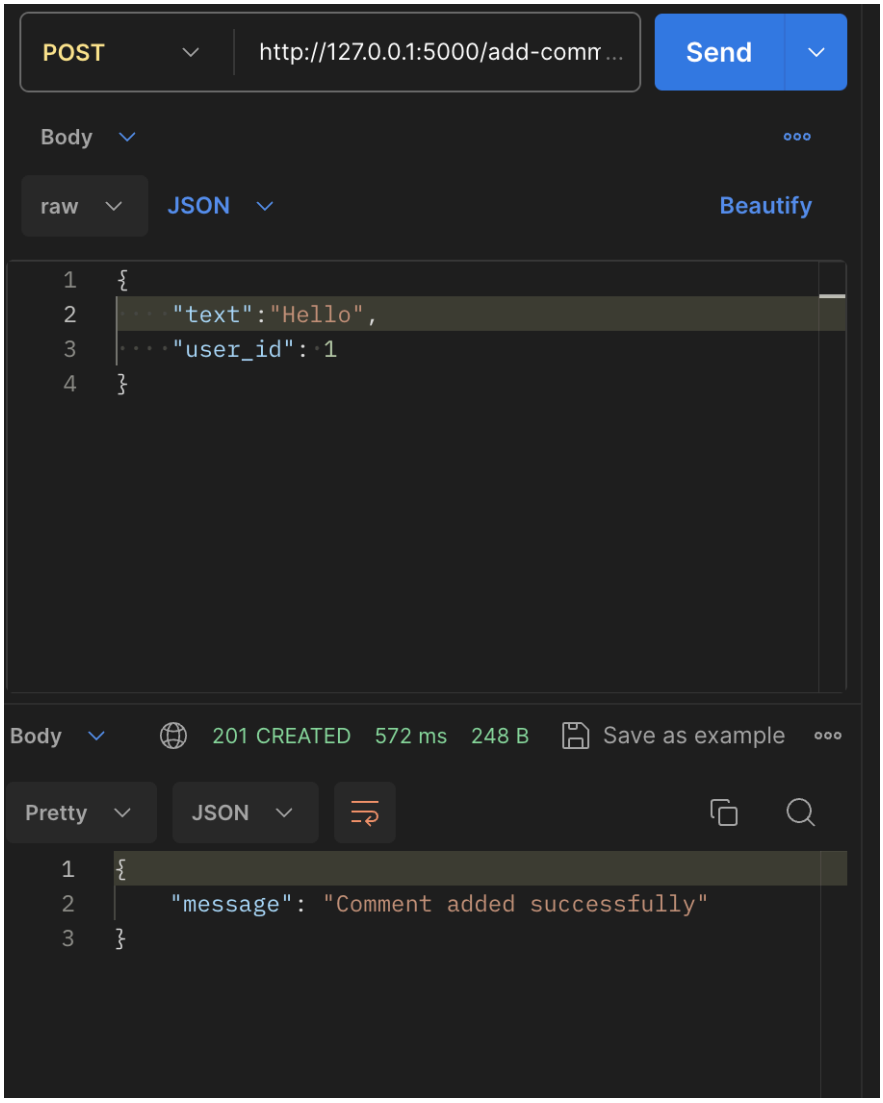
```
{
  "rss_feed": [
    {
      "description": "",
      "link": "https://azertag.az/xeber/bakida_neoklassika_dunyasi_adli_konsert_teskil_olunub-2875489",
      "pub_date": "Sun, 07 Jan 2024 22:57:00 +0400",
      "title": "Bak\u0131da \u201cNeoklassika d\u00fcnyas\u0131\u201d adl\u0131 konsert t\u0259\u015fkil olunub"
    },
    {
      "description": "",
      "link": "https://azertag.az/xeber/kurdemir_seki_ve_baki_seherinin_3_rayonunun_prokurorluqlari_uchun_yeni_inzibati_binalar_tikilir-2875486",
      "pub_date": "Sun, 07 Jan 2024 22:02:00 +0400",
      "title": "K\u00fcrd\u0259mir, \u015e\u0259ki v\u0259 Bak\u0131 \u015f\u0259h\u0259rinin 3 rayonunun prokurorluqlar\u0131 \u00fc\u00e7\u00fcn yeni inzibati binalar tikilir"
    },
    {
      "description": "",
      "link": "https://azertag.az/xeber/israil_tehlukesizlik_quvveleri_quds_yaxinliginda_hucum_cehdinin_qarsisini_alib-2875483",
      "pub_date": "Sun, 07 Jan 2024 21:17:00 +0400",
      "title": "\u0130srail t\u0259hl\u00fck\u0259sizlik q\u00fcvv\u0259l\u0259ri Q\u00fcds yax\u0131nl\u0131\u011f\u0131nda h\u00fccum c\u0259hdinin qar\u015f\u0131s\u0131n\u0131 al\u0131b"
    },
    {
      "description": "",
      "link": "https://azertag.az/xeber/bas_prokurorluqda_emeliyyat_musaviresi_kechirilib-2875480",
      "pub_date": "Sun, 07 Jan 2024 21:07:00 +0400",
      "title": "Ba\u015f Prokurorluqda \u0259m\u0259liyyat m\u00fc\u015favir\u0259si ke\u00e7irilib"
    },
    {
      "description": "",
      "link": "https://azertag.az/xeber/oten_il_bas_prokurorluqdan_konstitusiya_mehkemesine_6_sorgu_ve_8_rey_verilib-2875477",
      "pub_date": "Sun, 07 Jan 2024 21:03:00 +0400",
      "title": "\u00d6t\u0259n il Ba\u015f Prokurorluqdan Konstitusiya M\u0259hk\u0259m\u0259sin\u0259 6 sor\u011fu v\u0259 8 r\u0259y verilib"
    },
    {
      "description": "",
      "link": "https://azertag.az/xeber/almaniyanin_eks_prezidenti_indiki_dovru_soyuq_muharibe_dovrunden_daha_tehlukeli_hesab_edir-2875465",
      "pub_date": "Sun, 07 Jan 2024 20:40:00 +0400",
      "title": "Almaniyan\u0131n eks-prezidenti indiki d\u00f6vr\u00fc soyuq m\u00fcharib\u0259 d\u00f6vr\u00fcnd\u0259n daha t\u0259hl\u00fck\u0259li hesab edir"
    },
    {
      "description": "",
      "link": "https://azertag.az/xeber/samux_ve_xizi_rayonlarina_prokurorlar_teyin_olunub-2875462",
      "pub_date": "Sun, 07 Jan 2024 20:30:00 +0400",
      "title": "Samux v\u0259 X\u0131z\u0131 rayonlar\u0131na prokurorlar t\u0259yin olunub"
    },
    {
      "description": "",
      "link": "https://azertag.az/xeber/yuz_yasli_sehid_anasi_dogum_gununde_ziyaret_edilib-2875447",
      "pub_date": "Sun, 07 Jan 2024 19:49:00 +0400",
      "title": "Y\u00fcz ya\u015fl\u0131 \u015f\u0259hid anas\u0131 do\u011fum g\u00fcn\u00fcnd\u0259 ziyar\u0259t edilib"
    },
    {
      "description": "",
      "link": "https://azertag.az/xeber/xachmazda_zencirvari_avtomobil_qezasi_olub_video-2875444",
      "pub_date": "Sun, 07 Jan 2024 19:32:00 +0400",
      "title": "Xa\u00e7mazda z\u0259ncirvari avtomobil q\u0259zas\u0131 olub V\u0130DEO"
    },
    {
      "description": "",
      "link": "https://azertag.az/xeber/qerbi_narahat_eden_eminlik_amili__tehlil-2875435",
      "pub_date": "Sun, 07 Jan 2024 19:09:00 +0400",
      "title": "\u201cQ\u0259rbi narahat ed\u0259n \u201c\u0259minlik\u201d amili\u201d  T\u018fHL\u0130L"
    },
    {
      "description": "",
      "link": "https://azertag.az/xeber/turkiye_superliqasi_fenerbagcha_yeni_ile_darmadaginla_baslayib-2875423",
      "pub_date": "Sun, 07 Jan 2024 18:57:00 +0400",
      "title": "T\u00fcrkiy\u0259 Superliqas\u0131: \u201cF\u0259n\u0259rba\u011f\u00e7a\u201d yeni il\u0259 darmada\u011f\u0131nla ba\u015flay\u0131b"
    },
    {
      "description": "",
      "link": "https://azertag.az/xeber/sehid_herbi_pilot_resad_atakisiyevin_mezari_ziyaret_edilib_video-2875414",
```

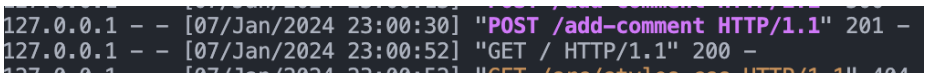
`/add-comment` endpoint:(ege)

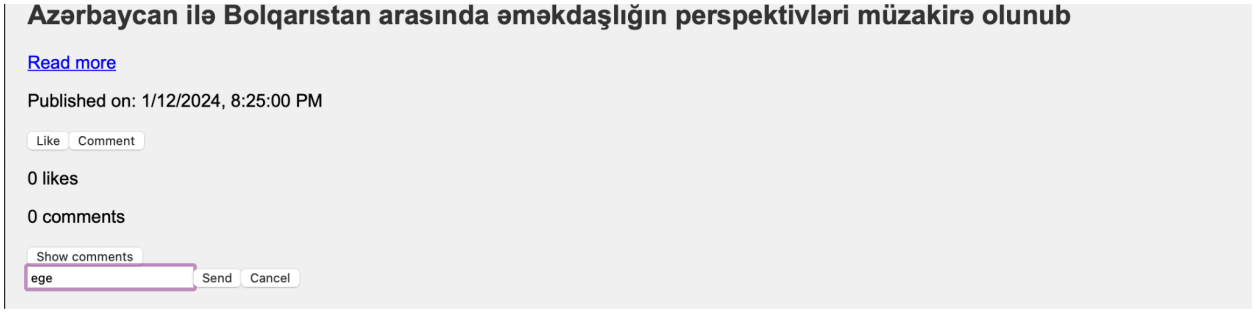
- **Purpose**: The add-comment endpoint serves the purpose of allowing users to submit new comments associated with news articles. It facilitates user engagement by providing a mechanism to express opinions and participate in discussions related to specific posts.
- **Method: POST:** This endpoint responds exclusively to HTTP POST requests.
- **Functionality:**
  Comment Submission:
  - Users submit comments by sending a POST request to this endpoint.
  - The request body must contain a JSON object with two mandatory properties: `text` (the content of the comment) and `post_url` (the URL of the associated news article).
  - If either of these properties is missing, the endpoint responds with a JSON message: "Please provide text and post_url" and a status code of 400 (Bad Request).

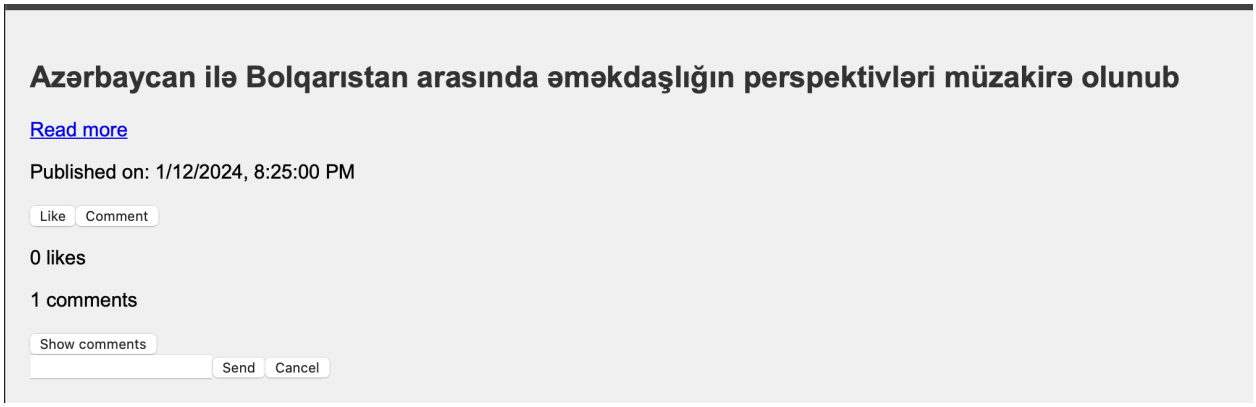
  Database Interaction:
  - Upon receiving a valid request, the server processes the data and creates a new comment record in the database using the SQLAlchemy ORM.
  - The new comment includes details such as the comment text and the URL of the associated news article.
  - If there are any issues while interacting with the database, the endpoint responds with a JSON message: "Error: [error message]" and a status code of 500 (Internal Server Error).

  Response:
  - If the comment is added successfully, the endpoint returns a JSON response with a message: "Comment added successfully" and a status code of 201 (Created).

- 
POST
http://127.0.0.1:5000/add-comm...
Send
Body
raw
JSON
Beautify
{
"text":"Hello",
"user_id":1
}
Body
201 CREATED
572 ms
248 B
Save as example
Pretty
JSON
{
"message": "Comment added successfully"
}


127.0.0.1 - - [07/Jan/2024 23:00:30] "POST /add-comment HTTP/1.1" 201 -
127.0.0.1 - - [07/Jan/2024 23:00:52] "GET / HTTP/1.1" 200 -

- 

- 

/ (News Search Endpoint)(ege&ömer):

Purpose:

The news search endpoint enables users to search for news articles based on a specified title. Users can access this endpoint to retrieve a list of news articles matching the provided search term.

Method:

GET: Responds to HTTP GET requests.

Functionality:

News Title Search:

When a user accesses the endpoint with a GET request, the provided news title in the URL is used to query the database for news articles with titles similar to the search term.

Database Query:

The endpoint utilizes the SQLAlchemy query `Post.query.filter(Post.title.ilike(f'%{news_title}%')).all()` to perform a case-insensitive search for news articles containing the specified title substring.

Results Presentation:

The search results, including titles, URLs, and the number of likes, are then formatted into a JSON response (`{'search_results': results_list}`) for the user to consume.

Customization:

Developers can customize the endpoint's behavior, such as adjusting the search logic or the format of the response, based on the application's requirements.

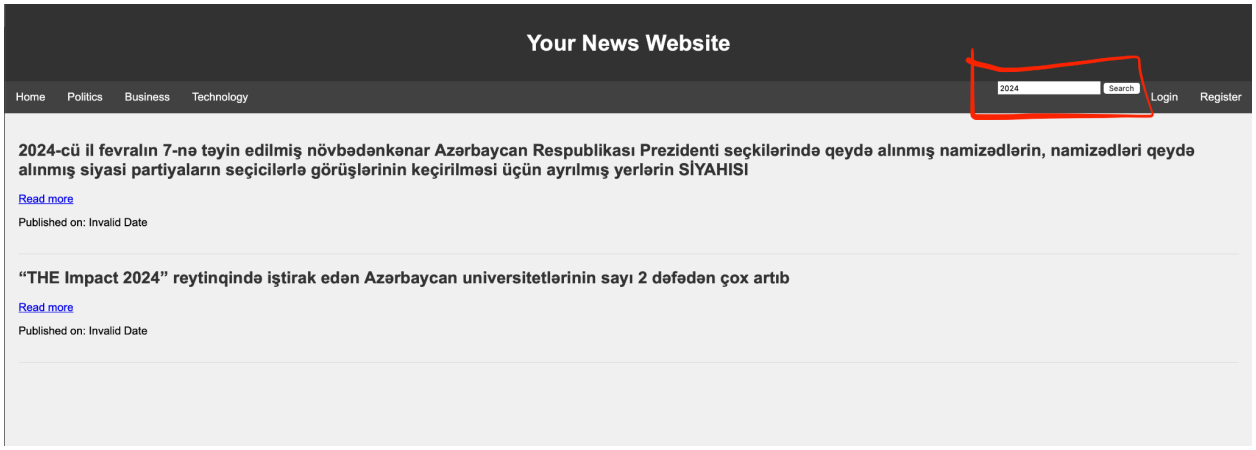
Potential Use Cases:

1. News Aggregation: The endpoint can be part of a news aggregation system where users search for articles of interest.
2. User Engagement: Offering users the ability to find specific news articles enhances their engagement with the application.
3. Content Discovery: Users can discover relevant news content by searching for specific topics or keywords.

Integration with Frontend:

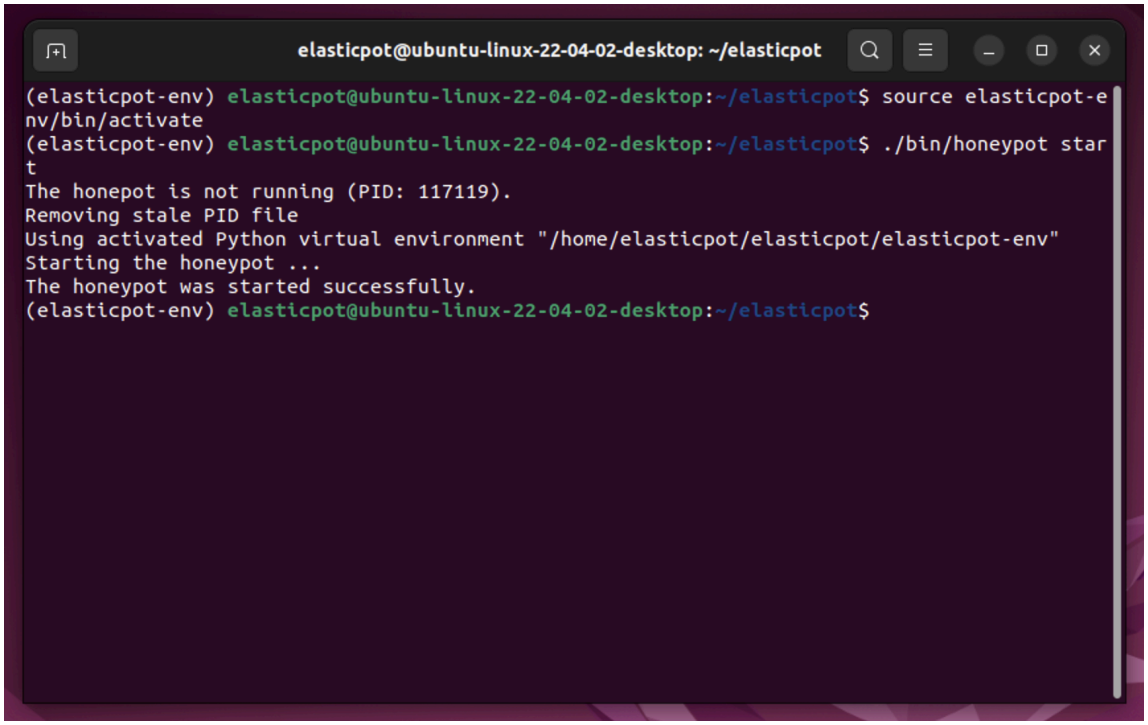
The endpoint is typically integrated into the frontend of the application, allowing users to interact with the search feature through a user-friendly interface.

This news search functionality enriches the user experience by providing a means to discover and access specific news articles within the application. It serves as a valuable tool for users seeking information on particular topics or events.



Configuring Honeypot

We’ve utilized Elasticpot for this project to monitor the traffic. The environment of the Honeypot is a virtual arm64 Ubuntu machine (basically a virtual honeypot). The honeypot was configured according to the installation steps provided at the GitHub repository.



Vulnerable Component 1

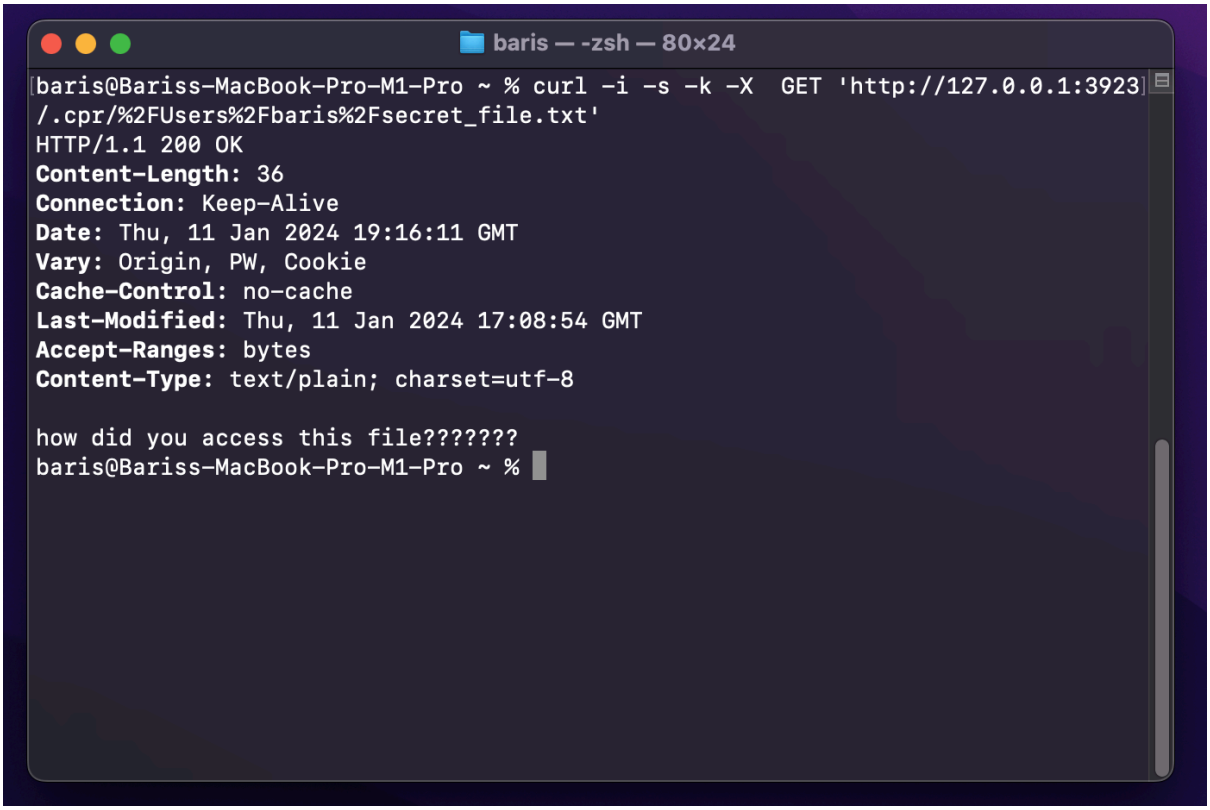
Package name: copyparty (1.8.2)
Vulnerability: Path Traversal

Copyparty is a file server that serves the contents of its working directory. Considering a scenario where a news agency would serve files from one of their databases for editors to upload news documents, this vulnerability allows arbitrary file reads (or modifications if privilege is sufficient).

According to Vartamtzidis Theodoros who is the author of this exploit, the vulnerability occurs from the ".cpr" subfolder inside the file server. Given the URL of the server, an attacker can access any files of the system.

Demonstration:

The server is running from the directory "/Users/baris/CS437-Assignment-1" (Figure x). Normally, this would disallow accessing any of the files outside of this directory. However, due to this vulnerability, we're able to access the document "/Users/baris/secret_file.txt" without any hassle. We're using the %2F character to exploit the vulnerability which translates to a forward slash "/" character in ASCII encoding.



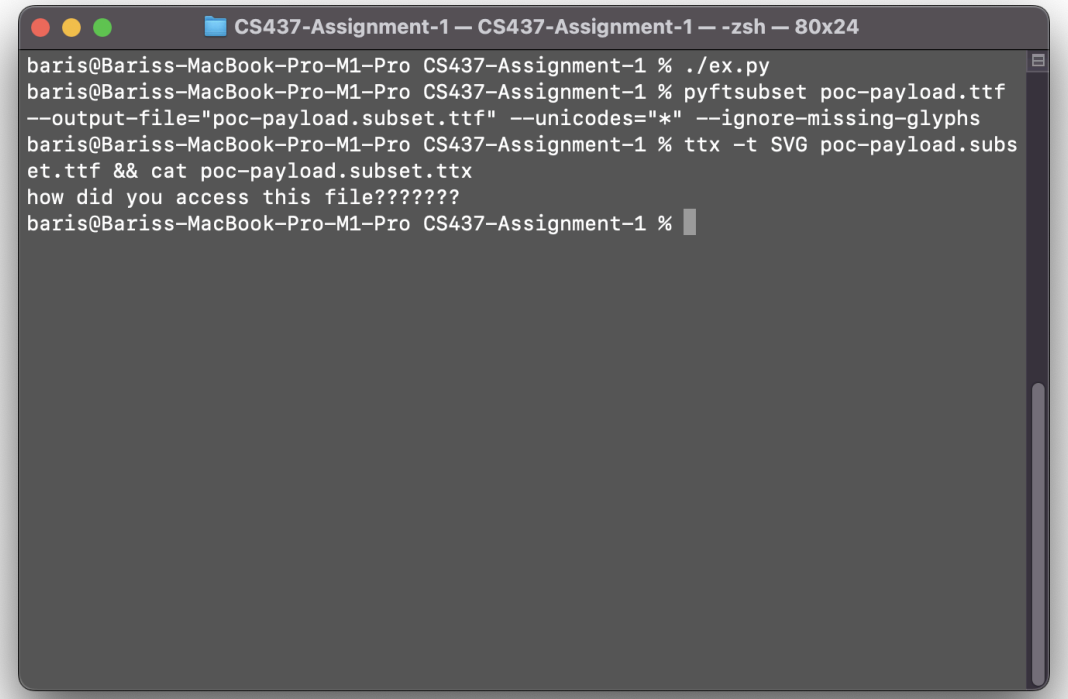
More Info: https://www.exploit-db.com/exploits/51636

Vulnerable Component 2

Package name: Fonttools
Vulnerability: XXE

Description: Fonttools version before 4.43.0 is vulnerable to XXE because the developers forgot to set resolve_entities=True in the XML parser of the code. Considering a scenario where the files are rendered with custom fonts, a user could arbitrarily access any file in the system. In the server code, this is used as an endpoint of /inspect-font which calls the font inspector function to decide if the website can be rendered using such font.

Demonstration: The demonstration for this part executes the script in the GitHub repository security section. The execution allows an attacker to view any file specified in the XXE_SVG variable of the code.



More Info: https://github.com/advisories/GHSA-6673-4983-2vx5

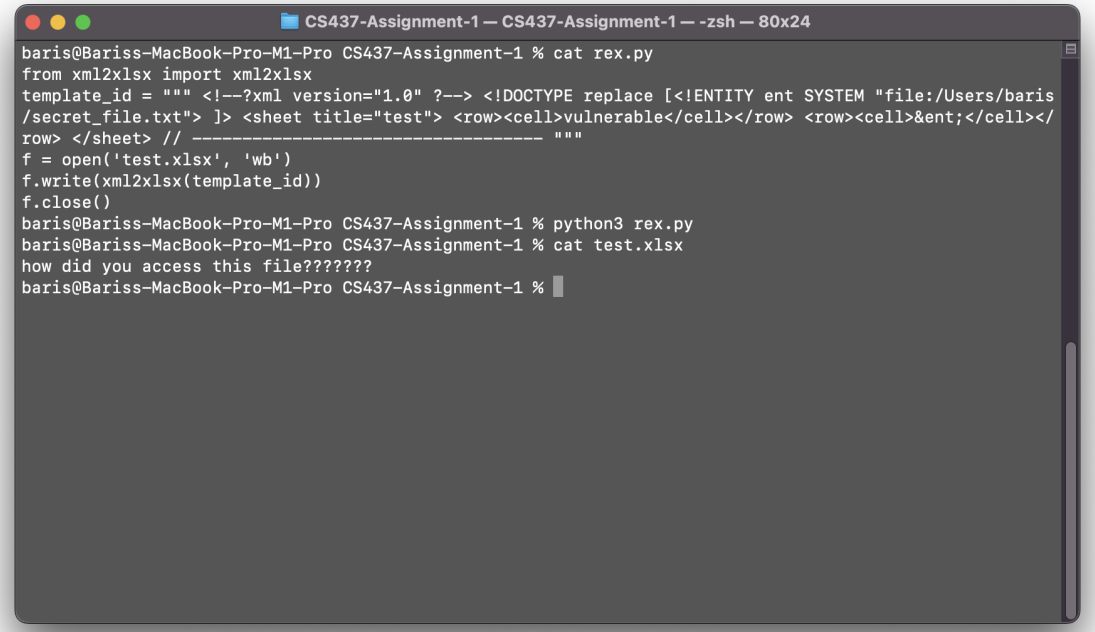
Vulnerable Component 3

Package name: xml2xlsx
Vulnerability: XXE

Description: xml2xlsx is a Python library for converting xml files to xlsx. Similar to Fonttools, older versions of this library also suffer from XXE attacks.

Given a scenario where the site editors of this news page want to archive their feed to xlsx, their entire system could be compromised. Similar to Fonttools, we're exploiting a PoC (Proof of Concept) code given by the exploit authors. The exploit is performed using Python using basic file I/O operations.

Demonstration:

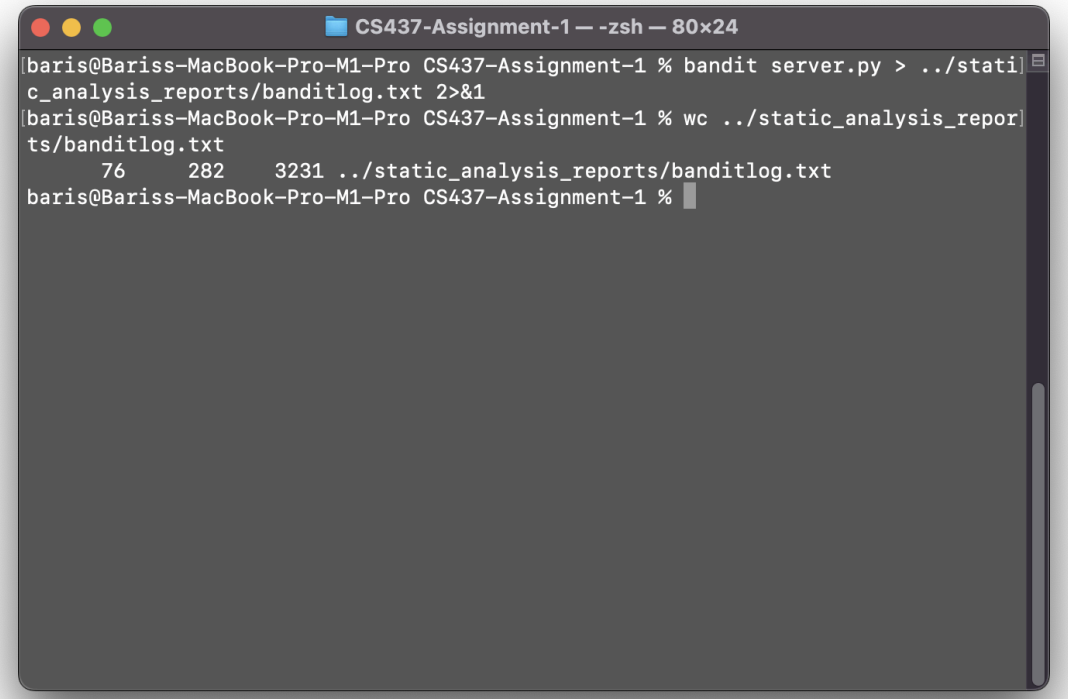


More Info: https://security.snyk.io/vuln/SNYK-PYTHON-XML2XLSX-5881344

Part 2: Static Analysis Tools

1. Bandit

Bandit was quite a bit faster than other static analyzers. It was able to find the issues with the hardcoded API key as well as the salt with medium confidence level. It has also found that the flask app was running with the debug parameter which exposes Werkzeug debugger. Other than the real issues, it has found two false positives, in which considering the nature of static code analyzers, not bad at all. Overall, Bandit is a pretty successful and fast tool.



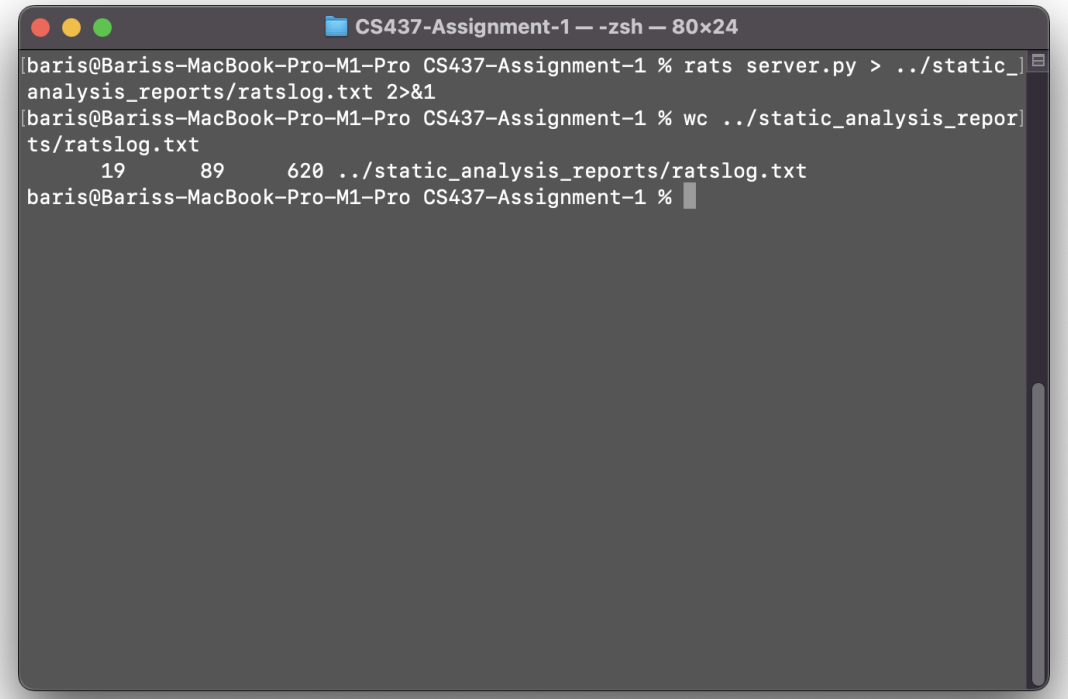
The full log of bandit is available here:
https://github.com/BarisUlas/437_SA_outputs/blob/main/static_analysis_reports/banditlog.txt

2. PYT (Python Taint)

I was unable to get PYT working on my machine, so I couldn't test this library. Though, given the last update date of 2018, I wouldn't assume this static analysis tool to be spectacular anyways.

3. Rough-Auditing-Tool-for-Security

Compared to other static analyzers, it was not as easy to get RATS running, due to it being written in C. The repository did not offer any pre-compiled binaries as well, so I downloaded the source code and compiled it according to the guide at the readme document.



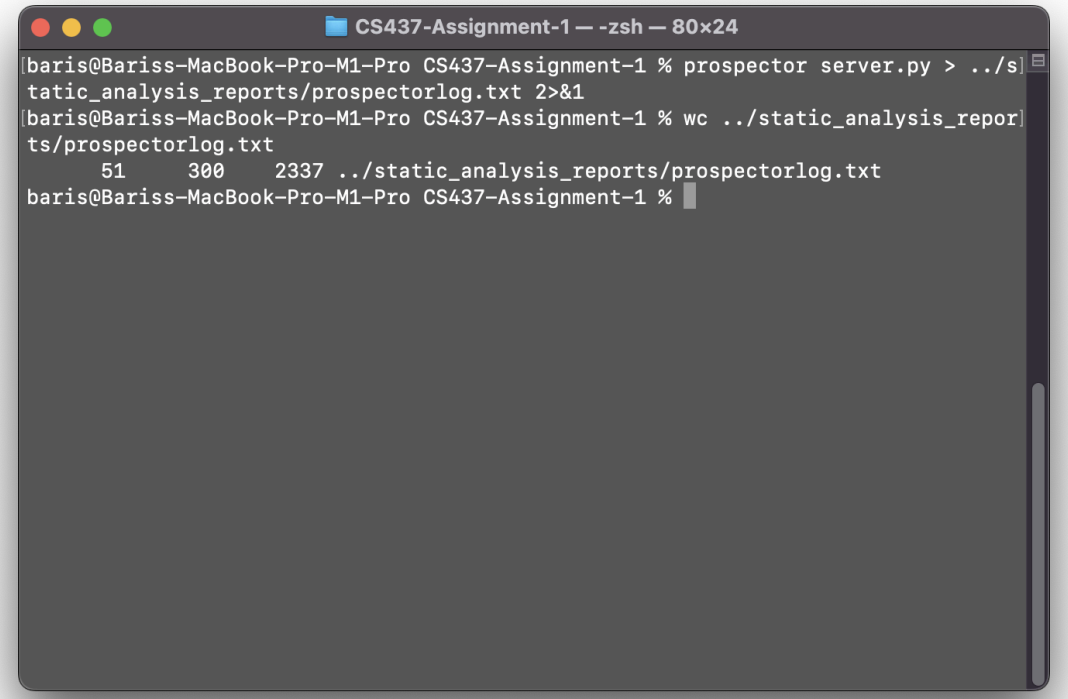
RATS was last updated 11 years ago, which makes is deprecated for any modern language code analysis. Unsurprisingly, trying RATS on our Python server code revealed nothing but disappointment.

RATS only complained about the “@” character, and said that it was a bad token. Other than that, it did not find any other vulnerabilities (false positives included).

The full log of RATS is available here:
https://github.com/BarisUlas/437_SA_outputs/blob/main/static_analysis_reports/ratslog.txt

4. Prospector

Prospector is a maintained and up-to-date static code analyzer which works great with Python. However, the analyzer is quite a bit slower than Bandit. I would say this was a negative, but prospector managed to find a lot more issues with the code than Bandit, so I believe it’s justified.

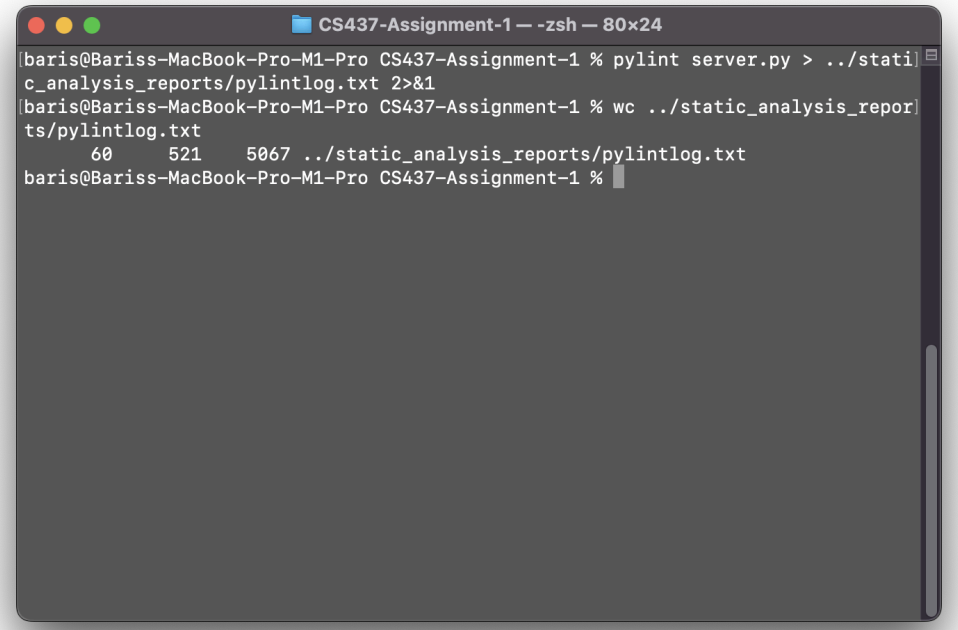


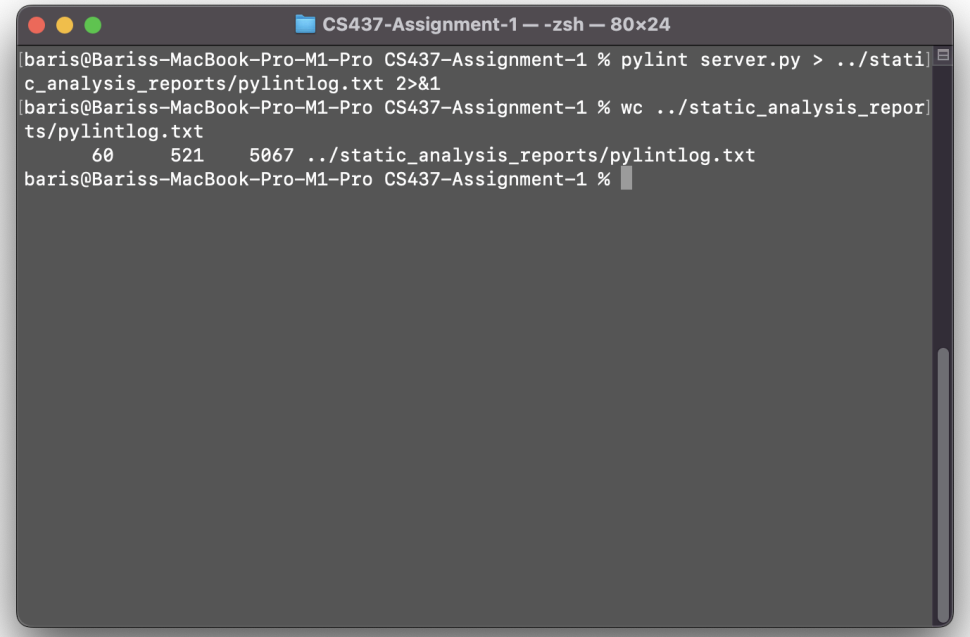
Prospector was able to find quite a few issues with the code, including redundant library imports, unused variables, unnecessary return statements etc.

The full log of Prospector is available here: https://github.com/BarisUlas/437_SA_outputs/blob/main/static_analysis_reports/prospectorlog.txt

5. Pylint

Pylint is one of the most popular static analyzer tools written in Python. Testing the code revealed most of the issues that was found by Prospector. However, it has also showed a lot more false positives, which is a good thing for static analyzers. Pylint also showed a fun little rating for the code at the end of the report, which was 6.1/10 for the project.





The full log of Pylint is available here:
https://github.com/BarisUlas/437_SA_outputs/blob/main/static_analysis_reports/pylintlog.txt