

School of Computer Science Engineering & Technology



Bennett University, Greater Noida, Uttar Pradesh

Corporate Management System

By: Samaksh Tyagi [E23CSEU1303]

Soumya Singh [E23CSEU1297]

Shobhit Choudhury [E23CSEU1304]

Batch No - 44

Course: CSET101: Computational Thinking and Programming

Lab Instructor: Ms. Shivani Tufchi

Submission Date: 21/11/23

Abstract:

Designing a robust corporate management system involves integrating strategic planning, resource allocation, and performance evaluation mechanisms. This system ensures streamlined operations, enhances decision-making, fosters effective communication, and promotes accountability within the organization. Its implementation requires careful analysis of organizational goals, technological infrastructure, and stakeholder engagement for optimal functionality.

Introduction:

Our project envisions a Business Management System that not only alleviates these challenges but also aligns with the sub-targets of SDG 8. Through efficient record management, transparent job application processes, advanced employee analysis tools, open room discussions, and a focus on equal opportunities, our system contributes to the overarching goal of achieving "Decent Work and Economic Growth."

1. Background:

- **Inefficient Record Handling:**

Existing systems often lack user-friendly interfaces for managing records, leading to time-consuming and error-prone processes. The need is for a robust solution that streamlines record handling.

- **Opaque Job Application Processes:**

Job seekers and employers face challenges in aligning skill sets with job requirements. A clear, user-friendly job application system is essential for creating a bridge between talent and opportunities.

- **Limited Employee Analysis Tools:**

Comprehensive analysis of employee performance is vital for informed decision-making. Existing tools may lack intuitive data visualization, making it difficult for organizations to derive actionable insights from their datasets.

- **Inequality in Career Advancement:**

The absence of transparent promotion criteria can lead to disparities in career advancement. There is a need for fair evaluation metrics based on work hours, skills, and achievements.

2. Objectives:

- **Business Sector :** To assist a wide range of businesses in dealing with all assessment data via the Database Management System (DBMS)
- **Data retrieval:** Provide a way to retrieve data quickly and easily using search queries.
- **Data manipulation:** Our DBMS provides tools to manipulate data, such as sorting, filtering, and aggregating data.
- **Security:** Provide security features to ensure that only authorized users have access to the data.
- **Multi-user access:** Allow multiple users to access and modify data simultaneously.
- **Reporting and analysis:** Provides tools to generate reports and analyse data to gain insights and make informed decisions.

Methodology

2.1 Tools and Technologies Used

- **Python:**

Python is a high-level, interpreted, and general-purpose dynamic programming language that focuses on code readability.

1. **Presence of third-party modules:** Python has a rich ecosystem of third-party modules and libraries that extend its functionality for various tasks.

2. **Extensive support libraries:** Python boasts extensive support libraries like `NumPy` and `Pandas` for data analytics and `Matplotlib`, `Seaborn` for data visualisation, making it suitable for scientific and data-related applications.
 3. **Open source and large active community base:** Python is open source, and it has a large and active community that contributes to its development and provides support.
 4. **Versatile, easy to read, learn, and write:** Python is known for its simplicity and readability, making it an excellent choice for both beginners and experienced programmers.
- **HTML + CSS:**

HTML : HTML serves as the foundation for building web pages and that is exactly why it was used with an integration with Django and CSS.

CSS : CSS is defined as a method sheet language that provides web designers control over how an internet site communicates with web browsers including the formatting and display of their HTML documents.
 - **JavaScript:**

JavaScript is a *lightweight, cross-platform, single-threaded, and interpreted compiled* programming language. It is also known as the scripting language for webpages. It is well-known for the development of web pages, and many non-browser environments also use it.

JavaScript is a **weakly typed language (dynamically typed)**. JavaScript can be used for **Client-side** developments as well as **Server-side** developments.

Python Libraries Used:

- **Pandas:**

Pandas is a software library written for the Python programming language for data manipulation and analysis.

- **Matplotlib:**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Customize visual style and layout.
- Export to many file formats.
- Use a rich array of third-party packages built on Matplotlib.

- **Seaborn:**

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Frameworks Used:

- **Django :**

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

2.2 Project Design:

The design of our Corporate Management System (CMS) encompasses a multi-tier architecture ensuring scalability, security, and efficient data flow. The system is built upon the Django framework with a robust backend and an intuitive frontend.

System Architecture:

Our CMS architecture comprises three main layers:

- 1. Presentation Layer:**

- **HTML, CSS, and JavaScript:** This layer handles the user interface, providing a seamless experience for users interacting with the system.
- **Django Templates:** Used for rendering dynamic content and facilitating user interaction.

2. Application Layer:

- **Django Framework:** Manages the application logic, processing user requests, and generating appropriate responses.
- **Python Backend:** Includes various modules and scripts responsible for handling data, user authentication, and business logic

3. Data Layer:

- **Pandas and NumPy:** Integrated for data manipulation and analysis.

System Flow:

The CMS follows a structured flow to cater to different functionalities:

1. User Authentication and Authorization:

- Users interact with the system via the frontend interface.
- Upon login, the system authenticates user credentials and grants access based on defined roles and permissions.

2. Data Management:

- Pandas libraries facilitate data manipulation for efficient storage and retrieval.

3. Analytics and Reporting:

- Matplotlib and Seaborn generate comprehensive reports and visualizations based on the stored data.
- These insights aid managers in decision-making and strategic planning.

4. Security Measures:
<ul style="list-style-type: none">• Encrypted connections and robust authentication mechanisms ensure data security.• Access controls and user permissions are implemented at both the application and database levels.
5. User Interface:
<ul style="list-style-type: none">• The frontend is designed using HTML, CSS, and JavaScript, offering a user-friendly interface for seamless navigation and interaction.• Django templates render dynamic content, providing a consistent and intuitive user experience across different sections of the CMS.

2.3 Implementation Details:

Backend Development:
The backend of our Corporate Management System (CMS) is predominantly developed using Python and Django framework. It encompasses various modules and functionalities crucial for data handling, authentication, and business logic.

User Authentication and Authorization:
<ul style="list-style-type: none">• Development of authentication mechanisms using Django's built-in authentication system.• Implementing role-based access control (RBAC) to manage user permissions and access levels.

Data Handling and Manipulation:
<ul style="list-style-type: none">• Integration of Pandas and NumPy libraries for efficient data manipulation and analysis.

- Defining data models and APIs for CRUD (Create, Read, Update, Delete) operations on records.

Frontend Development:

The frontend of the CMS is responsible for delivering a user-friendly interface and facilitating seamless interaction with the system.

1. **HTML, CSS, and JavaScript Integration:**

- Structuring web pages using HTML5 and applying CSS for styling and layout.
- Employing JavaScript for client-side functionality and user interactions, enhancing the user experience.

2. **Django Templates and Views:**

- Utilizing Django's template engine for dynamic content generation and rendering.
- Developing views to handle user requests and serve appropriate responses based on business logic.

Data Analysis and Visualization:

For generating reports and insights from the stored data, our system employs Matplotlib and Seaborn libraries for data visualization.

1. **Matplotlib Integration:**

- Creating static, animated, and interactive visualizations to represent various analytics.
- Customizing visual styles, layouts, and exporting the generated visuals in multiple formats.

2. **Seaborn for Statistical Graphics:**

- Utilizing Seaborn's high-level interface to draw informative statistical graphics.
- Generating visually appealing plots and charts to aid in decision-making and strategic planning.

Testing and Debugging:

Thorough testing and debugging procedures are implemented to ensure the reliability and functionality of the CMS.

1. **Unit Testing:**

- Writing and executing unit tests using Django's testing framework to validate individual components and functionalities.

2. Debugging and Error Handling:

- Identifying and resolving bugs, errors, and unexpected behaviour in the codebase.
- Implementing robust error handling mechanisms to gracefully manage exceptions and edge cases.

3. Results and Discussion

3.1 Project Outcomes:

- **Improved Efficiency :** Will streamline processes, reduce redundancy ; leading to increased operational efficiency.
- **Enhanced Decision Making :** Access to accurate and real-time data through the management system can empower managers to make informed decisions quickly, based on reliable information.
- **Data-driven Insights:** The system can generate comprehensive reports and analytics, providing valuable insights into the organization's performance, enabling strategic planning and improvements.
- **Better Communication and Collaboration:** A centralized system facilitates better communication and collaboration among teams and departments, breaking down silos and promoting a more cohesive work environment.

- **Scalability and Adaptability:** A robust management system can be scalable and adaptable, allowing the organization to grow and evolve without major disruptions.

3.2 Challenges Faced:

- **Understanding Business Logic:**

We often struggle to comprehend the intricate business processes and logic that need to be translated into code. Understanding how different departments within a company operate and interact is crucial for building an effective management system.

- **Learning Python and Related Frameworks:**

Newcomers may face challenges in mastering Python programming, along with libraries and frameworks relevant to corporate systems like Django or Pandas. The learning curve can be steep, particularly when applying these tools to practical business solutions.

<ul style="list-style-type: none">• Data Management Complexity:
Managing and processing vast amounts of corporate data efficiently and securely requires an understanding of database management systems (DBMS), data structures, and handling various data formats, which can be overwhelming for beginners.
<ul style="list-style-type: none">• System Integration Challenges:

Integrating different functionalities such as human resources, finance, and customer relations into a cohesive system can be complex.

- **Security Concerns:**

Ensuring data security and protecting sensitive corporate information is paramount. Novices might struggle with implementing robust security measures to safeguard the system against potential threats like data breaches or unauthorized access.

- **Testing and Debugging:**

Identifying and resolving bugs, errors, and unexpected behaviour in the code is a significant challenge for newcomers. Testing the system thoroughly to ensure its reliability and functionality often requires knowledge of testing methodologies and debugging tools.

3.3 Learnings and Insights :

Technical Skills Enhancement:

1. **Proficiency in Python and Frameworks:**

- Gained a deeper understanding of Python programming, including its libraries (such as Pandas, NumPy) and frameworks (specifically Django) for web development.

2. **Data Manipulation and Analysis:**

- Improved skills in data manipulation, analysis, and visualization using Pandas, Matplotlib, and Seaborn libraries.

3. **Database Management and ORM:**

- Enhanced knowledge of relational database management systems and learned to interact with databases using Django ORM.

Project Management and Collaboration:

1. **Team Collaboration and Communication:**

- Improved collaboration skills by working within a team, understanding diverse perspectives, and effectively communicating project objectives and progress.

2. **Adaptability and Problem-solving:**

- Developed adaptability in addressing unforeseen challenges during implementation, emphasizing problem-solving and troubleshooting skills.

Future Scope and Continuous Learning:

1. **Continuous Skill Enhancement:**

- Recognized the need for continuous learning and skill enhancement, especially in rapidly evolving technologies and frameworks relevant to corporate systems.

2. **Scope for System Refinement:**

- Identified potential areas for system refinement, such as improving user interfaces, enhancing data analytics capabilities, and refining security measures.

4.Conclusion :

The development of the Corporate Management System (CMS) has been an enlightening journey, encompassing the integration of computational thinking principles, technical expertise, and an understanding of corporate operational challenges. Our system addresses key issues such as inefficient record handling, opaque job application processes, limited employee analysis tools, and inequality in career advancement.

Through the utilization of Python, Django framework, and various libraries like Pandas, Matplotlib, and Seaborn, we successfully created a robust CMS with functionalities for efficient record management, data analysis, user-friendly interfaces, and data-driven

decision-making. The project emphasized the importance of scalable architecture, data security, and seamless system integration to cater to the complexities of modern corporate environments.

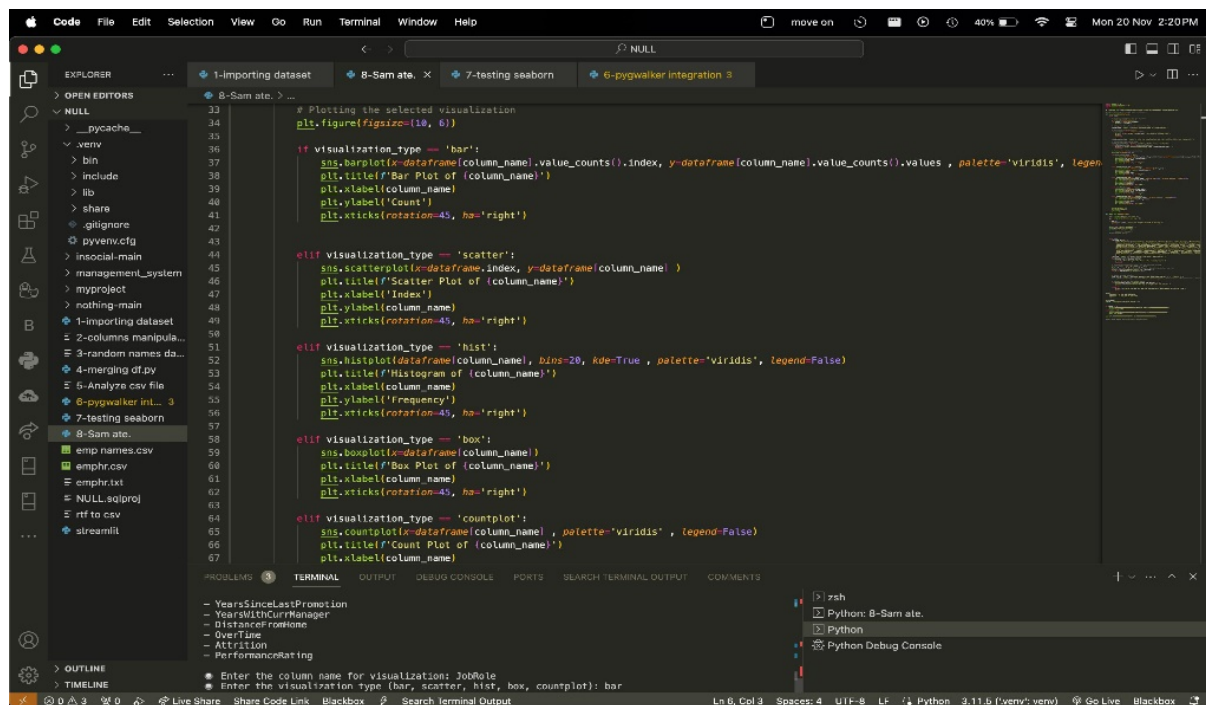
5. References:

Data collected from Kaggle:

<https://www.kaggle.com/datasets/patelprashant/employee-attrition>

6. Appendices :

Visualizing Data using Pandas, Matplotlib and Seaborn



```
1-Importing dataset
2-columns manipula...
3-random names da...
4-merging df by
5-Analyse csv file
6-pyqwalker int...
7-testing seaborn
8-Sam ate.
emp names.csv
empplr.csv
empplrproj
rtf to csv
streamill

# Plotting the selected visualization
plt.figure(figsize=(10, 6))

if visualization_type == 'bar':
    sns.barplot(dataframe[column_name].value_counts().index, y=dataframe[column_name].value_counts().values, palette='viridis', legend=False)
    plt.title(f'Bar Plot of {column_name}')
    plt.xlabel(column_name)
    plt.ylabel('Count')
    plt.xticks(rotation=45, ha='right')

elif visualization_type == 'scatter':
    sns.scatterplot(dataframe.index, y=dataframe[column_name])
    plt.title(f'Scatter Plot of {column_name}')
    plt.xlabel('Index')
    plt.ylabel(column_name)
    plt.xticks(rotation=45, ha='right')

elif visualization_type == 'hist':
    sns.histplot(dataframe[column_name], bins=20, kde=True, palette='viridis', legend=False)
    plt.title(f'Histogram of {column_name}')
    plt.xlabel(column_name)
    plt.ylabel('frequency')
    plt.xticks(rotation=45, ha='right')

elif visualization_type == 'box':
    sns.boxplot(dataframe[column_name])
    plt.title(f'Box Plot of {column_name}')
    plt.xlabel(column_name)
    plt.xticks(rotation=45, ha='right')

elif visualization_type == 'countplot':
    sns.countplot(dataframe[column_name], palette='viridis', legend=False)
    plt.title(f'Count Plot of {column_name}')
    plt.xlabel(column_name)

# Enter the column name for visualizations JobRole
# Enter the visualization type (bar, scatter, hist, box, countplot): bar
```

```
Code File Edit Selection View Go Run Terminal Window Help
move on 37% Mon 20 Nov 2:34PM

EXPLORER
> OPEN EDITORS
> NULL
> __pycache__
> .venv
> bin
> include
> lib
> share
> zignore
> pyvenv.cfg
> insocial-main
> management_system
> myproject
> nothing-main
1-importing dataset
2-columns manipula...
3-random names da...
4-merging df.py
5-Analyze csv file
6-pygwalker int... 3
7-testing seaborn
8-Sam ale.
emp_names.csv
empmtr.csv
empmtr.bst
NULL_solproj
rif to csv
streamlit

1-importing dataset
2-columns manipula...
3-random names da...
4-merging df.py
5-Analyze csv file
6-pygwalker int... 3
7-testing seaborn
8-Sam ale.
emp_names.csv
empmtr.csv
empmtr.bst
NULL_solproj
rif to csv
streamlit

def apply_for_job(dataframe):
    name = input("Enter your name: ")
    age = int(input("Enter your age: "))

    if age < 18:
        print(f"(name), you are not eligible to work at this age.")
        return

    print("\nList of job roles: ")
    print(df["jobrole"].unique())
    #for column in dataframe.columns:
    #    print(f"{column}")

    #print(job_role_skills)
    while True:
        job_role_skills = [
            'Sales Executive': ['Communication', 'Negotiation', 'Customer Service', 'Sales Strategy', 'Relationship Building', 'Market Analysis',
                                'Research Scientist': ['Research Design', 'Data Analysis', 'Lab Techniques', 'Statistical Analysis', 'Critical Thinking', 'Attention
            'Laboratory Technician': ['Lab Techniques', 'Attention to Detail', 'Analytical Skills', 'Critical Thinking', 'Time Management', 'Org
            'Manufacturing Director': ['Operations Management', 'Supply Chain Optimization', 'Quality Control', 'Team Leadership', 'Strategic Pl
            'Healthcare Representative': ['Medical Product Knowledge', 'Sales', 'Customer Service', 'Relationship Building', 'Communication', 'h
            'Manager': ['Leadership', 'Decision Making', 'Team Management', 'Communication', 'Problem Solving', 'Strategic Thinking', 'Adaptabil
            'Sales Representative': ['Sales', 'Communication', 'Customer Service', 'Negotiation', 'Product Knowledge', 'Relationship Building',
            'Research Director': ['Research Strategy', 'Project Management', 'Leadership', 'Data Analysis', 'Critical Thinking', 'Communication'
            'Human Resources': ['Recruitment', 'Employee Relations', 'Conflict Resolution', 'Communication', 'Strategic Planning', 'Organization
        ]

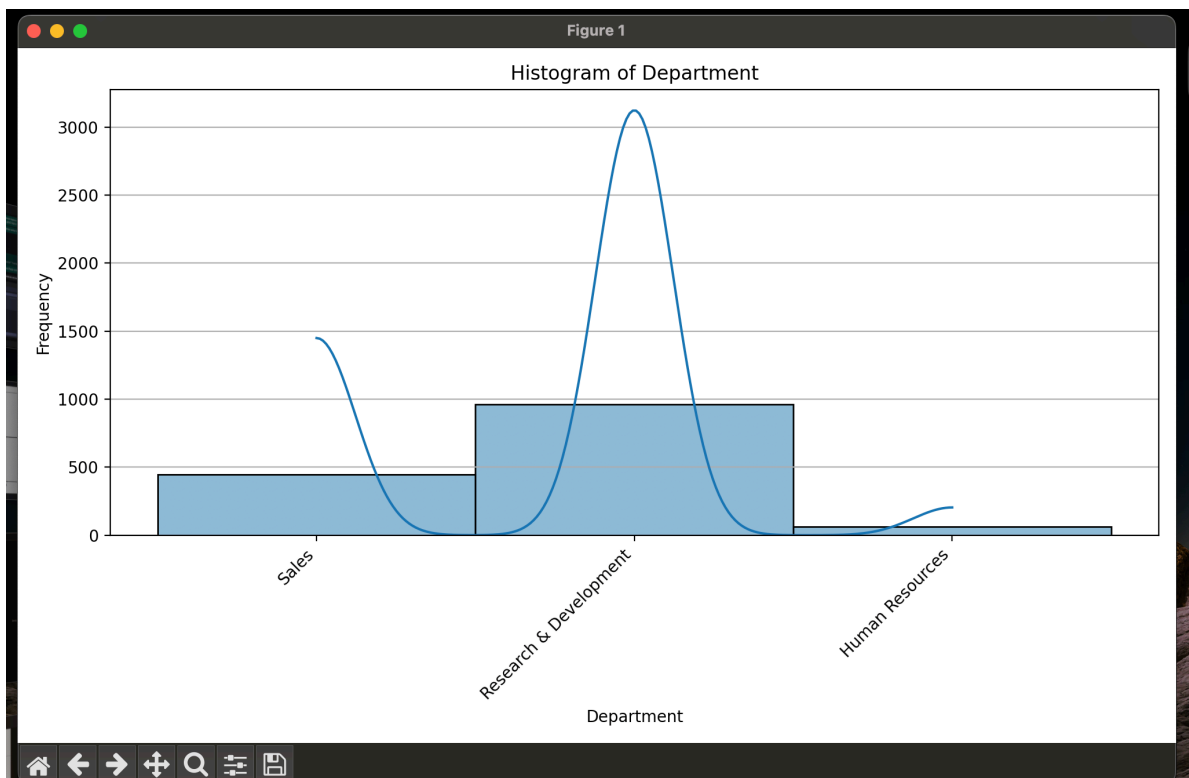
        job_role = input("Enter the job role you are applying for: ")
        # Checking if the job role is valid
        if job_role not in job_role_skills:
            print("\nInvalid job role. Please enter a valid job role.")
            continue

        # Displaying pre-requisite skills for the specified job role
        pre_req_skills = job_role_skills[job_role]
        print("pre_req = ", pre_req_skills, "type=", type(pre_req_skills), sep="\n")
        print("pre_req_skills = ", pre_req_skills, "type=", type(pre_req_skills), sep="\n")

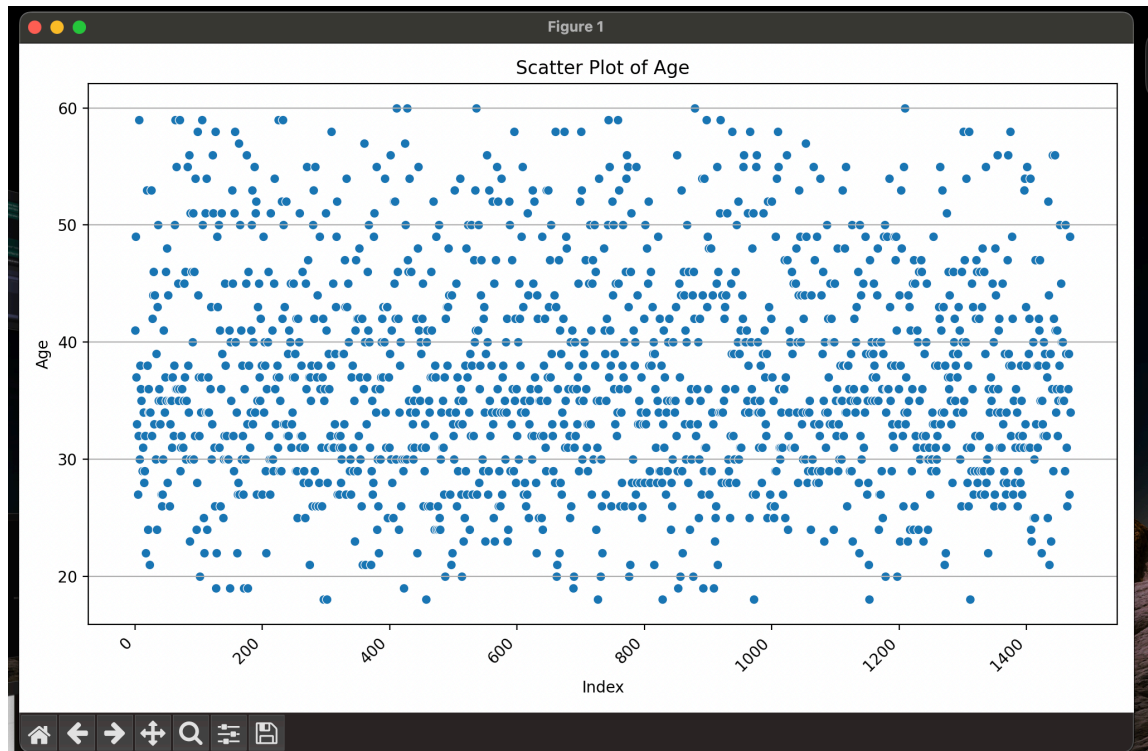
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS SEARCH TERMINAL OUTPUT COMMENTS
Python + Python 3.11.5 (.venv:venv)

Enter your age: 18
List of job roles:
['Sales Executive', 'Research Scientist', 'Laboratory Technician',
 'Manufacturing Director', 'Healthcare Representative', 'Manager',
 'Sales Representative', 'Research Director', 'Human Resources']
Enter the job role you are applying for: 
```

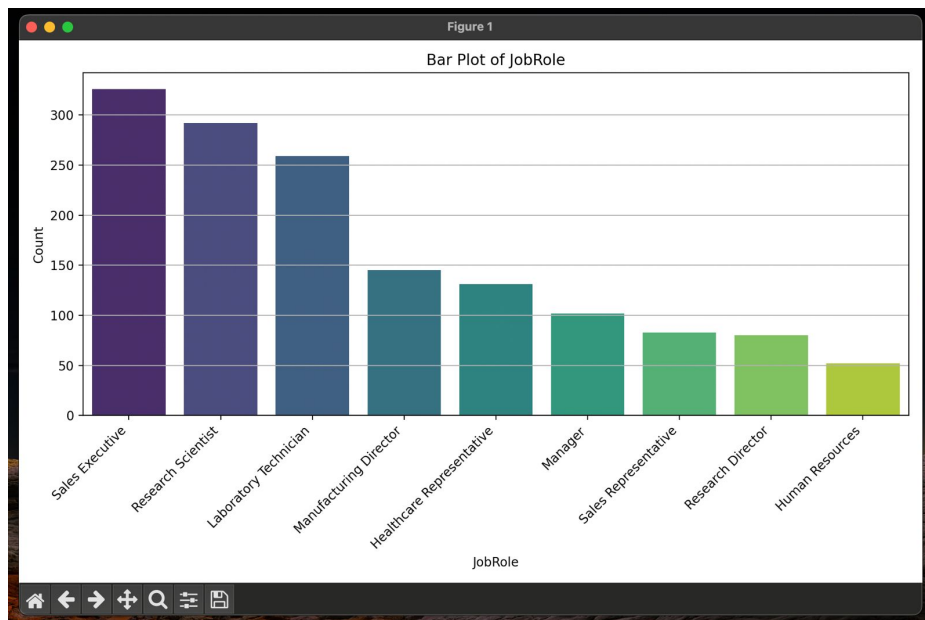
Histogram



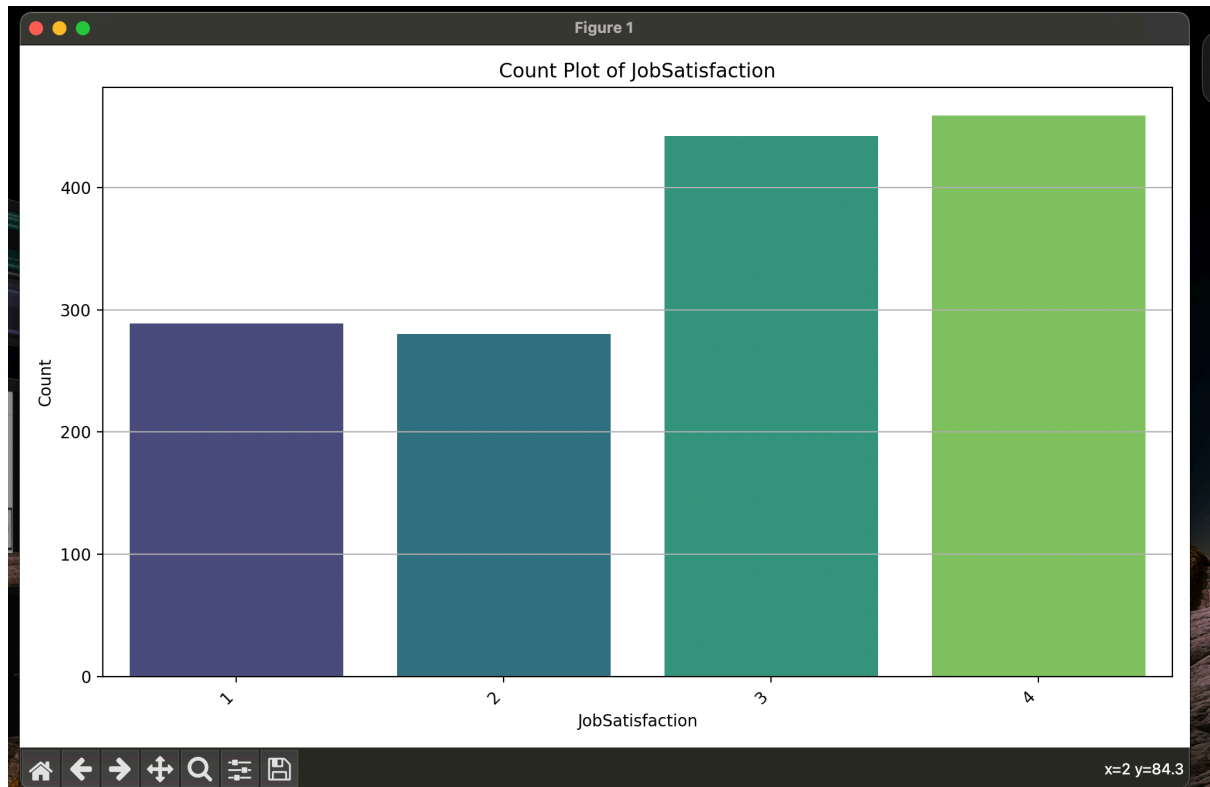
Scatterplot



Bar Graph

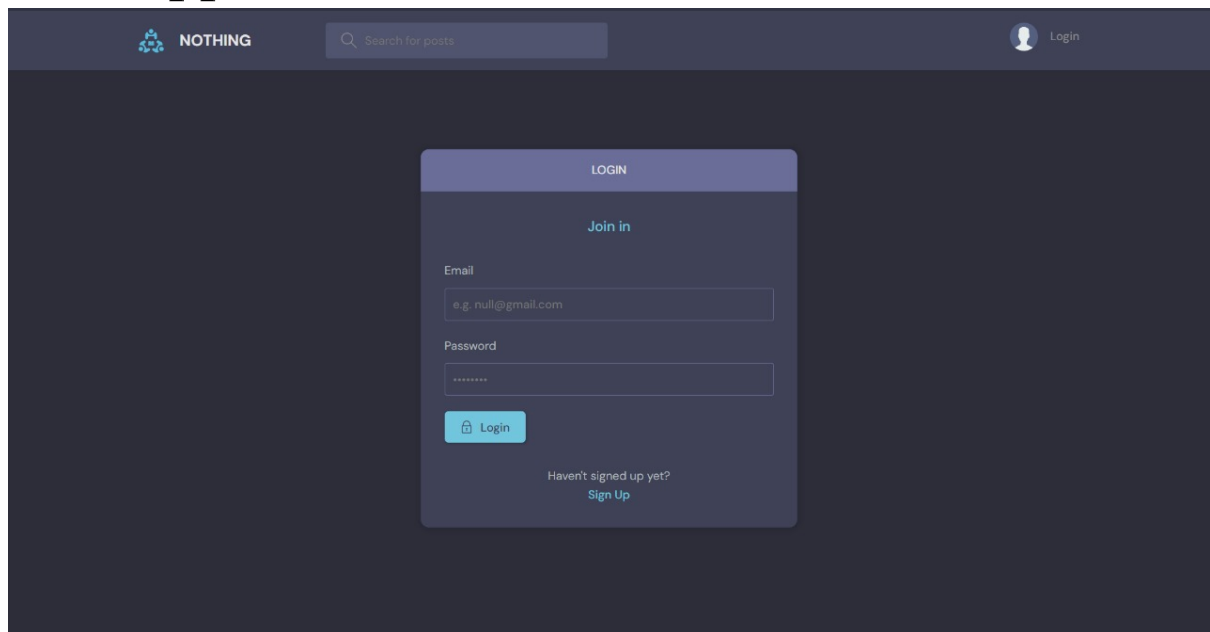


CountPlot

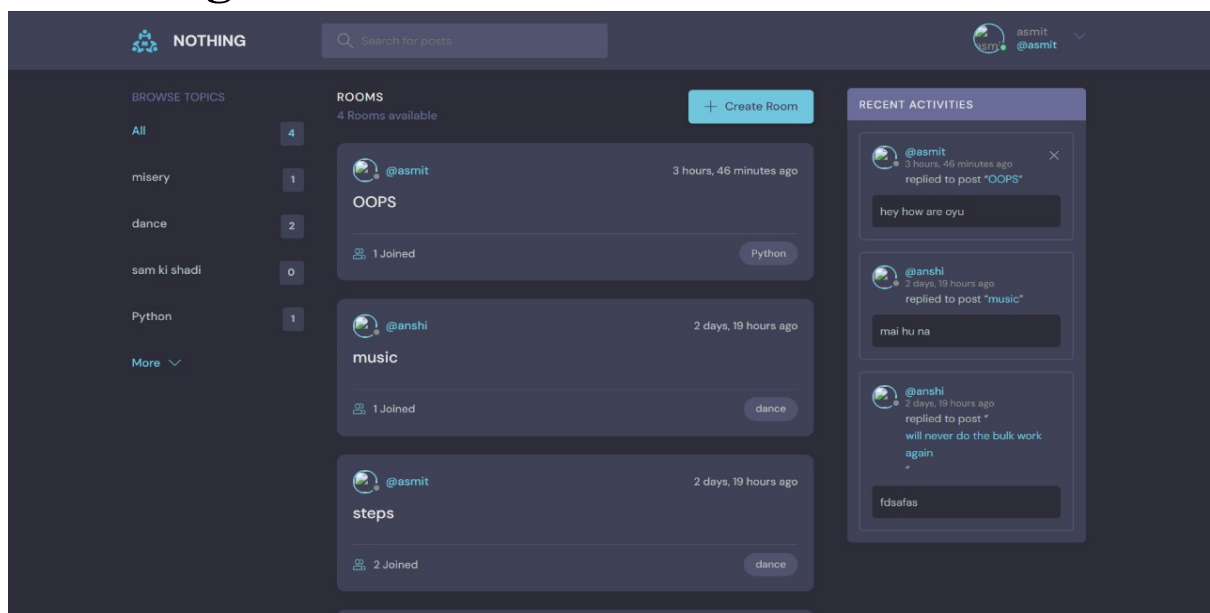


Our main website for our management system with a few code snippets :

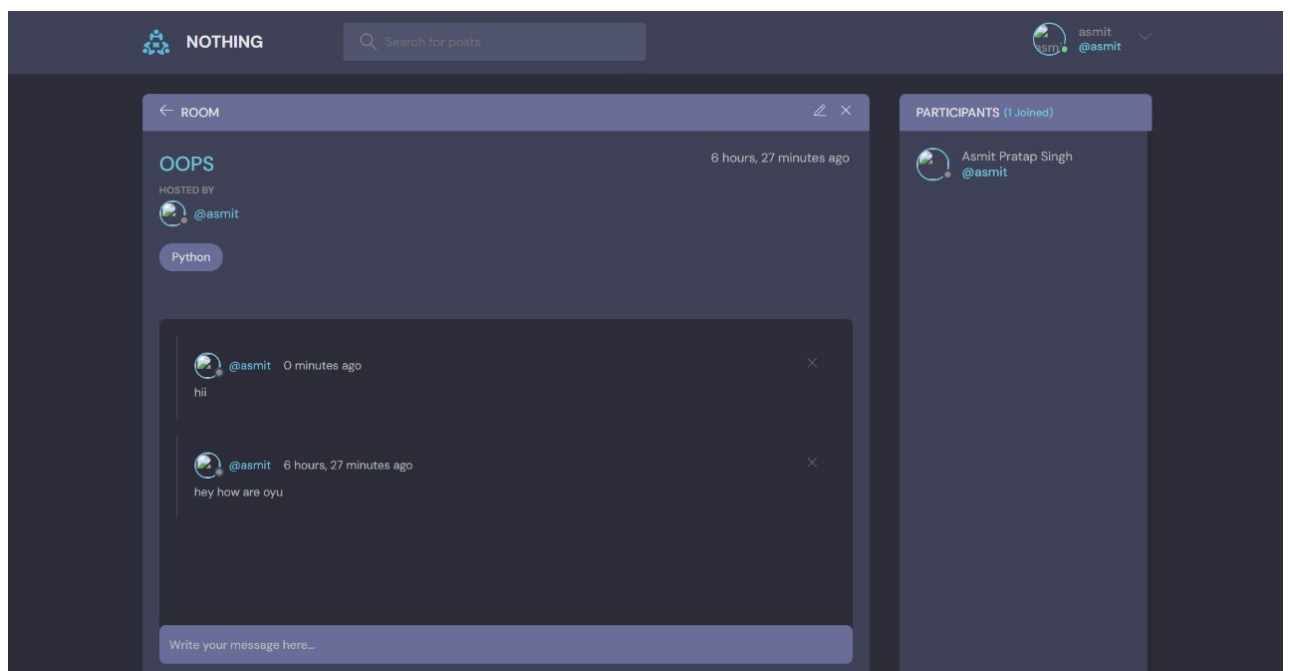
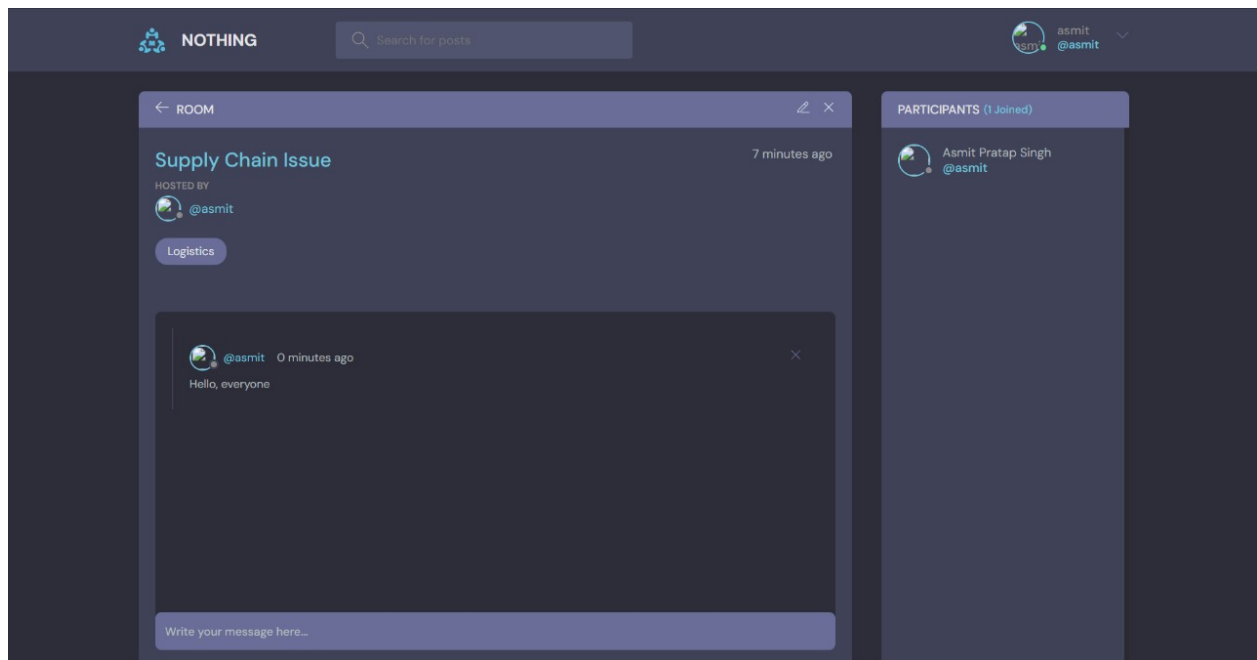
Web Application:



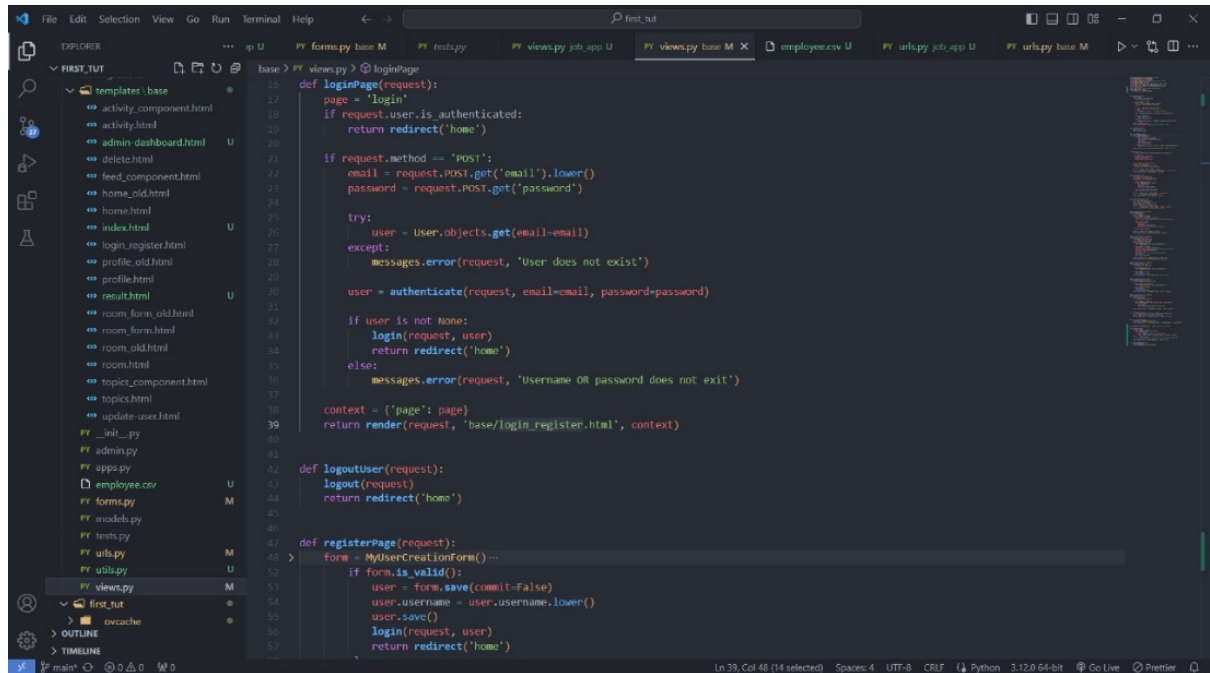
Main Page:



Discussion Rooms:



Django code:



```
def loginPage(request):
    page = 'login'
    if request.user.is_authenticated:
        return redirect('home')

    if request.method == 'POST':
        email = request.POST.get('email').lower()
        password = request.POST.get('password')

        try:
            user = User.objects.get(email=email)
        except:
            messages.error(request, 'User does not exist')

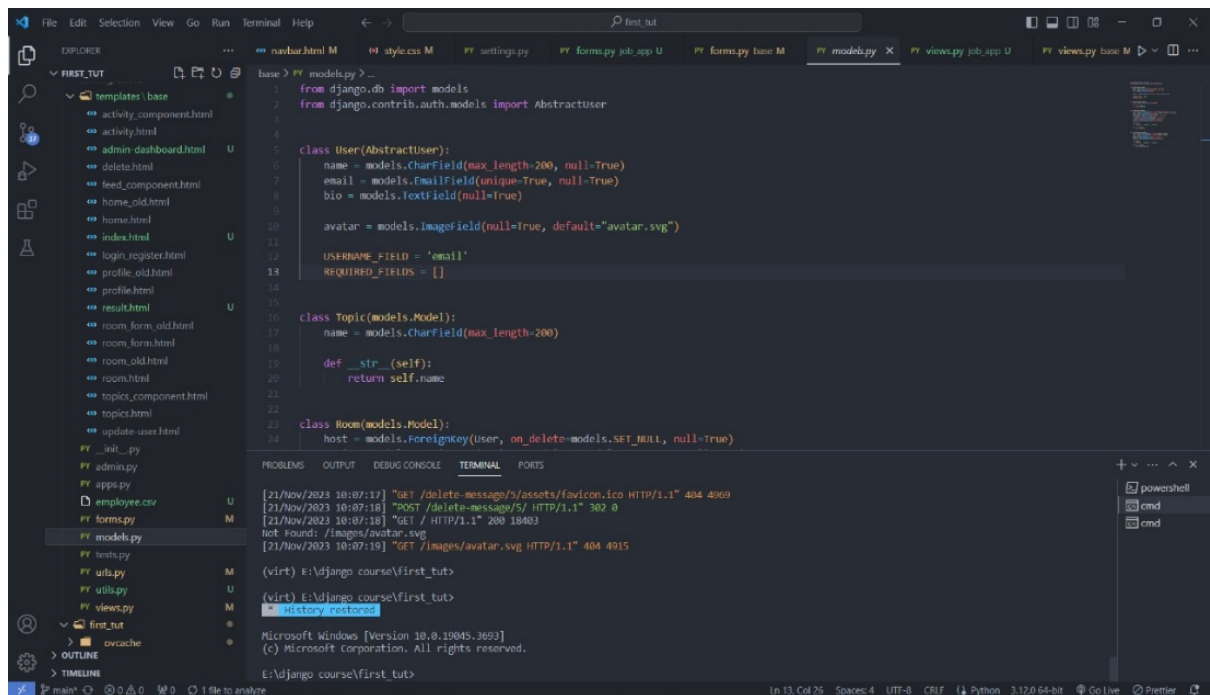
        user = authenticate(request, email=email, password=password)

        if user is not None:
            login(request, user)
            return redirect('home')
        else:
            messages.error(request, 'Username OR password does not exit')

    context = {'page': page}
    return render(request, 'base/login_register.html', context)

def logoutUser(request):
    logout(request)
    return redirect('home')

def registerPage(request):
    form = MyUserCreationForm()
    if form.is_valid():
        user = form.save(commit=False)
        user.username = user.username.lower()
        user.save()
        login(request, user)
        return redirect('home')
```



```
from django.db import models
from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
    name = models.CharField(max_length=200, null=True)
    email = models.EmailField(unique=True, null=True)
    bio = models.TextField(null=True)

    avatar = models.ImageField(null=True, default="avatar.svg")

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = []

class Topic(models.Model):
    name = models.CharField(max_length=200)

    def __str__(self):
        return self.name

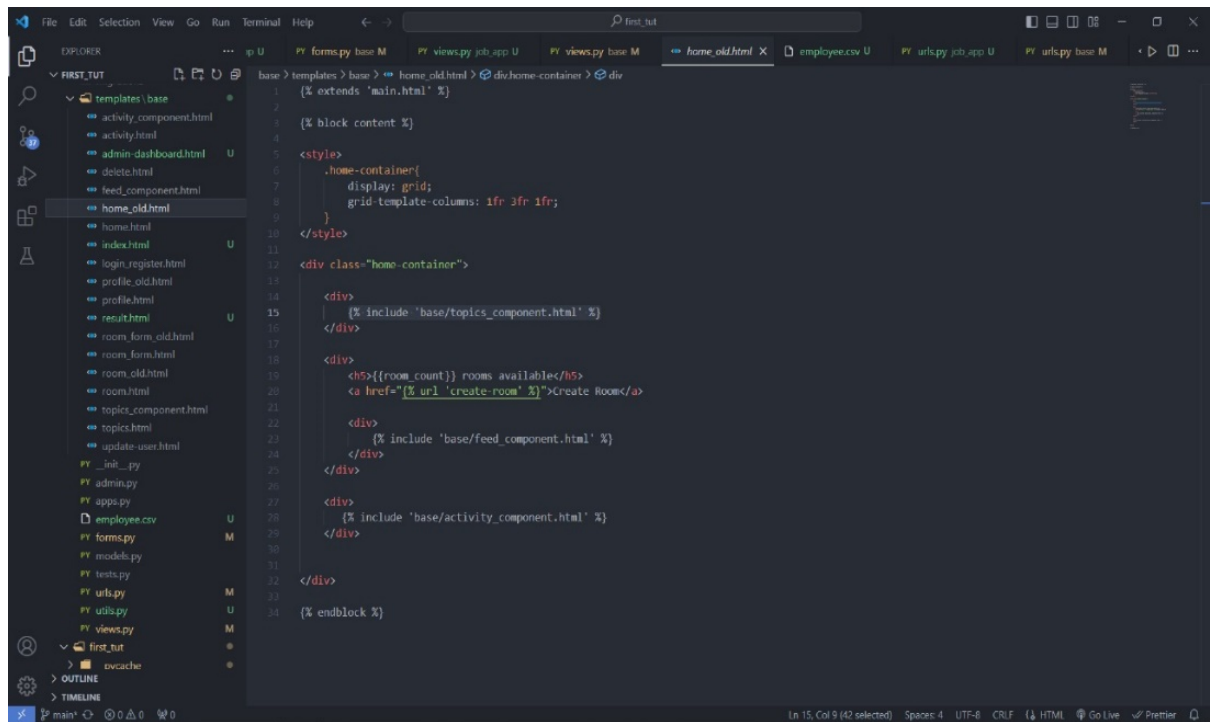
class Room(models.Model):
    host = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[21/Nov/2023 10:07:17] "GET /delete-message/5/assets/favicon.ico HTTP/1.1" 404 4069
[21/Nov/2023 10:07:18] "POST /delete-message/5/ HTTP/1.1" 302 0
[21/Nov/2023 10:07:18] "GET / HTTP/1.1" 200 18403
lock round: /images/avatar.svg
[21/Nov/2023 10:07:19] "GET /images/avatar.svg HTTP/1.1" 404 4015

(virt) E:\django course\first_tut>
(virt) E:\django course\first_tut>
History restored
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

E:\django course\first_tut>
```



THE END