CSCE 5250 – Fundamentals of Artificial Intelligence

Final Project Code and Report

AI-Driven Roommate Matching Platform

Group-23

University of North Texas

Team Members:

Leela Krishna Reddy Gudibandi
Trivikram Cheluri
Arthi Reddy Kota
Medha Sai Sigatapu
Sirichandana Reddy Katta

1. Abstract

An AI based roommate matching system that seeks to solve one of the most demanding tasks that is identifying suitable roommate. Developed on the proper Django platform, the system calculates compatibility concerns in relation to such parameters as a lifestyle, cleanness, social and fiscal behavior, etc. Regarding the matching process, the platform is constructed to automatically perform the searching and matching, improving the user experience and accessibility.

The architecture of the system is modular, consisting of three distinct applications: the Main App for user authentication, the Profile App for managing user data, Algorithm App for processing and calculating match scores. To put it in a more simple way, this design makes it possible to extend it and make further changes in the future, while being safe and easy to navigate. All of them are server-side applications that serve the platform and provide clients with an opportunity to perform various actions effectively. The users hold the authority to edit, preview, and decide on their profile and even the Apply/matches based on an AI recommendation. This approach not only increases the number of satisfied users but also greatly decreases the amount of time and energy that has been required for a simple roommate search in the past. The report is a detailed description of the general architecture of the system, its current performance indicators, and potential enhancements of the matching algorithm.

The platform is also built in such a way that it is also easy to use with a simple user interface and the privacy of a user is kept safe when using the platform. This dedication to security is essential in ensuring that it gains the trust of those who are in the use of the platform and make them engage in the platform with full confidence. The report also includes recommendations about future changes which may be effective to strengthen the system's effectiveness as well as user satisfaction.

2. Introduction

The search for the perfect roommate mostly takes a lot of time to most people; the three pieces are as follows: There are questionnaires, interviews or turning to third-party services which can be stressful and time-wasting in old ways. As people look for housemates, they basic needs and their housing choices bring issues of compatibility hence people end up chasing people that they do not like and, in most cases, they end up disliking the people they live with.

The first key challenge of manual roommate matching is therefore the inevitable bias that accompanies the process of evaluating compatibility. It may be hard to measure several factors like equivalent cleanliness, social etiquettes, and levels of financial responsibility which has made it altogether difficult for individuals to secure perfectly compatible partners. These are challenging that this platform recognizes and for the matching of compatible room-mates as well as for data accumulation for the right decision it recommends the use of artificial intelligence and machine learning.

According to Rental Buddy (2024), the goal of this AI platform is to establish a similar platform that can be easily used for a large number of people where the matching is done by using complex algorithms for roommate compatibility scores. By adopting an organized and logical structure of the system, the platform guarantees the proper usability of the matching strategy and the interaction with the system in general. Also, the protection of users' private information, as well as using authorization for the possibility of program utilities. To achieve these objectives, the system is divided into three main components: the Main App, which handles user authentication and security; the Profile App, where users can create and modify their profiles; and the Algorithm App, responsible for processing user data and calculating compatibility scores. However, such an organization does not only help improve the performance of the system but also for future developments and enhancements to the system.

3. Area of Applications/Features

Overview of the System

System Architecture

The AI-driven roommate matching platform relies on a well-structured and efficient architecture built using the Django framework, databases, and the k-nearest neighbors (KNN) algorithm to calculate compatibility between users. The platform is divided into three main applications: the Main App, Profile App, and Algorithm App, each serving a distinct purpose in the matching process. The combination of Django, relational databases, and the KNN algorithm allows for scalability, security, and accuracy in delivering optimal roommate matches.

The system is divided into three interdependent apps:

Main App: This app deals with user login, signing up, and managing user's accounts. This is the initial point where the user must get to interact with the platform. The Main App includes secure login/logout functionalities and session management to ensure that only authenticated users can access their profiles and interact with the system.

Profile App: These aspects of the application encompass user's profile where key aspects like, preferred lifestyles, cleanliness, and financial roles. Profiles of its users which include; creation profiles, editing profiles, and viewing profile can be performed in this app. The Profile App plays significant roles in collecting data since the information from the users' profiles is necessary for creating compatibility scores for the matching algorithm. Each profile is linked to the database, where the information is stored securely for future reference.

Algorithm App: This app is to determine compatibility of users using the KNN algorithm in determining compatibility. The Algorithm App retrieves data from the Profile App and applies machine learning techniques to find the best possible matches for each user. The profile data is received, computed by the algorithm and the app returns a list of potential roommates ranked by compatibility.

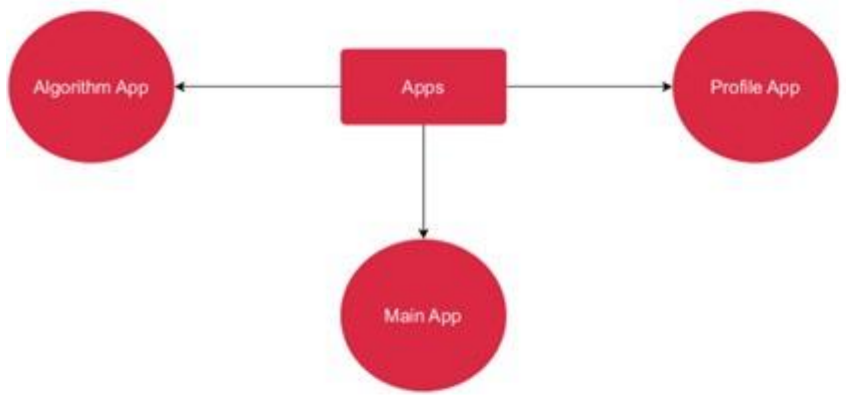


Fig: 1

4. **Methods**

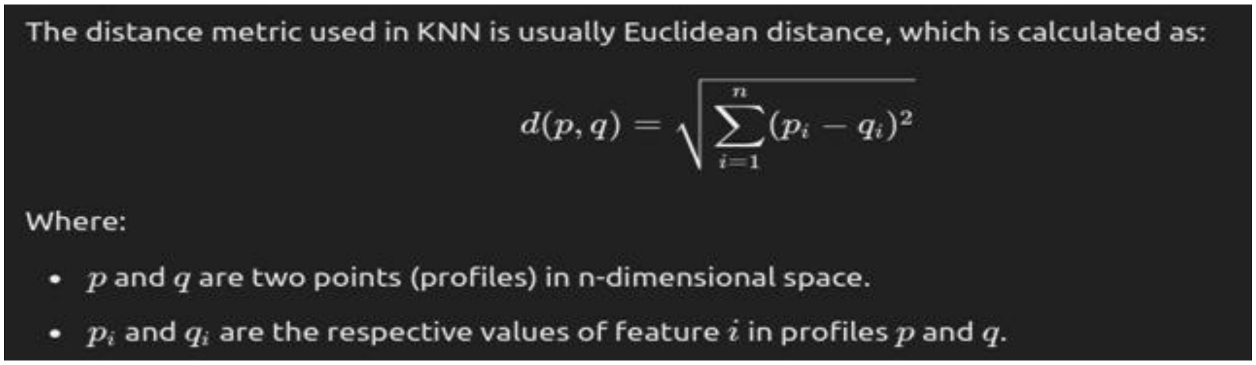
The technology choice for this platform is done with a great attention to the possibilities of integration between the front-end part and the back-end part of the further created application. Each technology plays a specific role in handling different aspects of the system's architecture:

Django Framework: Django was chosen for its robustness, scalability, and built-in features such as user authentication, security, and ORM (Object-Relational Mapping) for database interactions. Django will be useful in constructing a secure web application since it is composed of several sub-components that makes it easy to use. It handles the development of the Main App, Profile App, and Algorithm App,

PostgreSQL (Database): A relational database like PostgreSQL was chosen for its ability to handle complex queries, maintain data integrity, and scale efficiently. The database is used to store user profile data, login credentials, and matching scores. Since the KNN algorithm relies on a dataset (i.e., user profiles) to calculate matches, the database must allow for fast retrieval and updating of data.

KNN Algorithm (Machine Learning): The KNN algorithm is used as the main algorithm in the matchmaking process of the applications. This algorithm was selected because of its simplicity, flexibility and ability to provide strong indications of users similarity based on several parameters. In the case of this platform, KNN employs profile data to be utilized as features and measures the shortest distance apart between users.

The universal algorithm of the KNN approach is based upon its proximity (or similarity) of the data points in multidimensional space. In this system, the "neighbors" represent users with similar lifestyle preferences, and the "distance" between these users indicates how well they match.


The distance metric used in KNN is usually Euclidean distance, which is calculated as:

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

Where:

- p and q are two points (profiles) in n-dimensional space.
- p_i and q_i are the respective values of feature i in profiles p and q.

Fig: 2

<u>Applying KNN in Django</u>

<u>Data Retrieval</u>: The user profile data is retrieved from the PostgreSQL database using Django's ORM. It is then preprocessed, normalized and transformed into a form suitable for input into the KNN algorithm.

Algorithm Execution: After preparation of the data, KNN algorithm is performed. The algorithm in question was implemented using the scikit-learn library; it selects the current user and calculates distances between the latter and all other users in the dataset. The KNeighborsClassifier or the KNeighborsRegressor depending on how match is to be defined as a classification or regression problem. Matching Calculation: The algorithm provides a set of ordered suggestions of possible roommates. The results are then passed back to the user interface where the user sees his or her match results and a calculated compatibility rating. Continuous Updates: Every time a user changes something in the profile, the data is processed again, as well as the KNN algorithm to display the latest matching scores.

Key Features:

KNN Algorithm Execution: It takes user profile information and using the KNN algorithm determines the nearest neighbors by several characteristics.

Compatibility Score Calculation: The app then computes the compatibility of two users so that the people get to know how compatible they are in terms of preference.

Real-Time Matching: Whenever a user updates his or her profile, then the matching algorithm is triggered, guaranteeing that the compatibility scores are current.

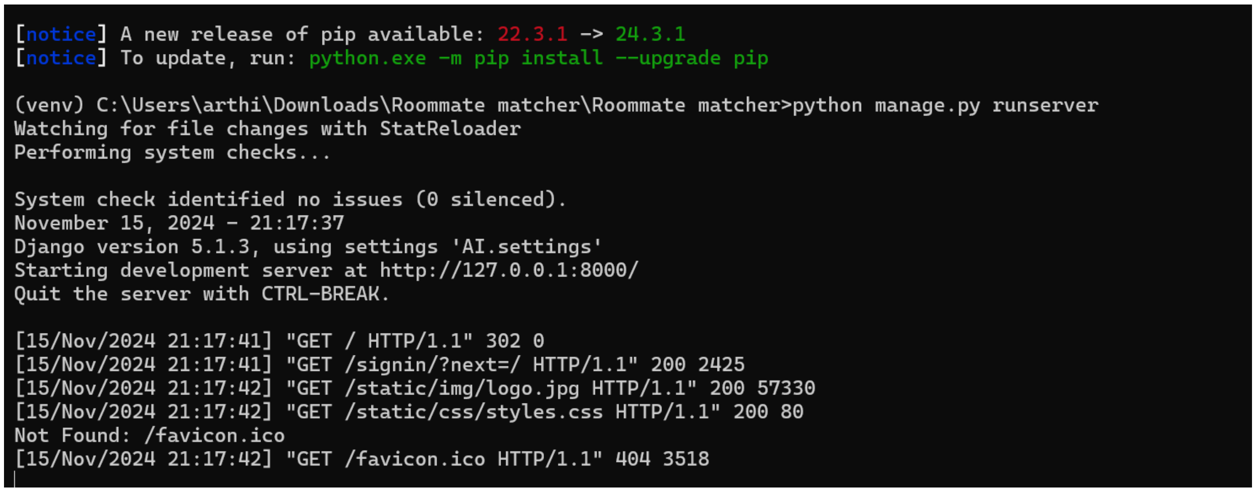
<u>Performance Analysis</u>

The platform evaluates the performance of the KNN algorithm using several key metrics: includes accuracy, precision, recall, and time of execution. Accuracy simply quantifies how well the algorithm performs in ranking groups that have a high-input compatibility, while precision and recall give the degree of relevance and encompassing of the groups identified. These are calculated based on the difference between the computer derived probability and actual user satisfaction or feedback after roommate matches have been made.

A final performance factor is proximity, the scalability of the algorithm. When more users register, the function's time complexity rises because the message will have to calculate distances between more users. To counter this, there is an option of using optimizations like KD-trees or Ball-trees which in a way will help to ease the computation hence making the algorithm to run efficiently without necessarily having to further a distance more than what is required.

5. Experimental Results

Working of the proposed AI based roommate matching system in the system environment. With all the satisfied libraries the server is running on windows with the help of the port number using the url: http://127.0.0.1:8000/ as shown in the image.



```
[notice] A new release of pip available: 22.3.1 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(venv) C:\Users\arthi\Downloads\Roommate matcher\Roommate matcher>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
November 15, 2024 - 21:17:37
Django version 5.1.3, using settings 'AI.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[15/Nov/2024 21:17:41] "GET / HTTP/1.1" 302 0
[15/Nov/2024 21:17:41] "GET /signin/?next=/ HTTP/1.1" 200 2425
[15/Nov/2024 21:17:42] "GET /static/img/logo.jpg HTTP/1.1" 200 57330
[15/Nov/2024 21:17:42] "GET /static/css/styles.css HTTP/1.1" 200 80
Not Found: /favicon.ico
[15/Nov/2024 21:17:42] "GET /favicon.ico HTTP/1.1" 404 3518
```

Fig: 3

This the login page to start with either login if account is already added or register if not as shown in the image.

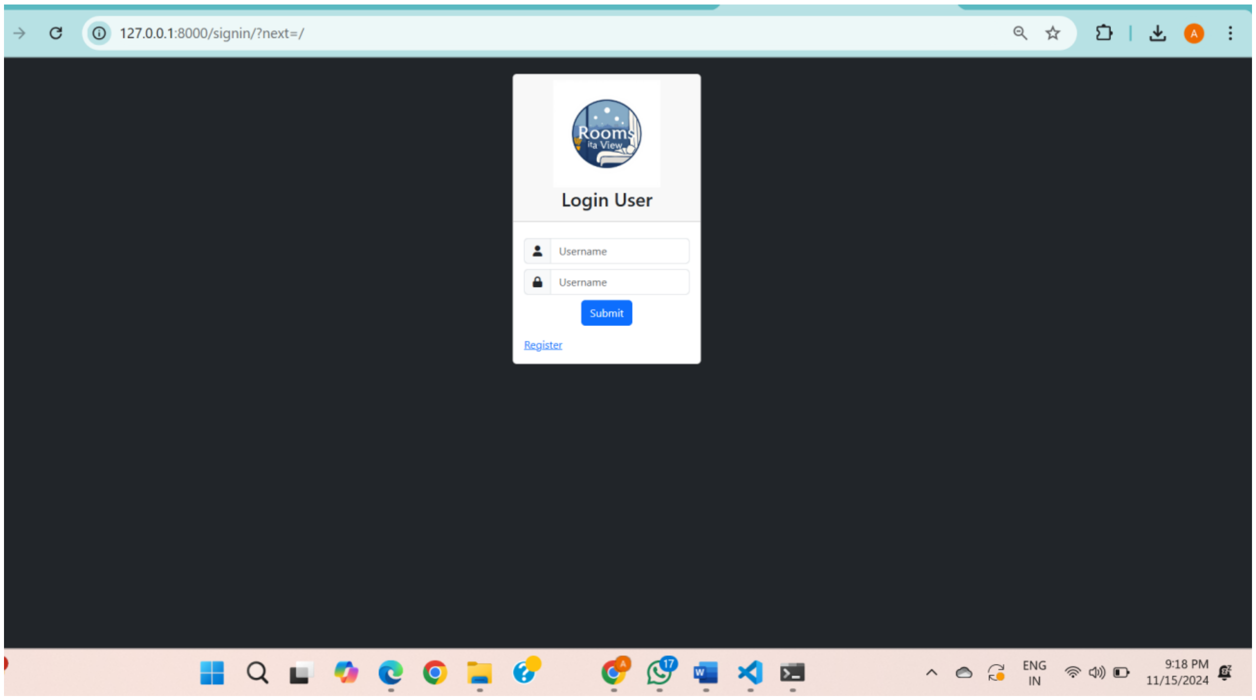


Fig:4

Since we do not have an account already, we have to register by giving the below shown details and once registration is done the details are saved to login after every logout. Below are the images that shows the details are given and account is created.

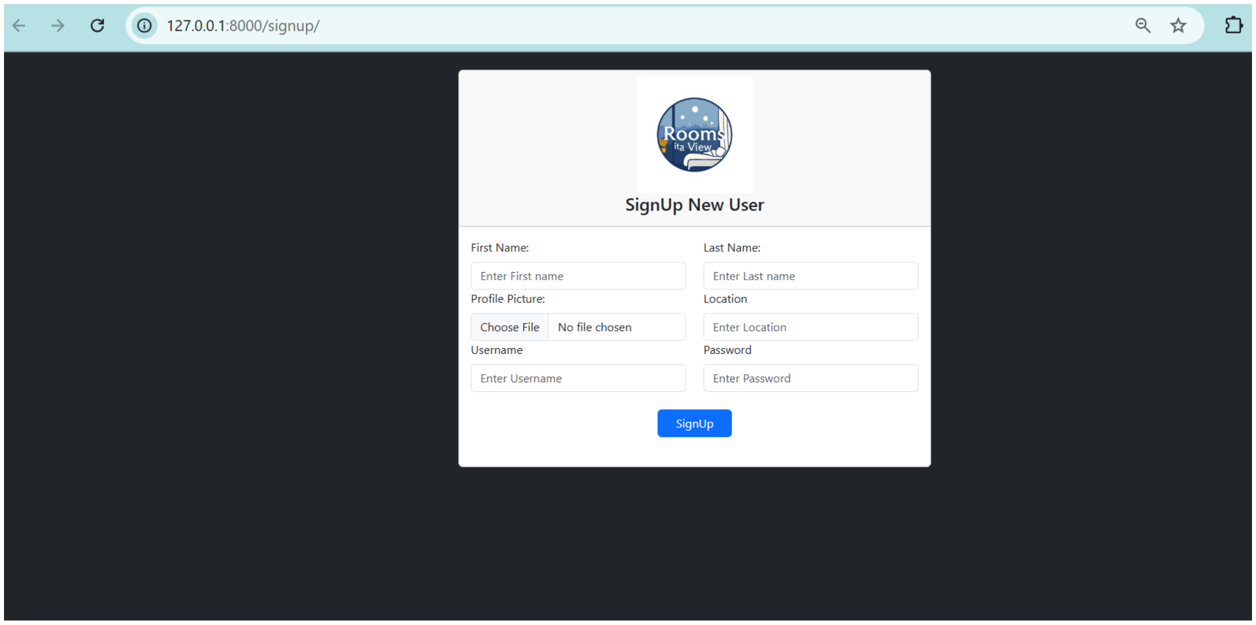
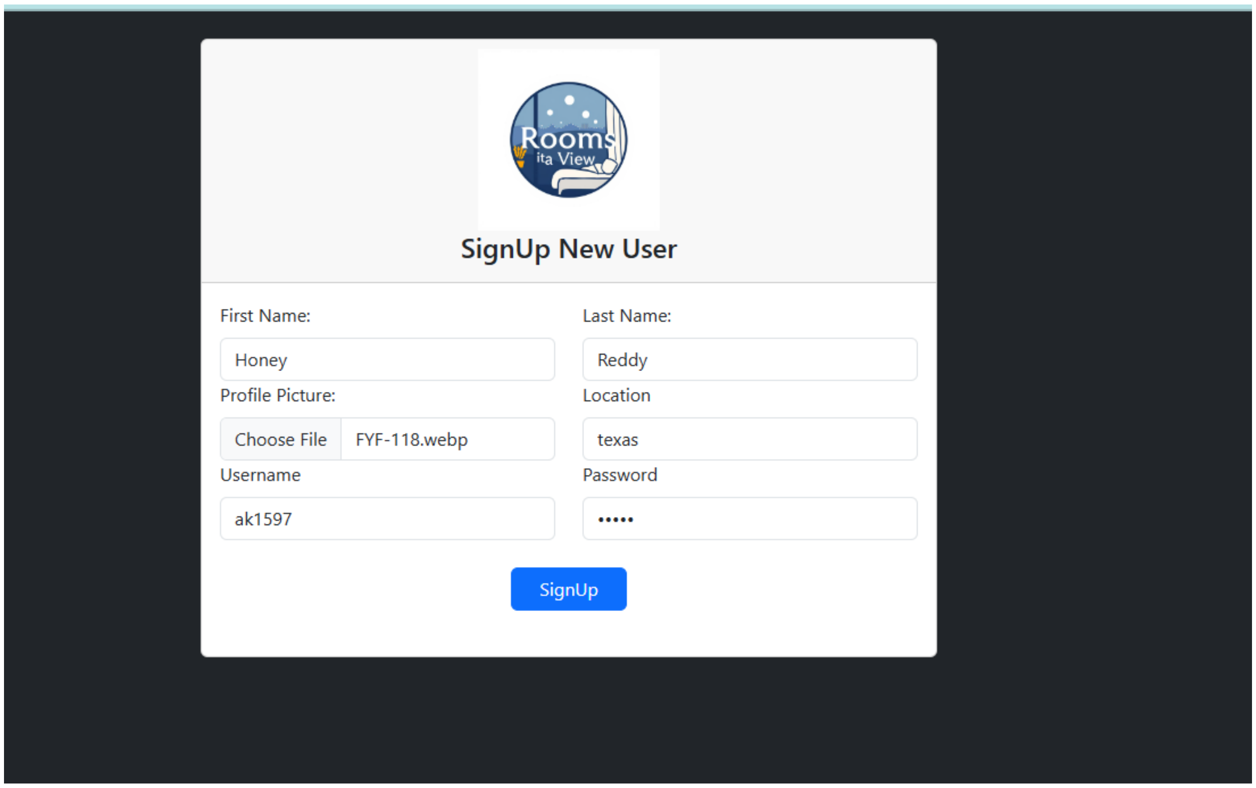


Fig:5



SignUp New User

First Name: Honey

Last Name: Reddy

Profile Picture: Choose File FYF-118.webp

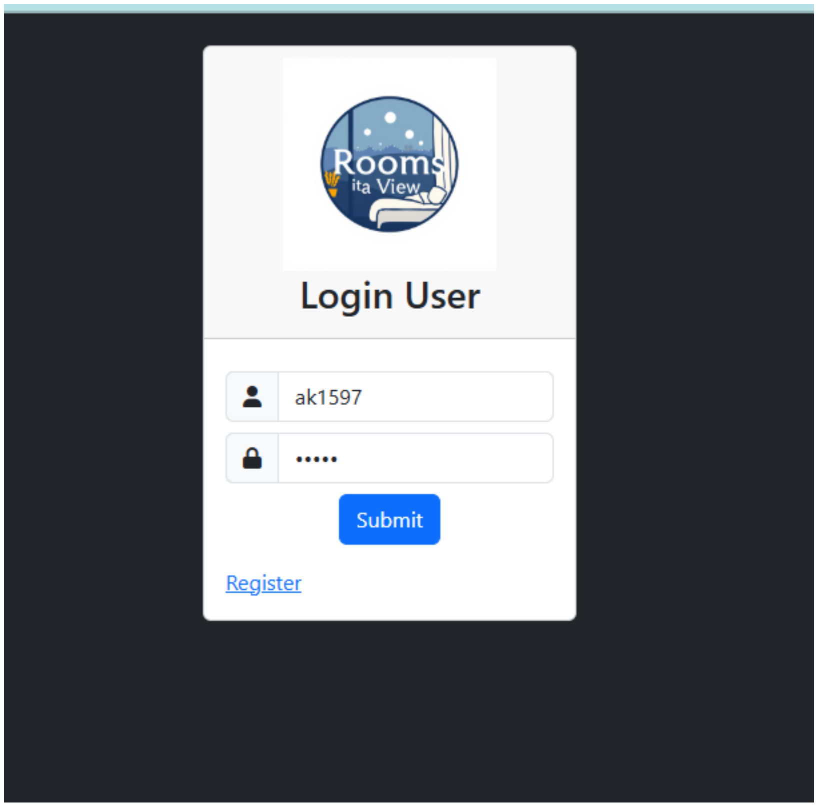
Location: texas

Username: ak1597

Password: •••••

Fig:6

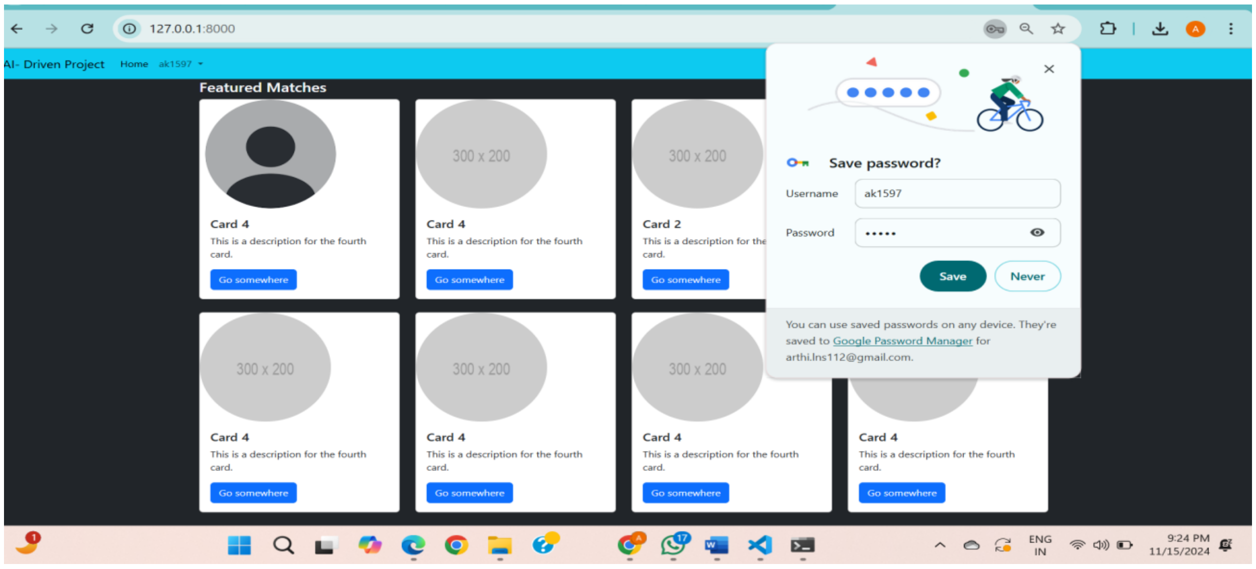
After the account is created we have to login and there is the proof of login credentials from the google as shown in the below images.



ak1597

•••••

Fig:7



Card 4
This is a description for the fourth card.

Card 2
This is a description for the fourth card.

Save password?

Username ak1597

Password •••••

You can use saved passwords on any device. They're saved to Google Password Manager for arthi.lns112@gmail.com.

Fig:8

On the home page we can see the profile and there are some options in the profile as shown in below image.

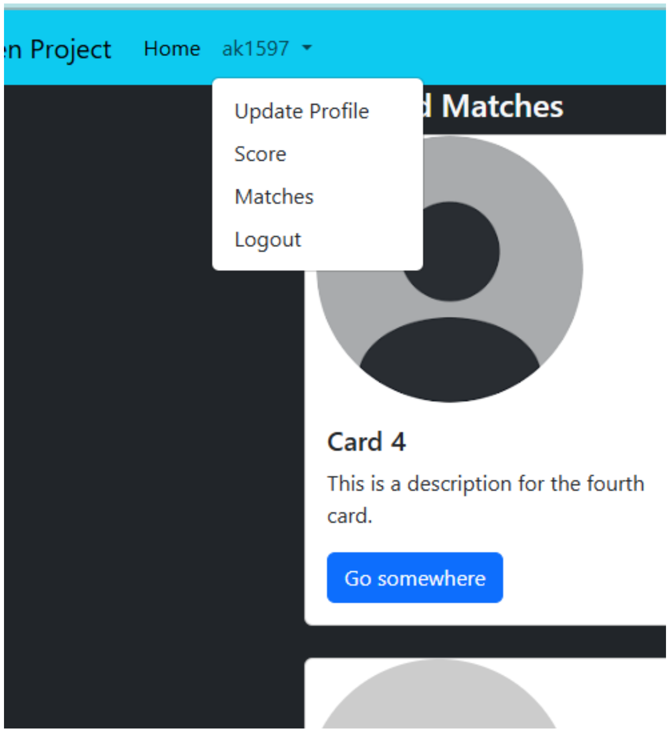


Fig:9

When you click on update profile it will give the options to update the below given options as shown in image.

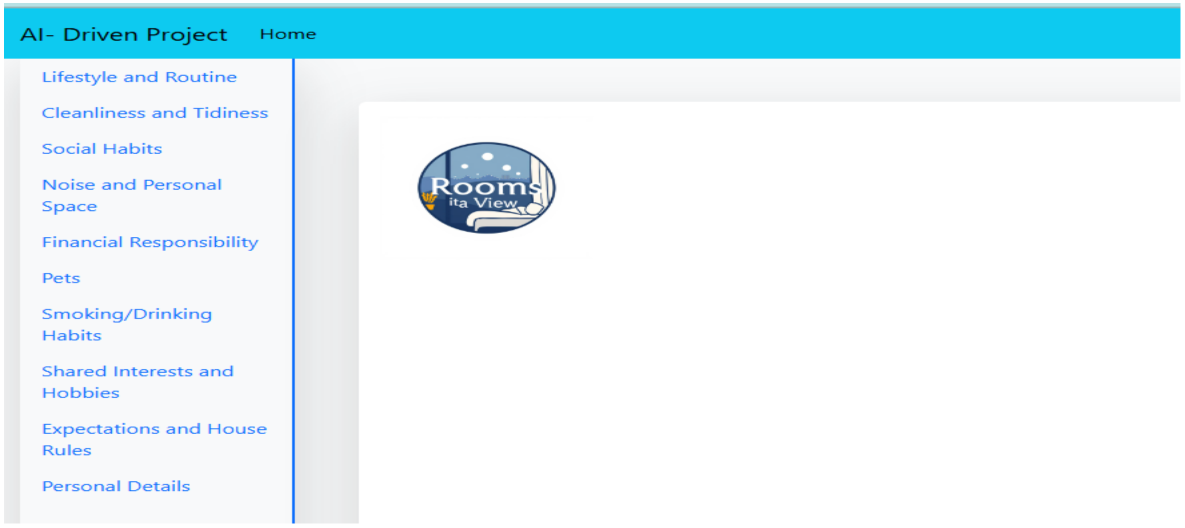
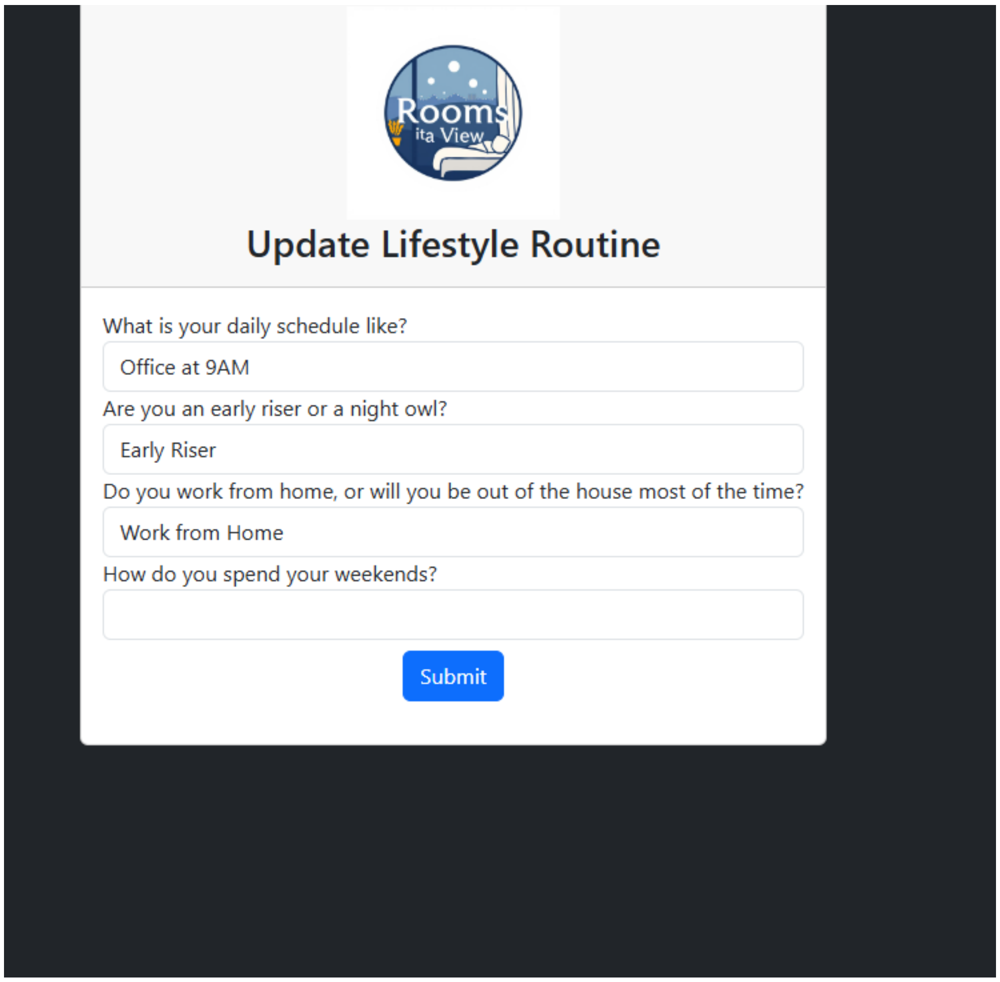


Fig:10

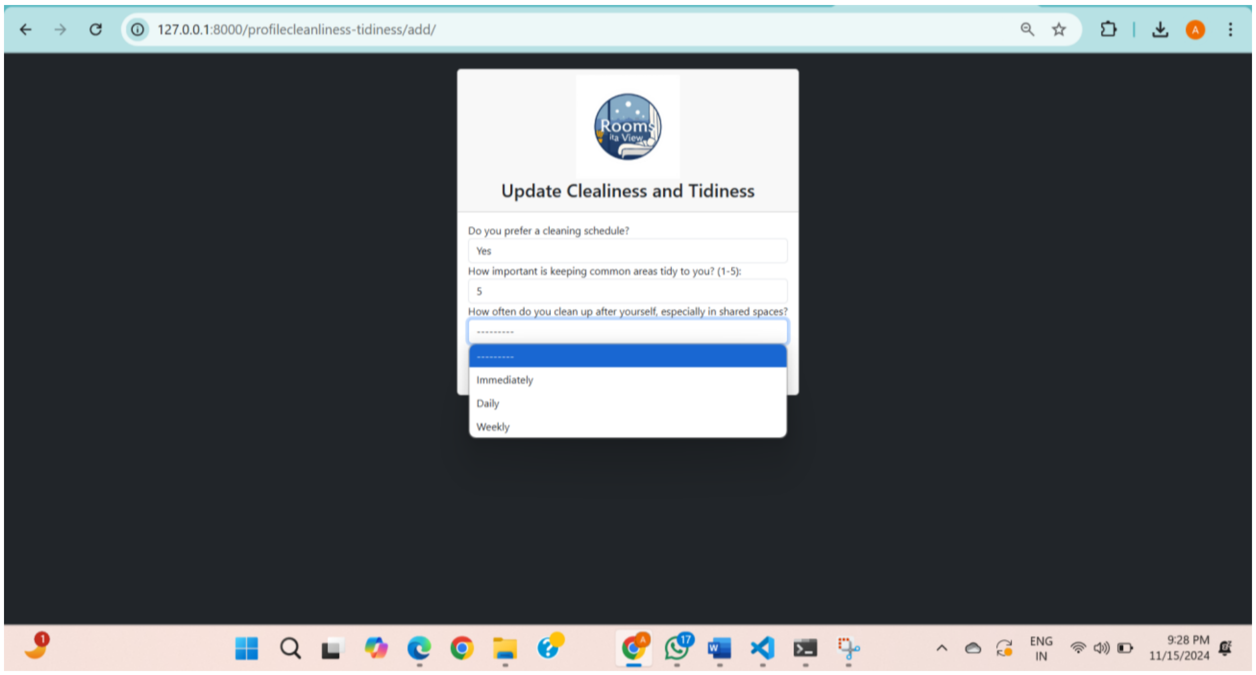
When I choose to add the lifestyle and routine it will have some pre classified questions and we have to fill the answers according to our interests. All these questions are made to understand the basic needs of the person so that they can match with their roomates,



Update Lifestyle Routine

What is your daily schedule like?	Office at 9AM
Are you an early riser or a night owl?	Early Riser
Do you work from home, or will you be out of the house most of the time?	Work from Home
How do you spend your weekends?	

Fig:11



Update Clealiness and Tidiness

Do you prefer a cleaning schedule?

Yes

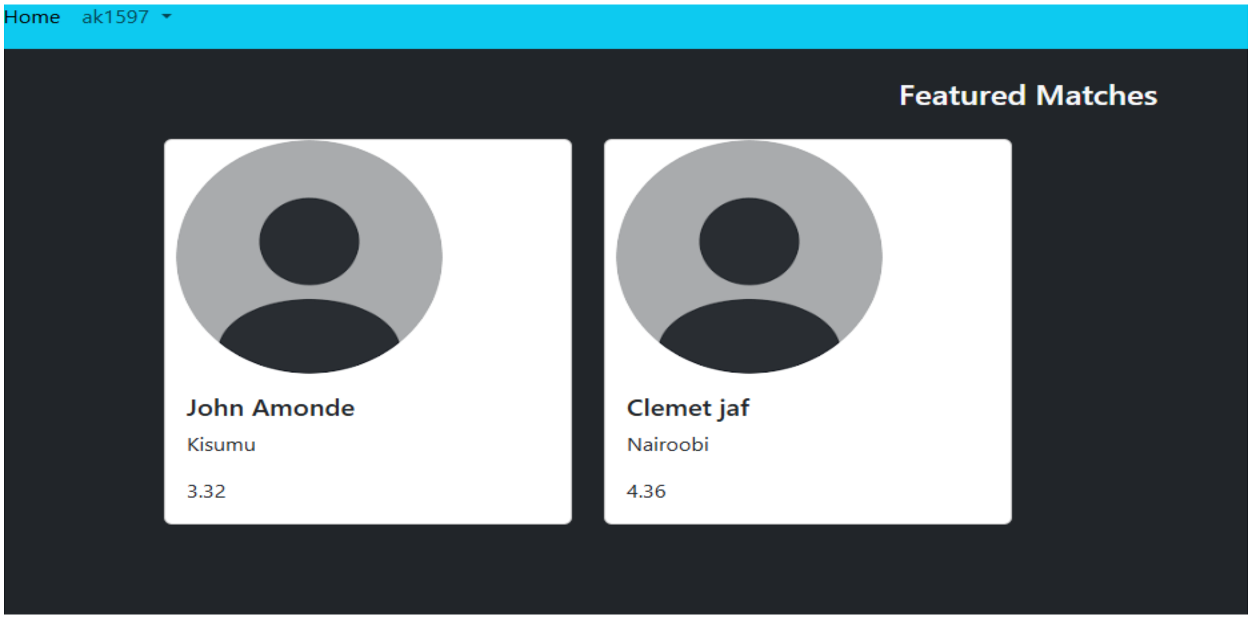
How important is keeping common areas tidy to you? (1-5):

5

How often do you clean up after yourself, especially in shared spaces?

Fig:12

In the same way we can fill all the sections and then we can submit the details. Since we may have clients that were looking for their roommates. All these data are saved in the account and when you go for matches it will give the most relevant matches according to our profile and give their details so that we can communicate further as shown in the image.



Featured Matches

John Amonde

Kisumu

3.32

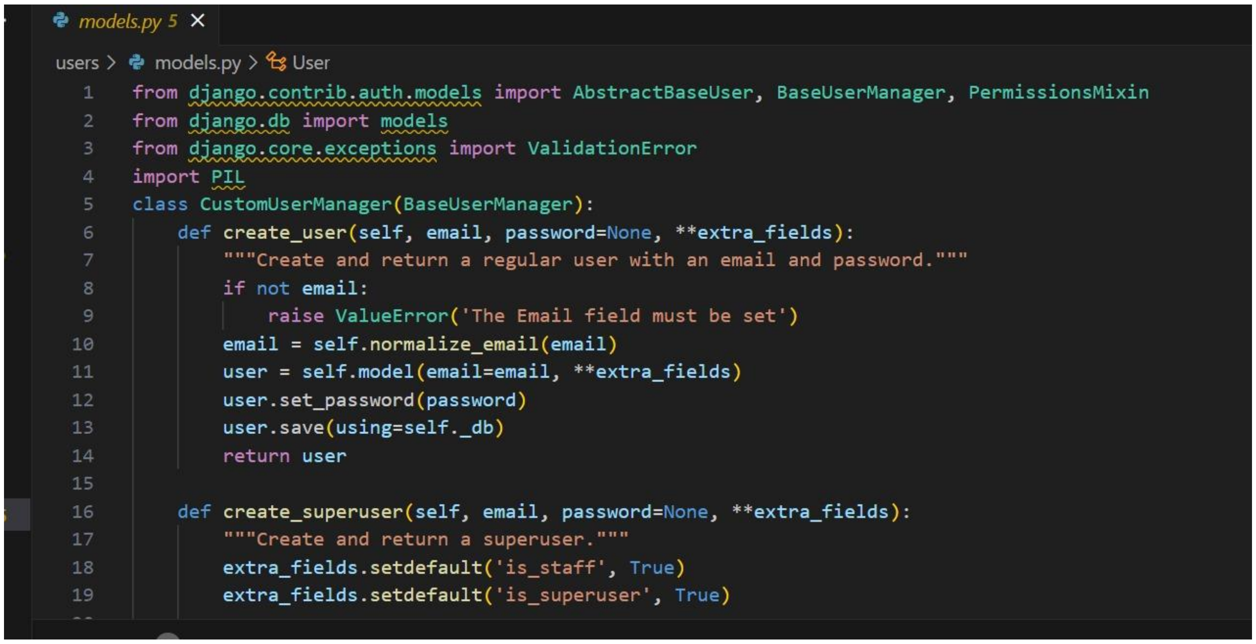
Clemet jaf

Nairoobi

4.36

Fig:13

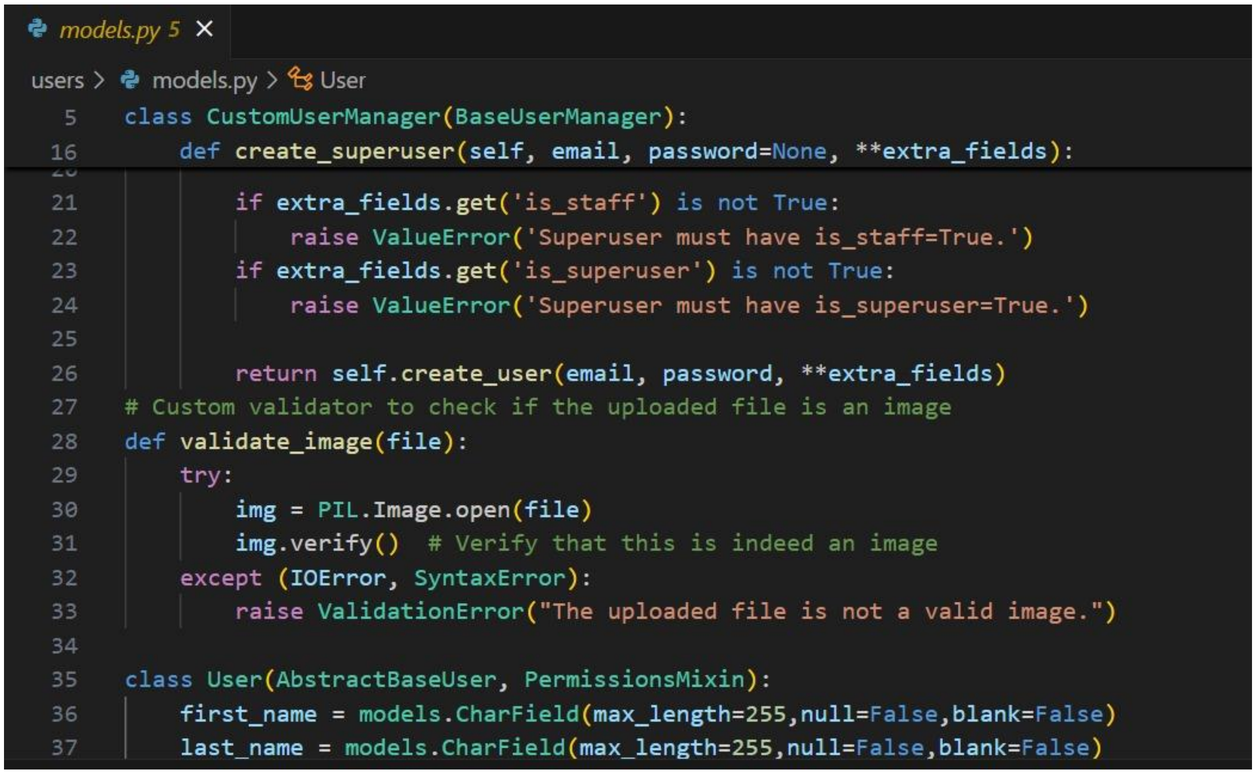
Piece of Code and explanation



```python
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager, PermissionsMixin
from django.db import models
from django.core.exceptions import ValidationError
import PIL
class CustomUserManager(BaseUserManager):
    def create_user(self, email, password=None, **extra_fields):
        """Create and return a regular user with an email and password."""
        if not email:
            raise ValueError('The Email field must be set')
        email = self.normalize_email(email)
        user = self.model(email=email, **extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, email, password=None, **extra_fields):
        """Create and return a superuser."""
        extra_fields.setdefault('is_staff', True)
        extra_fields.setdefault('is_superuser', True)
```

Fig:14

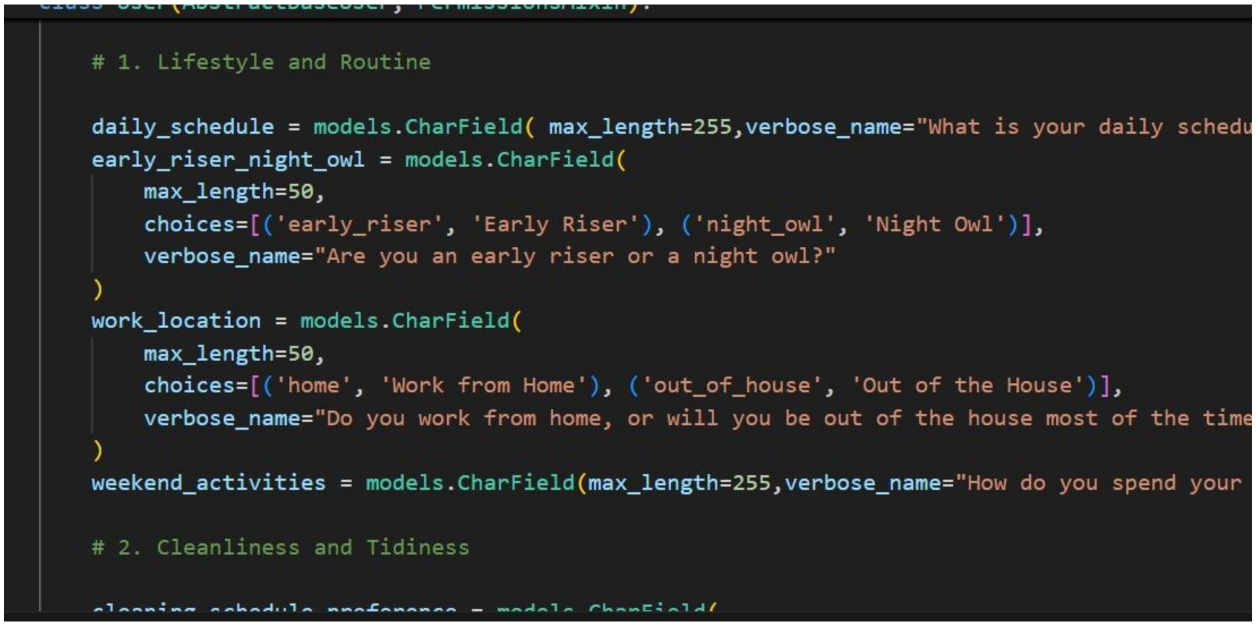


```python
class CustomUserManager(BaseUserManager):
    def create_superuser(self, email, password=None, **extra_fields):
        if extra_fields.get('is_staff') is not True:
            raise ValueError('Superuser must have is_staff=True.')
        if extra_fields.get('is_superuser') is not True:
            raise ValueError('Superuser must have is_superuser=True.')

        return self.create_user(email, password, **extra_fields)
# Custom validator to check if the uploaded file is an image
def validate_image(file):
    try:
        img = PIL.Image.open(file)
        img.verify()  # Verify that this is indeed an image
    except (IOError, SyntaxError):
        raise ValidationError("The uploaded file is not a valid image.")

class User(AbstractBaseUser, PermissionsMixin):
    first_name = models.CharField(max_length=255,null=False,blank=False)
    last_name = models.CharField(max_length=255,null=False,blank=False)
```

Fig:15



```
    # 1. Lifestyle and Routine

    daily_schedule = models.CharField( max_length=255,verbose_name="What is your daily schedu
    early_riser_night_owl = models.CharField(
        max_length=50,
        choices=[('early_riser', 'Early Riser'), ('night_owl', 'Night Owl')],
        verbose_name="Are you an early riser or a night owl?"
    )
    work_location = models.CharField(
        max_length=50,
        choices=[('home', 'Work from Home'), ('out_of_house', 'Out of the House')],
        verbose_name="Do you work from home, or will you be out of the house most of the time
    )
    weekend_activities = models.CharField(max_length=255,verbose_name="How do you spend your

    # 2. Cleanliness and Tidiness
```

Fig:16

Custom User Manager (CustomUserManager):

create_user: A method to create a regular user, ensuring that the email is set and normalizing it before saving.
create_superuser: A method to create a superuser with necessary flags (is_staff, is_superuser).

Custom User Model (User):

Inherits from AbstractBaseUser and PermissionsMixin to enable custom authentication and permission handling.

Fields: The user model contains several fields, including personal information (first_name, last_name, username, gender, age, etc.), preferences, and habits (e.g., lifestyle, cleanliness, pets, smoking, drinking, etc.).

Profile Picture: A field for uploading a profile picture, with a custom validator (validate_image) to ensure the file is an image. The save method resizes the image to 300x300 if necessary.
Field Types: Various field types are used, including CharField, TextField, IntegerField, BooleanField, ImageField, and PositiveIntegerField.

Custom Validation:

A custom validator is used to check if an uploaded file is a valid image, raising a ValidationError if it's not.
Manager Assignment:

The custom user manager (CustomUserManager) is assigned to the objects attribute of the User class to manage user creation and querying.

Username Field:

The USERNAME_FIELD is set to username, meaning users will log in using their username, not email.

6. Challenges

The development of the KNN algorithm in the roommate matching platform encountered: The most challenging task was the absence of complete or discrepancy in the data received from users. For instance, some users may not have completed every part of the profile and therefore; it cannot be possible to get compatibility scores. To this effect, such missing values are either imputed for the data or some default 0is placed based on a user profile analysis.

Another difficulty was how to adapt the algorithm the growth of the number of users in the social network. The basic KNN algorithm that involves comparing a user with all other users can be very slow when applied on large data sets. Some techniques, including the PCA for dimensionality reduction or by the use of approximate nearest neighbor search, have somewhat lessened this problem. Finally, handling the subjective nature of user preferences was complex, as some attributes, like "social habits," can be difficult to quantify accurately. One solution to this problem was weighting of attributes in accordance with feedbacks received from the users and other preferences.

7. Conclusion / Future works

It is becoming evident that the specific AI-application in terms of providing customers with the matching roommate service can be a solution that is appropriate for the automation of the roommate selection. With the help of KNN algorithm the system gives very effective match suggestions when the user’s options are combined with their lifestyle choices. The basic modular structure of the platform in which the Main App, Profile App, and Algorithm App are implemented makes work stable and scalable, and the adaptivity of the layout improves the convenience of working on PCs, tablets, and mobile devices. The results of testing also show that the platform is reliable, protected from various types of Cyber threats, and in stable operating mode, which makes it fit for use as a tool for finding suitable roommates.

The Django framework, PostgreSQL for database management, and the KNN algorithm form a solid technical foundation for reliable data management, security, and scalability. The platform's successful implementation is demonstrated by its stability during testing, its ability to securely manage user information, and its capability to accommodate multiple users without performance degradation.

While the existing platform offers a practical and efficient solution for roommate matching, there is potential for further enhancement and expansion:

Advanced Machine Learning Techniques: Upgrading the current KNN algorithm with advanced approaches like collaborative filtering or deep learning could significantly improve the personalization of roommate recommendations. These advanced models could incorporate not only profile data but also user behavior and feedback, creating a more dynamic, adaptive, and precise recommendation system.

User Feedback Integration: Incorporating a real-time feedback loop, where users can rate their matched roommates, could help the system refine future match recommendations based on user satisfaction. Additionally, introducing functionality to match multiple users for larger shared living spaces could expand the platform's appeal to individuals interested in co-living with multiple roommates.

These advancements would not only improve the matching accuracy but also enhance user satisfaction and retention, making the platform more adaptive and user-centric.

8. References

[i] Wang, Z., & Sun, Y. (2018). How to Design the Registration and Login Function of APP. Journal of Software Engineering and Applications, 11(5), 223–234. https://doi.org/10.4236/jsea.2018.115014

[ii] Rao, W.M. and Lin, D.G. (2016) Just Talk about Smart Phone App UI Design Techniques in the Development. Electronic Test, No. 13.

[iii] Zhang, P. (2016) Writing the Android Mobile Platform UI Design Research. Nanjing Normal University, Nanjing.

[iv] Hu, X.D. (2014) The Design and Implementation of a Music Player Based on Android Platform. Jilin University, Changchun.

[v] Tian, L.G. and Yan, H. (2014) Smart Phone APP Interface Design Study. Western Radio and Television, No. 21.

[vi] Rental Buddy. (2024). RentalBuddy. https://www.rentalbuddy.ai/student/