Project Report on
Study Buddy A Discussion Room

Prepared by:

Priyank Raychura: priyankraychura@gmail.com

Urvisha Solanki: solankiurvisha40@gmail.com

Submitted in partial fulfilment of the requirement for the degree

BCA SEM - V

October 2024 - 25

Guided by:

Mr. Anand John

Bachelor of Computer Applications

Christ College, Rajkot

(Affiliated to Saurashtra University, Rajkot 360005)

77

Certificate

I/We hereby certify that the work which is being presented in the BCA project report entitled "Study Buddy", in partial fulfilment of the requirements for the award of the Bachelor of Computer Applications and submitted to the Dept. of Computer Applications, Christ College, Rajkot is an authentic record of our own work carried out during a period from June 2024 to October 2024 under the supervision of **Mr. Anand John**.

The matter presented in this report has not been submitted by us for the award of any other degree elsewhere.

Signature of Students

1. *Priyank Raychura*
   [003303225240]

2. *Urvisha Solanki*
   [003303225254]

This is to certify that the above statement made by the candidate/s is/are correct to the best of my knowledge.

Head Of Department
Dr. Shailendrasinh Jadeja
Dept. Of Computer Applications
Christ College, Rajkot

Project Guide
Mr. Anand John
Dept. Of Computer Applications
Christ College, Rajkot

ACKNOWLEDGEMENT

When we embarked this project, it appeared to us as onerous task. Slowly as we progressed, we did realize that we were not alone after all. We wish to express our gratitude to **Rev. Fr. (Dr), Jomon Thommana Director, Christ Campus, Dr. Yvonne Fernandes Principal Christ College and Mr. Anand John** Project guide who have extended their kind help, guidance and suggestion without which it could not have been possible for us to complete this project report.

Our sincere thanks to **Ms. Pintukumari Bera** and all-entire faculty members, friends and parents for offering us all kinds of support and help in preparing the project. We are deeply indebted to our guide **Mr. Anand John** for not only his valuable and enlightened, guidance but also for the freedom he rendered us during this project work.

We are thankful to our group members and other classmates, well-wishers who with their magnanimous and generous help and support made it a relative easier affair. Our hearts go out to our parents who bear with us all the trouble we caused then with smile during the entire study period and beyond.

Priyank Raychura

Urvisha Solanki

BCA – September 2024

Christ College, Rajkot

PREFACE

This project documentation provides a comprehensive overview of the project titled "Study Buddy". The aim of this project is to develop a "web-based application that allows users to efficiently manage and share study information within a community of learners".

In the initial stages, I faced several challenges, such as understanding the appropriate technologies and methodologies to use. However, through consistent effort and guidance, I was able to overcome these difficulties and achieve the project's objectives.

The document is structured to provide an understanding of the problem addressed by the project, the solution proposed, and the technologies used in its implementation. It also outlines the testing procedures, results, and potential future enhancements that could be made to improve the system further.

This project is a reflection of my learnings, and the practical application of the theoretical concepts acquired during my course. I hope it serves as a valuable contribution to my academic experience and benefits anyone who wishes to further explore this area of study.

Table of Contents

1. Introduction

1.1. Problem Statement

In today's fast-paced educational environment, students and professionals alike often struggle to manage, access, and share relevant study materials or knowledge in an organized manner. Study groups and knowledge-sharing forums are usually fragmented, scattered across different platforms, or lack specific focus, which makes it difficult for learners to collaborate efficiently.

Furthermore, the absence of an easy-to-use platform to facilitate real-time discussions, ask questions, and share resources within focused study topics hinders both individual and collective learning.

1.2. Project Overview

"Study Buddy" is a web-based platform designed to help students collaborate in study rooms. Users can create, join, and manage study rooms, participate in discussions, and stay organized. The project is built using Django for the back end and front-end frameworks for the user interface.

- Create study rooms on specific topics.
- Host and participate in real-time discussions.
- Access a profile-based system, where each user has a personal dashboard to manage their activity.
- Interact with other participants through messages, with features like editing and deleting messages for room owners and moderators.

1.3. Purpose of the Project

The primary purpose of this project is to create a centralized, user-friendly platform that fosters collaborative learning among students, professionals, or anyone with an interest in knowledge sharing. By offering a virtual space for topic-specific discussions this platform will serve as a tool to improve the way users engage in learning, especially in group environments.

2. Existing System

2.1.Gap Analysis

Current platforms like social media or forums offer limited functionality in organizing study groups effectively. They lack features such as dedicated study rooms, resource sharing, and efficient collaboration tools. StudyBud addresses these gaps by providing a purpose-built solution for academic collaboration.

2.2.Need of Application

2.2.1. Goal of the Project / Application

The goal is to provide students with an easy-to-use platform where they can create study rooms, participate in discussions, and manage study resources in one place. This platform aims to improve academic collaboration and communication among students.

2.2.2. Scope of the Project / Application

Study Buddy will focus on creating study rooms where users can participate in discussions, share resources, and organize their study materials. The platform includes a user authentication system, a dashboard for room management, and a feature to track popular or active study rooms.

2.3.Feasibility Study

- Technical Feasibility: Using Django for the back end and front-end frameworks ensures that the project is scalable and easy to maintain.
- Operational Feasibility: The platform is designed to be user-friendly, requiring minimal technical knowledge from users, making it widely accessible to students.
- Financial Feasibility: As a web-based platform, hosting costs are minimal, and open-source tools like Django reduce development costs.

3. Requirements

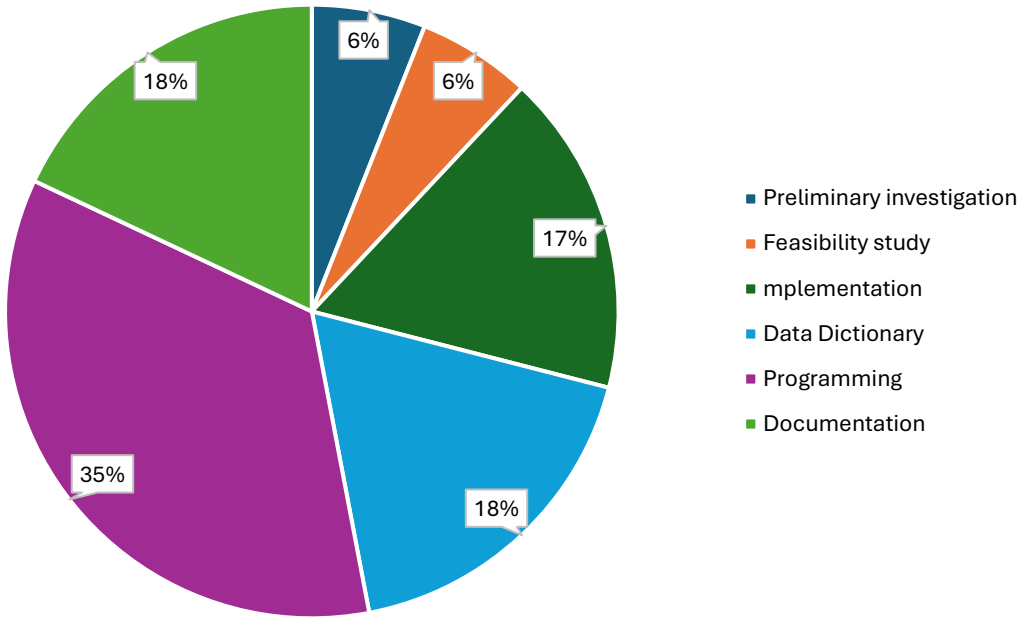
Hardware

- Processor: Intel i3 or higher (or equivalent AMD processor)
- RAM: Minimum 4 GB (8 GB recommended)
- Hard Disk: Minimum 20 GB free space
- Network Interface: Ethernet or Wi-Fi adapter
- Display: 1280x720 resolution or higher

Software

- Operating System: Windows 10/11, macOS or Linux (Ubuntu or any distribution)
- Text Editor/IDE: Visual Studio Code, PyCharm, or Sublime Text
- Backend Framework: Django 3.x/4.x (Python 3.x)
- Database: SQLite (for development) and PostgreSQL/ MySQL (for production)
- Frontend Technologies: HTML5, CSS3, JavaScript (Vanilla JS)
- Version Control: GitHub/Bitbucket/GitLab
- Web Browser: Chrome, Firefox, or Edge
- Package Management: pip (Python Package Installer)
- Local Server: Django's built-in development server
- Production Server: Gunicorn, Nginx (or similar)
- Deployment Platforms: Heroku, DigitalOcean, or AWS
- Requirements File: 'pip install -r requirements.txt'

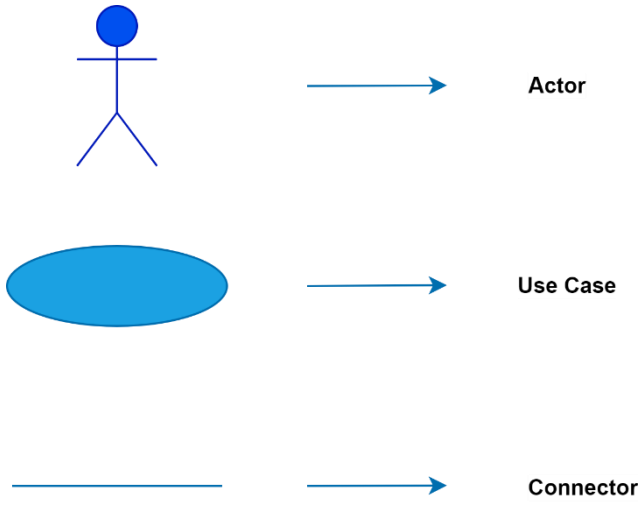
4. SDLC


6%
6%
17%
18%
35%
18%
Preliminary investigation
Feasibility study
mplementation
Data Dictionary
Programming
Documentation

5. Diagrams

5.1. Use Case Diagram

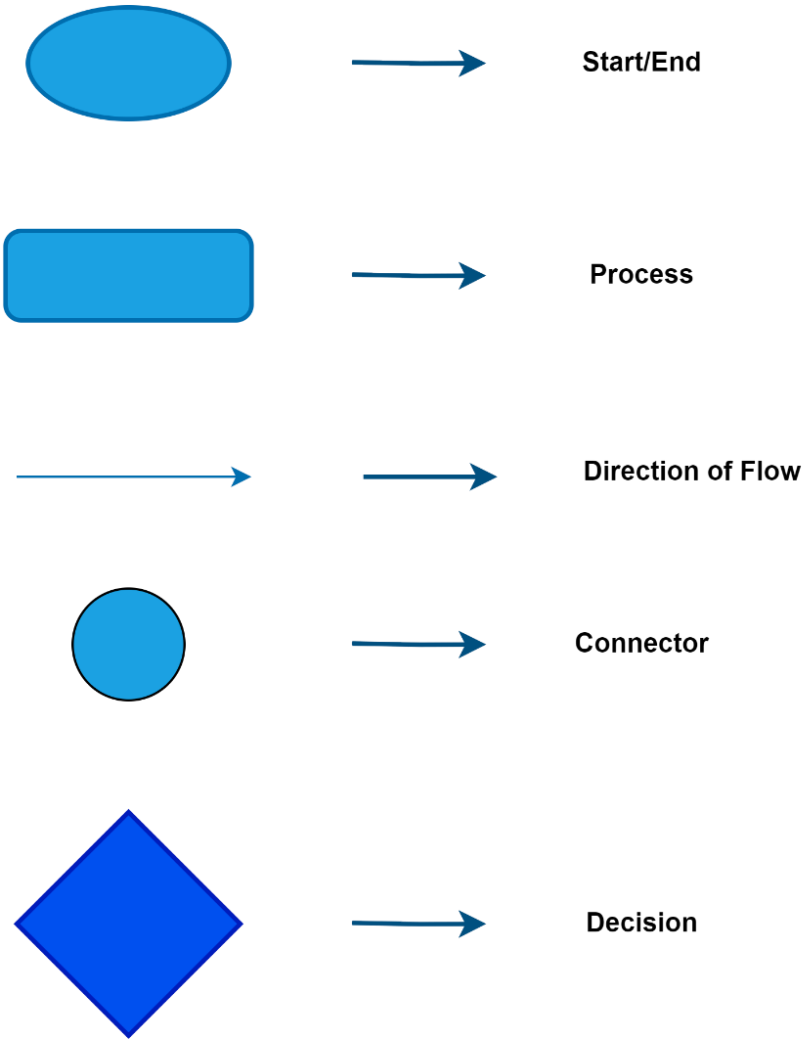
List of Figures



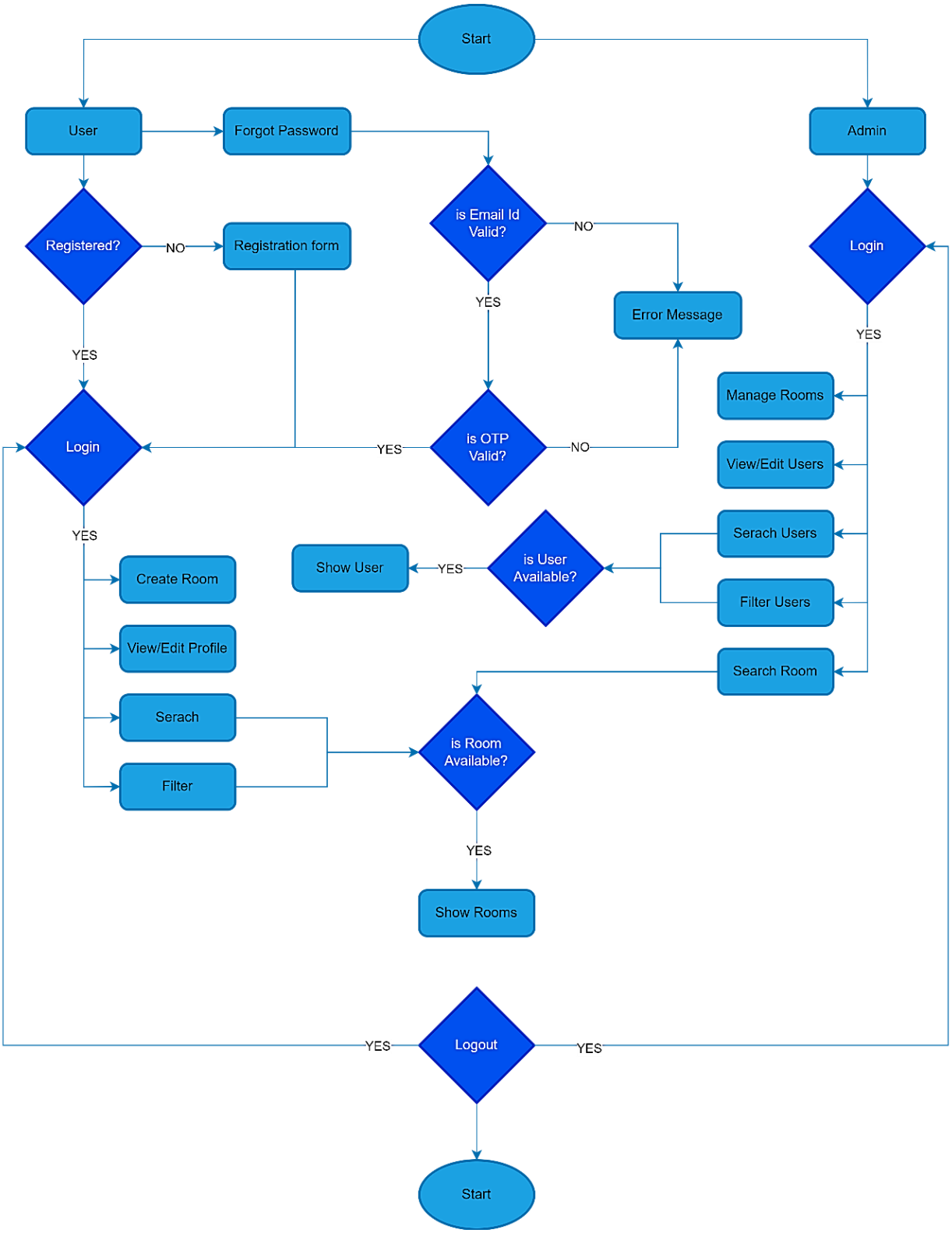
Diagram

5.2.Process Flow Diagram

- List of Figures

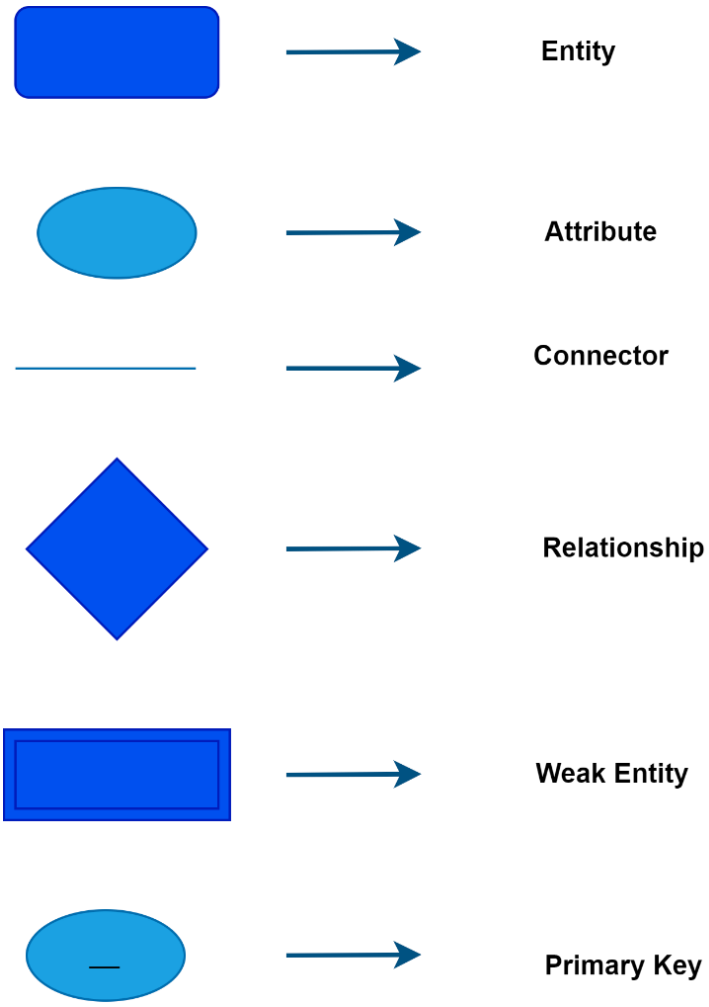


Diagram

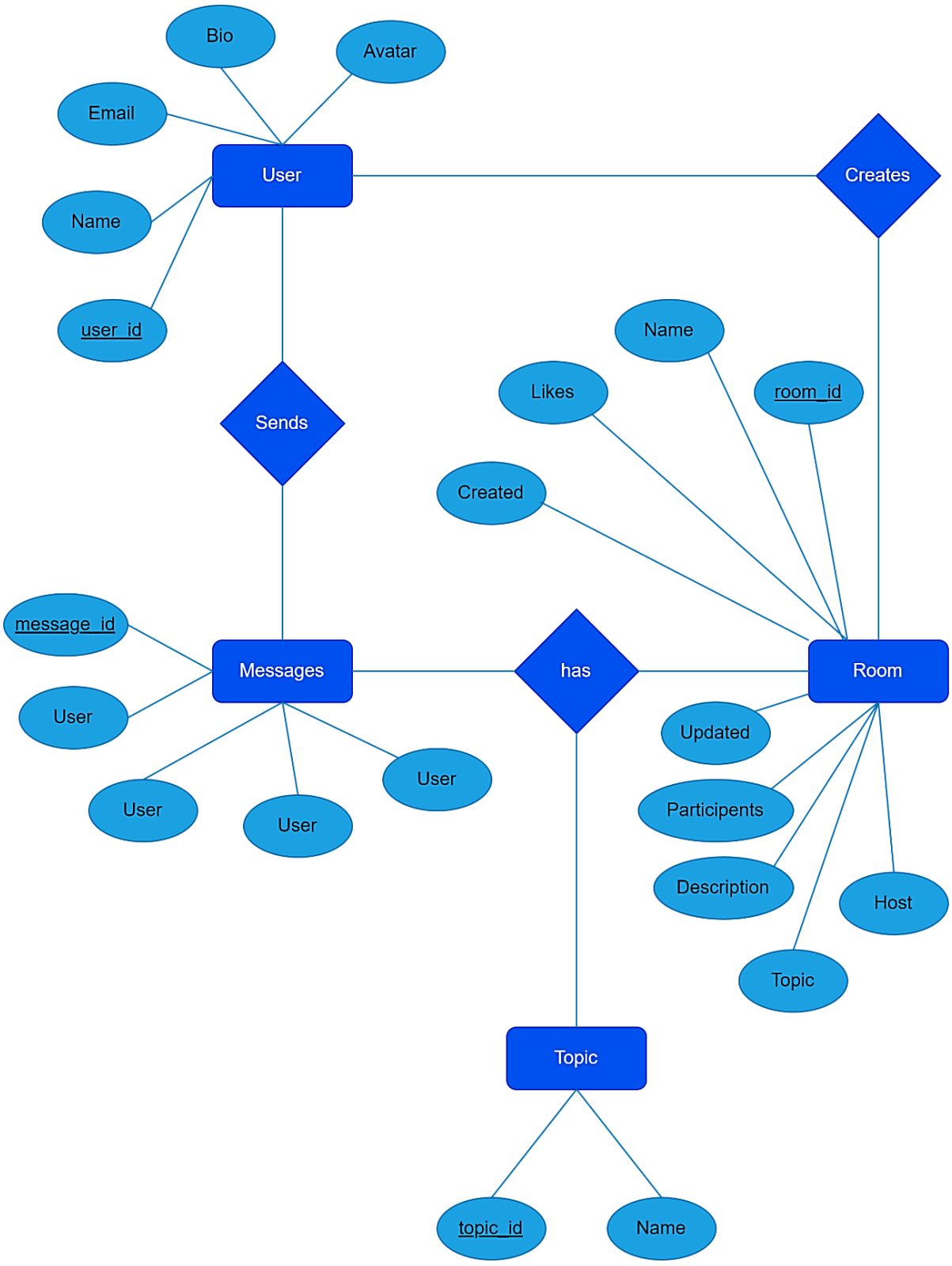

Start
User
Forgot Password
Admin
Registered?
NO
Registration form
is Email Id Valid?
NO
Login
YES
YES
Error Message
YES
Manage Rooms
Login
is OTP Valid?
YES
NO
View/Edit Users
YES
Serach Users
Create Room
Show User
YES
is User Available?
Filter Users
View/Edit Profile
Search Room
Serach
is Room Available?
Filter
YES
Show Rooms
YES
Logout
YES
Start

5.3.E-R Diagram

- List of Figures



Diagram


Bio
Avatar
Email
User
Creates
Name
user_id
Name
Likes
room_id
Sends
Created
message_id
Messages
has
Room
User
Updated
User
User
User
Participents
Description
Host
Topic
Topic
topic_id
Name

6. Data Dictionary

6.1. Data Flow Diagram

A DFD also known as “bubble chart” has the purpose of clarifying system requirements and identifying major transformations. It shows the flow of data through a system. It is a graphical tool because it presents a picture. The DFD may be partitioned into levels that represent increasing information flow and functional detail. Four simple notations are used to complete a DFD

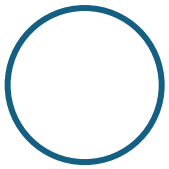
6.1.1. Data Flow

The data flow is used to describe the movement of information from one part of the system to another part. Flows represent data in motion. It is a pipeline through which information flows. Data flow is represented by an arrow.

Process Flow

6.1.2. Process

A circle or bubble represents a process that transforms incoming data to outgoing data. Process shows a part of the system that transforms inputs to outputs.



6.1.3. External Entity

A square defines a source or destination of system data. External entities represent any entity that supplies or receive information from the system but is not a part of the system.

6.1.4. Data Store

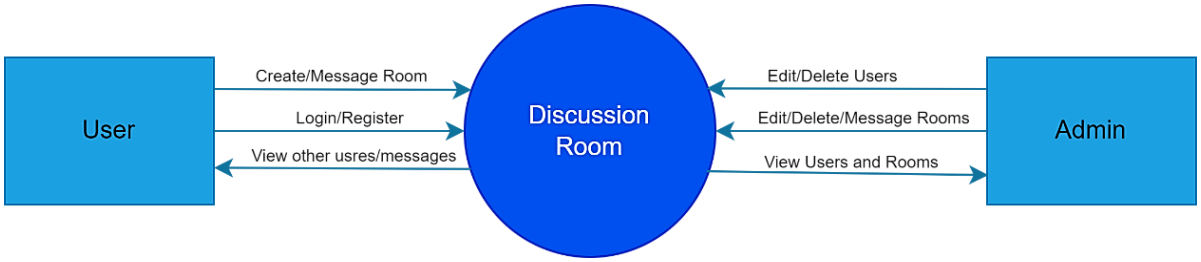
The data store represents a logical file. A logical file can represent either a data store symbol which can represent either a data structure or a physical file on disk. The data store is used to collect data at rest or a temporary repository of data. It is represented by open rectangle.

6.1.5. Output

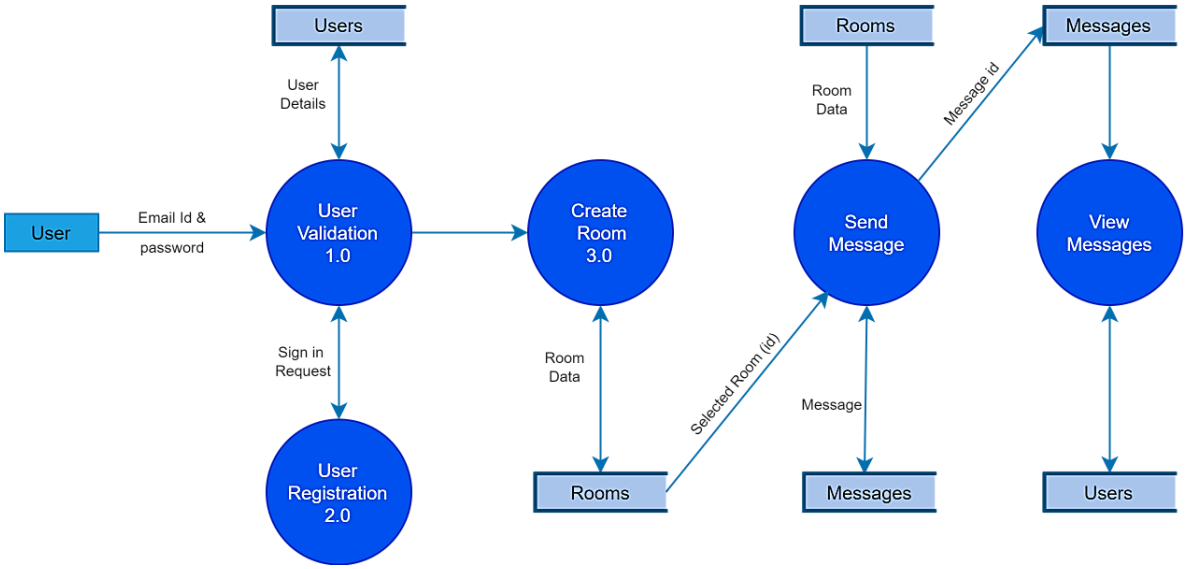
The output symbol is used when a hard copy is produced, and the user of the copies cannot be clearly specified or there are several users of the output.

6.2. Diagram

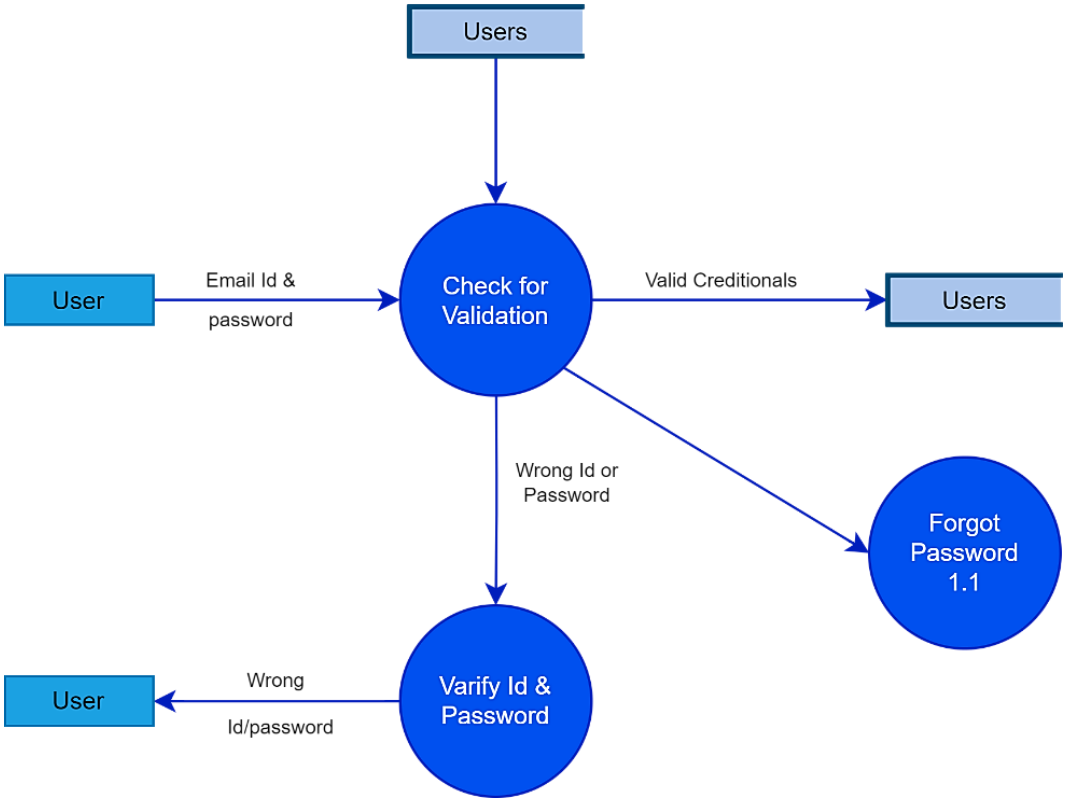
6.2.1. 0th Level Data Flow Diagram



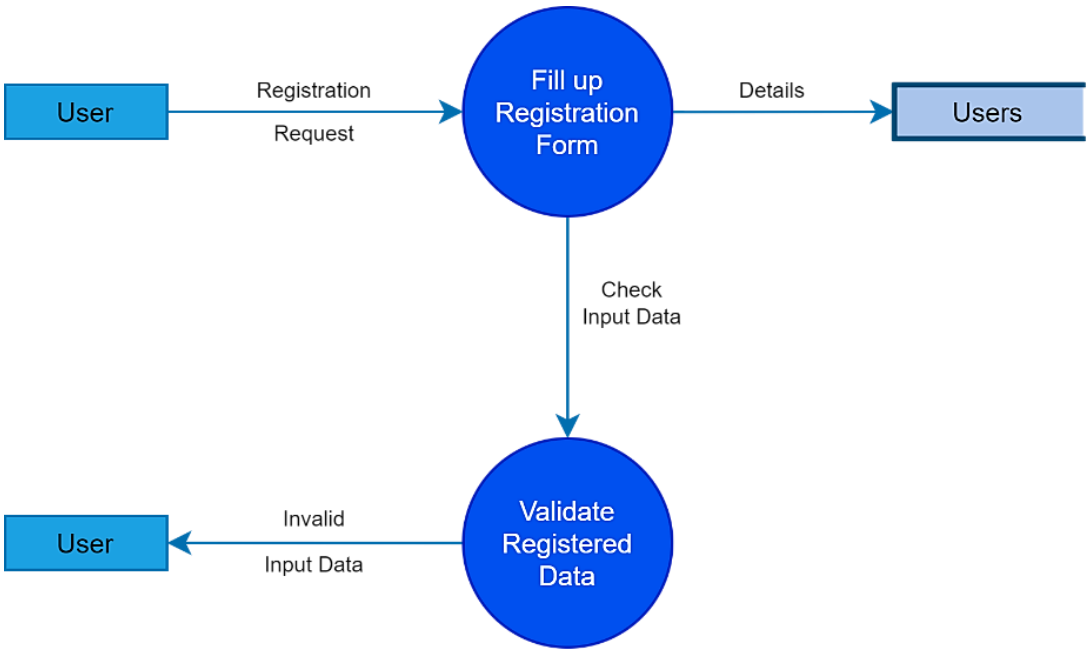
6.2.2. 1st Level Data Flow Diagram



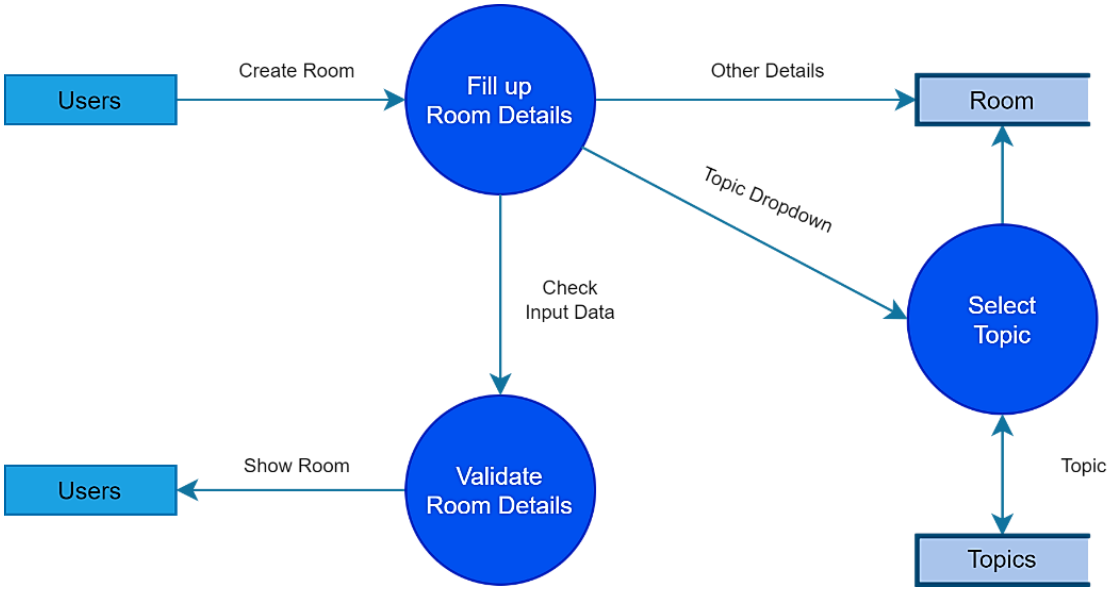
6.2.3. 2nd Level for 1.0



6.2.4. 2nd Level for 2.0



6.2.5. 2nd Level for 3.0



6.3 List of Tables

6.3.1. Users

Field Name	Data Type	Properties
name	CharField	max_length = 200, null = true
email	EmailField	unique = true, null = true
bio	TextField	null = true
avatar	ImageField	null = true default = “avatar.svg”
USERNAME_FIELD	String	‘email’
REQUIRED_FIELDS	List	[]

6.4.2 Messages

Field Name	Data Type	Properties
user	ForeignKey	to = User, on_delete = models.CASCADE
room	ForeignKey	to = Room, on_delete = models. CASCADE
body	TextField	-
updated	DateTimeField	auto_now = True
created	DateTimeField	auto_now_add = True

Meta Information

Property	Value
Ordering	[‘-updated’] [‘-created’]

6.4.3 Room

Field Name	Data Type	Properties
host	ForeignKey	to = User, on_delete = models.SET_NULL, null = True
topic	ForeignKey	to = Topic, on_delete = models.SET_NULL, null = True
name	CharField	max_length = 200
description	TextField	null = True, blank = True
participants	ManyToManyField	to = User, related_names = ‘participants’, blank = True
updated	DateTimeField	auto_now = True
created	DateTimeField	auto_now_add = True
likes	ManyToManyField	to = User, Related_name = ‘liked_items’, Blank = True

Meta Information

Property	Value
Ordering	[‘-updated’] [‘-created’]

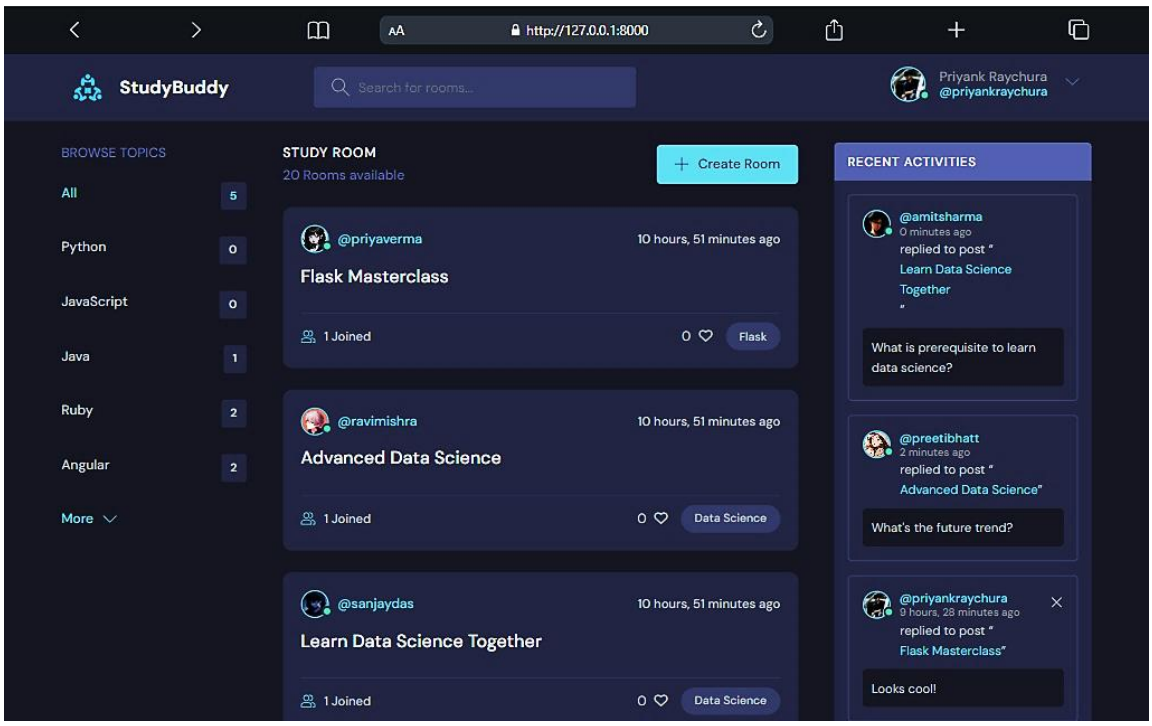
6.4.4 Topic

Field Name	Data Type	Properties
name	CharField	max_length = 200

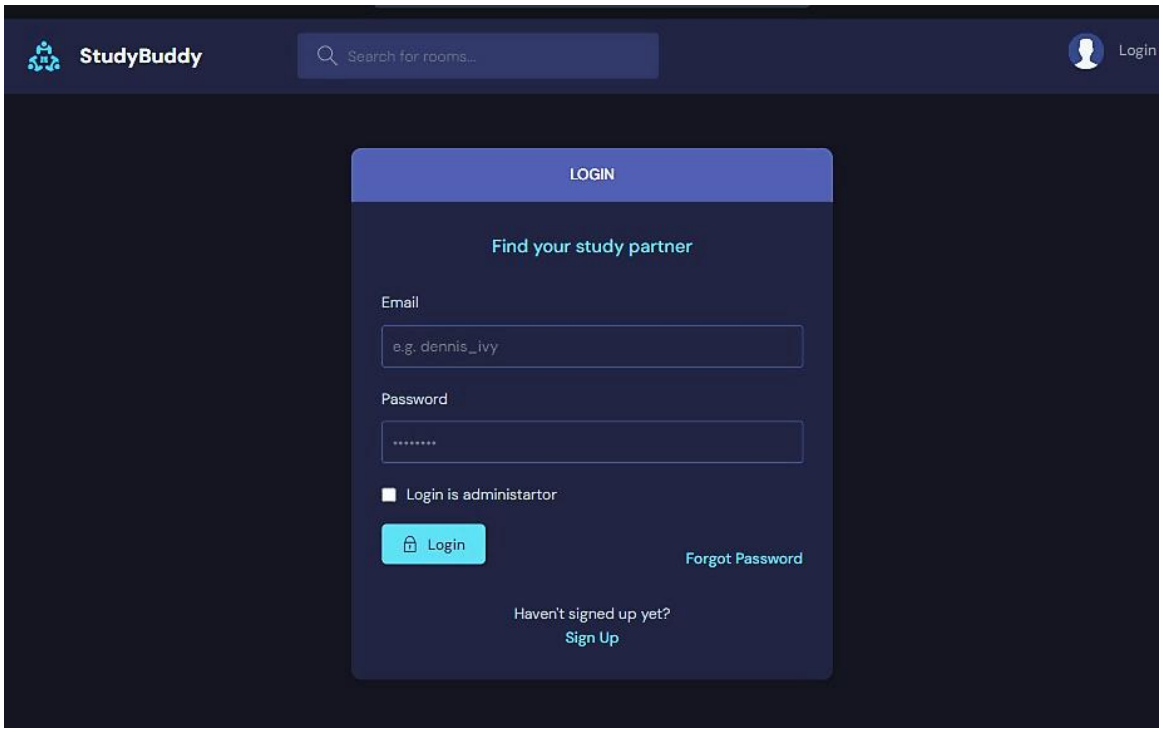
7 Project Overview

7.1 User Interface Screen Shots

7.1.1 Home Page

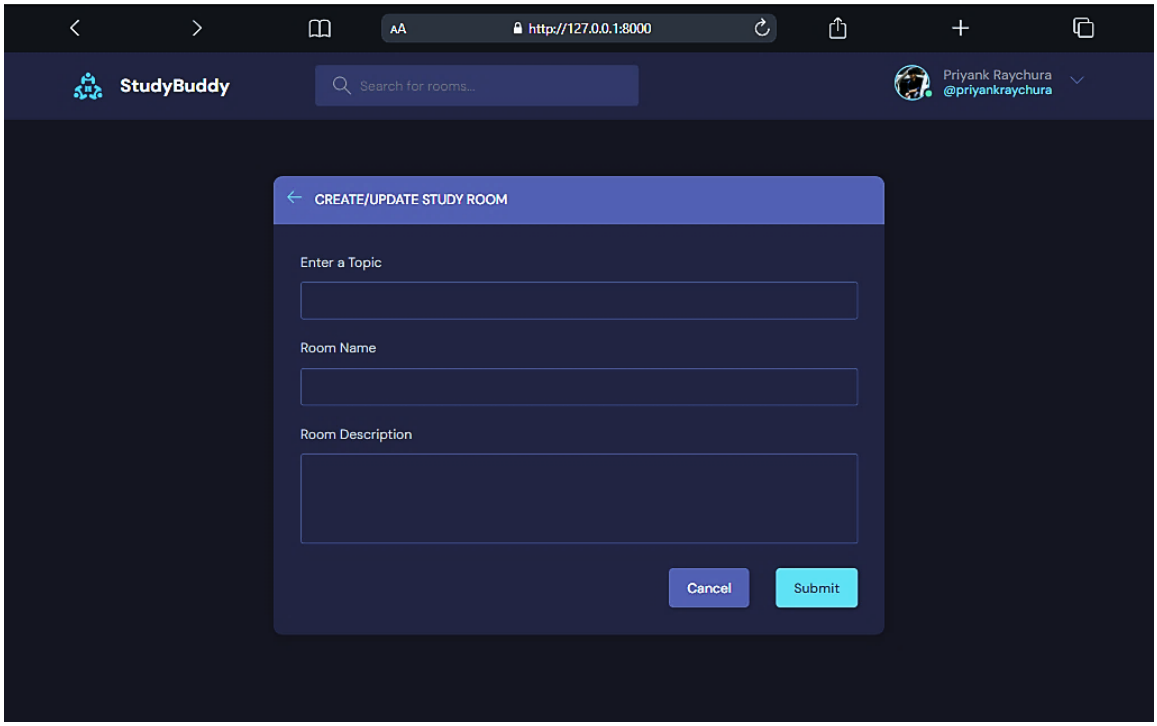


7.1.2 Log in

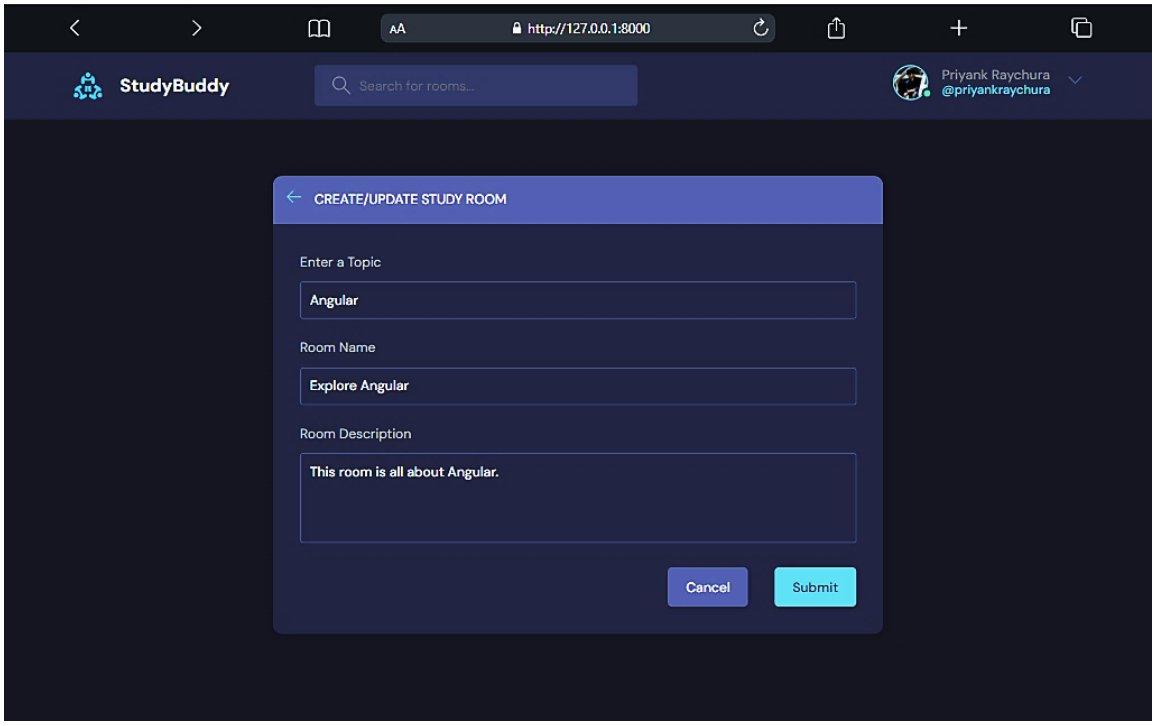


7.1.3 Room Section

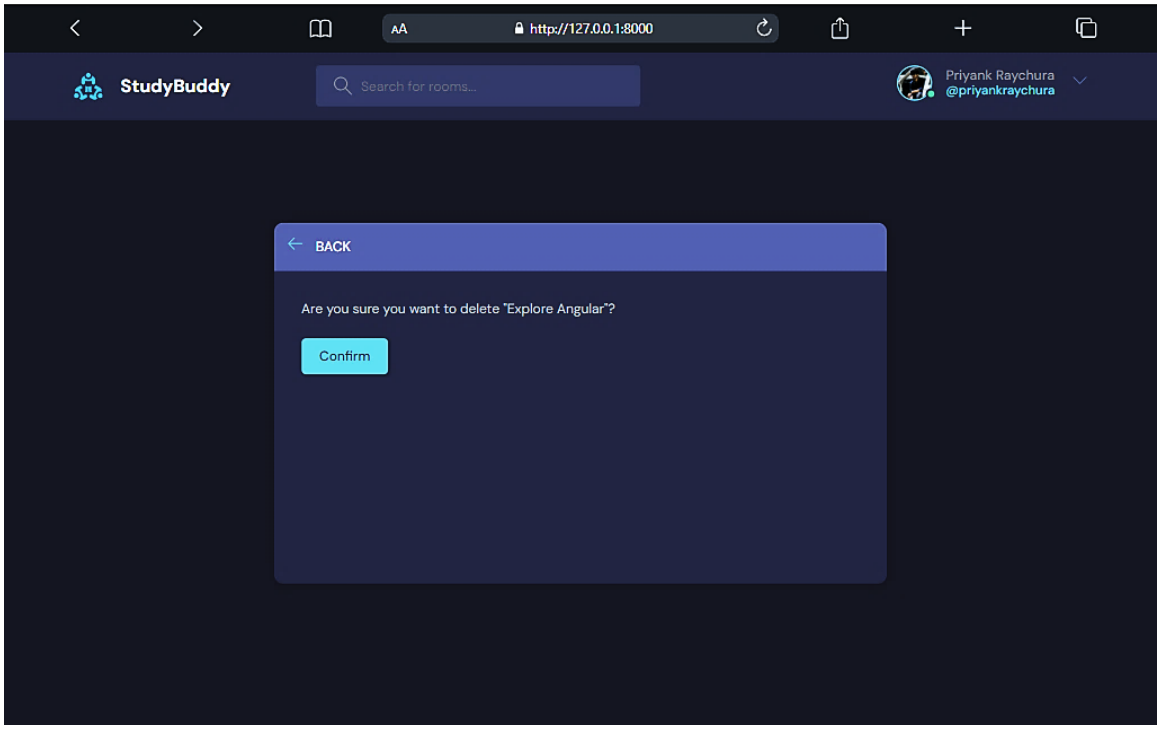
7.1.3.1 Create Room



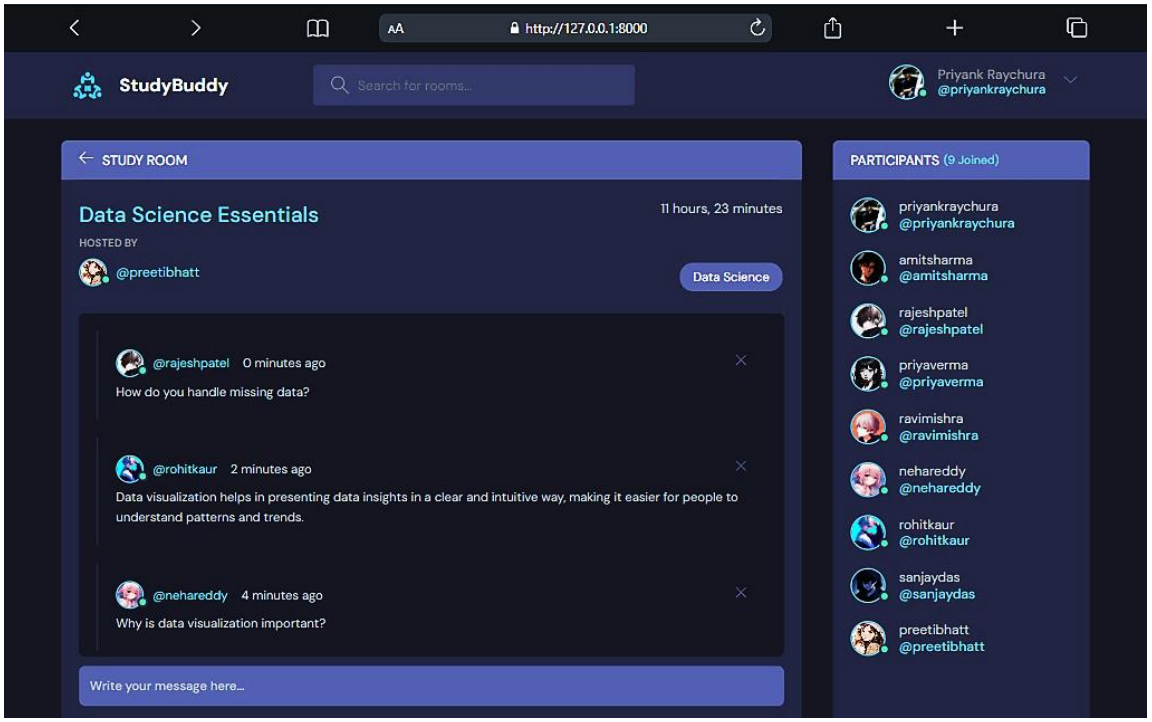
7.1.3.2 Update Room



7.1.3.3 Delete Room

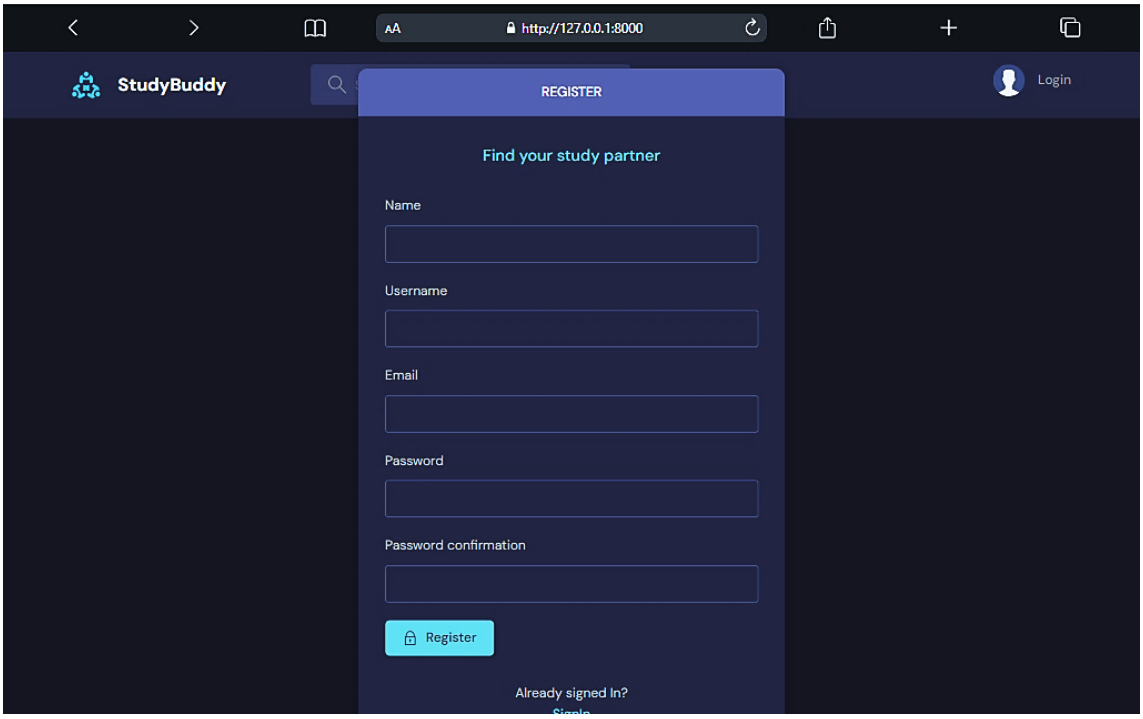


7.1.3.4 View Room

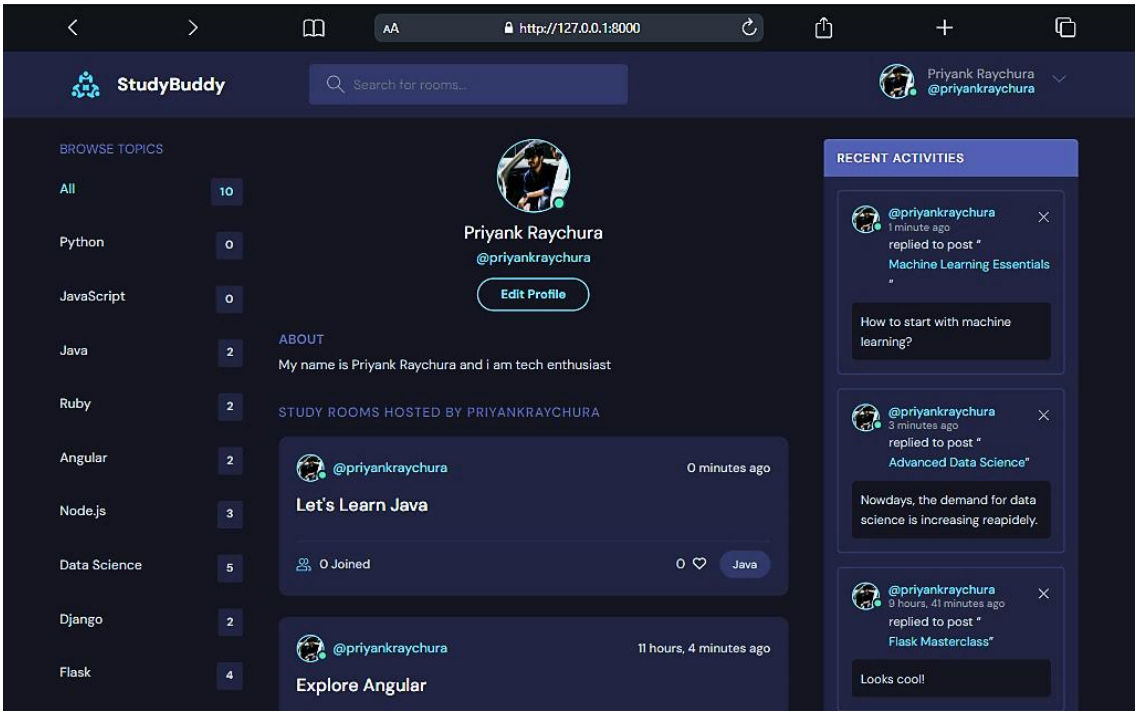


7.1.4 User Section

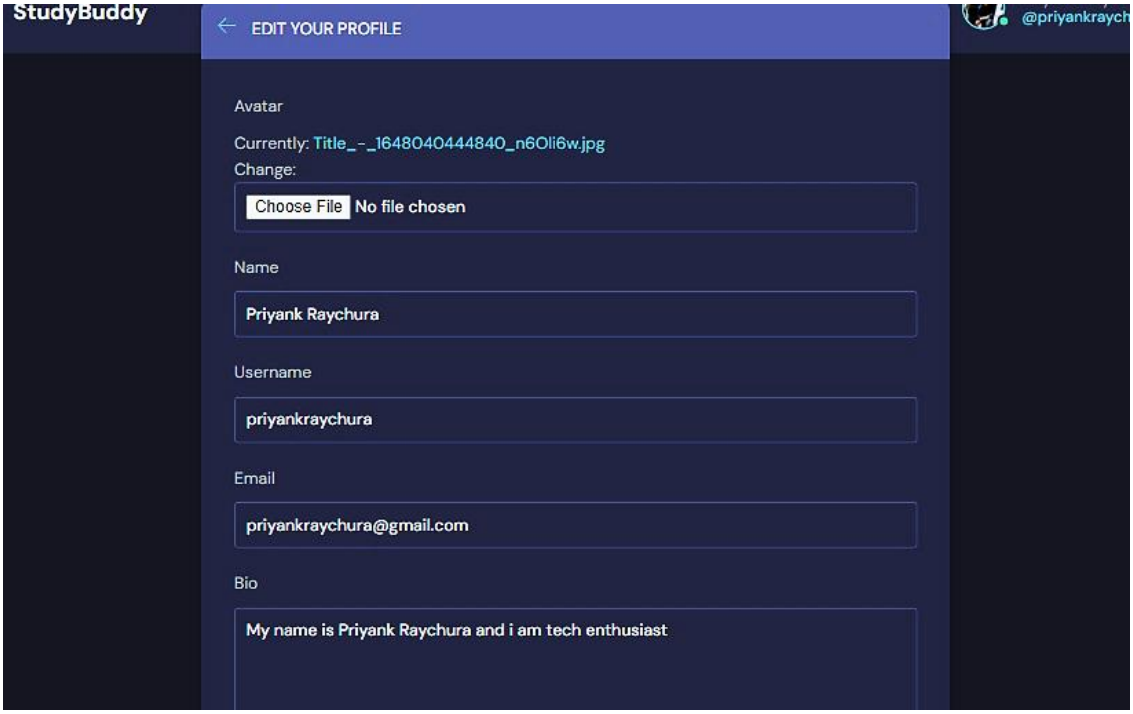
7.1.4.1 Register



7.1.4.2 View Profile

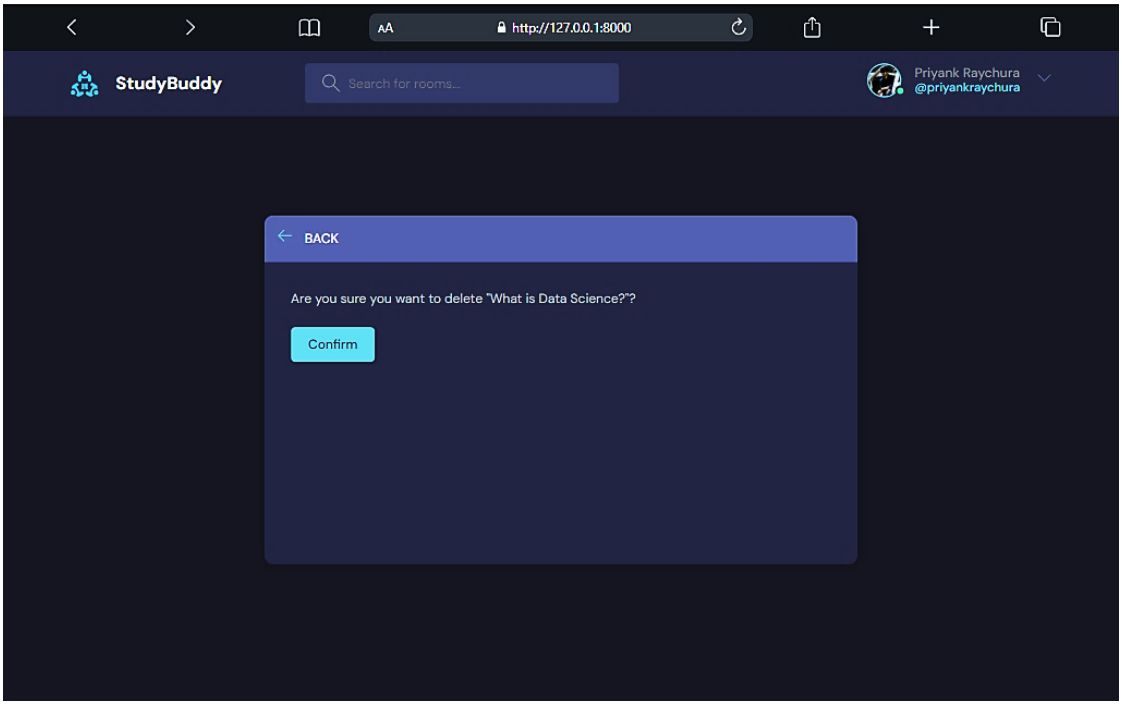


7.1.4.3 Edit Profile



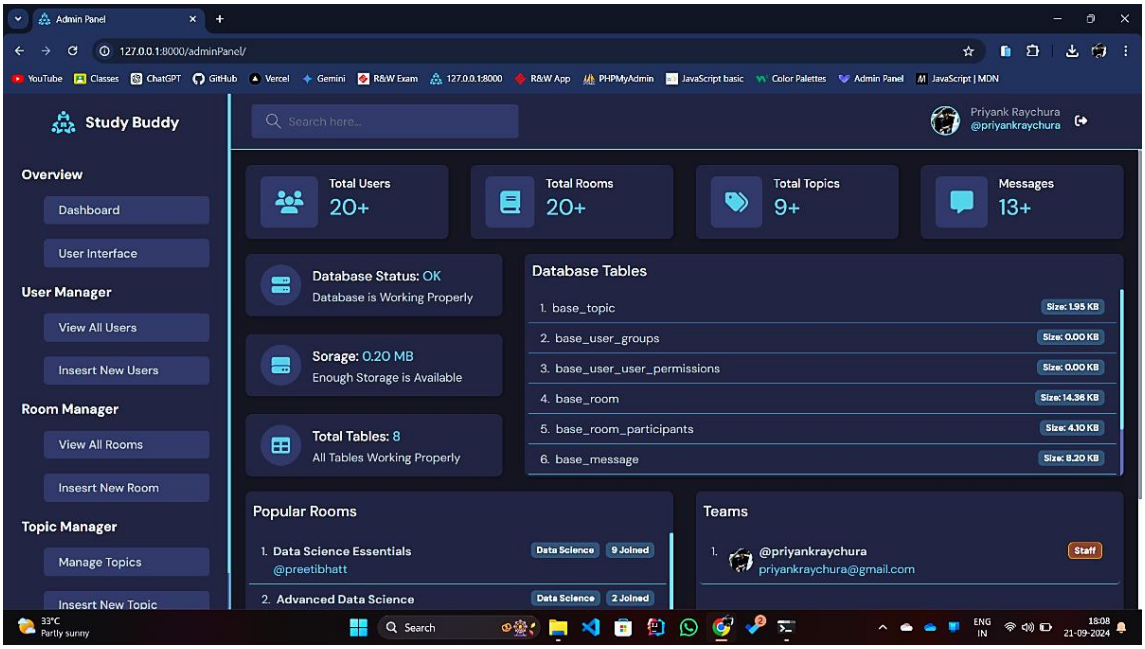
7.1.5 Messages

7.1.5.1 Delete Message

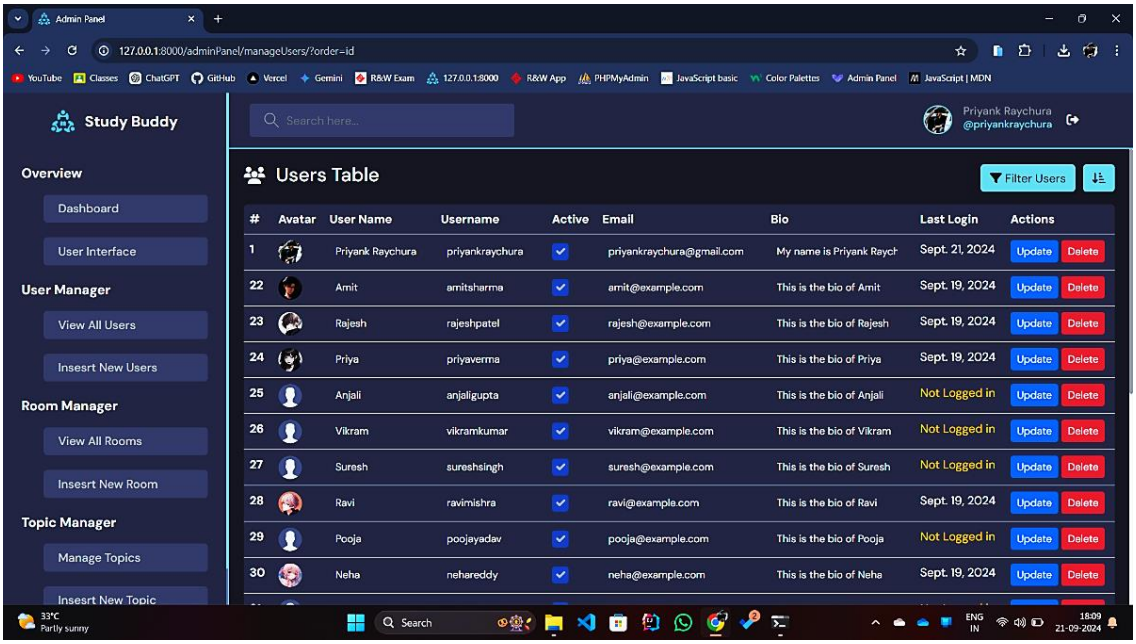


7.2Admin Panel Screen Shots

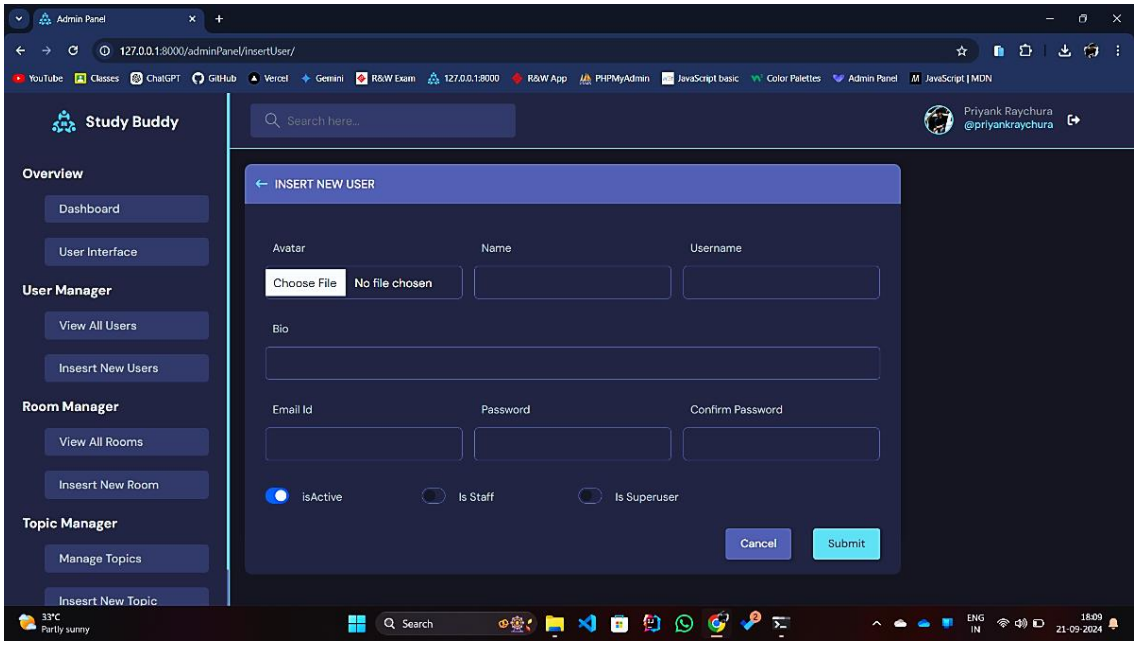
7.2.1 Dashboard



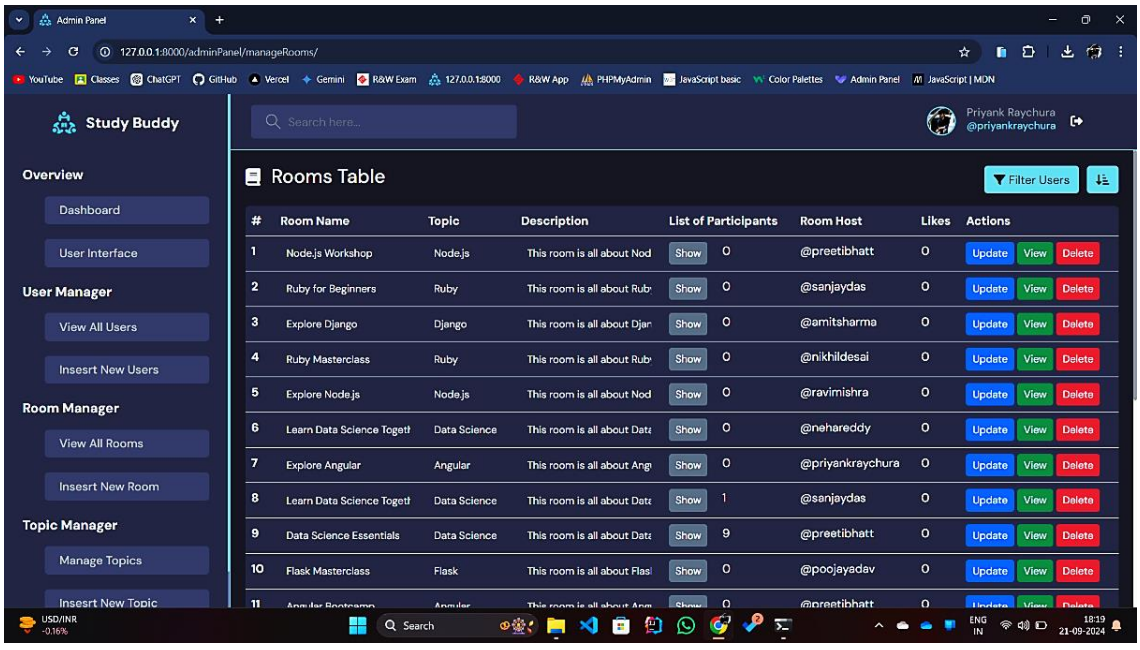
7.2.2 Users List



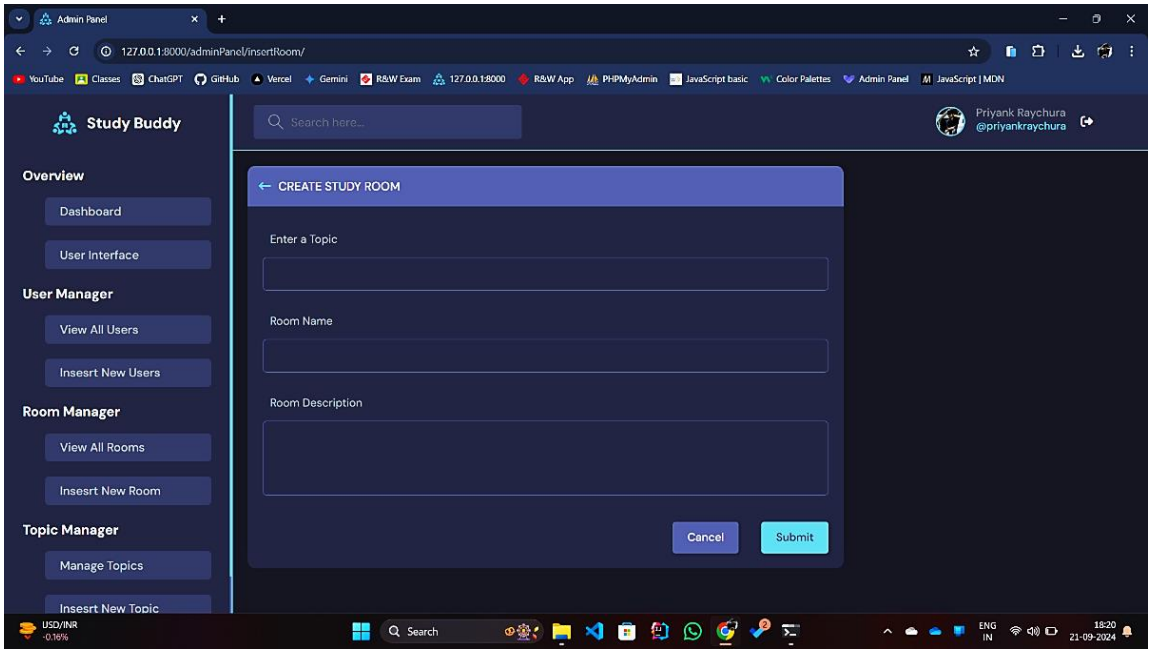
7.2.3 Insert User



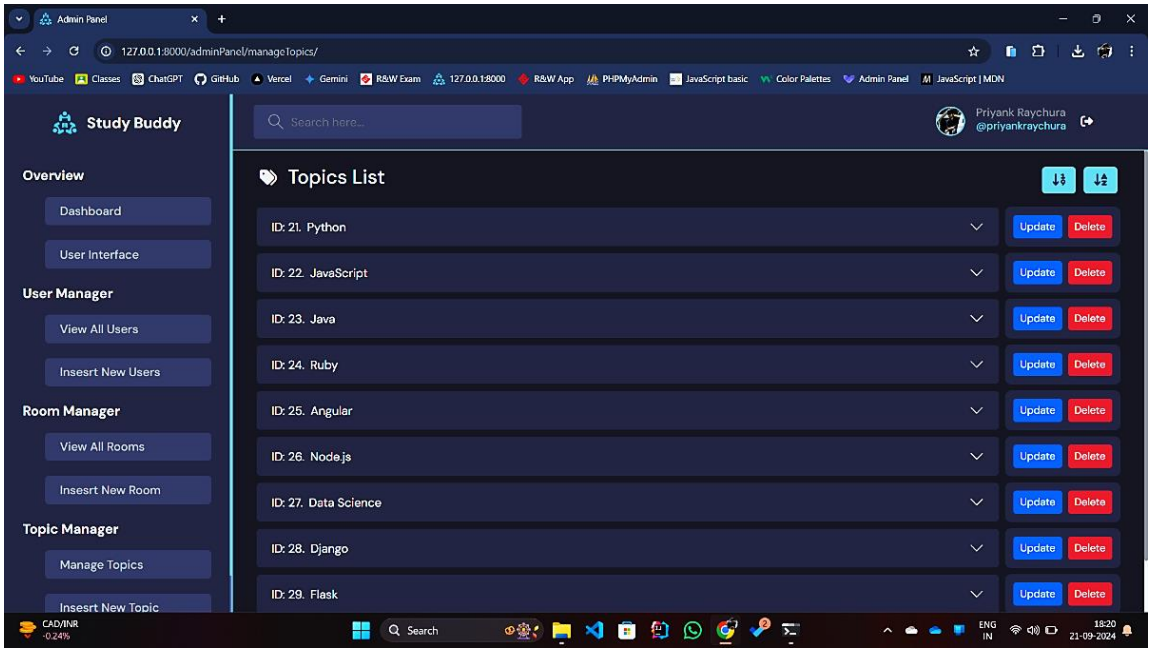
7.2.4 Rooms List



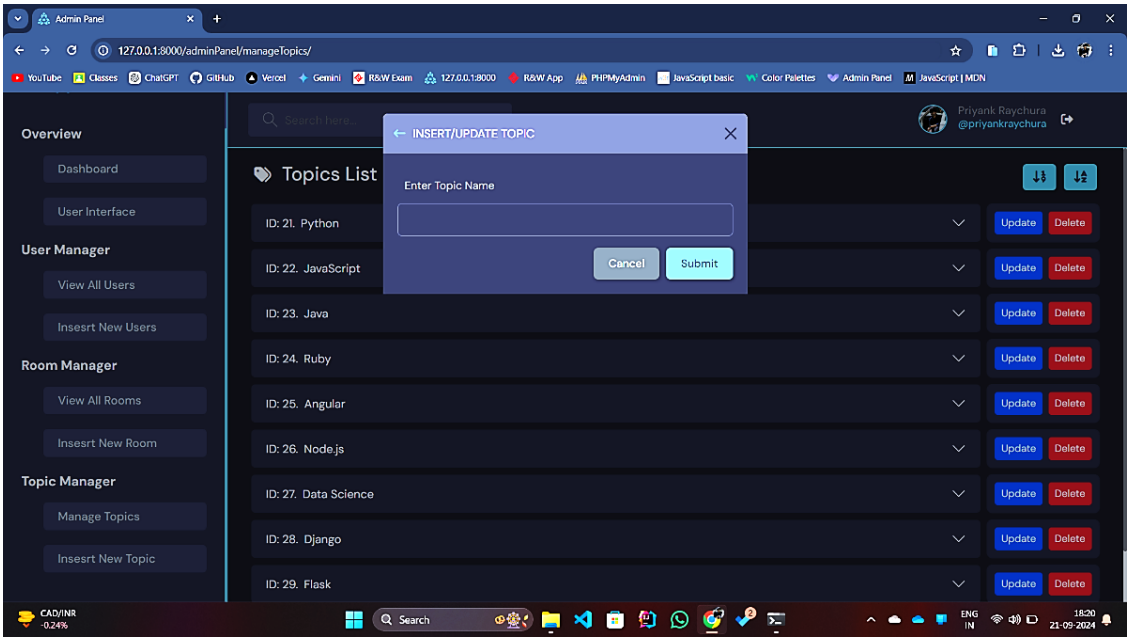
7.2.5 Insert Room



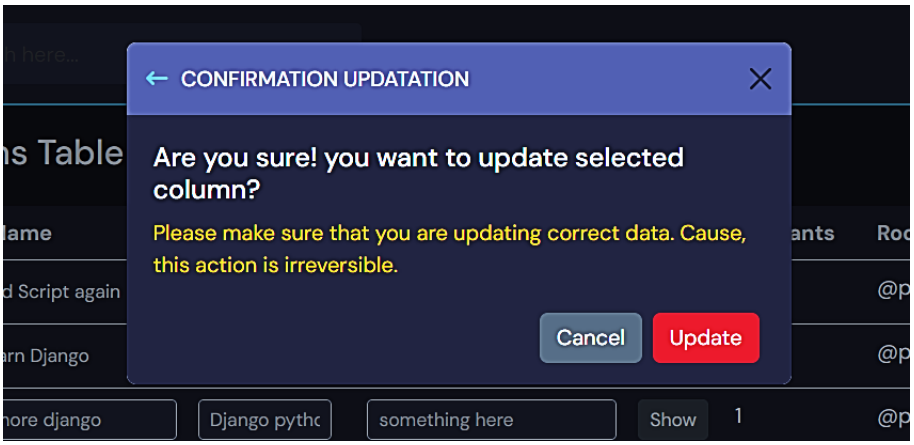
7.2.6 Topic List



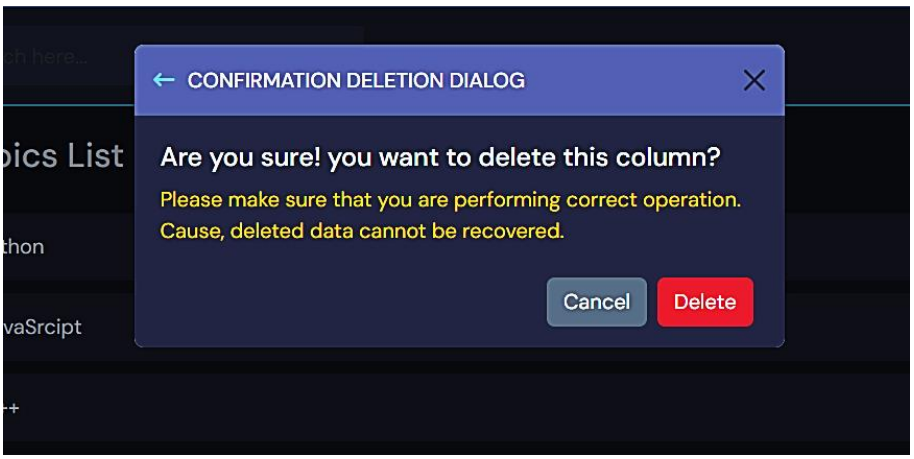
7.2.7 Insert Topic



7.2.8 Update



7.2.9 Delete



7.3 Code

7.3.1 Home

Home.html

```
{% for room in rooms %}
<div class="roomListRoom">
    <div class="roomListRoom__header">
        <a href="{% url 'user-profile' room.host.id %}" class="roomListRoom__author">
            <div class="avatar avatar--small {% if room.host.is_active %}active{% endif %}">
                <img src="{{ room.host.avatar.url }}" />
            </div>
            <span>@{{room.host.username}}</span>
        </a>
        <div class="roomListRoom__actions">
            <span>{{ room.created|timesince }} ago</span>
        </div>
    </div>
    <div class="roomListRoom__content">
        <a href="{% url 'room' room.id %}">{{room.name}}</a>
    </div>
    <div class="roomListRoom__meta">
        <a href="{% url 'room' room.id %}" class="roomListRoom__joined">
            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32
32">
                <title>user-group</title>
                <path
                    d="M30.539 20.766c-2.69-1.547-5.75-2.427-8.92-2.662 0.649 0.291 1.303 0.575 1.918
0.928 0.715 0.412 1.288 1.005 1.71 1.694 1.507 0.419 2.956 1.003 4.298 1.774 0.281 0.162 0.456 0.487
0.456 0.85v4.65h-4v2h5c0.553 0 1-0.447 1-1v-5.65c0-1.077-0.56-2.067-1.461-2.584z">
                </path>
                <path
                    d="M22.539 20.766c-6.295-3.619-14.783-3.619-21.078 0-0.901 0.519-1.461 1.508-1.461
2.584v5.65c0 0.553 0.447 1 1 1h22c0.553 0 1-0.447 1-1v-5.651c0-1.075-0.56-2.064-1.461-2.583zM22 28h-20v-
4.65c0-0.362 0.175-0.688 0.457-0.85 5.691-3.271 13.394-3.271 19.086 0 0.282 0.162 0.457 0.487 0.457
0.849v4.651z">
                </path>
                <path
                    d="M19.502 4.047c0.166-0.017 0.33-0.047 0.498-0.047 2.757 0 5 2.243 5 5s-2.243 5-5
5c-0.168 0-0.332-0.030-0.498-0.047-0.424 0.641-0.944 1.204-1.513 1.716 0.651 0.201 1.323 0.331 2.011
0.331 3.859 0 7-3.141 7-7s-3.141-7-7-7c-0.688 0-1.36 0.131-2.011 0.331 0.57 0.512 1.089 1.075 1.513
1.716z">
                </path>
                <path
                    d="M12 16c3.859 0 7-3.141 7-7s-3.141-7-7-7c-3.859 0-7 3.141-7 7s3.141 7 7 7zM12
4c2.757 0 5 2.243 5 5s-2.243 5-5 5-5-2.243-5-5c0-2.757 2.243-5 5-5z">
                </path>
            </svg>
            {{ room.participants.all.count }} Joined
        </a>
        <div class="like">
            <span id="likes-count-{{ room.id }}">{{ room.likes.count }}</span>
            <span class="like-button" data-item-id="{{ room.id }}">
                {% if user in room.likes.all %}
                <i class="fas fa-heart"></i>
                {% else %}
                <i class="far fa-heart"></i>
                {% endif %}
            </span>
        </div>
        <p class="roomListRoom__topic">
            {% if room.topic.name %}
                {{room.topic.name}}
            {% else %}
                None
            {% endif %}
        </p>
    </div>
</div>
{% endfor %}
```

Home View

```
def home(request):
    q = request.GET.get('q') if request.GET.get('q') != None else ''
    rooms = Room.objects.filter(
        Q(topic__name__icontains = q) |
        Q(name__icontains = q) |
        Q(description__icontains = q)
        )

    topics = Topic.objects.all()[0:5]
    room_count = rooms.count()

    room_messages = Message.objects.filter(Q(room__topic__name__icontains=q))

    context = {'rooms': rooms, 'topics': topics, 'room_count': room_count, 'room_messages':
room_messages}
    return render(request, 'base/home.html', context)
```

7.3.2 Login

Login.html

```
<form class="form" action="" method="POST">
          {% csrf_token %}
          <div class="form__group form__group">
            <label for="room_name">Email</label>
            <input id="username" name="email" type="text" placeholder="e.g. dennis_ivy" />
          </div>
          <div class="form__group">
            <label for="password">Password</label>
            <input id="password" name="password" type="password"
              placeholder="&bull;&bull;&bull;&bull;&bull;&bull;&bull;&bull;" />
          </div>

          <div class="isAdmin_checkbox">
            <input type="checkbox" name="isAdmin" value="True" id="">
            <label for="checkbox">Login is administartor</label>
          </div>

          <div id="forPass">
            <button class="btn btn--main" type="submit">
              <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0
32 32">
                <title>lock</title>
                <path
                  d="M27 12h-1v-2c0-5.514-4.486-10-10-10s-10 4.486-10 10v2h-1c-0.553 0-1 0.447-1 1v18c0
0.553 0.447 1 1 1h22c0.553 0 1-0.447 1-1v-18c0-0.553-0.447-1-1-1zM8 10c0-4.411 3.589-8 8-8s8 3.589 8
8v2h-16v-2zM26 30h-20v-16h20v16z">
                </path>
                <path
                  d="M15 21.694v4.306h2v-4.306c0.587-0.348 1-0.961 1-1.694 0-1.105-0.895-2-2-2s-2 0.895-2
2c0 0.732 0.413 1.345 1 1.694z">
                </path>
              </svg>

              Login
            </button>
            <div class="auth__action">
              <a href="{% url 'forgotPass' %}" class="btn btn--link">Forgot Password</a>
            </div>
          </div>

        </form>
```

Login View

```
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')
        isAdmin = request.POST.get('isAdmin')

        try:
            user = User.objects.get(email=email)
        except:
            pass

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

        if user is not None:
            login(request, user)

            if isAdmin:
                if user.is_staff:
                    return redirect('dashboard')
                else:
                    messages.error(request, 'You don\'t have the admin rights! Logged in as normal
user.')

            return redirect('home')
        else:
            messages.error(request, 'Username or Password does not exists')

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

7.3.3 Room Section

7.3.3.1 Create/Update Room

Create Room View

```
@login_required(login_url='login')
def createRoom(request):
    form = RoomForm()

    topics = Topic.objects.all()
    if request.method == 'POST':
        topic_name = request.POST.get('topic')
        topic, created =
Topic.objects.get_or_create(name=topic_name)
        Room.objects.create(
            host=request.user,
            topic=topic,
            name=request.POST.get('name'),
            description=request.POST.get('description'),
        )
        return redirect('home')

    context = {'form': form, 'topics': topics}
    return render(request, 'base/room_form.html', context)
```

Create/Update-Room.html

```
<main class="create-room layout">
    <div class="container">
      <div class="layout__box">
        <div class="layout__boxHeader">
          <div class="layout__boxTitle">
            <a href="{% url 'home' %}">
              <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0
                    32 32">
                <title>arrow-left</title>
                <path
                  d="M13.723 2.286l-13.723 13.714 13.719 13.714 1.616-1.611-10.96-10.96h27.625v-2.286h-
                    27.625l10.965-10.965-1.616-1.607z">
                </path>
              </svg>
            </a>
            <h3>Create/Update Study Room</h3>
          </div>
        </div>
        <div class="layout__body">
          <form class="form" action="" method="POST">
            {% csrf_token %}
            <div class="form__group">
              <label for="room_topic">Enter a Topic</label>
              <input type="text" value="{{ room.topic.name }}" name="topic" list="topic-list" required>
              <datalist id="topic-list">
                <select name="" id="room_topic">
                  {% for topic in topics %}
                    <option value="{{ topic.name }}">{{ topic.name }}</option>
                  {% endfor %}
                </select>
              </datalist>
            </div>
            <div class="form__group">
              <label for="room_name"> Room Name</label>
              {{ form.name }}
            </div>
            <div class="form__group">
              <label for="room_description"> Room Description </label>
              {{ form.description }}
            </div>
            <div class="form__action">
              <a class="btn btn--dark" href="{% url 'home' %}">Cancel</a>
              <button class="btn btn--main" type="submit">Submit</button>
            </div>
          </form>
        </div>
      </div>
    </div>
  </main>
```

UpdateRoom view

```
@login_required(login_url='login')
def updateRoom(request, pk):
    room = Room.objects.get(id = pk)
    form = RoomForm(instance=room)
    topics = Topic.objects.all()

    if request.user != room.host:
        return HttpResponse('Your are not allowed here!!')

    if request.method == 'POST':
        topic_name = request.POST.get('topic')
        topic, created =
Topic.objects.get_or_create(name=topic_name)form = RoomForm(request.POST,instance=room)
        room.name = request.POST.get('name')
        room.topic = topic
        room.description = request.POST.get('description')
        room.save()
        return redirect('home')

    context = {'form': form, 'topics': topics, 'room': room}
    return render(request, 'base/room_form.html', context)
```

7.3.3.2 Delete Room

DeleteRoom View

```
@login_required(login_url='login')
def deleteRoom(request, pk):
    room = Room.objects.get(id=pk)

    if request.user != room.host:
        return HttpResponse('Your are not allowed here!!')

    if request.method == 'POST':
        room.delete()
        return redirect('home')
    return render(request, 'base/delete.html', {'obj': room})
```

7.3.3.3 View Room

View-room.html

```
<div class="room">
  <div class="room__box scroll">
    <div class="room__header scroll">
      <div class="room__info">
        <h3>{{ room.name }}</h3>
        <span>{{ room.created|timesince}}</span>
      </div>
      <div class="room__hosted">
        <p>Hosted By</p>
        <div class="room_auto_with_topic">
          <a href="{% url 'user-profile' room.host.id %}" class="room__author">
            <div class="avatar avatar--small {% if user.is_active %}active{% endif %}">
              <img src="{{ room.host.avatar.url }}" />
            </div>
            <span>@{{ room.host.username }}</span>
          </a>
          <span class="room__topics">{{ room.topic }}</span>
        </div>
      </div>
    </div>
    <div class="room__conversation">
      <div class="threads scroll">
        {% for message in room_messages %}
        <div class="thread">
          <div class="thread__top">
            <div class="thread__author">
              <a href="{% url 'user-profile' message.user.id %}" class="thread__authorInfo">
                <div class="avatar avatar--small {% if message.user.is_active %}active{% endif %}">
                  <img src="{{ message.user.avatar.url }}" />
                </div>
                <span>@{{ message.user.username }}</span>
                {% if message.user.is_staff %}
                <span class="adminTag">STAFF</span>
                {% endif %}
              </a>
              <span class="thread__date">{{ message.created|timesince}} ago</span>
            </div>
            {% if request.user == message.user or request.user.is_staff %}
            <a href="{% url 'delete-message' message.id %}">
              <div class="thread__delete">
                  <title>remove</title>
                </svg>
              </div>
            </a>
            {% endif %}
          </div>
          <div class="thread__details">
            {{ message.body }}
          </div>
        </div>
        {% endfor %}
      </div>
    </div>
  </div>
  <div class="room__message">
    <form action="" method="POST">
      {% csrf_token %}
      {% if request.user.is_authenticated %}
      <input name="body" placeholder="Write your message here...">
      {% else %}
      <input name="body" placeholder=" Login to send messages..."
onclick="window.location.href='/login';"
        style="cursor: pointer;" readonly>
      <i class="fa-solid fa-user-lock"></i>
      {% endif %}
    </form>
  </div>
</div>
```

DisplayRoom View

```
def room(request, pk):
    room = Room.objects.get(id = pk)
    room_messages = room.message_set.all()
    participants = room.participants.all()

    if request.method == 'POST':
        message = Message.objects.create(
            user = request.user,
            room = room,
            body = request.POST.get('body')
        )
        room.participants.add(request.user)
        return redirect('room', pk = room.id)

    context = {'room': room, 'room_messages': room_messages, 'participants': participants}
    return render(request, 'base/room.html', context)
```

7.3.4 User Section

7.3.4.1 Register User

Register.html

```
<form class="form" action="" method="POST">
          {% csrf_token %}

          {% for field in form %}
          <div class="form__group form__group">
            <label for="room_name">{{ field.label }}</label>
            {{ field }}
          </div>
          {% endfor %}

          <button class="btn btn--main" type="submit">
            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32
32">
              <title>lock</title>
              <path
                d="M27 12h-1v-2c0-5.514-4.486-10-10-10s-10 4.486-10 10v2h-1c-0.553 0-1 0.447-1 1v18c0
0.553 0.447 1 1 1h22c0.553 0 1-0.447 1-1v-18c0-0.553-0.447-1-1-1zM8 10c0-4.411 3.589-8 8-8s8 3.589 8
8v2h-16v-2zM26 30h-20v-16h20v16z">
              </path>
              <path
                d="M15 21.694v4.306h2v-4.306c0.587-0.348 1-0.961 1-1.694 0-1.105-0.895-2-2-2s-2 0.895-2
2c0 0.732 0.413 1.345 1 1.694z">
              </path>
            </svg>

            Register
          </button>
        </form>
```

Register View

```
def registerPage(request):
    form = MyUserCreationForm()

    if request.method == 'POST':
        form = MyUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.username = user.username.lower()
            user.save()
            login(request, user)
            return redirect('home')
        else:
            messages.error(request, 'An error occured during registration!')

    return render(request, 'base/login_register.html', {'form': form})
```

7.3.4.2 View User

Profile.html

```
<div class="roomList">
        <div class="profile">
          <div class="profile__avatar">
            <div class="avatar avatar--large {% if user.is_active %}active{% endif %}">
              <img src="{{ user.avatar.url }}" />
            </div>
          </div>
          <div class="profile__info">
            <h3>{{ user.name }}</h3>
            <p> @{{user.username}} </p>
            {% if request.user == user %}
              <a href="{% url 'update-user' %}" class="btn btn--main btn--pill">Edit Profile</a>
            {% endif %}
          </div>
          <div class="profile__about">
            <h3>About</h3>
            <p>
              {{ user.bio }}
            </p>
          </div>
        </div>

        <div class="roomList__header">
          <div>
            <h2>Study Rooms Hosted by {{ user.username }}</a>
            </h2>
          </div>
        </div>

        {% include 'base/feed_component.html' %}
      </div>
```

7.3.4.3 Edit User

- *Profile View*

```python
def userProfile(request, pk):
    user = User.objects.get(id=pk)
    rooms = user.room_set.all()
    room_messages = user.message_set.all()
    topics = Topic.objects.all()
    context = {'user': user, 'rooms': rooms, 'room_messages': room_messages, 'topics': topics}
    return render(request, 'base/profile.html', context)
```

- *EditUser.html*

```
<main class="update-account layout">
        <div class="container">
            <div class="layout__box">
                <div class="layout__boxHeader">
                    <div class="layout__boxTitle">
                        <a href="{% url 'home' %}">
                            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
                                viewBox="0 0 32 32">
                                <title>arrow-left</title>
                                <path
                                    d="M13.723 2.286l-13.723 13.714 13.719 13.714 1.616-1.611-10.96-
                                    10.96h27.625v-2.286h-27.625l10.965-10.965-1.616-1.607z">
                                </path>
                            </svg>
                        </a>
                        <h3>Edit your profile</h3>
                    </div>
                </div>
                <div class="layout__body">
                    <form class="form" action="" method="POST" enctype="multipart/form-data">
                        {% csrf_token %}
                        {% for field in form %}
                        <div class="form__group">
                            <label for="profile_pic">{{ field.label }}</label>
                            {{ field }}
                        </div>
                        {% endfor %}
                        <div class="form__action">
                            <a class="btn btn--dark" href="{% url 'home' %}">Cancel</a>
                            <button class="btn btn--main" type="submit">Update</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
      </div>
    </main>
```

EditUser Views

```
@login_required(login_url='login')
def updateUser(request):
    user = request.user
    form = UserForm(instance=user)

    if request.method == 'POST':
        form = UserForm(request.POST, request.FILES, instance=user)
        if form.is_valid():
            form.save()
            return redirect('user-profile', pk=user.id)

    return render(request, 'base/update-user.html', {'form': form})
```

7.3.5 deleteMessage View

```
@login_required(login_url='login')
def deleteMessage(request, pk):
    message = Message.objects.get(id=pk)
    room = message.room  # Get the room associated with the message

    if request.user != message.user and not request.user.is_staff:
        return HttpResponse('Your are not allowed here!!')

    if request.method == 'POST':
        # Check if the user has any other messages in the room
        other_messages = Message.objects.filter(room=room, user=message.user).exclude(id=pk)

        # If no other messages by the user, remove the user from participants
        if not other_messages.exists():
            print(request.user)
            room.participants.remove(message.user)

        message.delete()
        return redirect('home')

    return render(request, 'base/delete.html', {'obj': message})
```

7.3.6 Delete.html

```
<main class="delete-item layout">
        <div class="container">
            <div class="layout__box">
                <div class="layout__boxHeader">
                    <div class="layout__boxTitle">
                        <a href="{{ request.META.HTTP_REFERER }}">
                            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
                                viewBox="0 0 32 32">
                                <title>arrow-left</title>
                                <path
                                    d="M13.723 2.286l-13.723 13.714 13.719 13.714 1.616-1.611-10.96-
10.96h27.625v-2.286h-27.625l10.965-10.965-1.616-1.607z">
                                </path>
                            </svg>
                        </a>
                        <h3>Back</h3>
                    </div>
                </div>
                <div class="layout__body">
                    <form class="form" action="" method="POST">
                        {% csrf_token %}
                        <div class="form__group">
                            <p>Are you sure you want to delete "{{obj}}"?</p>
                        </div>

                        <div class="for__group">
                            <input class="btn btn--main" type="submit" value="Confirm" />
                        </div>

                    </form>
                </div>
            </div>
        </div>
    </main>
```

7.3.7 Forms.py

```
from django.forms import ModelForm
from django.contrib.auth.forms import UserCreationForm
from .models import Room, User

class MyUserCreationForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['name', 'username', 'email', 'password1', 'password2']

class RoomForm(ModelForm):
    class Meta:
        model = Room
        fields = '__all__'
        exclude = ['host', 'participants']

class UserForm(ModelForm):
    class Meta:
        model = User
        fields = ['avatar', 'name', 'username', 'email', 'bio']

class UserAdminForm(ModelForm):
    class Meta:
        model = User
        fields = ['avatar', 'name', 'password', 'is_active', 'username', 'email', 'bio']
```

7.3.8 Modals

User Modal

```
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 = ['username']
```

Room Modal

```
class Room(models.Model):
    host = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    topic = models.ForeignKey(Topic, on_delete=models.SET_NULL, null=True)
    name = models.CharField(max_length=200)
    description = models.TextField(null = True, blank=True)
    participants = models.ManyToManyField(User, related_name='participants', blank=True)
    updated = models.DateTimeField(auto_now=True)
    created = models.DateTimeField(auto_now_add=True)
    likes = models.ManyToManyField(User, related_name='liked_items', blank=True)

    class Meta:
        ordering = ['-updated', '-created']

    def __str__(self):
        return self.name
```

Message Modal

```
class Message(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    room = models.ForeignKey(Room, on_delete=models.CASCADE)
    body = models.TextField()
    updated = models.DateTimeField(auto_now=True)
    created = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['-updated', '-created']

    def __str__(self):
        return self.body[0:50]
```

Topic Modal

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

    def __str__(self):
        return self.name
```

8 Testing and Results

8.1 Testing

A Test Plan can be defined as a document describing the scope, resources, and schedule of intended Testing activities. A project may fail without a complete Test Plan. Test planning is particularly important in large software system development.

8.2 Test Case Result

No.	Scenario	Result
1.	Verify that a user with valid credentials can successfully log in to the system.	Yes
2.	Verify that a user with invalid credentials cannot log in to the system and sees an error message.	Yes
3.	Verify that an admin user can successfully edit the details of a room they host	Yes
4.	Verify that a logged-in user can post a new message in the room's conversation thread.	Yes
5.	Verify that a logged-out user cannot post a message and is redirected to the login page.	Yes
6.	Verify that a user can view a list of participants currently joined in a room.	Yes
7.	Verify that a user can delete their own messages from the conversation thread.	Yes
8.	Verify that an admin user can delete any message posted in a room they host.	Yes

9 Conclusion

9.1 Future Enhancement

Potential enhancements include:

- Integration of video conferencing tools
- Real-time collaboration tools (e.g., live document editing)
- A mobile application for better accessibility
- Enhanced room analytics to track study habits
- Improved User Roles and Permissions
- Real-Time Notifications
- AI-Powered Features

9.2 Limitations

The platform currently lacks real-time collaboration features, such as live document editing or video conferencing, which could enhance the study experience. It is also limited to web access without a dedicated mobile application.

- Limited Scalability of SQLite
- Lack of Real-Time Features
- Dependency on Internet Connectivity
- Basic Security Features
- Limited Media Support

10 Bibliography

10.1 HTML

Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser. It defines the content and structure of web content.

10.2 CSS

Cascading Style Sheets is a style sheet language used for specifying the presentation and styling of a document written in a markup language such as HTML or XML.

10.3 JavaScript

JavaScript, often abbreviated as JS, is a programming language and core technology of the Web, alongside HTML and CSS. 99% of websites use JavaScript on the client side for webpage behaviour.

10.4 Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage collected.

10.5 Django

Django is a free and open-source, Python-based web framework that runs on a web server. It follows the model–template–views architectural pattern. It is maintained by the Django Software Foundation, an independent organization established in the US as a 501 non-profit.

10.6 SQLite

SQLite is a database engine written in the C programming language. It is not a standalone app; rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases.

11 References

11.1 Web References

- Python for Beginners (Full Course) | #100DaysOfCode Programming Tutorial in Hindi by CodeWithHarry
- Python Django 7 Hour Course by Traversy Media
- Complete Django Tutorial for Beginners to Advanced [Hindi] by WsCube

11.2 Books References

- "Core Python Programming" by Dr. R. Nageswara Rao – 2017 Edition, Dreamtech Press
- Django 5 By Example - Fifth Edition
- Ultimate Django for Web App Development Using Python: Build Modern, Reliable and Scalable Production-Grade Web Applications with Django and Python