Project Report

Software Engineering UCS-503

Notes Sharing Website

Prepared by : Team Pixel

Arshdeep Palial 102216099 Pranav Duggal 102216023

Jaskaran Singh 102216110

Tarun Bhatti 102216105

Software Bid/ Project Teams

UCS 503- Software Engineering Lab

Group : CS14 , CS11 Dated: 27/1/2024

Team Name: PIXEL
Team ID (will be assigned by Instructor):

Please enter the names of your Preferred Team Members.

*You are required to form **a three to four person** teams

*Choose your team members wisely. You will not be allowed to change teams.

Name	Roll No	Project Experience	Programming Language used
ARSHDEEP PALIAL	10216099	FULL STACK WEB APPS , BASIC GAMES , ARDUINO IOT PROJECTS	PYTHON, REACTJS , C/C++
TARUN BHATTI	102216105	FORNTEND DESIGNS FOR WEB APPS	JS , JQUERY
PRANAV DUGGAL	102216023	WEB AND ANDROID APPLICATIONS	REACTJS,FLUTTER

JASKARAN SINGH	102216110	FORNTEND DESIGNS FOR WEB APPS	JS , JQUERY

Programming Language / Environment Experience

List the languages you are most comfortable developing in, **as a team**, in your order of preference. Many of the projects involve Java or C/C++ programming.

1. Python
2. JavaScript
3. C/C++

Choices of Projects:

Please select **4 projects** your team would like to work on, by order of preference: *[Write at-least one paragraph for each choice (motivation, reason for choice, feasibility analysis, etc.)]*

First Choice	A WEB APP TO PROVIDE , CONNECT AND GUIDE STUDENTS .
Second Choice	A WEB APP TO CONNECT EVERY STUDENT IN COLLEGE .
Third Choice	HOSPITAL MANAGEMENT.

Fourth Choice	ECOMMERECE WEBSITE.

Additional Remarks/ Inputs

Please tell us about any other factors that we should take into consideration (e.g., if you really would like to work on a project for some particularly convincing reason).

Web app to provide, connect and guide students.

Helping students by

* PROVIDE -The app provides students with notes uploaded by authorized individuals. Student ratings ensure the quality and relevance of the notes.

* Connect - Experience a new level of social interaction! Connect with students from different branches and widen your social circle with just a few taps.

* Guide - Guiding light for students, helping them to learn the skills they need to achieve a brighter future.

Project Write Up

NOTO

Team Pixel

Executive Summary: Note Sharing Website Project

Noto is a revolutionary web-based platform that transforms notes sharing into a visually stimulating and collaborative experience. Inspired by the familiar and engaging interface of Instagram, Noto empowers students to learn , share , communicate using captivating and simple interface.

Problem:

Problems with Traditional Studying:

Isolation: Studying feels like a solitary activity, leading to a lack of motivation and engagement.

Inefficiency: Traditional note-taking methods can be time-consuming and result in disorganized information.

Limited Perspective: Students may rely solely on their own notes, which might not be the most comprehensive or effective.

Difficulty Finding Study Partners: Students might not know the right people to connect with to form study groups.

Solutions Offered by Noto:

Social Learning: Connect with classmates and study partners, fostering a collaborative and motivating learning environment.

Visually Engaging Notes: Create visually appealing notes with images, diagrams, and creative layouts to boost information retention.

Community-Sourced Knowledge: Explore a vast library of notes shared by other students, gaining access to diverse perspectives and valuable resources.

Streamlined Organization: Categorize notes with relevant hashtags for easy retrieval and efficient studying.

Novelty/ Unique Selling Point :

USP 1: **The Social Clipboard** - Noto fuses the visual engagement of Instagram with the collaborative power of shared study materials, creating a vibrant social space where students can learn from each other and boost their academic performance.

USP 2: **Take Control:** Noto empowers students to actively curate their own learning experience by selecting the best resources and collaborating with peers. It's not just passive consumption, but active engagement in the learning process.

Objectives :

Enhance Student Engagement: Transform note-taking from a solitary chore into a visually stimulating and socially interactive experience. This fosters deeper understanding and motivates students to actively participate in the learning process.

Promote Collaborative Learning: Facilitate seamless sharing of notes and resources between classmates. Encourage discussions and feedback through comments and mentions, fostering a supportive learning community.
Elevate Note-Taking Practices: Move beyond plain text notes. Encourage students to utilize images, diagrams, and creative layouts to create visually engaging and memorable study materials.

Empower Knowledge Discovery: Build a vast library of user-generated notes, categorized by hashtags and searchable for easy access. This allows students to discover diverse perspectives, find valuable resources, and fill any gaps in their own understanding.

Streamline Organization and Retrieval: Implement a user-friendly system for categorizing notes with relevant hashtags. This allows for efficient searching and retrieval, saving students valuable time during their studies.

Project Deliverables/Outcomes:

1. **An Note Sharing Website** - Web based platform accessible to students and the teachers.
2. **Documentation** - Software bid, Project write up, Feasibility report, UML diagrams, SRS, Project report.

Product Perspective: Noto

Stakeholders:

- **Primary:** Students (of all ages and disciplines)
- **Secondary:** Educators, Professionals

Product Vision:

- To revolutionize student learning by creating a visually engaging, collaborative platform for note-taking, knowledge sharing, and community building.

User Needs:

- **Students:**

- Capture and organize information effectively (visually appealing notes, hashtags for searchability).
- Collaborate with peers (share notes, discuss concepts, find study partners).

- Access diverse learning resources (explore public notes, discover different perspectives).
- Enhance information retention (visually-driven content, engaging learning environment).

- **Educators:**

- Facilitate student engagement (encourage active learning through visuallydriven notes).
- Promote collaborative learning (provide a platform for students to share resources and discuss concepts).
- Monitor student progress (access to student-generated notes, gauge understanding through discussion threads).
- Learn and upskill efficiently (access to user-generated content, discover best practices).
- Connect with like-minded individuals (build professional networks, share industry knowledge).

- **Product Features:**

- **Seamless sharing:** Share notes privately with study groups or publicly for wider access.
- **Public note discovery:** Explore a vast library of public notes shared by other users.
- **Number of active users:** Students, educators, and professionals using the platform.
- **Community growth:** Number of connections formed, active discussion threads, collaborative learning initiatives.
- **Knowledge discovery:** Frequency of searches, utilization of public notes, positive feedback on resource availability.
- **Scalability and Flexibility:** The platform is designed to be scalable and flexible, allowing small business owners to adapt and expand their online stores as their businesses grow. This includes features such as customizable pricing plans, and the ability to add new products and features as needed.

Noto: Look and Feel

Concept:

Noto draws inspiration from the familiar and user-friendly interface of Instagram, but tailors it specifically for the needs of note-taking and knowledge sharing. The overall aesthetic should be:

Visually Appealing: Leverage bright colors, clean layouts, and high-quality visuals to create an engaging experience.
Modern and Minimalist: Prioritize clean lines, intuitive navigation, and easy-tounderstand icons and buttons.
Social and Collaborative: Emphasize user profiles, connections, and features that encourage interaction between users.

Key Design Elements:

Home Feed: The main screen displays a curated feed of notes shared by users you follow and relevant public notes based on your interests and hashtags. Each note appears like a "post" with a visually captivating image preview, a short caption, and engagement options (likes, comments).

Note Creation: Creating a note resembles creating a new post on Instagram. Users can upload images, diagrams, or screenshots, add text overlays for explanations, and include relevant hashtags.

Profile Section: Users can personalize their profiles with a profile picture, a short bio, and links to their social media (optional). The profile also displays their uploaded notes, the number of followers and following, and badges earned (if a gamification element is implemented).

Search Function: A powerful search bar allows users to find specific notes using keywords and hashtags. Search results display relevant notes along with user profiles and relevant subject tags.

Navigation: A clear and intuitive navigation bar at the bottom of the screen provides easy access to the home feed, search function, profile section, and potentially a messaging or notification section.

Scope of Application

Higher Education: College and university students can create visually-driven lecture notes, collaborate on group projects, and discover study resources shared by peers. Professors can use Noto to share course materials, facilitate online discussions, and encourage peer learning.

Online Learning Platforms: Noto can integrate with online learning platforms to provide students with a collaborative note-taking and knowledge sharing tool. This can be particularly beneficial for self-paced learning courses.

Professional Development: Professionals can use Noto to capture key takeaways from training sessions, share industry best practices, and collaborate on ongoing learning initiatives.

Gant Chart for project

NOTO

Gantt Chart

TASK / PROCESS	Feburary	March	April	May
Inception	1 FEB - 7 FEB			
Requirement Gathering	7 FEB - 20 FEB			
Planning	20 FEB - 25 FEB			
Design	25 FEB - 10 MARCH			
Development	10 MARCH - 10 APRIL			
Test	10 APRIL- 10 MAY			
Deployment	11 MAY - 12 MAY			

UI/UX Design

Create an account

Log In

Email address

Password

Forgt your pasword?

Log in

"Lorem ipsum dolor sit amet. consectetur adipiscing elit, sed do eiusmod tempor incididunt ut Iabore et dolore magna aliqua"

John Doe, Happy Customer

BELIEVING

Focusing On What Matters Most

Lorem ipsum dolor sit amet, consectetur adipiscing eli, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam

Sed ut perspicitis unde omnis iste natus errorsit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae abillo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

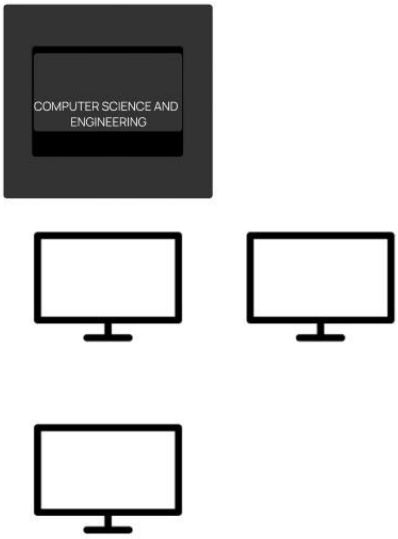
Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet

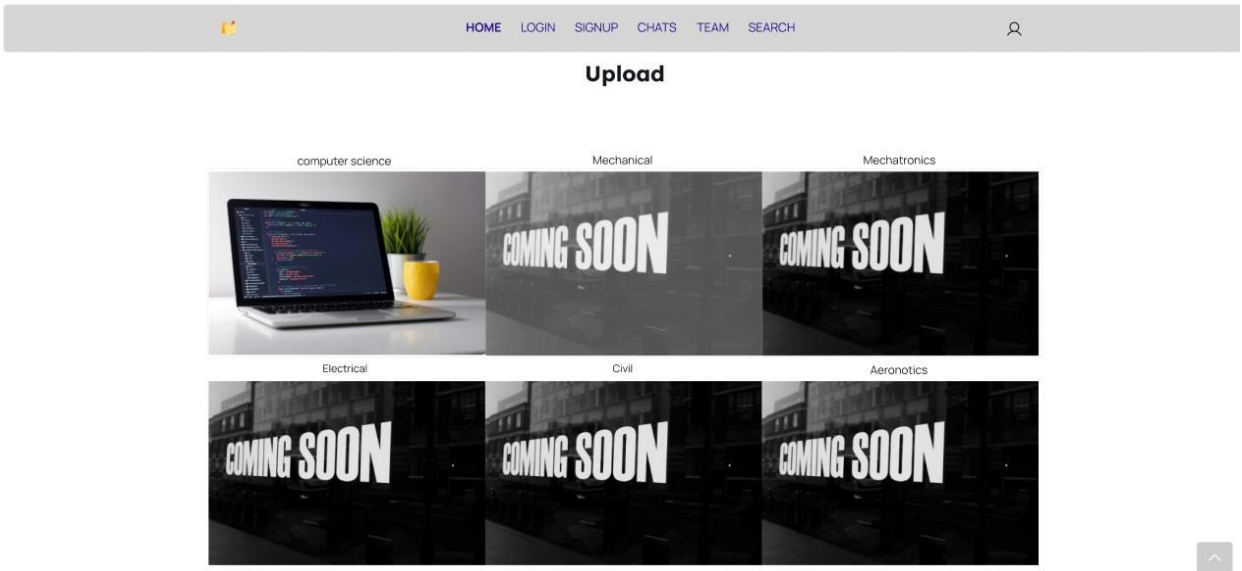
WATCH VIDEO

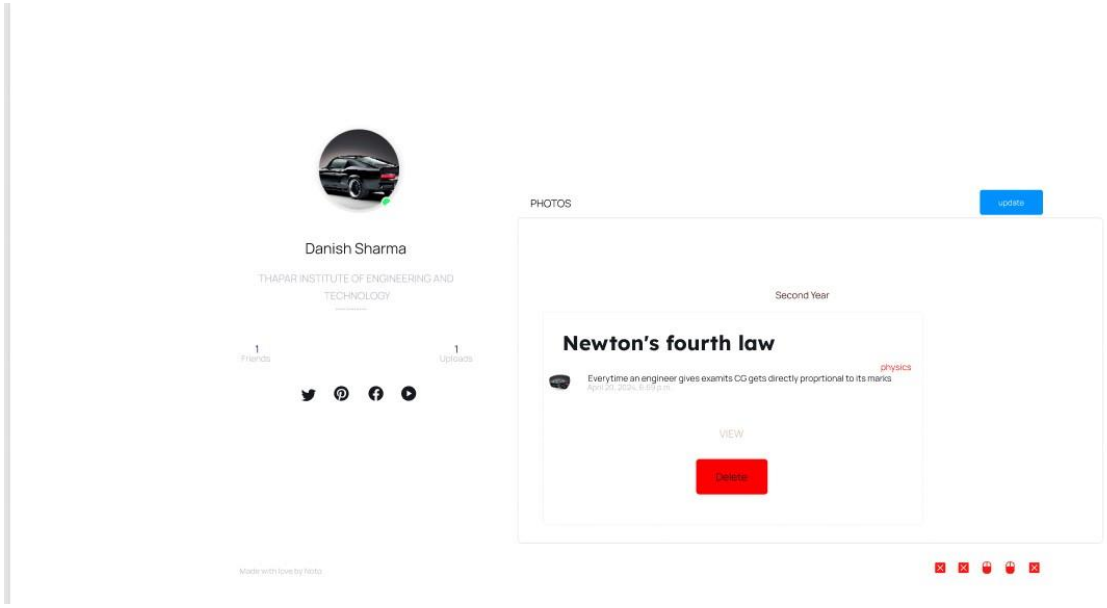
TRUST

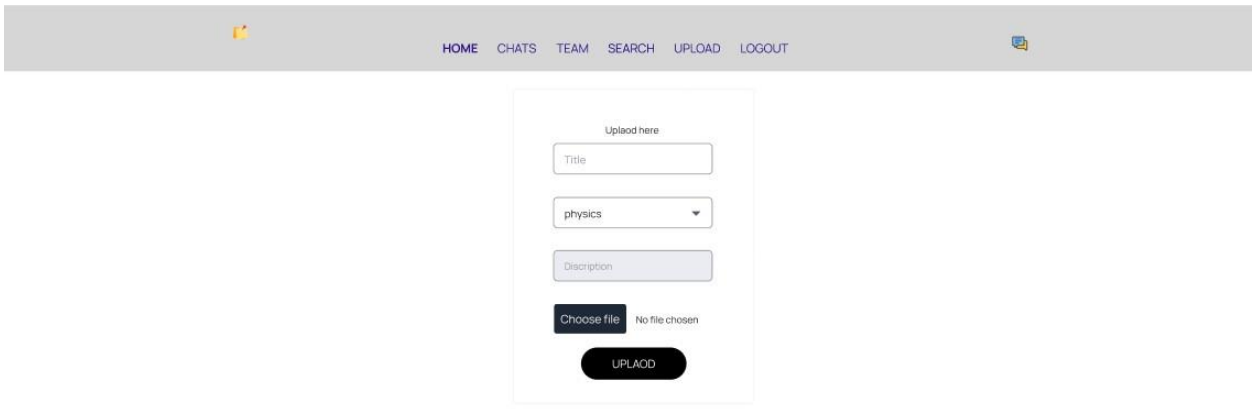
Branches we have

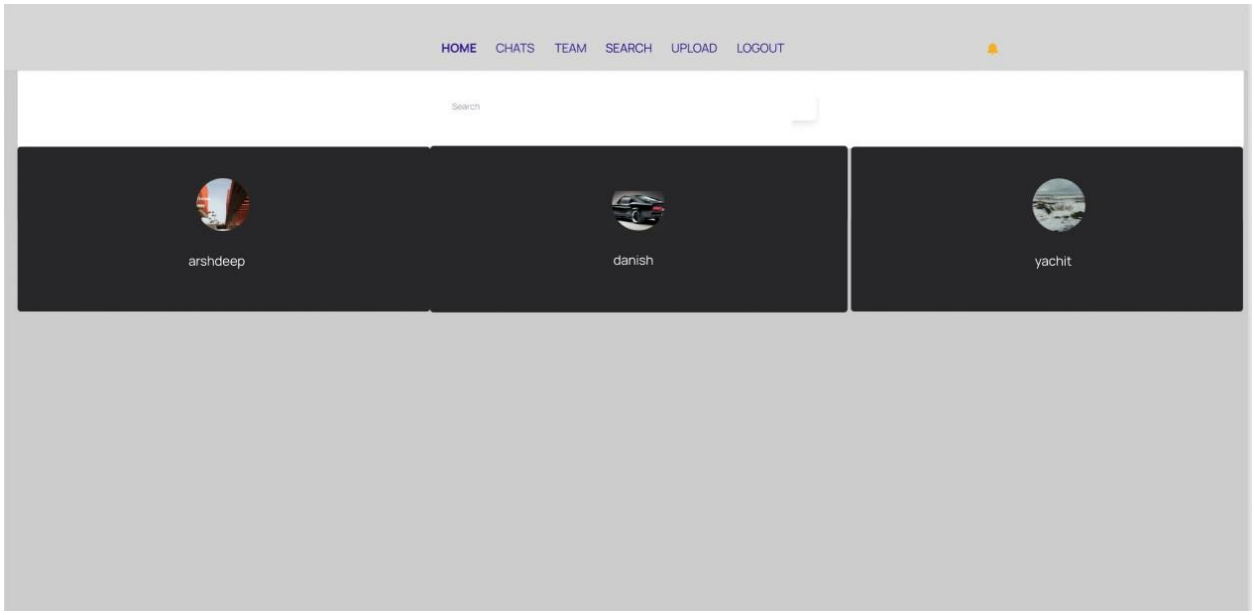
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam

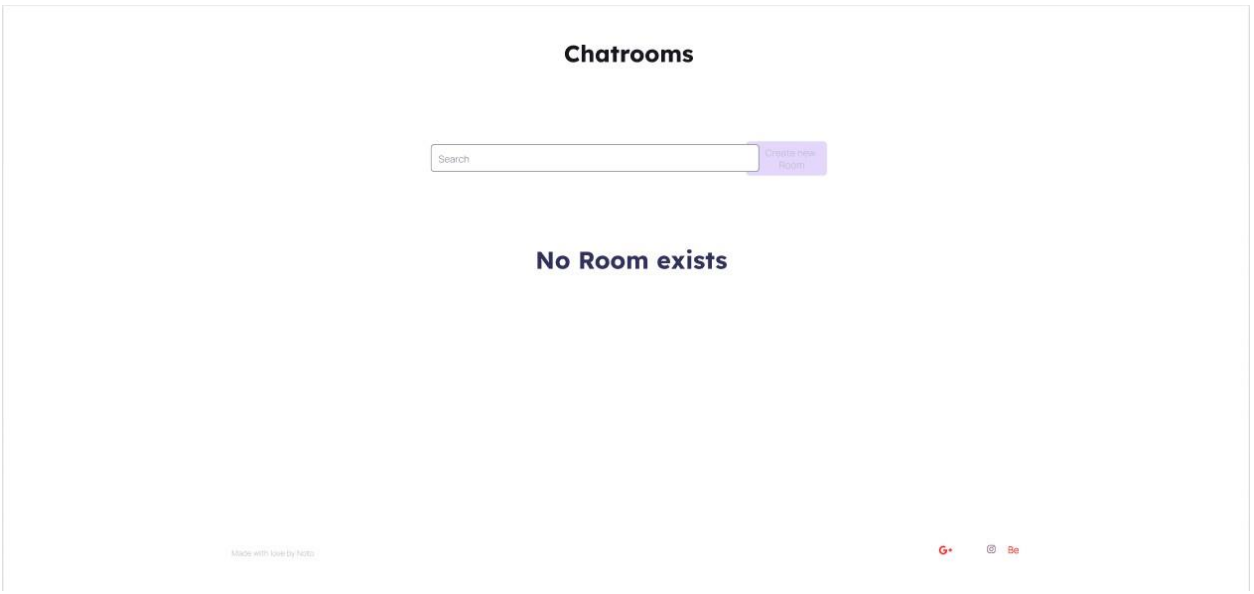


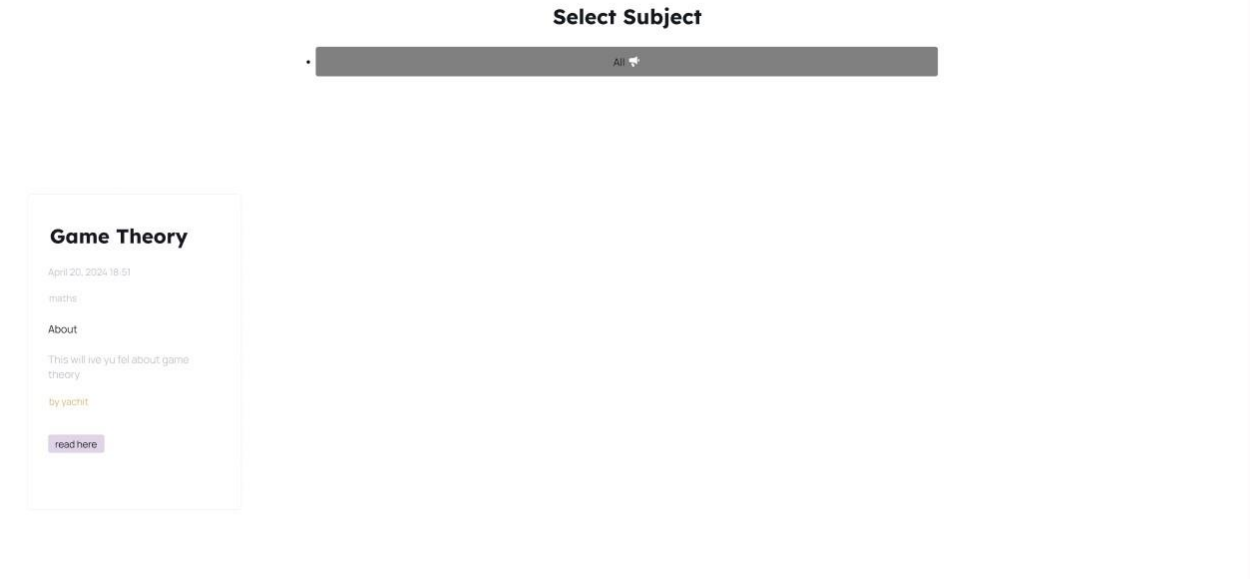

HOME
LOGIN
SIGNUP
CHATS
TEAM
SEARCH
Upload
computer science
Mechanical
Mechatronics
COMING SOON
COMING SOON
Electrical
Civil
Aeronotics
COMING SOON
COMING SOON
COMING SOON


Danish Sharma
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
1
Friends
1
Uploads
PHOTOS
update
Second Year
Newton's fourth law
physics
Everytime an engineer gives examits CG gets directly proprtional to its marks
VIEW
Delete


HOME
CHATS
TEAM
SEARCH
UPLOAD
LOGOUT
Uplaod here
Title
physics
Discription
Choose file
No file chosen
UPLAOD


HOME
CHATS
TEAM
SEARCH
UPLOAD
LOGOUT
Search
arshdeep
danish
yachit


Chatrooms
Search
Create new Room
No Room exists
Made with love by Notts
Be


Select Subject
All
Game Theory
April 20, 2024 18:51
maths
About
This will ive yu fel about game theory
by yachit
read here

Software Requirements Specification

for

Notes Sharing Website

Prepared by Team Pixel

Table of Contents

1. Introduction

Purpose

This document outlines the requirements for a web application, codenamed "Noto" designed to function as a question-and-answer platform specifically for university students and their coursework.

Document Conventions

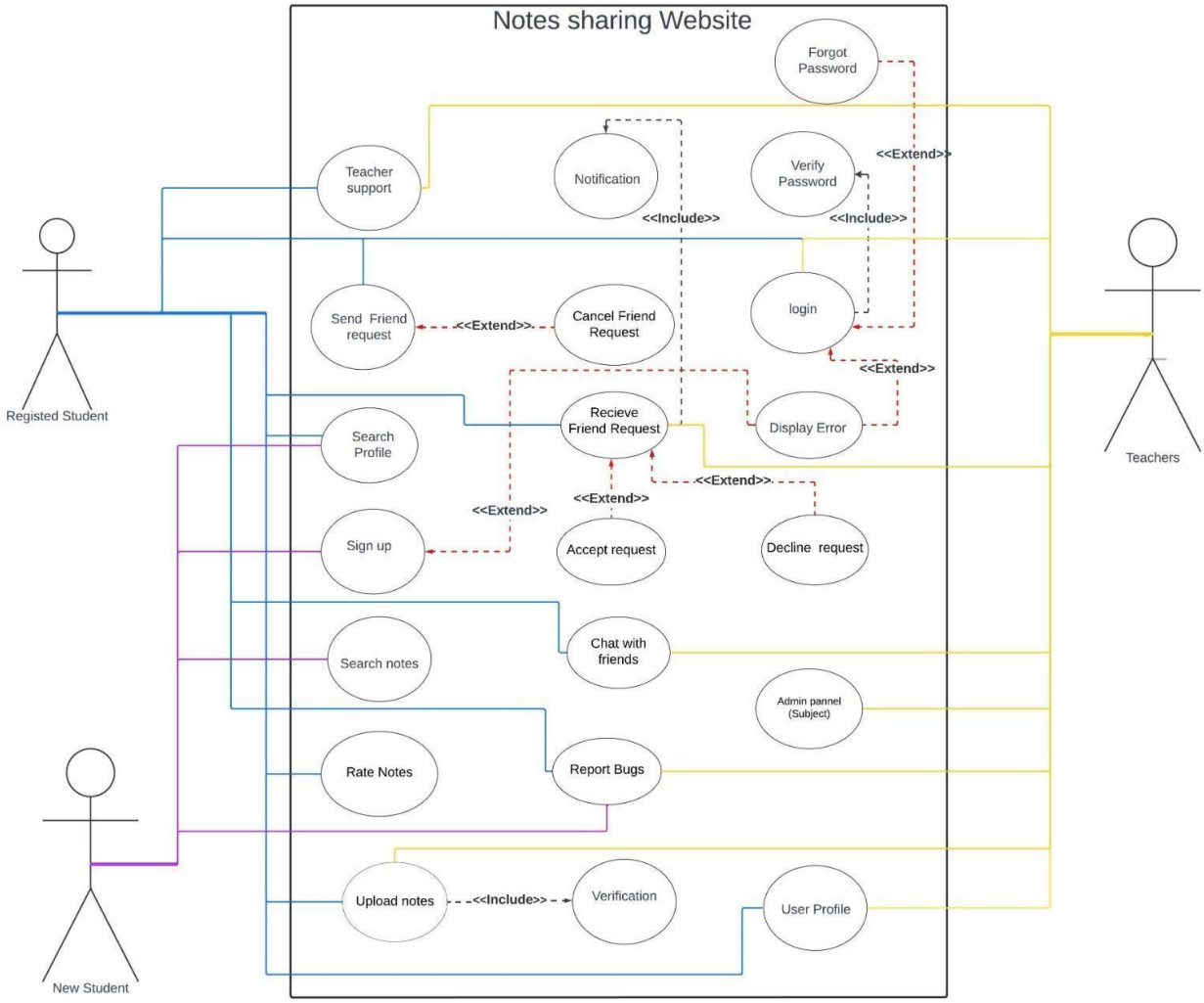
The functional requirements for this project are organized by use case within use case with user class.

Intended Audience and Reading Suggestions

Noto will be a web-based platform where students can ask questions related to their academic subjects, courses, assignments, and other university-related topics. The platform will foster a collaborative learning environment by allowing other students and verified educators to answer questions and provide explanations.

2. Overall Description

User Classes and Characteristics: UML Use Case Diagram



Operating Environment

Noto will be a web-based platform and shall be compatible with web browsers on windows and mac and web enabled smartphones.

Design and Implementation Constraints

CO-1: The Administrator shall be able to maintain and modify the site without requiring any specialized technical knowledge beyond basic HTML or the use of site editing tools provided by a commercial web site builder, such as WordPress.

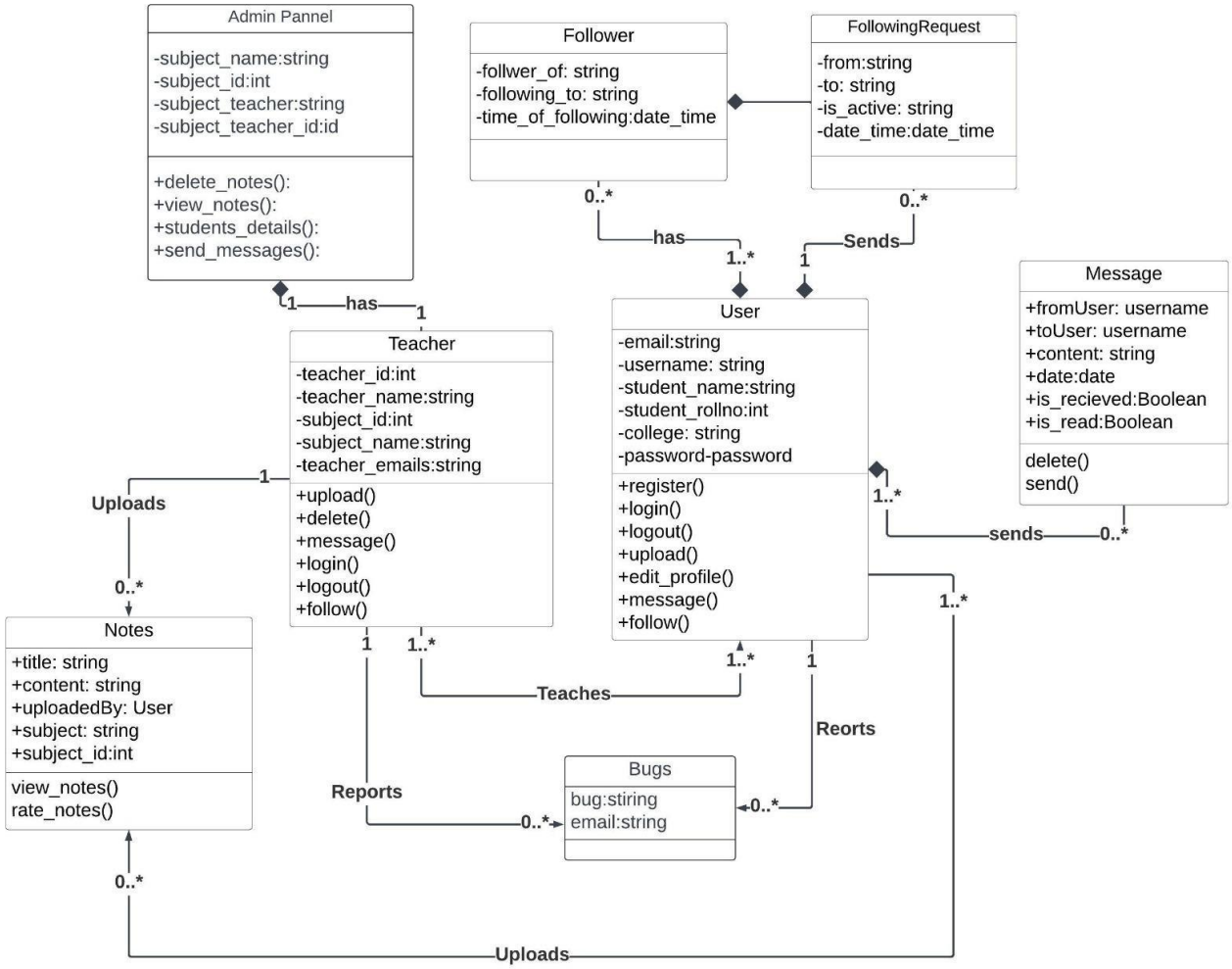
CO-2: The website shall be developed using the Java platform and design using HTML, CSS, and JS.

CO-3: The Backend of the website shall be developed using python framework Django.

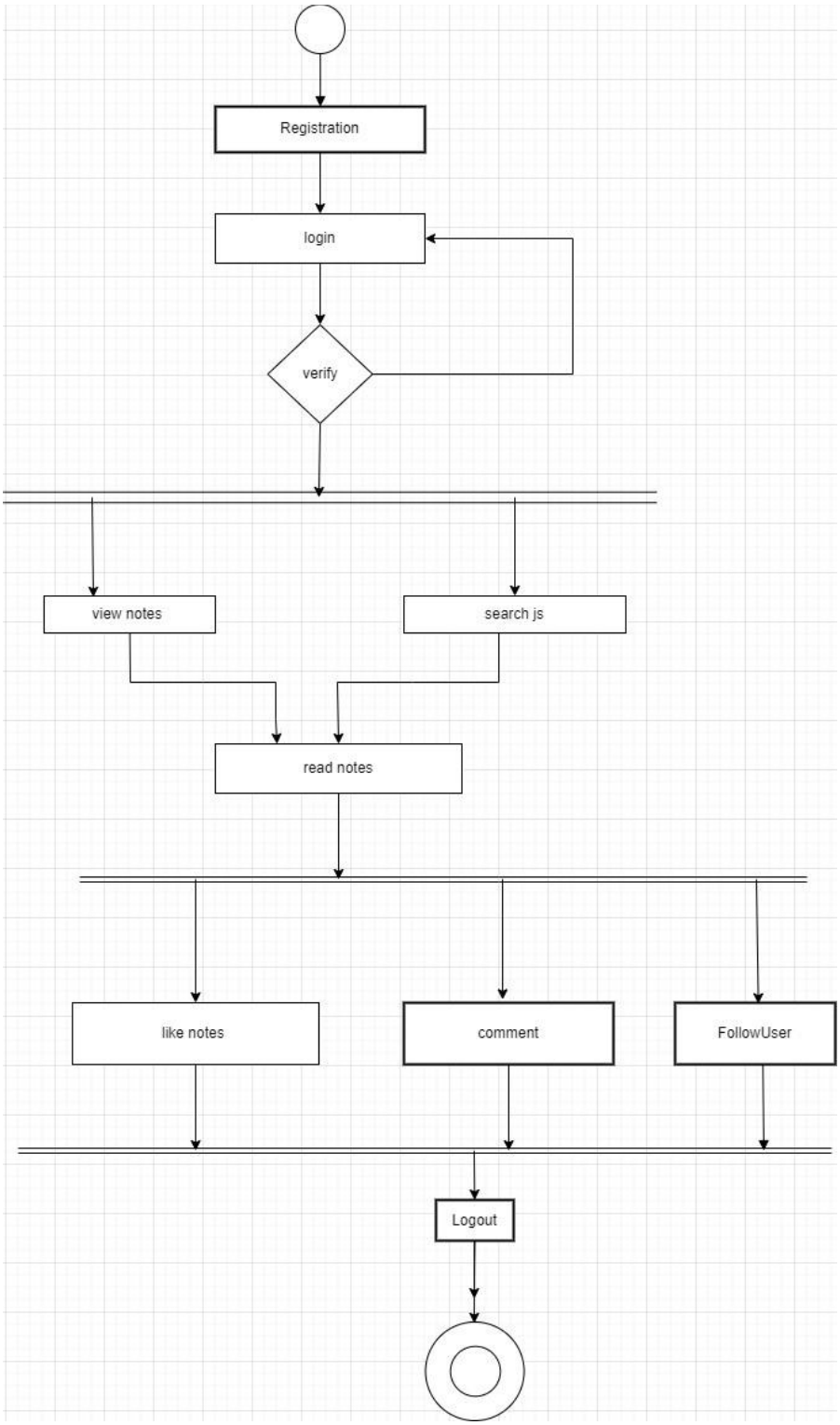
User Documentation

The developer shall provide necessary documentation to enable an Administrator to perform the use cases listed in section 2.1.4 Administrator.

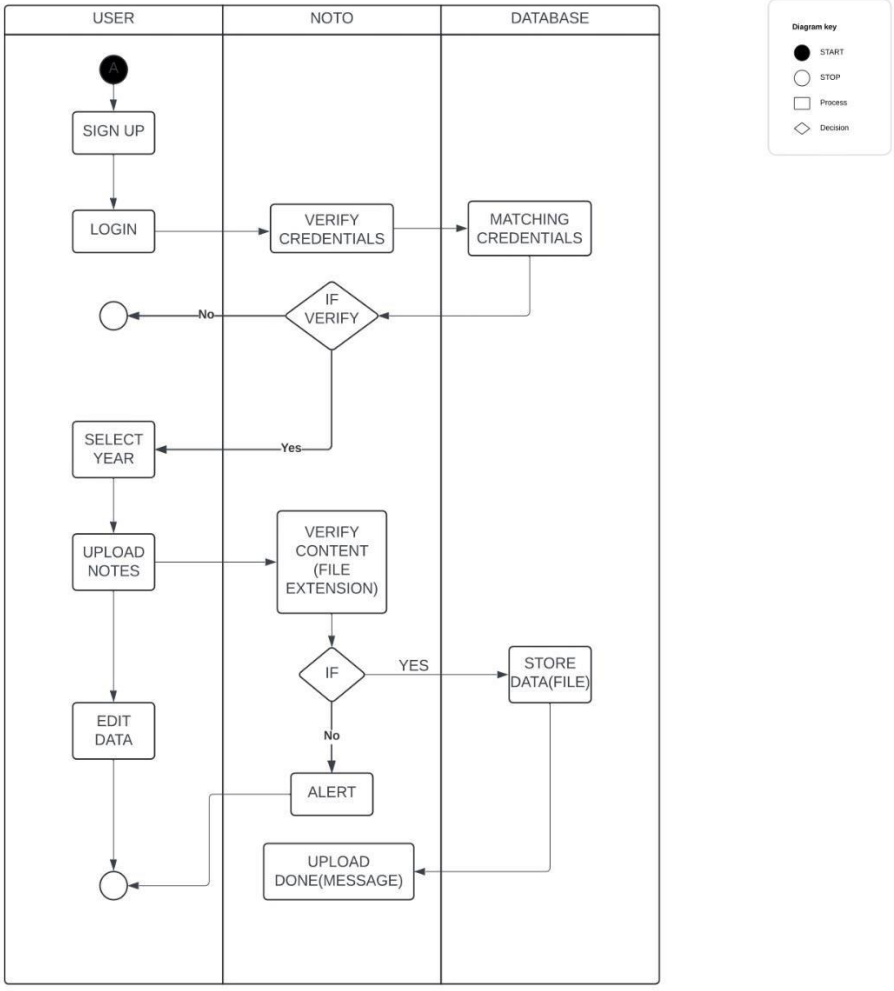
UML Class Diagram



Activity diagram



Activity Swimlane



3. System Features/Use Case Details

Use Cases for Guest

UC-G-1: Login to system

Description: A guest shall be able to login to the system to do more actions.

Login form:	A Guest shall be able to login to the system by entering their email address and password that are available in the system accounts list to the form.	(Priority=H)
Login method:	A Guest shall be allowed to login by connecting with a facebook account or a gmail account.	(Priority=H)
Login recaptcha:	A Guest shall check re-CAPTCHA to confirm that it is a human being.	(Priority=H)
Login recaptcha error:	If a Guest doesn't check re-CAPTCHA, the website shall display an error message.	(Priority=H)
Login valid account:	The application shall display an error message if the Guest enters invalid email address or password that is not available in the system account list.	(Priority=H)
Login redirect verify:	If a Guest enters an account that is inactive, the application shall redirect the guest to a verified page.	(Priority=H)
Login verify:	The Guest shall enter a code that the system has sent to their email.	(Priority=H)
Login verify error:	If a Guest enters an invalid verified code, the application shall display an error message. A code may be invalid because the code doesn't match the code in a guest's email.	(Proprity=H)
Login redirect home:	The application shall redirect a user to home if a user enters a valid account that has been verified and already checked reCAPTCHA. After login, the user may be a customer or an admin depending on their role and their using environment.	(Priority=H)

UC-G-2: Register

Register form:	A Guest shall be able to create new account by entering email address, password and confirm password and other information such as first name, last name, address, phone .	(Priority=H)

Register valid email:	The application shall display an error message if the e-mail address that the Guest enters is missing or not properly formatted. A valid e-mail address contains only letters and numbers, must contain exactly one "@" following one or more characters, must contain exactly one period following the "@" with any number of intervening characters, and must have at least one character following the period.	(Priority=H)
Register valid email:	The application shall display an error message if the e-mail address that the Guest enters is missing or not properly formatted. A valid e-mail address contains only letters and numbers, must contain exactly one "@" following one or more characters, must contain exactly one period following the "@" with any number of intervening characters, and must have at least one character following the period .	(Priority=H)
Register valid password	The application shall display an error message if the password is not matched with the proposed format of the application, that would be more than 6 characters and included: word, number and special characters (like @, #, $...)	(Priority=H)
Register valid confirm:	The website shall display an error message if a guest doesn't enter the confirm password that doesn't match the password.	(Priority=H)
Register valid field:	The website shall display error messages if any text fields are empty.	(Priority=H)
	The application shall display an error message if the guest enters invalid email address or password that is not available in the system account list.	(Priority=H)
Register method:	A Guest shall be able to sign up by connecting with a facebook account or a gmail account.	(Priority=H)
Register send:	The application shall send the verified code via email to a guest's email.	(Priority=H)
Register redirect login	The application shall redirect a guest to login form if a guest creates a new account successfully.	(Priority=H)

UC-G-4: View information details for th Notes

Notes detail view:	A Guest that on customer site shall be able to view detail of a car such as notes name, seats, feedbacks,…	(Priority=H)

UC-G-5: Search Notes

Notes search:	A Guest that on customer site shall be able to search available cars by entering notes name or choosing notes type, notes quantity, rental date and return date.	(Priority=H)

Use Cases for Authenticated User

UC-AU-1: Logout

Description: An Authenticated User shall be able to logout application.

UC-AU-2: Change password

Description: An Authenticated User shall be able change his password. The application shall display an error message if he entered the wrong previous password or invalid new password field.

Constraint: The valid password must be more than 6 characters and contains at least one word, one number and one special character (like @, #, $ ...)

UC-AU-3: Manage profile

Description: An Authenticated User shall be able to manage his profile.

Profile avatar add:	An Authenticated User shall be able to add an avatar to profile.	(Priority=L)
Profile add error:	If an Authenticated User add an image that bigger than 900 x 900, the application shall display an error message.	(Priority=L)
Profile information modify:	An Authenticated User shall be able to modify profile information such as name, address, phone.	(Priority=M)
Profile modify error:	The application shall display error messages if user enters empty text fields.	(Priority=M)

4. External Interface Requirements

User Interfaces

4.1.1 Standard website

- In case the users are using application on the website, includes: Customer, Associated Company and Admin. These are the following interface similarities:
- Must have a header with navigation bar with View Login and Register link items.
- Must have a footer with show the information such as: Address, Road map, Website copyright information, Link to social networking sites and menu. ○ Header and footer must have the same color and should not be too dark to see. ○ Content of the website must be on the center of the screen.

4.1.2 Login interface

When asks the user to type his username and password, if the user entered either his username and password incorrectly then an error message occurs.

Must have the link to register page.

4.1.3 Register interface

When the user type information details, if the user entered the input values have not validate, then error message shall be shown below the fields.

Must have the link to login page.

4.1.4 Search

The customer or associated company can enter the topic name he is looking for or select the course of subject he needs to search.

4.1.5 Admin's control panel:

This control panel will allow admin to add, remove or ban users; to get and delete feedback from other users on the platform.

Software Interfaces

Login using Google or Facebook or X(formerly Twitter).

5. Other Nonfunctional Requirements

Performance Requirements

The web pages shall fully paint in an average response time of 5 seconds or less over a broadband (DSL or cable) Internet connection.

Security

The website shall use standard Web security protocols (HTTPS/SSL 3.0) when transferring any private information regarding a Guest, Customer, or Associated Company and order information of customer.

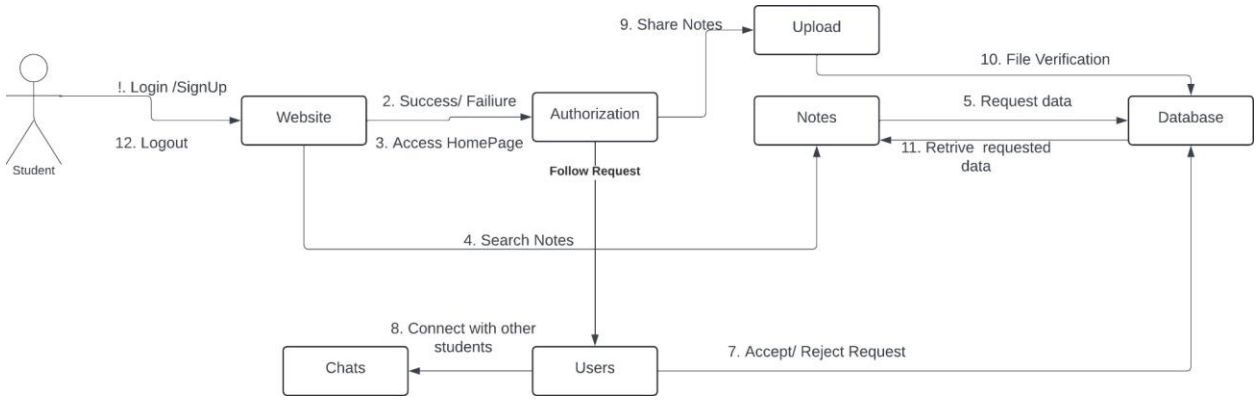
Maintainability

The code must be clear and match the correct syntax and rule in python(django) language.

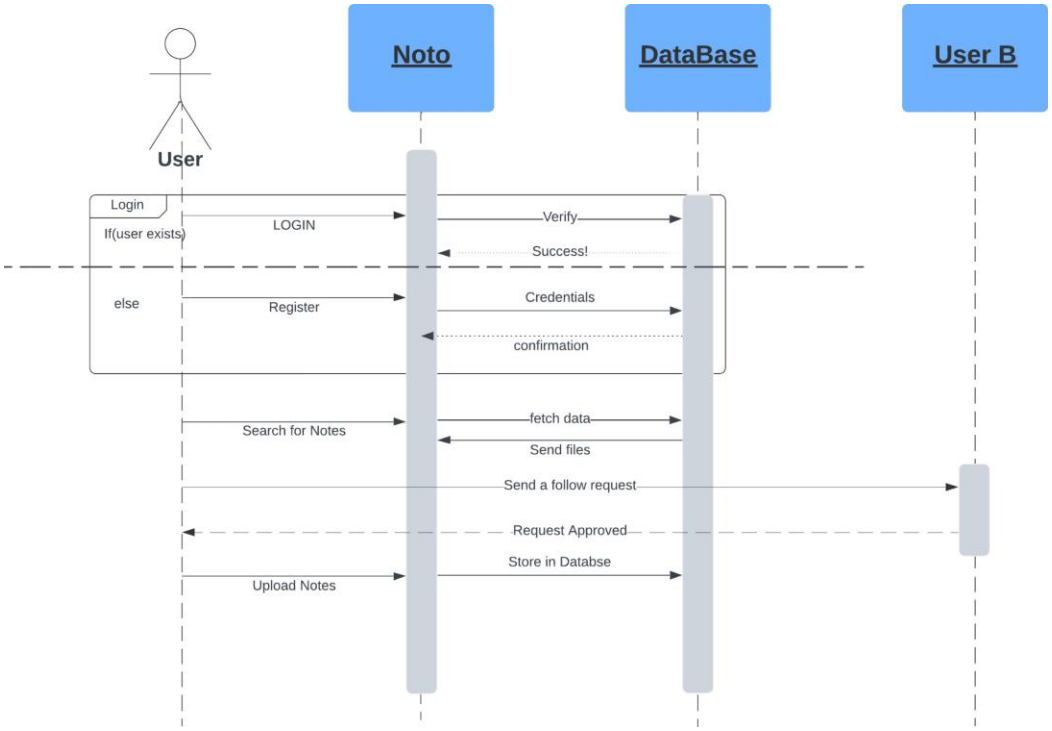
Legal and Copyright

Not allow to use image or information of associated company for external distribution.

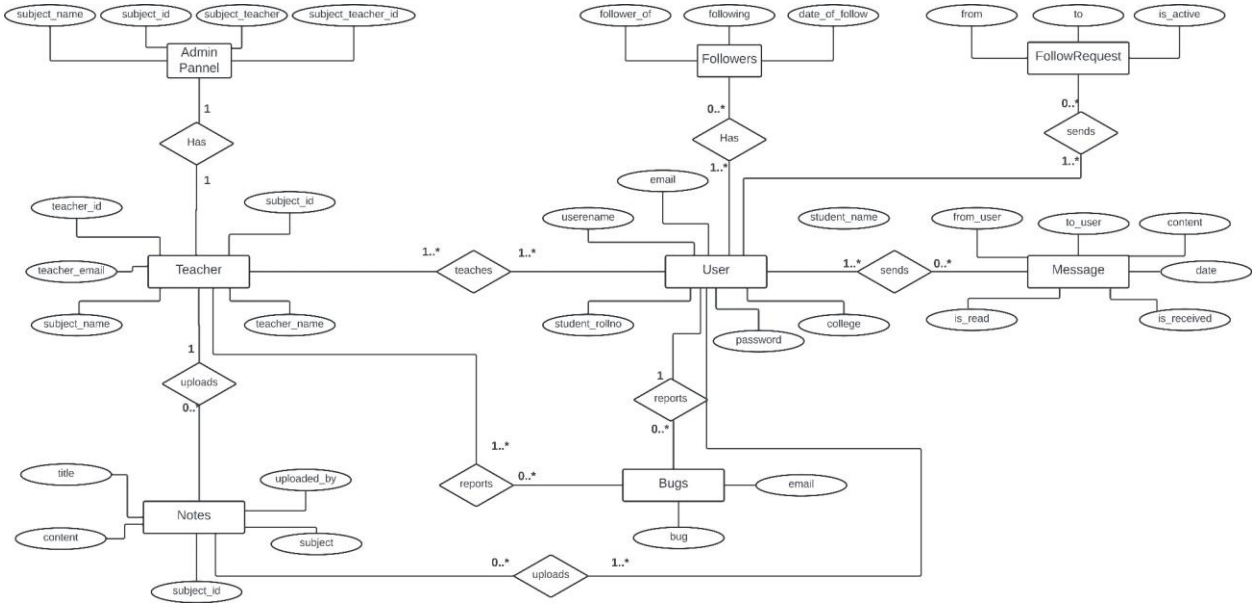
Collaboration Diagram



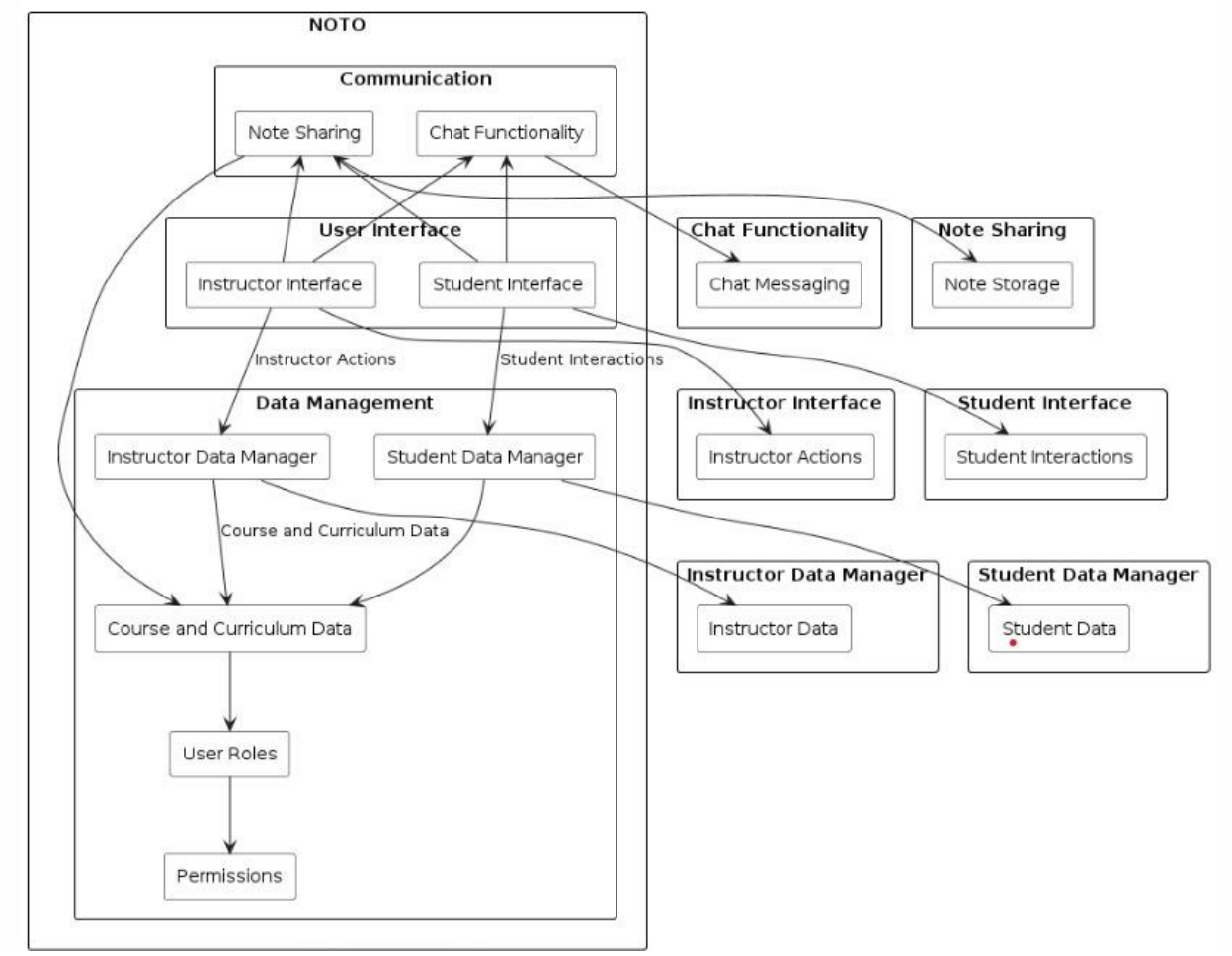
Sequence Diagram



Entity Relationship Diagram



Data Flow Diagram



Project Code Snippets

Functions

```
994  function notification(text){
995    $('.toast').addClass("active");
996    $('.text').text(text);
997  }
998
999  $('.progress').addClass("active");
1000
1001 timer1 = setTimeout(() => {
1002     $('.toast').removeClass("active");
1003 }, 5000);
1004
1005 timer2 = setTimeout(() => {
1006   $('.progress').removeClass("active");
1007 }, 5300);
1008
1009
1010 $('.close').click(() => {
1011   $('.toast').removeClass("active");
1012
1013 setTimeout(() => {
1014 $('.progress').removeClass("active");
1015 }, 300);
1016
1017   clearTimeout(timer1);
1018   clearTimeout(timer2);
1019
1020 });
1021
```

```
function sendFriendRequest(id , uiUpdateFunction) {
        console.log(id)
        payload = {
            'csrfmiddlewaretoken' : '{{csrf_token}}',
            'receiver_user_id' : id,
        }

        $.ajax({
        type: "POST",
        url: "{%url 'Friend:friend-request' %}",
        dataType: "json",
        timeout:5000,
        data : payload,

        success: function (data) {
            // console.log("SUCESS " : response)
            if(data['response'] == "friend request sent" ){

            }
            else if(data['response'] == null){
                alert(data['response'])
            }
        },
        complete: function(){
            uiUpdateFunction()
        }
        });
};
```

```css
@media (min-width: 868px) {
  .header__wrapper .cols__container {
    max-width: 1200px;
    margin: 0 auto;
    width: 90%;
    justify-content: space-between;
    display: grid;
    grid-template-columns: 1fr 2fr;
    gap: 50px;
  }
  .header__wrapper .cols__container .left__col {
    padding: 25px 0px;
  }
  .header__wrapper .cols__container .right__col nav ul {
    flex-direction: row;
    gap: 30px;
  }
  .header__wrapper .cols__container .right__col .photos {
    height: 365px;
    overflow: auto;
    padding: 0 0 30px;
  }
}

@media (min-width: 1017px) {
  .header__wrapper .cols__container .left__col {
    margin: 0;
    margin-right: auto;
  }
```

```
.header__wrapper .cols__container .left__col .img__container {
  position: absolute;
  top: -60px;
  left: 50%;
  transform: translatex(-50%);
}
.header__wrapper .cols__container .left__col .img__container img {
  width: 120px;
  height: 120px;
  object-fit: cover;
  border-radius: 50%;
  display: block;
  box-shadow: 1px 3px 12px rgba(0, 0, 0, 0.18);
}
.header__wrapper .cols__container .left__col .img__container span {
  position: absolute;
  background: #2afa6a;
  width: 16px;
  height: 16px;
  border-radius: 50%;
  bottom: 3px;
  right: 11px;
  border: 2px solid #fff;
}
.header__wrapper .cols__container .left__col h2 {
  margin-top: 60px;
  font-weight: 600;
  font-size: 22px;
  margin-bottom: 5px;
```

```css
@import url("https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap
body {
  width: 100%;
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  min-height: 100vh;
  font-family: "Poppins", sans-serif;
}

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  display: flex;
  align-items: center;
}

a {
  text-decoration: none;
}

.header__wrapper header {
  width: 100%;

  filter: blur(2px);
  -webkit-filter: blur(2px);
  min-height: calc(100px + 15vw);
}
```

```
<section class="sign-in">
  <div class="container">
      <div class="signin-content">
          <div class="signin-form">
                <form action="{%url 'login'%}" class="register-form" id="login-form" style="box-shadow: 0
                    </div>

                    <!-- Password input -->
                    <div data-mdb-input-init class="form-outline mb-4">
                        <input type="password" id="form3Example4" class="form-control" placeholder="Passwo
                    </div>

                    <div class="col">
                        <!-- Simple link -->
                        <a href="{%url 'password_reset'%}">Forgot password?</a>
                      </div>
                    </div>

                    <!-- Submit button -->
                    <button data-mdb-ripple-init type="submit" class="btn btn-primary btn-block mb-4 mt-4

                </form>
          </div>
      </div>
  </div>
</section>
{% endblock %}
```

```
<section class="sign-in">
  <div class="container">
      <div class="signin-content">
          <div class="signin-image">
              <figure><img src="{% static '/images/signin-image.jpg' %}" alt="sing up image"></figure>
              <a href="/sign_up" class="signup-image-link">Create an account</a>
          </div>

          <div class="signin-form">
              <h2 class="form-title">Log In</h2>
                <form action="{%url 'login'%}" class="register-form" id="login-form" style="box-shadow: 0 0

                    {%csrf_token%}
                    <!-- messages -->

                    {% if messages %}
                    <ul class="messages">
                        {% for message in messages %}
                        <li {% if message.tags %} class="{{ message.tags }}"{% endif %}>{{ message }}</li>
                        {% endfor %}
                    </ul>
                    {% endif %}

                    <!-- Email input -->
                    <div data-mdb-input-init class="form-outline mb-4">
                        <input type="email" id="form3Example3" class="form-control" placeholder="Email addr
                    </div>
```

```
    hide_email      =models.BooleanField(default=False)

    objects = Profile_Manager()

    USERNAME_FIELD ='email'
    REQUIRED_FIELDS =['username']

    def __str__(self):
        return self.username

    def get_profile_image_name(self):
        # selecting two substrings between indexies
        return str(self.prof_image)[str(self.prof_image).index(f'profile_images/{self.pk}/'):]

    def has_perm(self , perm , obj =None):
        return self.is_admin

    def has_module_perms(self,app_label):
        return True
```

Pixel > Pixel > User_profile > models.py > Profile_Manager > create_user

```
from django.db import models
from django.contrib.auth.models import AbstractBaseUser , BaseUserManager ,PermissionsMixin

# manager here


class Profile_Manager(BaseUserManager):

    def create_user(self,email,username,password=None):      Arshdeep Palial, 1 hour ago •
        if not email:
            raise ValueError('Email required')
        if not username:
            raise ValueError('Username required')

        user = self.model(
            email = self.normalize_email(email),
            username = username,
        )
        user.set_password(password)
        user.save(using = self.db)
        return user

    def create_superuser(self,email,username,password):

        if not email:
            raise ValueError('Email required')
        if not username:
            raise ValueError('Username required')
        if not password:
            raise ValueError('Username required')
```

```python
@login_required
def delete1(requests,pk):
        p1=Post_first.objects.filter(id=pk)
        p1.delete()
        return HttpResponseRedirect(f'{pk}/profile')

@login_required
def delete2(requests,pk):
        p2=Post_second.objects.filter(id=pk)
        p2.delete()
        return HttpResponseRedirect(f'{pk}/profile')

@login_required
def delete3(requests,pk):
        p3=Post_third.objects.filter(id=pk)
        p3.delete()
        return HttpResponseRedirect(f'{pk}/profile')

# @login_required
# def delete4(requests,pk,user):
#         p3=Post_fourth.objects.filter(id=pk,auname=user)
#         p3.delete()
#         return HttpResponseRedirect('/detail_thir')
```

Pixel > Pixel > User_profile > views.py > ...

```python

def search(requests):
    t=list()
    search = requests.GET.get('search')
    if search:
            result = Profile.objects.filter(Q(username__contains = str(search)) | Q(username__startswith =
            for res in result:
                if res.prof_image:
                    t.append(dict(
                            {
                            'username':res.username,
                            'profile_pic':res.prof_image.url,
                            }
                            ))
                else:
                    t.append(dict(
                            {
                            'username':res.username,
                            'profile_pic':None,
                            }
                            ))

            return JsonResponse({
            'payload' : t
            })
    # print(t)
    top = Profile.objects.all()
    return render(requests,'search_user.html',{'top':top})

@login_required
```

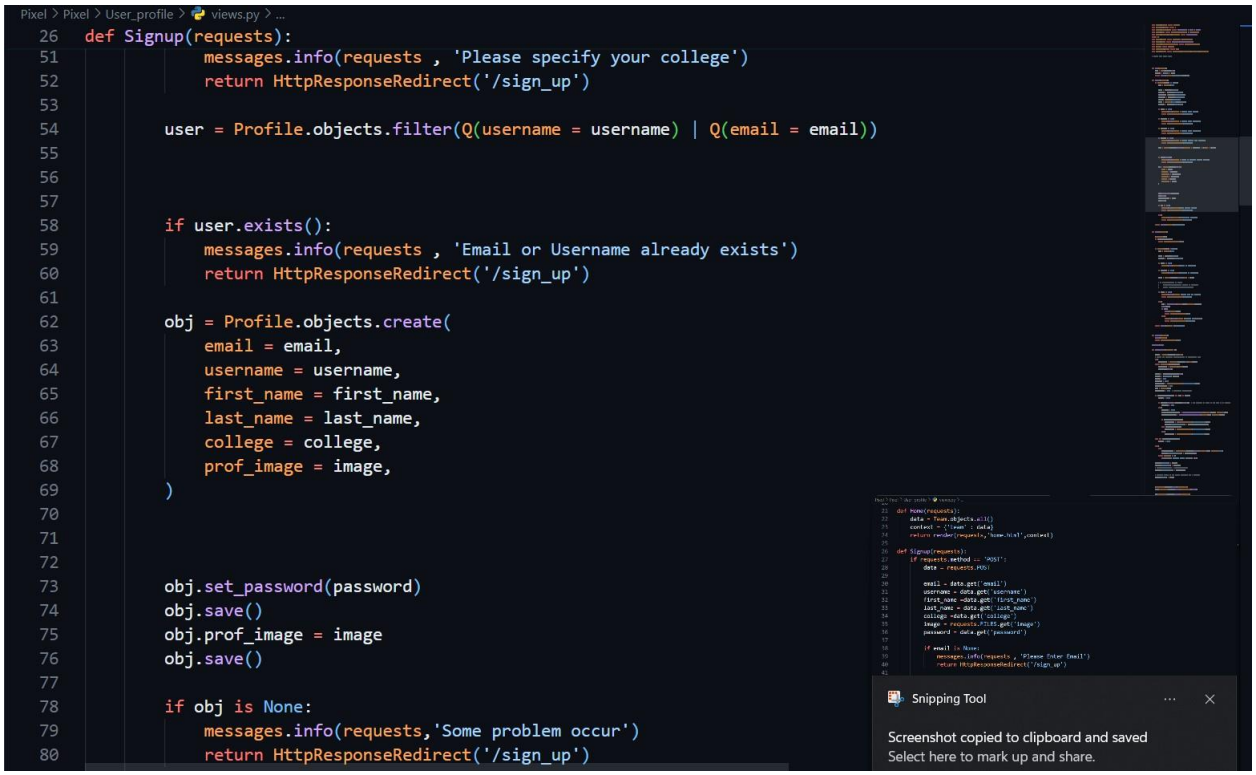
Pixel > Pixel > User_profile > views.py > ...

```python
def Log_out(request):
    logout(request)
    return HttpResponseRedirect('/')

@login_required

def Profile_page(requests, pk):

    author = Profile.objects.get(id=pk)
    # Remove the redundant initialization of friend_list here
    try:
        friend_list = FriendList.objects.get(user=author)
    except FriendList.DoesNotExist:
        friend_list = FriendList(user=author)
        friend_list.save()

    friends = friend_list.friends.all()
    context = {'friends': friends}
    is_self = True
    is_friend = False
    request_sent = FriendRequestStatus.NO_REQUEST_SENT.value
    friend_requests = None
    user = requests.user
    request_send = None  # Initialize request_send

    if user.is_authenticated and user != author:
        is_self = False

        if friends.filter(pk=user.id).exists():  # Use exists() to check if the user is in friends
            is_friend = True
```

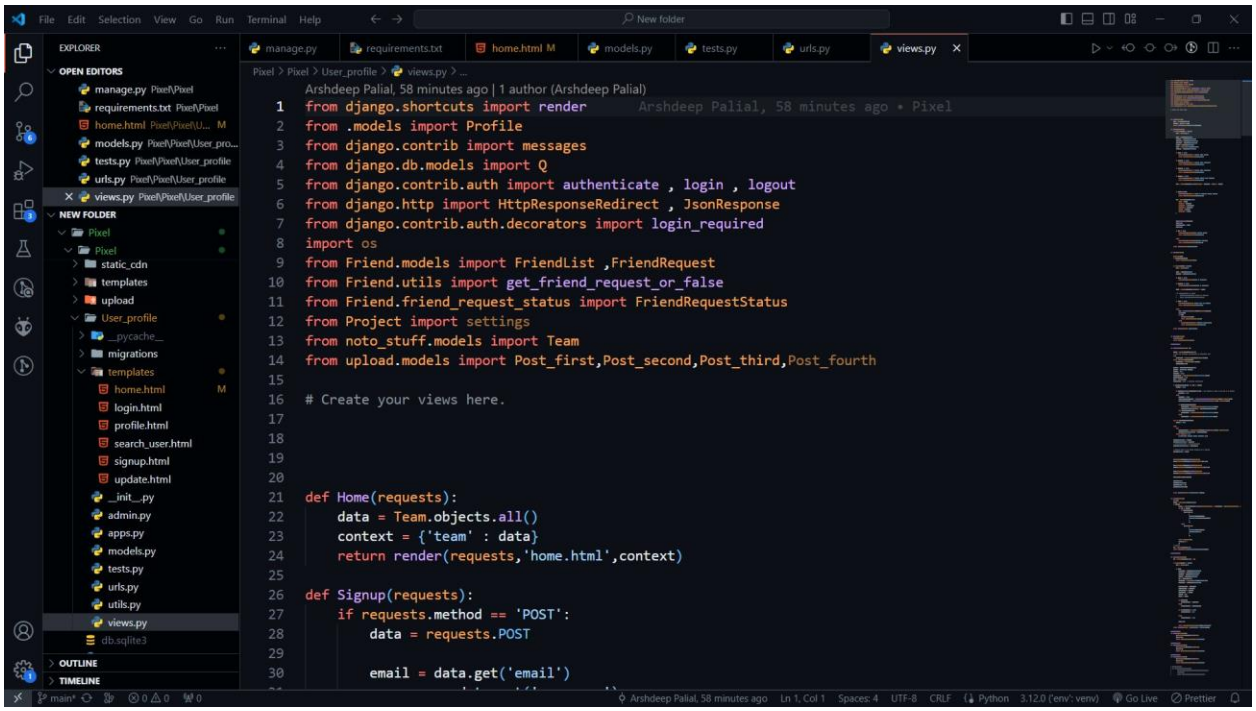
Pixel > Pixel > User_profile > views.py > ...

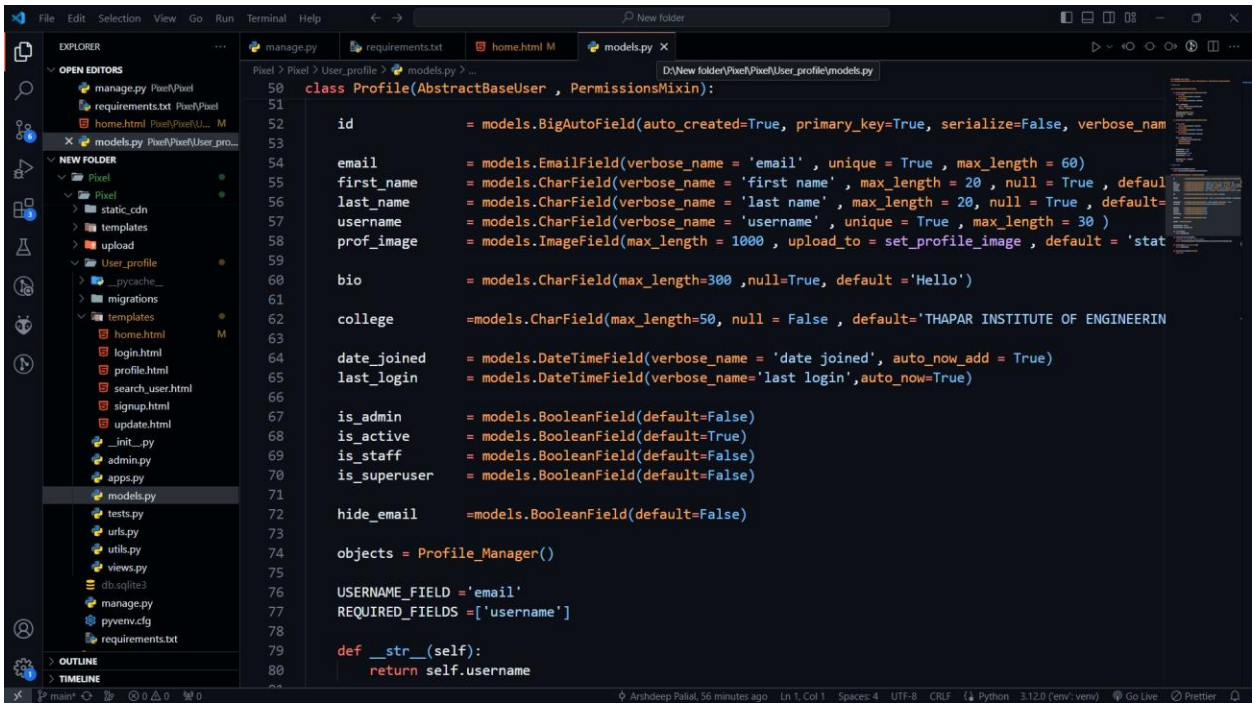
```python
def Login(requests):

    k=requests.user
    if k.is_authenticated:
        return HttpResponseRedirect('/')


    if requests.method =='POST':
        data = requests.POST

        email = data.get('email')
        password = data.get('password')

        if email is None:
            messages.info(requests,'email is required')

        if password is None:
            messages.info(requests,'password is required')

        check = Profile.objects.filter(email = email)

        # if check.is_active == False:
        #     messages.error(requests ,'email is blocked')
        #     return HttpResponseRedirect('/log_in')

        if check is None:
            messages.error(requests ,'email does not exists')
            return HttpResponseRedirect('/log_in')

```


Pixel > Pixel > User_profile > views.py > ...
def Signup(requests):
messages.info(requests , 'Please specify your college')
return HttpResponseRedirect('/sign_up')
user = Profile.objects.filter(Q(username = username) | Q(email = email))
if user.exists():
messages.info(requests , 'Email or Username already exists')
return HttpResponseRedirect('/sign_up')
obj = Profile.objects.create(
email = email,
username = username,
first_name = first_name,
last_name = last_name,
college = college,
prof_image = image,
)
obj.set_password(password)
obj.save()
obj.prof_image = image
obj.save()
if obj is None:
messages.info(requests,'Some problem occur')

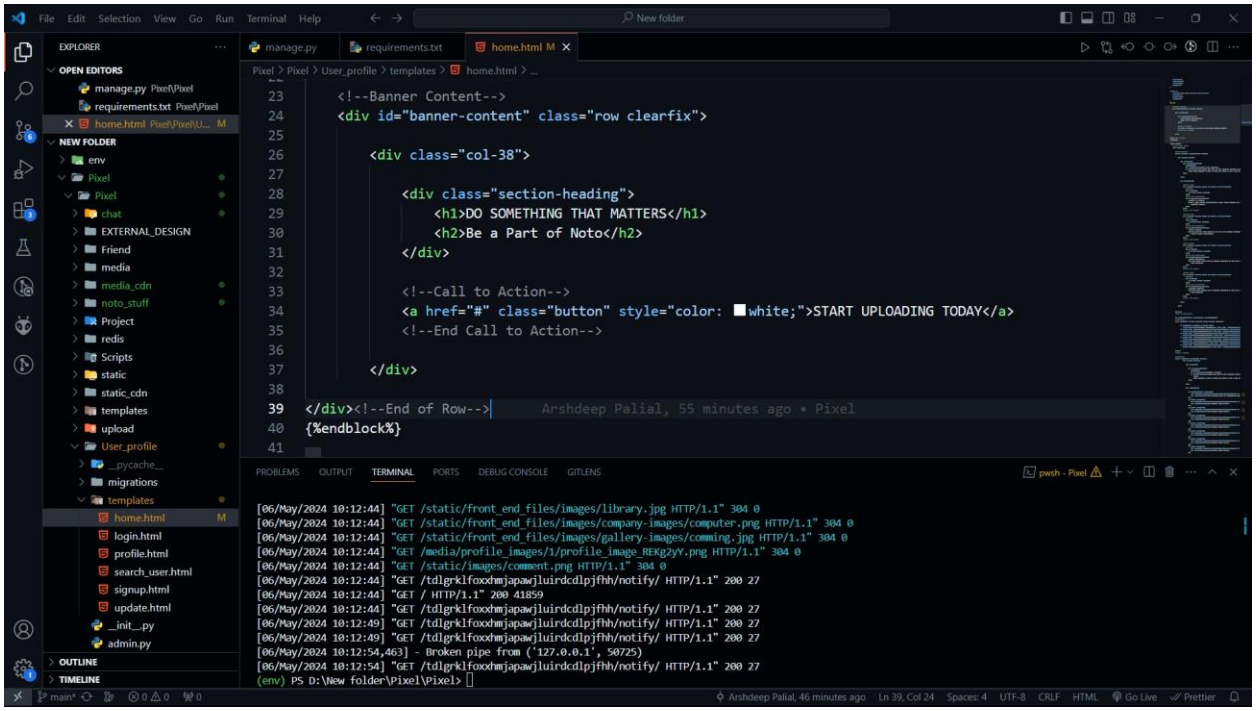



```
Pixel > Pixel > User_profile > views.py > ...
def Home(requests):
    data = Team.objects.all()
    context = {'team' : data}
    return render(requests,'home.html',context)

def Signup(requests):
    if requests.method == 'POST':
        data = requests.POST

        email = data.get('email')
        username = data.get('username')
        first_name =data.get('first_name')
        last_name = data.get('last_name')
        college =data.get('college')
        image = requests.FILES.get('image')
        password = data.get('password')

        if email is None:
            messages.info(requests , 'Please Enter Email')
            return HttpResponseRedirect('/sign_up')

        if username is None:
            messages.info(requests , 'Please Enter Username')
            return HttpResponseRedirect('/sign_up')

        if password is None:
            messages.info(requests , 'Please Enter Password')
            return HttpResponseRedirect('/sign_up')

        if college is None:
            messages.info(requests , 'Please specify your college')
```


File Edit Selection View Go Run Terminal Help
New folder
EXPLORER
OPEN EDITORS
manage.py Pixel\Pixel
requirements.txt Pixel\Pixel
home.html Pixel\Pixel\U... M
models.py Pixel\Pixel\User_pro...
tests.py Pixel\Pixel\User_profile
urls.py Pixel\Pixel\User_profile
views.py Pixel\Pixel\User_profile
NEW FOLDER
Pixel
Pixel
static_cdn
templates
upload
User_profile
__pycache__
migrations
templates
home.html M
login.html
profile.html
search_user.html
signup.html
update.html
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
utils.py
views.py
db.sqlite3
OUTLINE
TIMELINE
manage.py
requirements.txt
home.html M
models.py
tests.py
urls.py
views.py
Pixel > Pixel > User_profile > views.py > ...
Arshdeep Palial, 58 minutes ago | 1 author (Arshdeep Palial)
1 from django.shortcuts import render Arshdeep Palial, 58 minutes ago • Pixel
2 from .models import Profile
3 from django.contrib import messages
4 from django.db.models import Q
5 from django.contrib.auth import authenticate , login , logout
6 from django.http import HttpResponseRedirect , JsonResponse
7 from django.contrib.auth.decorators import login_required
8 import os
9 from Friend.models import FriendList ,FriendRequest
10 from Friend.utils import get_friend_request_or_false
11 from Friend.friend_request_status import FriendRequestStatus
12 from Project import settings
13 from noto_stuff.models import Team
14 from upload.models import Post_first,Post_second,Post_third,Post_fourth
15
16 # Create your views here.
17
18
19
20
21 def Home(requests):
22 data = Team.objects.all()
23 context = {'team' : data}
24 return render(requests,'home.html',context)
25
26 def Signup(requests):
27 if requests.method == 'POST':
28 data = requests.POST
29
30 email = data.get('email')
main*
Arshdeep Palial, 58 minutes ago Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.0 ('env': venv) Go Live Prettier



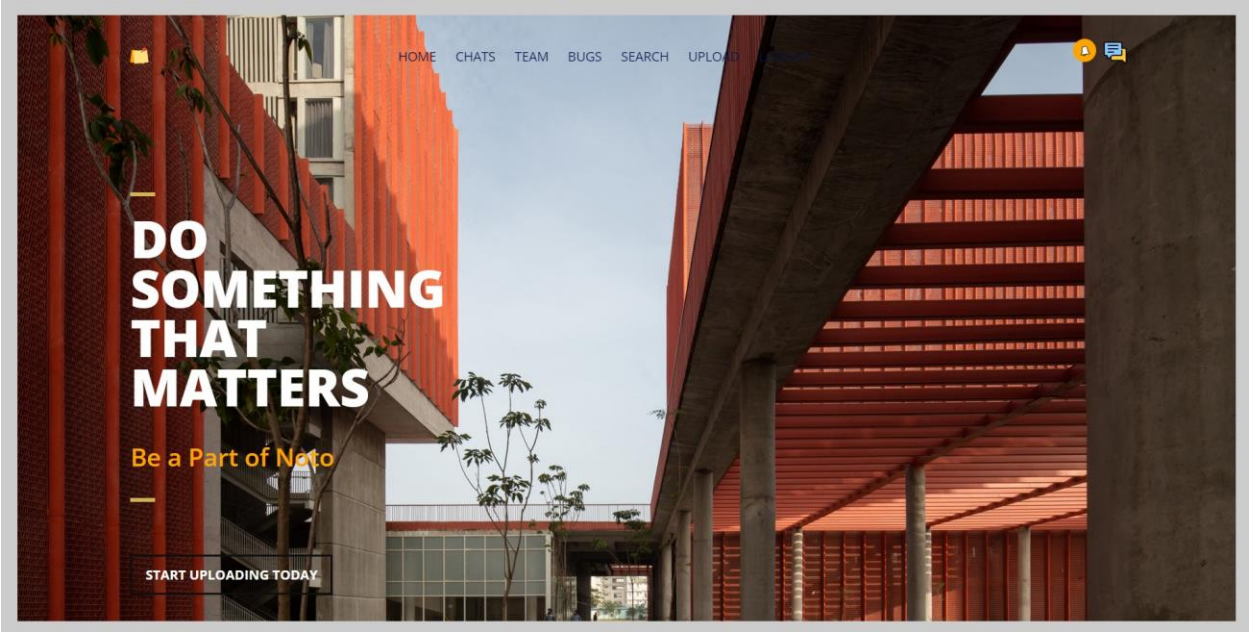
```python
class Profile(AbstractBaseUser , PermissionsMixin):

    id              = models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_nam

    email           = models.EmailField(verbose_name = 'email' , unique = True , max_length = 60)
    first_name      = models.CharField(verbose_name = 'first name' , max_length = 20 , null = True , defaul
    last_name       = models.CharField(verbose_name = 'last name' , max_length = 20, null = True , default=
    username        = models.CharField(verbose_name = 'username' , unique = True , max_length = 30 )
    prof_image      = models.ImageField(max_length = 1000 , upload_to = set_profile_image , default = 'stat

    bio             = models.CharField(max_length=300 ,null=True, default ='Hello')

    college         =models.CharField(max_length=50, null = False , default='THAPAR INSTITUTE OF ENGINEERIN

    date_joined     = models.DateTimeField(verbose_name = 'date joined', auto_now_add = True)
    last_login      = models.DateTimeField(verbose_name='last login',auto_now=True)

    is_admin        = models.BooleanField(default=False)
    is_active       = models.BooleanField(default=True)
    is_staff        = models.BooleanField(default=False)
    is_superuser    = models.BooleanField(default=False)

    hide_email      =models.BooleanField(default=False)

    objects = Profile_Manager()

    USERNAME_FIELD ='email'
    REQUIRED_FIELDS =['username']

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

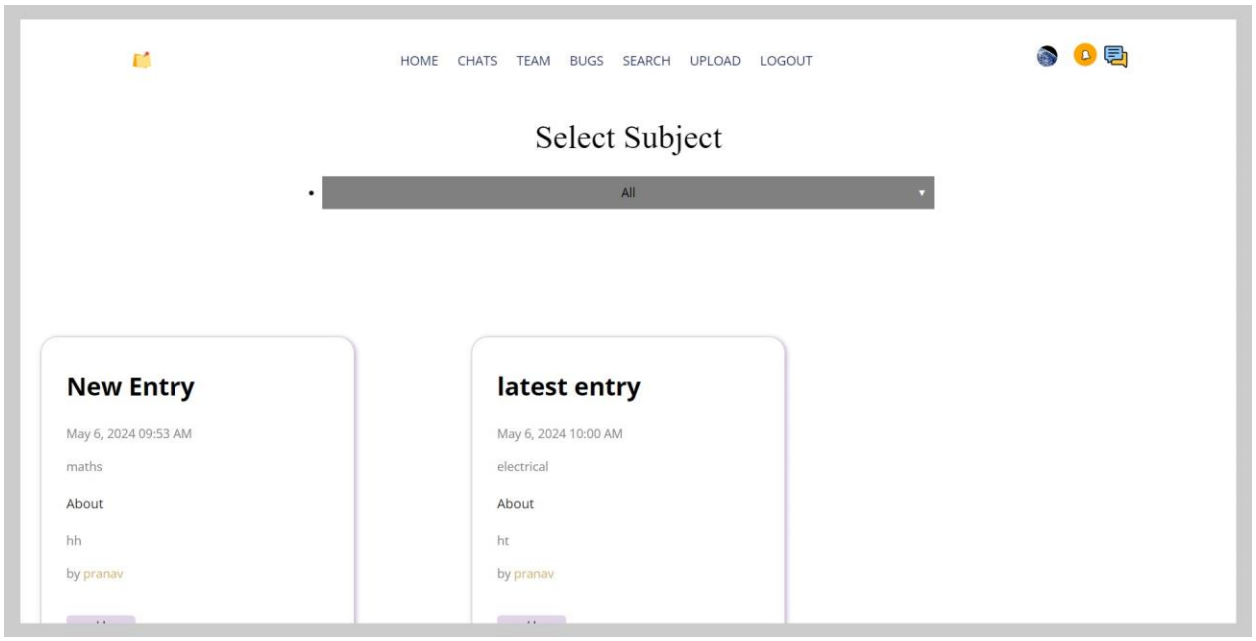


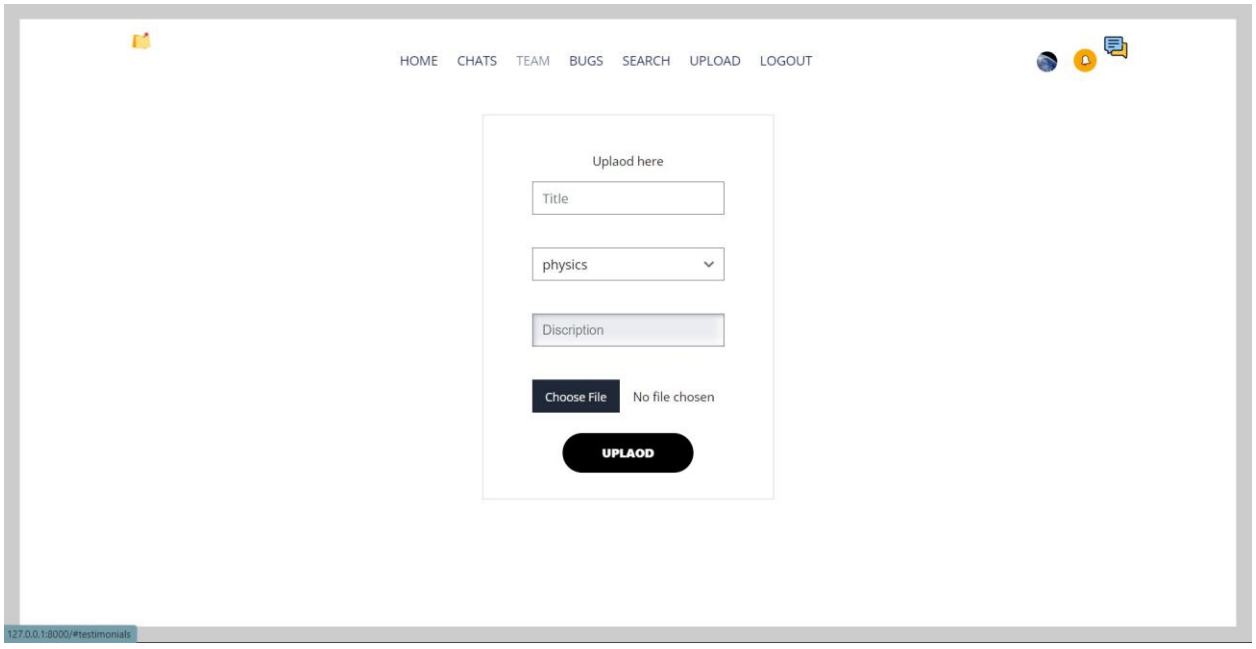
```
    <!--Banner Content-->
    <div id="banner-content" class="row clearfix">

        <div class="col-38">

            <div class="section-heading">
                <h1>DO SOMETHING THAT MATTERS</h1>
                <h2>Be a Part of Noto</h2>
            </div>

            <!--Call to Action-->
            <a href="#" class="button" style="color: white;">START UPLOADING TODAY</a>
            <!--End Call to Action-->

        </div>

</div><!--End of Row-->      Arshdeep Palial, 55 minutes ago • Pixel
{%endblock%}
```

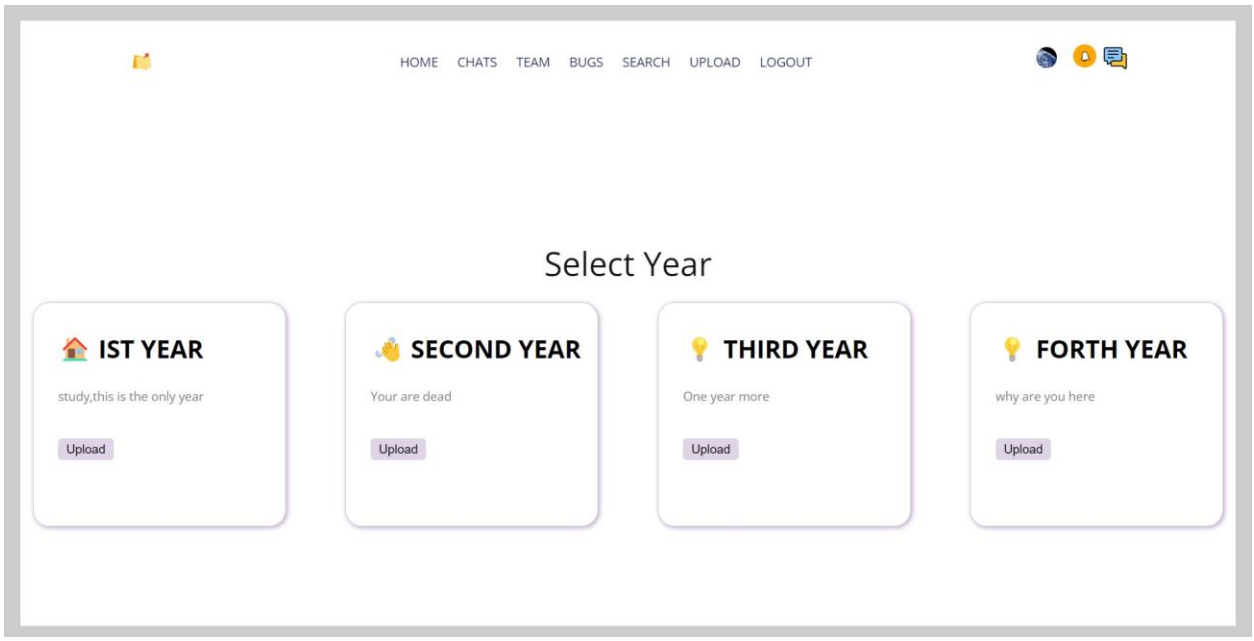
```
[06/May/2024 10:12:44] "GET /static/front_end_files/images/library.jpg HTTP/1.1" 304 0
[06/May/2024 10:12:44] "GET /static/front_end_files/images/company-images/computer.png HTTP/1.1" 304 0
[06/May/2024 10:12:44] "GET /static/front_end_files/images/gallery-images/comming.jpg HTTP/1.1" 304 0
[06/May/2024 10:12:44] "GET /media/profile_images/1/profile_image_REKg2yY.png HTTP/1.1" 304 0
[06/May/2024 10:12:44] "GET /static/images/comment.png HTTP/1.1" 304 0
[06/May/2024 10:12:44] "GET /tdlgrklfoxxhmjapawjluirdcdlpjfhh/notify/ HTTP/1.1" 200 27
[06/May/2024 10:12:44] "GET / HTTP/1.1" 200 41859
[06/May/2024 10:12:44] "GET /tdlgrklfoxxhmjapawjluirdcdlpjfhh/notify/ HTTP/1.1" 200 27
[06/May/2024 10:12:49] "GET /tdlgrklfoxxhmjapawjluirdcdlpjfhh/notify/ HTTP/1.1" 200 27
[06/May/2024 10:12:49] "GET /tdlgrklfoxxhmjapawjluirdcdlpjfhh/notify/ HTTP/1.1" 200 27
[06/May/2024 10:12:54,463] - Broken pipe from ('127.0.0.1', 50725)
[06/May/2024 10:12:54] "GET /tdlgrklfoxxhmjapawjluirdcdlpjfhh/notify/ HTTP/1.1" 200 27
(env) PS D:\New folder\Pixel\Pixel>
```

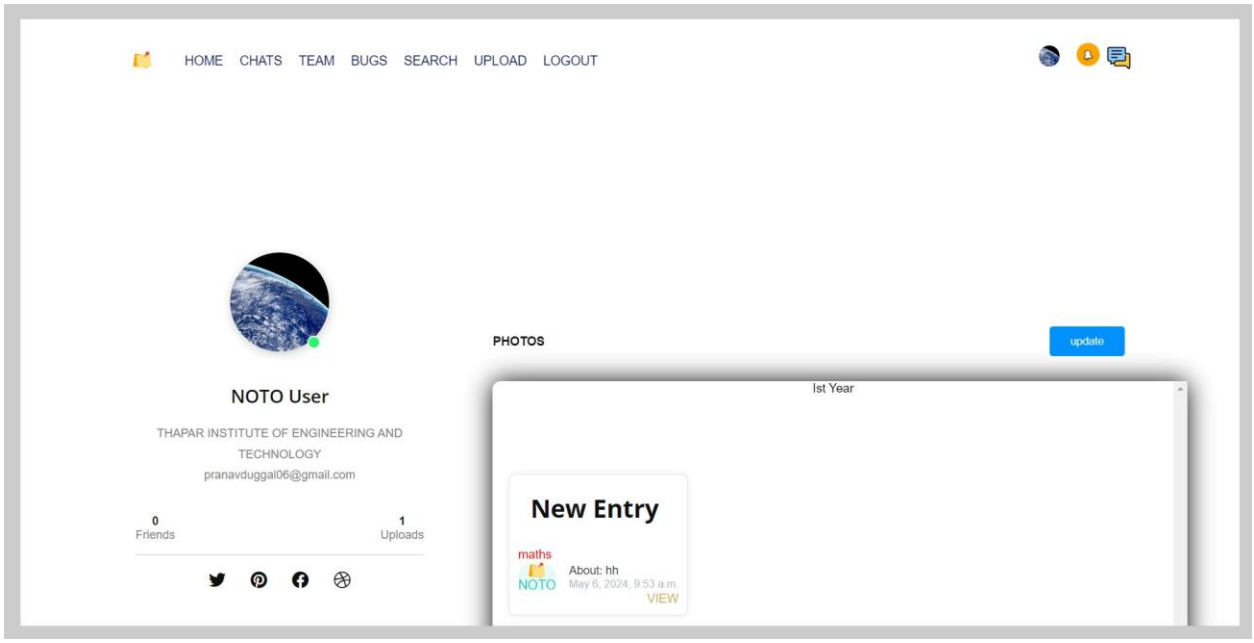
Final Project Screenshots




HOME CHATS TEAM BUGS SEARCH UPLOAD LOGOUT
Select Subject
All
New Entry
May 6, 2024 09:53 AM
maths
About
hh
by pranav
latest entry
May 6, 2024 10:00 AM
electrical
About
ht
by pranav


HOME CHATS TEAM BUGS SEARCH UPLOAD LOGOUT
Uplaod here
Title
physics
Discription
Choose File
No file chosen
UPLAOD
127.0.0.1:8000/#testimonials


HOME CHATS TEAM BUGS SEARCH UPLOAD LOGOUT
Select Year
IST YEAR
study,this is the only year
Upload
SECOND YEAR
Your are dead
Upload
THIRD YEAR
One year more
Upload
FORTH YEAR
why are you here
Upload


HOME CHATS TEAM BUGS SEARCH UPLOAD LOGOUT
NOTO User
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
pranavduggal06@gmail.com
0
Friends
1
Uploads
PHOTOS
update
Ist Year
New Entry
maths
NOTO
About: hh
May 6, 2024, 9:53 a.m.
VIEW

User Story cards 1

Title: Signing Up	Priority:	Estimate:

User story

As a user,

I want to *sign up for an account on Note using my email or social media credentials. Upon signing up, I want to set up my profile, including a profile picture and a short bio.*

so that I can *start creating and sharing notes with others.*

Acceptance criteria

Given that a user is logged into their Noto account,

when they create a new note and share it with the public,

then the note should appear in the user's profile and in the feeds of users who follow them, as well as in searches based on relevant hashtags.

User Story cards 2

Title: Creating Notes	Priority:	Estimate:

User story

As a user,

I want to *easily create new notes by drag and drop .on signing up, I want to set up my profile, including a profile picture and a short bio.*

so that I can *express myself and share my thoughts with others.*

Acceptance criteria

Given that a user is browsing notes on NoteGram,

when they like or comment on a note,

then the note's creator should receive a notification about the interaction, and the like or comment should be visible to other users who view the note.

User Story cards 3

Title: Discovering Notes	Priority:	Estimate:

User story

As a user,

I want to *have the option to add hashtags to my notes to categorize them and make them discoverable by other users interested in similar topics.*

so that I can able to share my notes publicly or privately, depending on my preference.

Acceptance criteria

Given that a user has edited a note on NoteGram,

when they save the changes,

then the updated version of the note should be displayed in the user's profile and in the feeds of users who follow them.

User Story cards 4

Title: Sharing Notes	**Priority:**	**Estimate:**

User story

As a user,

I want to *share my notes with my followers or specific individuals.*
I want to have the option to add hashtags to my notes to categorize them and make them discoverable by other users interested in similar topics.

so that I can like, comment on, or repost notes that I find interesting or inspiring..

Acceptance criteria

Given that [*user is authenticated],*

When *[notes are uploaded]*

then [*others can read it].*

User Story cards 5

Title: Privacy and Security	Priority:	Estimate:

User story

As a user,

I want to *have control over the privacy settings of my notes and profile.*

so that I can block or report other users who engage in inappropriate behavior.

Acceptance criteria

Given that [*the user is misbehaving],*

when [*inaproperiate]*

then [*block the user].*

User Story cards 6

Title: search profile	Priority:	Estimate:

User story

As a user,

I want to *profiles using username. Upon searching up, I want to see profile with searched username.*

so that I can *start searching and connect with people.*

Acceptance criteria

Given that a user has profile on Noto,

when they create a new note and share it with the public,

then the note should appear in the user's profile and in the feeds of users who follow them, as well as in s
earches based on relevant hashtags.