Alpha One and only classroom

Alpha One Labs

Synopsis

What good is a revolutionary learning platform if teachers still have to juggle **Zoom**, **docs**, and **third-party whiteboards** just to conduct a class? After working on Alpha One Labs for long, building new features everyday, I've realised that with all the current components like courses, videos, blogs, quizzes.... The one component that is missing is indeed the most important one - **A Virtual Classroom. (and what if this virtual classroom becomes a 2D user controlled experience?)**

Table of Contents

Contact info

Phone number - 91+ 9821430047

Email - ishaana612@gmail.com (personal) / ishaan24266@iiitd.ac.in (work)

Linkedin - www.linkedin.com/in/ishaan-arora-profile

Github - https://github.com/ishaan-arora-1

Portfolio Website - Portfolio

About Me

I'm **ISHAAN ARORA** , a **Computer Science student at IIIT Delhi**, with a deep passion for **web development**, which I've been exploring since my school days.I have had prior experience in competitive programming, AI excites me, and I love staying updated with the latest innovations in the field.
I began with competitive programming, using **C++**. Later, I delved into software development, mastering **HTML, CSS,** and **JavaScript**, and built projects using the **MERN** stack. I have immense familarity working with **generative AI** and other AI models . I've also learned **Python**, explored the **Django framework**.

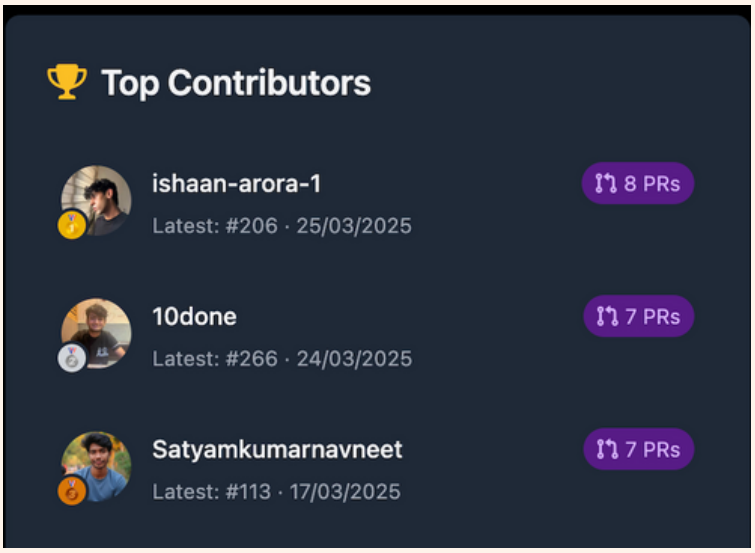
A **hackathon geek at heart**, I've won multiple **prestigious hackathons**, proving my ability to innovate under pressure. I won the first position in the Hackathon organized by **CodeDay Delhi at IIT DELHI**. I've won the yearly intra-college Hackathon **DevHaven**, participating solo against teams of 4. I have also won the **Foodoscope ForkIt Hackathon** organised by CosyLabs.

I'm a member of the **Technical Council at IIITD**, a vibrant community where students come together to build projects as a team, and also conduct many technical events.

I scored **AIR 8587** in the **JEE mains** examination where more than a million students participate every year.

My Contributions

As i write this proposal, i currently lead with the **highest number of merged pull requests.**



I have built :

- A Waiting Room for learning requests
- A Team Collaboration section where users form teams, set goals, and complete them.
- A customized Avatar addition using python avatars
- A section for Educational videos, for quick uploads while categorising them
- A section of posting Success Stories
- Made several ui improvements and added dark mode to pages where it lacked
- Resolved bugs where teachers did not have the right permissions to view their Course Materials
- A lot of gsoc members had a problem in the verification process, found the changes to be made for that too.

A full list of my merged pull requests can be found here :

Pull request - Ishaan Arora

Project Details

Part 1 : virtual classroom

When i say the word virtual classroom , i dont just mean a virtual meeting platform, i mean a whole new part of alpha one, which not only uses the current avaiable features, but brings out a whole new potential by a new **2D Classroom experience.**

Image a classroom where you can control where you want to go, just like an actual classroom. You can walk around, sit on your preferred seat

And not just that:

- the teacher is visible and writes on the whiteboard just as usual
- you can raise hands, and talk on permissions of the teacher
- the teacher has full control over the classroom
- you can have your customised avatars to show your creativity
- you can have breakout rooms with your classmates
- you can take quizzes, compete with other students and work hard for the first position
- you can work on assignments, alone or as a team.

You can be in a classroom while sitting at your homes.

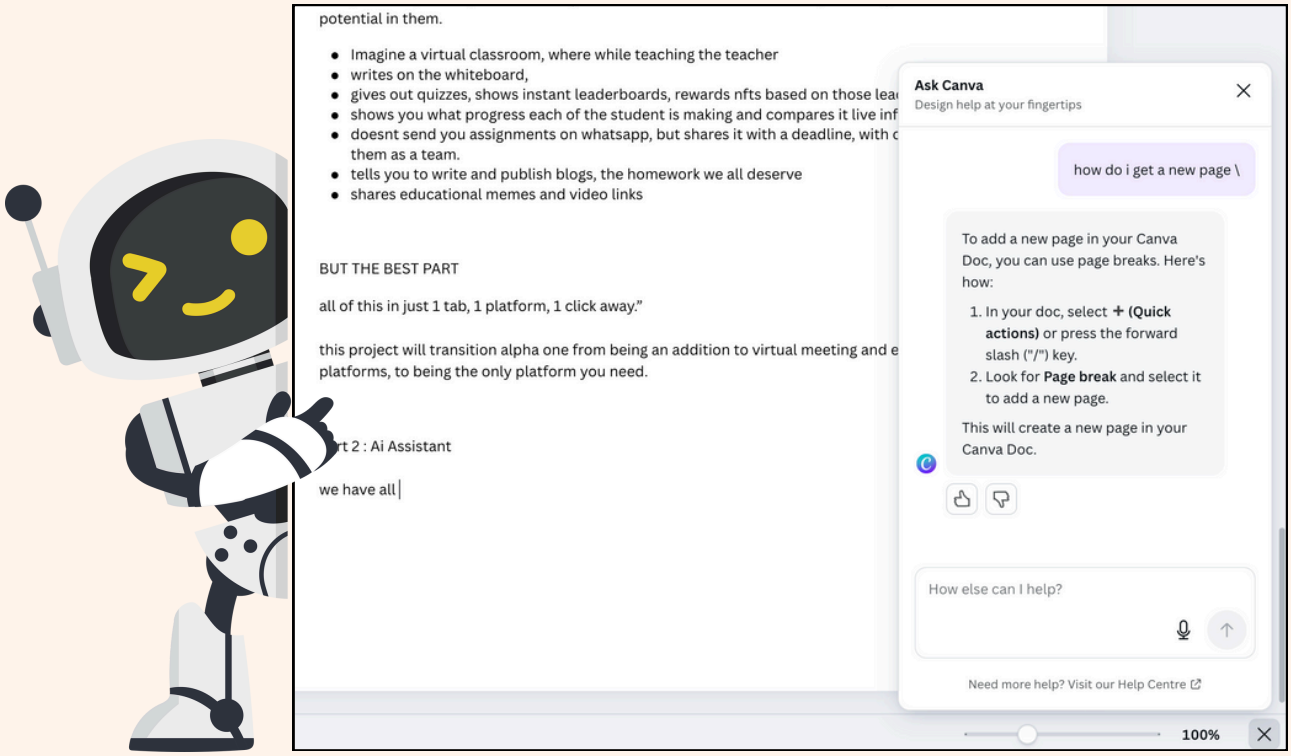
BUT THE BEST PART

All of this in just 1 tab, 1 platform, 1 click away.”

This project will transition alpha one from being an addition to virtual meeting and education platforms, to being the only platform you need.

Part 2 : Ai Assistant

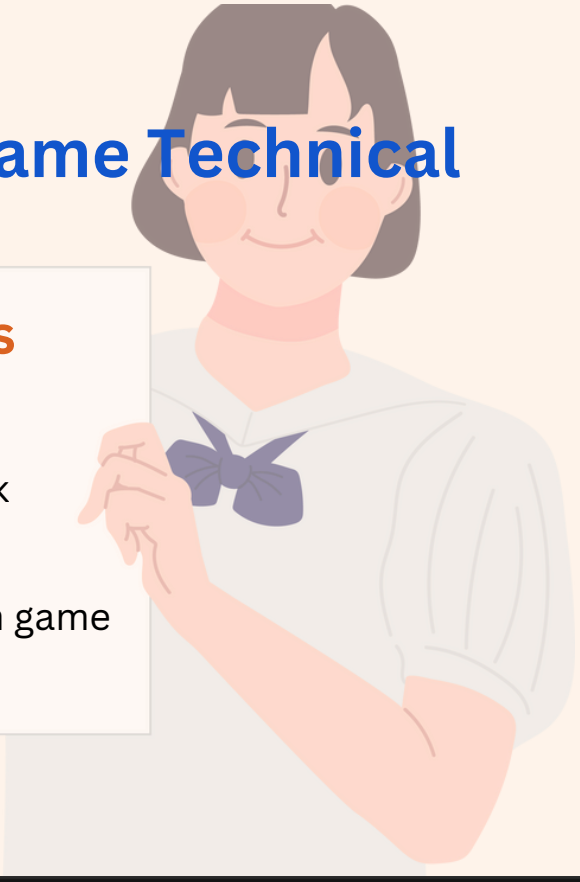
Currently as I write this proposal, I needed help regarding a feature of canva
My first instinct? look for an ai chatbot
and once i found it, my query was instantly resolved



This is a component alpha one needs, especially after the implementation of the virtual classroom, to help with any guidance the user needs.

This ai assistant would be available on all pages, in the form of a small icon, followed by a pop-up.

2D Classroom Game Technical Documentation



1. Core Technologies

Backend Stack

- **Pygame**: Game framework

Hosting

- **Pybag**: Hosting classroom game

2. File Structure

```
GAME-2/
|— code/                        # Contains all Python scripts
|   |— __pycache__/             # Compiled Python files for performance
|   |— groups.py                # Handles game groups (e.g., sprite groups)
|   |— main.py                  # Main game loop and execution entry point
|   |— player.py                # Player movement and logic
|   |— settings.py              # Game settings (e.g., resolution, FPS)
|   |— sprites.py               # Handles sprite management
|— data/                        # Stores game data files
|   |— maps/                    # Contains Tiled map files
|   |   |— world6.tmx           # Tile map file
|   |— tileset/                 # Tilesets for the game
|   |   |— tileset-classroom.png # Classroom tileset image
|   |— images/player/           # Player sprite animations
|       |— down/                # Sprites for moving down
|       |— left/                # Sprites for moving left
|       |— right/               # Sprites for moving right
|       |— up/                  # Sprites for moving up
|— myenv/                       # Virtual environment folder
|   |— bin/                     # Executable files for virtual environment
|   |— include/                 # C header files for virtual environment
|   |— lib/                     # Installed libraries for the virtual environment
|— pyvenv.cfg                   # Configuration file for the virtual environment
```

3. Why Pybag?

Pybag is the easiest and lightest option for hosting lightweight to medium sized games. Since the classroom will be a single map, with interactive links at places, Pybag should be able to handle the game easily.

4. Creating a Camera

groups.py

code > groups.py > AllSprites > draw

```python
from settings import *

class AllSprites(pygame.sprite.Group):
    def __init__(self):
        super().__init__()
        self.display_surface = pygame.display.get_surface()
        self.offset = pygame.Vector2()

    def draw(self, target_pos):
        self.offset.x = WINDOW_WIDTH // 2 - target_pos[0]
        self.offset.y = WINDOW_HEIGHT // 2 - target_pos[1]

        for sprite in self:
            # Apply offset to sprite's rect
            offset_pos = sprite.rect.topleft + self.offset
            self.display_surface.blit(sprite.image, offset_pos)
```

4. Rendering Controllable Character

player.py

code > player.py > Player > input

```python
from settings import *

class Player(pygame.sprite.Sprite):
    def __init__(self, pos,groups,collision_sprites):
        super().__init__(groups)
        self.load_images()
        self.state, self.frame_index = 'down', 0
        self.image = pygame.image.load(join("..","images","player",'down','0.png')).convert_alpha()
        self.rect = self.image.get_rect(center = pos)
        self.hitbox_rect = self.rect.inflate(-50,-20)

        self.direction = pygame.Vector2()
        self.speed = 500
        self.collision_sprites = collision_sprites

    def load_images(self):
        self.frames = {'left':[],'right':[],'up':[],'down':[]}

        for state in self.frames.keys():
            for folder_path, sub_folders, file_names in walk(join('..','images','player',state)):
                if file_names:
                    for file_name in sorted(file_names,key=lambda name: int(name.split('.')[0])):
                        full_path = join(folder_path,file_name)
                        surf = pygame.image.load(full_path).convert_alpha()
                        self.frames[state].append(surf)


    def input(self):
        keys = pygame.key.get_pressed()
        input_x = int(keys[pygame.K_RIGHT]) - int(keys[pygame.K_LEFT])
        input_y = int(keys[pygame.K_DOWN]) - int(keys[pygame.K_UP])
        self.direction = pygame.Vector2(input_x, input_y)
        if self.direction.length() > 0:  # Fix: Check length before normalizing
            self.direction = self.direction.normalize()
```

```python

    def move(self,dt):
        self.hitbox_rect.x += self.direction.x * self.speed * dt
        self.collision('horizontal')
        self.hitbox_rect.y += self.direction.y * self.speed * dt
        self.collision('vertical')
        self.rect.center = self.hitbox_rect.center

    def collision(self,direction):
        for sprite in self.collision_sprites:
            if sprite.rect.colliderect(self.hitbox_rect):
                if direction == 'horizontal':
                    if self.direction.x > 0:
                        self.hitbox_rect.right = sprite.rect.left
                    if self.direction.x < 0:
                        self.hitbox_rect.left = sprite.rect.right
                else:
                    if self.direction.y > 0:
                        self.hitbox_rect.bottom = sprite.rect.top
                    if self.direction.y < 0:
                        self.hitbox_rect.top = sprite.rect.bottom

    def animate(self,dt):

        if self.direction.x !=0:
            self.state = 'right' if self.direction.x > 0 else 'left'
        if self.direction.y !=0:
            self.state = 'down' if self.direction.y > 0 else 'up'

        self.frame_index = self.frame_index + 5 * dt if self.direction else 0
        self.image = self.frames[self.state][int(self.frame_index) % len(self.frames[self.state])]

    def update(self,dt):
        self.input()
        self.move(dt)
        self.animate(dt)
```

5. Settings

settings.py

code > settings.py > ...

```python
import pygame
from os.path import join
from os import walk,path

WINDOW_WIDTH, WINDOW_HEIGHT = 1280, 720
TITLE_SIZE = 64
```

6. Rendering Objects

sprites.py

code > sprites.py > ...

```python
from settings import *

class Sprite(pygame.sprite.Sprite):
    def __init__(self, pos, surf,groups):
        super().__init__(groups)
        self.image = surf
        self.rect = self.image.get_rect(topleft = pos)

class CollisionSprite(pygame.sprite.Sprite):
    def __init__(self, pos, surf,groups):
        super().__init__(groups)
        self.image = surf
        self.rect = self.image.get_rect(topleft = pos)

```

7. Rendering map using above classes

main.py

code > main.py > Game > __init__

```python
from settings import *
from player import Player
from sprites import *
from pytmx.util_pygame import load_pygame
from random import randint
from groups import AllSprites

class Game:
    def __init__(self):
        #setup
        pygame.init()
        self.display_surface = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
        pygame.display.set_caption("student")
        pygame.event.get()
        self.clock = pygame.time.Clock()
        self.running = True

        #group
        self.all_sprites = AllSprites()
        self.collision_sprites = pygame.sprite.Group()

        self.setup()

        #player


    def setup(self):
        map = load_pygame(join('..','data','maps','tile-6.tmx'))

        for x,y,image in map.get_layer_by_name('Tile Layer 1').tiles():
            Sprite((x*TITLE_SIZE,y*TITLE_SIZE),image,self.all_sprites)
        for obj in map.get_layer_by_name('obj'):
            CollisionSprite((obj.x,obj.y),obj.image,(self.collision_sprites,self.all_sprites))
        for obj in map.get_layer_by_name('col'):
            CollisionSprite((obj.x,obj.y),pygame.Surface((obj.width,obj.height)),self.collision_sprites)
        for obj in map.get_layer_by_name('ent'):
            if obj.name == 'Player':
                self.player = Player((obj.x,obj.y),self.all_sprites,self.collision_sprites)

```

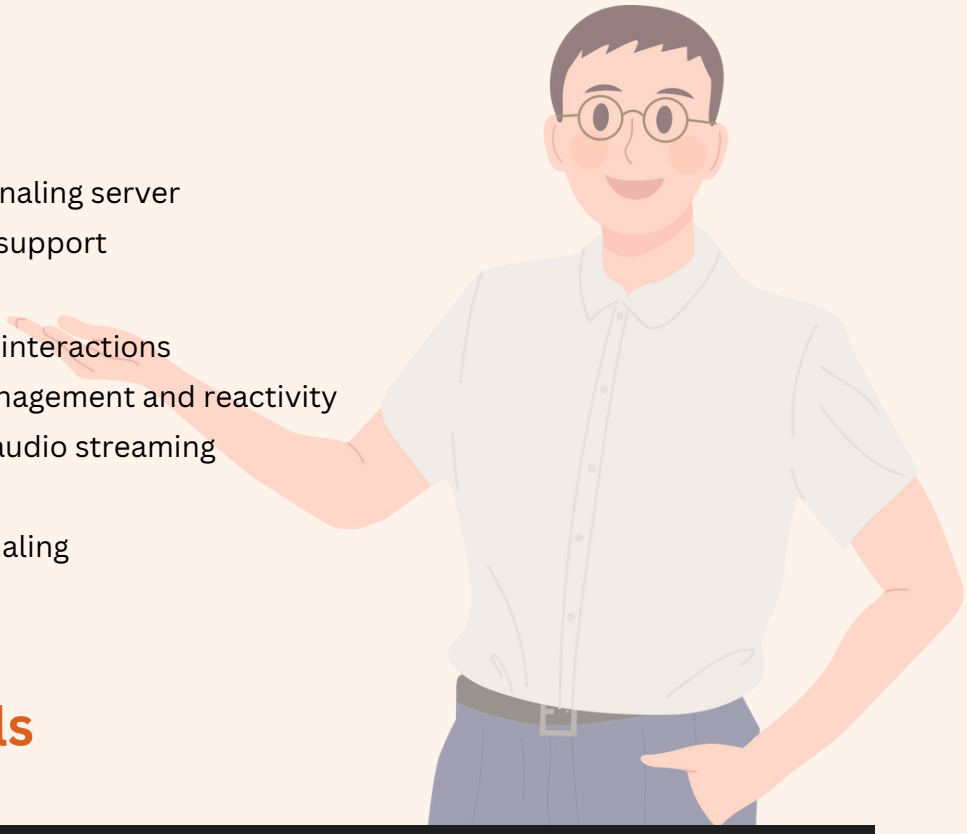
```python
    def run(self):
        while self.running:
            #dt
            dt = self.clock.tick()/1000
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    self.running = False

            #update
            self.all_sprites.update(dt)

            #draw
            self.display_surface.fill(('black'))
            self.all_sprites.draw(self.player.rect.center)
            pygame.display.update()

        pygame.quit()

if __name__ == "__main__":
    game = Game()
    game.run()
```

Concept Sketches

i have built this lo-fi version **using solely python, and i have built the map using tiled.**
the user has complete control over the character and can interact with elements.
the actual map of the classroom that will be implemented will have a much bigger and detailed map, with multiple classrooms, corridors and more elements.

Virtual Classroom Technical Documentation

1. Core Technologies



Backend Stack

- **Django**: Main web framework
- **Python-SocketIO**: WebRTC signaling server
- **Django Channels**: WebSocket support

Frontend Stack

- **HTMX**: Real-time updates and interactions
- **Alpine.js**: Client-side state management and reactivity
- **WebRTC**: Peer-to-peer video/audio streaming
- **Tailwind CSS**: UI styling
- **Socket.IO Client**: WebRTC signaling

2. DataBase Models

```python
class VirtualClassroom(models.Model):
    title = models.CharField(max_length=200)
    course = models.ForeignKey(Course)
    teacher = models.ForeignKey(User)
    grid_columns = models.IntegerField(default=4)
    max_participants = models.IntegerField(default=25)
    is_active = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)

class VirtualSeat(models.Model):
    classroom = models.ForeignKey(VirtualClassroom)
    student = models.ForeignKey(User, null=True)
    row = models.IntegerField()
    column = models.IntegerField()
    is_occupied = models.BooleanField(default=False)

class VirtualHand(models.Model):
    classroom = models.ForeignKey(VirtualClassroom)
    student = models.ForeignKey(User)
    raised_at = models.DateTimeField(auto_now_add=True)
    is_active = models.BooleanField(default=True)

class UpdateRound(models.Model):
    classroom = models.ForeignKey(VirtualClassroom)
    current_speaker = models.ForeignKey(User)
    duration = models.IntegerField()  # seconds
    started_at = models.DateTimeField(auto_now_add=True)

class ScreenShare(models.Model):
    classroom = models.ForeignKey(VirtualClassroom)
    user = models.ForeignKey(User)
    type = models.CharField(choices=['screenshot', 'link'])
    content = models.TextField()
    title = models.CharField(max_length=200)
    shared_at = models.DateTimeField(auto_now_add=True)
```

3. WebRTC integration

```python
import socketio
import eventlet

sio = socketio.Server(cors_allowed_origins='*')
app = socketio.WSGIApp(sio)

@sio.event
def connect(sid, environ):
    print(f'Client connected: {sid}')

@sio.event
def join_room(sid, data):
    room = data['room']
    if room not in rooms:
        rooms[room] = set()
    rooms[room].add(sid)

    # Notify others
    for participant in rooms[room]:
        if participant != sid:
            sio.emit('peer_joined', {'peerId': sid}, room=participant)

@sio.event
def relay_signal(sid, data):
    sio.emit('signal', {
        'peerId': sid,
        'signal': data['signal']
    }, room=data['target'])
```

4. Real Time updates : Using hx-trigger for periodic updates (every 2s)

```html
<!-- Classroom State Updates -->
<div id="classroom-state"
     hx-get="/classroom/{{ classroom.id }}/state/"
     hx-trigger="every 2s, refresh from:body"
     hx-target="#classroom-state"
     hx-swap="innerHTML">
```

```javascript
// Alpine.js Store
Alpine.store('classroom', {
    selectedSeat: null,
    handRaised: false,
    activeRound: null,
    timeRemaining: 0,
    currentSpeaker: null
});
```

5. Hand raise

```
// Toggle Hand
async toggleHand() {
    const url = `/classroom/${this.classroomId}/toggle-hand/`;
    const response = await fetch(url, {
        method: 'POST'
    });
    this.handRaised = !this.handRaised;
}
```

6. Screen Sharing

```
async startScreenShare() {
    const screenStream = await navigator.mediaDevices.getDisplayMedia();
    const videoTrack = screenStream.getVideoTracks()[0];

    // Replace video track in all peer connections
    Object.values(this.connections).forEach(pc => {
        const sender = pc.getSenders()
            .find(s => s.track.kind === 'video');
        sender.replaceTrack(videoTrack);
    });
}
```

7. Video Controls

```
// Video/Audio Controls
toggleVideo() {
    this.videoEnabled = !this.videoEnabled;
    this.localStream.getVideoTracks()
        .forEach(track => track.enabled = this.videoEnabled);
},

toggleAudio() {
    this.audioEnabled = !this.audioEnabled;
    this.localStream.getAudioTracks()
        .forEach(track => track.enabled = this.audioEnabled);
}
```

8.Performance Optimizations

A. Video Optimization

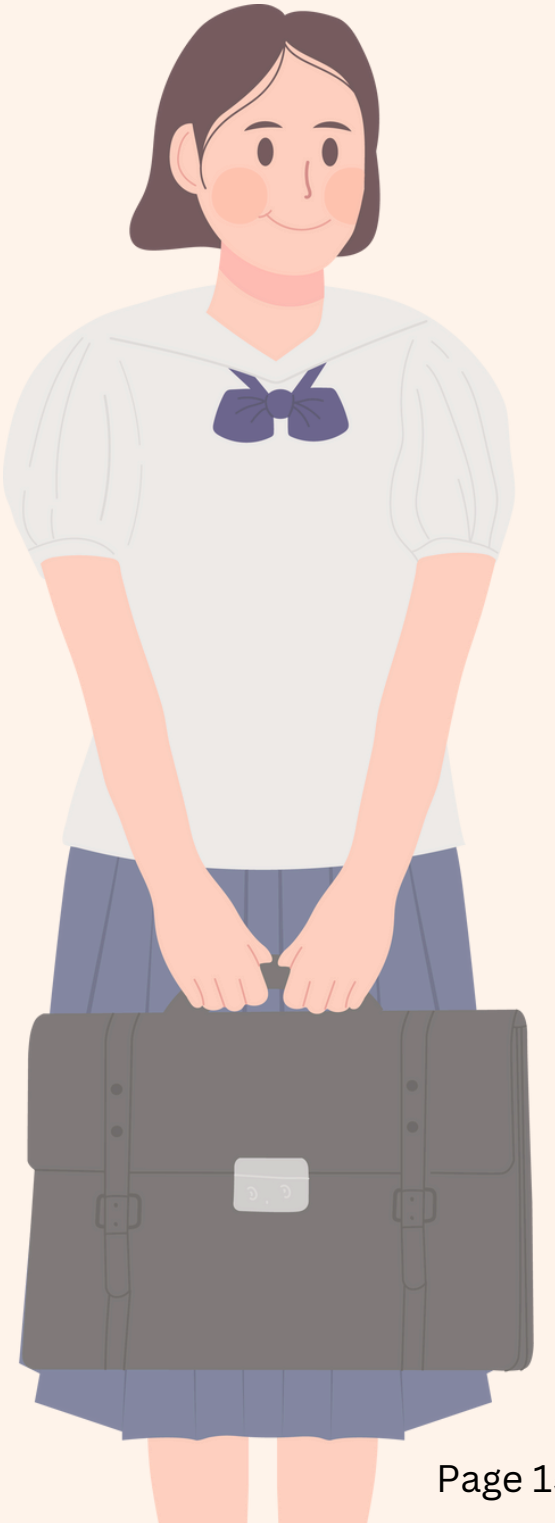
- Resolution capping (640x480)
- Frame rate limiting (24fps)
- Selective video streaming

B. Network Optimization

- P2P connections via WebRTC
- Efficient signaling protocol
- Connection state monitoring

C. UI Performance

- Virtual scrolling for large grids
- Lazy loading of video elements
- Efficient DOM updates via HTMX



9.Security Measures

A. Access Control

- User authentication required
- Room-based permissions
- Teacher/Student role separation
- Media permission checks

B. WebRTC Security

- Secure signaling server
- STUN server validation
- Connection monitoring

10.Resource Usage

Client-side:

- Memory: ~50-100MB per active connection
- CPU: 5-15% per video stream

AI assistant Technical Documentation

Powered by **Open ai API** - can be modified depending on the api used for other Ai features being built.

Backend:

- Django
- OpenAI API (gpt-3.5-turbo)
- Django caching framework

ChatBot View

```python
import openai
import hashlib
from django.http import HttpResponse, JsonResponse
from django.views.decorators.http import require_POST
from django.core.cache import cache
from django.conf import settings
```

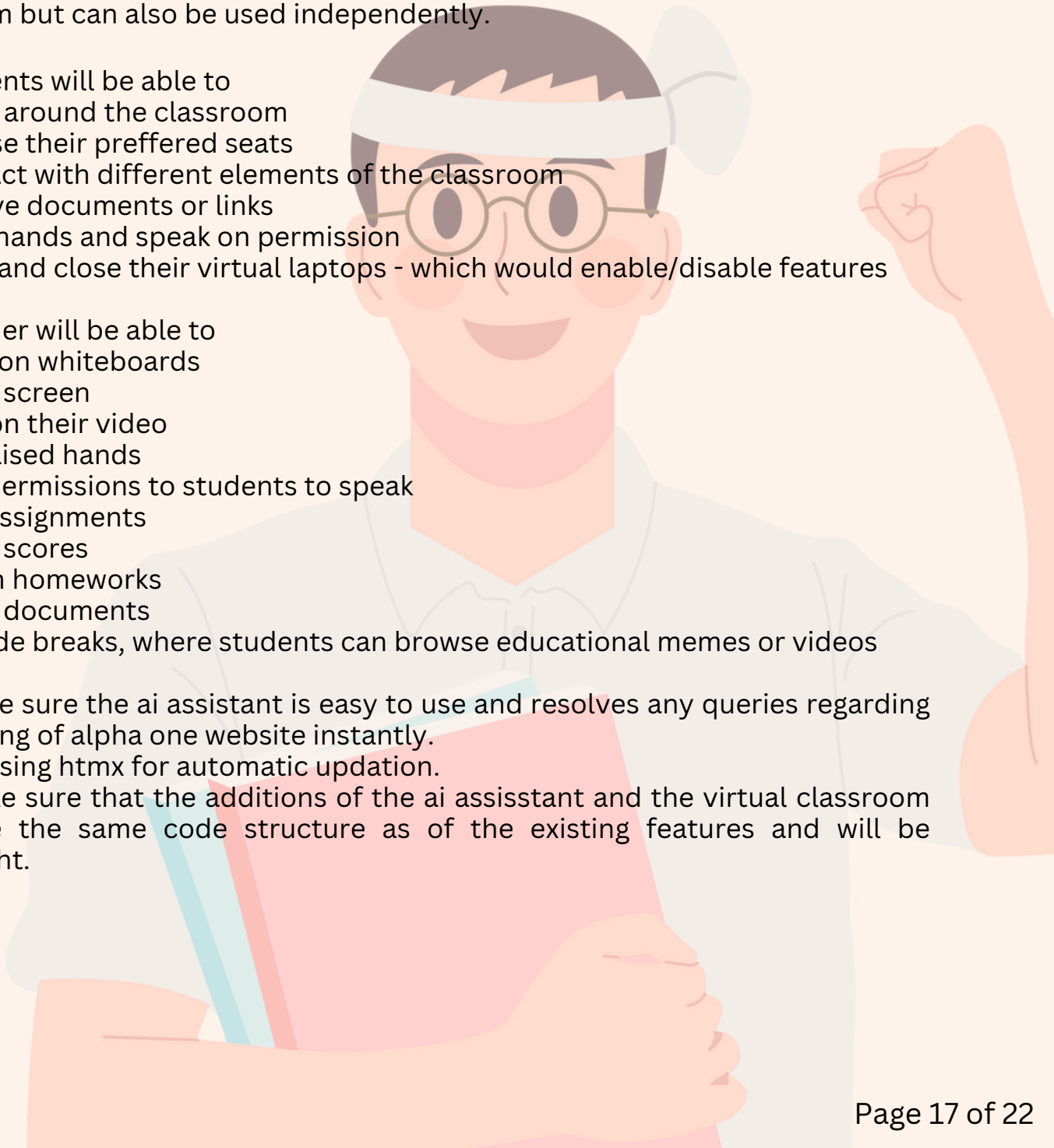
```python
@require_POST
def chatbot_query(request):
    """
    Process chatbot queries using OpenAI's API with caching.
    """
    query = request.POST.get('query', '').strip()
    if not query:
        return JsonResponse({'error': 'No query provided'}, status=400)

    # Generate cache key from query
    cache_key = f"chatbot_{hashlib.md5(query.encode()).hexdigest()}"
    cached_response = cache.get(cache_key)

    if cached_response:
        return HttpResponse(format_response(cached_response))

    try:
        # Configure OpenAI API
        openai.api_key = settings.OPENAI_API_KEY

        # Call the API
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[
                {"role": "system", "content": "You are a helpful assistant for the AlphaOne Education website.
                {"role": "user", "content": query}
            ],
            temperature=0.7,
            max_tokens=150
        )

        answer = response.choices[0].message.content.strip()

        # Cache response for 1 hour
        cache.set(cache_key, answer, 3600)

        return HttpResponse(format_response(answer))

    except Exception as e:
        return HttpResponse(format_response(f"Error: {str(e)}", is_error=True))

def format_response(text, is_error=False):
    """Format the response as HTML for insertion via HTMX"""
    css_class = "error-message" if is_error else "bot-message"
    return f' <div class="{css_class}"><p>{text}</p></div> '
```

Project Implementation

During the GSOC 25' period, i will deliver these features and functionalities :

Minimal deliverables

i will implement a **2D controllable game** with a **student character** in a school with **interactive elements** like a whiteboard, assignments table, breakout room, quizzes and more features that will be integrated along the way.
I will implement a unique but at the same time user-friendly UI/UX to ensure a better user experience for all types of users.
i will integrate all the existing features of alpha one into the virtual classroom and will also build new features which not only are a part of the virtual classroom but can also be used independently.



the students will be able to

- move around the classroom
- choose their preffered seats
- interact with different elements of the classroom
- receive documents or links
- raise hands and speak on permission
- open and close their virtual laptops - which would enable/disable features

the teacher will be able to

- write on whiteboards
- share screen
- turn on their video
- see raised hands
- give permissions to students to speak
- give assignments
- share scores
- assign homeworks
- share documents
- provide breaks, where students can browse educational memes or videos

i will make sure the ai assistant is easy to use and resolves any queries regarding the working of alpha one website instantly.
i will be using htmx for automatic updation.
i will make sure that the additions of the ai assisstant and the virtual classroom will have the same code structure as of the existing features and will be lightweight.

Additional deliverables

I'll explore the possibility of allowing video for all participants
I will try to integrate the AI assistant with any other AI component in Alpha One, to maintain consistency

Detailed Timeline

Time line	Deliverables
Community Bonding Period	
May 8 - June 1	• Familiarize with all progress made on the project by this point. Discuss with mentors and other developers to determine the next steps and prioritize tasks. • Join communication channels to engage with team members, establish rapport, and review documentation and code for completed portions. Identify necessary resources for reference before commencing new feature development. • Discussions with mentors about their preferred communication medium for project updates.
Coding official begins	
June 2 - June 30	• Build a classroom map consisting of multiple areas and interactive elements. • Build the first version of the virtual classroom, with video streaming, screen sharing, and integration with other features.
July 1 - July 14	• Refine the virtual classroom for mid term evaluation. • Do testing of the Virtual classroom • Alongside discussing the ai assistant approach with mentors, and coordinating with other ai driven projects for consistency.

Midterm Evaluation	
July 15 - July 31	• Building the AI assistant UI and optimising its UI, to blend with all the pages of the website. • Alongside making advancements on the working of the AI assistant.
August 1 - August 15	• Feeding AI assistant to store all data regarding the working of the website. • Optimising it for instant and valuable responses • Work on any issues at Alpha One.
August 15 - September 1	• Work on any issues at Alpha One. • Help with things outside of this project. • Do testing of the AI assistant for all possible scenarios. • Refinement of the final UI of features. • Receive confirmation from mentors. • Deploy the website.
Final Evaluation	

This timeline is flexible and i would be happy to help with other things at Alpha One anytime i am ahead of this schedule.

I would also love to collaborate with other GSOC participants and build other new features at Alpha One

Plan for Communication with Mentors

Throughout the project, I will uphold regular communication with mentors and other expert developers working on the project to ensure alignment with project goals. I plan to schedule a fixed time every week to update progress from the previous week, discuss goals for the upcoming week, and seek assistance if necessary. Although I am flexible and can work during nighttime hours as well, typically, I will be available for calls or messages from 12 AM IST to 12 PM IST. I'll stay connected via slack and discord.

Motivation for this project

I've always hated how most virtual classrooms feel transactional and lifeless—like glorified PDF viewers with a chatbox. During COVID, I struggled through clunky tools that made learning feel isolating: laggy video calls, disorganized resources, and zero ways to actually collaborate. It felt like we were all just staring at screens, not learning together. When I discovered Alpha One Labs, though, it clicked. Here was a platform that didn't just accept the status quo—it challenged it. Their interactive simulations and community-driven tools proved that remote learning could be dynamic, even joyful. Suddenly, I saw a way to channel my frustration into something meaningful. I don't just want to build another Zoom clone; I want to reinvent virtual classrooms entirely. With Alpha One's open-source foundation, I can create spaces where students can feel like they are in an actual classroom, and where the "aha!" moments of discovery aren't lost to technical glitches. This project isn't just code to me—it's fixing what effected my own education.

Why I'm a strong candidate

With 3+ years of expertise in HTML, CSS, JavaScript, Python, and Django, I've delivered impactful solutions for Alpha One Labs, including optimizing user workflows through real-time collaboration tools, resolving critical backend permissions, and enhancing community engagement via intuitive features like adaptive UI/UX and personalized avatars. My prototyping experience with basic virtual classrooms—testing features like seat selection and hand-raising—prepares me to tackle advanced challenges in this project.

For the AI Assistant: My familiarity with generative AI frameworks (TensorFlow, GPT-3.5 integrations) and NLP libraries positions me uniquely to develop the platform's AI guide. Outside of Alpha one I've built context-aware chatbots that streamline user navigation, answer queries dynamically, and adapt to individual learning styles—skills critical for creating an assistant that feels less like a tool and more like a mentor. By leveraging transformer models and fine-tuning them on datasets, I aim to ensure the chatbot aligns with Alpha One Labs' platform, offering personalized support without compromising scalability.

Beyond Code: My iterative approach to development—evident in refining the team collaboration module and fixing verification bottlenecks—ensures I prioritize user feedback at every stage. I thrive in balancing technical rigor with accessibility. My open-source contributions of building independent features and collaborative mindset, honed through resolving issues like permission mismatches and UI inconsistencies, guarantee I'll work cohesively with mentors to turn this vision into a transformative educational experience.

Is this the only project I'm applying for?

No, i have applied for another project (Alpha Forge) at Alpha One Labs

Working time

As this is a large project, I am willing to dedicate 5-6 hours per day until its completion. Overall, I plan to dedicate 35-40 hours per week (or more, if necessary) to building the project. Having built the basic outline and fundamental implementation already, I intend to allocate my time as follows:

- **30%** to backend development and database management.
- **40%** to frontend features.
- **30%** to optimizing approaches, exploring better APIs, and implementing advanced features and functionalities.