A Minor Project Final Report on

Presence: Automated Attendance System

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Bachelor of Engineering in Software Engineering

under Pokhara University

Submitted by:

Rupesh Budhathoki, 191721

Apsara Bishwokarma, 191704

Sampada Kharel, 191723

Sudarshan Kshetri, 191729

Under the supervision of:

Yogesh Deo

Date:

27 Sept 2023

Department of Software Engineering

NEPAL COLLEGE OF INFORMATION TECHNOLOGY

Balkumari, Lalitpur, Nepal

1 Acknowledgement

We would like to express our gratitude to the many individuals who played a vital role in making the ”Automated Face Attendance System” project a reality. Their support and guidance were crucial throughout the project, and we are sincerely thankful for their contributions.

We extend our special thanks to Mr.Birendra Bista, the Head of the Department of Software Engineering, for his invaluable guidance and technical support. Also our professor Dr. Roshan Chitrakar, for valuable suggestions, attention and time for our project.

We also appreciate our supervisor Yogesh Deo, for constant motivation and support during the course of our work throughout the semester which helped our project to grow and foster to a certain level we didn’t think of reaching in such a short period. We truly appreciate and value his esteemed guidance and encouragement from the beginning to the end of this project.

Above all, we would like to thank our friends whose direct and indirect support helped us to complete our project on time. This project would have been impossible without their moral support.

2 Abstract

The ”Presence” project is an innovative approach to attendance tracking in college classrooms. With the help of automation and data analysis, the system aims to provide a more efficient and accurate way of keeping track of student attendance while also providing valuable insights into student behavior and academic performance. The system uses cameras to detect students’ arrival. The data collected is then analyzed to provide a real-time attendance report that instructors can access from their devices. This eliminates the need for manual attendance-taking and reduces the potential for errors or inaccuracies. In addition to real-time attendance tracking, the system also provides detailed reports on student attendance patterns in class. By analyzing this data, instructors, and administrators can identify students who may be struggling with attendance or academic performance and take proactive steps to address these issues. They can also identify trends and patterns in attendance that can inform decisions about course scheduling and curriculum design.

The ”Presence” project is a valuable tool for both students and faculty. For students, it provides a streamlined attendance tracking process that eliminates the need for manual sign-ins and helps ensure that they are meeting attendance requirements. For faculty, it provides real-time insights into student attendance and behavior that can inform teaching strategies and improve academic outcomes.

Keywords– *automated attendance system, data analysis, college classrooms, attendance tracking, attendance report, academic performance, attendance patterns, student behavior, teaching strategies, curriculum design*

Contents

List of Figures

List of Tables

3 Introduction

3.1 Motivation/Problem Statement

In today’s technology-driven era, the trend is to switch from traditional systems to fast, smart, and interactive systems that can be linked to web applications, enabling users to access the system from anywhere at any time. In academic settings, regular attendance is crucial for students to acquire knowledge, participate in class discussions, and engage with their peers. However, traditional methods of taking attendance, such as calling out names or manually signing in, can be tedious, prone to errors, and inefficient.

The ”Presence” project is developed to address the challenges associated with manual attendance-taking in college classrooms. Traditional methods of attendance tracking, such as paper sign-ins or roll calls, are time-consuming, prone to errors, and may not provide accurate or timely data for instructors and administrators. Additionally, manual attendance-taking does not provide insights into student behavior or attendance patterns, which can be valuable for improving academic outcomes and making data-driven decisions.

3.2 Project Objectives

To address these challenges, developed a face detection and recognition system called ”Presence.” The ”Presence” system is an automated attendance system that utilizes facial recognition technology to identify and mark the attendance of students attending a lecture in a classroom. By automating the attendance process, ”Presence” aims to save time for instructors, reduce errors, and provide a more accurate and efficient way to monitor student attendance.

The project had put forward the following objectives:

- To automate attendance system for college classrooms
- To provide insights into attendance patterns and behavior.

- To support data-driven decision-making for improved student success.

3.3 Significance of the study

The ”Presence” project holds significant importance as it introduces an automated attendance system that addresses the limitations of manual methods. It streamlines attendance tracking, provides real-time reports, and offers insights into attendance patterns and student behavior. This empowers educators to make data-driven decisions, optimize instructional practices, and enhance student success in college classrooms.

4 Literature Review

This section consists description of the literature study performed during the development of this proposal.

4.1 Fareclock

Fareclock[1] is a web-based time clock software that offers features for employee time tracking, attendance management, and payroll processing. It provides businesses with tools to efficiently track employee work hours, monitor attendance, and generate accurate payroll reports. Fareclock allows employees to clock in and out using various methods such as biometric fingerprint scanning, web punch, mobile app, or phone call. The software also provides features like overtime tracking, PTO management, and shift scheduling. Fareclock aims to streamline the time management process and simplify payroll administration for businesses of all sizes.

4.2 Learning Management System (Moodle)

Moodle is a free and open-source learning management system. It has a variety of plugins to expand the features. There is a mod-attendance[2] plugin also called Attendance Activity[3] where the instructor clicks on the ”Update Attendance” button and is presented with a list of all the students in that course, along with configurable options and comments. The default options provided are Present, Absent, Late Excused. Instructors can download the attendance for their course in Excel format or text format. Sessions can also be configured to allow students to record their own attendance and a range of different reports are available.
Presence offers several advantages over the mod-attendance plugin in Moodle, making it a more beneficial solution for automated attendance tracking in college classrooms.

Firstly, Presence provides a seamless and automated attendance tracking process, eliminating the need for manual input by instructors. Unlike the mod-attendance plugin, which requires instructors to manually update attendance by selecting options for each student, Presence utilizes sensors or other automated methods to track student entry into the classroom. This not only saves instructors significant time and effort but also ensures more accurate and reliable attendance data.

Secondly, Presence offers advanced data analysis and reporting capabilities. While the mod-attendance plugin in Moodle allows instructors to generate basic attendance reports, Presence takes data analysis to a higher level. It compiles and analyzes attendance data, providing valuable insights into attendance patterns, subject preferences, and even student behavior. This level of analysis can help instructors and administrators make informed decisions regarding teaching strategies, course planning, and student support, ultimately improving overall educational outcomes.

However, since Moodle is open-source we can use this technology and integrate it with Moodle.

4.3 Existing Similar Appilcations

While there may be similar plugins or programs available that offer alternative approaches to attendance tracking, ”Presence” stands out by providing a comprehensive solution. It detects student arrivals, allowing for a more through analysis of attendance patterns. This valuable data can then be utilized to design improved timetables and curricula, ensuring optimal scheduling and enhancing the overall learning experience. By offering this unique functionality, Presence sets itself apart as a powerful tool for attendance management and curriculum optimization.

5 Methodology

This section describes the methodology that we followed during the development of our project, which is the waterfall model.

5.1 Software Development Life Cycle

For the development of our project, we chose to follow the waterfall model[4] of the software development life cycle as depicted in Figure 1. The reason for choosing this model was the lack of sufficient time duration for agile and iterative methods, as well as very low chances of changes of requirements in the process of development.

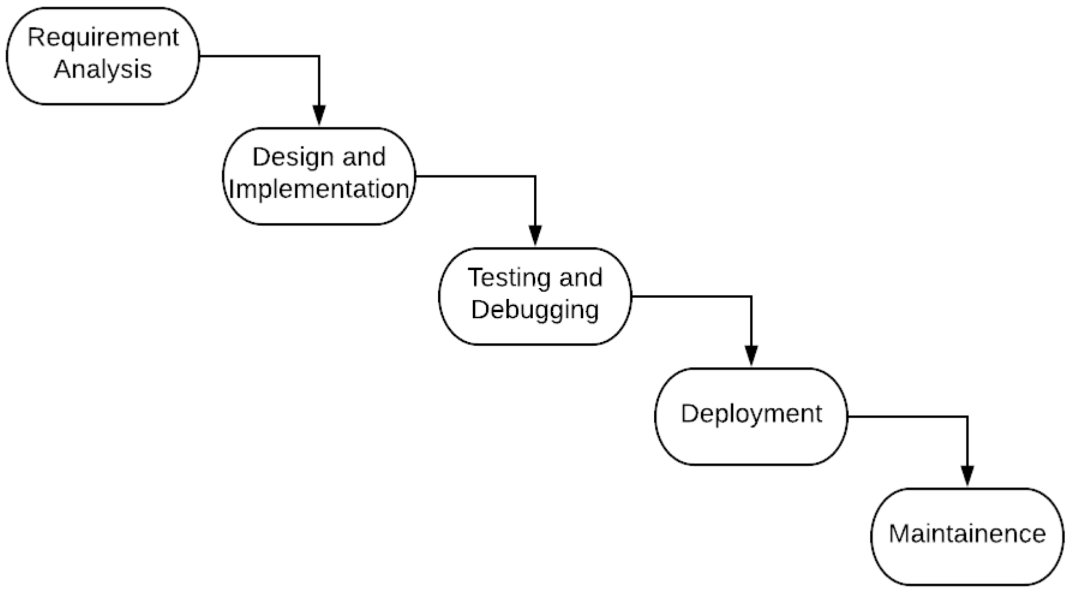


Figure 1: Software development life cycle

For the development of our project, we adopted the waterfall model as our chosen software development life cycle. In this model, the project life cycle began with the collection and evaluation of requirements for the application. We then proceeded to the design and implementation phase, where we designed and built both the API services and client applications. By the end of this phase, we had a minimal viable product (MVP) constructed.

In the testing and debugging phases, the quality control methods were applied to both the API and application. However, there might have been slight modifications in the original waterfall model where the design and implementation were changed slightly after the testing phase if seen as reasonable.

5.2 Technical Architecture

The application was built upon the client-server web architecture, as illustratedin Figure. 2.

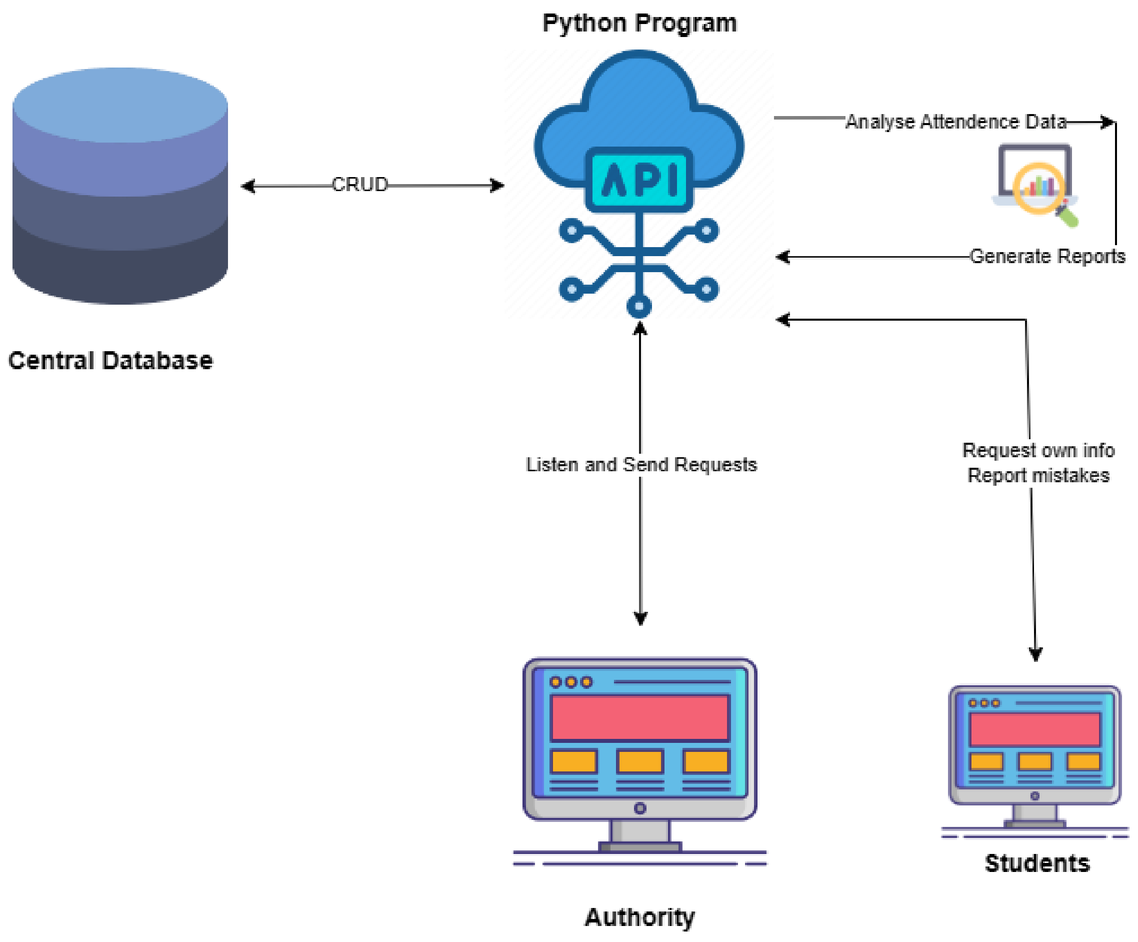


Figure 2: Architecture of the application

The architecture consists of a database that handles data storage and management. Python is responsible for handling Create, Read, Update, and Delete (CRUD) operations on the database. It also provides WebSocket communication capabilities to the client application.

The client application allows students to access their personal attendance information securely. Only authorized personnel, such as administrators or instructors, can access the entire dataset via the client application using WebSocket communication.

Python program acts as the intermediary between the client application and is also responsible for generating reports based on the attendance data. This communication ensures seamless integration and efficient report generation.

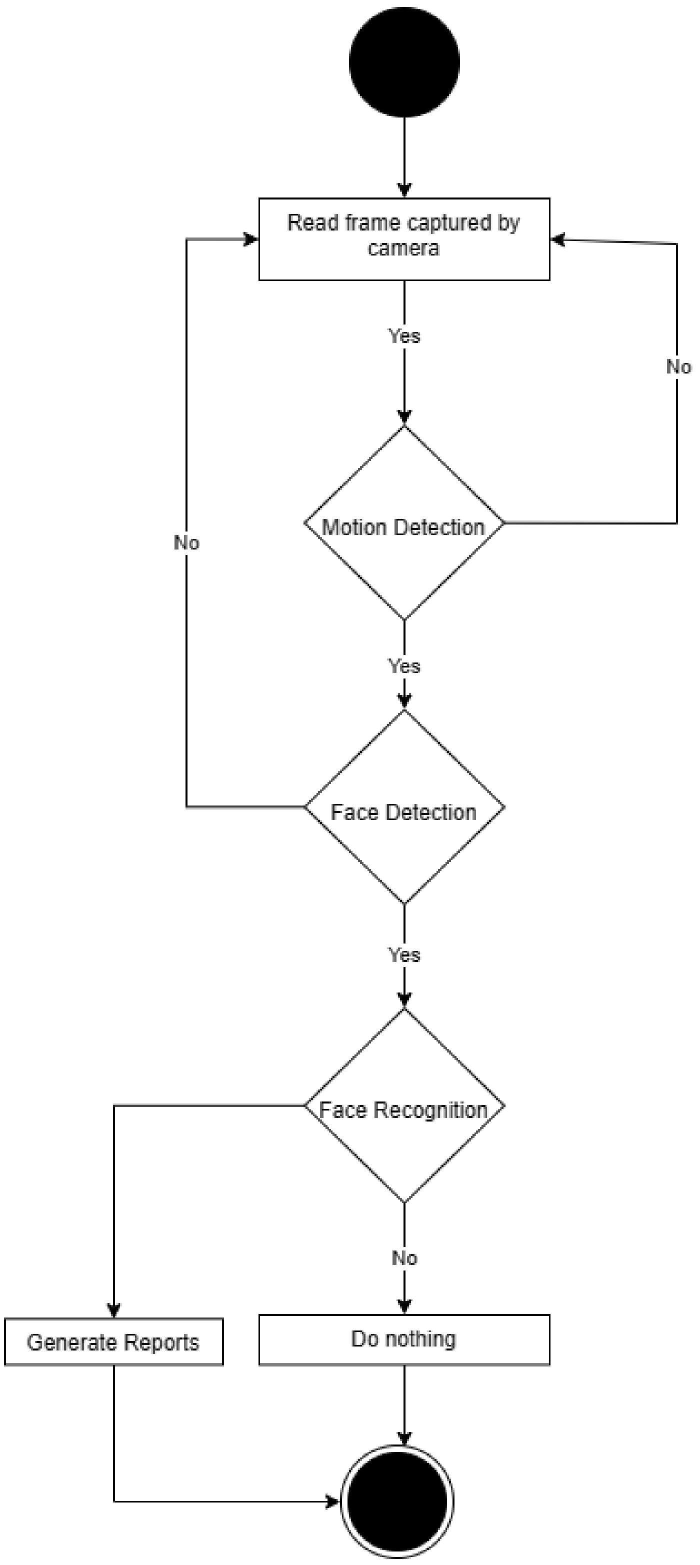


Figure 3: Activity Diagram[5] of the application

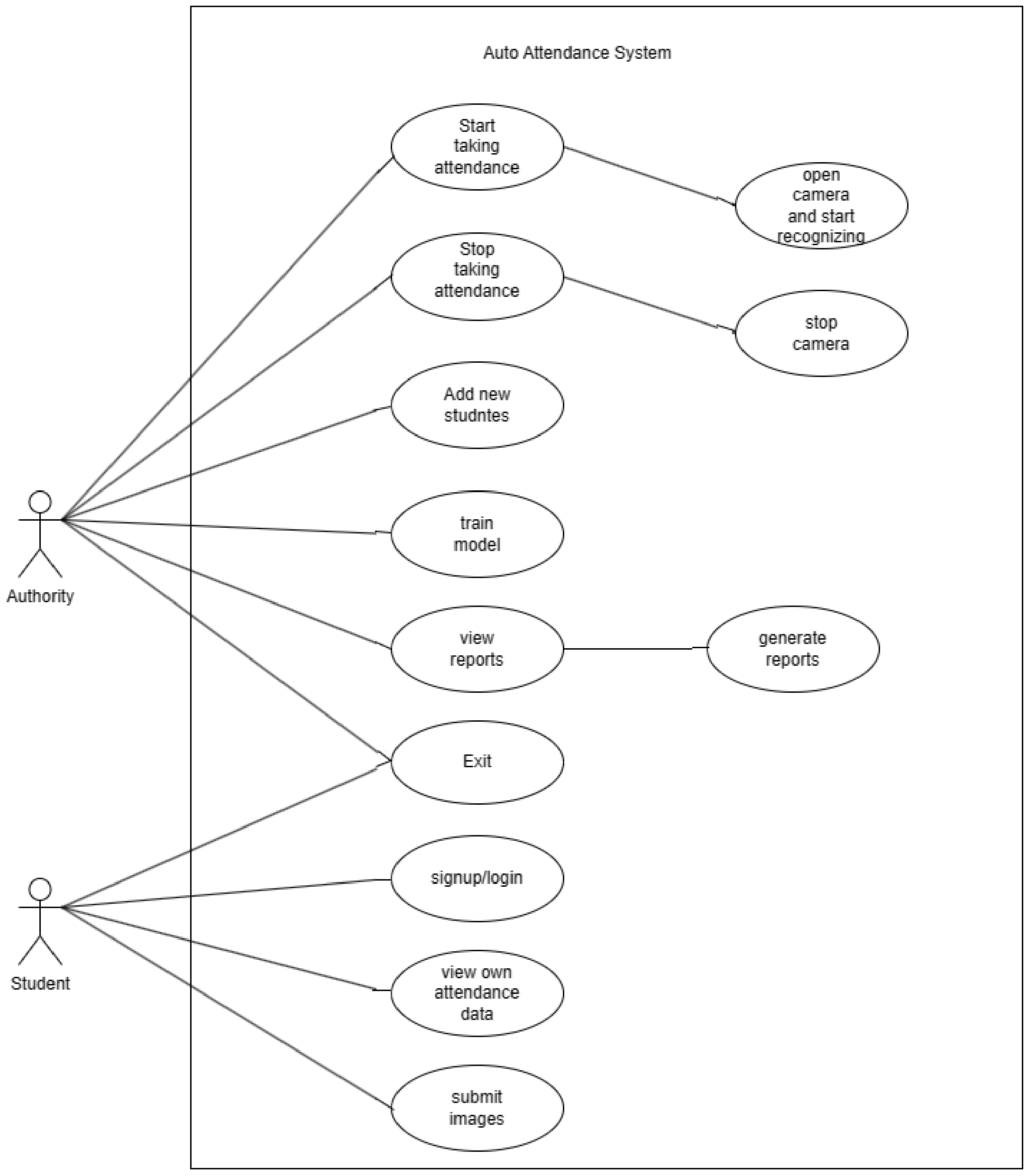


Figure 4: Use-case Diagram of the application

5.3 Technologies

Table 1 presents the major technologies that we used for the development and deployment of our application.

Subject	**Technology**
Backend Database	SQLite
Backend Service	django, Python
API Communication	REST APIs and WebSocket
Frontend(Client)	Next.js(React), Typescript, TailwindCSS

Table 1: Technologies we used

5.4 Face detection algorithm

For the development of our project, we utilized the OpenCV library in Python to train our model and perform face detection for attendance purposes. In this regard, we have two options for face detection: HOG(Histogram of Oriented Gradients) or CNN.

The HOG face detector offers faster processing speed, which aligns well with our limited resources and time constraints. It operates by dividing an image into small cells, computes each cell's gradient orientation and magnitude, and then aggregates the gradient information into a histogram of oriented gradients. These histograms describe the image features and detect objects within an image. However, it may be less accurate when dealing with changes in the viewing angle or rotation of faces.

For more robust face detection, we can employ the CNN face detector. This approach requires more computational resources, resulting in slower processing times. Nonetheless, it offers higher accuracy and is more resilient to variations in face rotation and viewing angles.

Furthermore, if we have access to a GPU, we can leverage it to run the CNN face detector, enabling real-time face detection. Combining the CNN face detector with a GPU delivers the perfect combination of deep neural network accuracy and the efficiency of a less computationally demanding model.

We have used HOG(Histogram of Oriented Gradients) because of low resources and less time.

Algorithm	**HOG**	**CNN**
Approach	Histogram of Oriented Gradients (HOG) feature extraction	Convolutional Neural Network (CNN)
Training Complexity	Less computationally intensive	More computationally intensive
Speed	Faster	Slower
Accuracy	Less accurate, especially with changes in viewing angles	More accurate and robust to face rotation
Face Rotation Tolerance	Low	High
GPU Support	Limited (may not utilize GPU)	Utilizes GPU for improved performance
Model Size	Smaller	Larger

Table 2: Comparison of HOG and CNN algorithms for face detection.

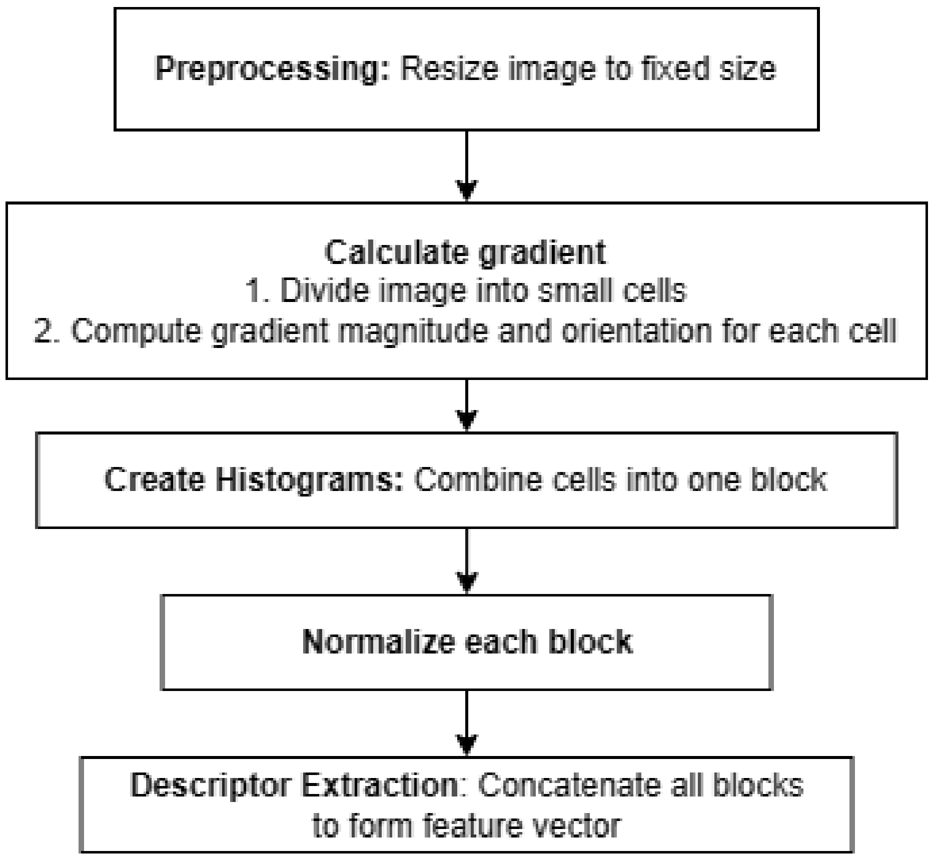

Preprocessing: Resize image to fixed size
Calculate gradient
1. Divide image into small cells
2. Compute gradient magnitude and orientation for each cell
Create Histograms: Combine cells into one block
Normalize each block
Descriptor Extraction: Concatenate all blocks to form feature vector

Figure 5: Hog process of feature extraction

5.5 Flow of user in the platform

Admin:

- Provides an admin panel where teachers, who are also admins, can add new student details.
- Admins then share these login details with students.
- Students use these credentials to log into the platform.
- Students can submit their images, which are processed using the HOG algorithm to create a pickle file containing facial details.
- Admins can take attendance using a camera.

- Real-time attendance logs are available in both text and video formats, which can be converted into reports.

Student:

- Students can log in using the credentials provided by their teachers.
- They have the option to request a password reset, which involves sending a reset link to their email.
- Clicking on the reset link verifies the provided tokens.
- Students can view their attendance in a calendar format.
- They can also track their attendance streak, which serves as a motivational factor.

6 Conclusion

In conclusion, ”Presence: an automated attendance system” represents a much-needed transition from outdated and error-prone manual attendance-tracking methods to a modern, efficient, and web-connected system. The primary goal of this project was to create a facial recognition system for tracking student attendance. Admins provide students with accounts and add their information to the database. The system then detects students’ faces using a camera and records their attendance in the database. Both teachers and admins have access to attendance reports.

Throughout this project, we conducted extensive research on various face recognition algorithms and selected the most suitable one, which happens to be HOG. We employed diverse techniques and programming languages to develop the system, thoroughly tested it, and achieved success. As a result, this system can now be implemented to efficiently manage student attendance records.

Looking ahead, there is ample potential for further development. Additional features beneficial to both students and teachers, such as assignment tracking, result recording, and grade management, could be seamlessly integrated into the system.

7 Further Works / Recommendations

- **Record attendance for each subject**: For now we are only recording attendance for a day. In future We will add functionality to record the attendance of each subject for better reports and insights.
- **Enhance Facial Recognition Accuracy**: improving the facial recognition algorithms to ensure high accuracy, especially in challenging conditions, such as low light or varying student appearances (e.g., facial hair, makeup, or accessories).
- **Mobile Integration**: Develop a mobile app for students to easily track their attendance using their smartphones, providing an additional convenience layer to the system.
- **Customization Options**: Provide customization features that allow educational institutions to tailor the system to their specific needs, including attendance policies and reporting formats.

8 Project Task and Time schedule

The working time period for the project was four months. The project was completed by the end of the spring semester as per the requirements of the university. The major task division among the team members was mentioned in the table. 3.

Team Member	**Assigned Task**
Rupesh Budhathoki	Project Management, Overall Coordination, FullStack Development
Apsara Bishwokarma	Frontend Development
Sampada Kharel	Backend Development
Sudarshan Kshettry	FullStack Development

Table 3: Division of tasks among project team members

The time schedule for the development of the project was illustrated in the following Gantt chart.

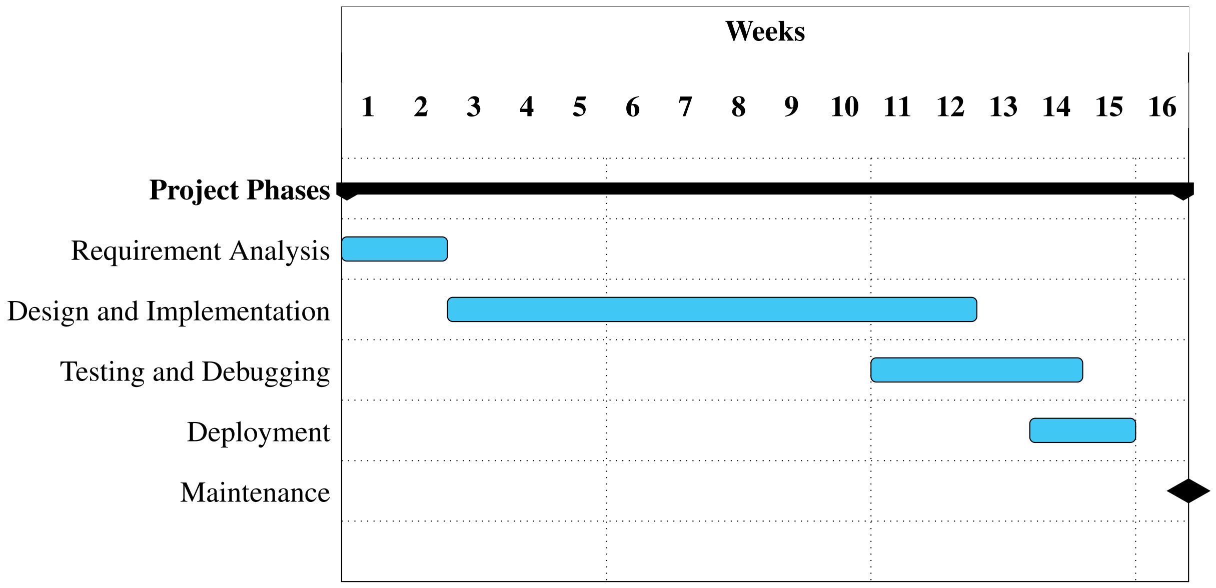


Table 4: Gantt Chart

9 References

[1] K.-O. Goh, C.-Y. Law, C.-K. Tin, C. Tee, and Y.-W. Sek, “Attendance management system with half-covered and full-facial recognition feature,” in *Proceedings of the 9th International Conference on Computational Science and Technology: ICCST 2022, 27–28 August, Johor Bahru, Malaysia*, Springer, 2023, pp. 247–258.

[2] D. Marsden, *Mod-attendance plugin on moodle lms*, 2023.

[3] Moodle, *Attendance activity on moodle lms*, 2023.

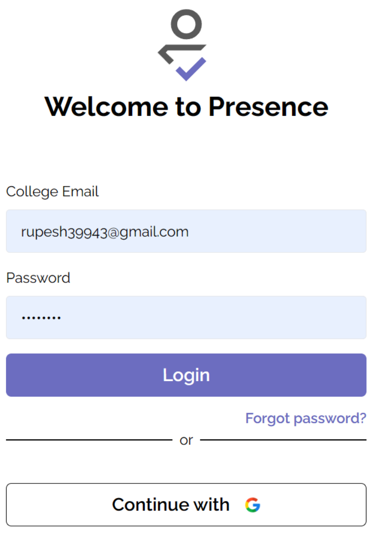
[4] D. Nurcahya, H. Nurfauziah, and H. Dwiatmodjo, “Comparison of waterfall models and prototyping models of meeting management information systems,” *Jurnal Mantik*, vol. 6, no. 2, pp. 1934–1939, 2022.

[5] M. L. Yasmeen, M. H. R. Reddy, B. Mridgala, M. P. Reddy, and P. Chandana, “Smart attendance using facial recognition,” *YMER*, vol. 22, no. 1, pp. 370–373, 2023.

10 Appendix

To ensure the security and accessibility of the system, we have implemented various authentication mechanisms, including:

- **Login**: Users can securely login using their credentials to access the system’s features and functionalities.


Welcome to Presence

College Email

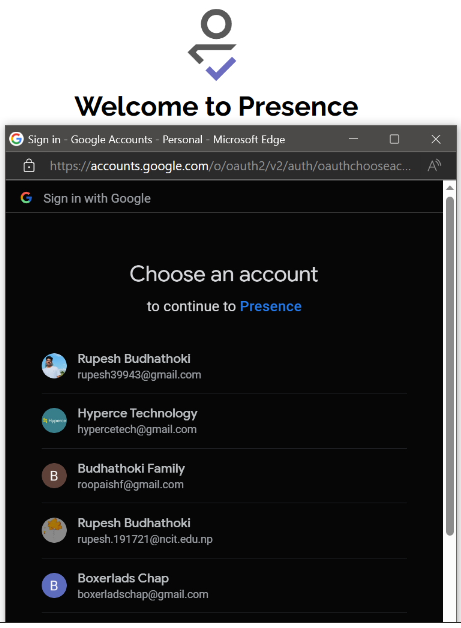
rupesh39943@gmail.com

Password

....

Figure 6: Login form for both users and admin

- **Login with Google**: Users have the option to authenticate themselves using their Google accounts, providing a convenient and streamlined login experience.


Welcome to Presence

Sign in - Google Accounts - Personal - Microsoft Edge
https://accounts.google.com/o/oauth2/v2/auth/oauthchooseac...
Sign in with Google

Choose an account

to continue to Presence

Rupesh Budhathoki
rupesh39943@gmail.com

Hyperce Technology
hypercetech@gmail.com

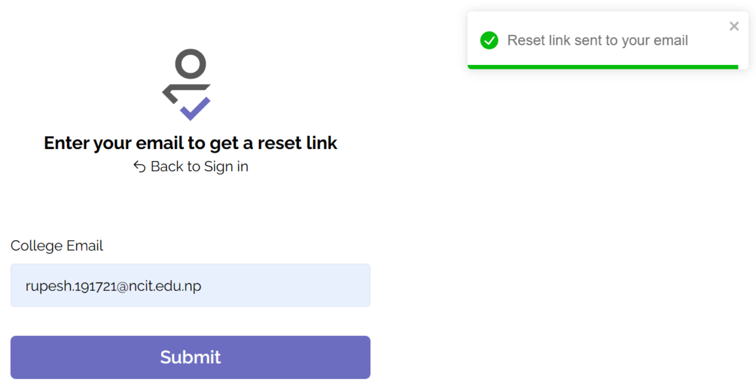
Budhathoki Family
roopaishf@gmail.com

Rupesh Budhathoki
rupesh.191721@ncit.edu.np

Boxerlads Chap
boxerladschap@gmail.com

Figure 7: Login with Google functionality

- **Forgot Password**: In the event of a forgotten password, a password recovery mechanism allows users to reset their passwords securely.


Reset link sent to your email

Enter your email to get a reset link

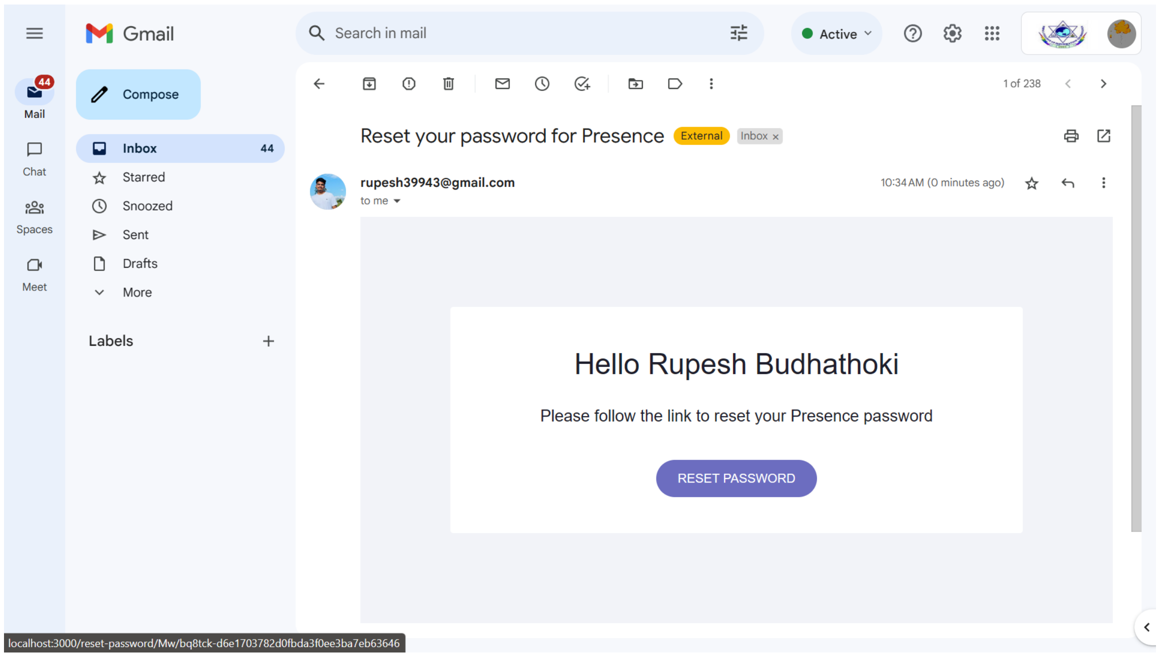
Back to Sign in

College Email

rupesh.191721@ncit.edu.np

Figure 8: Login with google functionality

- **Reset Password**: Users can initiate the password reset process and set a new password to regain access to their accounts.


Reset your password for Presence

rupesh39943@gmail.com

to me

10:34 AM (0 minutes ago)

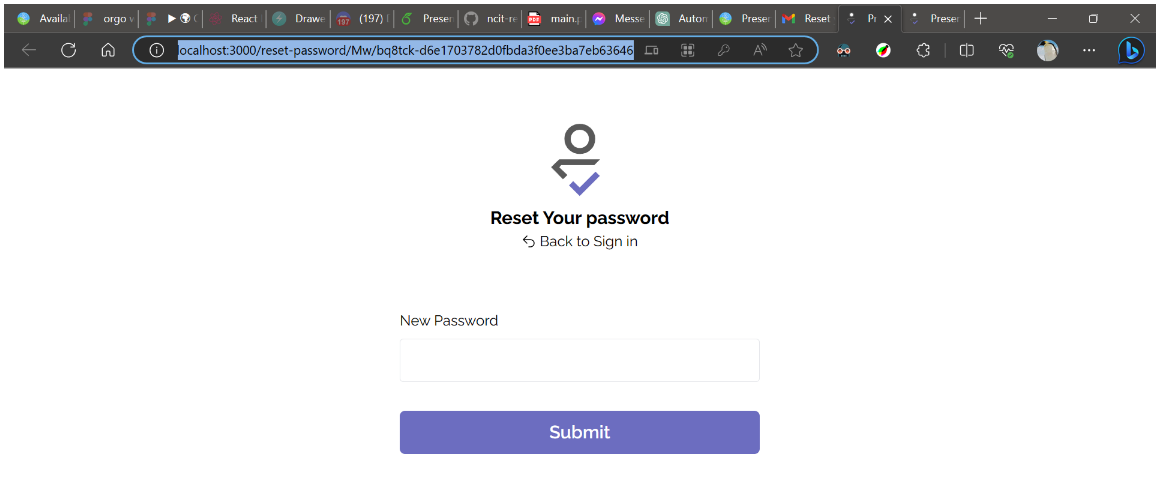
Hello Rupesh Budhathoki

Please follow the link to reset your Presence password

RESET PASSWORD

localhost:3000/reset-password/Mw/bq8tck-d6e1703782d0fbda3f0ee3ba7eb63646

Figure 9: Reset Password Email


localhost:3000/reset-password/Mw/bq8tck-d6e1703782d0fbda3f0ee3ba7eb63646

Reset Your password

Back to Sign in

New Password

Submit

Figure 10: Reset password

Key Features for Students:

- **Image Submission**: Students can submit their images to be used for facial recognition and attendance tracking.

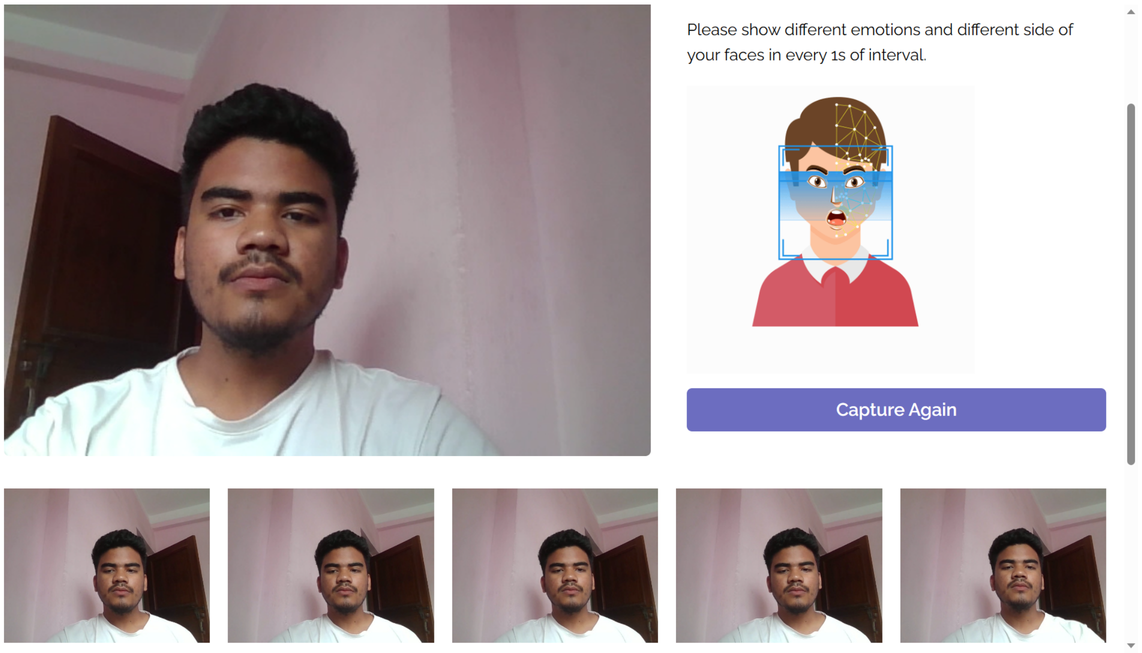

Please show different emotions and different side of your faces in every 1s of interval.

Figure 11: Image submission form

- **Attendance Tracking**: Students have access to view their attendance records, providing insights into their attendance patterns and history.

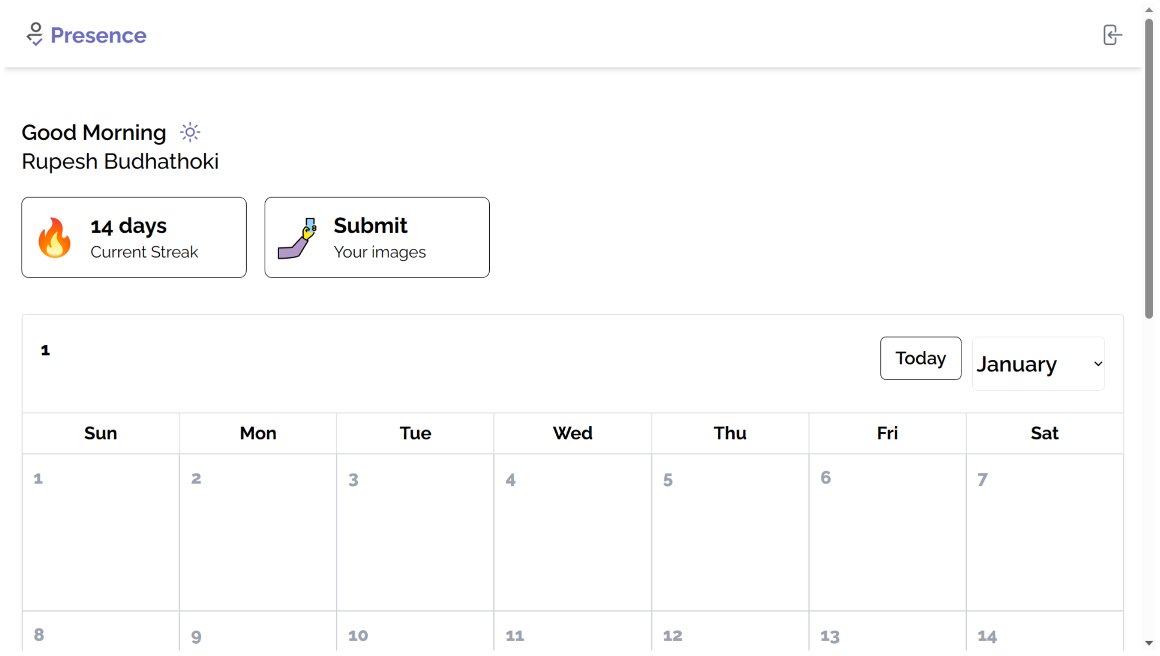
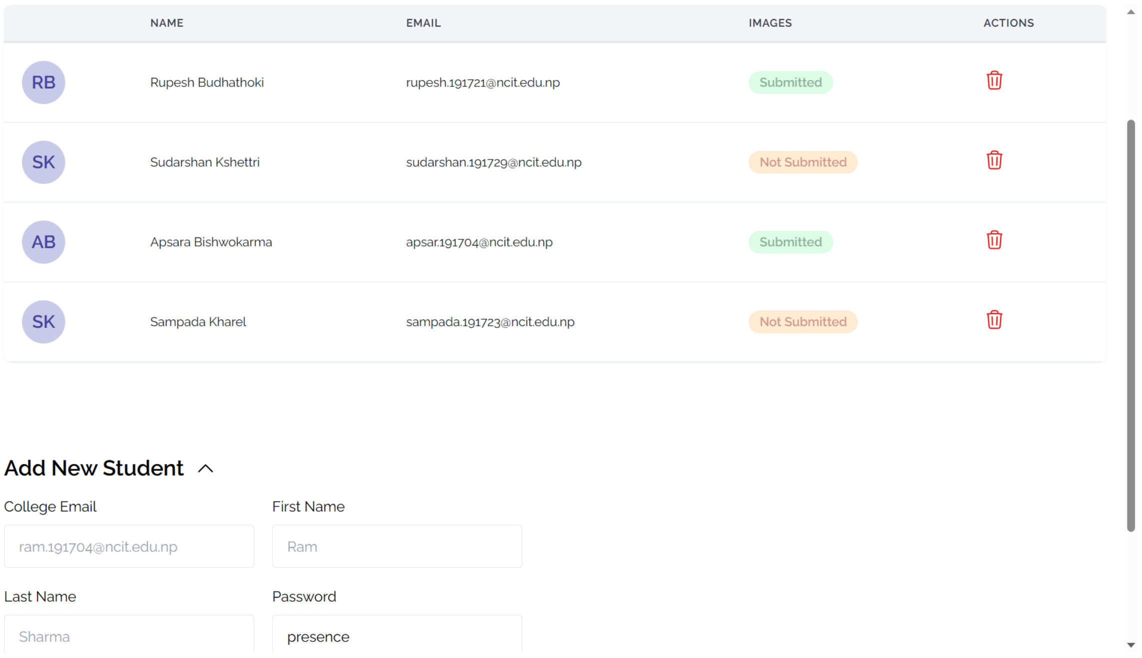


Figure 12: Student dashboard

Key Features for Authorities:

- **Student Management**: Authorities can add new students to the system, maintaining an up-to-date database of enrolled individuals.



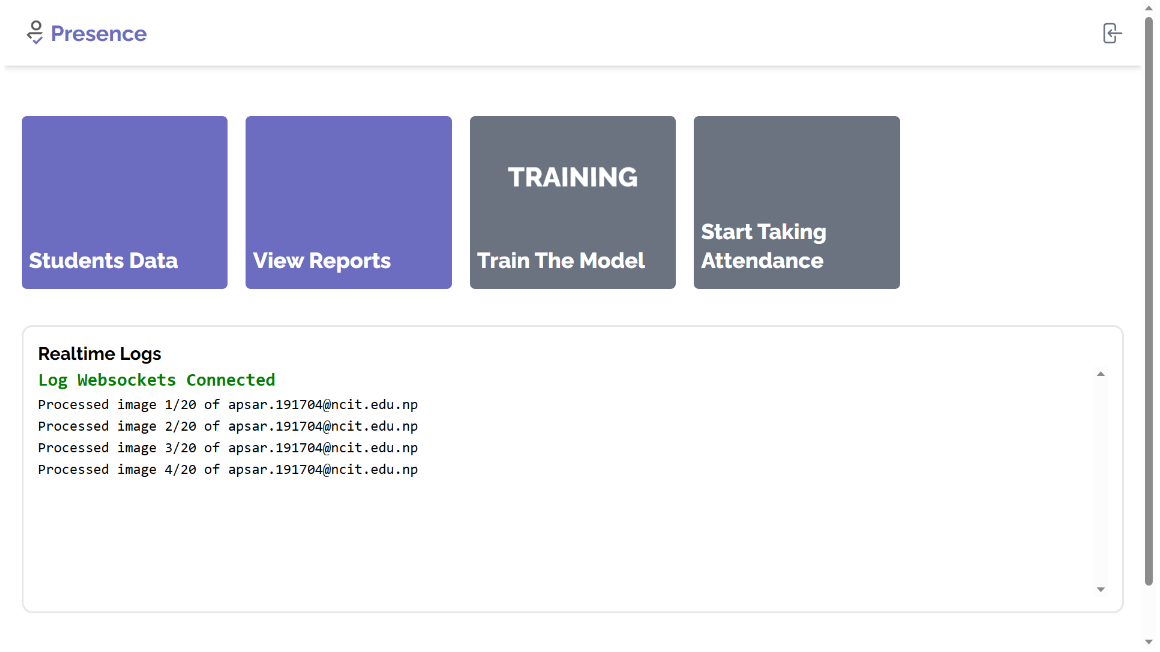
	NAME	EMAIL	IMAGES	ACTIONS
RB	Rupesh Budhathoki	rupesh.191721@ncit.edu.np	Submitted	
SK	Sudarshan Kshettri	sudarshan.191729@ncit.edu.np	Not Submitted	
AB	Apsara Bishwokarma	apsar.191704@ncit.edu.np	Submitted	
SK	Sampada Kharel	sampada.191723@ncit.edu.np	Not Submitted	

Add New Student

College Email	First Name
ram.191704@ncit.edu.np	Ram
Last Name	Password
Sharma	presence

Figure 13: Student Management

- **Model Training**: Authorities have the ability to train the facial recognition model using images submitted by students, ensuring accurate and reliable attendance detection.

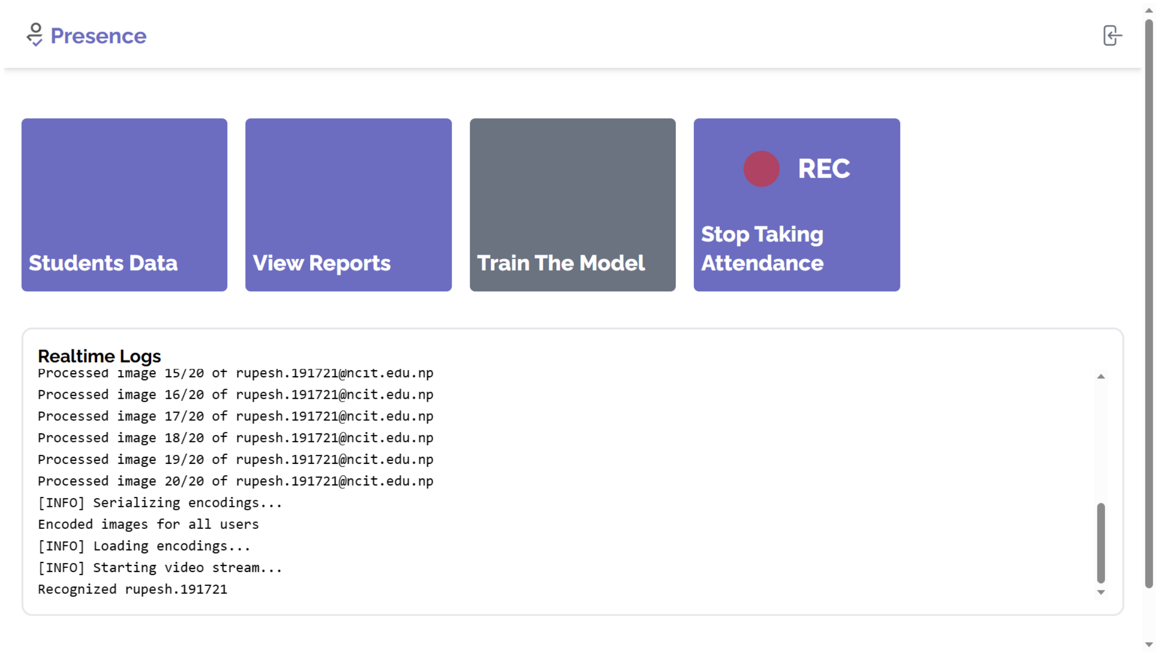


Realtime Logs

```
Log Websockets Connected
Processed image 1/20 of apsar.191704@ncit.edu.np
Processed image 2/20 of apsar.191704@ncit.edu.np
Processed image 3/20 of apsar.191704@ncit.edu.np
Processed image 4/20 of apsar.191704@ncit.edu.np
```

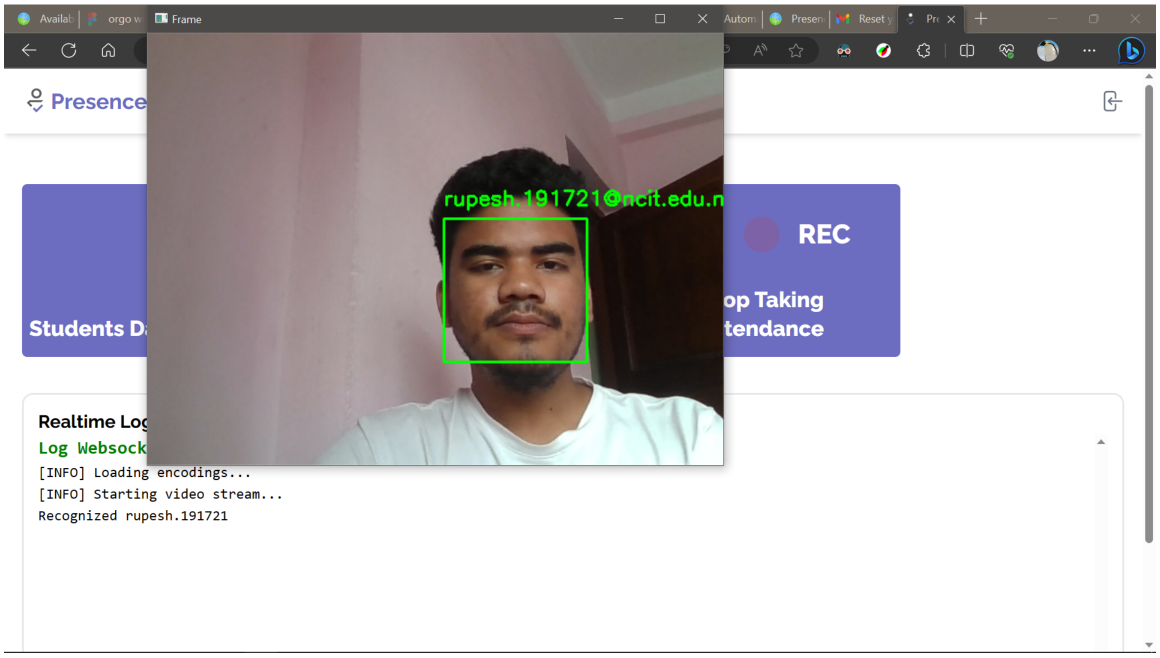
Figure 14: Training model with images submitted by students

- **Real-time Attendance**: Authorities can initiate the real-time attendance process, enabling the system to mark attendance as students are recognized.



```
Processed image 15/20 of rupesh.191721@ncit.edu.np
Processed image 16/20 of rupesh.191721@ncit.edu.np
Processed image 17/20 of rupesh.191721@ncit.edu.np
Processed image 18/20 of rupesh.191721@ncit.edu.np
Processed image 19/20 of rupesh.191721@ncit.edu.np
Processed image 20/20 of rupesh.191721@ncit.edu.np
[INFO] Serializing encodings...
Encoded images for all users
[INFO] Loading encodings...
[INFO] Starting video stream...
Recognized rupesh.191721
```

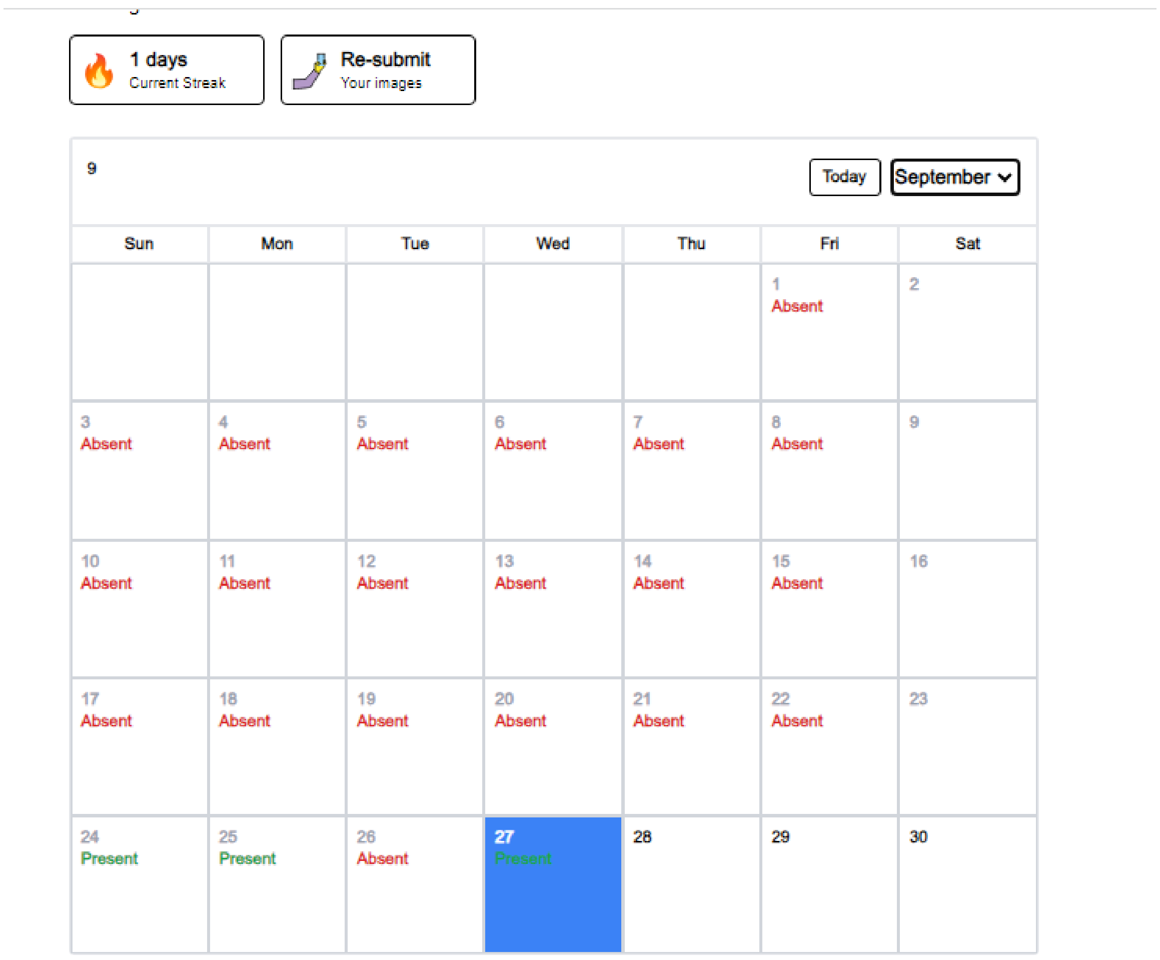
Figure 15: Realtime attendance log



```
Log Websock
[INFO] Loading encodings...
[INFO] Starting video stream...
Recognized rupesh.191721
```

Figure 16: Realtime attendance from cam

- **Student Calendar**: Student attendance status.



9

Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1 Absent	2
3 Absent	4 Absent	5 Absent	6 Absent	7 Absent	8 Absent	9
10 Absent	11 Absent	12 Absent	13 Absent	14 Absent	15 Absent	16
17 Absent	18 Absent	19 Absent	20 Absent	21 Absent	22 Absent	23
24 Present	25 Present	26 Absent	27 Present	28	29	30

Figure 17: Student Calendar

- **Encode Faces**: Code snippet to encode faces and make a pickle file.

```
@user_passes_test(lambda u: u.is_superuser)
def encode_images(request):
    dataset_path = 'datasets'
    encodings_path = 'encodings.pickle'
    channel_layer = get_channel_layer()
```

```
async_to_sync(channel_layer.group_send)(
    'status_group',
    {
        'type': 'status_message',
        'message': "encoding_images"
    }
)

if not os.path.exists(dataset_path):
    return JsonResponse({'success': False, 'message':
    'Dataset directory not found'}, status=400)

image_paths = list(paths.list_images(dataset_path))

known_encodings = []
known_names = []

for (i, image_path) in enumerate(image_paths):
    name = os.path.basename(os.path.dirname(image_path))
    image = cv2.imread(image_path)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    boxes = face_recognition.face_locations(rgb)
    encodings = face_recognition.face_encodings(rgb, boxes)

    for encoding in encodings:
        known_encodings.append(encoding)
        known_names.append(name)

    async_to_sync(channel_layer.group_send)(
        'log_group',
        {
            'type': 'log_message',
            'message': f"""Processed image
            {i+1}/{len(image_paths)} of {name}"""
        }
    )

async_to_sync(channel_layer.group_send)(
    'log_group',
    {
        'type': 'log_message',
        'message': f"[INFO] Serializing encodings..."
    }
)
data = {"encodings": known_encodings, "names": known_names}
with open(encodings_path, "wb") as f:
    f.write(pickle.dumps(data))

async_to_sync(channel_layer.group_send)(
    'log_group',
    {
        'type': 'log_message',
        'message': "Encoded images for all users"
    }
)
async_to_sync(channel_layer.group_send)(
    'status_group',
    {
        'type': 'status_message',
                'message': "none"
            }
        )

        return JsonResponse({'success': True})


stop_stream = False
```

- **Take Attendance**: Code snippet to take attendance of users.

```
@user_passes_test(lambda u: u.is_superuser)
@csrf_exempt
def take_attendance(request):
    global stop_stream


    if request.method == 'POST':
        json_data = json.loads(request.body)
        display_video = json_data.get('display_video') or False
        model = json_data.get('model') or 'hog'
        day = date.today().day
        month = date.today().month
        year = date.today().year

    # return JsonResponse({'success': month, 'message': 'month'},s


        if Attendance.objects.filter(day=day, month=month, year=ye
            return JsonResponse({'success': day, 'message': """Att
        for today is already taken"""}, status=200)

    channel_layer = get_channel_layer()
    dataset_path = 'datasets'
    encodings_path = 'encodings.pickle'
    image_paths = list(paths.list_images(dataset_path))

    if not os.path.exists(dataset_path):
        return JsonResponse({'success': False, 'message': """D
        directory not found"""}, status=400)

    # Load the known faces and embeddings
    async_to_sync(channel_layer.group_send)(
        'log_group',
        {
            'type': 'log_message',
            'message': "[INFO] Loading encodings..."
        }
    )
    async_to_sync(channel_layer.group_send)(
        'status_group',
        {
            'type': 'status_message',
            'message': "taking_attendance"
        }
    )
    data = pickle.loads(open(encodings_path, "rb").read())
    # Initialize the video stream and allow the camera sensor
    # to warm up
    async_to_sync(channel_layer.group_send)(
    'log_group',
    {
        'type': 'log_message',
        'message': "[INFO] Starting video stream..."
    }
)
vs = VideoStream(src=0).start()
time.sleep(2.0)

detected_users = []
names = []
# get all users and make a list of {name, email}
attendance_stream = {
    'present_users': [],
    'absent_users': [],
}
users = User.objects.all()
for user in users:
    attendance_stream['absent_users'].append({
        'name': user.first_name + ' ' + user.last_name,
        'email': user.email
    })

today = date.today()
attendance_queryset = Attendance.objects.filter(
    day=today.day, month=today.month, year=today.year)

# Loop over frames from the video file stream
while True:
    if stop_stream:
```

```
    break
# Grab the frame from the threaded video stream
frame = vs.read()

# Convert the input frame from BGR to RGB
# then resize it to have
# a width of 750px (to speed up processing)
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
rgb = imutils.resize(frame, width=750)
r = frame.shape[1] / float(rgb.shape[1])
# Detect the (x, y)-coordinates of the bounding boxes
# corresponding to each face in the input frame, then
# the facial embeddings for each face
boxes = face_recognition.face_locations(rgb, model=mod
encodings = face_recognition.face_encodings(rgb, boxes
names = []

# Loop over the facial embeddings
for encoding in encodings:
    # Attempt to match each face in the
    # input image to our known encodings
    matches = face_recognition.compare_faces(
        data["encodings"], encoding)
    name = "Unknown"
    # Check to see if we have found a match
    if True in matches:
        # Find the indexes of all matched faces
        # then initialize a
        # dictionary to count the
        # total number of times each face
        # was matched
        matchedIdxs = [i for (i, b) in enumerate(match
        counts = {}
        # Loop over the matched indexes and
        # maintain a count for
        # each recognized face face
        for i in matchedIdxs:
            name = data["names"][i]
            counts[name] = counts.get(name, 0) + 1
        # Determine the recognized face with the large
        # of votes (note: in the event of an unlikely
        # will select the first entry in the dictionar
        name = max(counts, key=counts.get)

        confidence = (counts[name] / len(matchedIdxs))
        print(
            f"""Recognized face: {name}
            (Confidence: {confidence:.2f}%)""")

    # Update the list of names
    names.append(name)
    if name not in detected_users:
        detected_users.append(name)
        striped_email = name.split('@')[0]
        async_to_sync(channel_layer.group_send)(
            'log_group',
            {
                'type': 'log_message',
                'message': f"Recognized {striped_email
            }
        )
        for user in attendance_stream['absent_users']:
            if user['email'] == name:
                attendance_stream['absent_users'].remo
                attendance_stream['present_users'].app
                async_to_sync(channel_layer.group_send
                    'attendance_group',
                    {
                        'type': 'attendance_message',
                        'message': attendance_stream
                    }
                )
                break

    if len(detected_users) >= len(image_paths):
        stop_stream = True
        break

if display_video:
    # Loop over the recognized faces
    for ((top, right, bottom, left), name) in zip(boxe
        # Rescale the face coordinates
        top = int(top * r)
        right = int(right * r)
        bottom = int(bottom * r)
        left = int(left * r)
        # Draw the predicted face name on the image
        cv2.rectangle(frame, (left, top),
                      (right, bottom), (0, 255, 0), 2)
        y = top - 15 if top - 15 > 15 else top + 15
```

```
                cv2.putText(frame, name, (left, y),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0

                # Display the output frame to the screen
                cv2.imshow("Frame", frame)
                key = cv2.waitKey(1) & 0xFF

        # If the `q` key was pressed, break from the loop
        if key == ord("q"):
            break

    stop_stream = False
    # Do a bit of cleanup
    cv2.destroyAllWindows()
    vs.stop()
    vs.stream.release()

    try:
        # Check attendance and mark it in the database
        today = date.today()
        attendance_queryset = Attendance.objects.filter(
            day=today.day, month=today.month, year=today.year)
        previous_day = today - timedelta(days=1)

        if previous_day.weekday() == 5:  # If previous day was
            previous_day -= timedelta(days=1)

        # Get the list of Attendance objects for the previous
        previous_attendance_queryset = Attendance.objects.filt
            day=previous_day.day,
            month=previous_day.month,
            year=previous_day.year)

        # Create a dictionary to store attendance status
        # (present or not) for each user
        prev_attendance_status = {}

        for email in names:
            if previous_attendance_queryset.filter(user__email
                prev_attendance_status[email] = True
            else:
                prev_attendance_status[email] = False

        for email, was_present in prev_attendance_status.items
            user = User.objects.get(email=email)
            attendance, created = Attendance.objects.get_or_cr
                user=user, day=today.day, month=today.month, y
            )
            striped_email = email.split('@')[0]

            if was_present:
                if email in detected_users:
                        attendance.streak = attendance.streak
                else:
                        attendance.streak = 1
                        async_to_sync(channel_layer.group_send
                            'log_group',
                        {
                            'type': 'log_message',
                            'message': f"Attendance taken for
                        }
                    )
                # attendance.streak = attendance.streak + \
                #     1 if attendance_queryset.filter(
                #         user__name=email).exists() else 1


                # async_to_sync(channel_layer.group_send)(
                #     'log_group',
                #     {
                #         'type': 'log_message',
                #         'message': f"Attendance taken for {s
                #     }
                # )
            else:
                attendance.streak = 1
                async_to_sync(channel_layer.group_send)(
                    'log_group',
                    {
                        'type': 'log_message',
                        'message': f"Attendance taken for {str
                    }
                )
            attendance.save()

        async_to_sync(channel_layer.group_send)(
            'log_group',
            {
                'type': 'log_message',
                'message': f"Stopped taking attendance for {to
```

```
                }
            )
            async_to_sync(channel_layer.group_send)(
                'status_group',
                {
                    'type': 'status_message',
                    'message': "none"
                }
            )
            return JsonResponse({'success': True, 'message': 'Atte
        except Exception as e:
            async_to_sync(channel_layer.group_send)(
                'log_group',
                {
                    'type': 'log_message',
                    'message': f"Error while taking attendance: {e
                }
            )
            async_to_sync(channel_layer.group_send)(
                'status_group',
                {
                    'type': 'status_message',
                    'message': "none"
                }
            )
            return JsonResponse({'success': False, 'message': f'Er

    else:
        return JsonResponse({'success': False, 'message': 'Invalid
```