DSD/BSE: 202.2

Project Proposal

Attendance Tracker Application

Sunny Gandhi 270237323
Datu Beech 270298599
Mandeep Singh 270294907
William Giles 270318633

Contents

Product Concept:

General Concept

A computer-vision based ‘attendance tracker’ system to automate the process of recording student attendance in a classroom.

Product Rationale/Field Review

Students with higher attendance rates have better grades.

Following the COVID-19 lockdowns there was a consistent drop in New Zealand school attendance and an associated decline in academic outcomes. For this reason improving the attendance rates of students has been identified as an important issue in education in New Zealand in 2024. An example of this is Yoobee Colleges, who have implemented a 90% minimum attendance policy.

Accurately recording the attendance (or absence) of a student is vital for implementing these policies and ensuring the wellbeing and academic success of a student.

Traditional methods of attendance tracking generally rely on a teacher manually updating a spreadsheet or a pen and paper record at the start of class.

These processes are outdated, disruptive, and create many opportunities for input error - they can even be forgotten about completely.

Our application will serve as a modern and reliable solution to these problems.

Proposed Features

The key proposed features of this application are:

- Connectivity to an external camera or video feed
- Face and feature classification and extraction
- Accurate comparison of extracted faces and/or features to a database of enrolled students
- Real time updating of the class roll/attendance in the database
- Administrator can add new students and remove students from the database
- Alerts for low-attendance and viewable class attendance data
- Teacher and Administrator Login/LogOut
- User-friendly Desktop GUI

Proposed Programming Framework Technology

Programming Languages

- Python

Python Libraries

- Open-CV

GUI Implementation

- Tkinter

Development Environment

- VS Code
- Anaconda Workstation

Production Plan:

Project Duration:
16 weeks

Project Timeline:

Week 1 - Week 4:

- UX/UI Research, UI Lofi Design and User Testing
- Tkinter, Python, OpenCV Research

Week 5 - Week 8:

- UI Development
- Development of Prototype (First Working Version of Application)

At the end of Week 8 we will submit a working prototype with basic/usable UI. We will begin development on the full/final application based on feedback.

Week 9 - Week 13:

- Development of Final Application Functionality
- Development of Final Application UI

Week 14 - Week 15:

- Testing/Demonstrating Working Software to Client for Feedback
- Debugging

Week 16:

- Deployment

Development Methodology:

Agile

- Agile is the industry standard in software development projects. Quickly developing iterations of working software is the key aspect of Agile software development. Fast and iterative development creates many opportunities for feedback from testers and clients and allows the team to quickly address design and development issues as they appear.

Scrum

- Scrum is a widely used style of Agile software development. The key aspect of Scrum is dividing a project timeline in smaller units called *Sprints* (in our project these will be one week long). Small goals and deliverables are established at the start of each sprint in a Sprint Planning meeting, short Scrum meetings are held daily at the start of the work day, and a *Sprint Retrospective* is held at the end of each sprint. The key advantage of using Scrum is facilitating consistent and up to date communication between team members.

Scrum Roles

Scrum Teams typically have members of a development team take specific roles.

- Scrum Master - Sunny
- Product Owner - Datu
- UX/UI Designer - Mandeep
- Developer - William

Communication Plan:

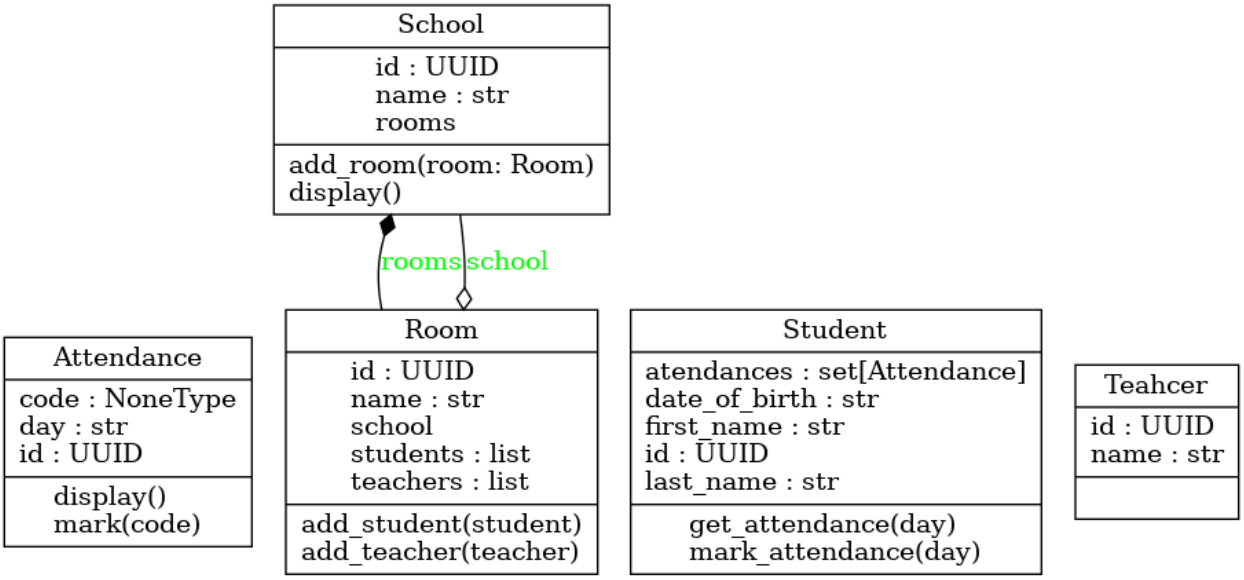
- Daily Scrum Meeting and Stand Up (10 Minutes)
- Weekly Scrum Planning/Retrospective
- MS Teams for Digital Communication

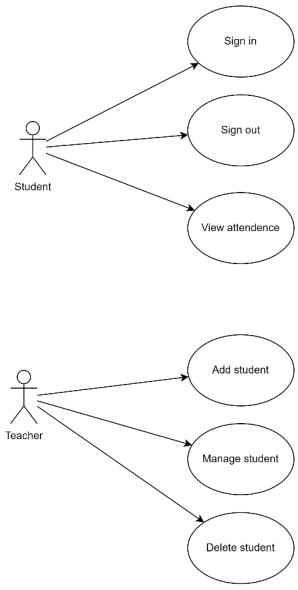
Project Management Tools:

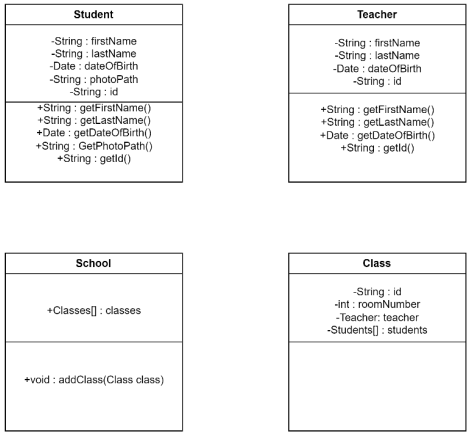
- Trello (Scheduling)
- GitHub (Version Control)
- Google Drive/Docs (Documentation)

Proposed Technical Design:

Use Case Diagrams & UML Sequence Diagrams







Proposed Class Structures:

```
import uuid

today = "2024-5-01"


class Attendance:
    def __init__(self, day: str):
        self.id = uuid.uuid4()
        self.day = day
        self.code = None

    def mark(self, code):
        self.code = code

    def display(self):
        print(f"Day: {self.day}, Code: {self.code}")

    def __hash__(self) -> int:
        return hash(self.id)

    def __eq__(self, o: object) -> bool:
        if not isinstance(o, type(self)):
            return NotImplemented
        return self.id == o.id
```

```
class Student:
    def __init__(self, first_name, last_name, date_of_birth):
        self.id = uuid.uuid4()
        self.first_name: str = first_name
        self.last_name: str = last_name
        self.date_of_birth: str = date_of_birth
        self.atendances: set[Attendance] = set()

    def mark_attendance(self, day):
        attendance = Attendance(day)
        attendance.mark("P")
        self.atendances.add(attendance)
        return attendance

    def get_attendance(self, day):
        for attendance in self.atendances:
            if attendance.day == day:
                return attendance
        return None
```

```
class Teahcer:
    def __init__(self, name):
        self.id = uuid.uuid4()
        self.name: str = name


class Room:
    def __init__(self, name, school):
        self.id = uuid.uuid4()
        self.name: str = name
        self.school: School = school
        self.teachers = []
        self.students = []

    def add_teacher(self, teacher):
        self.teachers.append(teacher)

    def add_student(self, student):
        self.students.append(student)


class School:
    def __init__(self, name):
        self.id = uuid.uuid4()
        self.name: str = name
        self.rooms: Room = []

    def add_room(self, room: Room):
        self.rooms.append(room)

    def display(self):
        for room in self.rooms:
            print("###################")
            print(f"Room: {room.name}")
            print(f"Teachers: ({len(room.teachers)})")
            for teacher in room.teachers:
                print(f"  - {teacher.name}")
            print(f"Students: ({len(room.students)})")
            for student in room.students:
                print(f"  - {student.first_name} {student.last_name}")
            print("###################")
```

```
school = School("School")
room = Room("Room 1", school)
school.add_room(room)

teacher = Teahcer("Teacher 1")
room.add_teacher(teacher)

student = Student("John", "Doe", "2000-01-01")
room.add_student(student)

student.mark_attendance(today)

school.display()

att = student.get_attendance(today)
# print(att)
att.display()
```

Proof of Concept:
Face Detection and Feature Extraction using OpenCV

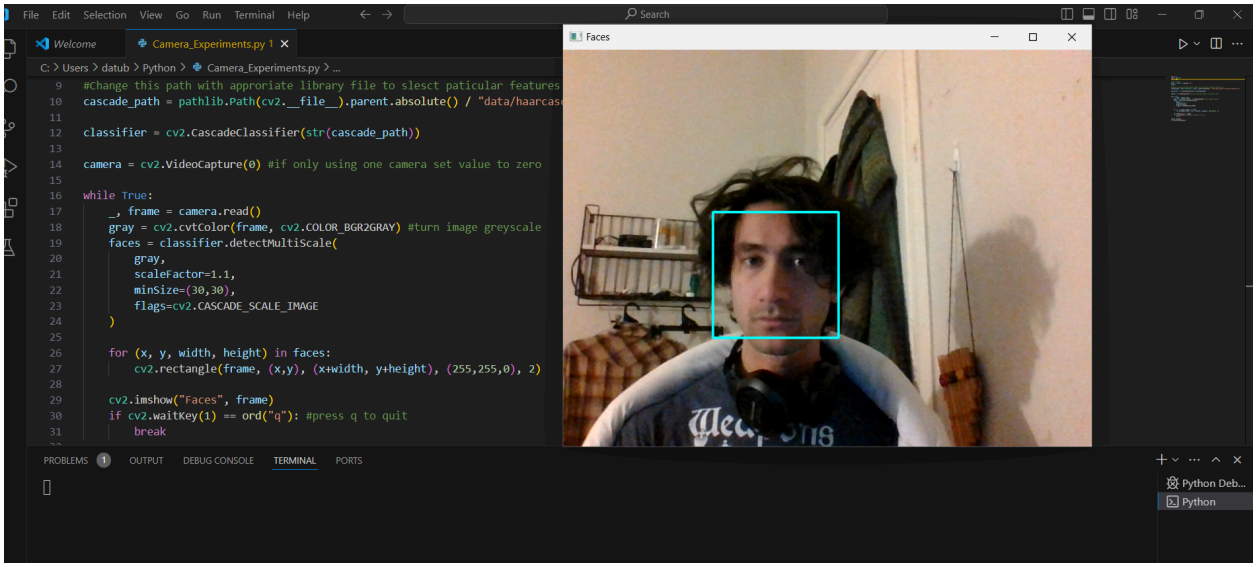
Development Environment

- Windows
- Anaconda Navigator
- VS Code

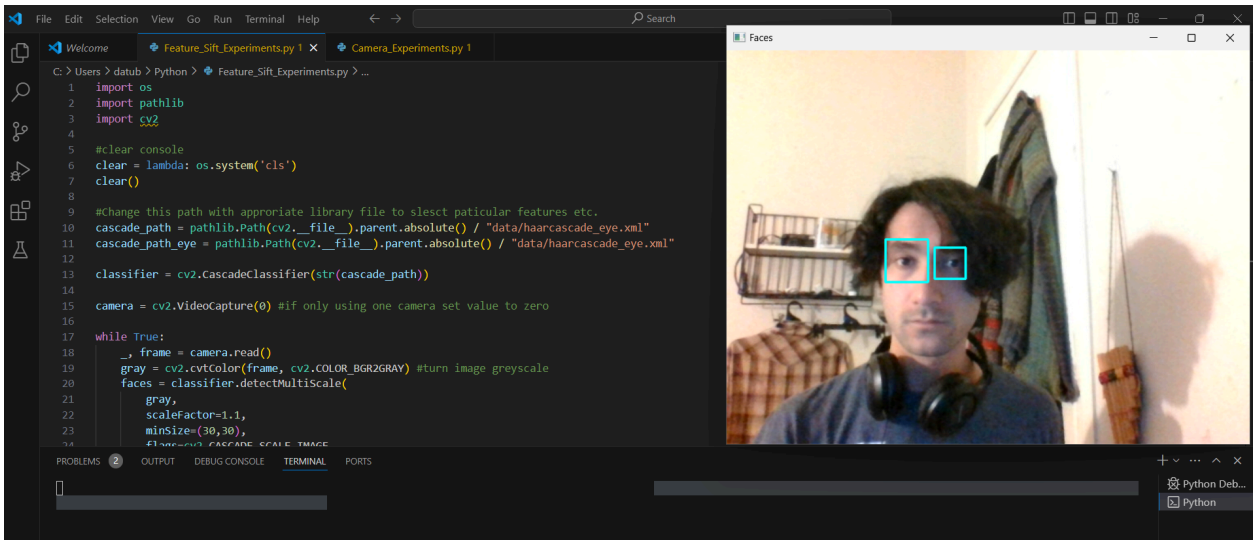
Hardware Requirements

- Laptop with Webcam

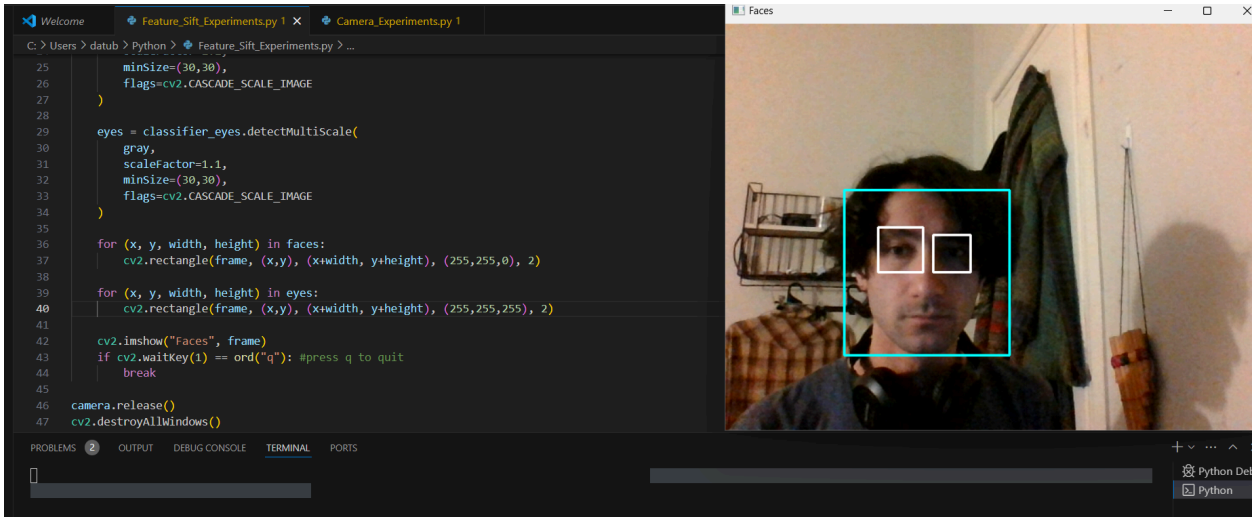
Detecting Frontal Face



Detecting Eye



Detecting Eye and Frontal Face



Installing OpenCV dependencies and packages

1. Install Open CV in cmd (if not already installed)

```
PS C:\Users\datub> pip install opencv-python
Defaulting to user installation because normal site-packages is not writeable
Collecting opencv-python
  Downloading opencv_python-4.9.0.80-cp37-abi3-win_amd64.whl.metadata (20 kB)
Requirement already satisfied: numpy>=1.21.2 in c:\programdata\anaconda3\lib\site-packages (from opencv-python) (1.26.4)
Downloading opencv_python-4.9.0.80-cp37-abi3-win_amd64.whl (38.6 MB)
   ---------------------------------------- 38.6/38.6 MB 10.4 MB/s eta 0:00:00
Installing collected packages: opencv-python
Successfully installed opencv-python-4.9.0.80
PS C:\Users\datub>
```

2. Assign cascade path, here I have chosen ‘frontalface’ from the CV2 Library

```
C: > Users > datub > Python > Camera_Experiments.py > ...
1   import os
2   import pathlib
3   import cv2
4
5   #clear console
6   clear = lambda: os.system('cls')
7   clear()
8
9   cascade_path = pathlib.Path(cv2.__file__).parent.absolute() / "data/haarcascade_frontalface_default.xml"
10  print(cascade_path)
```

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

C:\Users\datub\AppData\Roaming\Python\Python311\site-packages\cv2\data\haarcascade_frontalface_default.xml
PS C:\Users\datub>
```

3. Create the Classifier and assign a camera or video source, I have used the default laptop camera. You could also supply a string as the argument to supply a path to a video file.

```
classifier = cv2.CascadeClassifier(str(cascade_path))

camera = cv2.VideoCapture(0) #if only using one camera set value to zero
```

4. Main Loop (breaks if user presses ‘q’)

```
15
16  while True:
17      _, frame = camera.read()
18      gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #turn image greyscale
19      faces = classifier.detectMultiScale(
20          gray,
21          scaleFactor=1.1,
22          minSize=(30,30),
23          flags=cv2.CASCADE_SCALE_IMAGE
24      )
25
26      for (x, y, width, height) in faces:
27          cv2.rectangle(frame, (x,y), (x+width, y+height), (255,255,0), 2)
28
29      cv2.imshow("Faces", frame)
30      if cv2.waitKey(1) == ord("q"): #press q to quit
31          break
32
```

5. When loop is exited camera is released, windows are closed

```
32
33  camera.release()
34  cv2.destroyAllWindows()
35
```

Full Code

```
import os
import pathlib
import cv2

#clear console
clear = lambda: os.system('cls')
clear()

#Change this path with appropriate library file to select particular features etc.
cascade_path = pathlib.Path(cv2.__file__).parent.absolute() / "data/haarcascade_frontalface_default.xml"

classifier = cv2.CascadeClassifier(str(cascade_path))

camera = cv2.VideoCapture(0) #if only using one camera set value to zero

while True:
    _, frame = camera.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #turn image greyscale
    faces = classifier.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minSize=(30,30),
        flags=cv2.CASCADE_SCALE_IMAGE
    )

    for (x, y, width, height) in faces:
        cv2.rectangle(frame, (x,y), (x+width, y+height), (255,255,0), 2)

    cv2.imshow("Faces", frame)
    if cv2.waitKey(1) == ord("q"): #press q to quit
        break

camera.release()
cv2.destroyAllWindows()
```

UX/UI Research

User Persona 1

Name: Brian G. Smith
Age: 32
Occupation: human resource at Senior HR advisors
Income: $101,000

Bio:

Brian works as a human resources Senior HR advisor. Why brian is interested in facial recognition is that office attendance has been very low after covid.

Wants:

- Track staff attendance
- Easy to use

Frustrations:

- Hates when technology doesn't work
- Office attendance is very low

User Persona 2

Name: Charlie Corby
Age: 18
Occupation: checkout operator /student
Income: minimum wage

Bio:

Charlie is an 18 year old student at Kāpiti College, what charlie would like to see with the facial recognition is to be able to track his movement and displays the location he has been and to help him to get out of trouble.

Wants:

- High-quality Image
- Quick and reliable

Frustrations:

- Slow and unreliable

User Persona 3

Name: Edward Cronin
Age: 33
Occupation:Teacher
Income: $59,027

Bio:

Edward Cronin is a secondary school teacher at Tawa College. who wants to be able to track students' attendance while using facial recognition. The main reason why he wants this is because the attendance in his class has been very low, this has mainly transpired after Covid where students attending class has been low.

Wants:

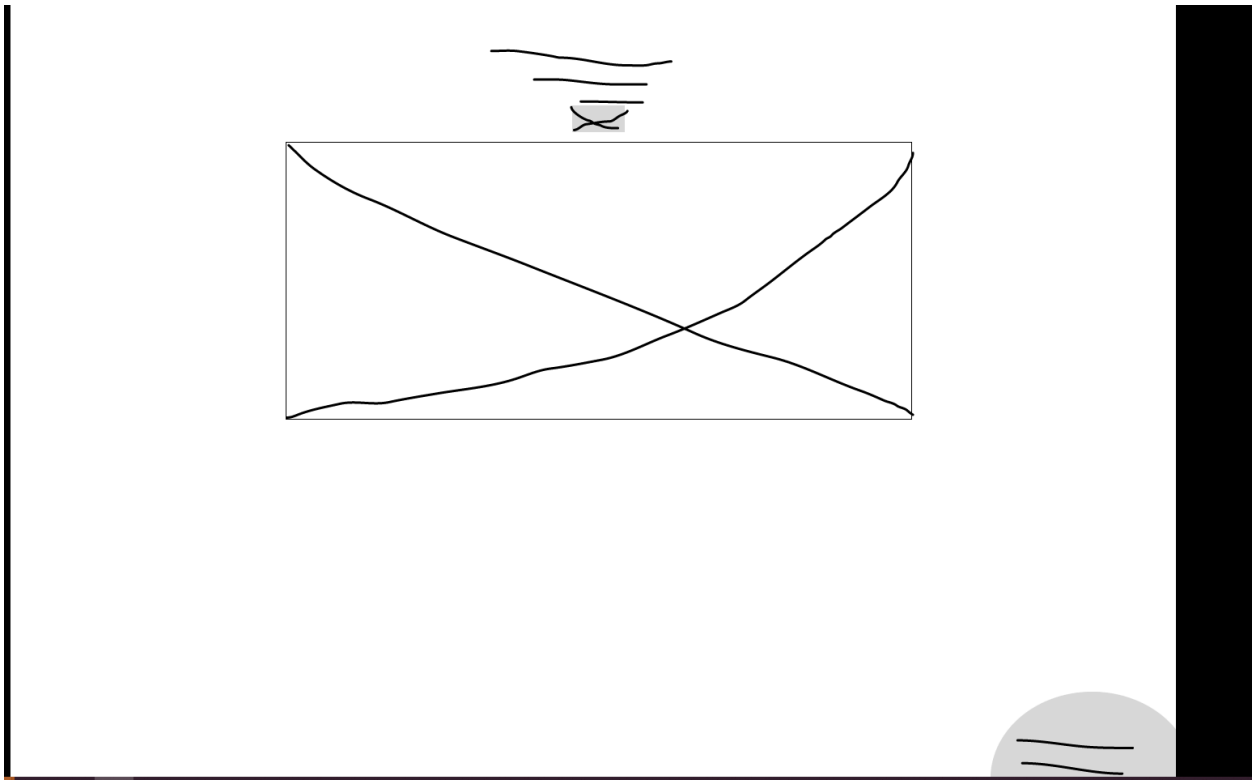
- Reliability.
- Easy to use.

Frustrations:

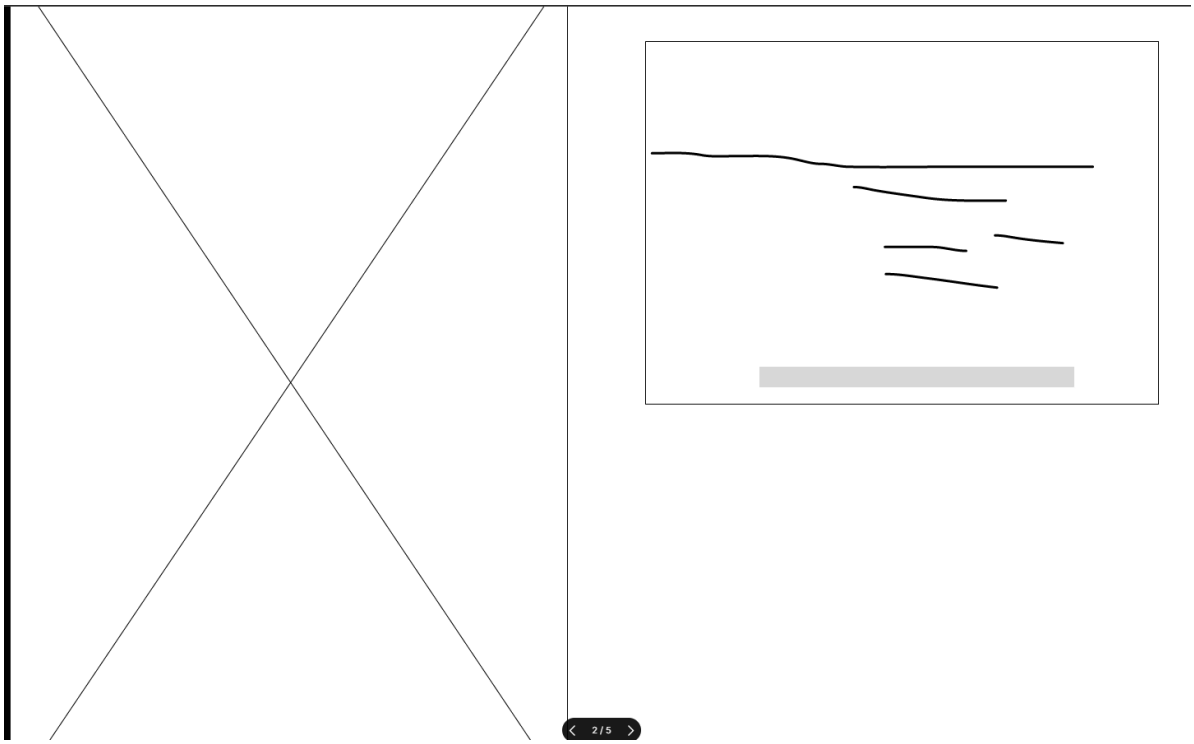
- Slow technology
- Less students attending

GUI LoFi:

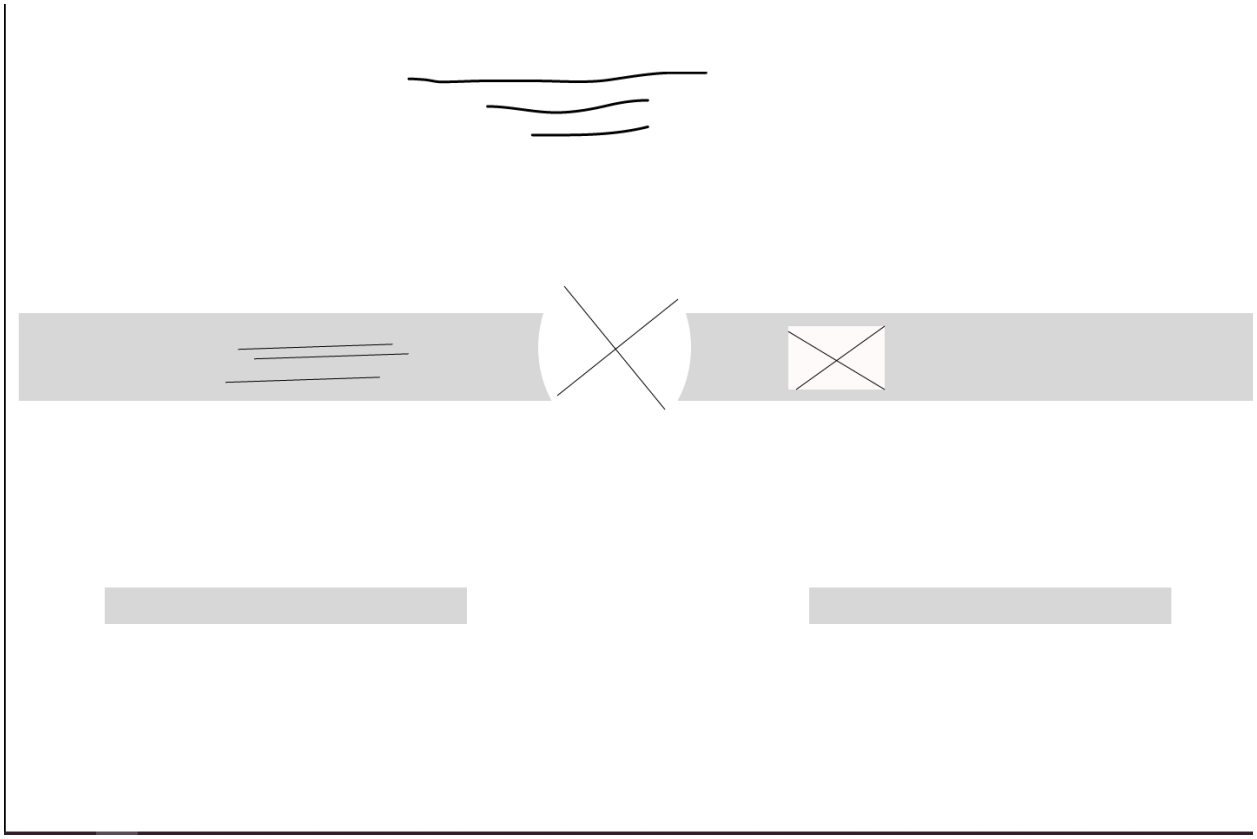
Home Screen:



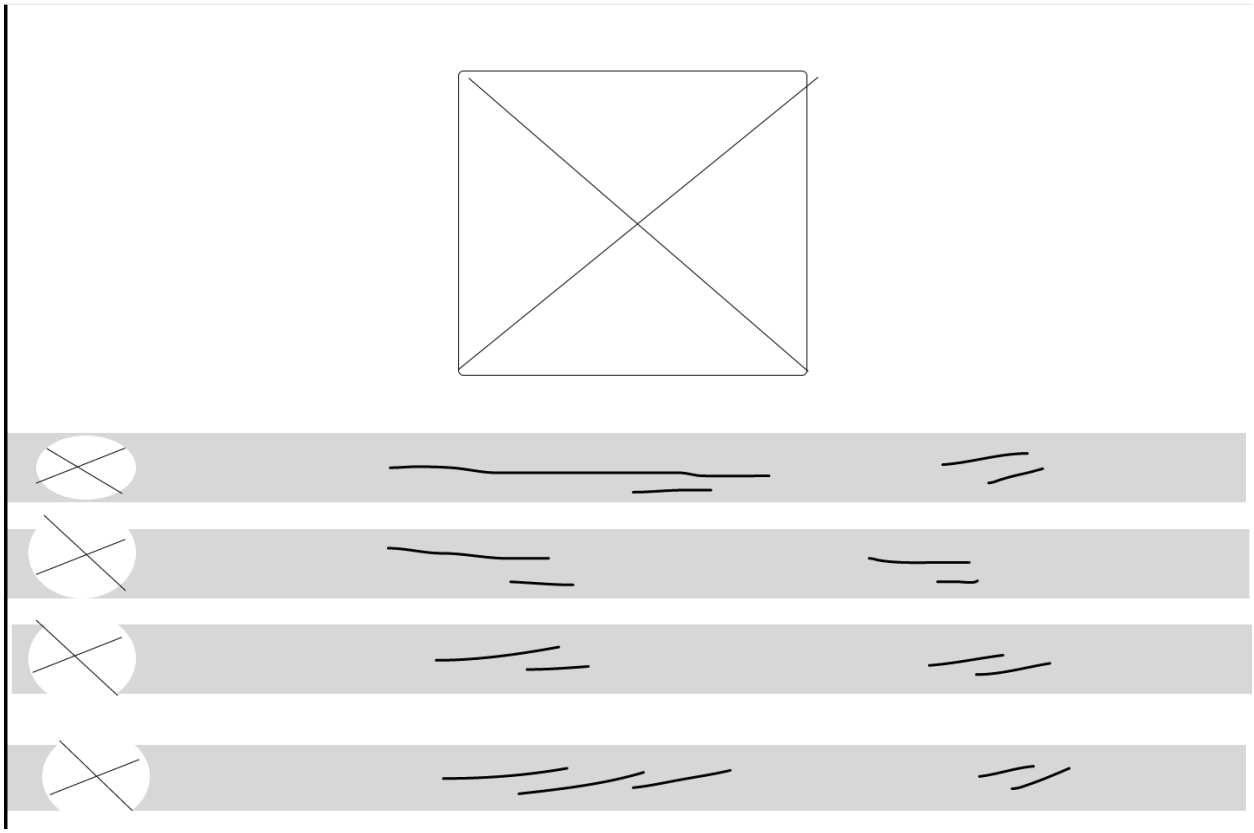
Add student :



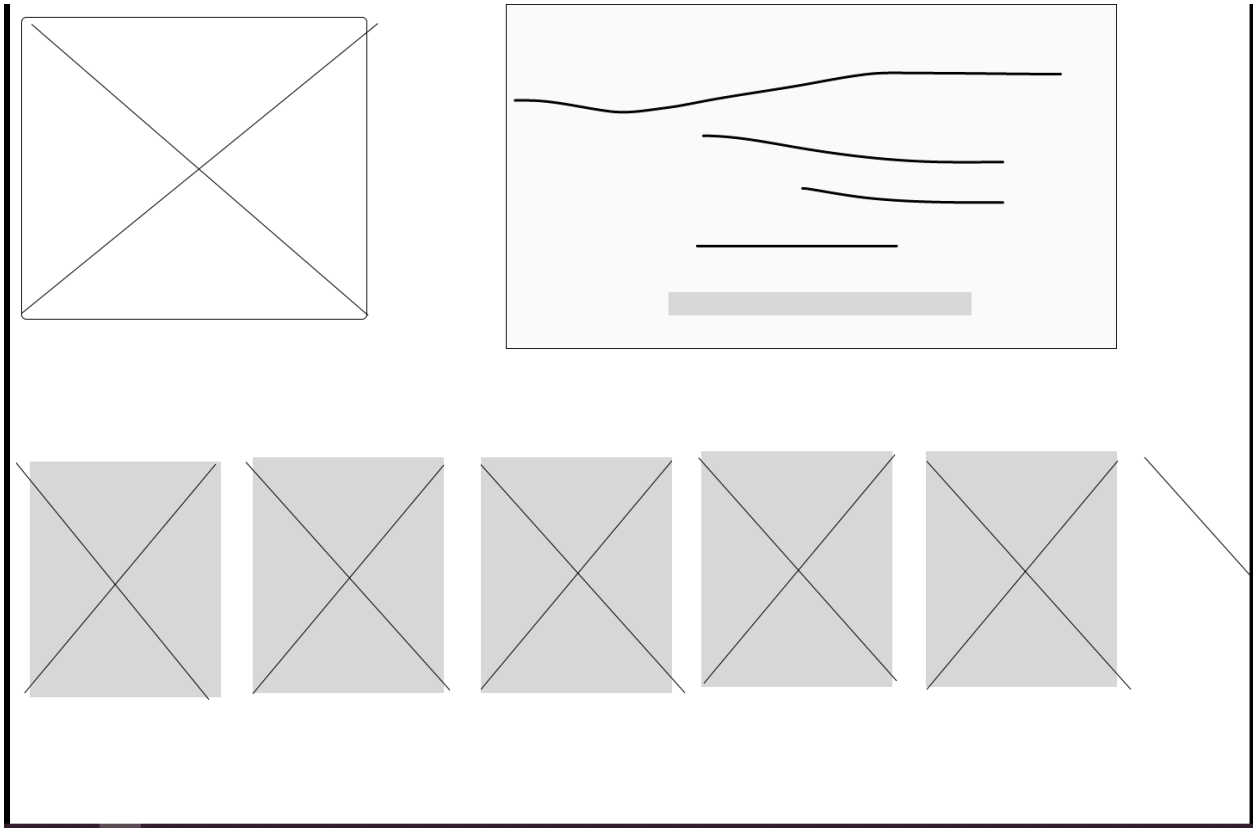
Attendance Layout:



Number of student:



Particular student details:



User testing:

Question1:	1. Clarity of Navigation: How intuitive was it to navigate through different sections of the attendance app? Were you able to easily find the features you were looking for?
Question2::	Ease of Use: Did you find the interface straightforward to interact with? Were there any specific actions or tasks that were particularly difficult or confusing?
Question3:	Visual Appeal: What are your initial impressions of the

	design aesthetics? Did the layout and visual elements enhance or detract from your experience?
Question4:	Functionality Feedback: Were all the necessary features present to fulfill your attendance tracking needs? Were there any features you expected to see that were missing?

User 1:

Navigation was straightforward, I easily found where to mark attendance."	"It took me a bit to figure out where everything was, but once I got the hang of it, it was smooth sailing."
The interface was pretty easy to use overall, but I struggled a bit with adding new attendees."	"I found it very intuitive, everything seemed to be where I expected it to be
The design was clean and modern, which I liked. However, some icons were a bit small and hard to tap accurately."	The colors were pleasing, but the font choice could be improved for better readability."

It had all the features I needed, but I wish there was an option to export attendance data."	I didn't expect to see a calendar view, but it was a pleasant surprise. Very useful!"
Interactions felt smooth and responsive, no issues there	I had trouble deleting a mistaken entry, it wasn't immediately obvious how to do

User 2:

Navigation was straightforward, I easily found where to mark attendance."	"It took me a bit to figure out where everything was, but once I got the hang of it, it was smooth sailing."
the app was simple to use, I didn't encounter any difficulties completing tasks like marking attendance or viewing reports."	"I found it very intuitive, everything seemed to be where I expected it to be
The design was a bit bland for my taste, adding some more vibrant colors could make it more engaging."	The colors were pleasing, but the font choice could be improved for better readability."
I liked the feature that showed attendance trends over time, it's a great way to monitor progress	"While most features were present, I

	struggled to find an option for setting up recurring attendance schedules."
Interactions were smooth overall, but there was a slight delay when loading attendance records	"The swipe gestures for navigating between dates felt natural and intuitive."

User 3:

I got lost a couple of times trying to find certain functions, maybe clearer labels could help.""	The navigation was intuitive, and I could easily switch between different views."
Adding new attendees was a breeze, but I had trouble editing existing entries	The app was straightforward to use, even for someone like me who isn't tech-savvy."
The design felt outdated, a more modern look would make the app more appealing."	"I liked the minimalist design, it felt clean and uncluttered."
It had all the features I needed, but I wish there was an option to export attendance data."	I didn't expect to see a calendar view, but it was a pleasant surprise. Very useful!"

Interactions felt smooth and responsive, no issues there	I had trouble deleting a mistaken entry, it wasn't immediately obvious how to do

User 4:

Navigation was straightforward, I easily found where to mark attendance."	"It took me a bit to figure out where everything was, but once I got the hang of it, it was smooth sailing."
The interface was pretty easy to use overall, but I struggled a bit with adding new attendees."	"I found it very intuitive, everything seemed to be where I expected it to be
The design was clean and modern, which I liked. However, some icons were a bit small and hard to tap accurately."	The colors were pleasing, but the font choice could be improved for better readability."
"The app lacked a search function, which would be useful for finding specific attendees quickly."	I appreciated the option to customize attendance categories, it added flexibility to the app."
The app responded quickly to my inputs, which made the experience seamless."	I found it confusing when long-pressing on an attendee didn't bring up additional options."

User 5:

"Navigating the app was straightforward, but I wish there was a home button for quicker access to the main screen.	"The navigation was clear and logical, I didn't have any trouble finding my way around."
The app was very user-friendly, I was able to complete tasks without any prior instruction.""	"I loved the design aesthetics, especially the use of animations to provide feedback on actions."
The design was clean and modern, which I liked. However, some icons were a bit small	The colors were pleasing, but the font choice could be improved for better readability."

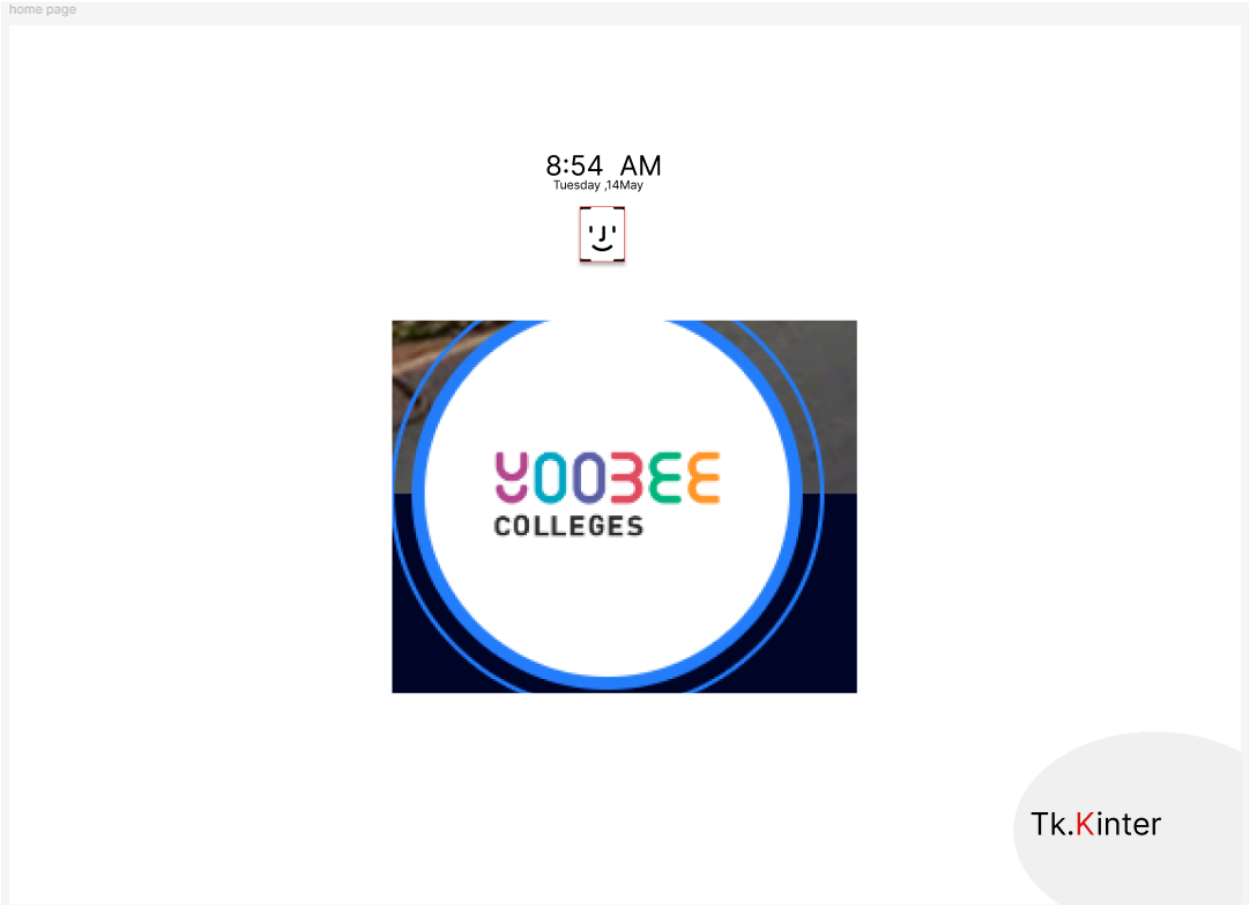
and hard to tap accurately."	
"I was pleasantly surprised by the option to set up automated reminders for attendance submissions."	"I encountered a bug when trying to import attendee data from a spreadsheet, it would be great if this could be fixed."
The app responded quickly to my inputs, which made the experience seamless."	I found it confusing when long-pressing on an attendee didn't bring up additional options."

Overall summary:

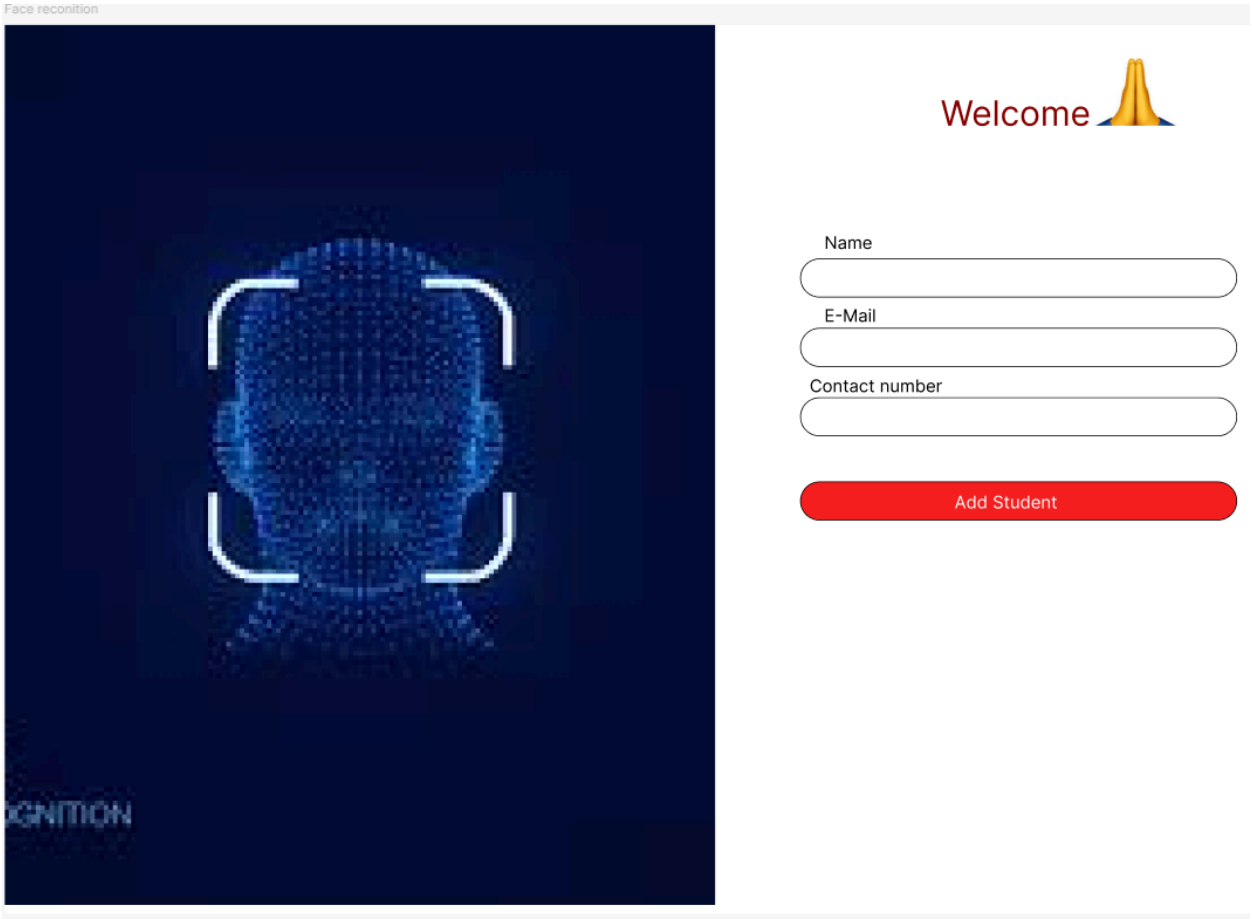
Overall, users found the navigation of the attendance app to be clear and intuitive. While a few minor suggestions for improvement were made, such as adding a home button for quicker access, the general consensus was positive, with users finding it easy to navigate through different sections and features.

GUI HiFi:

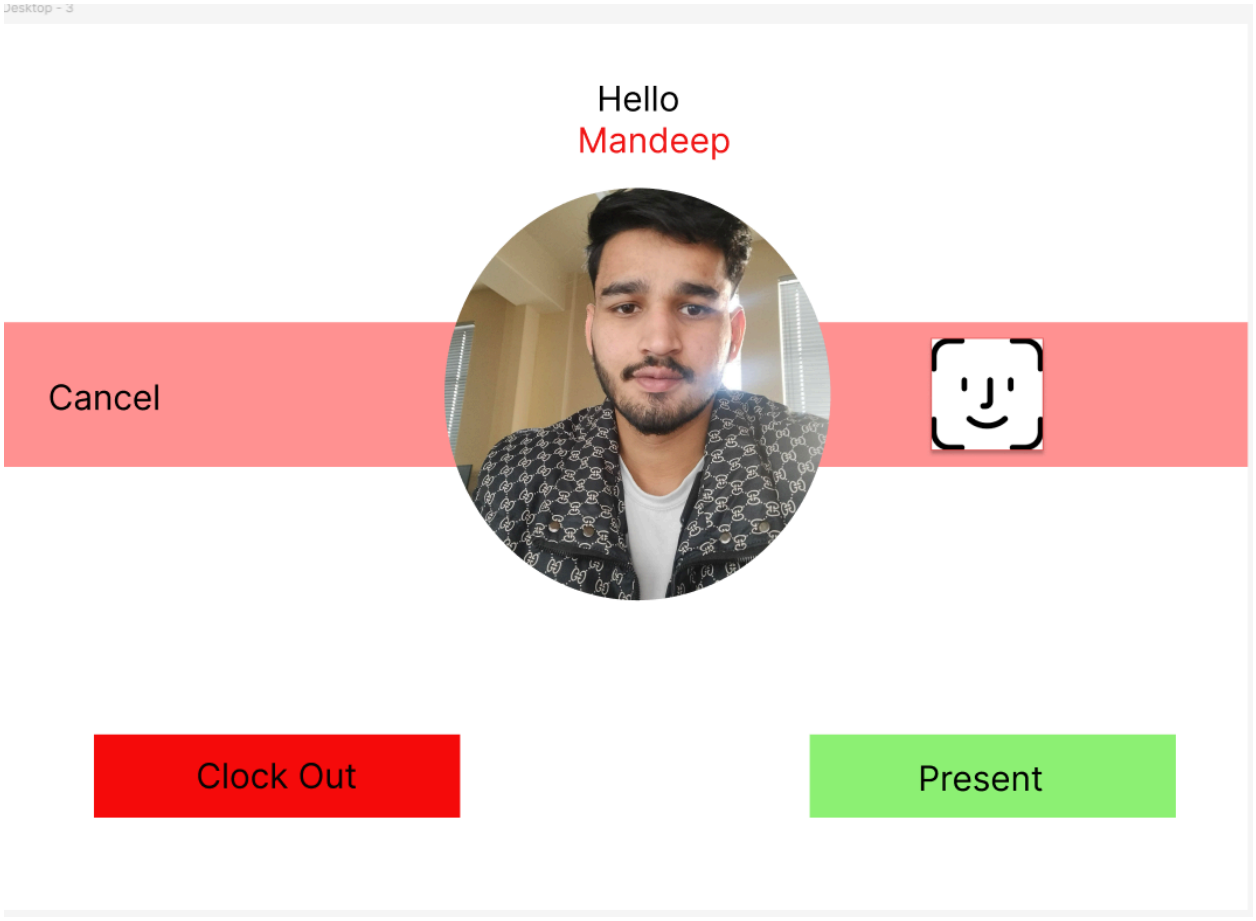
Home Screen:



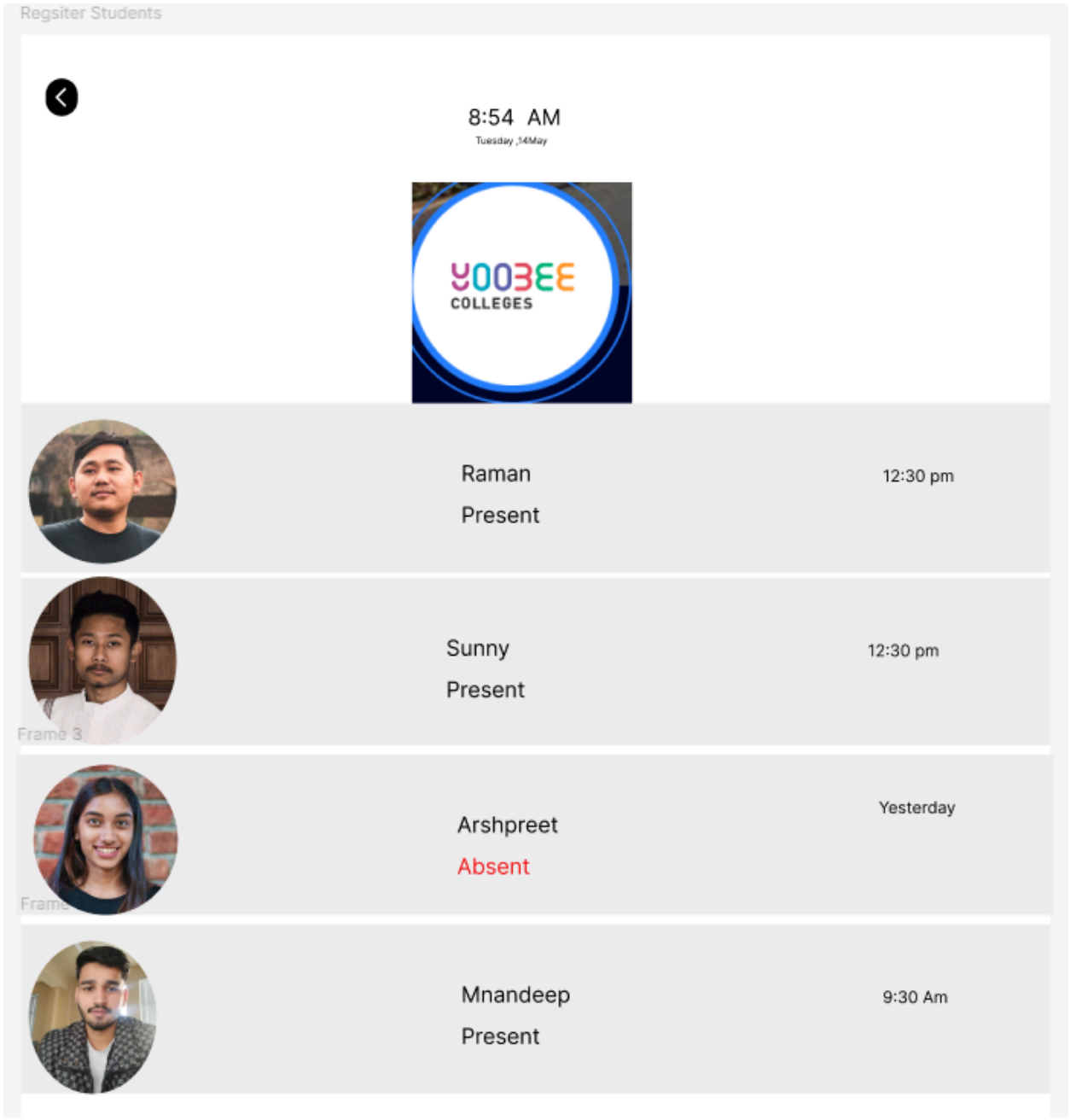
Adding Student:



Detecting Student:



Class Attendance:



Name: Mandeep Singh

E-Mail: Mandeep Singh@gmail .com

Contact :2540572654

Monday

Tuesday

Wednesday

Thrusday

Friday

Saturaday

Sunday

References

1. Klein, M., Sosu, E. M., & Dare, S. (2022). School Absenteeism and Academic Achievement: Does the Reason for Absence Matter? *AERA Open*, *8*(1), 233285842110711. https://doi.org/10.1177/23328584211071115

2. https://www.rnz.co.nz/news/political/513250/sickness-related-school-absences-to-be-targeted-under-government-plan

Images:

1. Mikael Cho. (2013, May 29). Beautiful Free Images & Pictures | Unsplash. https://unsplash.com/

2. Figma. (2016, September 27). *Figma*. https://www.figma.com/files/team/1215030665848135846/recents-and-sharing/recently-viewed?fuid=1215030654803861773