SPL-2 Project Report

Automated Attendance System

An approach for tracking attendance

Submitted by

Mussammat Maimuna Faria

(BSSE Roll No: **1222,** BSSE Session: **2019-20**)

Monayem Sarker

(BSSE Roll No: **1228,** BSSE Session: **2019-20**)

Supervised by

Dr. Sumon Ahmed

Associate Professor, IIT-DU

Institute of Information Technology

University of Dhaka

[29-05-2023]

Signature Page

Project: Automated Attendance System

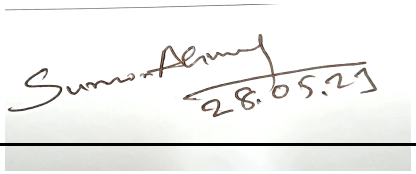
Author: Mussammat Maimuna Faria, Monayem Sarker

Submitted: 29.05.2023

Supervised By: Dr. Sumon Ahmed

Associate Professor, Institute of Information Technology,
University of Dhaka

Supervisor's Approval:



Letter of Transmittal

SPL-II Committee,
Institute of Information Technology,
University of Dhaka

Subject: Submission of the final report on Automated Attendance System**.**

Dear Sir,
With respect, we have attached our report on the "Automated Attendance System, a web application" here. We adhered to the guidelines you provided in class when preparing this report. The relevant topics that we studied throughout the course have been applied into practice.
We did our best to prepare an acceptable report, despite the fact that it may have some flaws. We sincerely hope that you will appreciate our effort and pardon our small mistakes.

Sincerely yours,

Mussammat Maimuna Faria (BSSE 1222)
Monayem Sarker (BSSE 1228)
Institute of Information Technology,
University of Dhaka

Acknowledgement

We would like to express our sincere gratitude to Dr. Sumon Ahmed Sir, Associate Professor, Institute of Information Technology, University of Dhaka, who is our Software Project Lab 2 supervisor for giving us the chance to complete this project and for his invaluable guidance throughout the entire course. We have been greatly inspired by his vision, sincerity, idea, and motivation. It was a real honor and privilege to work under his direction.

Table of contents

Index of Figures

1. Introduction

In today's fast-paced world, efficiently managing attendance records is crucial for organizations across various sectors. Attendance records must be managed effectively for universities and other educational institutions to effectively track student participation, ensure adherence to academic rules, and create a punctuality and engagement culture With the development of technology. An automatic attendance system equipped with a barcode scanner has become an effective and practical way to speed up this important task.

In a university setting, an automated attendance system using a barcode scanner makes use of modern technology to accurately and efficiently collect and handle attendance data. This system eliminates the need for time-consuming manual data entry, lessens administrative workload, and improves overall efficiency by replacing automated processes for traditional manual techniques.

It works by reading the distinctive barcodes that are given to each student using barcode scanners. On student ID cards these barcodes are printed. The system immediately gathers the information when students scan their IDs or barcodes for scanning, and it stores the attendance information in a database.

The barcode scanner offers universities a number of advantages. Compared to manual approaches, it lowers errors and saves time. Since scanning is quick and precise, manual data entering is not necessary. As a result, tracking attendance is simpler and less error-prone.

Universities can also compile thorough attendance reports using the system and examine attendance patterns. Universities can create targeted strategies to boost student performance and improve the overall campus experience by monitoring student behavior and engagement levels.

In conclusion, institutions may effectively and conveniently manage student attendance by using automated attendance systems with barcode scanners. They promote transparency, eliminate errors, save time, and offer useful information for making decisions. For universities that want to simplify processes and promote a positive learning environment, implementing such a system is crucial.

2. Background Study

Technological advancements have paved the path for the creation of automatic attendance systems that use a variety of identification methods, including barcode scanning. Because of their capacity to simplify the attendance monitoring process, minimize manual techniques and provide rapid access to attendance data, these systems have grown in popularity. To understand the importance of automated attendance systems, it is necessary to investigate their history and the reasons for their acceptance.

2.1 Limitations of Manual Attendance Systems

Manual attendance methods have long been the norm in businesses and educational institutions. However, they have certain limitations. Handwritten sign-in sheets or roll calls are used in manual processes, which are prone to errors, time consuming and inaccuracies during data entry. This can result in errors in attendance records.

Furthermore, manual methods require a huge amount of paperwork, resulting in extra costs. Physical attendance records can be time-consuming and resource-intensive to get, store, and manage.

Manual systems also have problems about the confidentiality and security of attendance data. Unauthorized access or tampering with attendance records may compromise their integrity, perhaps resulting in privacy violations or academic fraud.

2.3 Advancements in Automated Attendance Systems

As technology has advanced, automated attendance systems have evolved as an alternative to the limitations of manual approaches. To simplify the attendance tracking process, these systems make use of numerous identification technologies, such as barcode scanning.

ID cards printed with unique barcodes issued to people are used in automated attendance systems. Individuals can rapidly record their presence by scanning their ID cards at selected checkpoints, reducing the need for manual sign-in procedures. The barcode scanners capture the information printed in the barcode and record attendance statistics in real-time in a centralized database. Also it can check the proxy by using a camera.

2.4 Benefits of Automated Attendance Systems

Over manual techniques, automated attendance systems provide various advantages. For starters, they reduce the likelihood of attendance record errors and inaccuracies. Another significant advantage of automated systems is real-time data accessibility. Administrators may rapidly access up-to-date attendance data, allowing them to make informed judgments and take appropriate measures based on accurate attendance records.

Furthermore, automated attendance systems create detailed attendance reports that enable managers to study attendance patterns, detect trends, and make data-driven choices. Integration capabilities allow for smooth integration with other systems, such as payroll or student information.

3. Project Description

We can divide our project into some parts. Here is the description of the parts in short:

Account Management is divided into two sub-modules: Authentication and Updating Information. Authentication involves four users: Admin, Director, Teachers and Students. Users can register, log in, log out and reset passwords any time they want. Updating Information involves adding information and changing passwords. Students must include certain further details, such as Name, Phone number, Photo, Registration Number, Class Roll, Gender, Date of Birth, Semester, Phone number, Current address, Permanent address. Verification involves the admin performing all forms of verifications. Course Plan involves uploading the teacher's id with their associate courses to the system at the beginning of the semester. Also the admin should add the courses with a semester number.

Each student at IIT has a smart card with a barcode with their registration number. To implement this system, the teacher must insert the course code and students will have to scan their cards. When a teacher gives the course code, the class will start. Students must upload their image when creating an account and the camera in the room will match the uploaded photo with the student's face. If any proxy is detected, the teacher will receive a notification and the system will detect and prevent the proxy.

Otherwise, the information will be stored in the database. Notifications will be provided to the faculty member's dashboard if any student misses three classes in a row, and if the attendance percentage of any student is below 60%, the teacher and student both will receive a notification. The system will also calculate the percentage of each student's attendance, generate reports, and approve applications. The system will inform the head of the academic committee, director sir, who has the authority to grant applications, if any student's total attendance rate falls below 60%. The admin will act in accordance with the notice that the director will send to them and provide the students with instructions.

4. Project Implementation

4.1 Technology Stack

Automated Attendance System has been developed using the following tech stack:
FrontEnd: ReactJS
BackEnd: NodeJS
Database: MySQL
REST API: Axios (FrontEnd), ExpressJS (BackEnd)

4.2 Front End

4.2.1 Frontend Files

Pages were created to separate the front-end files. The App.js file contained the routes for the pages.
Based on the functions of the pages, other categories were created. The Navbar, Notifications, and Reset Passwords were presented as modules because they were used frequently for every user.

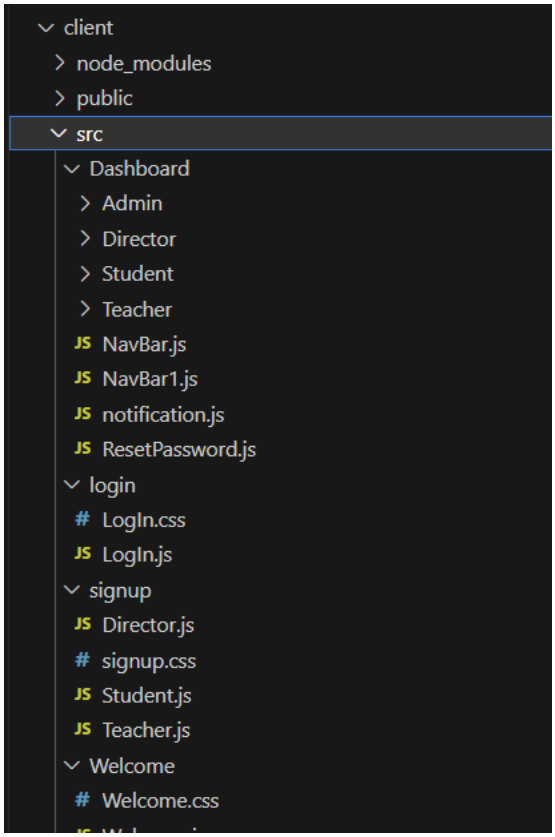


Figure 4.1: *Code Snippet - Frontend Files*

4.2.2 Authentication

Before accessing the page's content , authentication was validated on each page. After logging in, the jsonwebtoken was generated using the user's email address, password and verification status, saved as a session key, and then used to verify the user's identity. The session key becomes invalid whenever somebody logs out, requiring the user to log in again.

```
Axios.post('http://localhost:12280/user/login',
  {
    email: email,
    password: password,

  }).then((response) => {
    if (response.data.error)
      toast.error(response.data.error);
    else {
      localStorage.setItem("accessToken", response.data);

      Axios.get("http://localhost:12280/auth/", {
        headers: {
          accessToken: localStorage.getItem("accessToken")

        }
      }).then((response) => {
        if (response.data.error) {
          toast.error("Wrong Email or Password");
        } else {
          console.log(response.data);
          if(response.data.verificationStatus==1)
            {navigate(`/${response.data.userType}/${response.data.email}`);}
            else{
              toast.error("Not Verified")
            }
```

S OUTPUT TERMINAL DEBUG CONSOLE

Figure 4.2: *Code Snippet - Authentication*

4.2.3 HTTP requests

The HTTP requests of the frontend have been handled by Axios. The useEffect, useParams and useState hooks of react have been used to facilitate the requests.

```
const handleAddTeacherToCourse = (event) => {
  event.preventDefault();

  const newCourseTeacher = {
    courseCode1,
    teacherEmail
  };
  Axios.post('http://localhost:12280/teacherCourse/add', newCourseTeacher)
    .then((response) => {
      console.log(response.data)
      toast.success("Teacher Added");
    }).catch((error) => {
      toast.error('Error occurred');
    })


}
```

```
47
48      const handleEnterKeyPress = (e) => {
49          if (e.key === 'Enter') {
50            e.preventDefault();
51            handleUpdateDigitalAttendance();
52            setBarcode('');
53          }
54        };
55
56        const handleUpdateDigitalAttendance = () => {
57          const attendanceInfo = {
58              courseCode1,
59              barcode
60          }
61
62          axios.put(`http://localhost:12280/attendance/scanCount`, attendanceInfo).then(response => {
63              toast.error(response.data.error);
64          })
65          .catch(error => {
66              toast.error(error);
67          });
68        };
69
```

Figure 4.3: *Code Snippet - HTTP requests*

4.2.4 Frontend Libraries

Axios library has been used to make HTTP requests. React-Bootstrap has been used to add styling to the pages.

4.3 Backend

There are two backends in the system: NodeJS and Python.

4.3.1 Backend Files

In NodeJS, there are three types of core folder:

- Models: For defining the database table structures.
- Controllers: For defining all the functionalities of the specific models.
- Routes: For defining routes for the routes of every individual controller.

To access any model through any request the flow : Router->Controller->Model

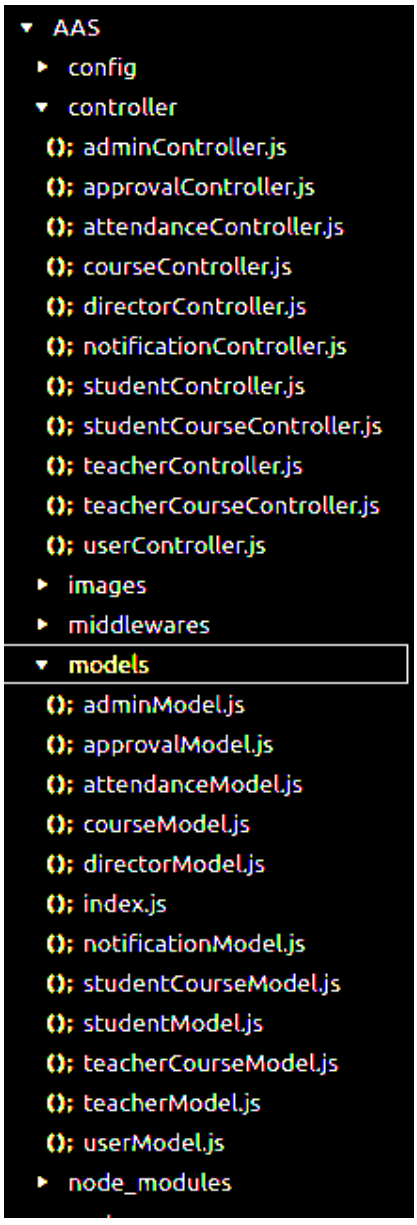


Figure 4.4: *Code Snippet - Backend Files*

4.3.2 Handling HTTP requests

Express has been used to handle HTTP requests in the backend.

```
const express = require("express");
const router = express.Router();
```

```
const routerUser = require('./routes/userRoute.js')
app.use('/user', routerUser);
```

```
router.post('/add', userController.addUser);
router.delete('/delete', userController.deleteUser)
router.get('/all', userController.getAllUser)
router.get('/one', userController.getOneUser)
```

Figure 4.5: *Code Snippet - Handling Requests using HTTP*

In python, FastAPI has been used to handle the requests.

```
app=FastAPI()
```

```
@app.on_event('startup')
async def database_connect():
    await database.connect()
```

Figure 4.6: *Code Snippet - Handling Requests using HTTP*

4.3.3 Backend Libraries

NodeJS:

1. Sequelize: Used to define the Database schema and handle Database query and operations.

```
module.exports = (sequelize, DataTypes) => {
    const Attendance = sequelize.define('Attendance', {
        status: {
            type: DataTypes.BOOLEAN,
            allowNull: false,
            defaultValue: false
        },
        createdAt: {
            type: DataTypes.DATEONLY,
            defaultValue: DataTypes.NOW,
        },
    })

    Attendance.associate = (models) => {
        Attendance.belongsTo(models.StudentCourse, {
            onDelete: 'CASCADE',
        });
    }

    return Attendance
}
```

Figure 4.7: *Code Snippet - Database model using Sequelize*

2. Multer: Used for uploading images, saving them in a static public route, and inserting the route to the database.
3. Bcrypt: Used to hash the password of users.
4. Jsonwebtoken: Used for authentication.

```
const userLogIn = async (req, res) => {
    const { email, password } = req.body;
    const user = await db.User.findOne({ where: { email: email } });
    if (user) {
        bcrypt.compare(password, user.password).then((match) => {
            if (match) {
                const accessToken = sign({
                    email: user.email,
                    userType: user.userType
                }, "tokenTry");
                res.json(accessToken);
            } else {
                res.json({ error: "Wrong email or password" })
            }
        })
    } else {
        res.json({ error: "No user found" })
    }
}
```

Figure 4.8: *Code Snippet - Using Bcrypt and jsonwebtoken to generate AccessToken*

Python:

1. OpenCV: Used for processing videos and images.

```python
def faceMatching(path):
    try:
        fixed_image = fr.load_image_file("../images/studentImages/"+path)


        incoming_image = fr.load_image_file("images/img.jpg")

        fixed_encodings = fr.face_encodings(fixed_image)
        if len(fixed_encodings) > 0:
            fixed_encoding = fixed_encodings[0]


        incoming_encodings = fr.face_encodings(incoming_image)
        if len(incoming_encodings) > 0:
            incoming_encoding = incoming_encodings[0]


        results = fr.compare_faces([fixed_encoding], incoming_encoding)
        distance = fr.face_distance([fixed_encoding], incoming_encoding)[0]
        similarity = (1 - distance) * 100
        print(similarity)

        is_match = distance < 0.5
        return is_match
    except:
        return False
```

Figure 4.9: *Code Snippet - Using Face_recognition to process matching*

```python
    vcap = cv2.VideoCapture("http://192.168.107.37:4747/video")

    ret, frame = vcap.read()
    cv2.imwrite("./images/img.jpg", frame)
    return faceMatching(path)
except cv2.error:
    return False
```

Figure 4.10: *Code Snippet - open-cv to capture frame*

2. PDFkit: Used for pdf generation
3. Face_recognition: Used to match face.

5. User Manual

The welcome page is given below, where one can view the details and features of our project. From here, one can get the sign-in and registration option.

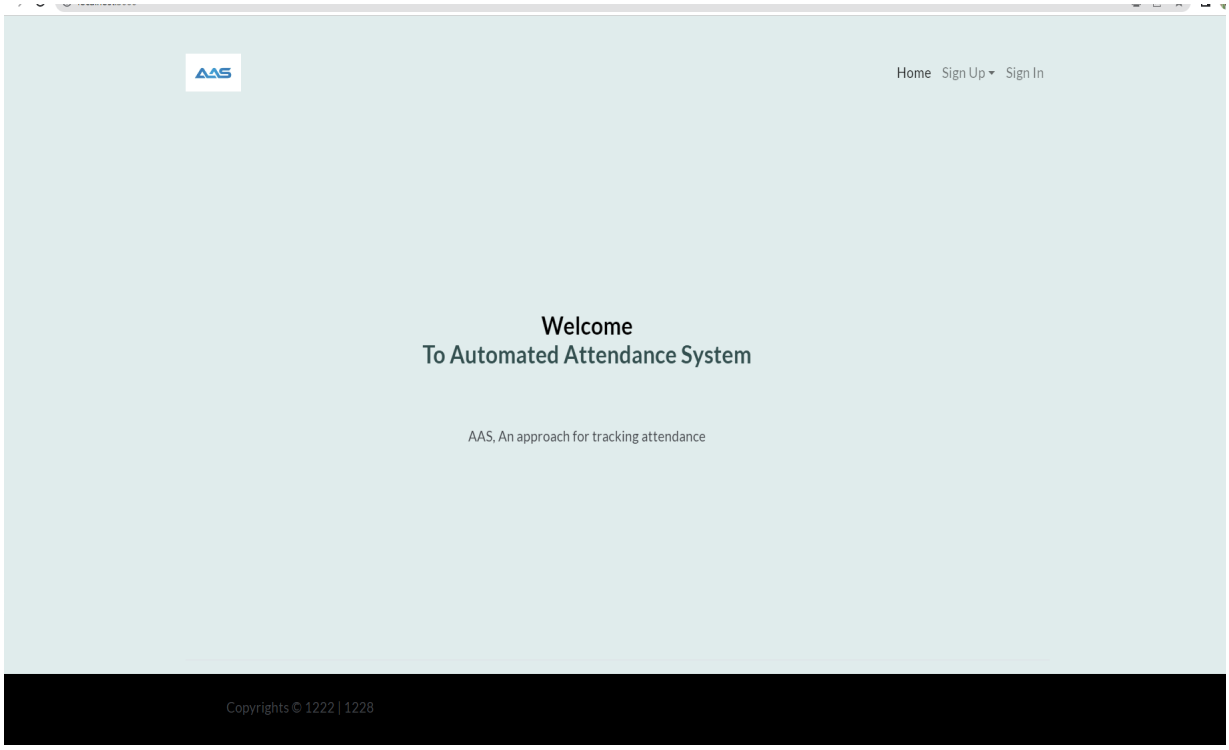


Figure 5.1: *Welcome Page*

Users must sign in using their email address and password on this page in order to access their dashboard.

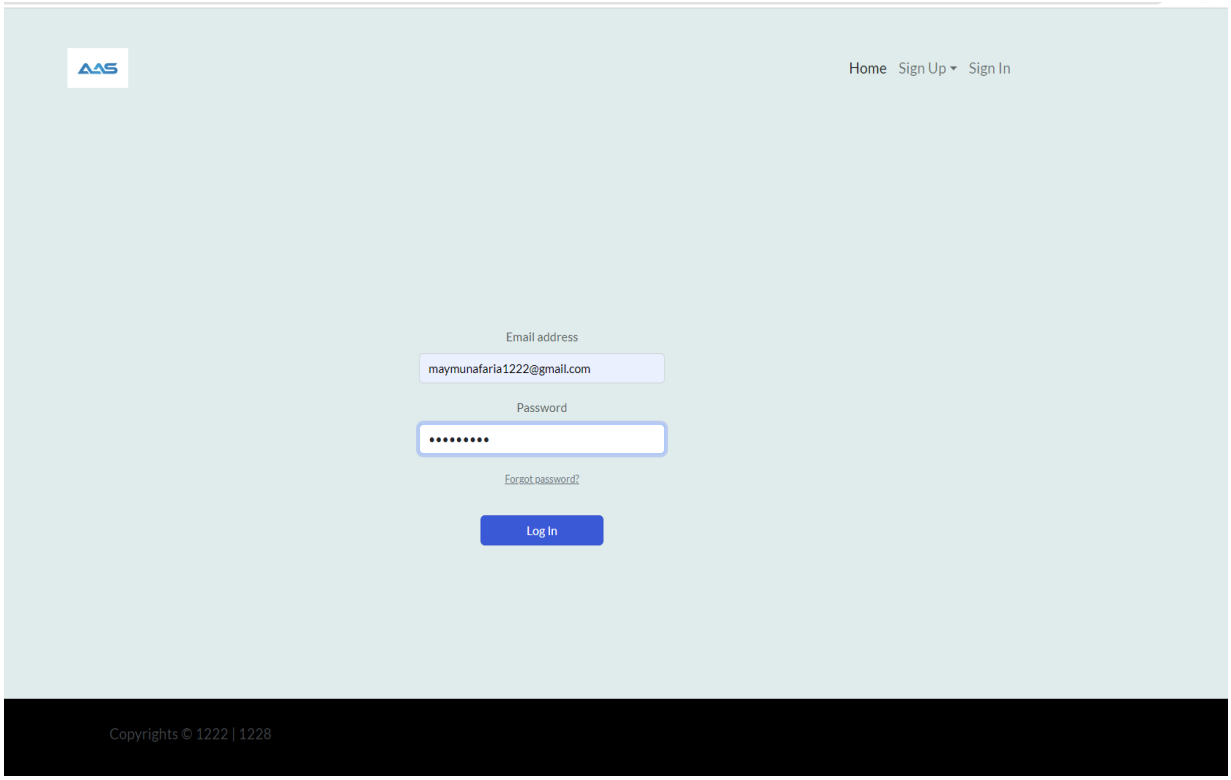


Figure 5.2: *Sign In Page*

To reset their passwords, there is also a forget password option.

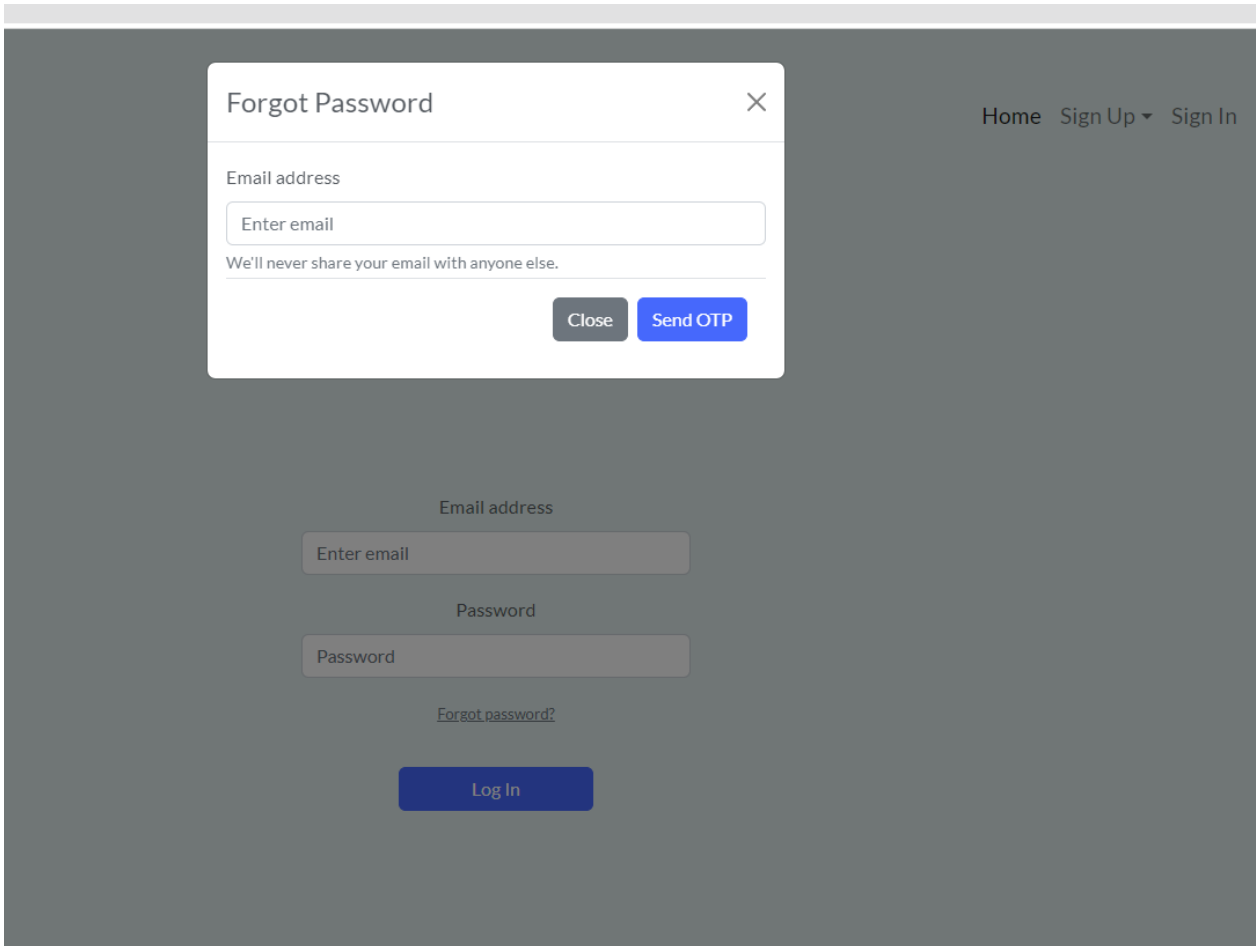


Figure 5.3: *Forget Password Page*

After confirming their email address and entering the relevant information, users can create an account.

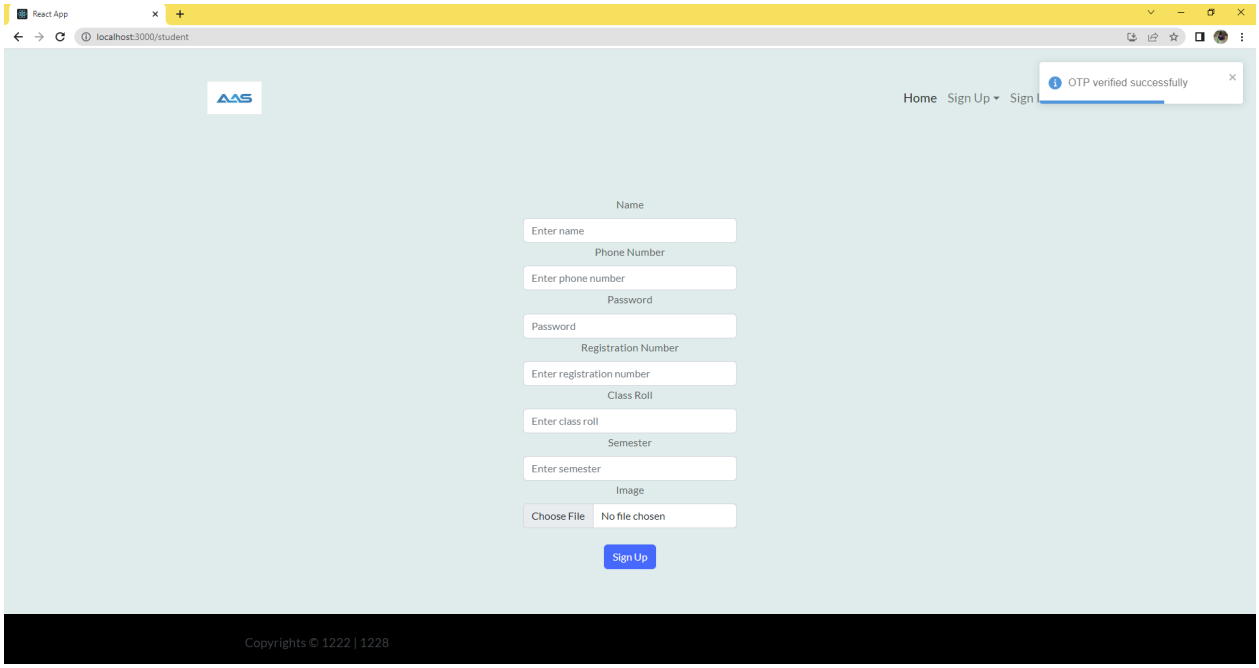


Figure 5.4: *Sign Up Page*

All users share a few features in common. They have a navigation bar on every page that allows them to access the dashboard, log out, and access the pages for updating their information, receiving notifications, and changing their password.

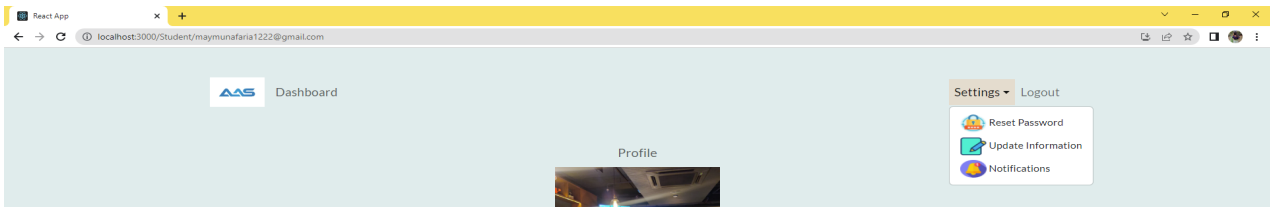


Figure 5.5: *Navigation Bar*

Below are the update information pages and the reset password page.

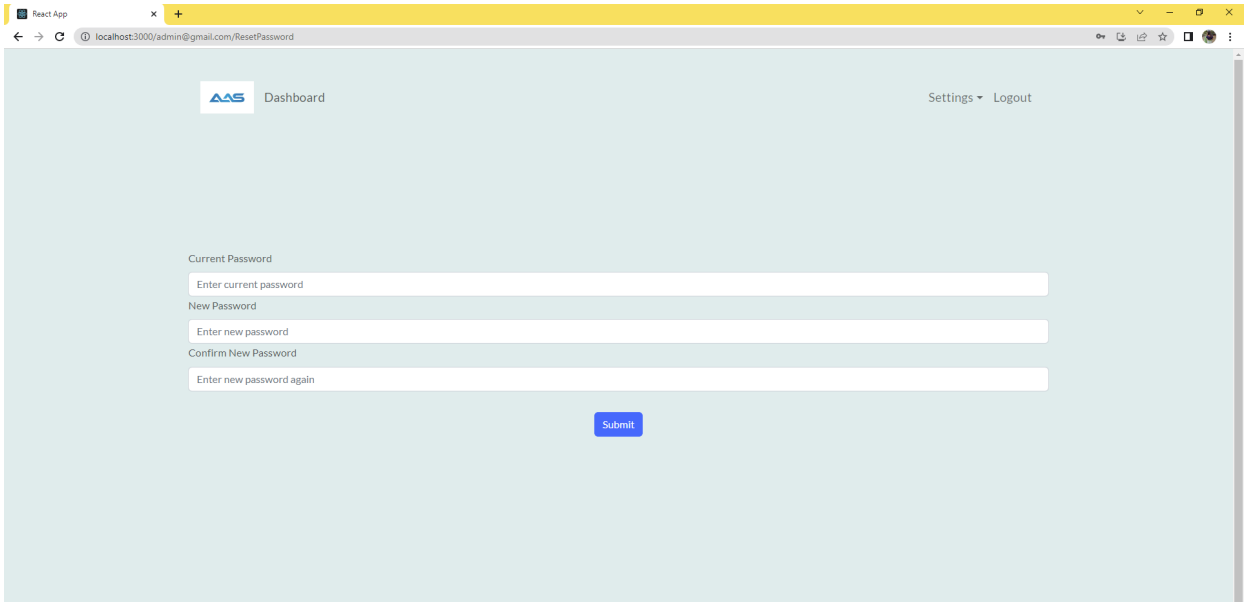


Figure 5.6: *Reset Password Page*

Figure 5.7: *Edit Information Page*

To view their information and notifications, they can visit their profile and notification, respectively. All users can see attendance reports.

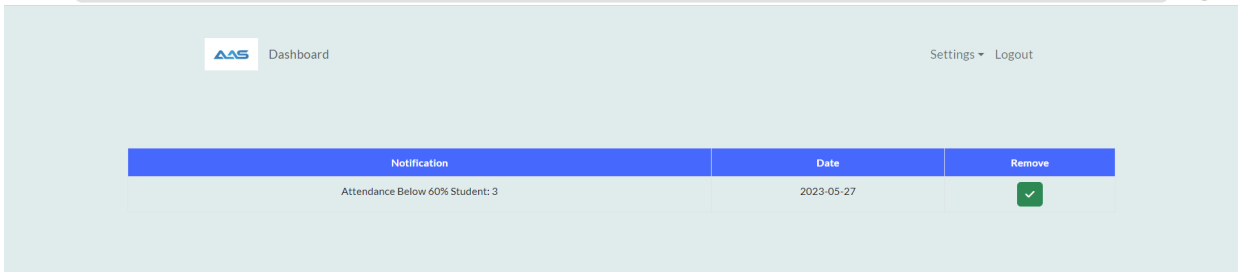


Figure 5.8: Notification Page

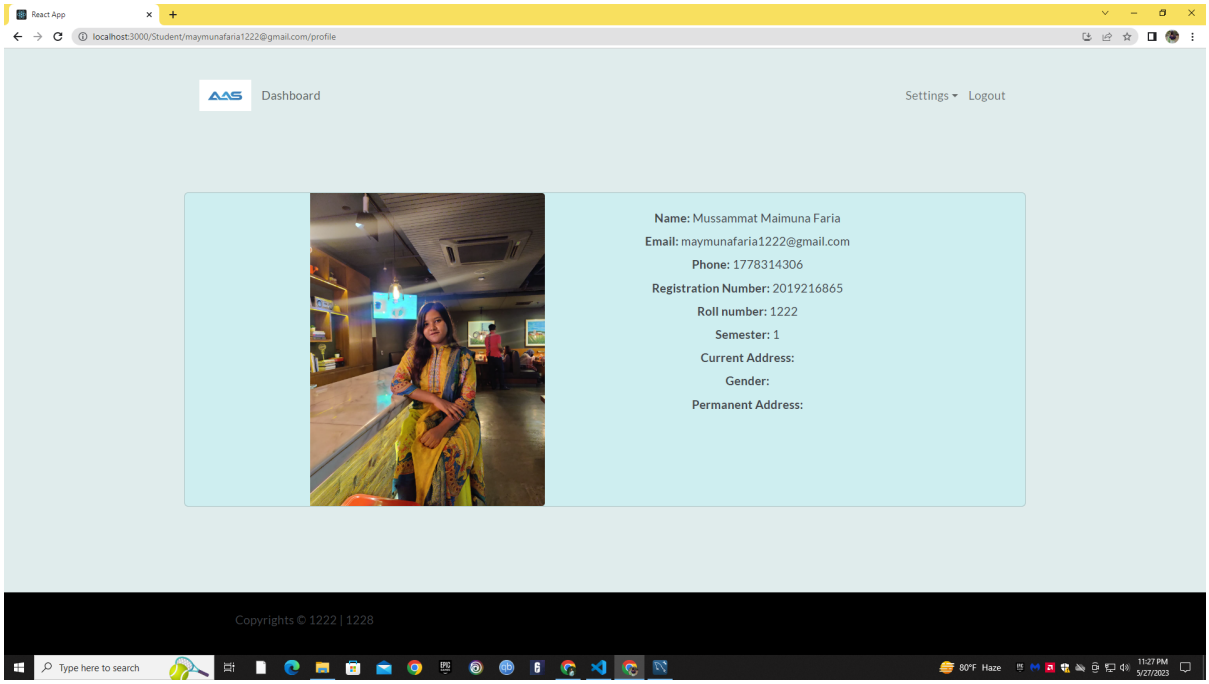


Figure 5.9: *Profile Page(Student)*

5.1 Admin Dashboard

If the usertype is admin then they can visit the admin dashboard which has some functionalities.

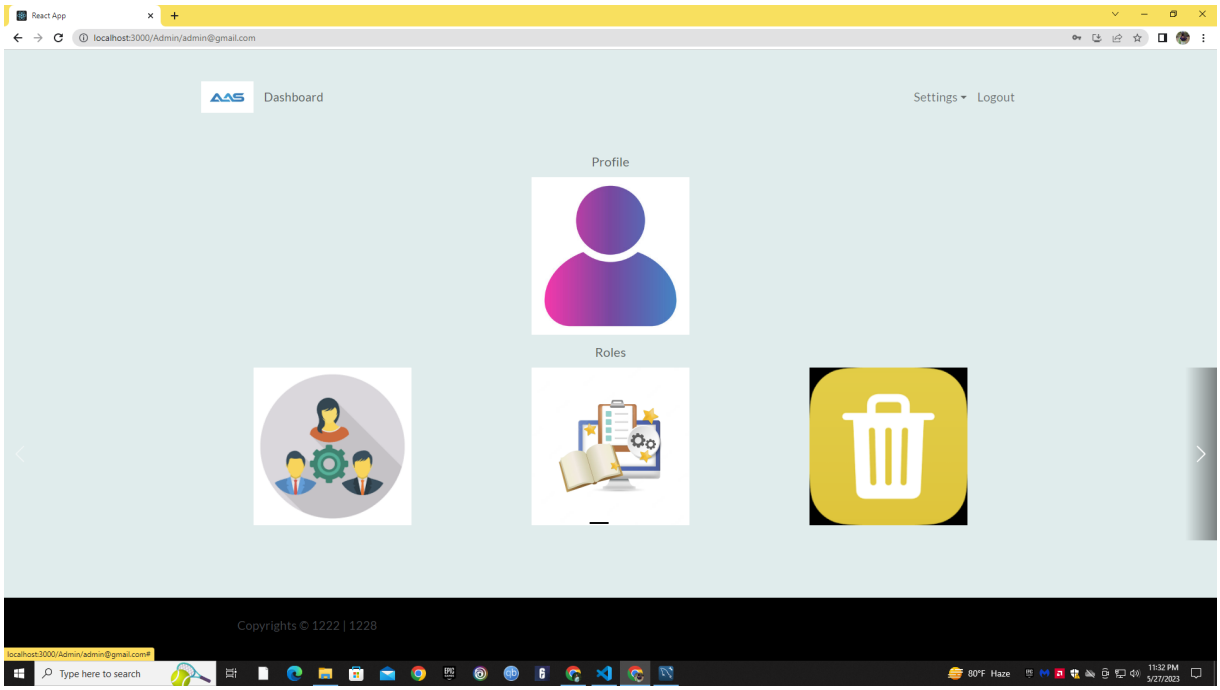


Figure 5.10: *Admin Dashboard*

5.1.1 Manage User

By providing their email address, an admin may remove any user. Teachers and directors can be added by admin by providing their details. Before a user may access their accounts, an admin must verify their account.

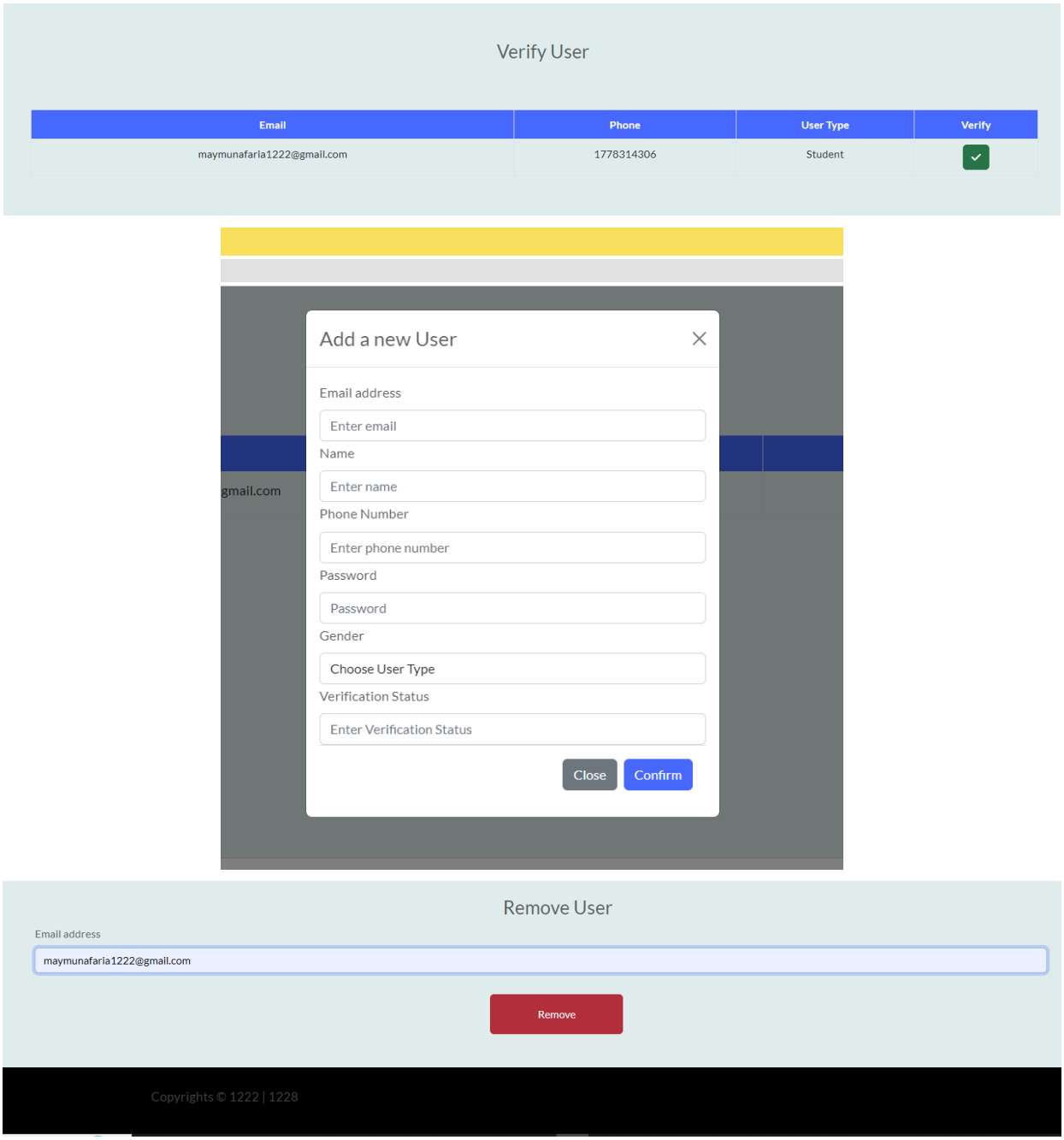


Figure 5.11: *Manage Users(verify,add,remove)*

5.1.2 Manage Course

Courses must be allocated to semesters by the admin so that once a student creates an account, those courses are automatically added to their semester. The admin will also assign teachers to the courses.

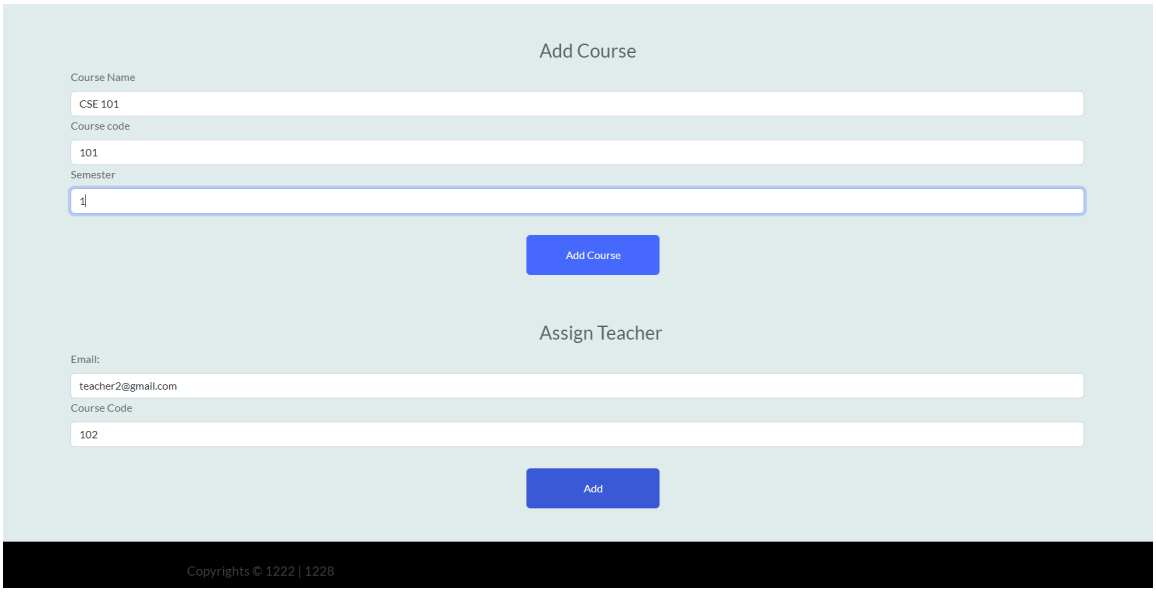


Figure 5.12: *Manage Course*

5.1.3 Manage Notice

Director notices will be received by the admin, who will then send notices to the students by providing their registration number.

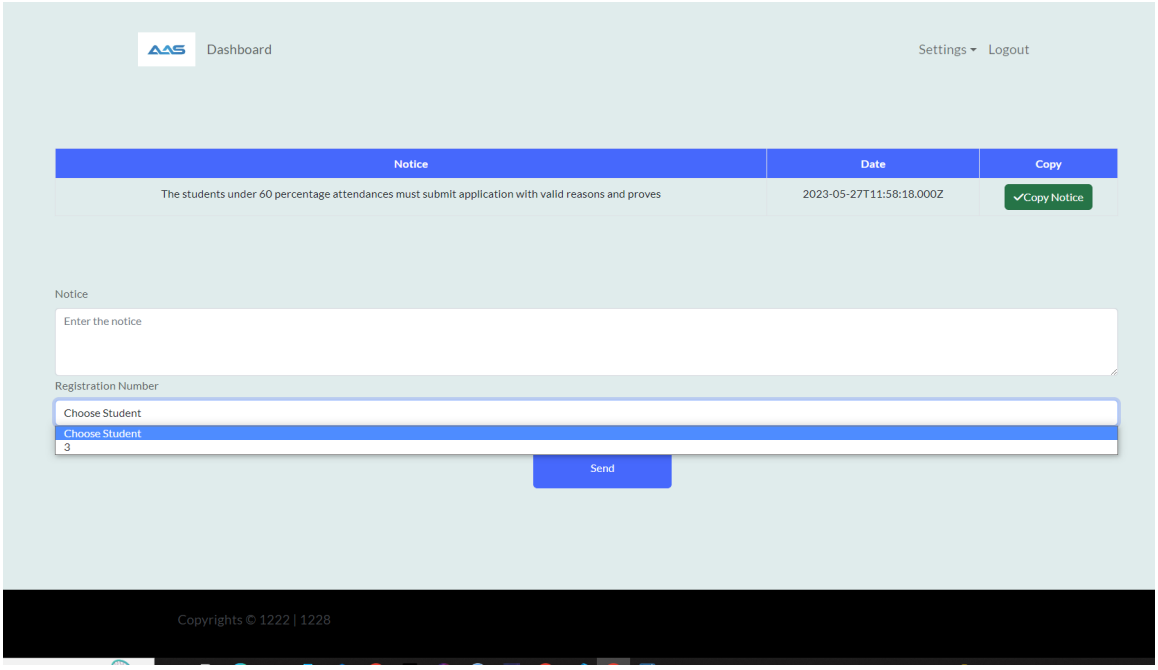


Figure 5.13: *Manage Notice*

5.2 Teacher Dashboard

If the usertype is a teacher then they can visit their teacher dashboard which has some additional functionalities.

5.2.1 Start class

To start a class, a teacher can provide the course code. They can then decide to manually take attendance by using the registration numbers of each student who is enrolled in the course. Students must scan their barcodes to provide attendance if the teacher chooses the digital option. When class is over, the teacher will select the finish option.

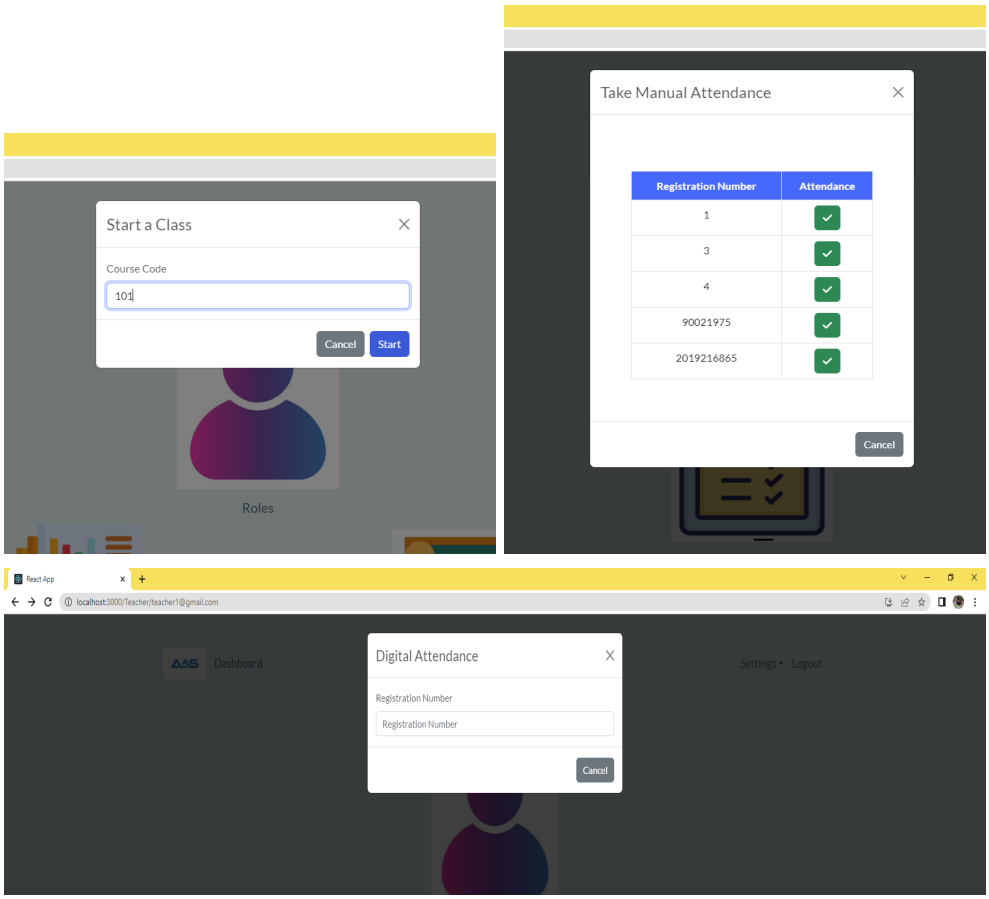


Figure 5.15: *Take attendance (manual and digital)*

5.2.2 Edit attendance Information

The teacher can edit attendance records by giving the course code , registration numbers and attendance status.

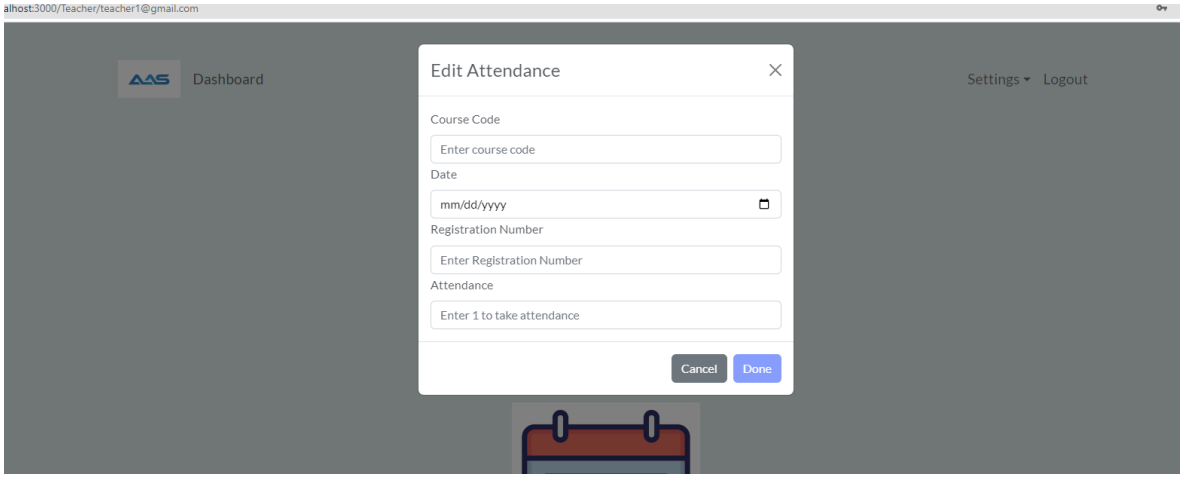


Figure 5.14: *Edit attendance*

5.3 Director Dashboard

The admin can manage all of the directors' functionalities. The director may:

5.3.1 Send Notice

Director can send notices to the admin about anything.

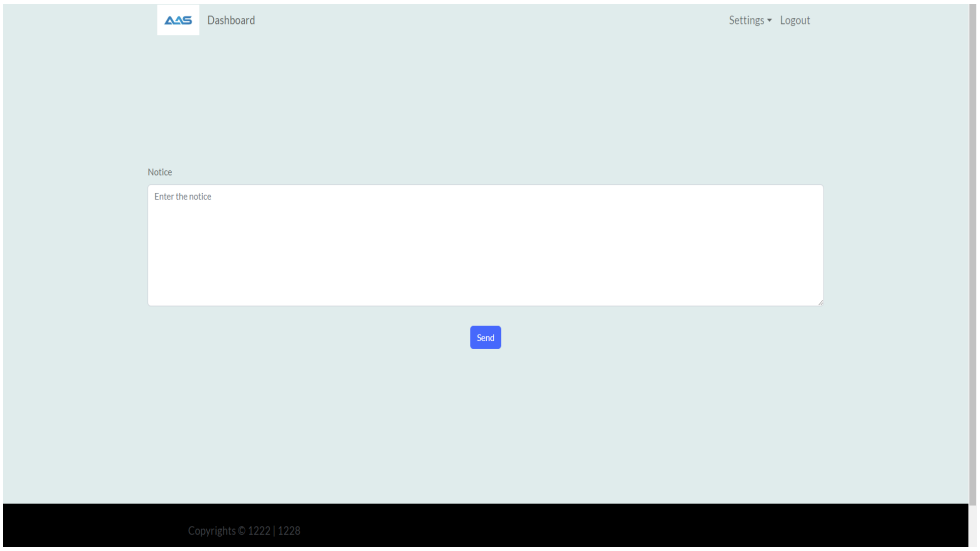


Figure 5.15: *Send Notice*

5.3.2 Approve Application

In order to maintain track of this, those who have attendance below 60% can manually submit applications to the director, who can then approve them.

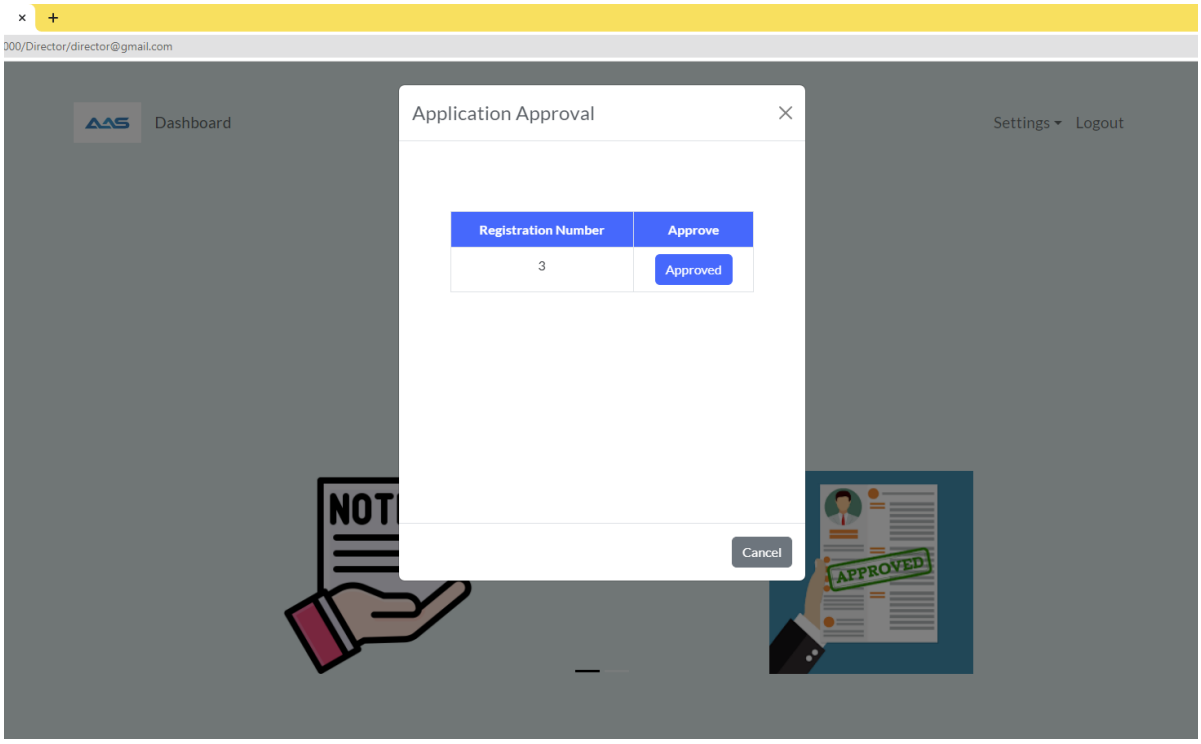


Figure 5.16: *Application Approval*

5.4 Student Dashboard

If the usertype is student then they can visit their student dashboard

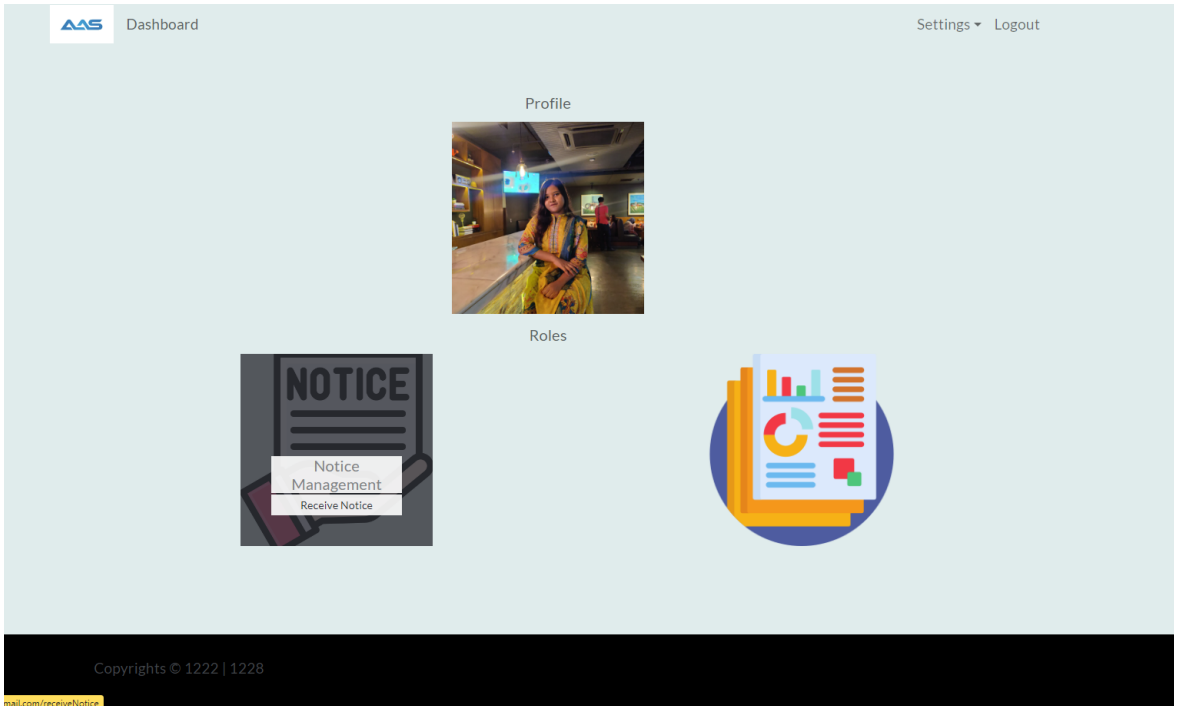


Figure 5.17: *Student Dashboard*

5.4.1 Receive Notice

Students will receive notice from the admin.

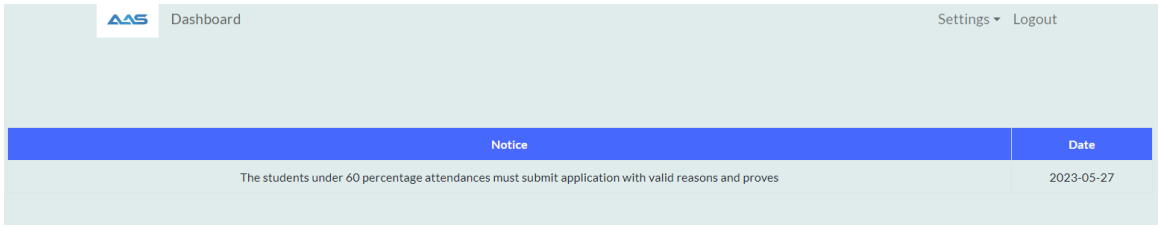


Figure 5.13: *Receive Notice*

6. Challenges Faced

We had a number of other difficulties in addition to adjusting to new surroundings and creating a full stack application for the first time, including:

6.1 ReactJS dependency issues

Updates for ReactJS that include new functionality and syntax modifications are routinely released. Every new version could bring about adjustments that demand programmers modify their code accordingly. Additionally, the kind of hooks that can be utilized may differ between ReactJS versions.

Our team faced difficulties when using ReactJS in the beginning because we had to adapt our code to the particular needs and syntax of the selected version.

6.2 Integration of Barcode Scanners

Getting the barcode scanners to work with the attendance system was one of the biggest challenges we ran into. To ensure smooth communication between the scanners and the system, we need to work carefully as the scanner will read the barcodes continuously. During the implementation phase, problems including inaccurate barcode reads, broken scanners, and software compatibility conflicts appeared.

6.3 Barcode Placement and Quality

We found that the accuracy of the system was strongly impacted by the quality of the barcodes on identity cards or tags. Inaccurate attendance records were frequently the consequence of failed readings from damaged or fading barcodes. Additionally, it was difficult to guarantee consistent placement of the barcode inside the scanning range of the scanner because minute misalignments could result in inaccurate data acquisition. Also sometimes the scanner can't read the barcodes because of the cards' poor quality.

6.4 Lighting Conditions

The changing lighting conditions were another difficulty for us. The accuracy with which the barcode scanner can read the barcodes may be hampered by poor lighting, shadows. It became essential for dependable performance to adjust the lighting in the attendance area to maximize barcode visibility and reduce interference. Same difficulties are faced for proxy detection as cameras also can't detect faces if the lighting conditions aren't good.

6.5 Camera-Based Proxy identification

Putting camera-based proxy identification into practice had its own set of challenges. Detecting proxies and accurately identifying persons required complex image processing methods. Low-resolution photos, differences in facial look owing to various lighting

conditions, and probable occlusions produced by accessories or masks were among the difficulties the system had to overcome.

7. Conclusion

In conclusion, the use of automated attendance systems that use barcode scanning technology and ID cards represents a significant advancement in the field of attendance tracking. These systems have several advantages over traditional manual approaches, including increased efficiency, reliable data collecting, and real-time reporting. Educational institutions can save time and money by removing the need for physical record-keeping and simplifying administrative processes.

Furthermore, by lowering the danger of unwanted access or manipulation, automated attendance systems improve data security and confidentiality. Manual attendance systems, on the other hand, have a number of drawbacks, including the possibility of errors, time-consuming paperwork, and being exposed to dishonest tactics such as proxy. These limitations may threaten data accuracy, delay administrative operations, and result in financial and academic problems.

Overall, the implementation of automated attendance systems has become an important task for us. As technology advances, automated attendance systems will likely become increasingly more complex, offering expanded features and capabilities to improve attendance management efficiency and accuracy.

References

[1]https://www.academia.edu/35876517/Barcode_based_Student_Attendance_System
[2]https://ignited.in/I/a/305535#:~:text=Every%20student%20is%20given%20a,attendance%20according%20to%20the%20dates
[3]https://pypi.org/project/face-recognition/

Appendix

Github Link of the project

https://github.com/MonayemSarker/AAS