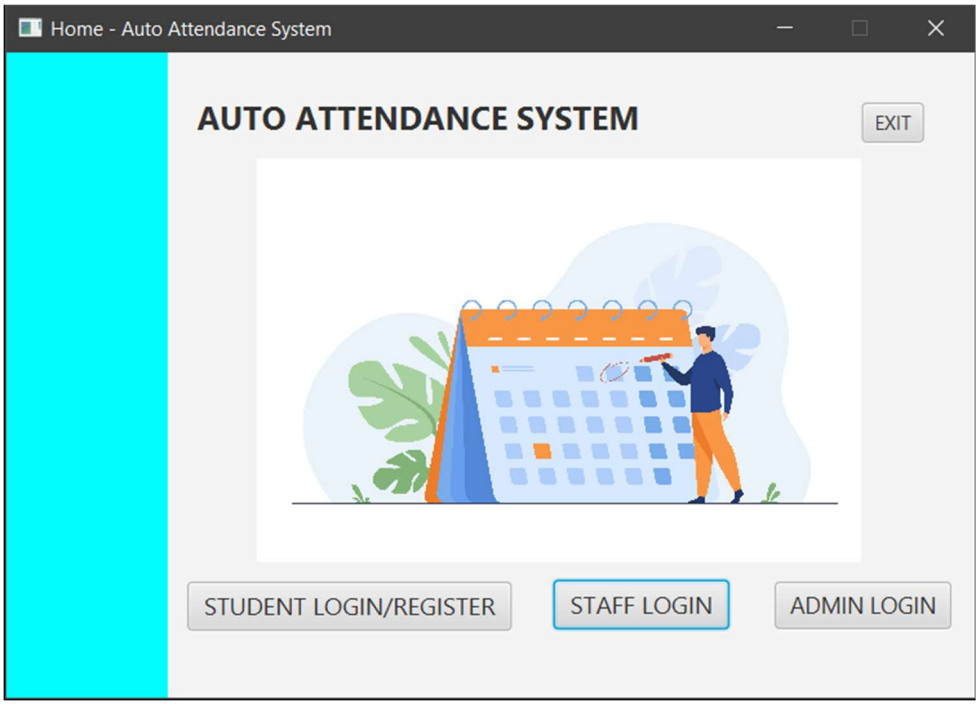
CS6308 – JAVA PROGRAMMING

BE COMPUTER SCIENCE AND ENGINEERING (RUSA)

PROJECT REPORT

TITLE : AUTO-ATTENDANCE SYSTEM USING FACE RECOGNITION



Name : Ajitesh M

Reg. No : 2019103503

Sem & Batch : V – Q Batch

Date : 12.01.2022

ABSTRACT

Maintenance of attendance is an important task in every school and colleges since it is one of the primary ways to check the regularity of every student. As of now, attendance in schools and colleges is done by maintaining the attendance sheet which is a time-consuming procedure.

Being virtually present in online mode, verifying each student in the class is difficult. Hence, the mechanism of face detection and face recognition method is proposed to maintain the attendance. It not only save the time but, also provides a feeling of virtually present in offline classes as the students keep their video on throughout the class and it does not allow proxy or false attendance. This system, which is based on HAAR Cascade Classification and LBPH algorithms, detects and recognizes the presence of student via the webcam throughout the class session. It sends the time-series data of predicted student register number at regular intervals and by the end of the class it calculates the attendance percentage.

An automatic attendance system is a necessary tool for any learning management schools. Our approach aims to solve the issues by integrating face recognition in the process. The system can be enhanced in such a way that the accuracy, detection rate and recognition rate can be increased so that more number of students can be detected and recognized for those who are present in the class.

INTRODUCTION

Maintenance of attendance is an important issue of every school and colleges since it is one of the primary ways to check the regularity of every student. As of now, attendance in schools and colleges is done by maintaining the attendance sheet which is a time-consuming procedure. Utilization of the same time may help the student to get some more information from the instructor. Moreover, one can easily manipulate the attendance since it is manually recorded. Also, verifying each student in the class is difficult. Hence, the mechanism of face detection and face recognition method is proposed to maintain the attendance. It not only save the time but, also prevents the students from giving the fake attendance. Hence, it creates the awareness for the students to attend classes daily since the attendance is going to be monitored by an automated machine.

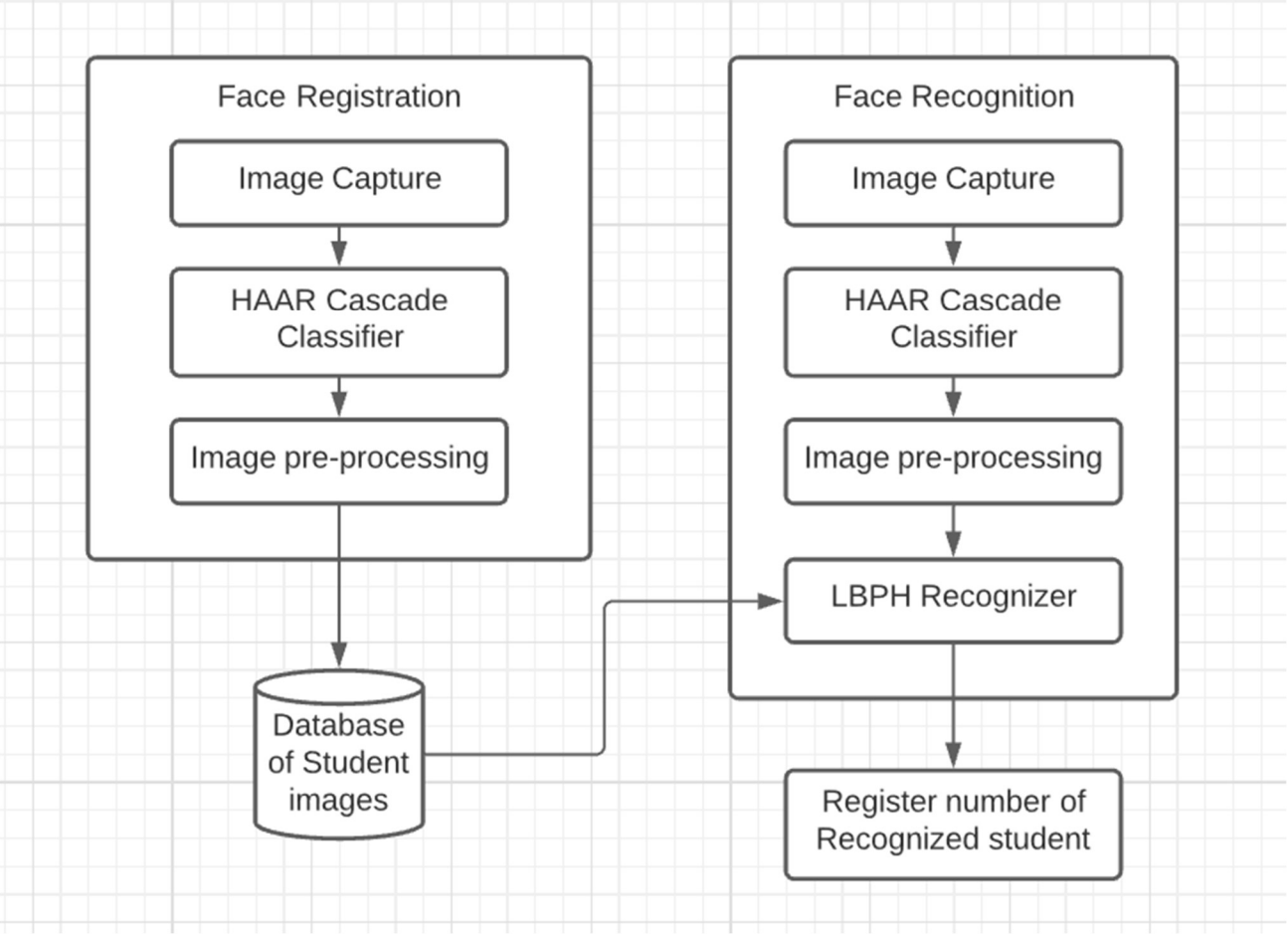
Over the past few years, extensive research has been done on the field of face recognition which is one of the best ways to find the human identity. Face recognition from images is a popular research in biometrics. One of the most useful applications of face recognition is understanding the image analysis.

Though there exist many systems to detect the faces and to recognize them from images. Many research works are going to find better factors that increase the efficiency and accuracy. In general, the factors like pose, occlusion, illumination and so on are influencing the efficiency and needs high processing capacity for retrieving from the large image dataset. It may lead to focus on large image dataset and also on new algorithms which reduce the computational issues and to increase the accuracy. Efficient face recognition of human being from image dataset is an ultimate goal. In the field of biometric face recognition, research has been done to identify the individuals from a picture of the group of people based on facial features. There are many Applications for the face recognition and are widely used in security based applications and biometric systems. In general, face recognition system has three important blocks namely face detection, training of detected faces, and face recognition. In this project, HAAR Cascade Classifier is used for face detection and LBPH algorithm for Face Recognition.

TECH STACK USED:

- GUI – JavaFx with FXML
- Database – MySQL
- IDE - IntelliJ
- Face Detection and Recognition – JavaCV + OpenCV for Java

ARCHITECTURE DIAGRAM :



IMPLEMENTATION METHODOLOGY

1. Face detection

Face Detection has the objective of finding the faces (location and size) in an image and extract them to be used by the face recognition algorithm.

Haar Cascade classifiers are an effective way for object detection. This method was proposed by Paul Viola and Michael Jones in their paper Rapid Object Detection using a Boosted Cascade of Simple Features. Haar Cascade is a machine learning-based approach where a lot of positive and negative images are used to train the classifier. Here the object we have to detect is human face.

2. Collection of dataset

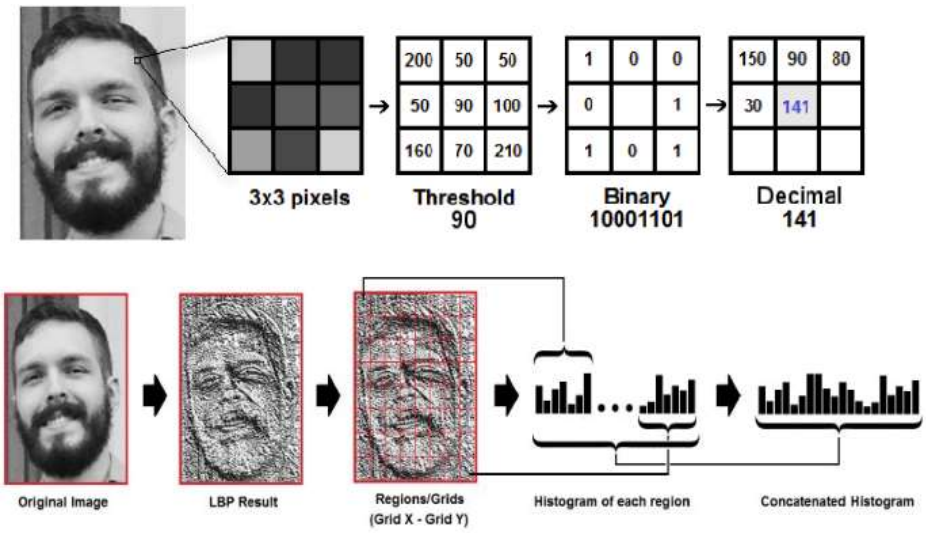
The dataset of student images are collected when a student is registered. 10 images of each student are recorded. Each image is labeled with the corresponding student's register number and is used to train the face recognizer model.

3. Face Recognition

In Face Recognition, with the facial images already extracted, cropped, resized and usually converted to grayscale, algorithm is responsible for finding characteristics which best describe the image.

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. For the face recognition process, Local Binary Pattern Histogram algorithm is applied. The LBP operator uses local binary patterns to reduce the local spatial distribution of a face image. The LBP operator is a collection of binary pixel value ratios in the center at regular pixel intervals and is around 8 pixels.

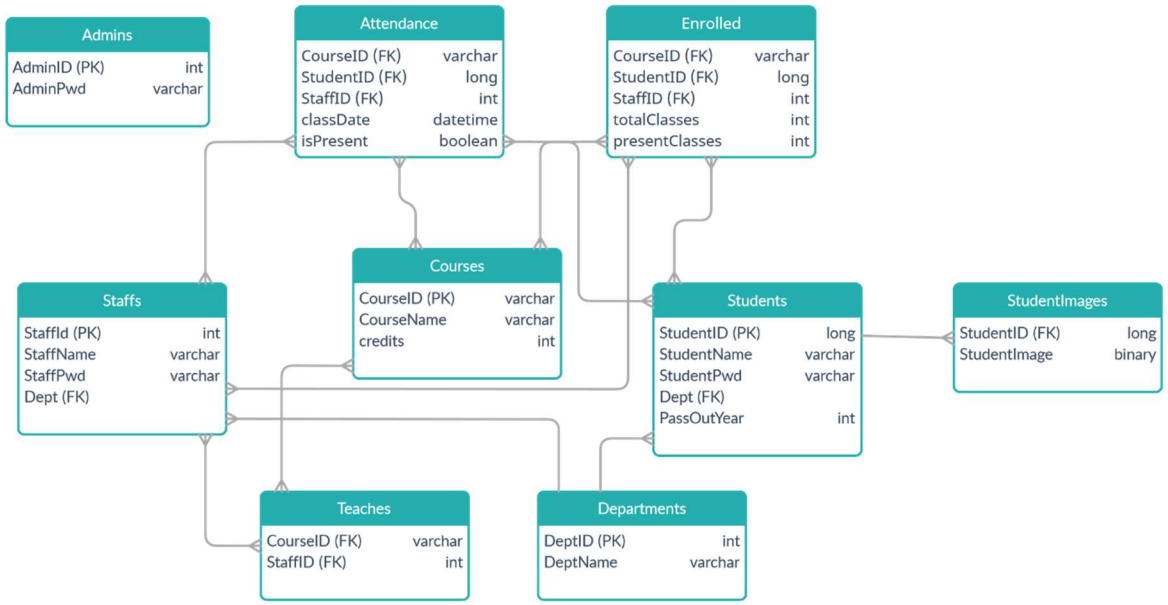
The LBPH Recognizer model is trained with the collected face dataset. The detected face extracted from the captured image is passed to the model for prediction.



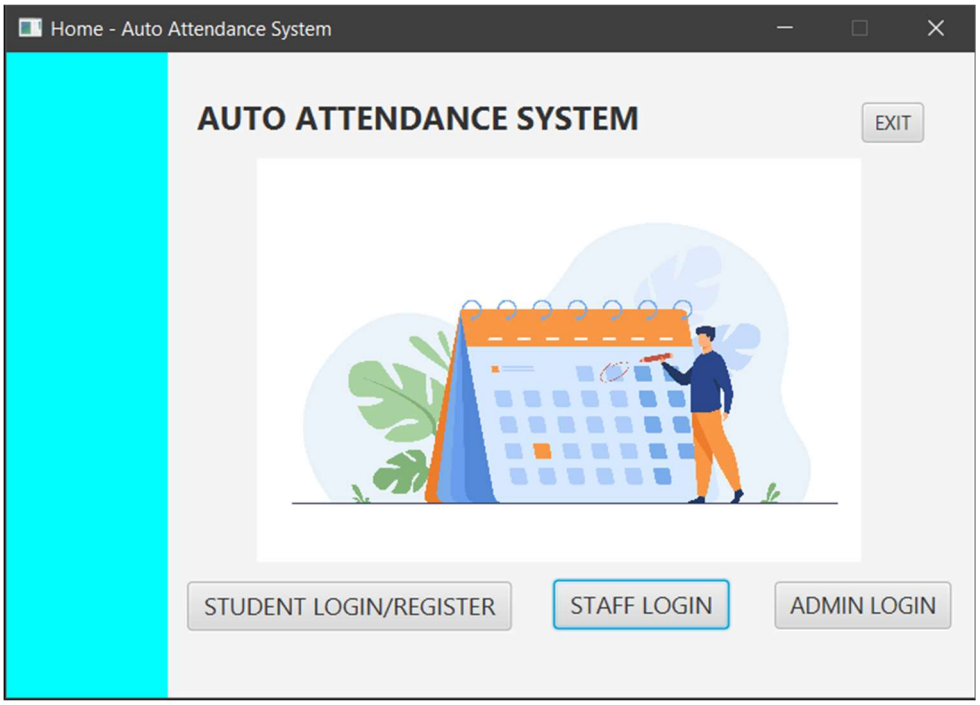
4. Mark Student Attendance

Students join the class session started by the staff. The time series data of the predicted roll number of the student image captured is sent regularly to the staff, where the attendance of all students are maintained.

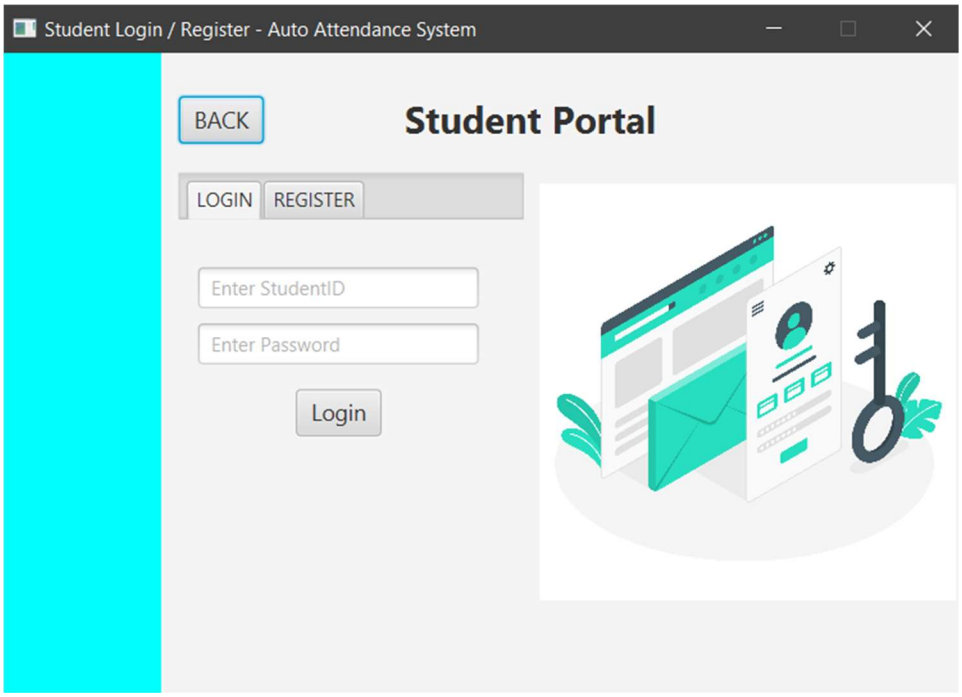
DATABASE DESIGN :



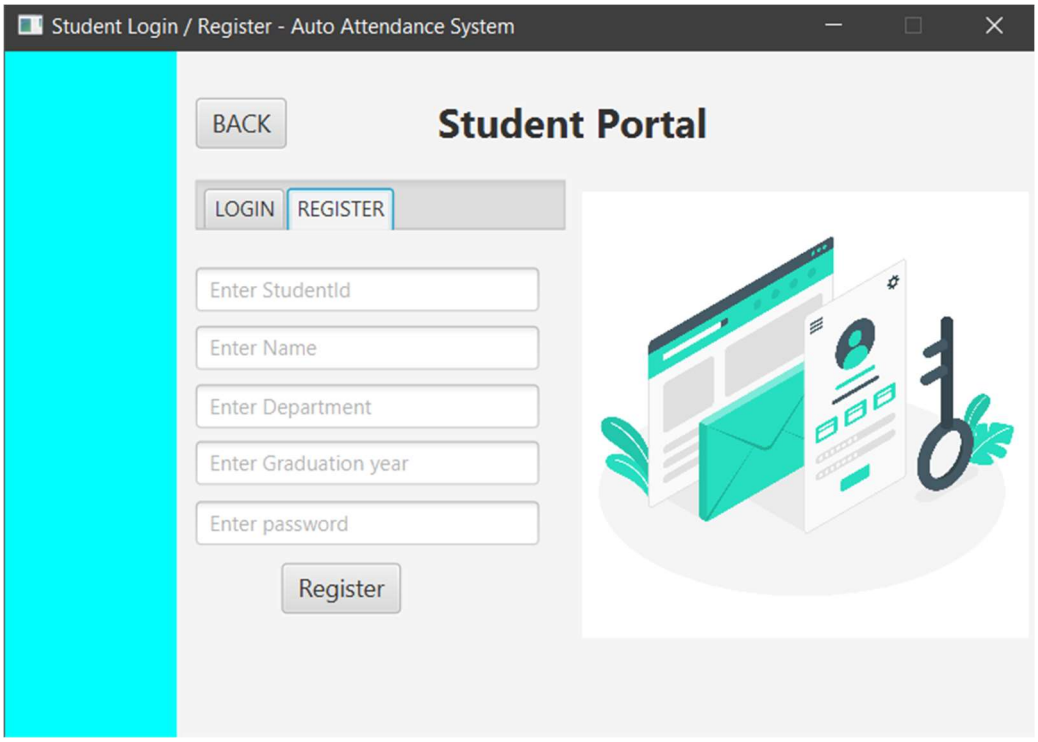
USER INTERFACE DESIGN :



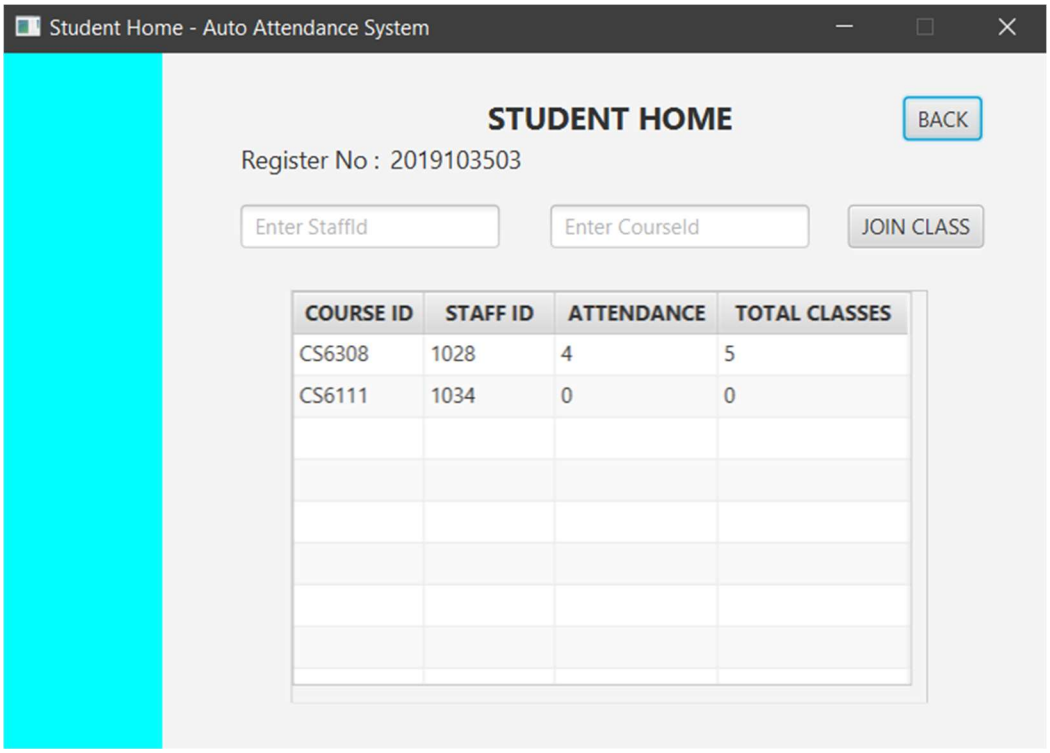
Home Screen



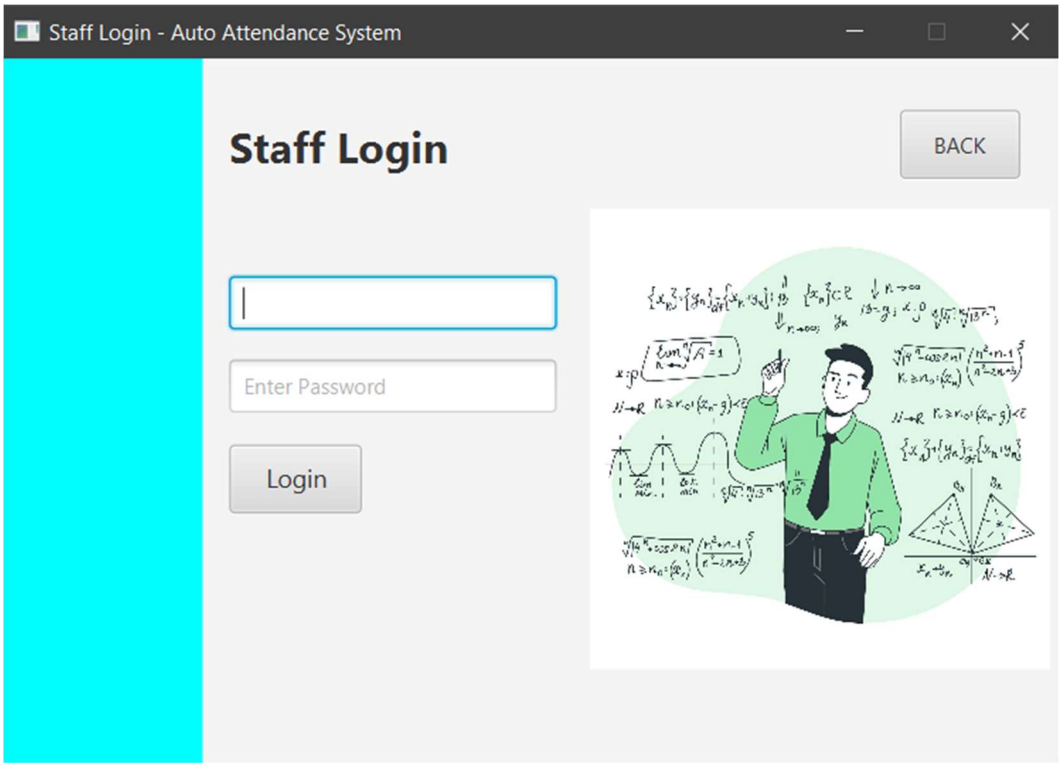
Student Login Screen



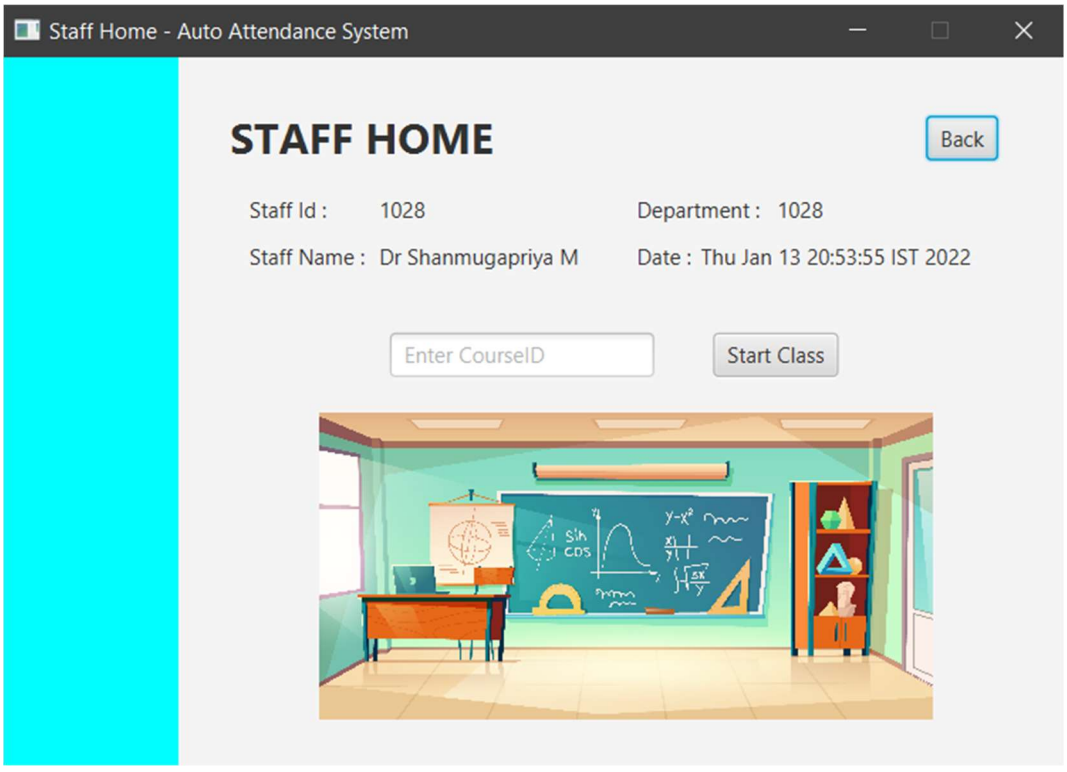
Student Register Screen



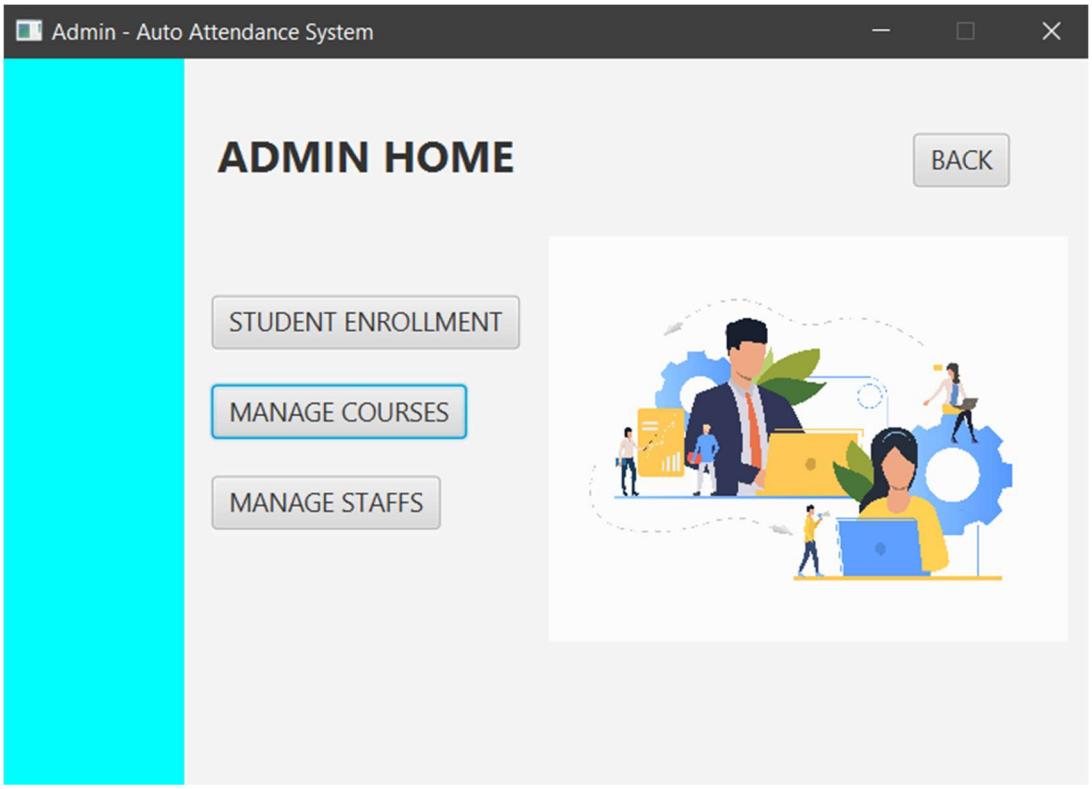
Student Home Screen



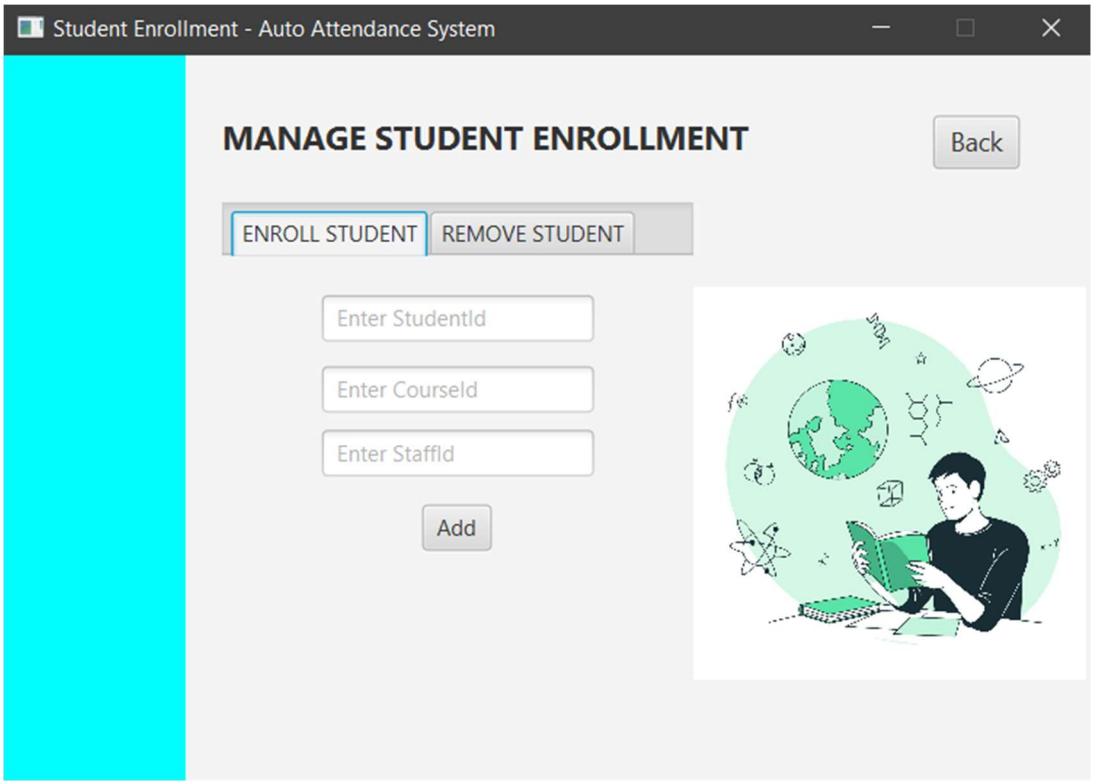
Staff Login Screen



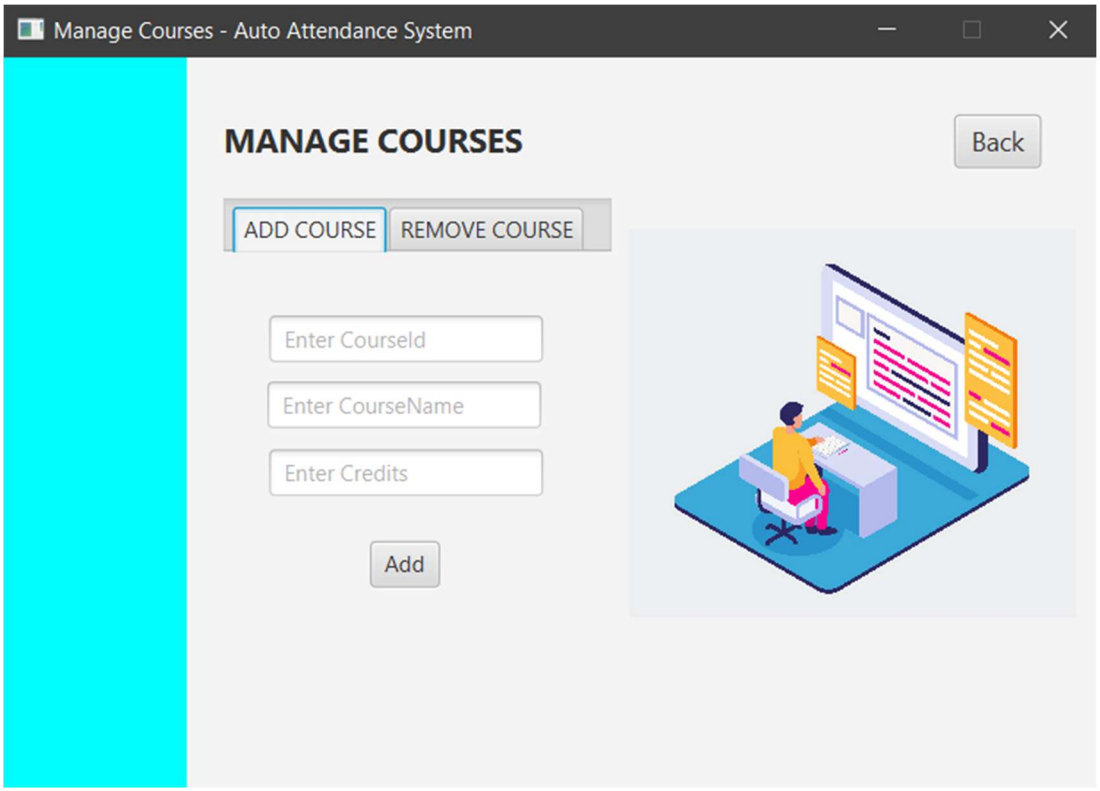
Staff Home Screen



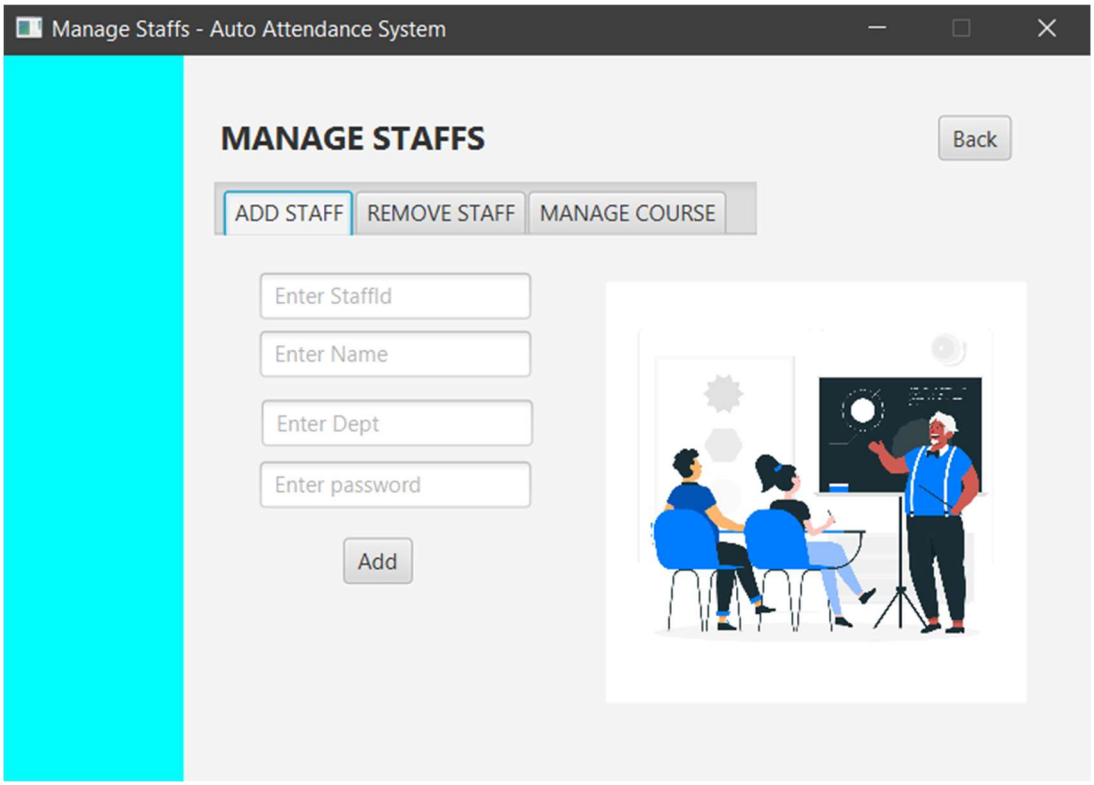
Admin Home Screen



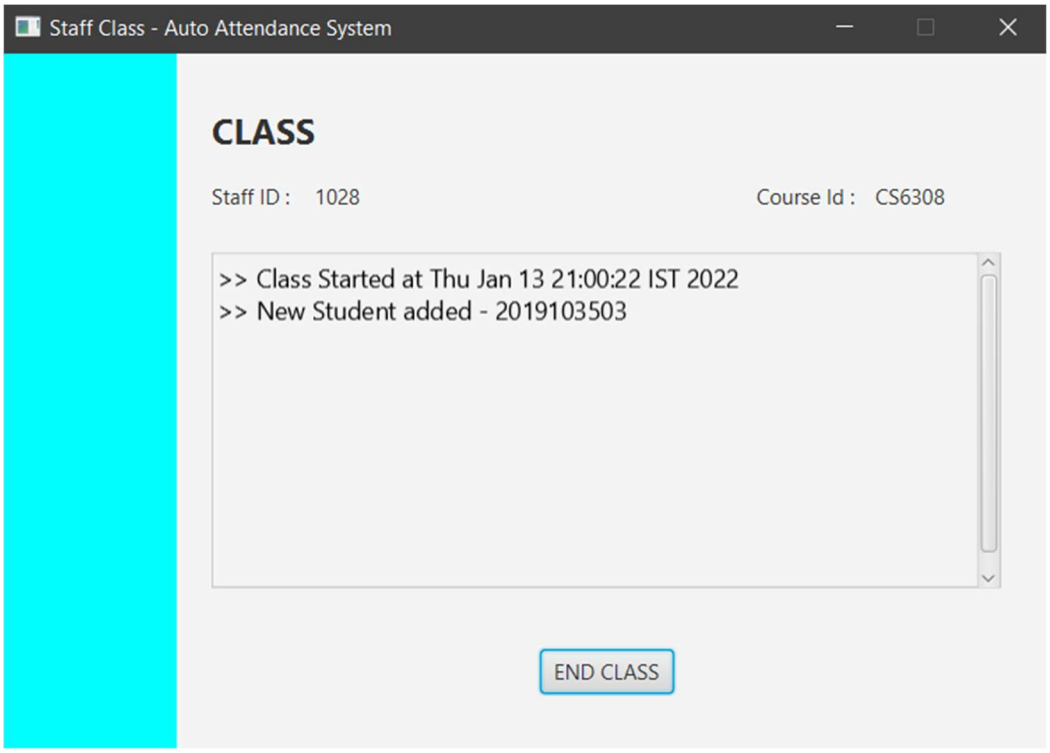
Student Enrollment Screen



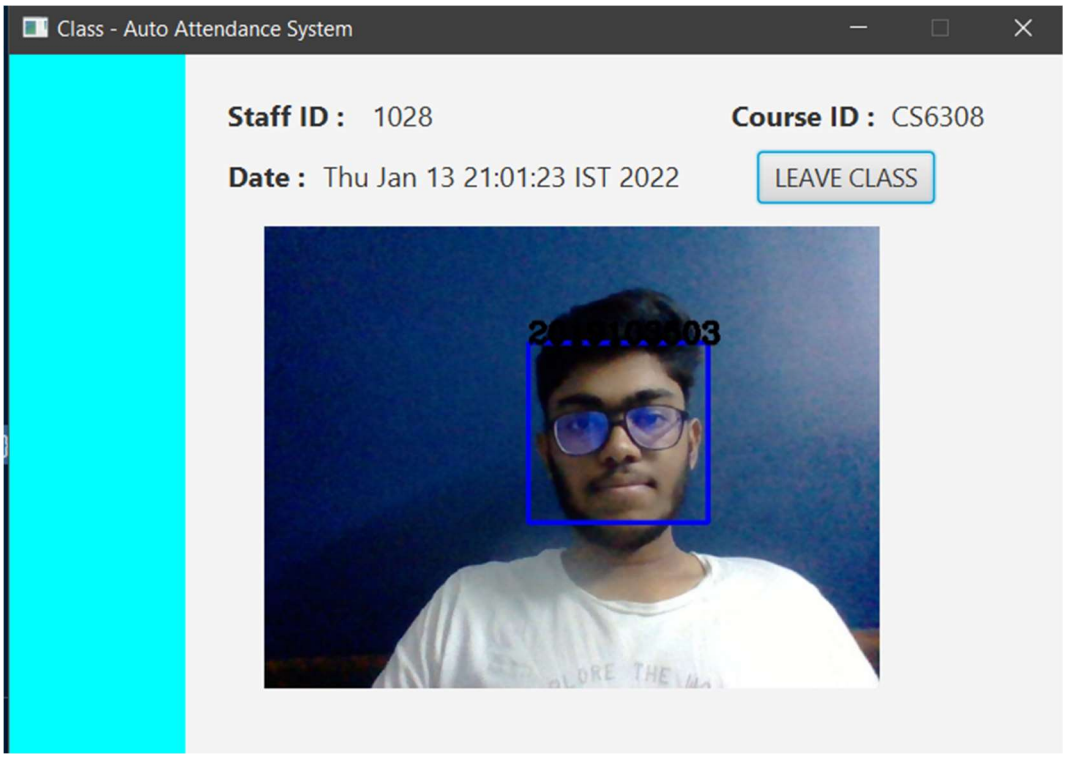
Manage Courses Screen



Manage Staffs Screen



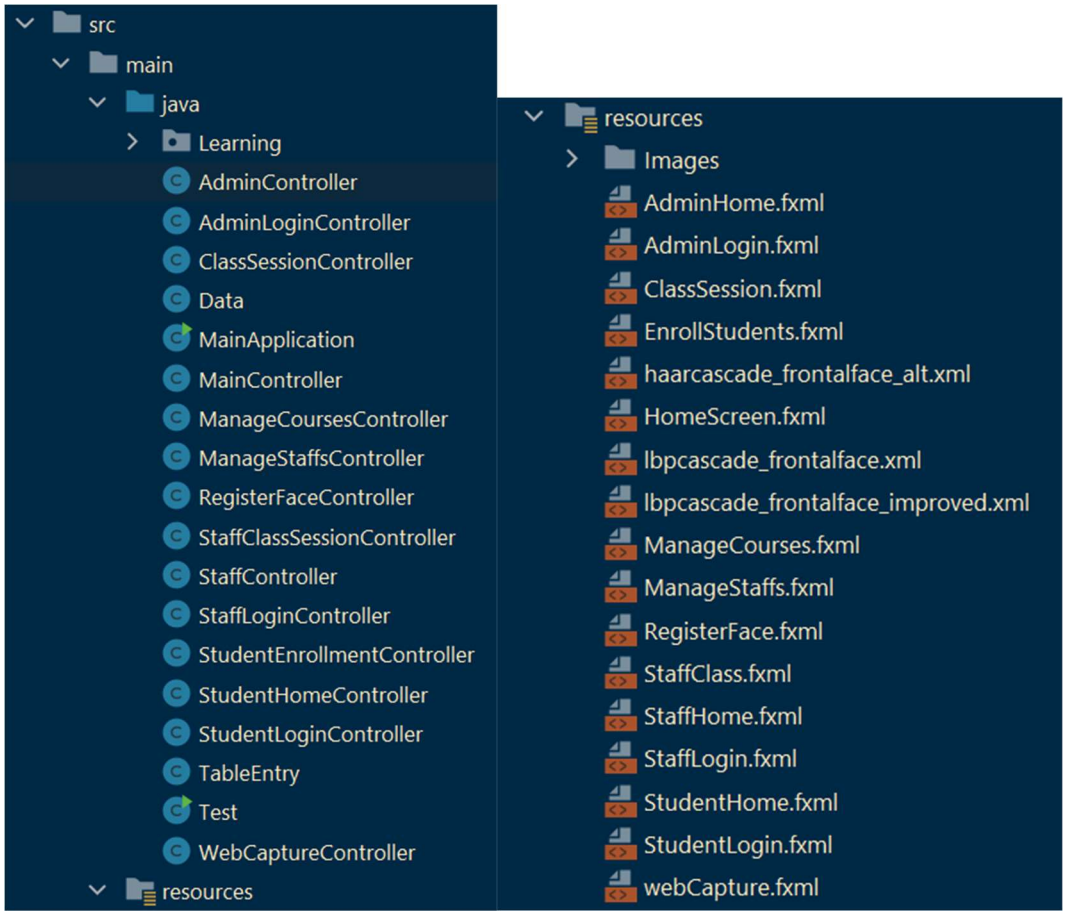
Class Screen at Staff side



Class Screen at Student side

IMPLEMENTATION AND RESULTS

The classes and resources used in this project,



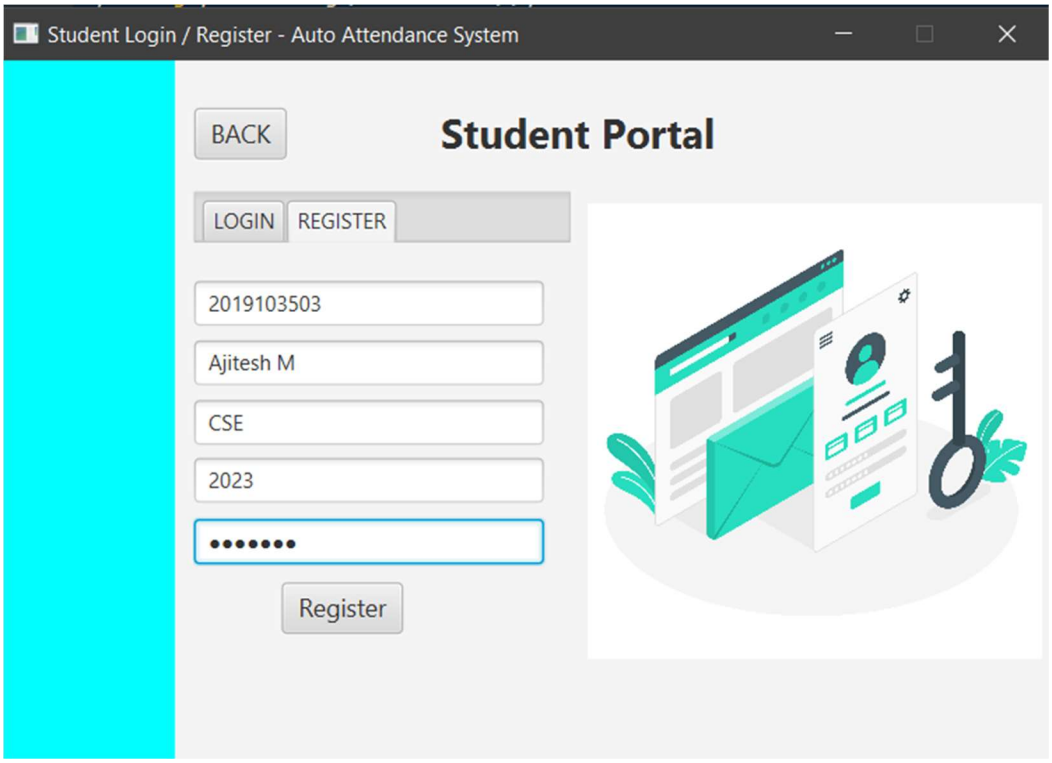
1. Registering a new student

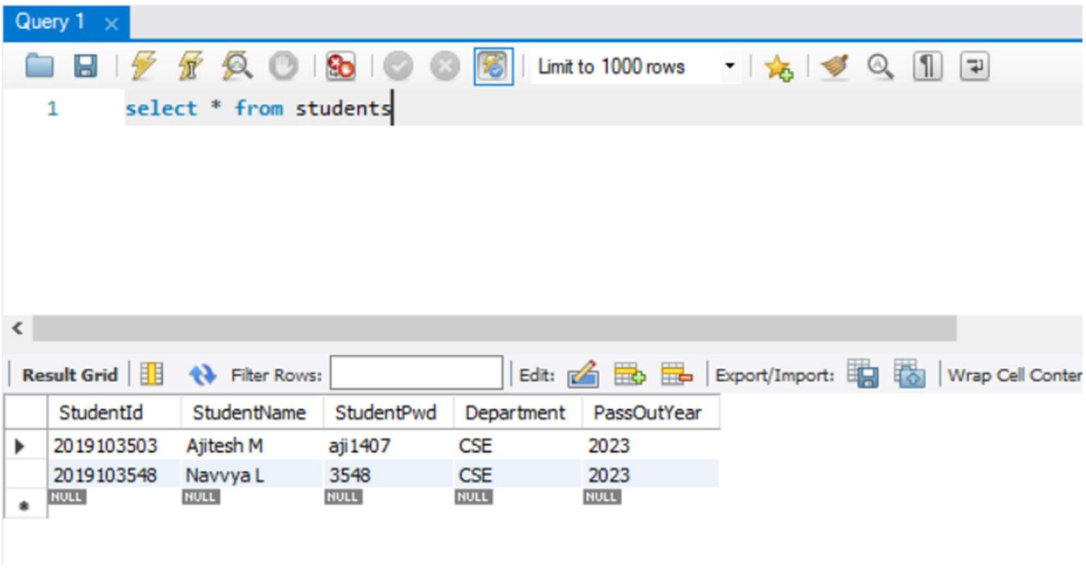
To register a new student, (from StudentLogincontroller.java)

```
public void registerHandler(ActionEvent e) {
    String studentId = regIdTxt.getText();
    String name = regNameTxt.getText();
    String dept = regDeptTxt.getText();
    String gradYear = regYearTxt.getText();
    String password = regPassTxt.getText();

    if(studentId.trim().equals("") || name.trim().equals("") ||
dept.trim().equals("") || gradYear.trim().equals("") ||
password.trim().equals("")) {
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle("Register status");
        alert.setHeaderText("Required");
```

```
        alert.setContentText("All fields are required !!");
        alert.showAndWait();
    } else {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/AUTO_AT
TENDANCE", "Torvus", "Torvus@1407");

            PreparedStatement statement =
conn.prepareStatement("insert into Students values (?, ?, ?, ?,
?)");
            statement.setLong(1, Long.parseLong(studentId));
            statement.setString(2, name);
            statement.setString(3, password);
            statement.setString(4, dept);
            statement.setInt(5, Integer.parseInt(gradYear));
            statement.executeUpdate();

            statement = conn.prepareStatement("select * from
Students where StudentId = ?");
            statement.setLong(1, Long.parseLong(studentId));
            ResultSet resultSet = statement.executeQuery();
            if(resultSet.next()){
                regIdTxt.setText("");
                regPassTxt.setText("");
                regYearTxt.setText("");
                regDeptTxt.setText("");
                regNameTxt.setText("");

                FXMLLoader loader = new
FXMLLoader(getClass().getResource("RegisterFace.fxml"));
                root = loader.load();
                RegisterFaceController controller =
loader.getController();
                controller.getRegNum(Long.parseLong(studentId));

                stage = (Stage) ((Node)
e.getSource()).getScene().getWindow();
                stage.getScene().setRoot(root);
                stage.setTitle("Register - Auto Attendance
System");
                stage.show();
            } else {
```

```
                Alert failure = new
Alert(Alert.AlertType.ERROR);
                failure.setTitle("Register status");
                failure.setHeaderText("Registration failed.");
                failure.showAndWait();
            }
        } catch (SQLIntegrityConstraintViolationException
exception){
            Alert failure = new Alert(Alert.AlertType.ERROR);
            failure.setTitle("Register status");
            failure.setHeaderText("Registration failed.");
            failure.setContentText(exception.getMessage());
            failure.showAndWait();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
}
```





After successfully registered a new student and inserted his entry into the students table, the student has to register his face by taking 10 pictures of his face at various angles in a well lit room which is then inserted into the studentImages table which contains the images of all student faces.

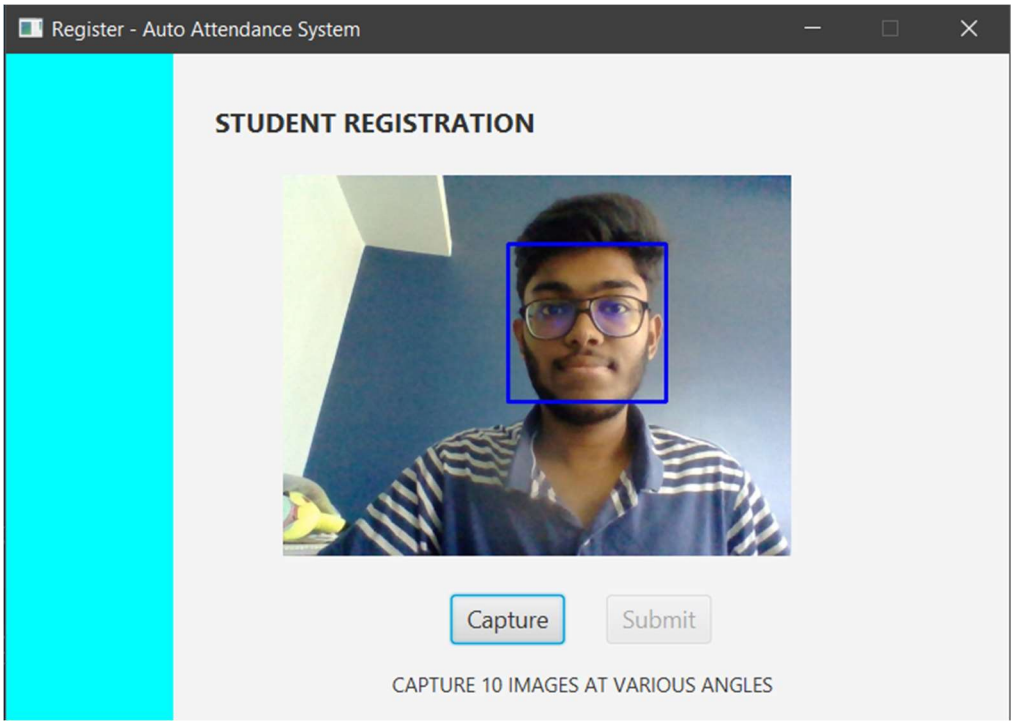
To initialize the webcam, perform face detection and capture the detected face, (from RegisterFaceController.java)

```
@Override
public void initialize(URL location, ResourceBundle resources) {
    capture = new VideoCapture(0);
    capturedCount = 0;
    capturedImages = new InputStream[10];
    thread = new DaemonThread();
    Thread t = new Thread(thread);
    t.setDaemon(true);
    thread.runnable = true;
    t.start();
    submitBtn.setDisable(true);
}

public void captureImage(ActionEvent e){
    if(capturedCount < 10) {
        if(capture.isOpened()){
            Mat capturedFrame = new Mat();
            MatOfByte capturedMem = new MatOfByte();
            capture.retrieve(capturedFrame);
            Mat faceCrop = new Mat(capturedFrame, faceCoord);
```

```
            Imgcodecs.imencode(".jpg", faceCrop, capturedMem);
            byte[] byteArray = capturedMem.toArray();
            capturedImages[capturedCount] = new
ByteArrayInputStream(byteArray);
            capturedCount += 1;
        }
    }
}

class DaemonThread implements Runnable {
    protected volatile boolean runnable = false;

    @Override
    public void run() {
        synchronized (this) {
            while(runnable) {
                if(capture.grab()){
                    try {
                        // Reading the image frame from webcam
                        capture.retrieve(frame);
                        faceDetection(frame);
                        // Encoding the image
                        Imgcodecs.imencode(".jpg", frame, mem);
                        // Storing the encoded Mat in a byte
array
                        byte[] byteArray = mem.toArray();

                        // Displaying the image
                        InputStream in = new
ByteArrayInputStream(byteArray);
                        BufferedImage image = ImageIO.read(in);
                        WritableImage writableImage =
SwingFXUtils.toFXImage(image, null);
                        regImgView.setImage(writableImage);

                        if(capturedCount == 10) {
                            submitBtn.setDisable(false);
                            captureBtn.setDisable(true);
                        }

                        if(!runnable)
                            this.wait();

                    } catch (Exception exception) {
```

```
                        exception.printStackTrace();
                    }
                }
            }
        }
    }
}
void faceDetection(Mat image) {
    // Build and load the cascade classifier
    CascadeClassifier classifier = new CascadeClassifier();

if(!classifier.load("D:\\Image_Processing\\Auto_Attendence_Track
er\\src\\main\\resources\\haarcascade_frontalface_alt.xml"))
        System.err.println("Unable to load the classifier.");
    // Covert the image to grayscale and eqaulize the frame
histogram to get better results
    Mat imageGrey = new Mat();
    Imgproc.cvtColor(image, imageGrey, Imgproc.COLOR_BGR2GRAY);
    Imgproc.equalizeHist(imageGrey, imageGrey);
    // Set minimum size of the face to detect - Here 20% of the
frame size
    int absoluteFaceSize = 0;
    int height = imageGrey.rows();
    if(Math.round(height * 0.2f) > 0)
        absoluteFaceSize = Math.round(height * 0.2f);
    // Detection using detectMultiScale function detects objects
of different sizes in the input image.
    // The detected objects are returned as a list of
rectangles.
    MatOfRect faces = new MatOfRect();
    classifier.detectMultiScale(imageGrey,faces,1.1, 2,
Objdetect.CASCADE_SCALE_IMAGE, new Size(absoluteFaceSize,
absoluteFaceSize), new Size());
    // Drawing Rectangles around detected faces
    Rect[] facesArray = faces.toArray();
    int i = 1;
    for(Rect face : facesArray) {
        // System.out.println(face); // face -> coordinates of
face location
        // Send to recognizer
        faceCoord = face;
        Mat faceCrop = new Mat(image, face);
        i += 1;
        Imgproc.rectangle(image, new Point(face.x, face.y), new
```

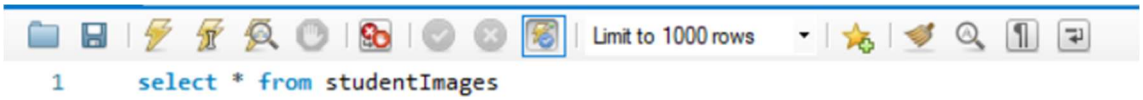
```
Point(face.x + face.width, face.y + face.height), new
Scalar(255, 0, 0), 4);
    }

}
```

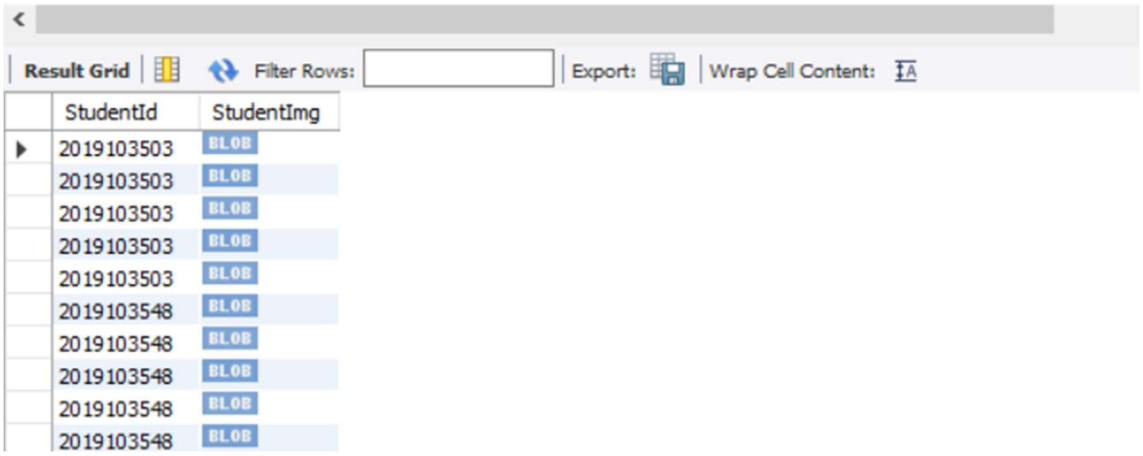


To Submit the captured images, (from RegisterFaceController.java)

```
public void submitImages(ActionEvent e){
    try {
        boolean flag = true;
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/AUTO_AT
TENDANCE", "Torvus", "Torvus@1407");

        PreparedStatement stmt = conn.prepareStatement("insert
into StudentImages values (?, ?)");
        for(InputStream in : capturedImages){
            stmt.setLong(1, this.regNum);
```

```
            stmt.setBinaryStream(2, in);
            if(stmt.executeUpdate() == 0)
                flag = false;
        }
        if(flag){
            Alert success = new
Alert(Alert.AlertType.INFORMATION);
            success.setTitle("Register status");
            success.setHeaderText("Registration Successful.");
            success.setContentText("Student : " + this.regNum +
" registered.");
            success.showAndWait();
            capture.release();
            root =
FXMLLoader.load(getClass().getResource("StudentLogin.fxml"));
            stage = (Stage) ((Node)
e.getSource()).getScene().getWindow();
            stage.getScene().setRoot(root);
            stage.setTitle("Student Login / Register - Auto
Attendance System");
            stage.show();
        } else {
            throw new Exception("Image upload failed.");
        }
    } catch(Exception exception) {
        Alert failure = new Alert(Alert.AlertType.ERROR);
        failure.setTitle("Register status");
        failure.setHeaderText("Registration failed.");
        failure.setContentText(exception.getMessage());
        failure.showAndWait();
        exception.printStackTrace();
    }

}
```





StudentId	StudentImg
2019103503	BLOB
2019103503	BLOB
2019103503	BLOB
2019103503	BLOB
2019103503	BLOB
2019103548	BLOB
2019103548	BLOB
2019103548	BLOB
2019103548	BLOB
2019103548	BLOB

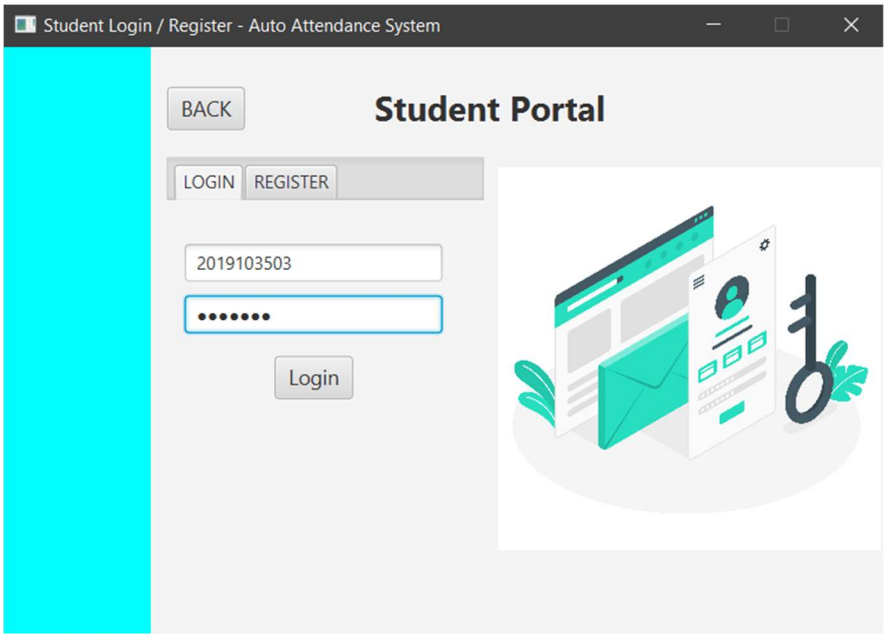
2. Logging in a registered student

After successful registration, the student is redirected to login page using his/her register number and password.

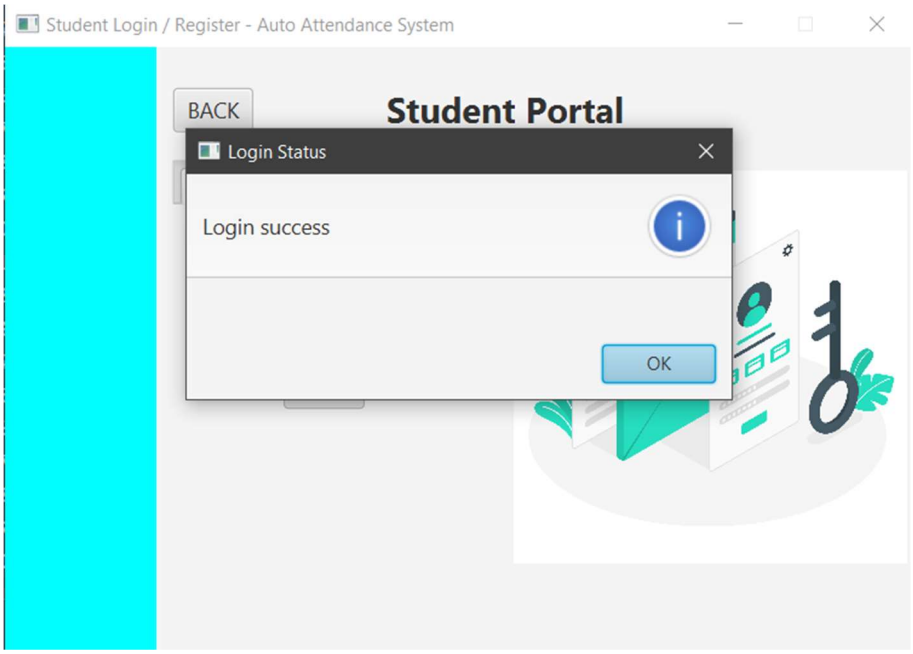
To login to the student account, (from StudentLoginController.java)

```
public void loginHandler(ActionEvent e) {
    String studentId = loginIdTxt.getText();
    String password = LoginPassTxt.getText();
    if(studentId.trim().equals("") ||
password.trim().equals("")){
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle("Required");
        alert.setContentText("Username / Password required !!");
        alert.showAndWait();
    } else {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/AUTO_AT
TENDANCE", "Torvus", "Torvus@1407");
```

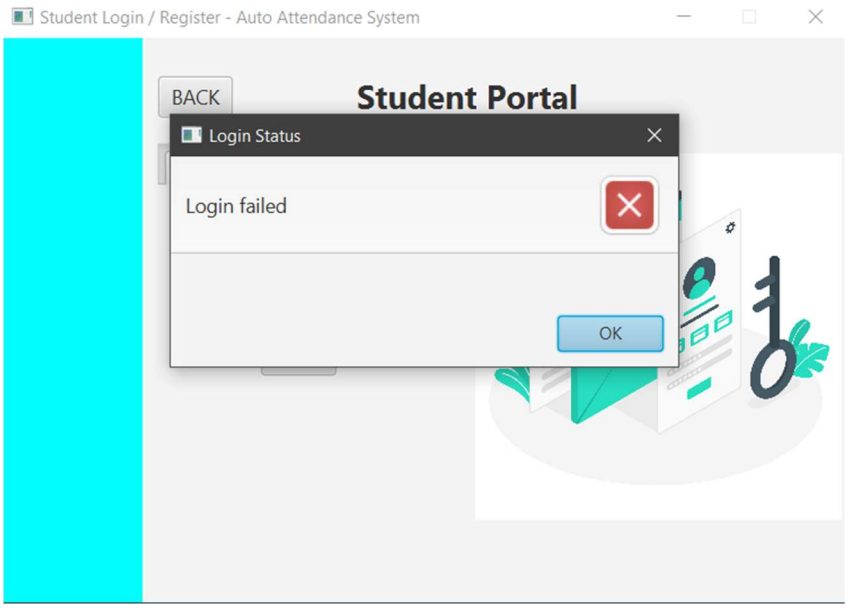
```
            PreparedStatement stmt =
con.prepareStatement("select * from Students where StudentId=?
and StudentPwd=?");
            stmt.setString(1, studentId);
            stmt.setString(2, password);

            ResultSet rs = stmt.executeQuery();
            if (rs.next()){
                Alert success = new
Alert(Alert.AlertType.INFORMATION);
                success.setTitle("Login Status");
                success.setHeaderText("Login success");
                success.showAndWait();

                FXMLLoader loader = new
FXMLLoader(getClass().getResource("StudentHome.fxml"));
                StudentHomeController homeController = new
StudentHomeController(Long.parseLong(studentId));
                loader.setController(homeController);
                root = loader.load();
                stage = (Stage) ((Node)
e.getSource()).getScene().getWindow();
                scene = new Scene(root);
                stage.setScene(scene);
                stage.setTitle("Student Home - Auto Attendance
System");
                stage.show();
            } else {
                Alert fail = new Alert(Alert.AlertType.ERROR);
                fail.setTitle("Login Status");
                fail.setHeaderText("Login failed");
                fail.showAndWait();
            }

            con.close();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```



On successful login by providing valid credentials, Login Success window is prompted and directed to student home page.



On invalid login credentials, the student is prompted with login failure alert and student has to enter valid credentials.



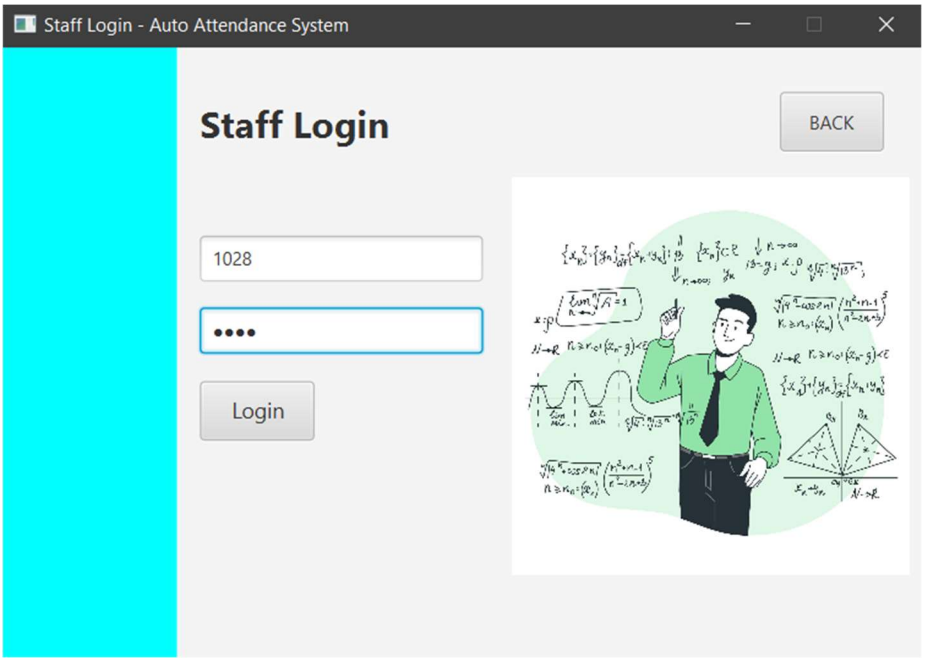
3. Staff Login

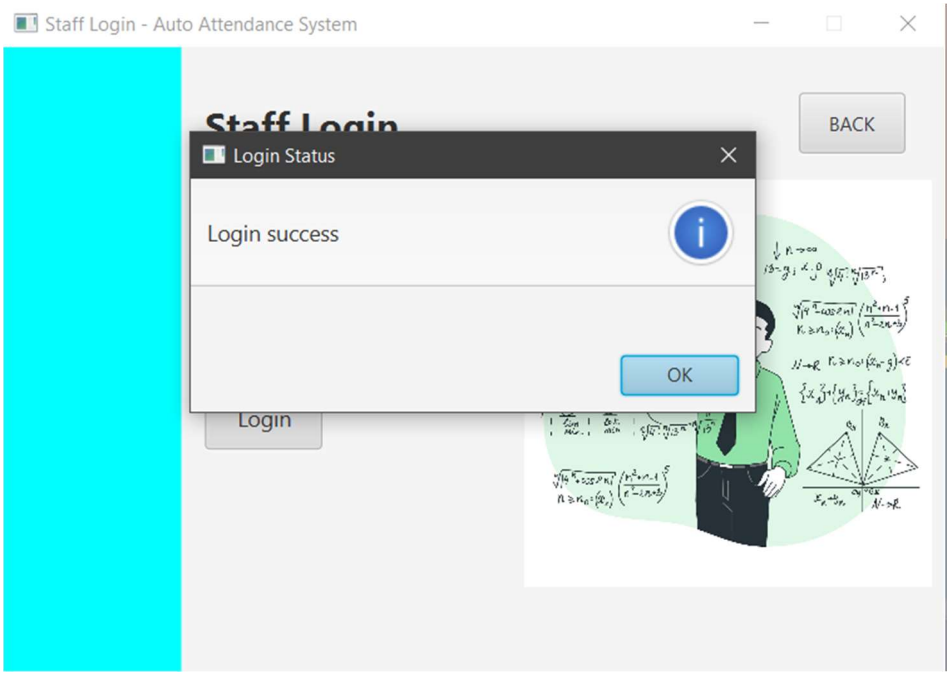
For staff login, (from StaffLoginController.java)

```
public void loginHandler(ActionEvent e) {
    String staffId = staffIdTxt.getText();
    String password = passTxt.getText();
    if(staffId.trim().equals("") || password.trim().equals("")){
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle("Required");
        alert.setContentText("Username / Password required !!");
        alert.showAndWait();
    } else {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/AUTO_AT
TENDANCE", "Torvus", "Torvus@1407");

            PreparedStatement stmt =
con.prepareStatement("select * from Staffs where StaffId=? and
StaffPwd=?");
            stmt.setString(1, staffId);
            stmt.setString(2, password);

            ResultSet rs = stmt.executeQuery();
```

```
            if (rs.next()){
                Alert success = new
Alert(Alert.AlertType.INFORMATION);
                success.setTitle("Login Status");
                success.setHeaderText("Login success");
                success.showAndWait();

                FXMLLoader loader = new
FXMLLoader(getClass().getResource("StaffHome.fxml"));
                root = loader.load();
                StaffController staffController =
loader.getController();
                staffController.getStaffId(rs.getInt(1));
                stage = (Stage) ((Node)
e.getSource()).getScene().getWindow();
                scene = new Scene(root);
                stage.setScene(scene);
                stage.setTitle("Staff Home - Auto Attendance
System");
                stage.show();
            } else {
                Alert fail = new Alert(Alert.AlertType.ERROR);
                fail.setTitle("Login Status");
                fail.setHeaderText("Login failed");
                fail.showAndWait();
            }

            con.close();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

On providing correct login credentials the staff is logged in to the Staff Home Screen.





4. Admin Login and Admin Tasks

Admin performs several attendance, student, staff and course management routines. He performs the following tasks

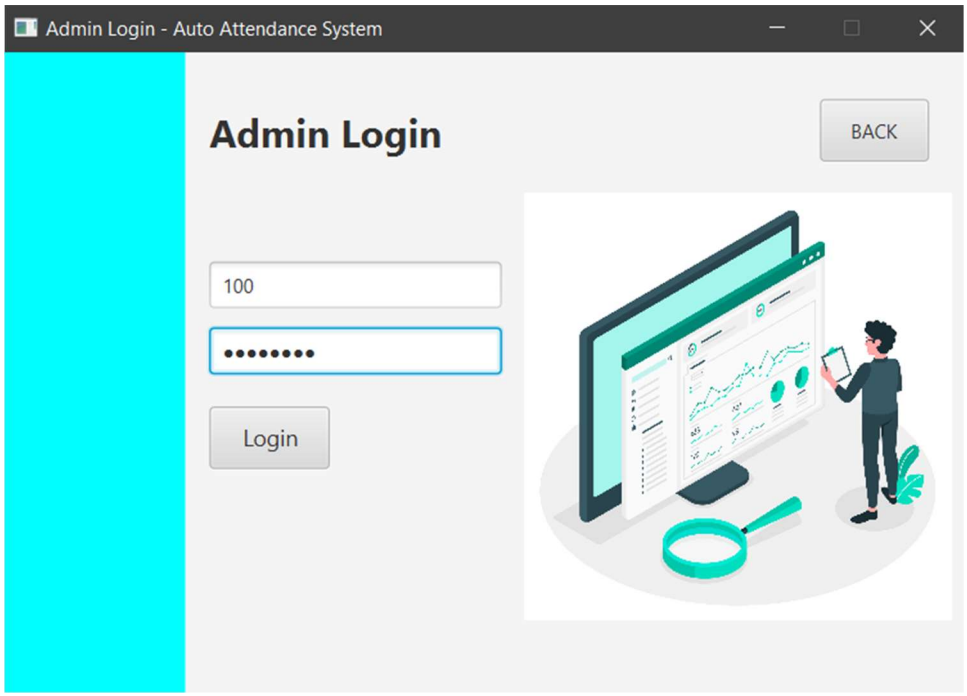
- ✓ Student Enrollment/De-enrollment

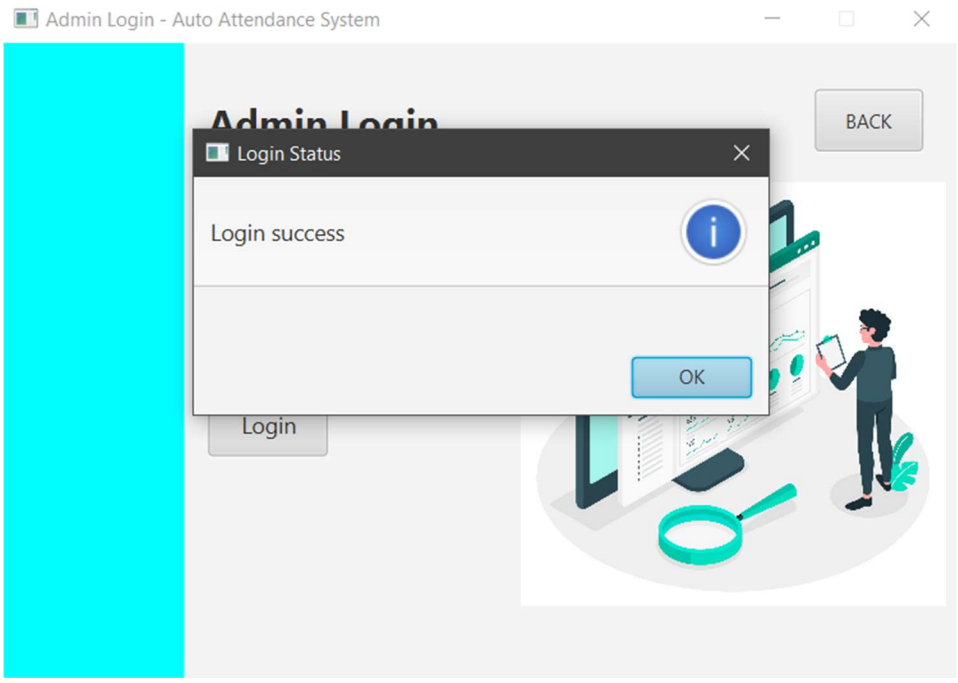
- ✓ Adding a new course
- ✓ Removing an existing course
- ✓ Add / Remove Staff
- ✓ Add / Remove courses taught by a staff

To login an admin, (from AdminLoginController.java)

```
public void loginHandler(ActionEvent e) {
    String adminId = adminIdTxt.getText();
    String password = passTxt.getText();
    if(adminId.trim().equals("") || password.trim().equals("")){
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle("Required");
        alert.setContentText("Username / Password required !!");
        alert.showAndWait();
    } else {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/AUTO_AT
TENDANCE", "Torvus", "Torvus@1407");

            PreparedStatement stmt =
con.prepareStatement("select * from Admins where AdminId=? and
AdminPwd=?");
            stmt.setString(1, adminId);
            stmt.setString(2, password);

            ResultSet rs = stmt.executeQuery();
            if (rs.next()){
                Alert success = new
Alert(Alert.AlertType.INFORMATION);
                success.setTitle("Login Status");
                success.setHeaderText("Login success");
                success.showAndWait();

                root =
FXMLLoader.load(getClass().getResource("AdminHome.fxml"));
                stage = (Stage) ((Node)
e.getSource()).getScene().getWindow();
                stage.getScene().setRoot(root);
                stage.setTitle("Admin - Auto Attendance
System");
                stage.show();
```

```
            } else {
                Alert fail = new Alert(Alert.AlertType.ERROR);
                fail.setTitle("Login Status");
                fail.setHeaderText("Login failed");
                fail.showAndWait();
            }

            con.close();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

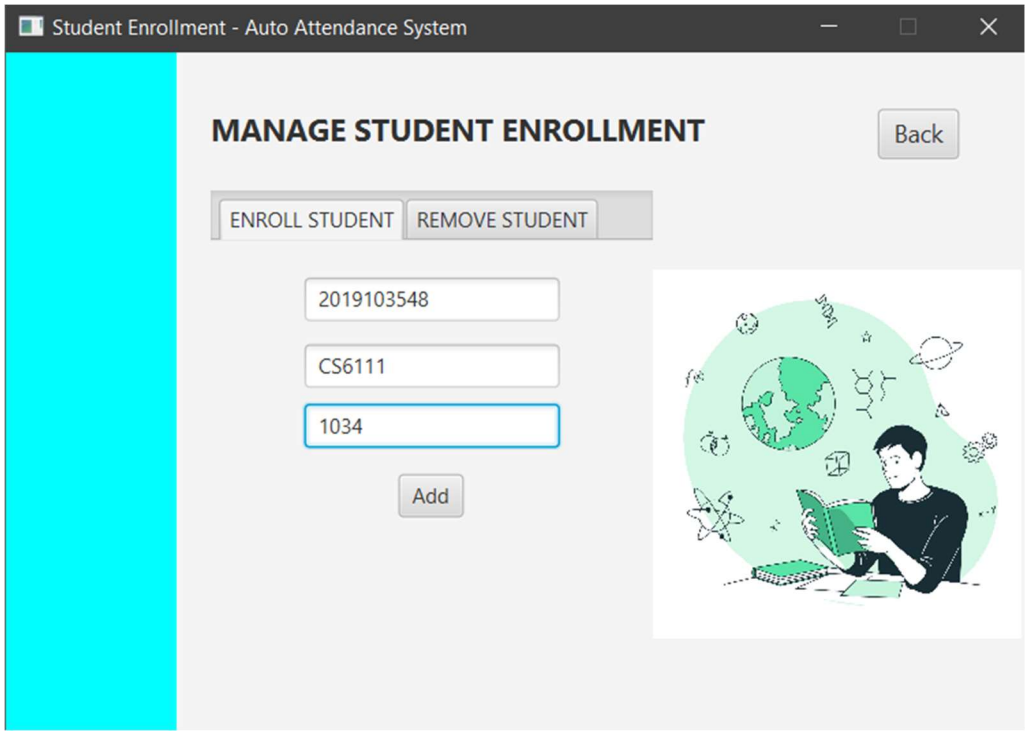




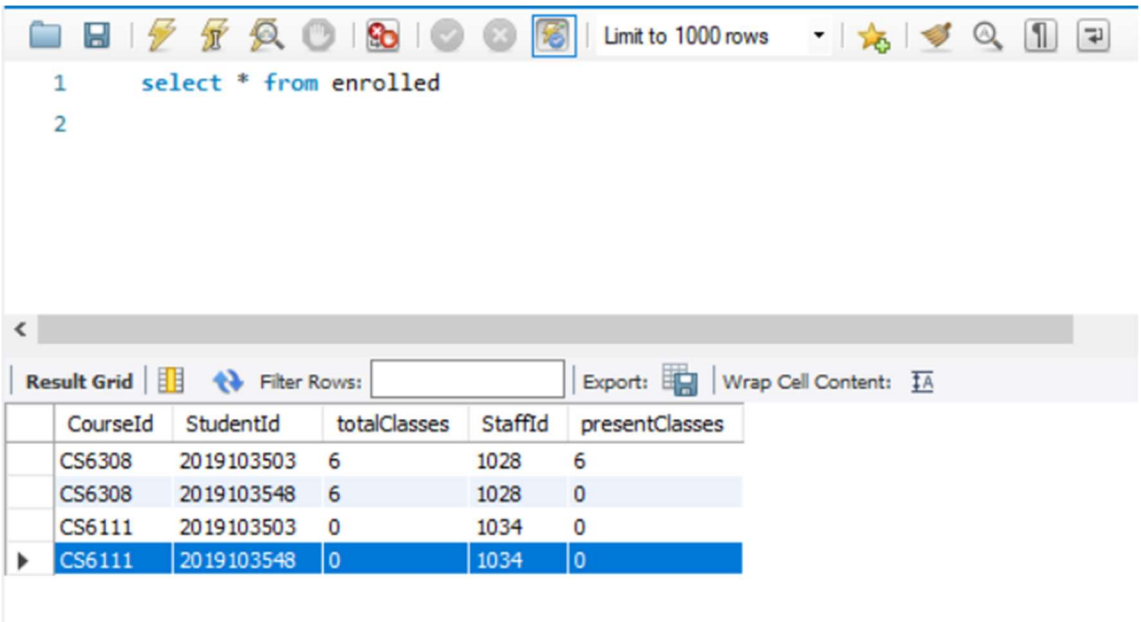
To perform Student Enrollment to a course, (from StudentEnrollmentController.java)

```
public void enrollStudentHandler (ActionEvent e) {
    String studentId = enrollStudentId.getText();
    String courseId = enrollCourseId.getText();
    String staffId = enrollStaffId.getText();

    if(studentId.trim().equals("") || courseId.trim().equals("")
|| staffId.trim().equals("")){
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle("Required");
        alert.setContentText("All fields are required");
        alert.showAndWait();
    } else {
        try {
            this.stmt = this.conn.prepareStatement("select *
from teaches where StaffId = ? and CourseId = ?");
            stmt.setInt(1, Integer.parseInt(staffId));
            stmt.setString(2, courseId);
            ResultSet rs = stmt.executeQuery();
            if(rs.next()){
                stmt = conn.prepareStatement("insert into
enrolled values (?, ?, 0, ?, 0)");
```

```
                stmt.setString(1, courseId);
                stmt.setLong(2, Long.parseLong(studentId));
                stmt.setInt(3, Integer.parseInt(staffId));
                if(stmt.executeUpdate() == 1){
                    enrollStudentId.setText("");
                    enrollCourseId.setText("");
                    enrollStaffId.setText("");
                    Alert success = new
Alert(Alert.AlertType.INFORMATION);
                    success.setTitle("Enrollment status");
                    success.setHeaderText("Student enrolled
successfully");
                    success.showAndWait();
                } else {
                    Alert failure = new
Alert(Alert.AlertType.ERROR);
                    failure.setTitle("Enrollment status");
                    failure.setHeaderText("Student enrolled
failed");
                    failure.showAndWait();
                }
            } else {
                Alert failure = new
Alert(Alert.AlertType.ERROR);
                failure.setTitle("Enrollment status");
                failure.setHeaderText("Student enrolled
failed");
                failure.setContentText("Enter correct staff and
course details");
                failure.showAndWait();
            }
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
}
```



An entry from the entered details have been created and inserted into the enrolled table, (the one highlighted in the table).

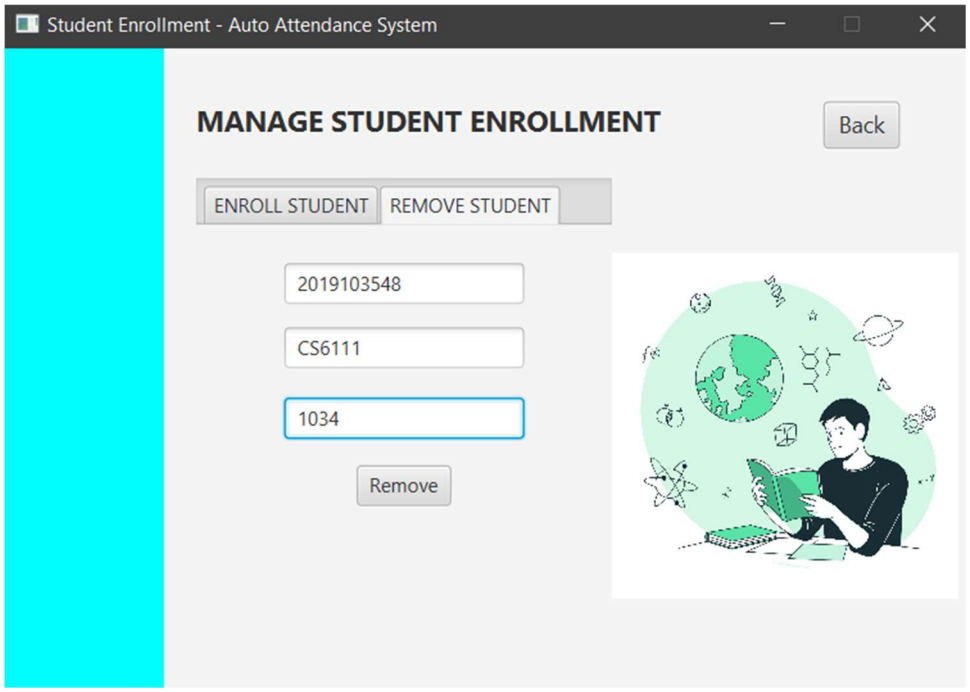


CourseId	StudentId	totalClasses	StaffId	presentClasses
CS6308	2019103503	6	1028	6
CS6308	2019103548	6	1028	0
CS6111	2019103503	0	1034	0
CS6111	2019103548	0	1034	0

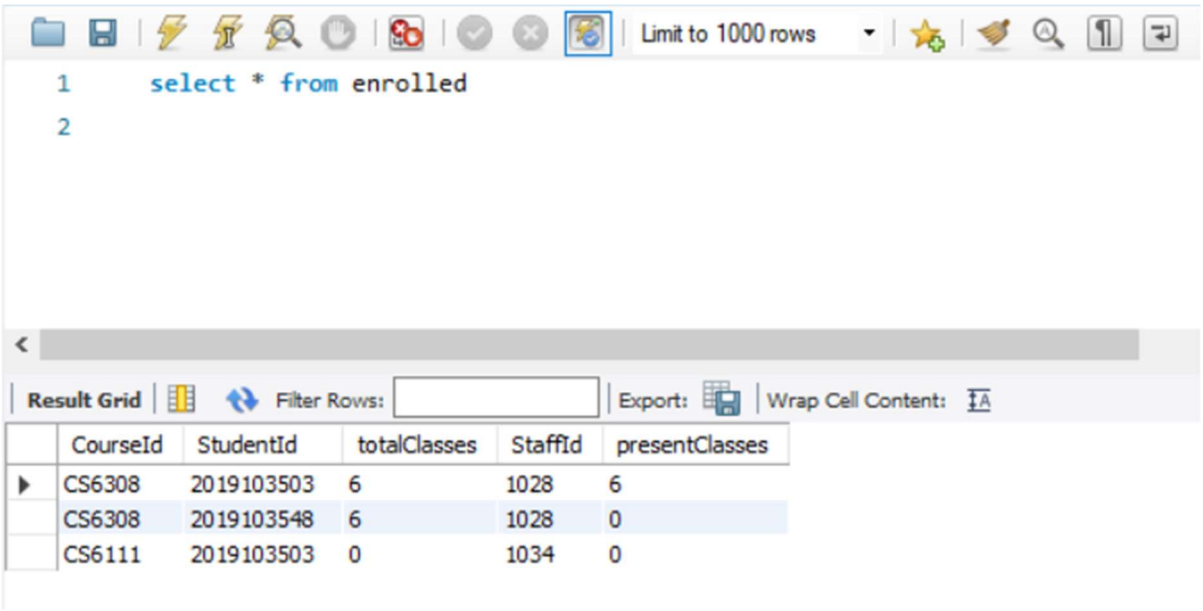
To perform Student De - enrollment to a course, (from StudentEnrollmentController.java)

```
public void removeStudentHandler(ActionEvent e) {
    String studentId = removeStudentId.getText();
    String courseId = removeCourseId.getText();
    String staffId = removeStaffId.getText();

    if(studentId.trim().equals("") || courseId.trim().equals("")
|| staffId.trim().equals("")){
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle("Required");
        alert.setContentText("All fields are required");
        alert.showAndWait();
    } else {
        try {
            this.stmt = this.conn.prepareStatement("select *
from teaches where StaffId = ? and CourseId = ?");
            stmt.setInt(1, Integer.parseInt(staffId));
            stmt.setString(2, courseId);
            ResultSet rs = stmt.executeQuery();
            if(rs.next()){
                stmt = conn.prepareStatement("delete from
enrolled where CourseId = ? and StudentId = ? and StaffId = ?");
                stmt.setString(1, courseId);
                stmt.setLong(2, Long.parseLong(studentId));
                stmt.setInt(3, Integer.parseInt(staffId));
                if(stmt.executeUpdate() == 1){
                    removeStudentId.setText("");
                    removeCourseId.setText("");
                    removeStaffId.setText("");
                    Alert success = new
Alert(Alert.AlertType.INFORMATION);
                    success.setTitle("Disenrollment status");
                    success.setHeaderText("Student disenrolled
successfully");
                    success.showAndWait();
                } else {
                    Alert failure = new
Alert(Alert.AlertType.ERROR);
                    failure.setTitle("Disenrollment status");
                    failure.setHeaderText("Student disenrolled
failed");
                    failure.showAndWait();
```

```
                }
            } else {
                Alert failure = new
Alert(Alert.AlertType.ERROR);
                failure.setTitle("Disenrollment status");
                failure.setHeaderText("Student disenrolled
failed");
                failure.setContentText("Enter correct staff and
course details");
                failure.showAndWait();
            }
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
```



The entry with the entered details are removed successfully from the enrolled table.



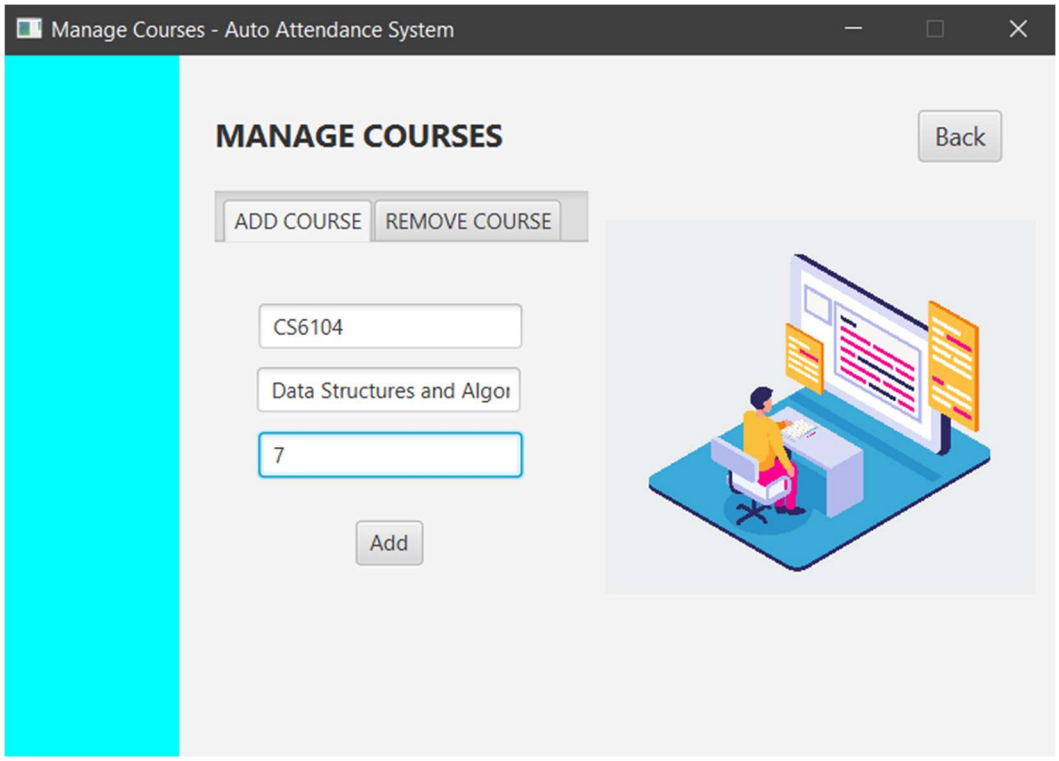
CourseId	StudentId	totalClasses	StaffId	presentClasses
CS6308	2019103503	6	1028	6
CS6308	2019103548	6	1028	0
CS6111	2019103503	0	1034	0

To manage courses provided, ie, add / remove courses, (from ManageCoursesController.java)

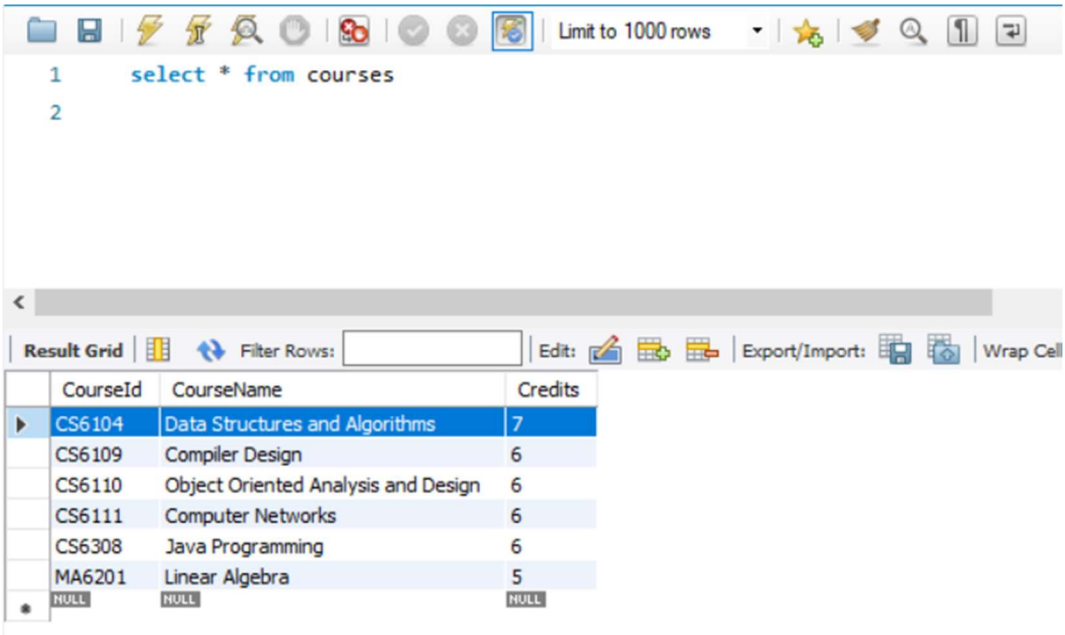
```
public void addCourseHandler(ActionEvent event) {
    String courseId = addCourseId.getText();
    String courseName = addCourseName.getText();
    String credits = addCredits.getText();

    if(courseName.trim().equals("") ||
courseId.trim().equals("") || credits.trim().equals("")){
        Alert error = new Alert(Alert.AlertType.WARNING);
        error.setTitle("Course status");
        error.setHeaderText("Required");
        error.setContentText("Enter CourseId, Name and
credits.");
        error.showAndWait();
    } else {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/AUTO_AT
TENDANCE", "Torvus", "Torvus@1407");
            PreparedStatement stmt =
con.prepareStatement("insert into courses values (?, ?, ?)");
            stmt.setString(1, courseId);
            stmt.setString(2, courseName);
            stmt.setInt(3, Integer.parseInt(credits));
            stmt.executeUpdate();
```

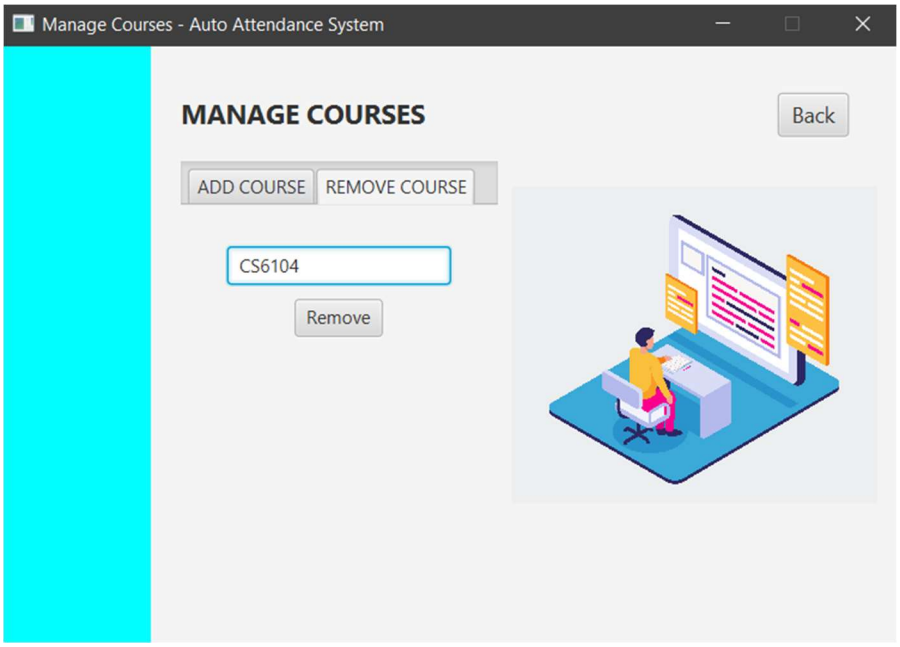
```
            stmt = con.prepareStatement("select * from courses
where courseId = ?");
            stmt.setString(1, courseId);
            ResultSet rs = stmt.executeQuery();
            if(rs.next()){
                addCourseId.setText("");
                addCourseName.setText("");
                addCredits.setText("");
                Alert success = new
Alert(Alert.AlertType.INFORMATION);
                success.setTitle("Course status");
                success.setHeaderText("Course added
successfully");
                success.setContentText("Course - " +
rs.getString(1) + " created");
                success.showAndWait();
            } else {
                Alert failure = new
Alert(Alert.AlertType.ERROR);
                failure.setTitle("Course status");
                failure.setHeaderText("Add course failed");
                failure.setContentText("Unable to create the
course");
                failure.showAndWait();
            }
        } catch (Exception exception) {
            exception.printStackTrace();
        }

    }
}
public void removeCourseHandler(ActionEvent event) {
    String courseId = removeCourseId.getText();
    if(courseId.trim().equals("")){
        Alert error = new Alert(Alert.AlertType.WARNING);
        error.setTitle("Course status");
        error.setHeaderText("Required");
        error.setContentText("Enter CourseId");
        error.showAndWait();
    } else {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con =
```

```
DriverManager.getConnection("jdbc:mysql://localhost:3306/AUTO_AT
TENDANCE", "Torvus", "Torvus@1407");
            PreparedStatement stmt =
con.prepareStatement("delete from courses where CourseId = ?");
            stmt.setString(1, courseId);
            stmt.executeUpdate();
            stmt = con.prepareStatement("select * from courses
where CourseId = ?");
            stmt.setString(1, courseId);
            ResultSet rs = stmt.executeQuery();
            if(rs.next()){
                Alert failure = new
Alert(Alert.AlertType.ERROR);
                failure.setTitle("Course status");
                failure.setHeaderText("Course remove failed");
                failure.showAndWait();
            } else {
                removeCourseId.setText("");
                Alert success = new
Alert(Alert.AlertType.INFORMATION);
                success.setTitle("Course status");
                success.setHeaderText("Course removed
successfully");
                success.showAndWait();
            }
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
}
```

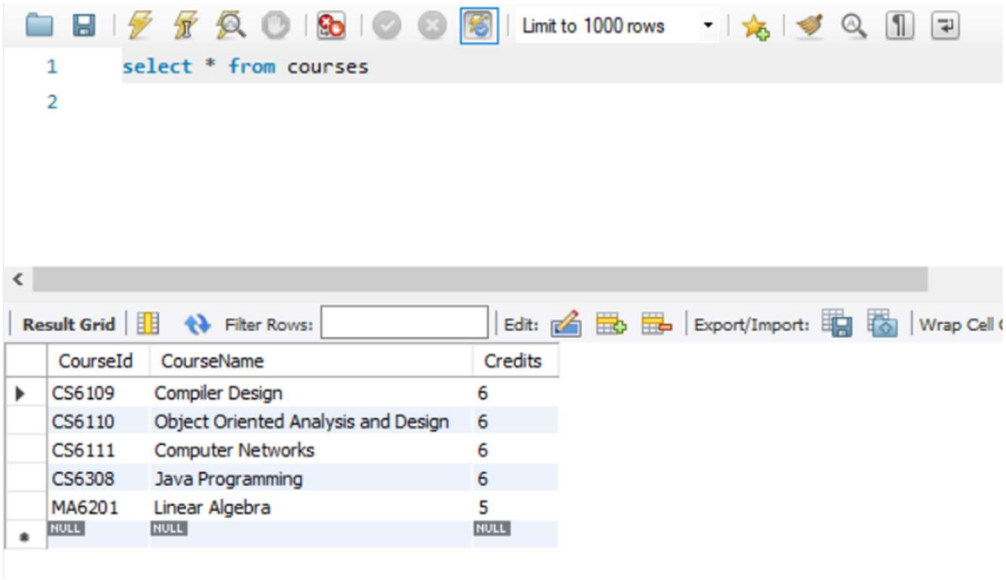


A new entry with the entered details is inserted into the courses table successfully.





The course entry with the entered details is removed from the courses table successfully.

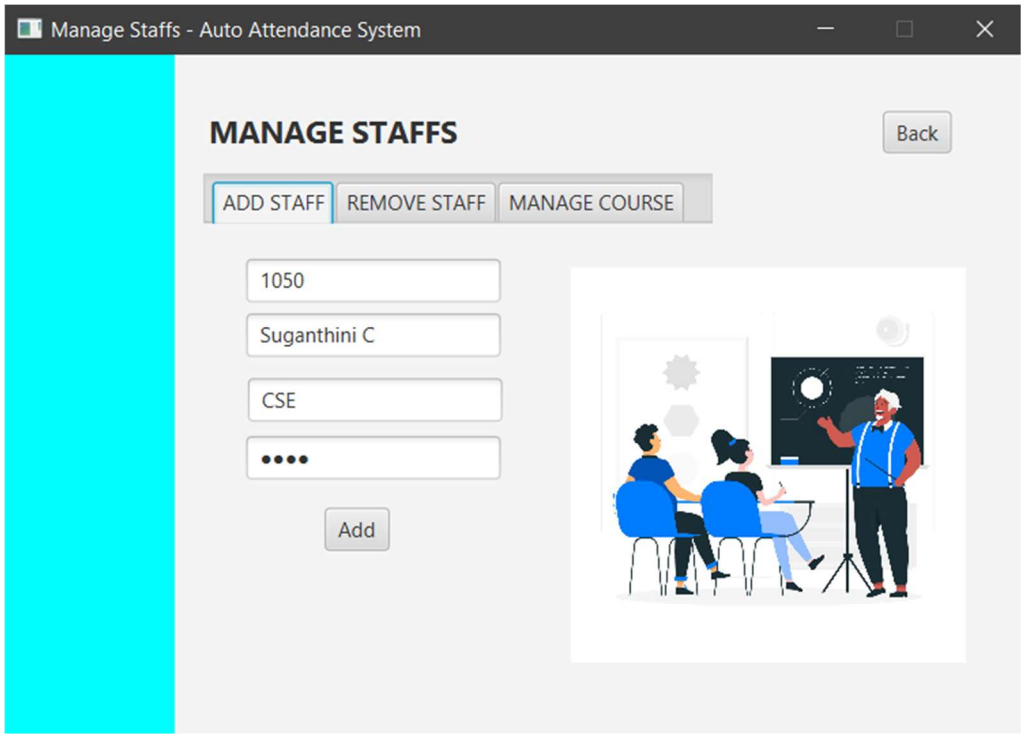


To add/ remove a staff from the staffs table, (from ManageStaffsController.java)

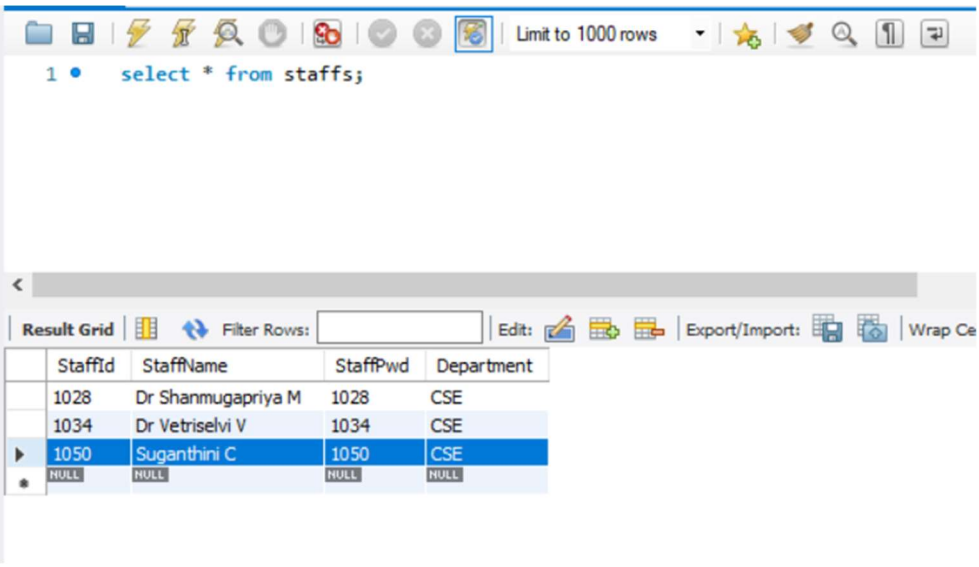
```
public void addStaffHandler(ActionEvent event) {
    try {
        String staffId = addStaffId.getText();
        String staffName = addStaffName.getText();
        String staffPwd = addStaffPwd.getText();
        String staffDept = addDept.getText();

        if (staffId.trim().equals("") ||
staffName.trim().equals("") || staffPwd.trim().equals("") ||
staffDept.trim().equals("")){
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setTitle("Required");
            alert.setHeaderText("All fields are required");
            alert.showAndWait();
        } else {
            this.stmt = this.conn.prepareStatement("insert into
staffs values (?, ?, ?, ?)");
            this.stmt.setInt(1, Integer.parseInt(staffId));
            this.stmt.setString(2, staffName);
            this.stmt.setString(3, staffPwd);
            this.stmt.setString(4, staffDept);
            if(this.stmt.executeUpdate() == 1){
                addStaffId.setText("");
                addStaffName.setText("");
                addStaffPwd.setText("");
                addDept.setText("");

                Alert alert = new
Alert(Alert.AlertType.INFORMATION);
                alert.setTitle("Add Staff Status");
                alert.setHeaderText("Staff " + staffId + " added
successfully.");
                alert.showAndWait();
            } else {
                Alert alert = new Alert(Alert.AlertType.WARNING);
                alert.setTitle("Add Staff Status");
                alert.setHeaderText("Unable to add staff");
                alert.showAndWait();
            }
        }
    } catch (SQLDataException se) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Add Staff Status");
        alert.setHeaderText(se.getMessage());
```

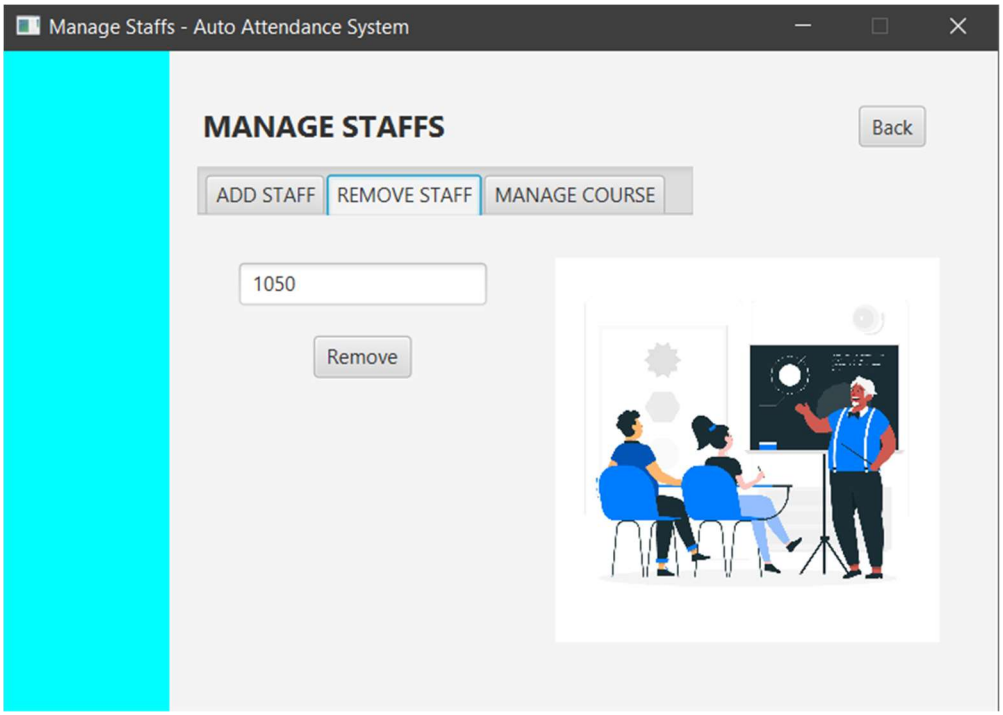
```
        alert.showAndWait();
    } catch (Exception  e) {
        e.printStackTrace();
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Add Staff Status");
        alert.setHeaderText(e.getMessage());
        alert.showAndWait();
    }
}

public void removeStaffHandler(ActionEvent event) {
    try {
        String staffId = removeStaffId.getText();

        if (staffId.trim().equals("")){
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setTitle("Required");
            alert.setHeaderText("All fields are required");
            alert.showAndWait();
        } else {
            this.stmt = this.conn.prepareStatement("delete from
staffs where StaffId = ?");
            this.stmt.setInt(1, Integer.parseInt(staffId));
            if(this.stmt.executeUpdate() == 1){
                removeStaffId.setText("");

                Alert alert = new
Alert(Alert.AlertType.INFORMATION);
                alert.setTitle("Remove Staff Status");
                alert.setHeaderText("Staff " + staffId + " removed
successfully.");
                alert.showAndWait();
            } else {
                Alert alert = new Alert(Alert.AlertType.WARNING);
                alert.setTitle("Remove Staff Status");
                alert.setHeaderText("Unable to remove staff");
                alert.showAndWait();
            }
        }
    } catch (SQLDataException se) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Remove Staff Status");
        alert.setHeaderText(se.getMessage());
        alert.showAndWait();
    } catch (Exception  e) {
        e.printStackTrace();
```

```
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Remove Staff Status");
            alert.setHeaderText(e.getMessage());
            alert.showAndWait();
        }
}
```

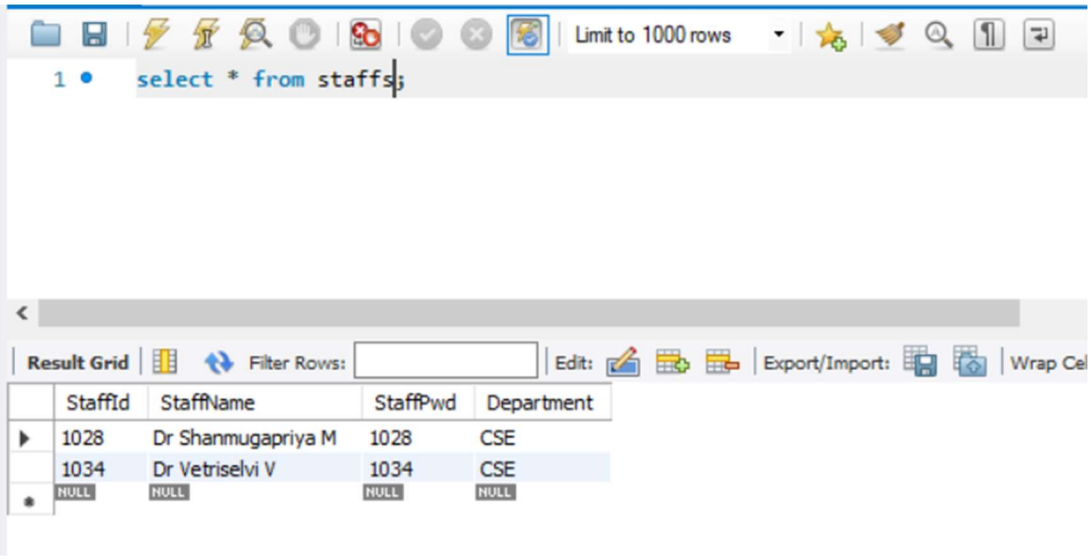


A new staff record with entered details is inserted into the staffs table successfully.





The staff record with the entered staffId is removed from the staffs table successfully.

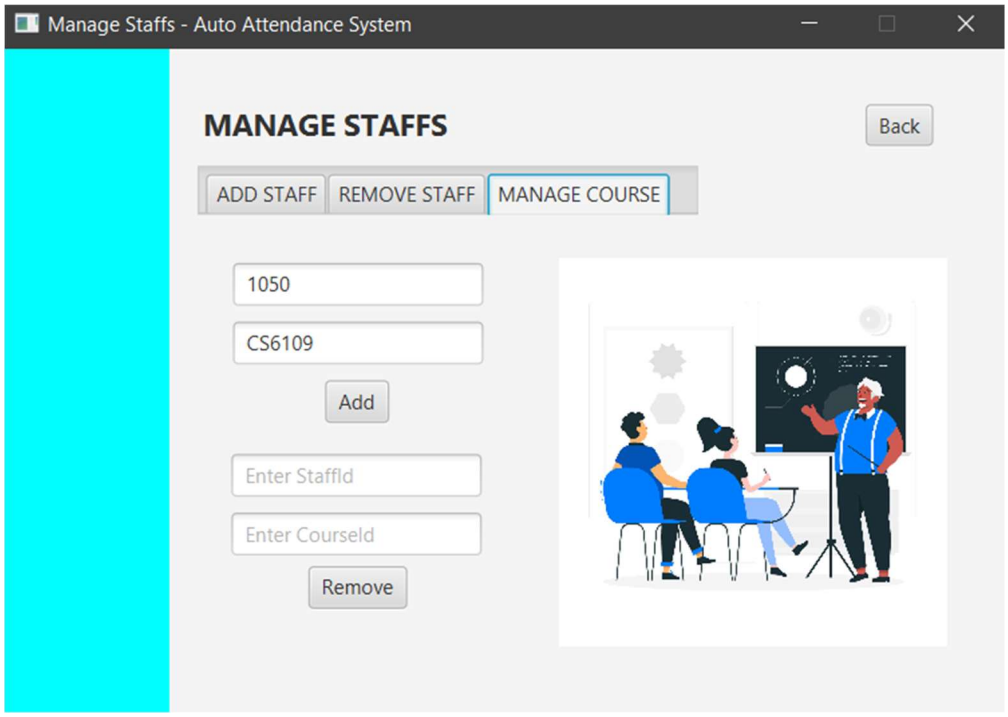


To manage the courses taught by a staff, (from ManageStaffsController.java)

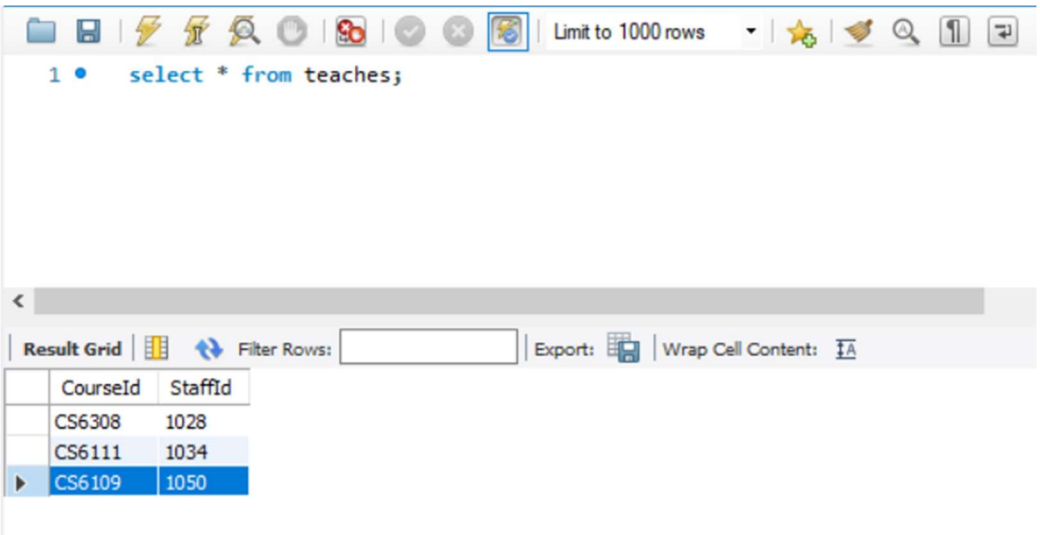
```
public void addCourseHandler(ActionEvent event){
    try {
        String staffId = addCourseStaffId.getText();
        String courseId = addCourseId.getText();

        if (staffId.trim().equals("") ||
courseId.trim().equals("")){
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setTitle("Required");
            alert.setHeaderText("All fields are required");
            alert.showAndWait();
        } else {
            this.stmt = this.conn.prepareStatement("insert into
teaches values (?, ?)");
            this.stmt.setInt(2, Integer.parseInt(staffId));
            this.stmt.setString(1, courseId);
            if(this.stmt.executeUpdate() == 1){
                addCourseId.setText("");
                addCourseStaffId.setText("");

                Alert alert = new
Alert(Alert.AlertType.INFORMATION);
                alert.setTitle("Add Course Status");
                alert.setHeaderText("Course added successfully.");
                alert.showAndWait();
            } else {
                Alert alert = new Alert(Alert.AlertType.WARNING);
                alert.setTitle("Add Course Status");
                alert.setHeaderText("Unable to add course");
                alert.showAndWait();
            }
        }
    } catch (SQLDataException se) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Add Course Status");
        alert.setHeaderText(se.getMessage());
        alert.showAndWait();
    } catch (Exception  e) {
        e.printStackTrace();
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Add Course Status");
        alert.setHeaderText(e.getMessage());
        alert.showAndWait();
```

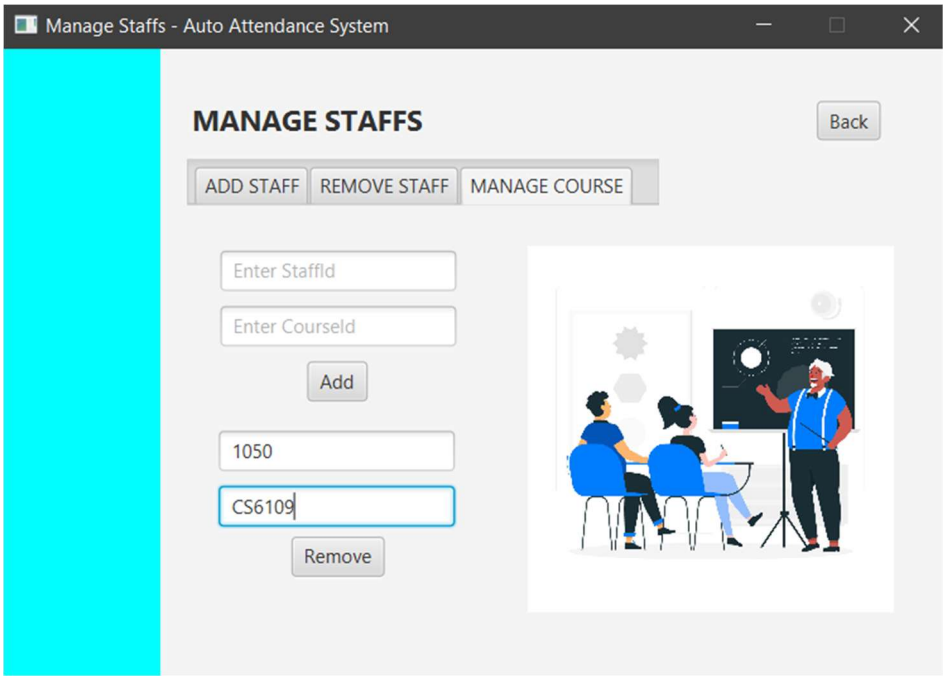
```
    }
}

public void removeCourseHandler(ActionEvent event) {
    try {
        String staffId = removeCourseStaffId.getText();
        String courseId = removeCourseId.getText();

        if (staffId.trim().equals("") ||
courseId.trim().equals("")){
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setTitle("Required");
            alert.setHeaderText("All fields are required");
            alert.showAndWait();
        } else {
            this.stmt = this.conn.prepareStatement("delete from
teaches where CourseId = ? and StaffId = ?");
            this.stmt.setInt(2, Integer.parseInt(staffId));
            this.stmt.setString(1, courseId);
            if(this.stmt.executeUpdate() == 1){
                removeCourseStaffId.setText("");
                removeCourseId.setText("");

                Alert alert = new
Alert(Alert.AlertType.INFORMATION);
                alert.setTitle("Remove Course Status");
                alert.setHeaderText("Course removed
successfully.");
                alert.showAndWait();
            } else {
                Alert alert = new Alert(Alert.AlertType.WARNING);
                alert.setTitle("Remove Course Status");
                alert.setHeaderText("Unable to remove course");
                alert.showAndWait();
            }
        }
    } catch (SQLDataException se) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Remove Course Status");
        alert.setHeaderText(se.getMessage());
        alert.showAndWait();
    } catch (Exception  e) {
        e.printStackTrace();
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Remove Course Status");
        alert.setHeaderText(e.getMessage());
```

```
            alert.showAndWait();
        }
}
```

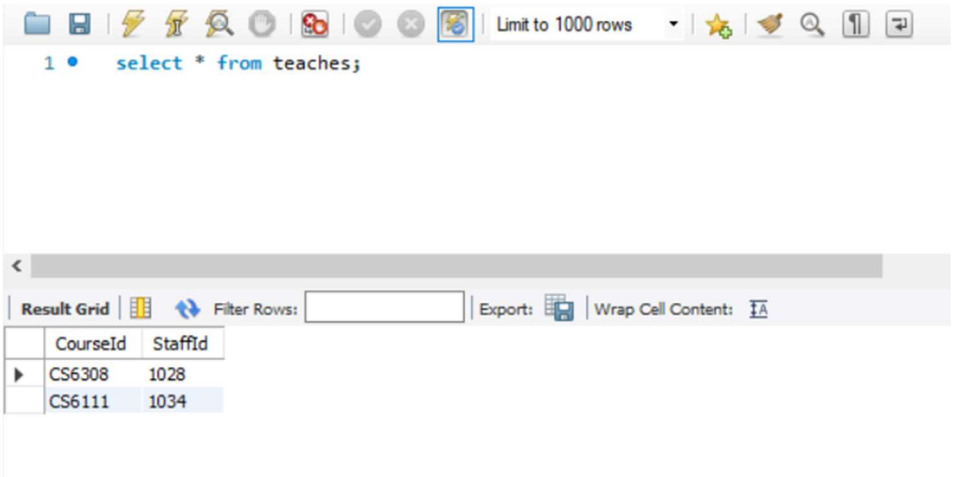


A new entry in the teaches table with the entered staffId and courseId is inserted successfully.





The entry in the teaches table with the entered staffId and courseId is removed successfully.



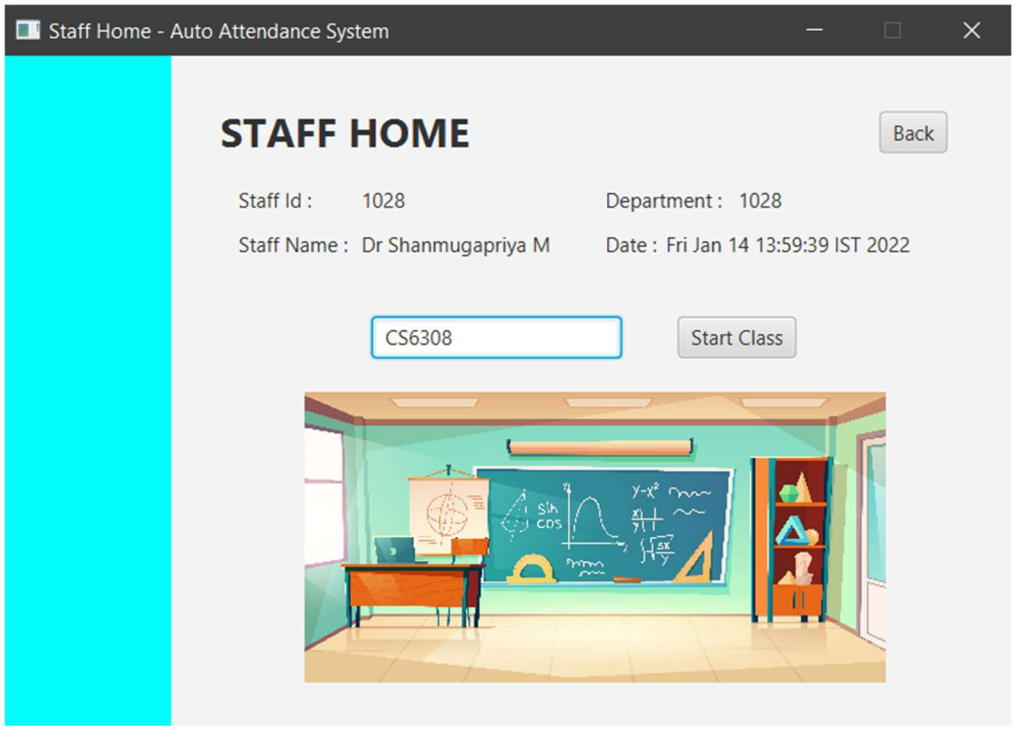
5. Creating a class session by Staff

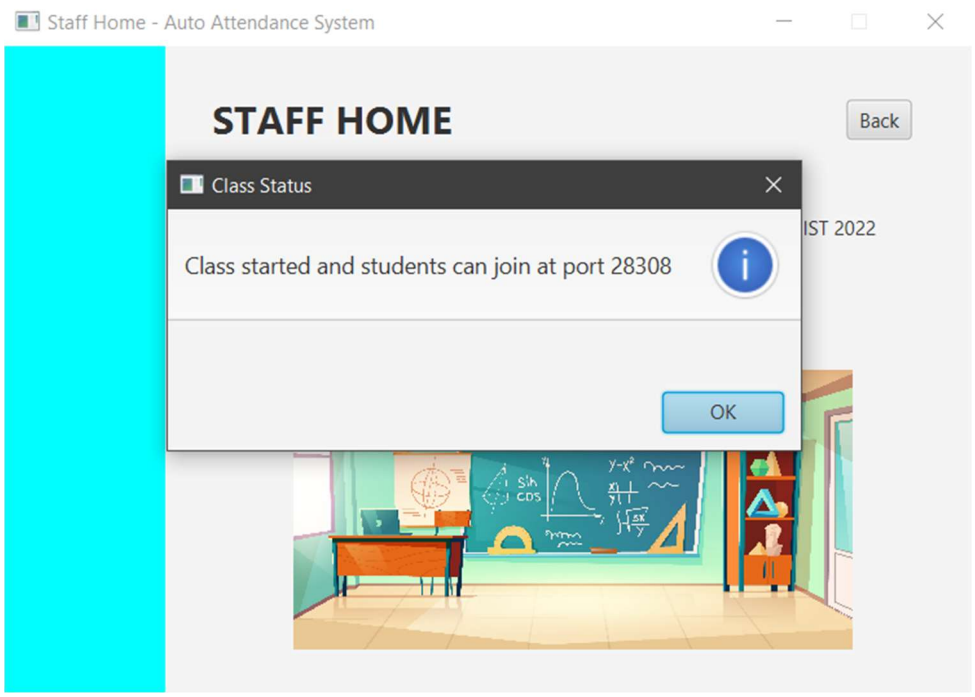
After logged in, a staff creates the class session by entering the course id. A new ServerSocket is created at the port number obtained by combining the staff and course id, thereby creating a unique port number for students to join. Then, the

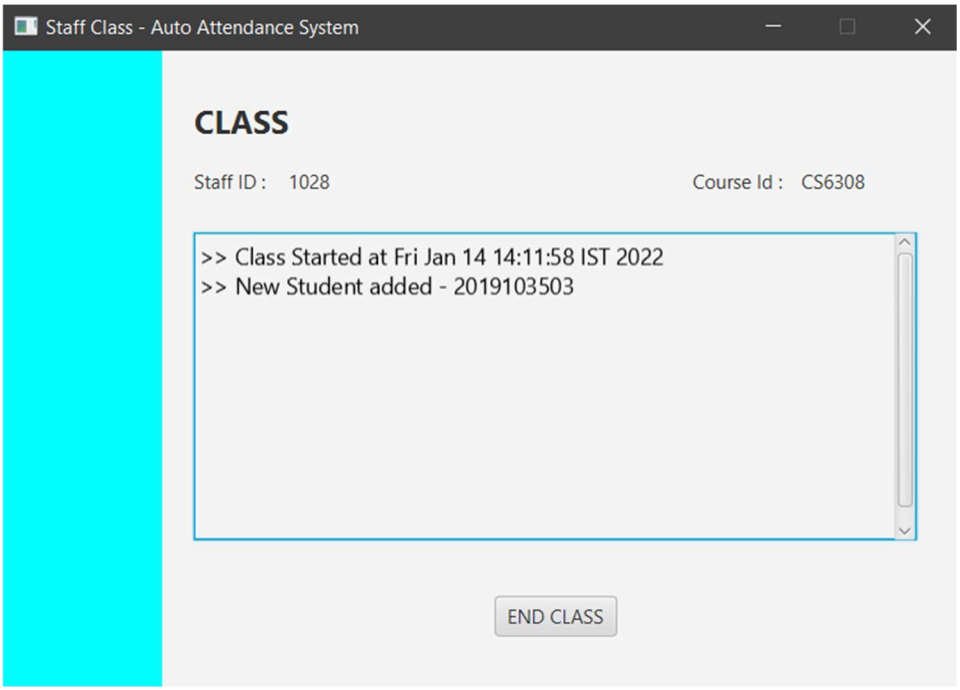
staff is directed to the Staff class session screen, which displays the class details, time of class commencement and the students joining the class.

To create a class session, (from StaffController.java)

```
public void startClassHandler(ActionEvent e) {
    try {
        String course = courseId.getText();
        if(course.trim().equals("")){
            Alert warning = new Alert(Alert.AlertType.WARNING);
            warning.setTitle("Class Status");
            warning.setHeaderText("Enter the course id of the
class.");
            warning.showAndWait();
        } else {
            Alert success = new
Alert(Alert.AlertType.INFORMATION);
            success.setTitle("Class Status");
            success.setHeaderText("Class started and students
can join at port " + Integer.parseInt((staffId % 100) +
course.substring(3)));
            success.showAndWait();

            FXMLLoader loader = new
FXMLLoader(getClass().getResource("StaffClass.fxml"));
            StaffClassSessionController classController = new
StaffClassSessionController(this.staffId, course);
            loader.setController(classController);
            root = loader.load();
            stage = (Stage) ((Node)
e.getSource()).getScene().getWindow();
            scene = new Scene(root);
            stage.setScene(scene);
            stage.setTitle("Staff Class - Auto Attendance
System");
            stage.show();
        }
    } catch(Exception exception) {
        exception.printStackTrace();
    }

}
```


Staff Home - Auto Attendance System
STAFF HOME
Back
Staff Id : 1028
Department : 1028
Staff Name : Dr Shanmugapriya M
Date : Fri Jan 14 13:59:39 IST 2022
CS6308
Start Class


Staff Home - Auto Attendance System
STAFF HOME
Back
Class Status
Class started and students can join at port 28308
OK
IST 2022



Staff class session screen displaying the class details and the time of class commencement and students joined.

6. Student joining the class session

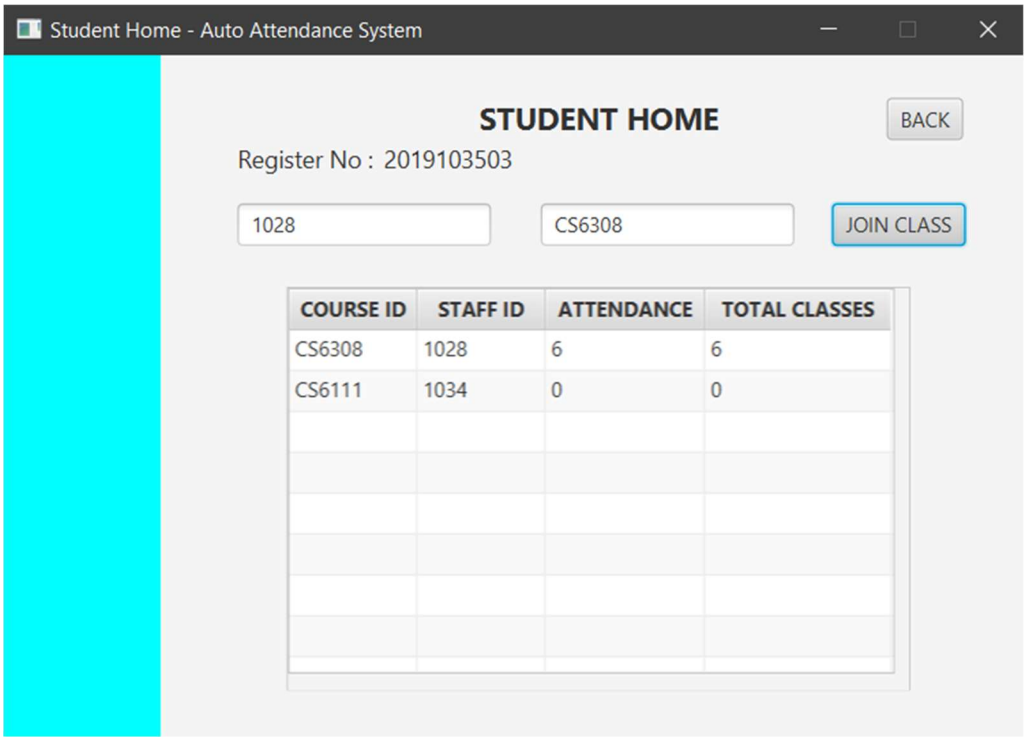
Students, after logging in, can join the class session by entering the staffId and courseId once after the staff created and started the classs session.

A Socket is created and requests the Serversocket of the staff to join the class session. To join a class session, (from StudentHomeController.java)

```
public void joinClassHandler (ActionEvent e){
    try {
        if(staffIdTxt.getText().trim().equals("") ||
courseIdTxt.getText().trim().equals("")){
            Alert warning = new Alert(Alert.AlertType.WARNING);
            warning.setTitle("Class Status");
            warning.setHeaderText("Enter the staff id and course
id of the class.");
            warning.showAndWait();
        } else {
            this.courseId = courseIdTxt.getText();
            this.staffId =
Integer.parseInt(staffIdTxt.getText());
```

```
            this.stmt = this.con.prepareStatement("select * from
teaches where CourseId = ? and StaffId = ?");
            this.stmt.setString(1, this.courseId);
            this.stmt.setInt(2, this.staffId);

            ResultSet rs = this.stmt.executeQuery();
            if(rs.next()){
                FXMLLoader loader = new
FXMLLoader(getClass().getResource("ClassSession.fxml"));
                ClassSessionController classController = new
ClassSessionController(this.regNum, this.staffId, this.courseId,
new Socket("localhost",Integer.parseInt((this.staffId % 100) +
this.courseId.substring(3))));
                loader.setController(classController);
                root = loader.load();
                stage = (Stage) ((Node)
e.getSource()).getScene().getWindow();
                scene = new Scene(root);
                stage.setScene(scene);
                stage.setTitle("Class - Auto Attendance
System");
                stage.show();
            } else {
                Alert warning = new
Alert(Alert.AlertType.WARNING);
                warning.setTitle("Class Status");
                warning.setHeaderText("Enter the valid staff id
and corresponding course id of the staff.");
                warning.showAndWait();
            }
        }
    } catch (BindException be) {
        be.printStackTrace();
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Class Status");
        alert.setContentText( be.getMessage() + "\nPort : " +
Integer.parseInt((this.staffId % 100) +
this.courseId.substring(3)) );
        alert.showAndWait();

    }  catch (ConnectException ce) {
        ce.printStackTrace();
        Alert warning = new Alert(Alert.AlertType.ERROR);
        warning.setTitle("Class Status");
```

```
            warning.setHeaderText(ce.getMessage());
            warning.showAndWait();

        }  catch (Exception exception) {
            exception.printStackTrace();
        }
}
```



Once the student joined the class session, the webcam starts capturing the student image and performs face detection and recognition. The register number of the captured student is predicted and it is sent to the staff every second, where the attendance of each students are maintained. Finally the student can leave the meeting, thereby marking his attendance at the staff end.

To capture student image and perform face detection and recognition, sending the predicted student register number to the staff, (from ClassSessionController.java)

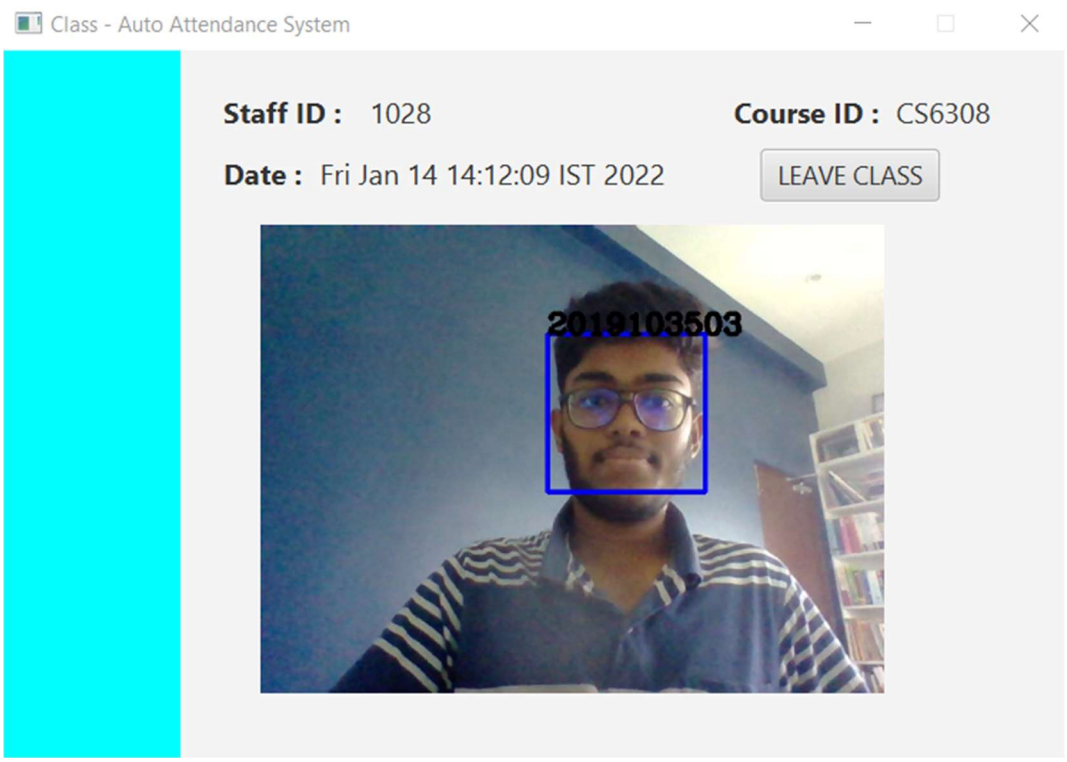
```
@Override
public void initialize(URL location, ResourceBundle resources) {
    try {
        this.output = new
ObjectOutputStream(socket.getOutputStream());
        this.output.writeObject(new Data(this.regNum));
```

```
        this.courseLabel.setText(courseId);
        this.staffLabel.setText(String.valueOf(staffId));
        this.dateLabel.setText(new Date().toString());

        capture = new VideoCapture(0);
        thread = new DaemonThread();
        Thread t = new Thread(thread);
        t.setDaemon(true);
        thread.runnable = true;
        t.start();
        ArrayList<Mat> images = new ArrayList<>();
        ArrayList<Long> labels = new ArrayList<>();

        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/AUTO_AT
TENDANCE", "Torvus", "Torvus@1407");
        PreparedStatement stmt = con.prepareStatement("select *
from StudentImages");
        ResultSet rs = stmt.executeQuery();
        while(rs.next()){
            InputStream in = rs.getBinaryStream(2);
            Long regNum = rs.getLong(1);
            int nRead;
            byte[] data = new byte[16 * 1024];
            ByteArrayOutputStream buffer = new
ByteArrayOutputStream();
            while ((nRead = in.read(data, 0, data.length)) != -
1) {
                buffer.write(data, 0, nRead);
            }
            byte[] bytes = buffer.toByteArray();
            Mat mat = Imgcodecs.imdecode(new MatOfByte(bytes),
IMREAD_UNCHANGED);
            Mat grey = new Mat();
            Imgproc.cvtColor(mat, grey, Imgproc.COLOR_RGB2GRAY);
            images.add(grey);
            labels.add(regNum);
        }
        int n = labels.size();
        Mat labelsMat = new Mat(n, 1, CvType.CV_32SC1);
        for(int i = 0; i < n; i++)
            labelsMat.put(i, 0, labels.get(i));
```

```
        model = LBPHFaceRecognizer.create();
        model.train(images, labelsMat);

        Runnable send = new sendData();
        sendThread = new Thread(send);
        sendThread.start();
    } catch (Exception e) {
        e.printStackTrace();
    }

}

public void leaveClass(ActionEvent e) {
    try {
        output.writeObject(new Data(-1));
        input = new ObjectInputStream(socket.getInputStream());
        Data inpData = (Data) input.readObject();
        if(inpData.predictedRegNum == -1) {
            System.out.println("Left class");
            socket.close();
        }
        capture.release();
        switchHome(e);
    } catch (Exception exception) {
        exception.printStackTrace();
    }
}
class DaemonThread implements Runnable {
    protected volatile boolean runnable = false;

    @Override
    public void run() {
        synchronized (this) {
            while(runnable) {
                if(capture.grab()){
                    try {
                        // Reading the image frame from webcam
                        capture.retrieve(frame);
                        faceDetection(frame);
                        // Encoding the image
                        Imgcodecs.imencode(".jpg", frame, mem);
                        // Storing the encoded Mat in a byte
array
```

```
                        byte[] byteArray = mem.toArray();

                        // Displaying the image
                        InputStream in = new
ByteArrayInputStream(byteArray);
                        BufferedImage image = ImageIO.read(in);
                        WritableImage writableImage =
SwingFXUtils.toFXImage(image, null);
                        camImgView.setImage(writableImage);

                        if(!runnable)
                            this.wait();

                    } catch (Exception exception) {
                        exception.printStackTrace();
                    }
                }
            }
        }
    }
}

class sendData implements Runnable {
    @Override
    public void run(){
        try {
            while(true){
                TimeUnit.SECONDS.sleep(1);
                output.writeObject(new Data(predictedLabel));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

void faceDetection(Mat image) {
    // Build and load the cascade classifier
    CascadeClassifier classifier = new CascadeClassifier();

if(!classifier.load("D:\\Image_Processing\\Auto_Attendence_Track
er\\src\\main\\resources\\haarcascade_frontalface_alt.xml"))
        System.err.println("Unable to load the classifier.");
    // Covert the image to grayscale and eqaulize the frame
```

```
histogram to get better results
    Mat imageGrey = new Mat();
    Imgproc.cvtColor(image, imageGrey, Imgproc.COLOR_BGR2GRAY);
    Imgproc.equalizeHist(imageGrey, imageGrey);
    // Set minimum size of the face to detect - Here 20% of the
frame size
    int absoluteFaceSize = 0;
    int height = imageGrey.rows();
    if(Math.round(height * 0.2f) > 0)
        absoluteFaceSize = Math.round(height * 0.2f);
    // Detection using detectMultiScale function detects objects
of different sizes in the input image.
    // The detected objects are returned as a list of
rectangles.
    MatOfRect faces = new MatOfRect();
    classifier.detectMultiScale(imageGrey,faces,1.1, 2,
Objdetect.CASCADE_SCALE_IMAGE, new Size(absoluteFaceSize,
absoluteFaceSize), new Size());
    // Drawing Rectangles around detected faces
    Rect[] facesArray = faces.toArray();
    int i = 1;
    predictedLabel = 0;
    for(Rect face : facesArray) {
        // System.out.println(face); // face -> coordinates of
face location
        // Send to recognizer
        Mat faceCrop = new Mat(imageGrey, face);
        predictedLabel = model.predict_label(faceCrop);
        if(model.predict_label(faceCrop) == this.regNum)
            predictedLabel = this.regNum;
        i += 1;
        Imgproc.rectangle(image, new Point(face.x, face.y), new
Point(face.x + face.width, face.y + face.height), new
Scalar(255, 0, 0), 4);
        Imgproc.putText(image, String.valueOf(predictedLabel),
new Point(face.x, face.y) , Imgproc.FONT_HERSHEY_COMPLEX, 1, new
Scalar(0, 0, 0), 4);
    }
}
```



The register number of captured student image is predicted and displayed above the detected face.

7. Mark attendance

Once a student joins a class session, a new thread is created to handle the predicted register number results from each student, thereby marking his attendance. To handle students, (from StaffClassSessionController.java)

```
@Override
public void initialize(URL location, ResourceBundle resources) {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/AUTO_AT
TENDANCE", "Torvus", "Torvus@1407");

        stmt = con.prepareStatement("update enrolled set
totalClasses = totalClasses + 1 where CourseId = ? and StaffId =
?");
        stmt.setString(1, this.courseId);
        stmt.setInt(2, this.staffId);
```

```
        stmt.executeUpdate();

        courseIdLabel.setText(courseId);
        staffIdLabel.setText(String.valueOf(staffId));
        classTxt.setText(classTxt.getText() + ">> Class Started
at " + new Date() + "\n");
        this.port = Integer.parseInt((staffId % 100) +
courseId.substring(3));
        this.server = new ServerSocket(this.port);

        this.startTime = new Date();
        ConnectStudents connect = new ConnectStudents();
        connect.start();

    } catch(Exception exception) {
        exception.printStackTrace();
    }
}

class ConnectStudents extends Thread {
    @Override
    public void run() {
        try {
            while(true) {
                Socket student = server.accept();

                ReadData task = new ReadData(student);
                Thread readThread = new Thread(task);
                readThread.start();
            }
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}

class ReadData implements Runnable {
    Socket socket;

    ReadData(Socket socket) {
        this.socket = socket;
    }
    @Override
    public void run() {
```

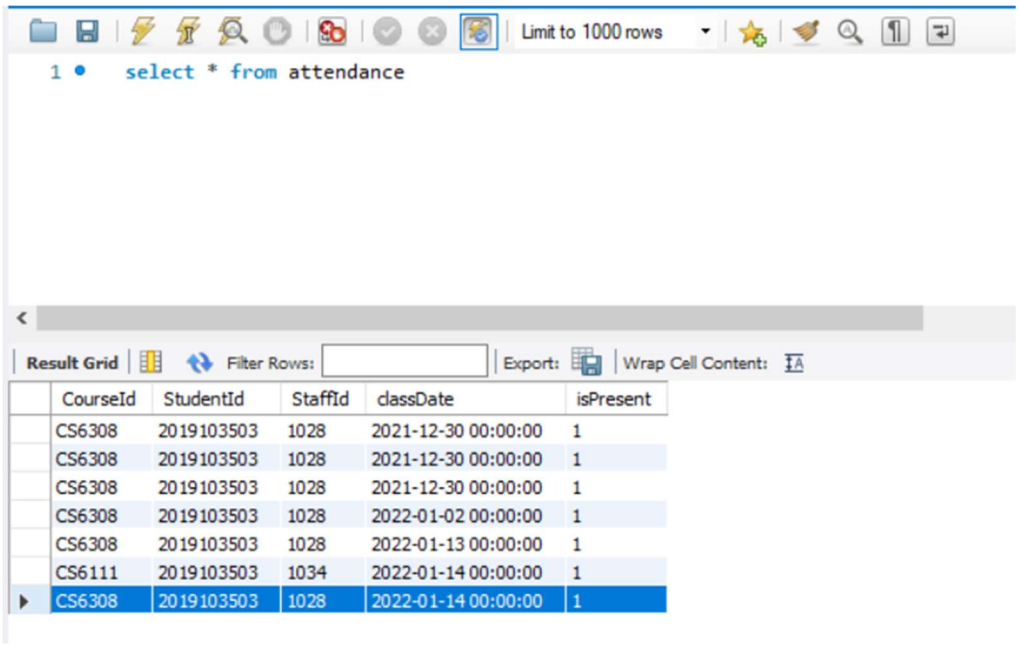
```
        try {
            ObjectInputStream input = new
ObjectInputStream(this.socket.getInputStream());
            Data details = (Data) input.readObject();
            long studentRegNum = details.predictedRegNum;
            long attendance = 0;
            classTxt.setText(classTxt.getText() + ">> New
Student added - " +  studentRegNum + "\n");
            while(true) {
                Data data = (Data) input.readObject();
                if(data.predictedRegNum == -1) {
                    classTxt.setText(classTxt.getText() + ">>
Student left - " + studentRegNum + "\n");
                    ObjectOutputStream output = new
ObjectOutputStream(socket.getOutputStream());
                    output.writeObject(new Data(-1));
                    output.close();
                    markAttendance(studentRegNum, attendance);
                    break;
                }
                if (data.predictedRegNum == studentRegNum) {
                    attendance += 1;
                }
                System.out.println(data.predictedRegNum + "," +
data.time.toString());

            }
            input.close();
            socket.close();
        } catch (Exception e){
            e.printStackTrace();
        }

    }
}

synchronized void markAttendance(long studentReg, long
attendance){
    attendanceList.add(new AttendanceDetails(studentReg,
attendance));
}
```

Once after all the students left the class session, the staff ends the class to record their attendance to database.

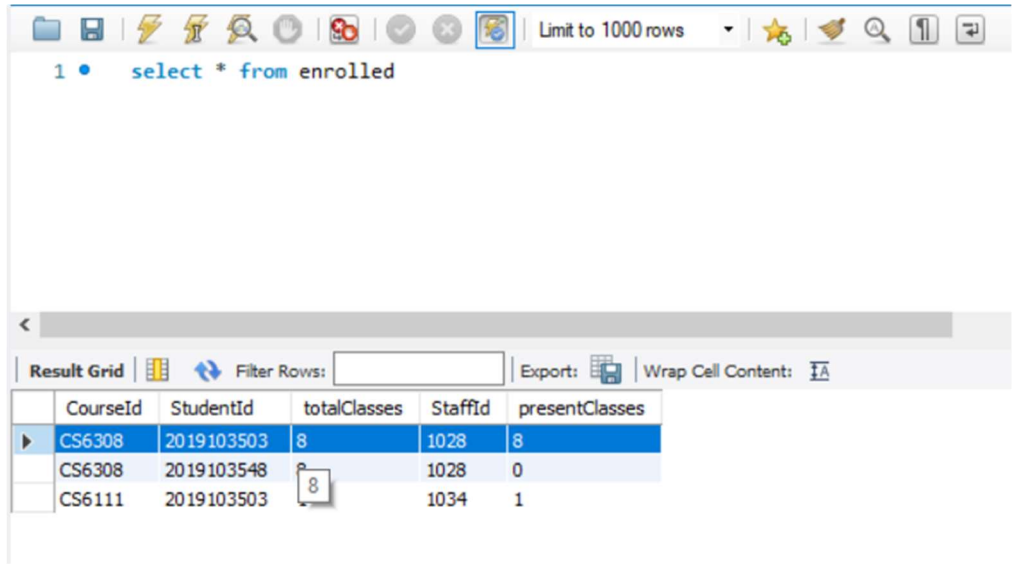
```
public void endClassHandler(ActionEvent e){
    try{
        this.endTime = new Date();
        this.classCompleted = true;
        server.close();
        long duration = (this.endTime.getTime() -
this.startTime.getTime())/1000;
        System.out.println("Class Ended - Duration : " +
duration);
        for(AttendanceDetails a : attendanceList){
            System.out.println(a.regNum + "-" + a.attendance);
            a.attendance = (a.attendance*100)/duration;
            System.out.println(a.regNum + "-" + a.attendance);
            stmt = con.prepareStatement("Insert into attendance
(CourseId, StaffId, StudentId, classDate, isPresent) values (?,
?, ?, ?, ?)");
            stmt.setString(1, this.courseId);
            stmt.setInt(2, this.staffId);
            stmt.setLong(3, a.regNum);
            stmt.setDate(4, new
java.sql.Date(this.startTime.getTime()));
            if(a.attendance >= 50)
                stmt.setBoolean(5, true);
            else
                stmt.setBoolean(5, false);
            stmt.executeUpdate();
        }

        FXMLLoader loader = new
FXMLLoader(getClass().getResource("StaffHome.fxml"));
        root = loader.load();
        StaffController staffController =
loader.getController();
        staffController.getStaffId(this.staffId);
        stage = (Stage) ((Node)
e.getSource()).getScene().getWindow();
        scene = new Scene(root);
        stage.setScene(scene);
        stage.setTitle("Staff Home - Auto Attendance System");
        stage.show();
    } catch(SocketException se) {
```

```
            System.out.println(se.getMessage());
        } catch (Exception ex){
            ex.printStackTrace();
        }

}
```

An entry in the attendance table is created for those students who attended a class session. If the student's presence is recorded more than 60% of the class session, then the student is awarded attendance for the class session and this count of present classes increments by 1 in enrolled table.



CourseId	StudentId	StaffId	classDate	isPresent
CS6308	2019103503	1028	2021-12-30 00:00:00	1
CS6308	2019103503	1028	2021-12-30 00:00:00	1
CS6308	2019103503	1028	2021-12-30 00:00:00	1
CS6308	2019103503	1028	2022-01-02 00:00:00	1
CS6308	2019103503	1028	2022-01-13 00:00:00	1
CS6111	2019103503	1034	2022-01-14 00:00:00	1
CS6308	2019103503	1028	2022-01-14 00:00:00	1



CourseId	StudentId	totalClasses	StaffId	presentClasses
CS6308	2019103503	8	1028	8
CS6308	2019103548	8	1028	0
CS6111	2019103503		1034	1

8. Display student attendance

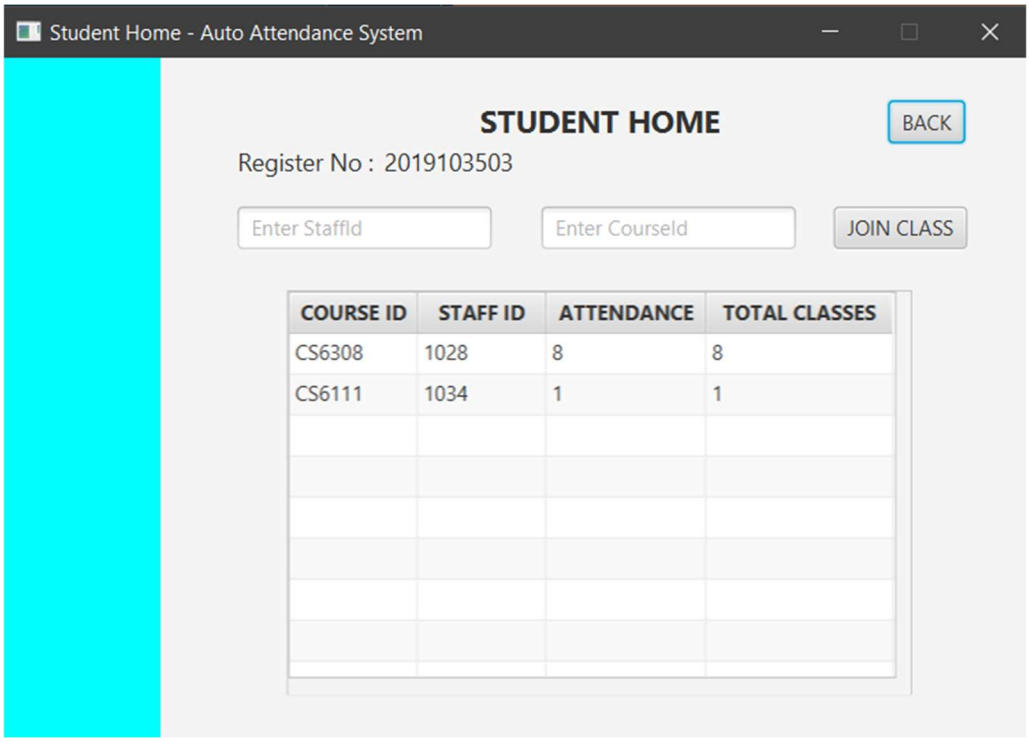
In the student home page, the attendance – the number of classes present and total classes – for each course the student enrolled for is displayed.
To display the attendance details course-wise, (from StudentHomeController.java)

```
@Override
public void initialize(URL url, ResourceBundle resourceBundle) {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        this.con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/AUTO_AT
TENDANCE", "Torvus", "Torvus@1407");
        regNoLabel.setText(this.regNum + "");

        this.stmt = con.prepareStatement("select * from enrolled
where StudentId = ?");
        this.stmt.setLong(1, this.regNum);
        ResultSet rs = this.stmt.executeQuery();
        List<TableEntry> list = new ArrayList<>();
        while(rs.next()){
            list.add(new TableEntry(rs.getString("CourseId"),
rs.getInt("StaffId"), rs.getInt("presentClasses"),
rs.getInt("totalClasses")));
        }
        table.getItems().setAll(list);
    } catch(Exception e) {
        e.printStackTrace();
    }
}
```

The table entry class, (from TableEntry.java)

```
public class TableEntry {
    String  courseId;
    int staffId;
    int presentClasses;
    int totalClasses;
    TableEntry(String CourseId, int StaffId, int presentClasses,
int totalClasses){
        this.courseId = CourseId;
        this.staffId = StaffId;
        this.presentClasses = presentClasses;
        this.totalClasses = totalClasses;
```

```
    }
    public String getCourseId() {
        return this.courseId;
    }
    public int getStaffId() {
        return this.staffId;
    }
    public int getPresentClasses(){
        return this.presentClasses;
    }
    public  int getTotalClasses() {
        return this.totalClasses;
    }
}
```



The course-wise attendance of a student displayed in the student home screen.

CONCLUSION

An automatic attendance management system is a necessary tool for any learning management schools. Most of the existing systems are time consuming, requiring a semi manual work from the teacher or students and prone to malpractices in the online mode. This project aims to solve the issues by integrating face recognition in the process. Even though this system has disadvantages like poor detection in low-lit environment, there is much more room for improvement. Since we implement a modular approach we can improve different modules until we reach an acceptable detection and identification rate. The system can be enhanced in such a way that the accuracy, detection rate and recognition rate can be increased.

REFERENCES

[1] Intro to Java Programming, Comprehensive Version (11th Edition) [Liang, Y. Daniel]

[2] An Approach to Maintain Attendance using Image Processing Techniques - 978-1-5386-3077-8/ - 2017 IEEE

[3] https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b.

[4] https://docs.opencv.org/4.x/da/d60/tutorial_face_main.html

[5] https://docs.opencv.org/4.x/javadoc/index.html