Project on Java Programming: Facial recognition-based attendance systems

1 Introduction

Facial recognition technology has revolutionized attendance systems, replacing traditional methods like roll calls, signatures, or swipe cards with a more efficient approach. This technology uses cameras to capture and analyze facial features to identify individuals accurately. Companies like Google and Amazon are leading examples of its application, with Google implementing facial recognition for security at its Washington campus and Amazon offering these services through AWS for HR management.

We're looking to put together a facial recognition attendance system that links up with a MySQL database. It's a project that'll need Java chops and some serious database management skills to pull off. Think of it as a real-world application that's both cutting-edge and super practical. Here's a breakdown of the project tasks for a team of 2-4 developers:

This project is the most challenge one! You are going to use MySQL and OpenCV.

- **Submission Requirements:** Please compile all files into a ZIP folder named `JAVA_project_VR.zip` and include a detailed report explaining the project.
- **My advice on Role Distribution:**
  - **Database Management:** One developer should focus on integrating and managing the MySQL database. – Database Interaction
  - **User Interface:** One developer should handle the development of user interface panels. –GUI Design
  - **Facial Recognition:** One developer should specialize in implementing facial recognition features using OpenCV. – How to integrate outside library "OpenCV"
- Fell free to check the resources on the website, as long as you can finish this project.

2 Requirements Analysis

The attendance system is designed to perform three core functions: maintaining employee records, facial recognition-based attendance, and viewing attendance logs. We will detail these functionalities as follows:

- **Employee Data Management**:
  - The system utilizes a camera to capture facial data and searches the company's database for a match. If recognized, the system logs the check-in time and displays a successful attendance message.
  - New employees are registered by capturing their facial image with the camera using the "Capture" button. This process generates and stores the facial data file necessary for attendance.
  - Access to use the facial recognition feature for attendance is granted to all employees, but only the system administrator can register new employees or delete existing records.
- **Analysis of Attendance**:
  - The official working hours are set from 10:00 AM to 4:30 PM.
  - **Normal Attendance**: Employees who clock in before 10:00 AM and clock out after 4:30 PM are marked as normal.

Table Name	Illustration	Table Name	Illustration
`t_emp`	Employee Information	`t_lock_in_record`	Attendance Records
`t_user`	Admin Users Table	`t_work_time`	Work Schedule

Table 1: All tables used in the system

- **Late**: Employees clocking in after 10:00 AM are marked as late.
- **Early Departure**: Employees clocking out before 4:30 PM are noted as early departure.
- Attendance issues such as late arrivals, early departures, and absences are monitored and recorded.
- Only the system administrator is authorized to view and generate attendance reports.

3 Dataset Analysis

This section describes the data management strategies and database structures used in our facial recognition attendance system.

1. **Analysis of the Dataset**: As the attendance system accumulates data over time, efficient data management becomes crucial. We employ a MySQL database to facilitate the addition, deletion, updating, and querying of large data volumes. This structured approach ensures the data's integrity and accessibility.
2. **Employee Information**: The heart of our system is the employee data, which includes:
   (a) An `name` to identify the employee.
   (b) A `unique employee ID` that auto-increments with each new registration.
   (c) A `photo` file named `feature_code.png`, where 'feature code' is a UUID generated by the system for unique identification.
3. **Data Separation**: Our database categorizes information into two main types:
   (a) **Employee Information**: This includes names, IDs, and feature codes.
   (b) **Attendance Records**: This tracks the clock-in and clock-out times of each employee.

   Each employee can have multiple attendance records, warranting separate storage for clarity and efficiency.
4. **Permissions**: Access to modify employee information is restricted to maintain data security:
   (a) **Regular users** are only permitted to clock in using their facial data.
   (b) **Administrators** have the ability to add, update, or delete employee records.

The database, named `db_time_attendance`, consists of four primary tables that correlate with the data models outlined:

1. *Employee Information*: Details about each employee. -table2
2. *Attendance Records*: Logs of when employees clock in and out. -table3
3. *Administrator Users*: Information on system administrators. -table4
4. *Work Schedule*: Specifies the standard working hours. -table5

Field Name	Data Type	Field Size	Primary Key	Description
id	INT	DEFAULT	YES	Employee ID
name	VARCHAR	20		Employee Name
code	CHAR	36		UUID

Table 2: `t_emp`: Employee Information Table

Field Name	Data Type	Field Size	Primary Key	Description
emp_id	INT	DEFAULT	YES	Employee ID
lock_in_time	DATETIME	DEFAULT		Clock-in and out Time

Table 3: `t_lock_in_record`: Attendance Records Table

4 System folder organization

Before the system development, we should make a good folder organization structure to divide the various functional modules and achieve unified management. The folder organization structure of the system is shown in figure 1. In particular:

- `com.mr.clock.util` - Utility classes.
  - `DateTimeUtil.java` - Utility class for handling date and time operations.
  - `JDBCUtil.java` - Utility class for managing JDBC operations. Change the UESR and PASSWORD about your MYsql database correspondingly!
- `com.mr.clock.dao` - Responsible for database interactions.
  - `DAO.java` - Interface for standard data access operations. Be careful it is an interface rather than a class.
  - `DAOFactory.java` - Factory pattern implementation to create DAO instances. The factory class is an important part of the interface pattern and is responsible for creating database interface class objects. The advantage of this is that when the front-end business code calls the database interface, the caller can be unaware of which class implements their interface object.
  - `DAOMysqlImpl.java` - Implementation of the DAO interface for MySQL. –Based on mysql's database interface implementation class, the implementation of the DAO interface is detected through the keyword of IMPLEMENTS, the class needs to implement all the abstract methods of the DAO interface.
  - `DAOTest.java` - Unit tests for DAO functionalities.
- `com.mr.clock.session` - Manages user sessions. A global session class that provides a caching mechanism for frequently used data in the system. We use HashSet or HashMap as the data container.
- `com.mr.clock.service` - Services layer handling business logic.

Field Name	Data Type	Field Size	Primary Key	Description
id	INT	DEFAULT	YES	Admin ID
username	VARCHAR	20		Admin Username
password	VARCHAR	20		Admin Password

Table 4: `t_user`: Admin User Table

Field Name	Data Type	Field Size	Primary Key	Description
start	TIME	DEFAULT		Check-in Time
end	TIME	DEFAULT		Check-out Time

Table 5: `t_work_time`: Work Schedule Table

  - `CameraService.java` - Manages camera operations and interfaces. Start the camera, show the camera screen, take a picture
  - `CameraServiceTest.java` - Tests for the CameraService.
  - `FaceEngineService.java` - Manages facial recognition functionalities. Face Feature Extraction, Face Feature Comparison
  - `HRService.java` - Manages employee-related operations.View employee information, delete employees, add new employees, generate daily attendance report, generate monthly attendance report, set work and rest time.
  - `HRServiceTest.java` - Tests for the HRService.
  - `ImageService.java` - Handles image processing tasks. Read all employee photo files and save new employee photo files
  - `OpenCVCameraService.java` - Specialized service for handling camera operations using OpenCV.
  - `TestFaceEngineService.java` - Tests for the FaceEngineService.
- `com.mr.clock.frame` - Contains classes for the graphical user interface.These part is left for you, the idea is to design GUI to interact with users. explore this part and good luck
  - `LoginDialog.java` - Dialog for handling user logins.
  - `MainFrame.java` - The main window of the application.
  - `MainPanel.java` - Central panel used within the MainFrame.
  - `AddEmployeePanel` - Add new employee panel
  - `AttendanceManagementPanel.java`
  - `EmployeeManagementPanel.java`
- `com.mr.clock.pojo` -Data Model Package
  - `Employee.java` - Represents an employee entity.
  - `User.java` - Represents a user entity, typically for system users.
  - `WorkTime.java` - Represents working hours or schedules.
- `com.mr/clock.main`
  - `Main.java`– start point
- `Other Components`
  - `faces` - Contains sample images for testing or operational use.
  - `DatabaseConnection.java` - Manages database connectivity.
  - `TestOpenCV.java` - The main entry point of the application.
  - `UserDAO.java` - Data access object for user operations.I used `DatabaseConnection.java` and `UserDAO.java` to test if I successfully connect to database.

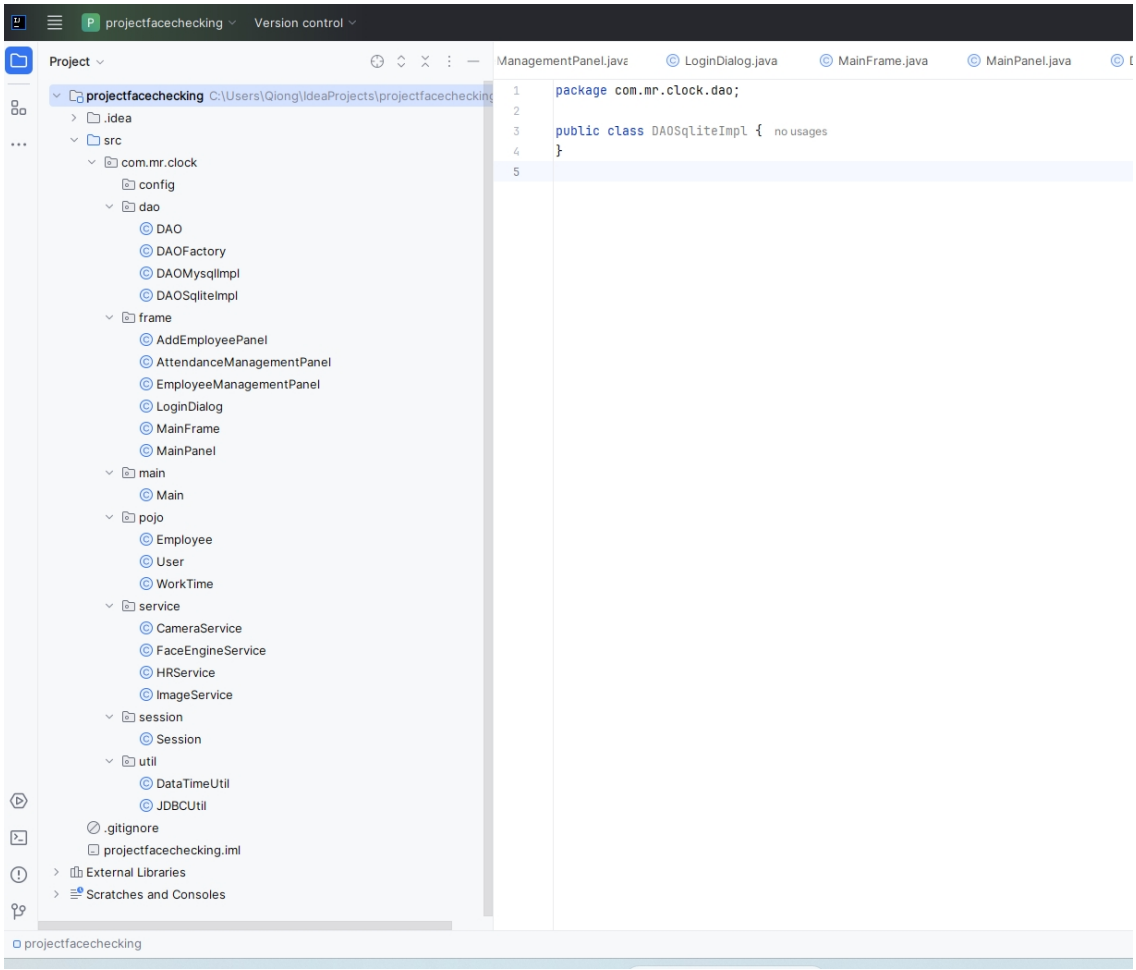

Figure 1: Just an advice!

5 Errors

5.1 How to connect with MySQL?

See my slides of CM5, there is a detailed example.

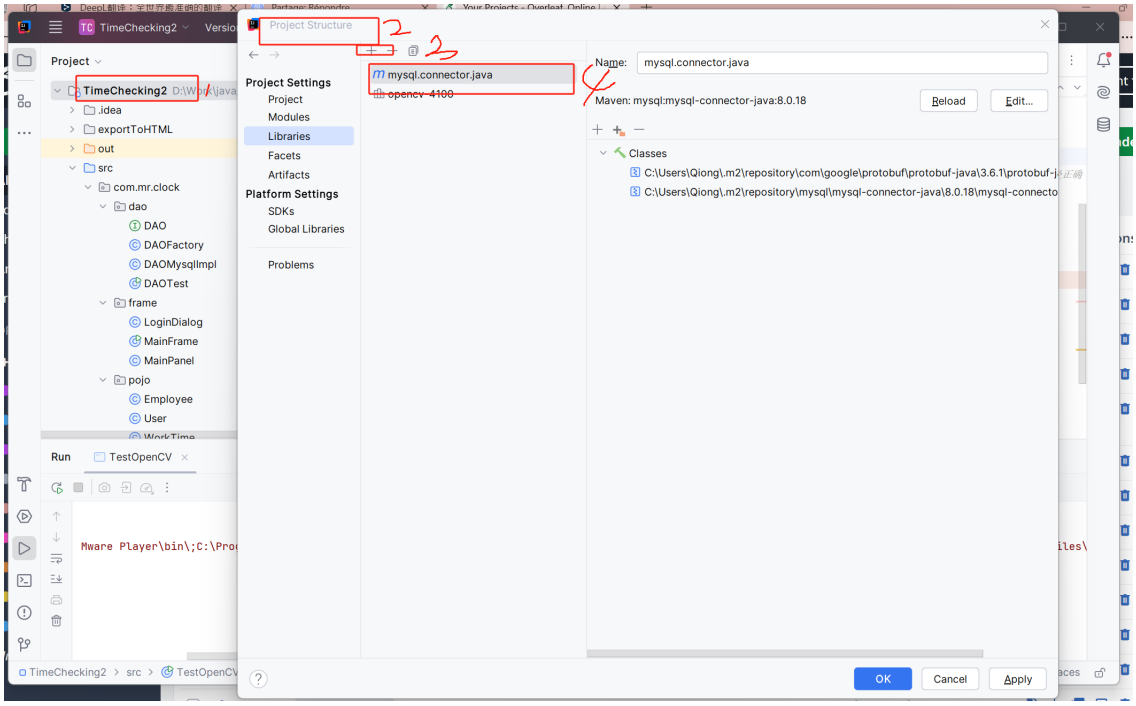

Figure 2: Add MYSQL driver

5.2 Can not connect with openCV?

The OpenCV native library (opencv_java4100.dll) is not correctly loaded. To resolve this, you need to ensure the native library is available in the java.library.path.

2. Add the Library Path to `java.library.path`

Option 1: Add via IntelliJ IDEA VM Options

1. In IntelliJ IDEA, go to **Run > Edit Configurations**.
2. Select your run configuration.
3. In the **VM Options** field, add:

```
javascript

-Djava.library.path=C:/Users/Qiong/opencv/build/java/x64
```

Figure 3: Add the Library Path to `java.library.path`

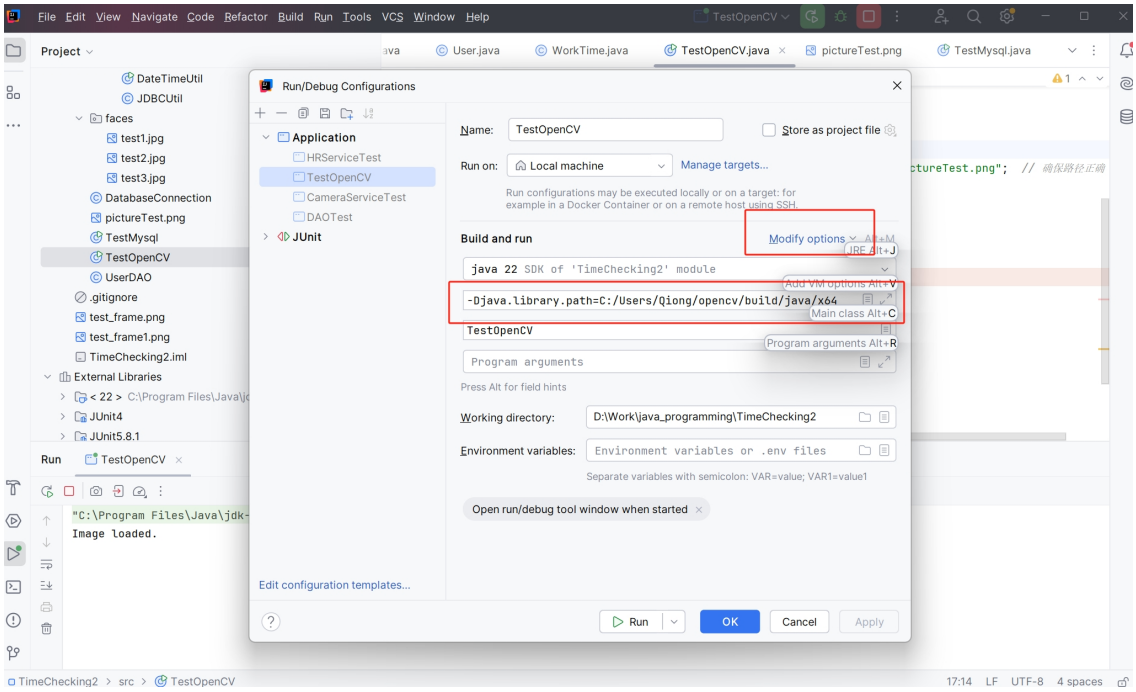

Figure 4: how to add opencv path to your java library