**Student Feedback Management System**

**1. Overview**

Educational institutions often require a platform where students can provide feedback on courses and faculty. Manual collection of feedback is time-consuming and error-prone. A **Student Feedback Management System** can digitize this process, making it easier to collect, manage, and analyze feedback securely.

You are tasked with developing a **web-based application** using **Python (Flask framework)** that enables students to submit feedback, and administrators to view and manage it. The project must use **classes and objects for design**, **file handling for logging activities**, **exception handling for error management**, and **MySQL as the backend database**.

**2. Key Features**

1. **User Module**
   * Students can register and log in.
   * Students can submit feedback for courses/faculty.
   * Each student can only submit feedback once per course.
2. **Admin Module**
   * Admin can log in.
   * Admin can view all submitted feedback.
   * Admin can download feedback logs stored in a file.
3. **Database Integration (MySQL)**
   * Store student details, course details, faculty details, and feedback records.
   * Validate login credentials using database queries.
4. **File Handling**
   * Store application logs (login attempts, feedback submissions, errors) in a text file.
   * Admin can download/view the log file.
5. **Exception Handling**
   * Handle errors like:
     + Database connection failure.
     + Duplicate feedback submission.
     + Invalid login credentials.
     + File read/write errors.
6. **Object-Oriented Programming**
   * Use **classes** like Student, Admin, Course, Feedback, and DatabaseConnection.
   * Implement methods (e.g., submit\_feedback(), view\_feedback(), log\_activity()).

**3. Modules and Classes**

**Classes**

1. DatabaseConnection – Manages MySQL connection (with exception handling).
2. Student – Handles student registration, login, and feedback submission.
3. Admin – Handles admin login and viewing feedback.
4. Feedback – Represents a feedback entry (course, faculty, comments, rating).
5. Logger – Handles file operations for logging activities/errors.

**4. Database Schema (MySQL)**

**Database Name:** feedback\_system

**Tables:**

1. students(student\_id, name, email, password)
2. courses(course\_id, course\_name, faculty\_name)
3. feedback(feedback\_id, student\_id, course\_id, rating, comments)
4. admins(admin\_id, username, password)

**5. Workflow**

1. Student registers → Login → Chooses course → Submits feedback → Stored in DB.
2. Logger writes “Feedback submitted by student\_id at timestamp” into a log file.
3. Admin logs in → Views feedback (retrieved from DB).
4. Admin can also open/download log file for monitoring activities.
5. If database connection fails or student tries duplicate feedback, exceptions are raised and logged.

### 6. ****Example Exception Handling Scenarios****

* **DatabaseConnectionError** – If MySQL server is not reachable.
* **DuplicateFeedbackError** – If a student tries to submit feedback twice for the same course.
* **FileHandlingError** – If logs cannot be written/read.

**Project Scope & Requirements**

## **Class & Object-Oriented Design**

**What to Do:**

1. Write Python class blueprints for:
   * DatabaseConnection (methods: connect, disconnect).
   * Student (methods: register, login, submit\_feedback).
   * Admin (methods: login, view\_feedback).
   * Feedback (attributes: student\_id, course\_id, rating, comments).
   * Logger (methods: write\_log).
2. Keep them as skeletons (methods can be empty for now).

**Deliverable:** Python file with class definitions only (no Flask yet).

## **Database Connectivity with Exception Handling**

**What to Do:**

1. Implement the DatabaseConnection class using mysql.connector or pymysql.
2. Wrap connection code in **try-except blocks**.
3. If connection succeeds → log “Database connected successfully.”
4. If connection fails → log error and raise custom DatabaseConnectionError.

**Deliverable:** Working database connection class + log entry on connect/disconnect.

## **Student Registration & Login Module (Flask + OOP)**

**What to Do:**

1. Create Flask routes /register and /login.
2. In /register:
   * Take input (name, email, password).
   * Insert into students table.
   * Handle duplicate email exception (IntegrityError).
   * Log registration attempt.
3. In /login:
   * Take email & password.
   * Validate against database.
   * If success → session created.
   * If failure → show error, log it.

D**eliverable:** Working student registration and login pages.

## **Feedback Submission Module**

**What to Do:**

1. Create Flask route /submit\_feedback.
2. Fetch courses from the courses table and show them in a dropdown.
3. Let student submit: course, rating (1–5), comments.
4. Insert feedback into feedback table.
5. Before inserting:
   * Check if student already submitted feedback for that course.
   * If yes → raise DuplicateFeedbackError.
6. Log submission success/failure.

**Deliverable:** Student can submit feedback once per course.

## **Admin Module (Login & View Feedback)**

**What to Do:**

1. Create /admin/login route:
   * Validate username & password from admins table.
   * If success → redirect to /admin/view\_feedback.
2. Create /admin/view\_feedback route:
   * Fetch all feedback from DB.
   * Display in a table (student, course, rating, comments).
3. Add option for admin to **download the log file** (app.log).

**Deliverable:** Admin can login and see all feedback + logs.

## **Exception Handling Scenarios**

**What to Do:**

1. Test and handle these errors:
   * Database down → raise DatabaseConnectionError.
   * Duplicate student registration → catch SQL error, log it.
   * Duplicate feedback → raise custom DuplicateFeedbackError.
   * Invalid login → raise AuthenticationError.
   * File handling error (log file missing/locked).
2. Ensure all exceptions are **logged** in app.log.

**Deliverable:** Robust exception handling across system.

## **Complete Workflow Testing**

**What to Do:**

1. Register at least 2 students.
2. Register an admin.
3. Login as student → submit feedback for a course.
4. Try submitting duplicate feedback → check error handling.
5. Login as admin → view all feedback.
6. Admin downloads app.log → verify logs contain all actions.

**Deliverable:** Fully working mini project that satisfies all requirements.