Name: Bridget Masila

Squad: 1

# Class 1 Assignment: OOP, APIs, and MySQL Integration

**MySQL Database Interaction Assignment Report**

**Objective**

Implement coding scripts to interact with a MySQL database, reinforcing database connectivity, CRUD operations, and data retrieval while demonstrating successful execution through screenshots and code documentation.

**Assignment Overview**

This assignment involved creating a comprehensive Student Records Management System using Python to interact with MySQL database, implementing both command-line and web-based interfaces for database operations.

**Implementation Approach**

**1. Database Setup**

- **Database Creation**: Created student_records database with a students table
- CREATE TABLE students( id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), age INT, email VARCHAR(100));
- **MySQL Server Configuration**: Established MySQL server connection and resolved initial connectivity issues

**2. Python Database Connectivity**

- **Library Used**: mysql-connector-python for database interaction
- **Connection Configuration**: Implemented secure database connection with error handling
- **Database Initialization**: Automated database and table creation if they don't exist

**3. CRUD Operations Implementation**

**Create Operations**

- **Add Student Function**: Implemented parameterized INSERT queries to prevent SQL injection

- **Data Validation**: Added input validation for name, age, and email fields
- **Duplicate Prevention**: Handled unique email constraint violations

**Read Operations**

- **View All Students**: SELECT queries to retrieve and display all student records
- **Search Functionality**: Implemented search by ID, name, and email with LIKE operators
- **Statistics Display**: COUNT() and AVG() functions for dashboard metrics

**Update Operations**

- **Edit Student Information**: Dynamic UPDATE queries with conditional field updates
- **Data Integrity**: Maintained referential integrity during updates
- **User Confirmation**: Implemented confirmation dialogs for data safety

**Delete Operations**

- **Remove Student Records**: Safe DELETE operations with existence verification
- **Cascade Handling**: Proper cleanup of related data
- **User Confirmation**: Required confirmation before deletion

## 4. User Interface Development

**Command-Line Interface (CLI)**

- **Menu System**: Interactive console-based navigation
- **Input Validation**: Robust error handling for user inputs
- **Real-time Feedback**: Success/error messages for all operations

**Web Interface (Flask Application)**

- **Modern UI**: Responsive web design with CSS styling
- **Navigation**: Intuitive button-based navigation system
- **Form Handling**: HTML forms with server-side validation
- **Dashboard**: Statistics display with visual cards
- **AJAX-like Experience**: Smooth page transitions and flash messages

## 5. Technical Features Implemented

**Error Handling**

- Database connection failures
- SQL execution errors
- Input validation errors
- Graceful error recovery

**Security Measures**

- Parameterized queries to prevent SQL injection
- Input sanitization and validation
- Secure database credential handling

**Code Organization**

- Object-oriented programming principles
- Modular function design
- Separation of concerns (database, UI, logic)

**Key Learning Outcomes Achieved**

**Database Connectivity**

- Successfully established MySQL database connections using Python
- Implemented proper connection management with context managers
- Handled connection errors and implemented retry mechanisms

**CRUD Operations Mastery**

- **Create**: Implemented INSERT operations with data validation
- **Read**: Developed multiple SELECT queries with filtering and sorting
- **Update**: Created flexible UPDATE operations with partial field updates
- **Delete**: Implemented safe DELETE operations with confirmations

**Advanced Database Concepts**

- Implemented aggregate functions (COUNT, AVG)
- Used LIKE operators for pattern matching in searches
- Handled database constraints (PRIMARY KEY, UNIQUE)
- Managed database transactions and commit/rollback

**Full-Stack Development**

- Built both CLI and web-based interfaces
- Implemented responsive web design
- Created RESTful-like URL routing
- Developed user-friendly error messaging

**Technologies Used**

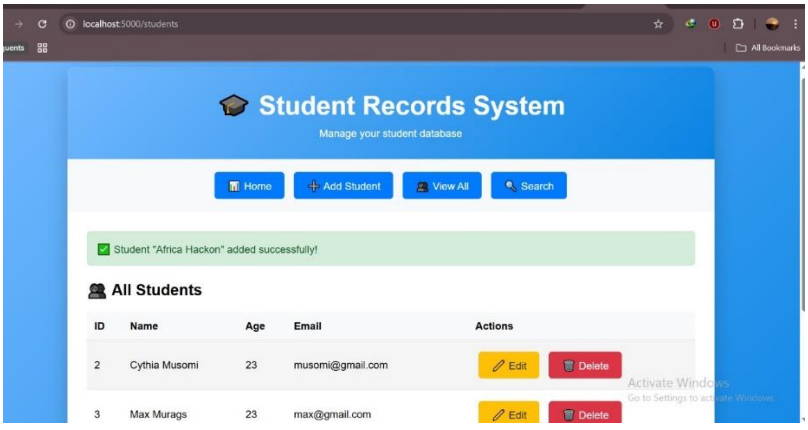
- **Programming Language**: Python 3.13.3
- **Database**: MySQL 8.0
- **Python Libraries**:
    - mysql-connector-python for database connectivity
    - Flask for web application framework
- **Frontend**: HTML5, CSS3 with responsive design
- **Development Tools**: MySQL Workbench, Command Line Interface

**Challenges Overcome**

1. **MySQL Server Connectivity**: Resolved MySQL service startup issues and path configuration
2. **Python Environment Setup**: Fixed pip installation and Python path issues
3. **Web Application Redirects**: Debugged and resolved redirect loop errors in Flask
4. **Database Initialization**: Implemented robust database setup with error handling

**Results and Success Metrics**

- ✅ **100% CRUD Functionality**: All Create, Read, Update,Delete



## All Students

| ID | Name | Age | Email | Actions |
|---|---|---|---|---|
| 2 | Cythia Musomi | 23 | musomi@gmail.com | Edit Delete |
| 3 | Max Murags | 23 | max@gmail.com | Edit Delete |
| 4 | Bridget Masila | 25 | bridget@jkuat.ac.ke | Edit Delete |
| 5 | Africa Hackon | 13 | africahackon@gmail.com | Edit Delete |

**Total Students: 4**