Project Report: UniTrack - University Attendance Management System

Group Number: 06
Course: NANO2142 - Introduction to Software Development

1. Introduction & Background

1.1 Background
University attendance tracking is a critical administrative task that ensures student engagement and compliance with academic regulations. Traditionally, this process involves manual paper sheets, which are prone to loss, errors, and inefficient data entry. Managing medical absences adds another layer of complexity, often requiring physical document submission and manual verification.
1.2 Overview
UniTrack is a web-based University Attendance Management System designed to modernize this workflow. It serves as a centralized platform where administrators, lecturers, and students can interact to manage attendance records, track participation statistics, and process medical reports efficiently.

2. Problem Definition & Objectives

2.1 Problem Definition
The current manual system suffers from several issues:

- **Inefficiency:** Lecturers spend valuable class time calling names or circulating paper sheets.
- **Data Latency:** Attendance data is not immediately available for analysis or warning letters.
- **Lack of Transparency:** Students cannot easily track their own attendance percentage in real-time.
- **Medical Verification:** The process of submitting and approving medical certificates is manual and disjointed.

2.2 Objectives
The primary objectives of UniTrack are to:

1. **Automate Attendance Tracking:** Replace paper sheets with digital individual and bulk marking tools.
2. **Streamline Medical Absences:** Provide a digital interface for students to upload medical documents and for admins to approve/reject them.
3. **Enhance Accountability:** Give students real-time access to their attendance records

and calculate monthly percentages automatically.
4. **Data Management:** Enable bulk data operations via CSV uploads and exports.

3. System Requirements

3.1 Functional Requirements

- **Authentication:** Secure login for Students, Admins (Lecturers), and Administrators using hashed passwords.
- **Attendance Marking:**
  - Admins must be able to mark attendance individually or in bulk for a specific course and date.
  - Support for CSV bulk upload for mass data entry.
- **Reporting:**
  - System must calculate monthly attendance percentages.
  - Visual indicators (Green/Yellow/Red) based on attendance performance.
- **Medical System:**
  - Students must be able to upload medical proofs (PDF/Images).
  - Admins must be able to view, approve, or reject these requests.
  - Approved medical reports must automatically update the attendance status to "Medical".

3.2 Non-Functional Requirements

- **Security:** Passwords must be hashed (using Werkzeug). Access to routes must be restricted by role.
- **Usability:** The interface must be responsive (Bootstrap 5) and accessible on mobile devices.
- **Reliability:** The database must ensure data integrity using foreign keys and unique constraints.

4. System Design

4.1 High-Level Architecture

The system follows the **Model-View-Controller (MVC)** architectural pattern, implemented via the Flask framework:

- **Model (SQLAlchemy):** Represents the data structure (User, Attendance, MedicalReport).
- **View (Jinja2 Templates):** Handles the UI rendering (HTML/Bootstrap).
- **Controller (Flask Routes):** Manages the business logic, input processing, and interaction between Model and View.

4.2 Database Design

The system uses a Relational Database (SQLite) with the following schema:

- **User Table:** Stores credentials and roles.
  - *Attributes:* id (PK), index_number (Unique), password (Hashed), role, name.
- **Attendance Table:** Tracks daily records.
  - *Attributes:* id (PK), student_index, course_code, date, status (Present/Absent/Medical).
- **StudentCourse Table:** Maps students to enrolled courses.
  - *Attributes:* id (PK), student_index, course_code.
- **MedicalReport Table:** Manages excuse documentation.
  - *Attributes:* id (PK), attendance_id (FK), document_path, approved (Boolean).

(Note: In your final submission, attach the ER Diagram here as required by the checklist).

4.3 OOP Design & Class Structure

Although implemented in Python, the system utilizes Object-Oriented principles through the **Object-Relational Mapping (ORM)** pattern provided by SQLAlchemy.

- Encapsulation:
  The User class encapsulates all user-related data. Direct database queries are abstracted into class methods and property access, protecting the raw SQL layer from the rest of the application.
- Inheritance:
  All model classes (User, Attendance, etc.) inherit from db.Model. This grants them powerful methods for query construction, saving, and deletion without rewriting SQL code.
- Polymorphism:
  The Flask route handlers treat different user roles (student, admin) polymorphically in the login session, though specific access control is enforced via conditional logic in the controllers.

5. Implementation Details

5.1 Technology Stack

- **Backend:** Python 3.x, Flask 3.0.0
- **Database:** SQLite, SQLAlchemy ORM
- **Frontend:** HTML5, Bootstrap 5, Jinja2 Templating
- **Security:** Werkzeug Security (Password Hashing)

5.2 Key Algorithms

- Monthly Attendance Calculation:
  The system iterates through all attendance records for a logged-in student, grouping them by (Month, Year). It dynamically calculates the percentage:

$$\text{Percentage} = \left(\frac{\text{Present Count}}{\text{Total Sessions}} \right) \times 100$$

This logic is handled in the student_dashboard route in app.py.

- Bulk CSV Processing:
  The bulk_upload route utilizes Python's csv library to parse uploaded files. It performs row-by-row validation (checking if the student exists and if the date format is correct) before committing the transaction to the database to ensure atomicity.

6. Testing Strategy & Results

6.1 Testing Strategy

We employed manual functional testing to verify system operations.

- **Unit Testing:** Verified individual functions like allowed_file for uploads.
- **Integration Testing:** Verified that approving a medical report correctly updates the Attendance table status from "Absent" to "Medical".

6.2 Test Cases (Sample)

ID	Test Description	Input Data	Expected Result	Pass/Fail
TC01	Student Login	User: S1001, Pass: 1234	Redirect to Student Dashboard	Pass
TC02	Invalid Login	User: S1001, Pass: Wrong	Error: "Invalid Credentials"	Pass
TC03	Medical Upload	File: report.pdf	File saved to uploads/	Pass
TC04	Bulk Mark	Course: NANO2112	All students marked Present	Pass
TC05	Admin Approval	Click "Approve" on Report	Status updates to "Medical"	Pass

7. Limitations & Future Improvements

7.1 Limitations

- Currently limited to local file storage for medical reports (not cloud-based).
- No email notification system for alerts.
- Biometric authentication is not currently supported.

7.2 Future Improvements

- Implement email notifications using SMTP.
- Integrate a mobile application (Flutter/React Native).
- Migrate database to PostgreSQL for higher scalability.

8. Conclusion

The **UniTrack** system successfully digitizes the attendance management process. By integrating attendance tracking with medical excuse management, it closes the loop on student participation monitoring. The system meets the core requirements of functionality, usability, and database integration, providing a robust foundation for future academic management tools.

Part 3: User Manual (Short Guide)

1. Getting Started

- **Access:** Open a web browser and navigate to http://localhost:5000.
- **Login:** Enter your Index Number and Password.
    - *Default Admin:* Dulsara / Dulsara
    - *Default Student:* S1001 / 1234

2. Student Features

- **View Attendance:** Upon login, your dashboard displays a history of all attendance records.
- **Check Statistics:** The top of the dashboard shows a progress bar for your monthly attendance percentage.
- **Submit Medical:**
    1. Click "Upload Medical Report".
    2. Select the specific date you were absent from the dropdown.
    3. Upload your medical certificate (PDF/Image) and click Submit.

3. Administrator/Lecturer Features

- **Mark Attendance:**
  1. Go to "Mark Attendance by Subject".
  2. Select the Course and Date.
  3. A list of students will appear; toggle "Present" or "Absent".
  4. Click "Mark Attendance for All".
- **Approve Medicals:**
  1. On the Admin Dashboard, locate "Pending Medical Reports".
  2. Click "View Document" to verify the proof.
  3. Click "Approve" to accept (this updates the student's status automatically) or "Reject".
- **Bulk Upload:**
  1. Select "Bulk Upload Attendance".
  2. Upload a CSV file with columns: student_index, date, status.