Final Project Report: Z-UPA Academic Hub

Submitted to: School of Computational Sciences and Artificial Intelligence (CSAI)
Course: Python Programming **Student Name:** Omar Issam Mohamed Mohamed Abdel Halim **Student ID:** 2025081341 **Date:** January 3, 2026

---

1. Executive Summary

The Z-UPA Academic Hub is a comprehensive desktop application engineered to serve as a centralized digital ecosystem for university students. Developed using Python, the system addresses the fragmentation of academic resources by integrating course management, grade tracking, task scheduling, and administrative services into a single, cohesive interface. Leveraging the **CustomTkinter** library for a modern Graphical User Interface (GUI) and **Matplotlib** for data analytics, the project delivers a user-centric solution that enhances organizational efficiency and academic oversight.

2. Introduction

2.1 Problem Statement

University students often struggle with managing academic responsibilities scattered across various platforms—learning management systems for content, separate portals for grades, and external tools for task management. This fragmentation leads to administrative friction and reduced focus on actual learning.

2.2 Project Objective

The primary objective of this project is to architect a "**Single Point of Contact**" platform. The system allows students to:

- **Monitor Performance:** View real-time GPA calculations and attendance statistics.
- **Manage Workload:** Track assignments and deadlines through an integrated task manager.
- **Access Content:** Retrieve lecture videos and materials directly within the application.

3. System Architecture & Methodology

3.1 Development Methodology

The project follows a **GUI-Driven Application** architecture, utilizing **Object-Oriented Programming (OOP)** principles to ensure modularity. The codebase is structured around the `ZUPA_App` class, which manages the application state and view transitions, ensuring a clean separation between the user interface and backend logic.

3.2 Technology Stack

The application is built upon a robust stack of Python libraries selected for efficiency and user experience:

Component	Technology	Description
Core Logic	Python 3.x	The backbone of the application's business logic and algorithms.
Interface	CustomTkinter	Used to create a modern, high-DPI aware, dark-mode capable GUI.
Analytics	Matplotlib	Integrated to generate dynamic charts for task analysis and grade trends.
Data Persistence	JSON & File I/O	A lightweight, local file-based database system for storing student records and configurations.
Utilities	Datetime / OS	Handles scheduling logic, file directory navigation, and external link execution.

3.3 Data Structure

To ensure portability and speed without the overhead of a server-based SQL engine, the system utilizes a structured file system database:

- **Authentication:** `students.txt` stores encrypted user credentials.
- **Academic Records:** `courses.txt`, `grades.txt`, and `att.txt` maintain the relational data between students and their academic history.
- **Configuration:** `config.json` persists user preferences such as theme settings.

4. Implementation & Core Features

4.1 Authentication & Profile Management

- **Secure Login:** The system implements a validation function `validate_login` to verify student IDs and passwords against the database.
- **Profile Control:** Students can view their major and year details and securely update their passwords via the `change_password` function.
- **Data Export:** A report generation feature allows students to export their academic history to a text file for external use.

4.2 The Interactive Dashboard

Upon logging in, the user is presented with a high-level overview of their academic standing:

- **Statistical Cards:** Immediate display of the current GPA, Pending Tasks, and Enrolled Courses.
- **Visual Analytics:** A dynamic Pie Chart (powered by Matplotlib) visualizes the ratio of "**Pending**" vs. "**Completed**" tasks to encourage productivity.

4.3 Course & Content Management

- **Dynamic Filtering:** The `filter_courses` algorithm automatically displays relevant courses based on the student's year and major.

- **Resource Access:** Each course page acts as a hub, providing direct buttons to open PDF lectures or launch video URLs in the web browser.
- **Assessment Module:** An integrated quiz system loads questions from file, records student answers, calculates scores instantly using `calc_score`, and saves the results.

4.4 Academic Performance Tracking

- **GPA Calculation:** The system features a `calc_gpa` engine that aggregates all recorded grades to compute a cumulative GPA dynamically.
- **Attendance Monitoring:** The `calc_attendance_rate` function analyzes attendance logs. It visually flags courses with low attendance (below 75%) in red to alert the student.
- **Grade Visualization:** A bar chart visualizes performance across different subjects, helping students identify strengths and weaknesses.

4.5 Productivity & Services

- **Task Manager:** A fully functional To-Do list allows students to mark tasks as completed, which updates the database and the dashboard statistics in real-time.
- **Administrative Services:** Includes a complaint submission form (`save_complaint`) and a digital notice board for important university announcements.

5. Conclusion

The Z-UPA Academic Hub successfully fulfills its mission to digitize and streamline the student experience at the School of Computational Sciences and Artificial Intelligence. By effectively combining data persistence, algorithm-driven analytics, and a professional graphical interface, the project stands as a complete, functional product. It demonstrates a mastery of Python programming, specifically in the areas of data manipulation, OOP architecture, and GUI development, providing a robust solution for academic management.