Academic Management System

Pranesh Vijay B 22CS10013

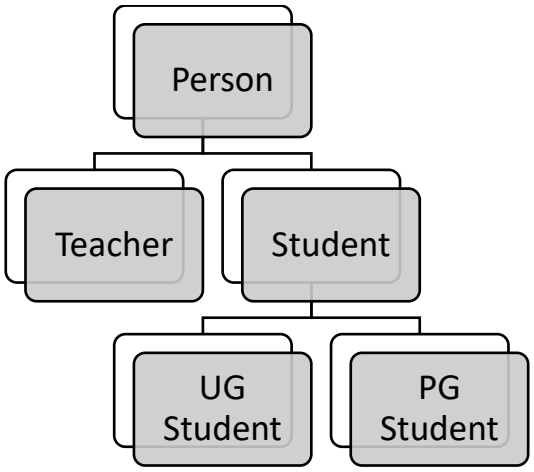
1. Objective

The primary objective of this project is to develop an efficient and user-friendly Academic Management System. This software application aims to streamline various processes within an academic unit, focusing on user registration, authentication, profile management, and deregistration.

2. Class Hierarchy:

The class hierarchy comprises the following main classes:

- Person
- Teacher (inherits from Person)
- Student (inherits from Person)
    - UG Student (undergraduate student, inherits from Student)
    - PG Student (postgraduate student, inherits from Student)



3. Scope

a. User Registration

- The system allows new users to register by providing essential details such as a valid email address (used as the User ID) and a secure password.
- Password validation ensures that registered users create strong and secure passwords.
- User registration is the initial step for gaining access to the system

b. Authentication and Sign-in

- Registered users can sign in to the system using their unique User ID and password.
- The system authenticates users' credentials to ensure legitimate access.
- A limited number of attempts are provided for authentication, with the account being deactivated after three unsuccessful attempts for security purposes.

c. Profile Management:

- Authenticated users have the ability to manage their profiles.
- Users can view and update their personal information and change their password.

d. Deregistration

- Users have the option to submit a deregistration request if they wish to deactivate their accounts.
- Upon successful submission, the user's account is deleted from the system

3.Implementation

The Academic Management System (AMS) is implemented in Python using the Tkinter library for the graphical user interface (GUI) and SQLite for database management. The system consists of several classes, each responsible for specific functionalities. Below are the key implementation details for the main components:

Classes

- ***person***: A base class representing a generic person with attributes like userid, password, name, date of birth, and type.
- ***student***: A subclass of ***person*** representing a student with additional attributes like roll number and grade.
- ***teacher***: A subclass of ***person*** representing a teacher with an additional attribute for employee ID.
- ***ugstudent***: A subclass of ***student*** representing an undergraduate student with an additional attribute for the hall of residence.
- ***pgstudent***: A subclass of ***student*** representing a postgraduate student.

Database

- SQLite is used for database operations.
- Tables are created *for users, ugstudents, pgstudents,* and *teachers.*
- Each user type (UG Student, PG Student, Teacher) has its respective table to store additional information.

GUI

- ***class registerframe ():*** This class is responsible for creating a registration frame where users can register themselves with the system. The registration process involves entering a user ID, password, and selecting a user type (UG Student, PG Student, or Teacher).

  __init__ method:
  - Initializes the Tkinter window with specific attributes (title, size, etc.).
  - Creates a frame within the master window to organize UI elements.
  - Sets up labels, entry fields, dropdowns, and buttons for the registration form.

  _register method:

  - Validates user input for completeness and correctness.
  - Checks if the user ID is a valid email address.
  - Ensures that passwords match and meet specified strength criteria.
  - Checks if the user ID already exists in the database.
  - Inserts user data into the corresponding database tables based on the selected user type.
  - Displays appropriate error or success messages.
  - Destroys the current registration frame and navigates back to the login frame upon successful registration.

_back method:

- Destroys the current registration frame and navigates back to the main frame.

_close method:

- Closes the current registration frame.

- ***class loginframe ():*** This class serves as the login interface for the Academic Management System, offering a Tkinter window with organized UI elements. It provides user-friendly login and account security features, including credential validation, login attempt tracking, and account deactivation for multiple incorrect attempts.

  __init__ method:

  - Initializes the loginframe object with the master window and configures the window attributes such as title, size, and background.
  - Creates a frame within the master window (_frame) and sets up labels, entry fields, and buttons for the login form.
  - Defines the layout and appearance of UI elements using Tkinter widgets.

  _login method:
  - Validates user input for user ID and password.
  - Checks if the user ID exists in the database and verifies the password.
  - Handles the logic for tracking login attempts, deactivating accounts, and resetting attempts upon successful login.
  - Transitions to the dashboard frame (dashboardframe) upon successful login.

  _back method:

  - Destroys the current window and navigates back to the main page by creating a new instance of mainframe.

  _close method:

  - Closes the current window.

- ***class dashboardframe ():*** The dashboardframe class represents the dashboard interface of the Academic Management System, allowing users to access features such as editing their profile, deactivating their account, changing their password, and logging out. The class initializes a Tkinter window with specific attributes and sets up buttons for each functionality.

  _editprofile method:
  - Initializes the editprofileframe with the user's ID and destroys the current dashboard window.
  - The editprofileframe allows users to modify their profile information, including fields specific to their user type (UG student, PG student, or teacher).

  _deactivateaccount method:
  - Initializes the deactivatecccountframe with the user's ID and destroys the current dashboard window.
  - The deactivatecccountframe provides the option for users to deactivate their account.

  _logout method:
  - Logs out the user by destroying the current dashboard window and initializing a new login frame.

  _changepassword method:
  - Initializes the changepasswordframe with the user's ID and destroys the current dashboard window.
  - The changepasswordframe allows users to change their account password.

  _close method:
  - Closes the current dashboard window.

- ***editprofileframe ()*** class: This class allows users to edit their profile information, including fields like name, date of birth, roll number, graduation year, hall (for UG students), and employee ID (for teachers). Users can save their changes or go back to the dashboard.

  _save method:
  - Validates and saves the changes made by the user, updating the user's information in the database.
  - For UG students, PG students, and teachers, the specific fields related to their user type are updated.
  - After saving, a success message is displayed, and the user is redirected back to the dashboard with the same user ID.

  _back method:
  - Navigates back to the dashboard without saving changes, maintaining the same user ID.

  _close method:
  - Closes the current edit profile window.

- ***deactivatecccountframe ()*** class: represents the interface for users who wish to deactivate their accounts. It initializes a Tkinter window, asks for confirmation, and provides options to proceed or cancel.

  _yes method:
  - Deletes user information from the database, including type-specific details (UG student, PG student, or teacher).
  - Displays success message, destroys the current window, and redirects to the login page.

  _no method:
  - Destroys the current window and returns to the dashboard without deactivating the account.

  _close method:
  - Closes the current window.

- ***changepasswordframe ()*** class: represents the interface for users to change their account passwords. It provides input fields for the old password, new password, and confirmation of the new password.

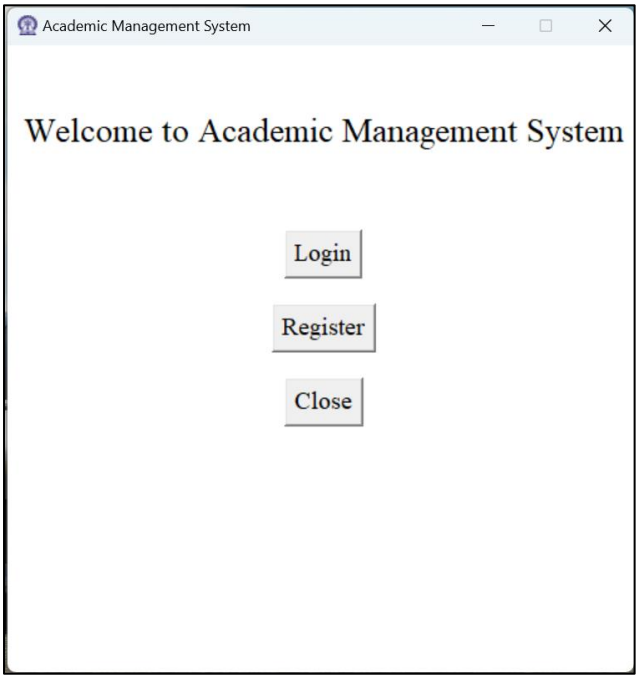
  _changepassword method:
  - Validates the old password and new password fields.
  - Checks if the old password is correct and if the new password meets specified criteria (length, character types).
  - Updates the password in the database, displays success message, and returns to the dashboard.

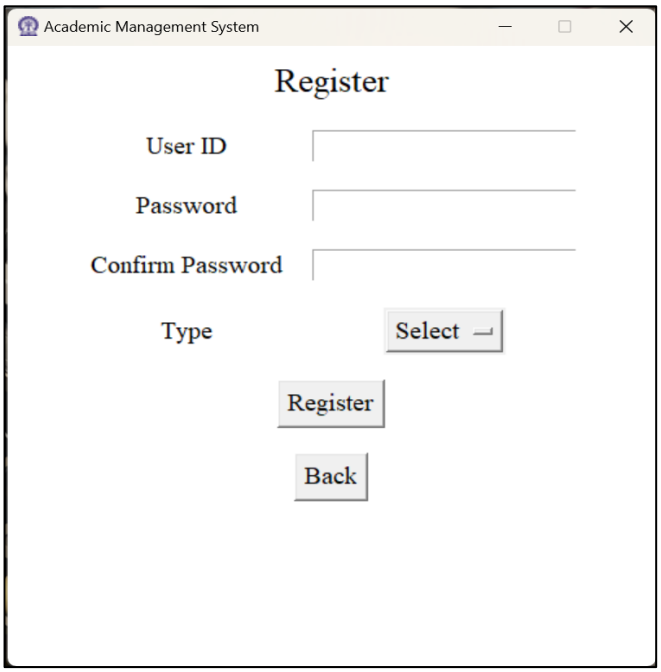
  _back method:
  - Destroys the current window and returns to the dashboard without changing the password.

  _close method:
  - Closes the current window.

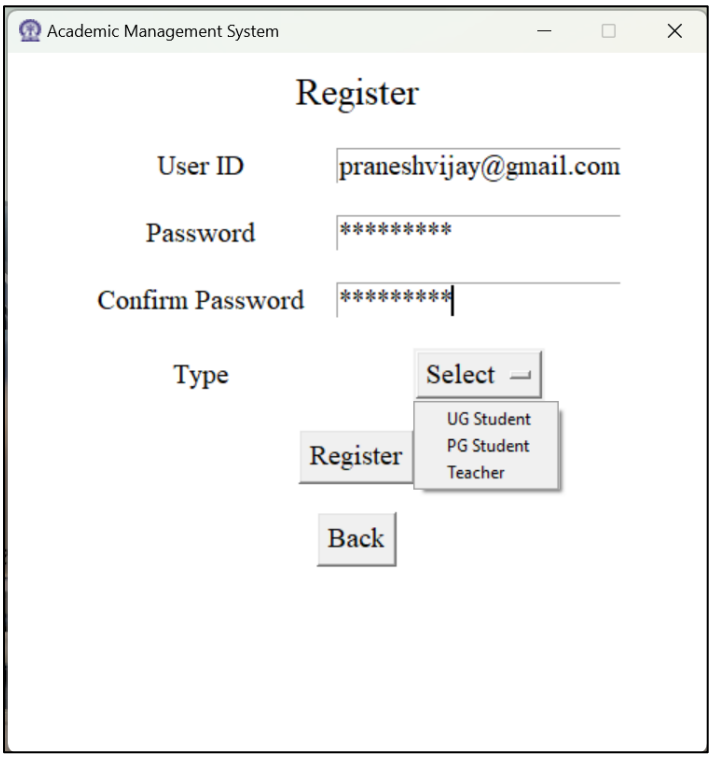
These classes collectively serve as the user interface components for account deactivation, password change, and the main entry point to the Academic Management System. The functionalities are designed to provide a straightforward and intuitive user experience while ensuring proper handling of user data and interactions.

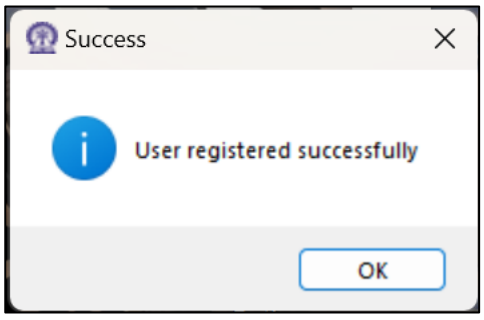
4. Screenshots of the system:



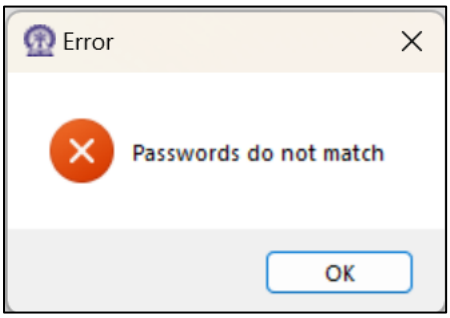


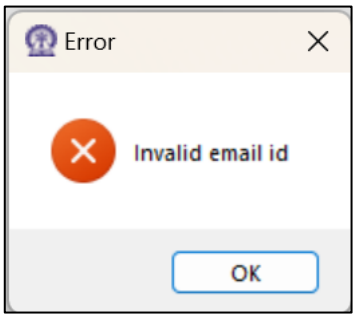
The system starts with the mainframe which has three options: Login, Register and Close. Since this is the first time, we are opening the system, there are no users registered yet. On clicking the register button, the register frame shows up. It asks for User ID which is a valid email address and a strong password (is within 8-12 characters, has at least one lowercase letter, one uppercase letter, one digit, and one special character). The password needs to typed again for security reasons. Then there is a dropdown for selecting type. If everything is in correct format, the user id registered successfully.

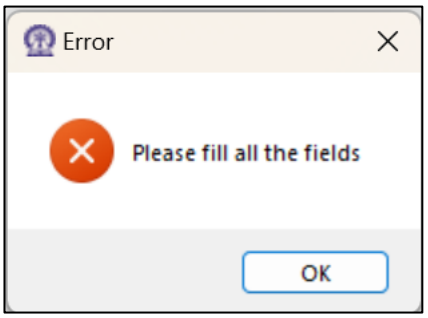


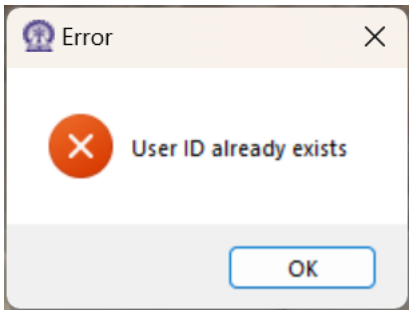


Possible error messages during registration:

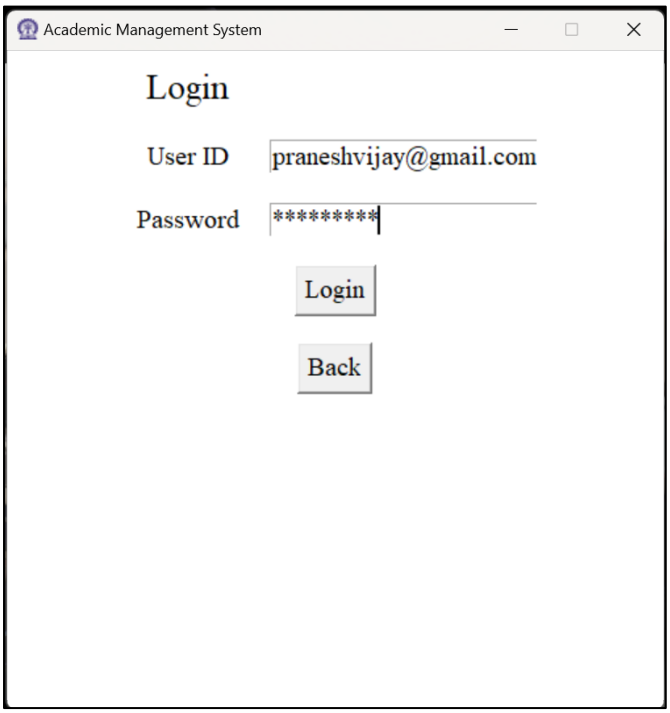


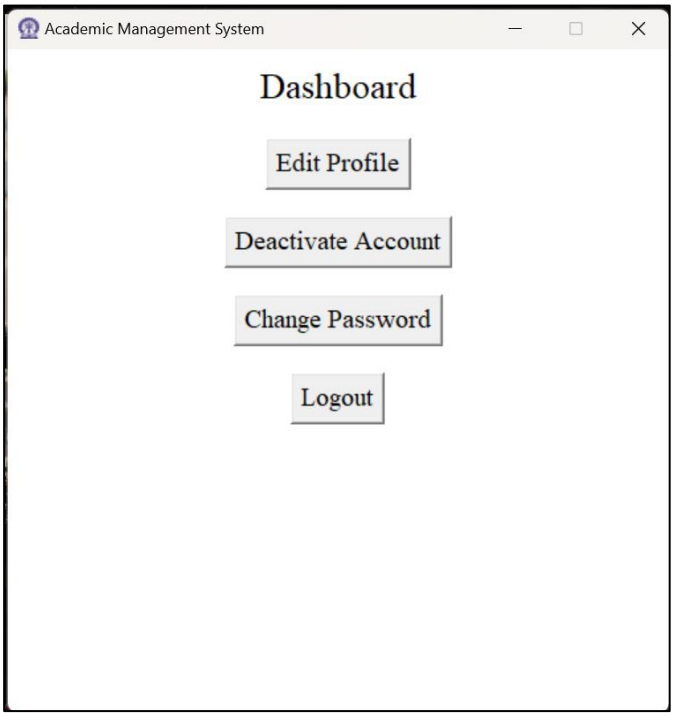






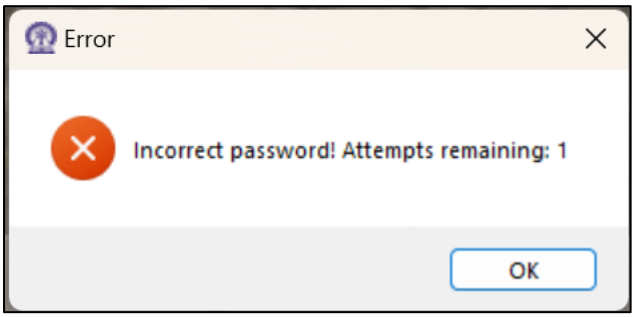
For demonstration, we register a UG Student and a teacher. After successful registration, we are redirected to the login page.

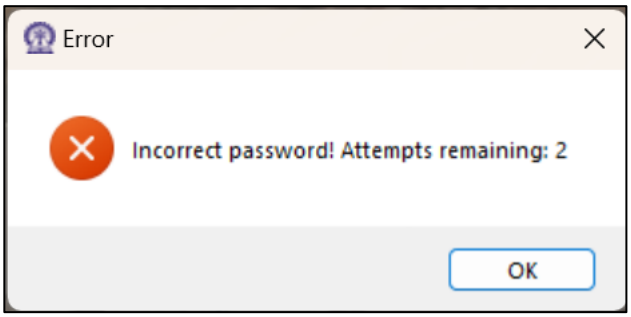


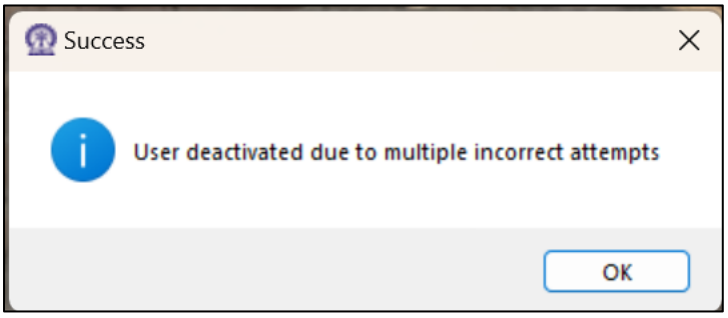


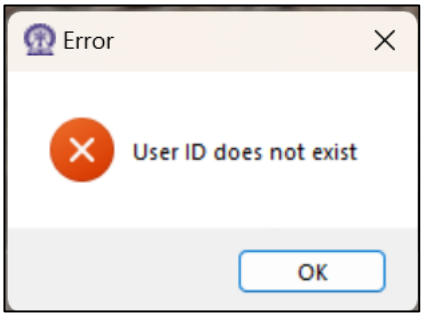
If the login credentials are entered correctly, we go to the dashboard. Else we are left with two more attempts for login, after which the account is deactivated.

Possible Error messages:

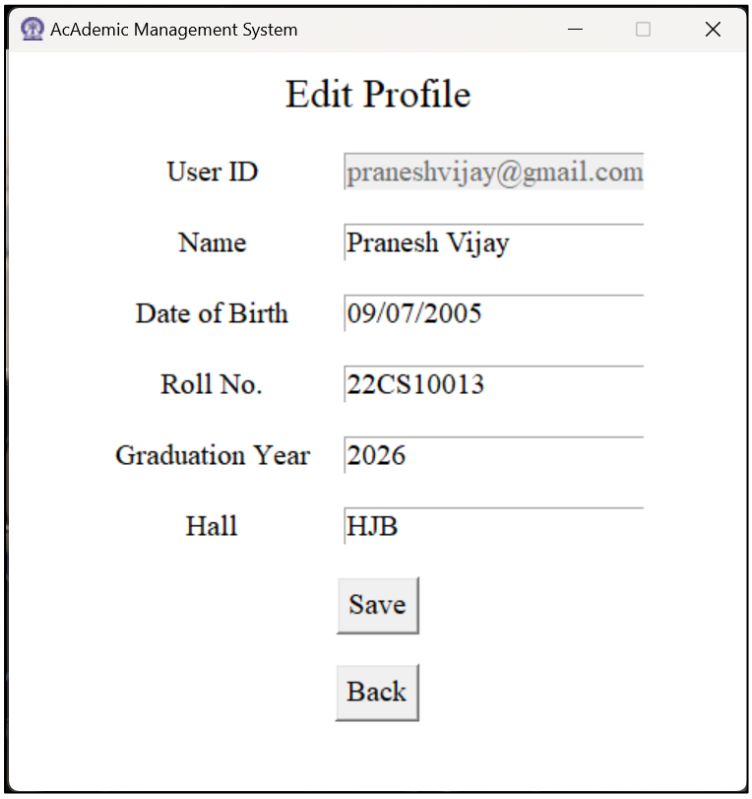


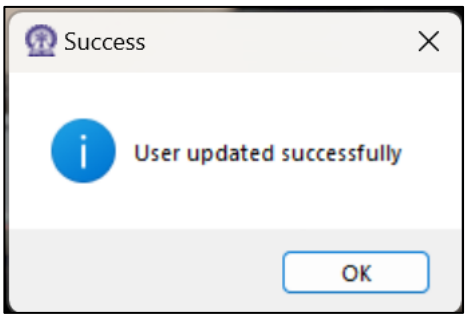




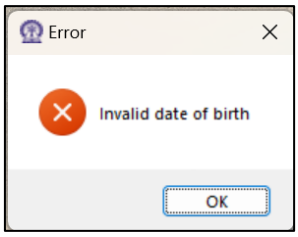


The dashboard has options to edit profile, deactivate account, change password and logout. The edit profile page shows the details of the user depending on their type, we can edit the details and save.

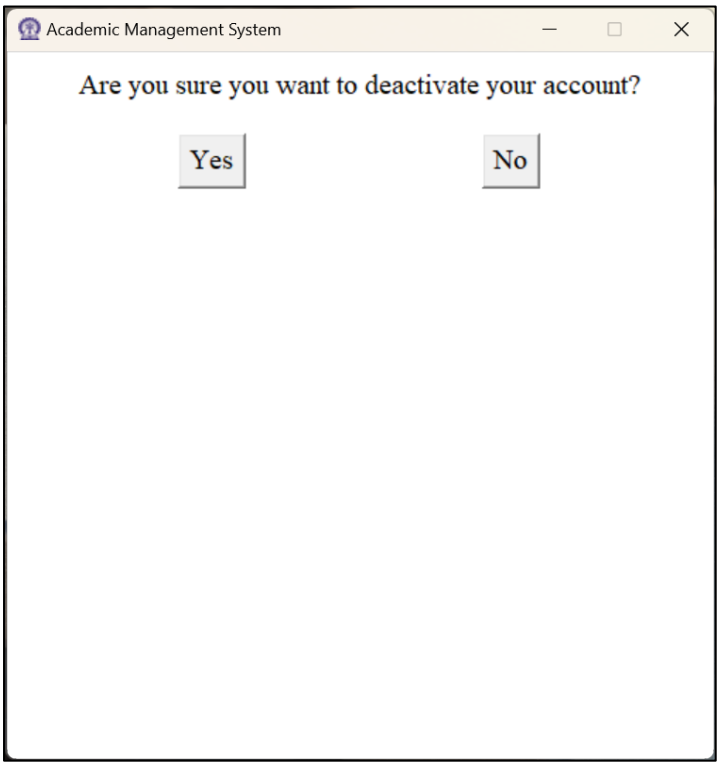


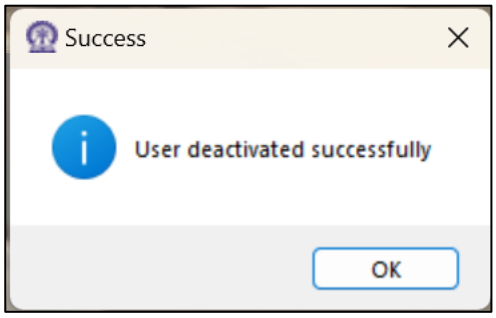


Possible Error Messages:

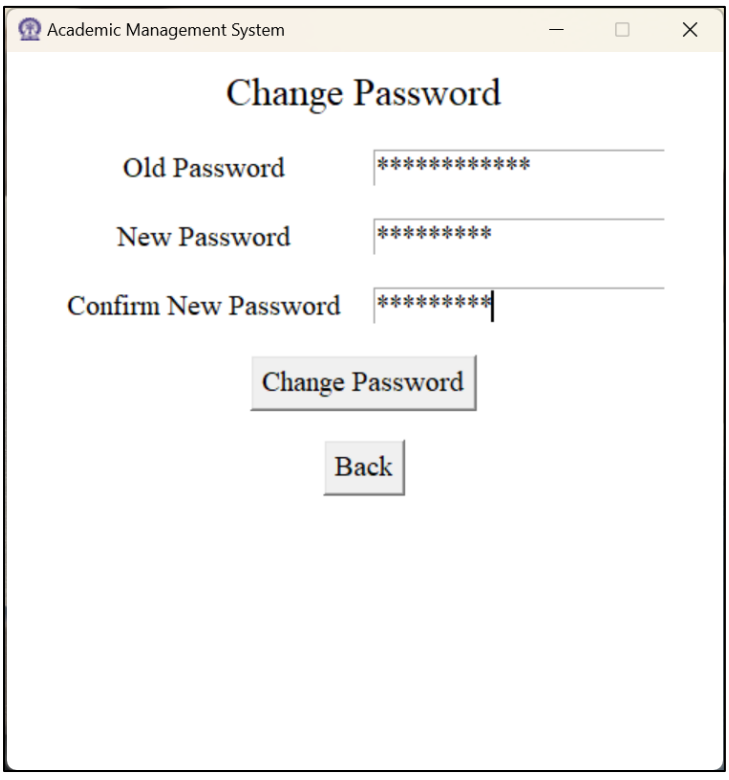


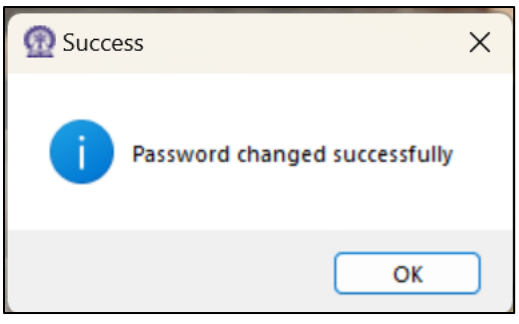
The deactivate page asks for confirmation for deleting the account.



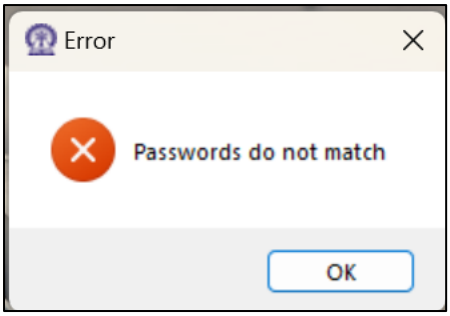


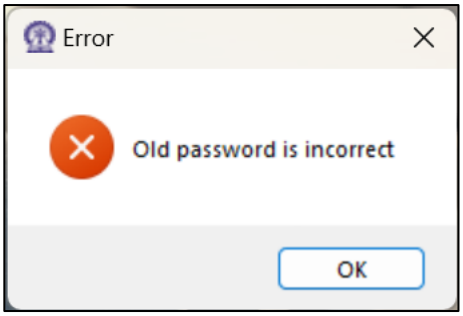
The change password page first verifies your old password and then lets you change it again checking if it is strong.

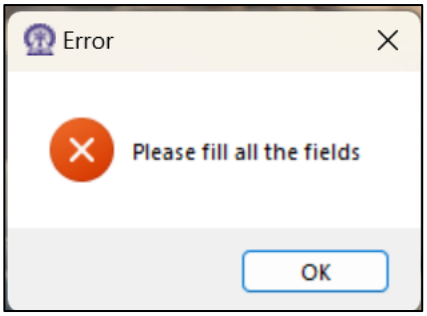




Possible error messages:



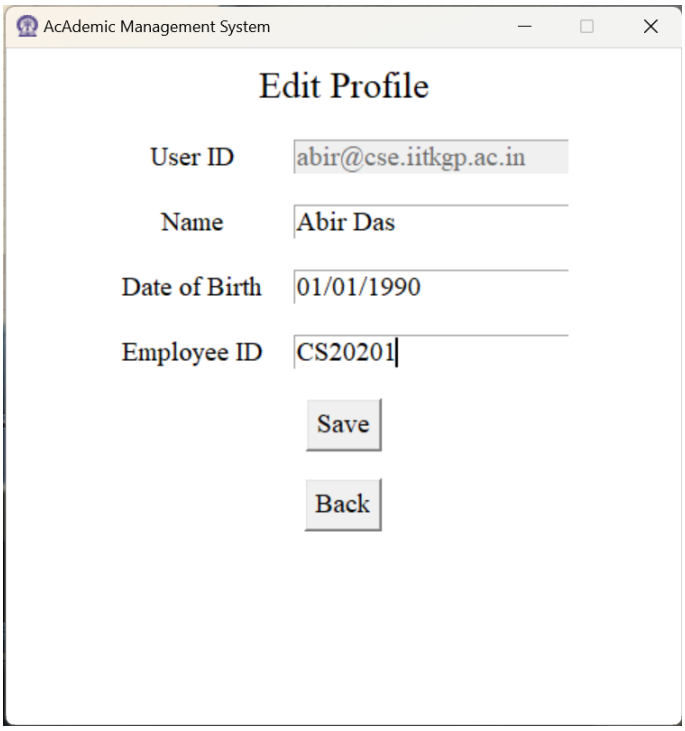




The Logout button take you back to the login page.

Variations for users of various types:

Edir Profile page for teachers:



5. Conclusion:

The implementation, utilizing Python with Tkinter for GUI and SQLite for database management, provides a robust and accessible platform for users. The system's scope, including user registration, authentication, profile management, and deregistration, has been effectively translated into the implementation, ensuring a seamless user experience. The registration process, with password validation and type-specific user details, contributes to data accuracy and system security.

The graphical user interface is intuitive, and the error-handling mechanisms throughout the system enhance user-friendliness. Security features, such as account deactivation after multiple failed login attempts, demonstrate a commitment to safeguarding user data. The provided screenshots offer a visual walkthrough of the system, showcasing its functionality from user registration to profile management and account deactivation. The adaptability to different user types is evident in the variations observed in the edit profile pages.