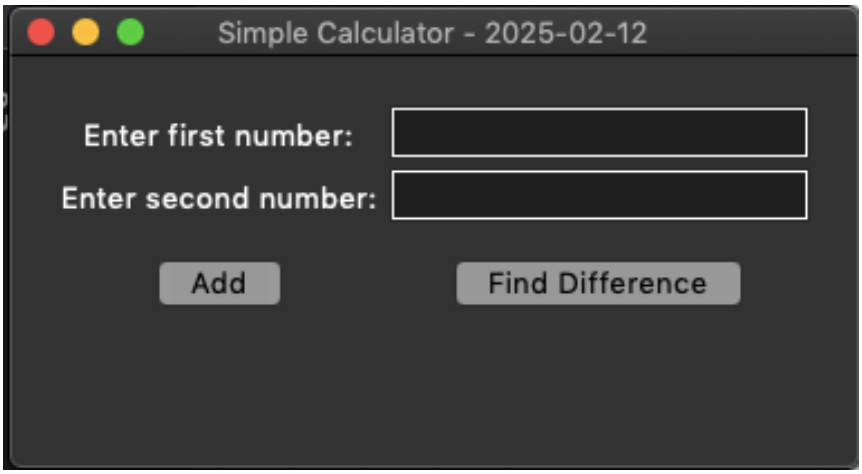
DATABASE MANAGEMENT LAB REPORT
ASSIGNMENT-0

NAME: AYUSH MONDAL
CLASS: BCSE UG-3
SECTION: A2
ROLL: 002210501066

Problem Statement: Develop an application as follows.

User will enter two numbers. Provide button to add or to find the difference. Depending on the option result will be shown in result box. Add a suitable title to the screen. Whenever the form is loaded, also display current date in the title bar.



Components

User Interface Components

- **Main Window (root)**: The primary application window initialized using Tkinter.
- **Frame (frame)**: A container widget for better UI organization.
- **Labels (tk.Label)**: Used to display text prompts for input fields.
- **Entry Fields (tk.Entry)**: Input fields for users to enter numbers.
- **Buttons (tk.Button)**: Two buttons, one for addition and another for finding the difference.
- **Result Display (tk.StringVar and tk.Label)**: Displays the computed result dynamically.

Implementation Details

Functional Implementation

- **Addition Function (add_numbers)**:
    - Retrieves input values from the entry fields.
    - Converts them to floating-point numbers.
    - Computes the sum and updates the result label.

- Handles invalid input cases.

- **Difference Function (find_difference)**:
  - Retrieves input values.
  - Computes the absolute difference between the two numbers.
  - Updates the result label.
  - Handles invalid input cases.

UI Implementation

- **Main Window (root)**:
  - Displays the current date in the title bar.
  - Contains a frame widget to structure UI elements.
- **Grid Layout**:
  - Labels, entry fields, buttons, and the result display are arranged using grid().
  - Buttons are placed in row 2, ensuring easy access.

Execution and Usage

- Run the script using Python.
- Enter two numerical values in the respective fields.
- Click on either the “Add” button or the “Find Difference” button.
- The result is displayed below the buttons.

This application serves as a simple demonstration of Tkinter-based GUI development in Python. The modular design and error-handling mechanisms make it user-friendly and reliable.

Problem Statement: Consider list of departments (dept code and name), list of students (roll, dept code,
name, address and phone) preloaded in array. Now develop an application for the following.
User may add/search/edit/delete/display all student record. While adding, ensure roll must be unique, a list of dept name to be shown from which user selects one and corresponding dept code to be stored. On collecting the data user may choose

CANCEL/SAVE button to decide course of action. For searching user provides roll. If it exists details are shown else suitable message to be displayed. To delete user provides roll. If it does not exist then suitable message is to be displayed. To edit also user provides roll. If it exists user may be allowed to edit any field except roll. User may select CANCEL/SAVE to decide course of action. To display all records, at a time five records are to be shown. IT will also have PREV/NEXT button to display previous set and next set respectively. When first set is displayed PREV button must be disabled and at last set NEXT button must be disabled.

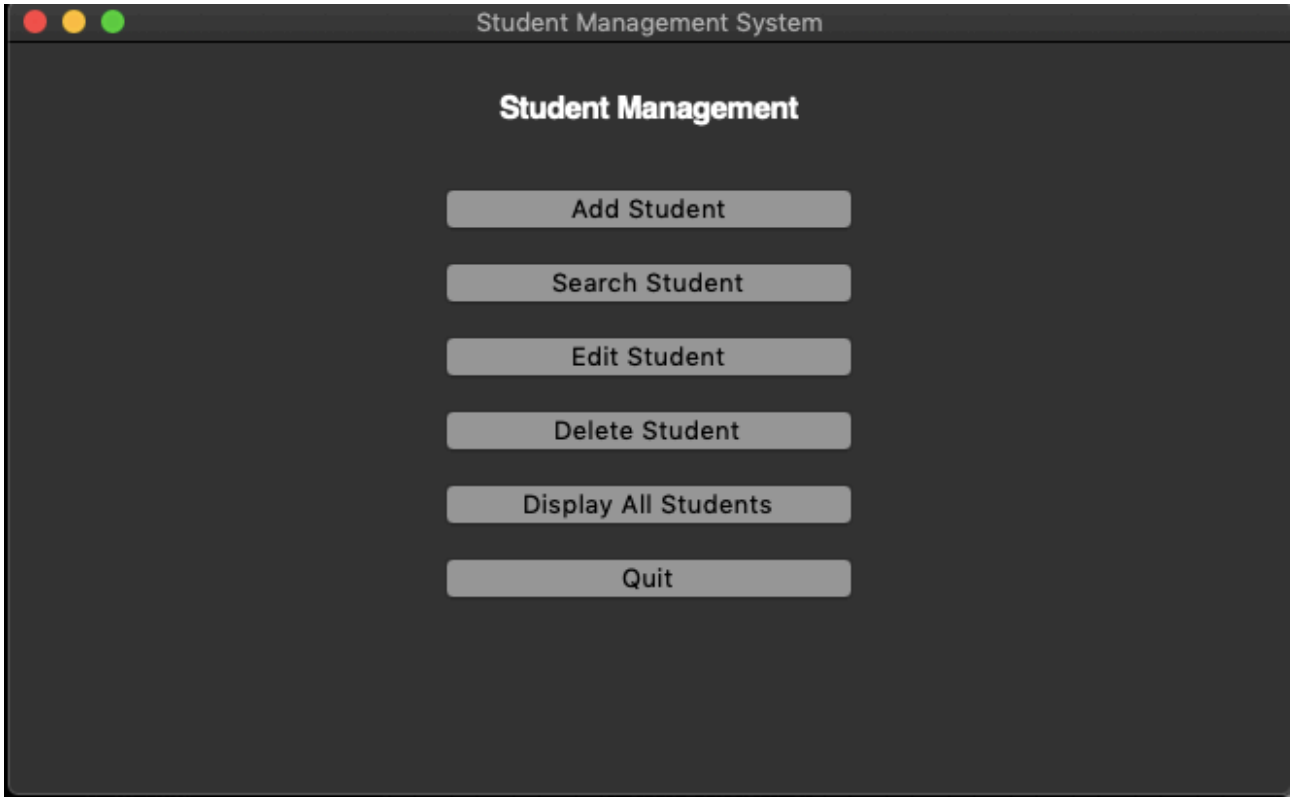
Components:

Data Storage and Handling

- **JSON Storage:**
  All student records are persistently stored in a JSON file. The system loads data from the file if it exists, or initializes the file with an empty list otherwise.
- **Department Data:**
  A maintained collection holds department code–name pairs. This list is used when adding or editing a student record so that the user can select a department via a dropdown menu.

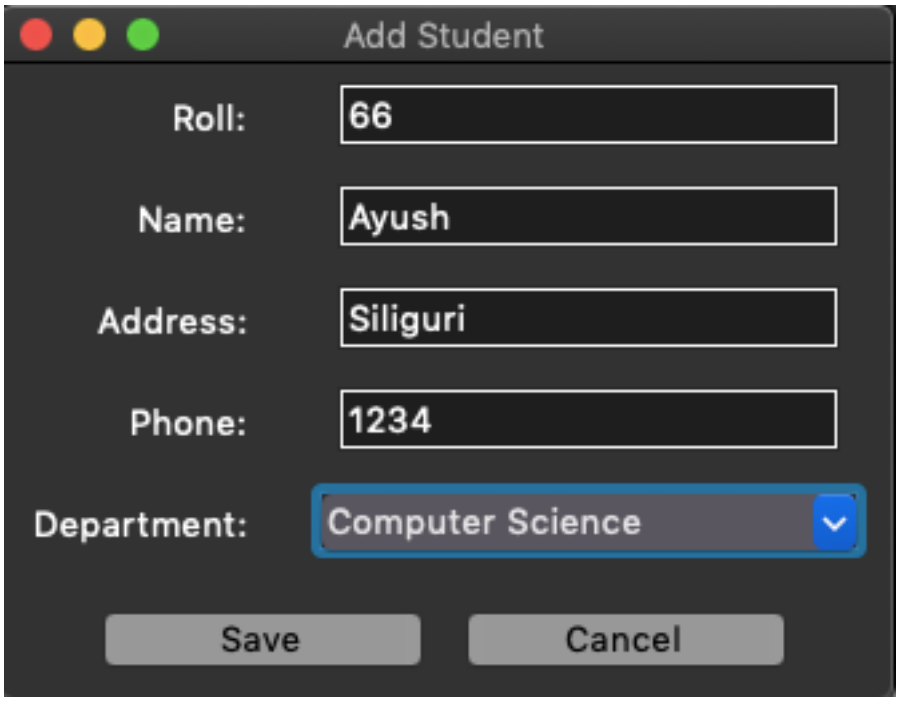
GUI Components

- **Main Window (Home Screen):**



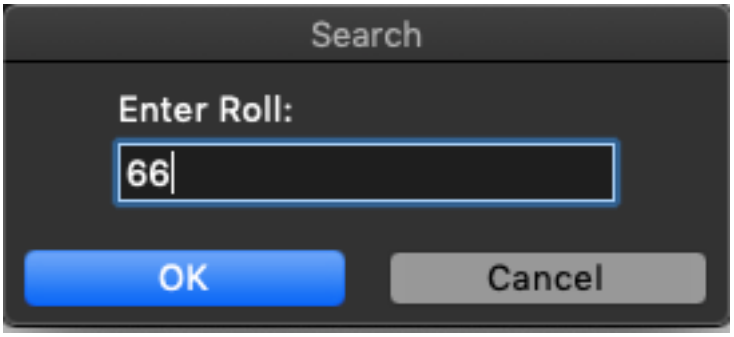
- **Layout:** A centered frame with ample padding.
- **Elements:** A prominent title displayed in a large, bold font and a series of uniformly sized buttons arranged either vertically or in a grid.
- **Actions:** Buttons trigger different operations (Add, Search, Edit, Delete, Display All, and Quit) by opening separate Toplevel windows or executing immediate actions.

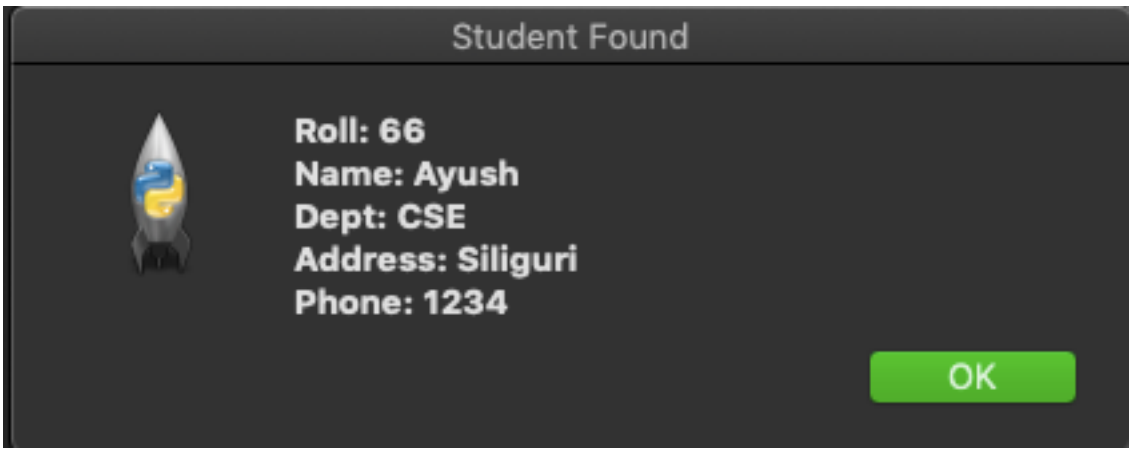
- **Add Student Window:**



- **Layout:** Utilizes a grid layout with two columns—labels on one side and entry fields on the other.
- **Elements:** Input fields for Roll, Name, Address, and Phone. A dropdown (Combobox) presents a list of department names while storing the associated department code.
- **Controls:** “Save” and “Cancel” buttons are placed at the bottom within a dedicated button frame.

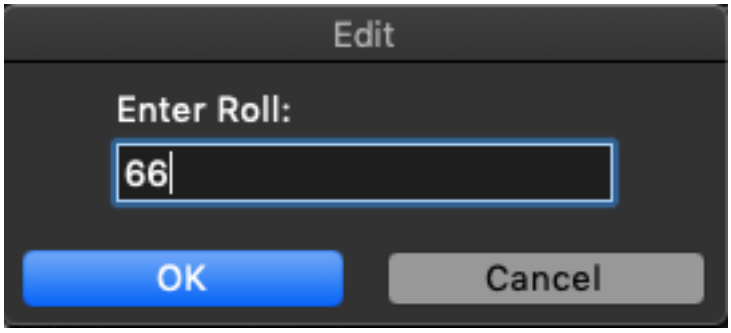
- **Search Student Functionality:**

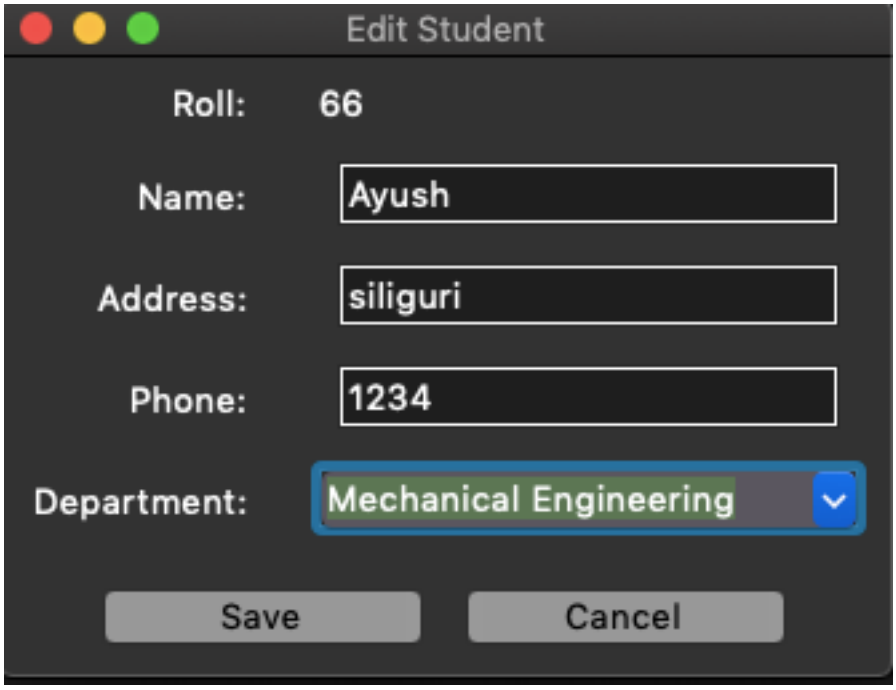




- **Layout:** A simple dialog box prompts the user to enter a roll number.
- **Display:** If a matching record is found, its details (roll, name, department, address, phone) are shown in an informational message box; otherwise, a “not found” message is displayed.

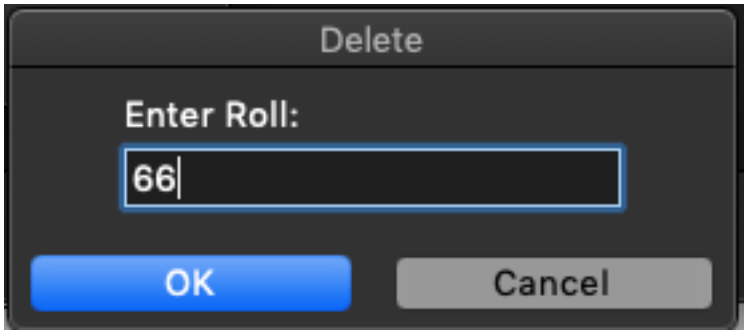
- **Edit Student Window:**





- **Layout:** Similar to the Add Student window, this Toplevel window employs a grid layout with pre-populated fields based on the student's current data.
- **Elements:** A non-editable label shows the roll number, while other fields (name, address, phone, department via dropdown) are editable.
- **Controls:** "Save" and "Cancel" buttons allow the user to commit or discard changes.

- **Delete Student Functionality:**



- **Layout:** A simple dialog requests the roll number of the student to be deleted.
- **Process:** Once the roll is entered, the system verifies the record's existence, prompts for confirmation, and then deletes the record if confirmed.

- **Display All Students (Pagination):**

- **Layout:** A separate window displays records in a Listbox widget, showing five records per page.
- **Navigation:** "Prev" and "Next" buttons are placed at the bottom to navigate between pages. The buttons are dynamically enabled or disabled based on the current page (first or last).

Implementation Details

Data Handling and Storage:

- **Loading and Saving:**
  Dedicated functions manage the reading and writing of student records to the

JSON file. The data is loaded at startup, and any modifications (additions, edits, deletions) are saved immediately to maintain consistency.

- **Data Integrity:**
  The system enforces the uniqueness of the roll number when adding new records. Department information is stored as codes, while the user interacts with department names via a dropdown.

GUI Layout and Interaction:

- **Main Window:**
  The main window is designed to be visually appealing and user-friendly. A centered frame with clear padding contains a title and operation buttons arranged in a grid or vertical stack, ensuring a consistent look.
- **Form Windows (Add and Edit):**
  Both the Add and Edit windows use a two-column grid layout. Labels are aligned on the left and corresponding entry widgets on the right. Consistent padding and spacing create a neat, orderly interface. The department selection is implemented using a Combobox that lists department names.
- **Dialogs and Confirmation:**
  Search, delete, and edit operations utilize simple dialogs and message boxes to prompt the user and confirm actions. This approach provides immediate feedback and ensures that users are aware of successful operations or errors.
- **Pagination Interface:**
  The display window employs a Listbox for listing student records, with navigation buttons (“Prev” and “Next”) at the bottom. The design ensures that only five records are displayed per page, and the navigation buttons are updated dynamically to reflect the current page status (disabling “Prev” on the first page and “Next” on the last page).

User Interaction and Feedback:

- **Validation:**
  Input fields are validated to ensure required information (e.g., roll and name) is provided. Uniqueness of roll numbers is checked before a new record is added.
- **Action Confirmation:**
  Each operation that alters data (add, edit, delete) is accompanied by confirmation messages or dialogs, ensuring that users can verify their actions before finalizing them.
- **Error Handling:**
  The system displays error messages via dialogs if any issue occurs (e.g., duplicate roll numbers, record not found), maintaining a robust user experience.