“PL-3 Project Ideas”

1. Text Analyzer:

Objective: Build a program in F# to analyze text files and provide detailed insights.

Features

1. **Input Handling**:
    - Allow users to input text directly via GUI directly or load a .txt file using F# file-handling functions.
2. **Text Analysis**:
    - Count the number of words, sentences, and paragraphs in the text.
    - Calculate word frequency and display the most frequently used words.
    - Measure text readability, such as average sentence length.
3. **Output**:
    - Display results as you want.
4. **Code Requirements**:
    - Use F# sequences and pattern matching to parse text.
    - Use F# libraries like System.IO for file handling.
    - …… .

2. Simple Store Simulator (User-Only Interaction):

Objective: Create a virtual store management system using F#.

Build a simple store simulation where users can browse products, add them to a cart, and calculate total costs.

Features

1. **Product Catalog (Displayed to User)**:
    - The store will have a list of available products, each with the following details:
        - **Name**
        - **Price**
        - **Description**
    - The list of products will be stored in an F# collection (e.g., List or Map) which will be initialized by you in the F# code.
2. **User Interaction**:
    - **Browse Products**: The user will be able to view the products available in the store.
    - **Add to Cart**: Users can select items by name and add them to their cart.
    - **View Cart**: The user can view the items in their cart at any time.
    - **Checkout**: Once the user is done shopping, they can view the total price of the items in their cart.
3. **Cart Management**:
    - The cart will be represented as a list of product names.
    - Users can add and remove products from their cart.
    - At checkout, the program will calculate the total cost by summing the prices of the items in the cart.

3. Student Grades Management System:

Objective: Build a program in F# to manage and analyze student grades.

Features

1. **Student Database**:
   - Store student data (ID, name, and grades) in F# Record or List structures.
   - Support adding, editing, and removing student records.

2. **Grade Management**:
   - Calculate individual student averages and class-wide statistics (including pass\fail rate ,.....) .
   - Identify the highest and lowest grades in the class.

3. **User Roles**:
   - **Admin**: Full control over the database and grade editing.
   - **Viewer**: Read-only access to grades and reports.

---

4. Dictionary:

Objective: Build a digital dictionary using F# for managing and searching word definitions.

Features

1. **Word Management**:
    - Store words and their definitions in an F# Map.
    - Allow adding, updating, and deleting entries.
2. **Search Functionality**:
    - Enable case-insensitive searches by word or partial keyword.
3. **Output**:
    - Save dictionary data to a file (e.g. → .json or .xml file , ...) and reload it when needed.

---

5. Cinema Seat Reservation System:

Objective: Build a cinema seat booking system using F#.

Features

1. **Seat Layout**:
    - Represent the seating chart as a 2D array or a list of tuples.
    - Display available and reserved seats in a basic GUI.
2. **Booking System**:
    - Allow users to select seats by row and column indices.
    - Mark seats as reserved and prevent double-booking.
3. **Ticket Management**:
    - Generate a unique ticket ID for each booking.
    - Save ticket details (e.g., seat, showtime, customer name) to a file.

6. Library Management System

Idea: Develop a simple program to manage a library's books.

Tasks:

- **Add a new book:**
  Allow users to add a new book with details like **title**, **author**, and **genre**.
- **Search for a book:**
  Enable users to search for books by **title**.
- **Borrow a book:**
  Let users borrow a book and record the **borrow date**.
- **Return a book:**
  Allow users to return a borrowed book and update its status.
- **Display available and borrowed books:**
  Show all the books in the library, with clear labels indicating whether they are **available** or **borrowed**.

Data organization:

Use **Record** or **Map** to organize the books and their status (borrowed/available).

Accurate borrowing and returning:

Handle borrowing and returning books correctly to ensure a book is not borrowed twice at the same time.

User Interface (UI):

Create a **Windows Forms** interface where users can interact with the system to add books, search, and borrow/return books.

7. Quiz Application:

Idea: Develop a quiz program where users can take a quiz, answer questions, and get scores.

Tasks:

- **Create quiz questions** with ***multiple-choice*** answers and ***some questions are written.***
- **Store correct answers** for comparison.
- **Track scores** as users answer questions.
- **Display results** after the quiz is finished.

- ➢ Use **Map** or **Record** to store questions and answers.
- ➢ Implement **functions** to calculate the score based on correct answers.
- ➢ Create a **Windows Forms** UI to display the questions and collect user answers.

---

8. Contact Management System:

Idea: A program to manage a list of contacts, including adding, updating, and deleting contacts.

Tasks:

- **Add new contacts** with details like name, phone number, and email.
- **Search for contacts** by name or phone number.
- **Edit contact details**.
- **Delete contacts**.

Evaluation:

- Use **Map** or **Record** to store contact information.
- Implement search and update functionality with **functional programming techniques**.
- Create a **Windows Forms** UI for managing contacts.

Notes:

- The **programming language** you will use is **F#**, which is part of the .NET framework. It's powerful for solving problems using functional programming techniques (as you know).

- **The UI (User Interface)** will be created using **Windows Forms** in F#. This will allow you to design windows where users can click buttons, enter text, and interact with the system.

- You will be using **Functional Programming concepts** such as:
  - **Immutable data**: Data that cannot be changed once it's created.
  - **First-class functions**: Functions that can be passed around like any other value.
  - **Pattern matching**: A way to handle different types of data or cases easily.

- You will also use **GitHub** to manage and store your code. GitHub helps you:
  - Track changes to your code.
  - Collaborate with others and share your work.
  - Organize your projects and keep them safe.