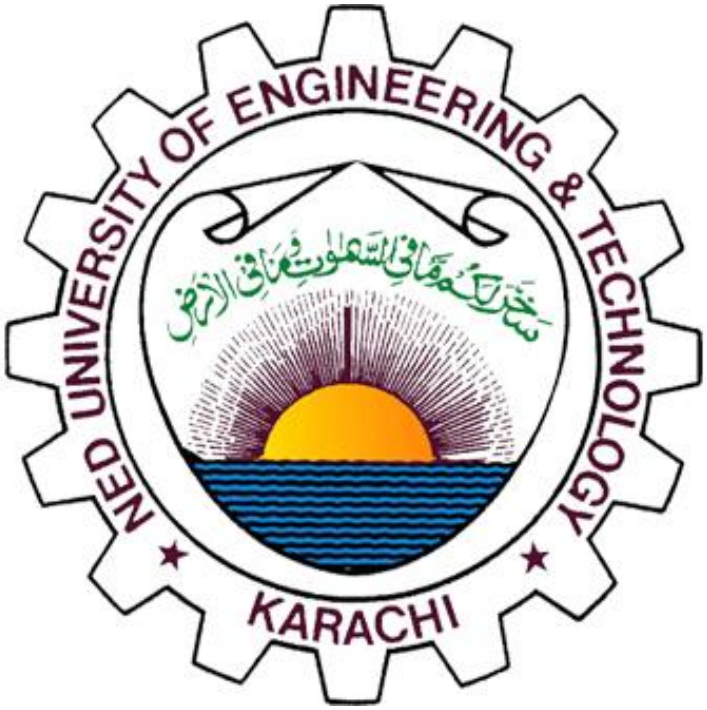
DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-115
Course Title: Computer Programming
Complex Engineering Problem
FE Batch 2024, Fall Semester 2024
Grading Rubric
TERM PROJECT Group

Members:

Student No.	Name	Roll No.
S1		
S2		
S3		

CRITERIA AND SCALES				Marks Obtained		
				S1	S2	S3
Criterion 1: Does the application meet the desired specifications and produce the desired outputs? (CPA-1, CPA-3) **[8 marks]**						
1	2	3	4			
The application does not meet the desired specifications and is producing incorrect outputs.	The application partially meets the desired specifications and is producing incorrect or partially correct outputs.	The application meets the desired specifications but is producing incorrect or partially correct outputs.	The application meets all the desired specifications and is producing correct outputs.			
Criterion 2: How well is the code organization? **[2 marks]**						
1	2	3	4			
The code is poorly organized and very difficult to read.	The code is readable only to someone who knows what it is supposed to be doing.	Some part of the code is well organized, while some part is difficult to follow.	The code is well organized and very easy to follow.			
Criterion 3: How friendly is the application interface? (CPA-1, CPA-3) **[2 marks]**						
1	2	3	4			
The application interface is difficult to understand and use.	The application interface is easy to understand and but not that comfortable to use.	The application interface is very easy to understand and use.	The application interface is very interesting/ innovative and easy to understand and use.			
Criterion 4: How does the student performed individually and as a team member? (CPA-2, CPA-3) **[4 marks]**						
1	2	3	4			
The student did not work on the assigned task.	The student worked on the assigned task, and accomplished goals partially.	The student worked on the assigned task, and accomplished goals satisfactorily.	The student worked on the assigned task, and accomplished goals beyond expectations.			
Criterion 5: Does the report adhere to the given format and requirements? **[4 marks]**						
1	2	3	4			
The report does not contain the required information and is formatted poorly.	The report contains the required information only partially but is formatted well.	The report contains all the required information but is formatted poorly.	The report contains all the required information and completely adheres to the given format.			
			Total Marks:			

_____
Teacher's Signature



Complex Engineering Problem

(CS-115)

Group Memembers:

CS-137 Muzzammil

CS-145 Shahmir

CS-130 Misbah

Contents

Project Report: Command-Line Database Management System

1. Problem Description

The project implements a **Command-Line Interface (CLI)-based Simple Database Management System (DBMS)** in Python. The primary goal of this system is to provide users with a simple, efficient way to create and manage databases with customizable fields. The database uses JSON format for storage, allowing flexibility in how data is organized and stored. The system supports essential database operations like Create, Read, Update, and Delete (CRUD).

The problem this project addresses is providing an easy-to-use tool for managing small-scale databases where users can:

- Define the structure of the database with user-defined fields.
- Perform CRUD operations on database records.
- Manage multiple databases at once.
- Use simple commands to interact with databases without the need for complex SQL commands or database management software.
-

Sample Output

```
+--------------+-----------+-----+--------+-----------+
| patient_name | D-O-B     | Age | gender | address   |
+--------------+-----------+-----+--------+-----------+
| zia          | 1/12/1999 | 33  | M      | nazimabad |
| khan         | 5/3/2000  | 45  | M      | Malir     |
| najma        | 5/4/1992  | 76  | F      | gulshan   |
+--------------+-----------+-----+--------+-----------+
```

```
+---------+----------+--------+-----------+-------+
| Roll No | fname    | lname  | address   | class |
+---------+----------+--------+-----------+-------+
| 1       | muzammil | khalid | fb area   | X     |
| 2       | misbah   | hassan | nazimabad | IX    |
| 3       | shahmir  | khan   | johar     | V     |
+---------+----------+--------+-----------+-------+
```

2. Distinguishing Features of the Project

Several unique features make this project stand out:

2.1 User-friendly CLI

The application is designed with simplicity in mind. It provides a **text-based user interface (CLI)** that allows users to navigate through database operations with simple options and easy-to-understand prompts. The system guides users step-by-step through the process of creating and interacting with databases.

2.2 Customizable Databases

One of the distinguishing features of this project is that users can define their own database schema. When creating a database, users specify the field names and the maximum allowed length for each field, providing flexibility for different types of data storage.

2.3 Dynamic CRUD Operations

The application supports dynamic CRUD operations on user-created databases. This means:

- Users can add records with data that conforms to the field constraints.
- Users can edit existing records by modifying individual fields.
- Users can delete records by specifying their index in the list.
- Users can view all records stored in a database, formatted neatly in a table.

2.4 Robust Error Handling

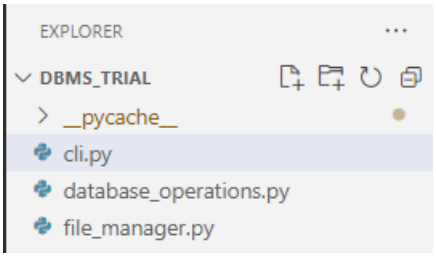
The system includes several layers of validation to ensure data integrity and smooth operation. If users attempt to input data that exceeds field length constraints or perform actions on nonexistent databases, the program provides informative error messages to guide the user.

2.5 Modular Structure

The project is split into three primary modules:

- **CLI.py**: Handles the main user interface, displaying menus and processing user input.
- **file_manager.py**: Manages the database files, including creating, deleting, and loading data from JSON files.
- **database_operations.py**: Contains the logic for performing CRUD operations on the data stored in the databases.

This modular design enhances the maintainability of the system, allowing for easier updates or future feature additions.



3. Flow of the Project

The project is structured to guide users through various stages of database management via a simple, menu-driven interface.

3.1 Main Menu

The main menu presents the following options to the user:

1. **Create a New Database**: Prompts the user for a database name and the fields to define the database schema (e.g., field names and maximum lengths).
2. **Open an Existing Database**: Allows the user to perform operations like adding, editing, deleting, and viewing records.
3. **Delete a Database**: Allows the user to select a database and delete it from the system.
4. **Exit**: Exits the application.

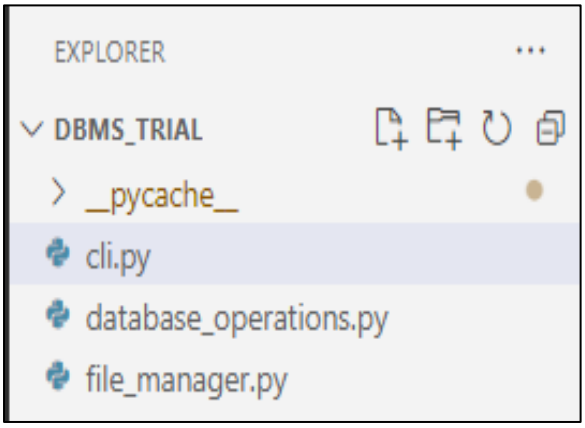
3.2 Database Menu

After opening a database, the user is presented with the following options:

1. **Add a Record**: Prompts the user to input data for each field.
2. **Edit a Record**: Allows the user to select an existing record and modify its values.
3. **Delete a Record**: Allows the user to select a record to delete from the database.
4. **View All Records**: Displays all records in a tabular format.
5. **Back to Main Menu**: Returns the user to the main menu.

3.3 How to Use the Software

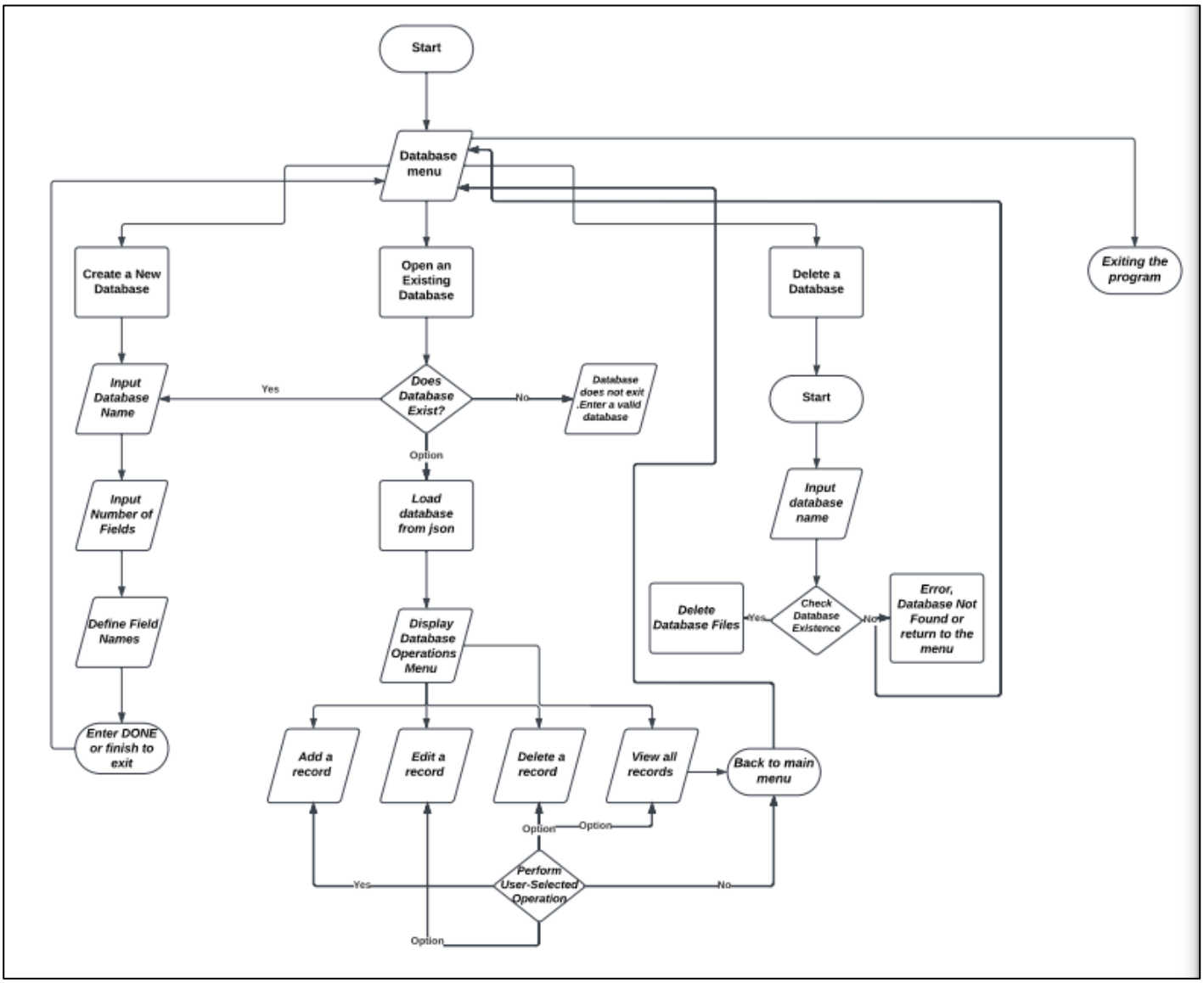
- **Running the Program**: To begin using the DBMS, the user must run the `CLI.py` script from the terminal or command prompt.
- **Navigating Menus**: The program will display a series of menus prompting the user to choose different operations. The user can select an option by entering the corresponding number.
- **Performing Operations**: After selecting an option, the program will guide the user through the necessary steps to perform the operation, whether it's creating a database, adding records, or viewing/editing data.




```
Simple DBMS Main Menu
1. Create a new database
2. Open an existing database
3. Delete a database
4. Exit
Select an option: 2
Available Databases:
1. sample
2. student_info
Enter the name of the database to open: sample

Database Menu - sample
1. Add a record
2. Edit a record
3. Delete a record
4. View all records
5. Back to Main Menu
Select an option:
```

3.4 Flow Chart



4. Most Challenging Part

The most challenging part of this project was implementing the **dynamic CRUD system** that adapts to the user-defined database schema. Specifically, the challenges included:

- **Field Validation**: Ensuring that user inputs respect the maximum length constraints defined for each field. For example, if a user enters a name longer than the allowed limit, the system should reject the input and prompt the user to provide a valid entry.
- **Data Integrity**: Ensuring that the records added, modified, or deleted from a database are correctly saved to the corresponding JSON file, and that any operation that modifies the file does not corrupt the database.
- **Error Handling**: Handling edge cases such as attempting to delete a record that doesn't exist, entering invalid data types, or attempting to perform operations on a non-existent database.

---

5. New Things Learned in Python

Working on this project allowed me to gain new insights and skills in Python programming:

5.1 Working with JSON Files

I learned how to use Python's `json` module to read and write structured data in JSON format. This included handling file I/O, serializing Python objects to JSON, and deserializing JSON data back into Python objects.

5.2 File System Operations

Using Python's `os` module, I learned how to interact with the file system to check for the existence of files, create new files, and delete files. This was crucial for managing database files and ensuring the application works as intended.

5.3 Modular Programming

I gained a deeper understanding of **modular programming** by dividing the system into multiple Python files (`CLI.py`, `file_manager.py`, `database_operations.py`). This structure improved the code's readability, maintainability, and scalability.

5.4 Input Validation and Error Handling

I honed my skills in **input validation** and **error handling** by ensuring that all user inputs are validated before being processed and by adding try-except blocks to handle any potential errors gracefully.

---

6. Individual Contributions

- **Misbah CS-130**: Developed the main CLI interface, including the menus and user prompts. Ensured the smooth navigation between different operations.

- **Shahmir CS-145**: Focused on the CRUD operations, implementing functions for adding, editing, deleting, and viewing records in the database.

- **Muzzammil CS-137**: Handled the file management tasks, including creating, reading, saving, and deleting database files. Ensured data integrity and proper error handling.

7. Future Expansions

While the current system is functional, there are several areas where it could be expanded:

1. **User Authentication**: Adding user authentication to restrict access to certain databases or operations.

2. **Graphical User Interface (GUI)**: Building a GUI to provide a more intuitive user experience, especially for non-technical users.

3. **Advanced Query Support**: Implementing more advanced query features, such as searching for records based on specific field values or filtering records.

4. **Backup and Recovery**: Implementing a system to backup databases and recover lost data in case of corruption or accidental deletion.

8. Test Case Runs

Test Case 1: Create a New Database

- **Input**: Database Name: `contacts`, Fields: `name (50 chars),phone (15 chars)`
- **Output**: Successfully created database `contacts` with specified fields.

```
Simple DBMS Main Menu
1. Create a new database
2. Open an existing database
3. Delete a database
4. Exit
Select an option: 1
Enter the name of the new database: contacts
Enter field name (or type 'done' to finish): name
Enter maximum length for field 'name': 50
Enter field name (or type 'done' to finish): phone
Enter maximum length for field 'phone': 15
Enter field name (or type 'done' to finish): done
Database 'contacts' created successfully.
Database 'contacts' created successfully with fields: {'name': 50, 'phone': 15}
```

Test Case 2: Add Record

- **Input**: Database: `contacts`, Record: `{name: "Taimoor", phone: "123456789"}`
- **Output**: Successfully added record to `contacts`.

```
1. Create a new database
2. Open an existing database
3. Delete a database
4. Exit
Select an option: 2
Available Databases:
1. contacts
2. sample
3. student_info
Enter the name of the database to open: contacts

Database Menu - contacts
1. Add a record
2. Edit a record
3. Delete a record
4. View all records
5. Back to Main Menu
Select an option: 1
Enter value for 'name' (max 50 chars): Taimoor
Enter value for 'phone' (max 15 chars): 123456789
Record added successfully.
```

Test Case 3: Delete Record

- **Input**: Database: `contacts`, Record Index: `1`
- **Output**: Successfully deleted record from `contacts`.

```
Database Menu - contacts
1. Add a record
2. Edit a record
3. Delete a record
4. View all records
5. Back to Main Menu
Select an option: 3
Enter the index of the record to delete: 1
Are you sure you want to delete record 1? (yes/no): yes
Record 1 deleted successfully.
```