**My Digital Library**

**Project Overview 📝**

My Digital Library is a console-based application built using C++. It uses forward\_list data structure to store the records data in binary files and also give flexibility to user to export the data from binary to csv and vice-versa. Furthermore, it also provides various command line arguments to interact with application and it includes various features as follows:

* Command line arguments
* Perform CURD Operations
* Store Books Based on Categories
* Search Books Based on Category, Author Name and Book Title
* Smooth Architecture
* Save Records with Unique ID's and more.

**Structure BookModel:** It is a structure type ADT which defines the schema for My Digital Library. It is responsible to input data and store it into member variables with unique ids and display them.

**Utility Class:** This class contains reusable functions i.e., reading/writing data to binary and csv files, also it holds frontend menus, all member functions it includes are as follows:

* int getBinaryFileLength(const char\* filename); // return length of records in binary file
* void writeToBinaryFile(const char\* filename, std::forward\_list<BookModel>& books, int size); // write records to binary file
* void readFromBinaryFile(const char\* filename, std::forward\_list<BookModel> &books, int size); // read records from binary file
* int getCSVFileLength(const char\* filename); // return length of records in csv file
* void readFromBinaryAndWriteToCSV(const char\* filename, int size); // write binary records to csv file
* void readFromCSVFileAndWriteToBinary(const char\* filename); // write csv records to binary file
* void toLowerCase(char\* str); // convert char string to lowercase
* void home\_page\_menu(const char\* page\_title) const; // home page menu
* void search\_page\_menu(const char\* page\_title) const; // search page menu
* void help\_page\_menu(const char\* page\_title) const; // help page menu
* void argumentsList() const; // show all available arguments to the user

**BACKEND Class:** This class holds the application logic and it is designed to perform the CURD operations, it hold various member functions as follows:

* void GET(const char\* filename); // display all the available records
* void POST(const char\* filename, int size); // post a record
* void PUT(const char\* filename, int index, int size); // update a record
* void DELETE(const char\* filename, int index, int size); // delete a record
* void SearchByCategory(const char\* filename, const char \*category); // Search record based on category
* void SearchByBookName(const char\* filename, const char \*book\_title); // search record using book name
* void SearchByAuthorName(const char\* filename, const char \*author\_name); // search record using author name

**FRONTEND Class:** This class is designed for display of contents as it serve the different pages based on users commands, it has following member variables as follows:

* void home\_page(const char\* filename, const char\* menu\_title); // display homepage
* void search\_page(const char\* filename); // display search page
* void help\_page(const char\* filename); // display help page

Besides these classes there are few other useful functions that handle various things as follows:

* void RunApp(const char\* filename); // entry point function to initialize the application.
* void handleFirstTimeRun(const char\* filename); // run if there is no records available to the system
* void argumentsHandler(const char \*filename, int records\_size, int argc, char\* argv[]); // Execute only if user has passed the arguments
* bool FindByUID(const char\* filename, int uid); // validate unique id while creation of the record

The main function contains the utility object and application get the file binary file size and call functions accordingly.

**Getting Started 🛠️**

To get started with this project follow these steps:

1. **Clone this repository:**

git clone <https://github.com/saqibbedar/dsa.git>

cd .\dsa\Assignments\03\_Assignment

1. **Compile and run C++ program:**

g++ -o info info.cpp // compile C++ program

.\info.exe // you can pass arguments i.e., .\info.exe --help (optional)

1. **Compile and run C++ program in isolated container using Docker (optional):**

If you are docker user then you can build a docker image and run this program in an isolated environment. To run C++ using docker follow the instructions below correctly, the **Dockerfile** is already setup in 03\_Assignment directory, so just run these commands to create docker image and start a container and test the application.

**Step 1: Build docker image:**

docker build -t <docker-image-name> . // replace <docker-image-name> i.e., main-cpp etc.

**Step 2: Run your C++ program:**

docker run --rm -it main-cpp

**Contributions 🤝**

If you'd like to contribute:

* You can add your code to the [Contribute](https://github.com/saqibbedar/DSA/tree/main/Contribute) directory.
* For more extensive contributions, please create a separate repository with your organized code.
* If you find any errors in my solutions or have improvements, feel free to suggest updates.
* Please refer to the [CONTRIBUTE.yaml](https://github.com/saqibbedar/dsa/blob/main/Contribute/CONTRIBUTE.yaml) file for detailed contribution guidelines.

**License 📄**

This project is licensed under MIT License – see the [LICENSE](https://github.com/saqibbedar/dsa?tab=MIT-1-ov-file) file for more details.