**Ahmed Merza**

**CST2550**

**M00664455**

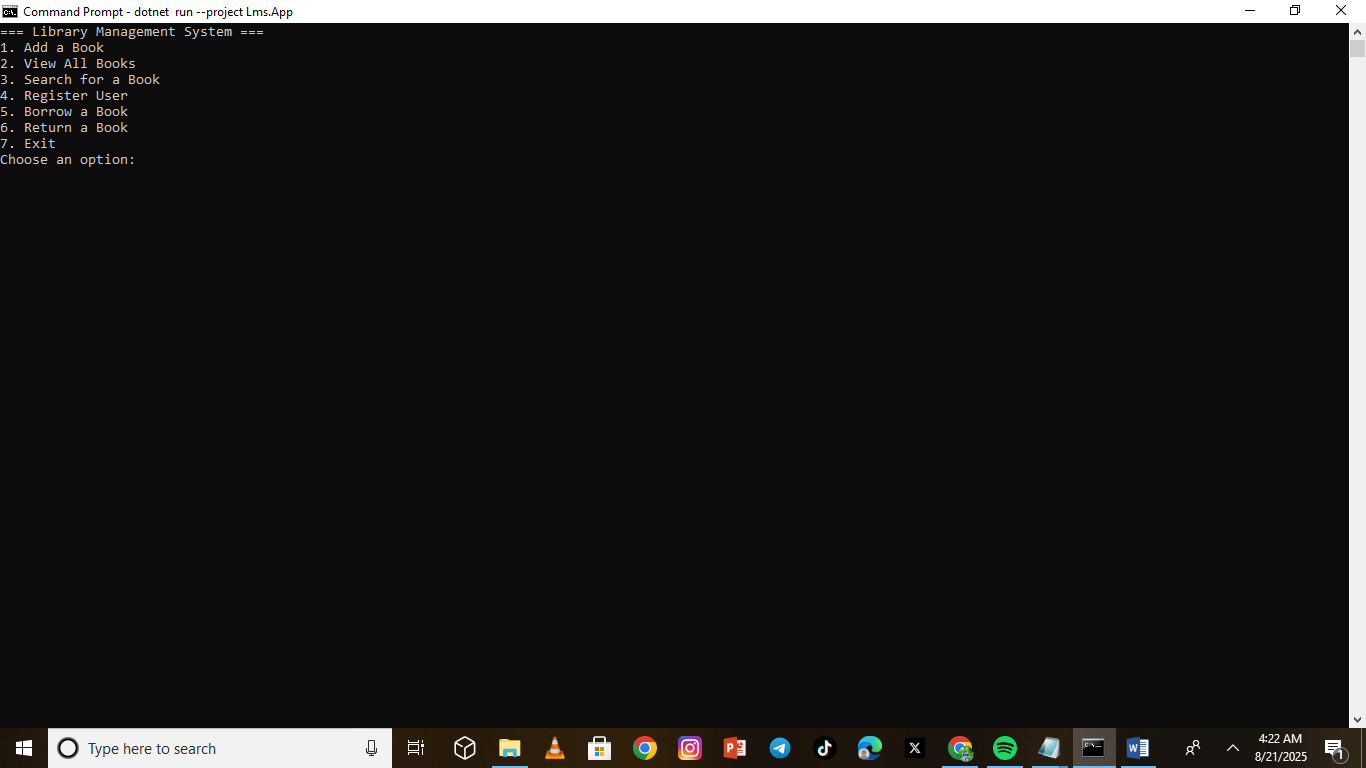
# Library Management System

## 1. Introduction

## Libraries play a vital role in storing and providing access to knowledge and information, but many community and small libraries still rely on manual systems for cataloguing and loan management. These systems are often slow, error-prone, and inconvenient for both librarians and users (Daphine, 2013). To address these challenges, this project focused on developing a digital Library Management System (LMS) using C#, .NET, and SQL. The LMS provides an efficient and structured way to manage books, users, and loans, while also enabling search functionality and borrowing/returning workflows. The project was implemented using a three-layer architecture consisting of a core library (Lms.Core), a data layer (Lms.Data), and a console application (Lms.App). Git was used for version control and GitHub was used as the collaboration platform when tracking changes and ensuring project transparency. The reason as to why Git was important in this project is to manage changes in your code or project over time (Blischak et al 2016).

## 2. Project Objectives

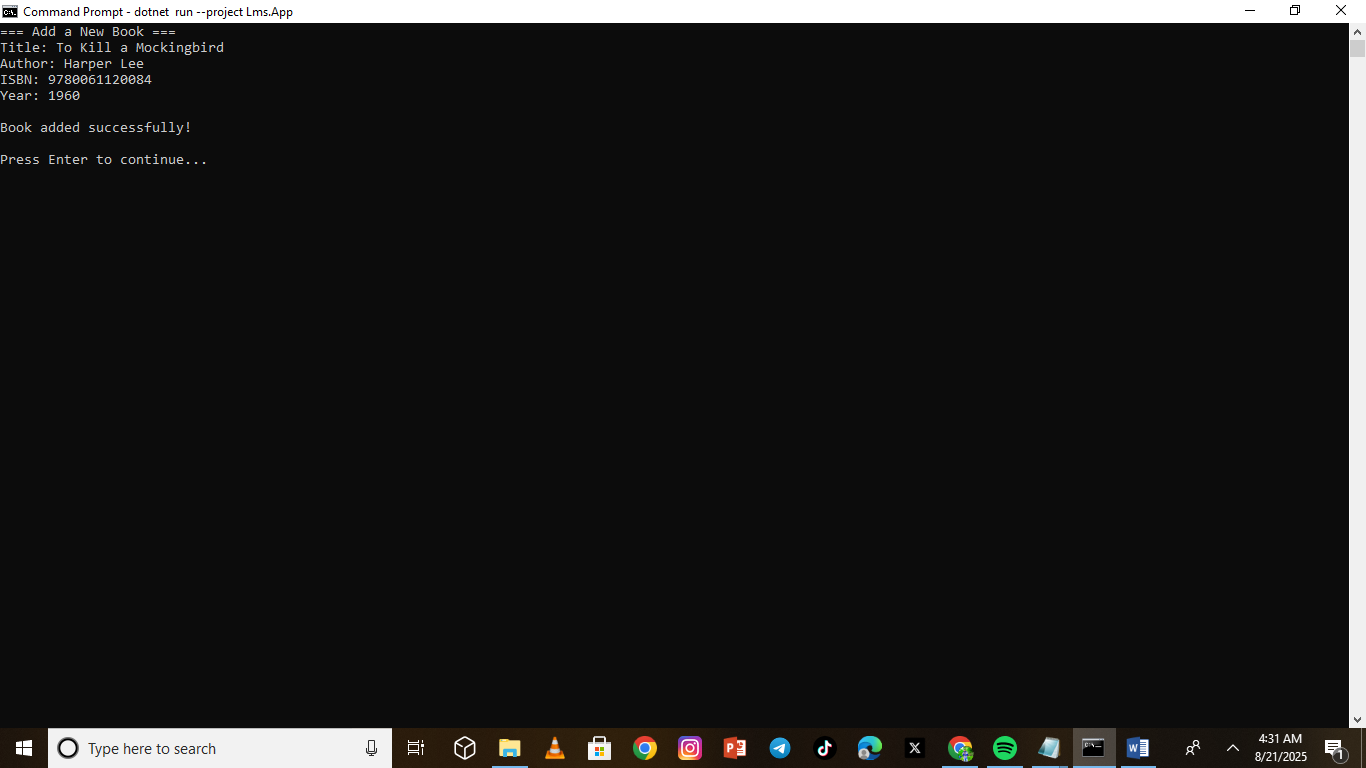
The main objectives of this project were:  
- To design and implement a digital Library Management System that replaces paper-based methods.  
- To provide essential features such as adding, viewing, and searching for books.  
- To allow user registration and management of borrowing and returning books.  
- To ensure scalability and integration with a relational database (SQLite).  
- To demonstrate software engineering practices such as database design, modular coding, testing, and version control. Below is a screenshot of the Program Menu in CMD after I successfully run the project.



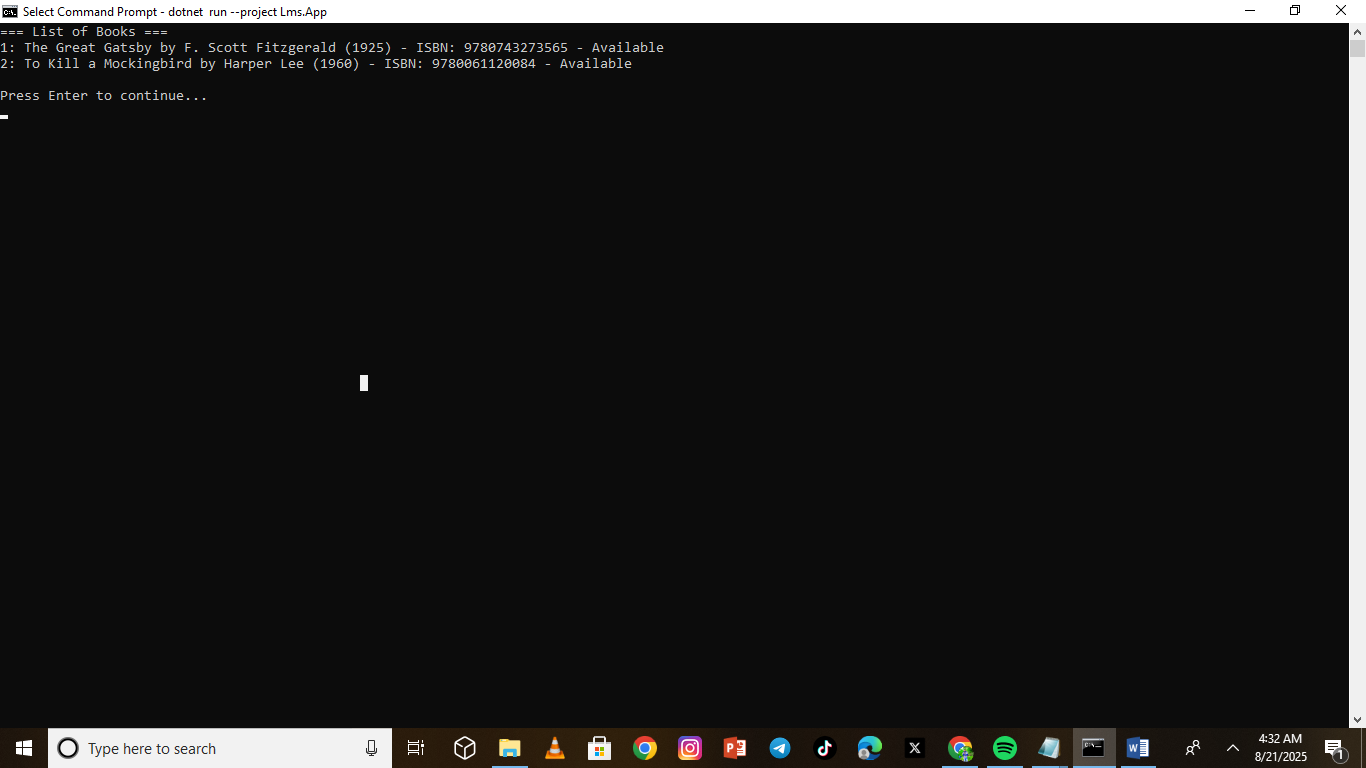
## 3. System Design & Implementation

The project was developed using a structured three-layer architecture:  
1. Lms.Core – This layer contains the core domain classes such as Book, User, and Loan. These classes model the real-world entities of a library system.  
2. Lms.Data – This layer handles database operations. A DatabaseHelper class initializes the database using a schema.sql file and provides functions for interacting with SQL.  
3. Lms.App – This is the console application that provides a user interface for interacting with the system. It implements features such as adding books, registering users, borrowing and returning books.  
  
The schema.sql file defined the structure of the database with three tables: Books, Users, and Loans. Books contained attributes like title, author, ISBN, year, and availability status. Users had name and email attributes. Loans connected users and books with loan date, due date, and return date.  
  
SQL was chosen as the database because of its simplicity and integration with .NET projects. The project used MicrosoftData.Sqlite as the data provider package. GitHub was used to manage commits and ensure all progress was documented. I will provide two screenshots below showing me adding a book and then after adding the book I proceed to view all the books available to confirm if the book I just added is showing under all the books available in my stytem.

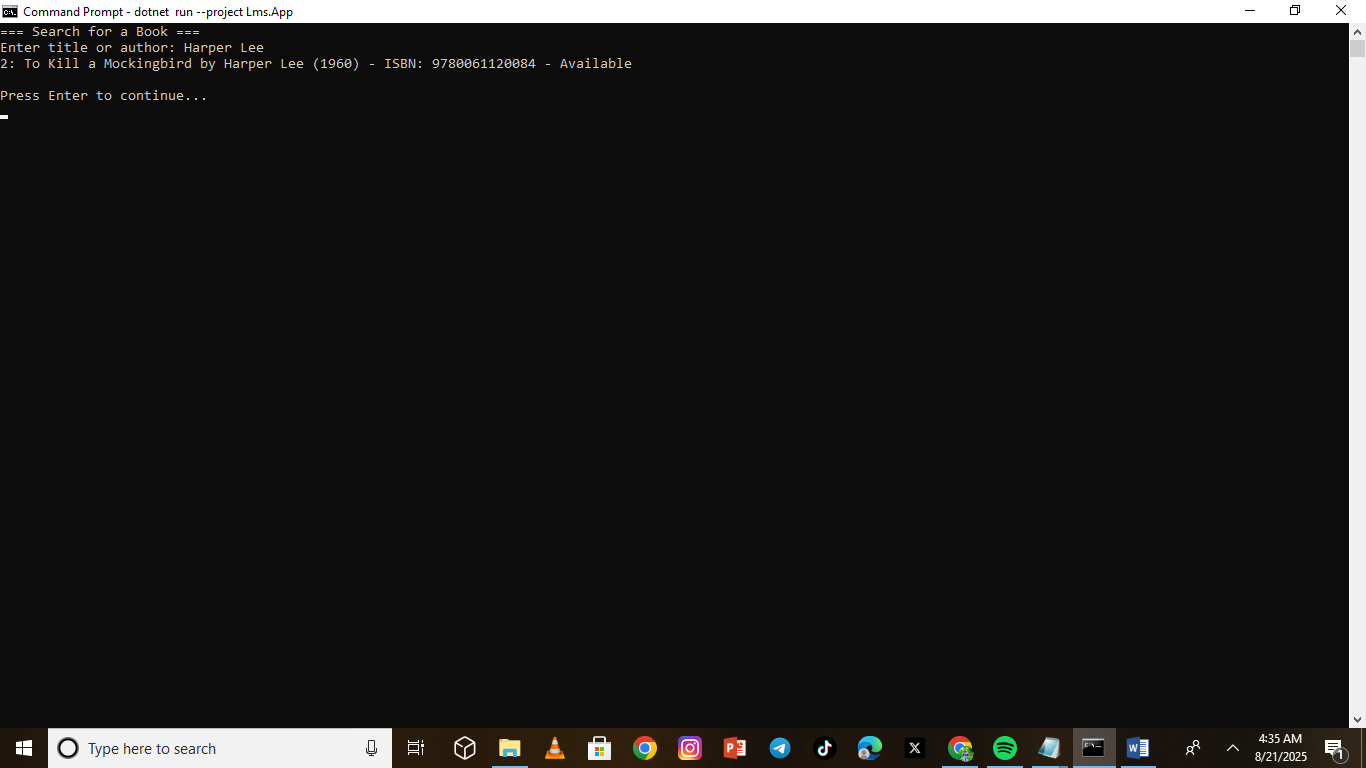
Adding books.



## Viewing all books available



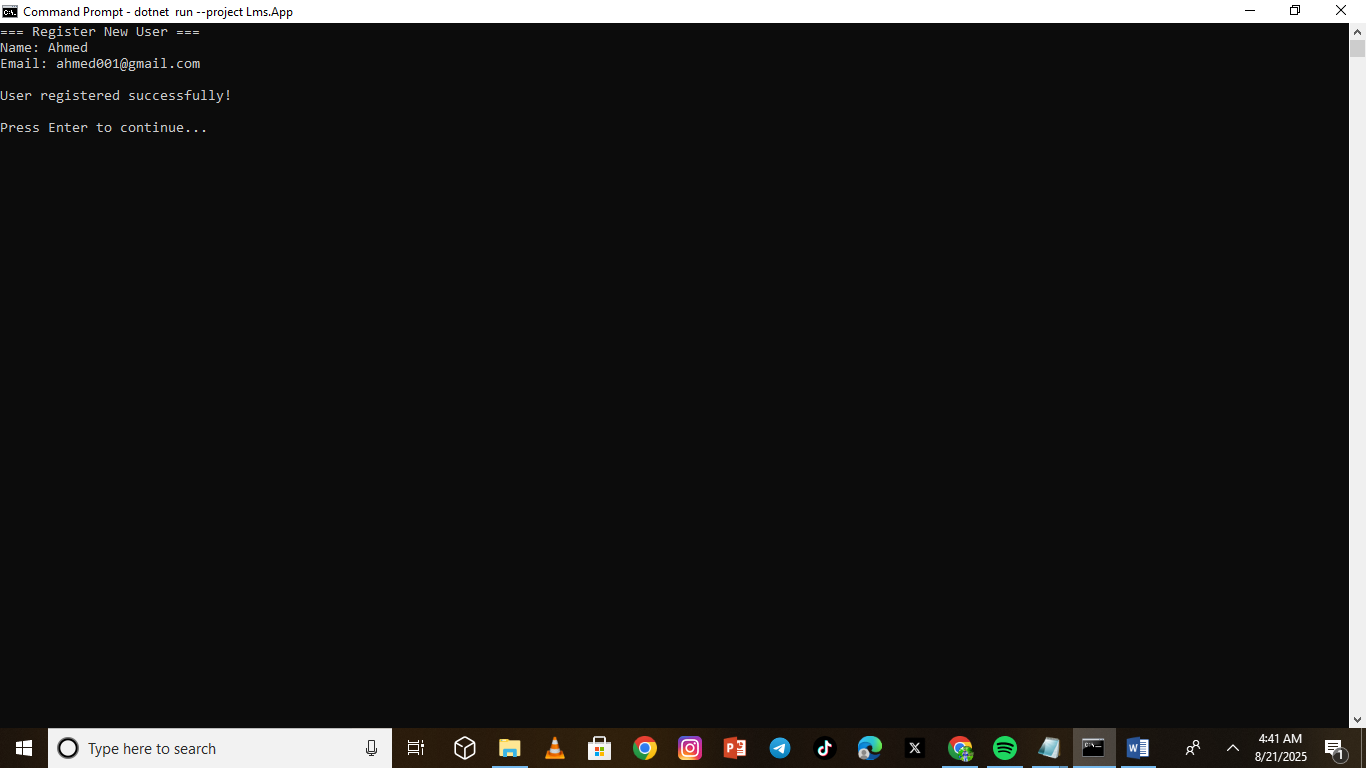
## Searching for a book by title/author



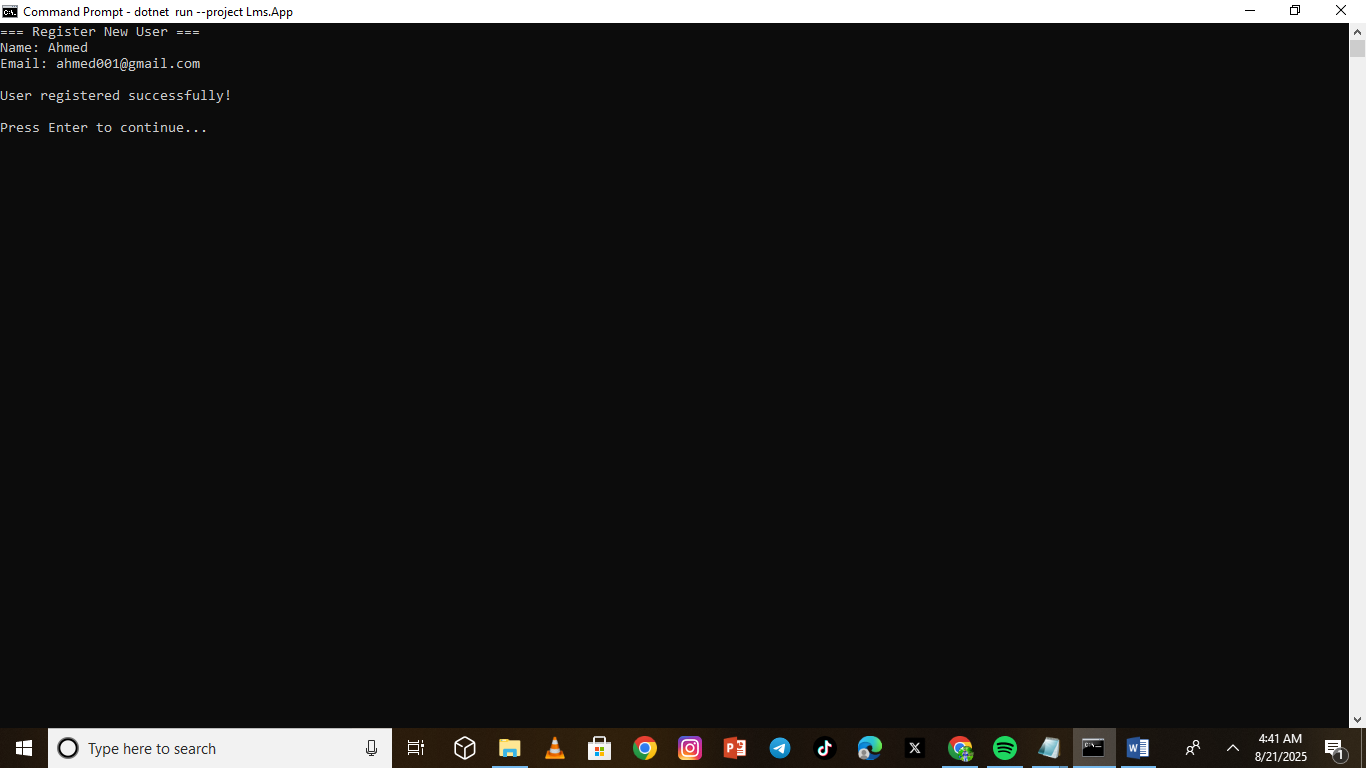
## 4. Results & Testing

After implementation, the system was tested for all core functionalities:  
- Adding a book successfully inserted data into both the in-memory list and the database.  
- Viewing all books displayed the correct records.  
- Searching for a book by title or author returned accurate results.  
- Registering a user ensured that the user was stored properly with unique email validation.  
- Borrowing a book correctly marked it as unavailable, while returning it restored availability.  
  
The tests also ensured that constraints such as unique ISBN and user email were properly enforced by the database. Unit tests written with xUnit validated the correctness of core logic such as availability checks and loan return status. The system allowed users to register themselves so that they can borrow and return a book using unique Id’s. if a book was already borrowed by a specific user and not already returned if you searched for the same book the system would show you that the book was not available. This meat the book would only be available once retuned.

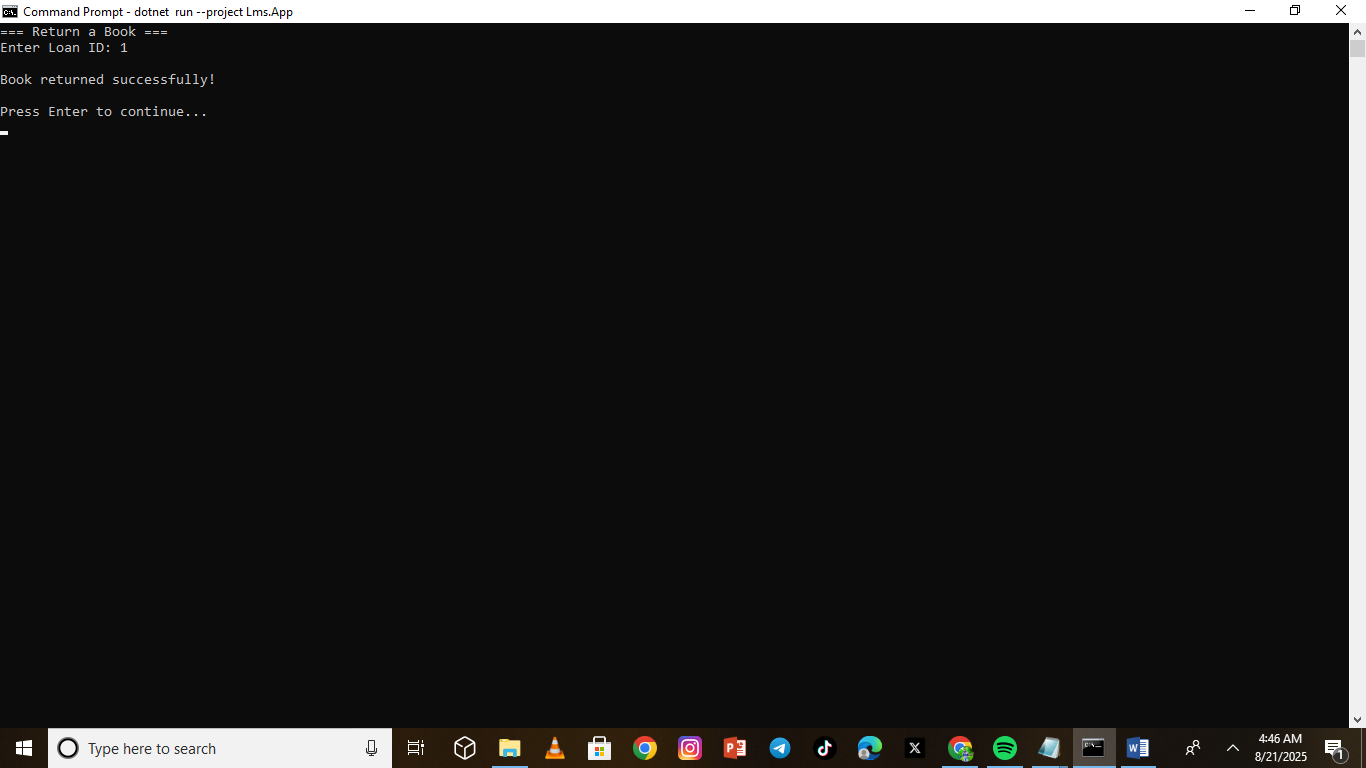
**User registration**



## Borrowing a book



**Returning a book**



## 5. Discussion

The introduction of a database-backed system demonstrated significant advantages over traditional paper-based or in-memory list approaches. SQL ensured that the system could scale and persist data even after restarting the application. The use of schema design provided clarity on relationships between books, users, and loans.  
  
Furthermore, Git-based version control provided a professional workflow, ensuring all code versions and changes were properly tracked (Ghodke and Chavan 2024) . This practice reflects real-world software engineering standards.  
  
However, some limitations remain. For example, the current system is limited to a command-line interface (CLI). While functional, a graphical user interface (GUI) or a web-based interface would improve user experience. Additionally, the reporting and analytics features could be expanded to include overdue loan tracking or book popularity trends.

## 6. Conclusion

This project successfully developed and tested a Library Management System using C#, .NET, and SQL. It achieved the key objectives of allowing book management, user registration, and loan transactions. The use of schema-driven database design and modular architecture provided a strong foundation for scalability. Through this project, important skills in database integration, version control, and modular coding were gained.  
  
This system can be extended into a full desktop or web application. Integrating authentication, advanced reporting, and user notifications could provide further value to libraries. Nonetheless, the project demonstrates a complete, functional solution to the problem of transitioning from manual to digital library management.

**References**

Blischak, J.D., Davenport, E.R. and Wilson, G., 2016. A quick introduction to version control with Git and GitHub. *PLoS computational biology*, *12*(1), p.e1004668. <https://doi.org/10.1371/journal.pcbi.1004668>

Daphine, A., 2013. An automated library management system a case study of Kashenyi vocational senior secondary school. <http://hdl.handle.net/20.500.12306/13387>

Gaffney, K.P., Prammer, M., Brasfield, L., Hipp, D.R., Kennedy, D. and Patel, J.M., 2022. SQLite: past, present, and future. *Proceedings of the VLDB Endowment*, *15*(12). <https://doi.org/10.14778/3554821.3554842>

Ghodke, G.M. and Chavan, T., 2024. An Overview of Git. *International Journal of Scientific Research in Modern Science and Technology*, *3*(6), pp.17-23. <https://doi.org/10.59828/ijsrmst.v3i6.216>