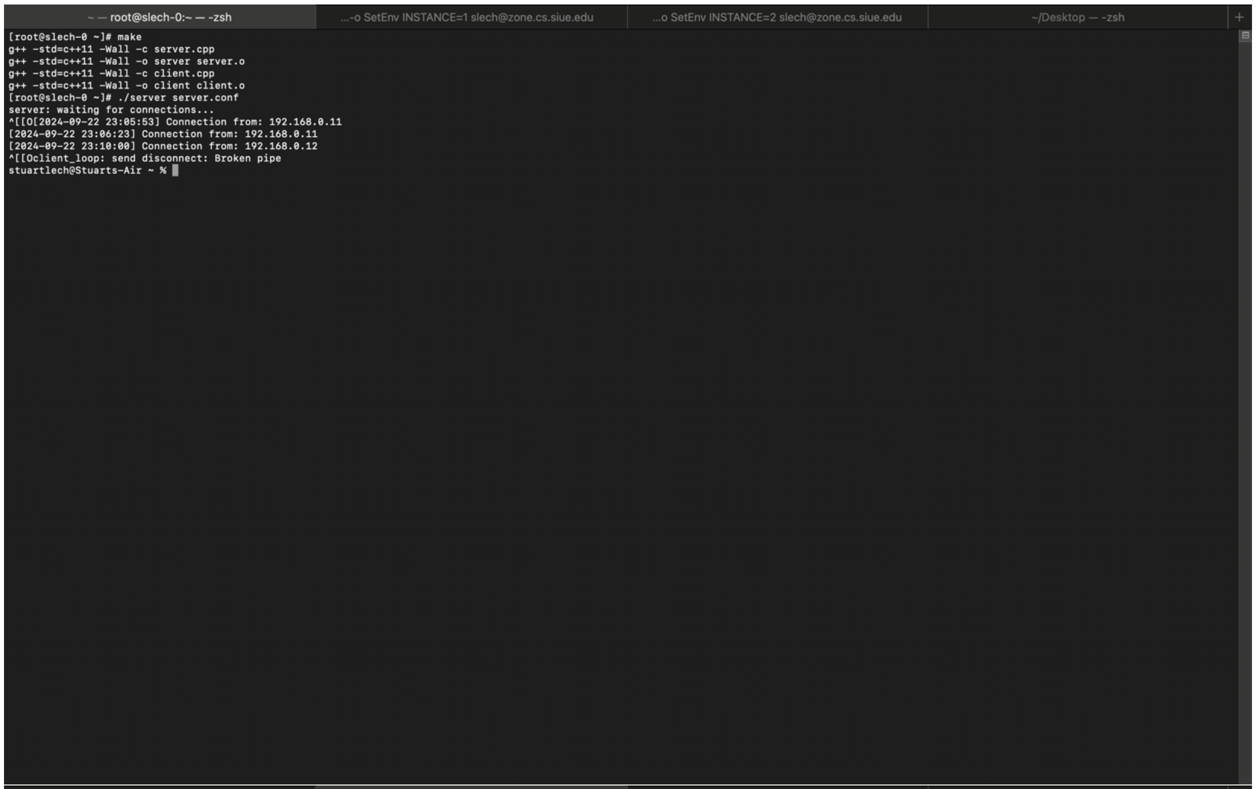
Introduction:
The goal of this project is to put in place a client-server library management system that runs over TCP and enables several clients to search for books, control checkouts, and get suggestions. Gaining hands-on experience with Linux system calls, socket programming, and concurrency management for many client connections are the main goals. My goal in finishing this assignment is to improve my knowledge of client-server communication, protocol accuracy, and TCP network protocols. The project will also give experience managing real time data transmission and guaranteeing appropriate concurrency in networked systems.
Design:

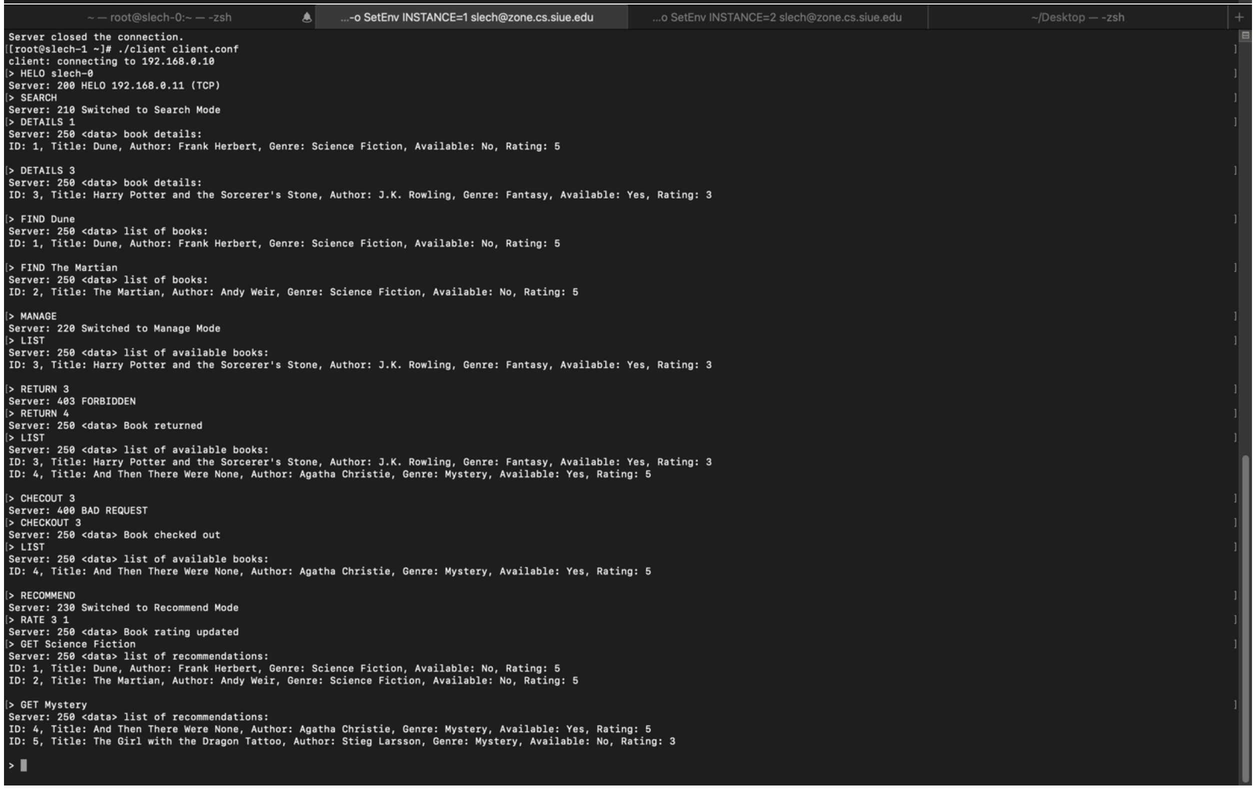
A shared book database can be accessed concurrently by several clients via the TCP-based client-server Library Book Management System. The fork() system call, which starts a new process for each client connection, allows the server to manage numerous clients. This guarantees that clients can operate independently and don't obstruct other users. All of the book data is contained in a books.db file that is delivered to the system. The server stores this data into memory upon connection, allowing for quick updates and searches. To carry out operations like discovering books, checking them out, or getting recommendations, each client must first establish a connection with the HELO command. Afterward, they can choose from various modes, including SEARCH, MANAGE, and RECOMMEND.

A command-based protocol governs client-server interactions, with the server employing distinct reply codes to answer to each client command. For instance, the server replies with 200 HELO (TCP) to validate the connection following a successful HELO command. Clients using FIND in SEARCH mode can look up books by author or title; if no matches are discovered, the server will return a 304 NO CONTENT response. Customers can check out or return books in MANAGE mode. If the book is unavailable, they will receive a response with the code 403 FORBIDDEN, otherwise 250 Book checked out for a successful transaction. Additionally, the system manages ratings and recommendations in RECOMMEND mode, which allows users to score or propose books based on category. To maintain consistency between sessions, every action is recorded, and the books.db file is updated following modifications such as checkouts or returns.

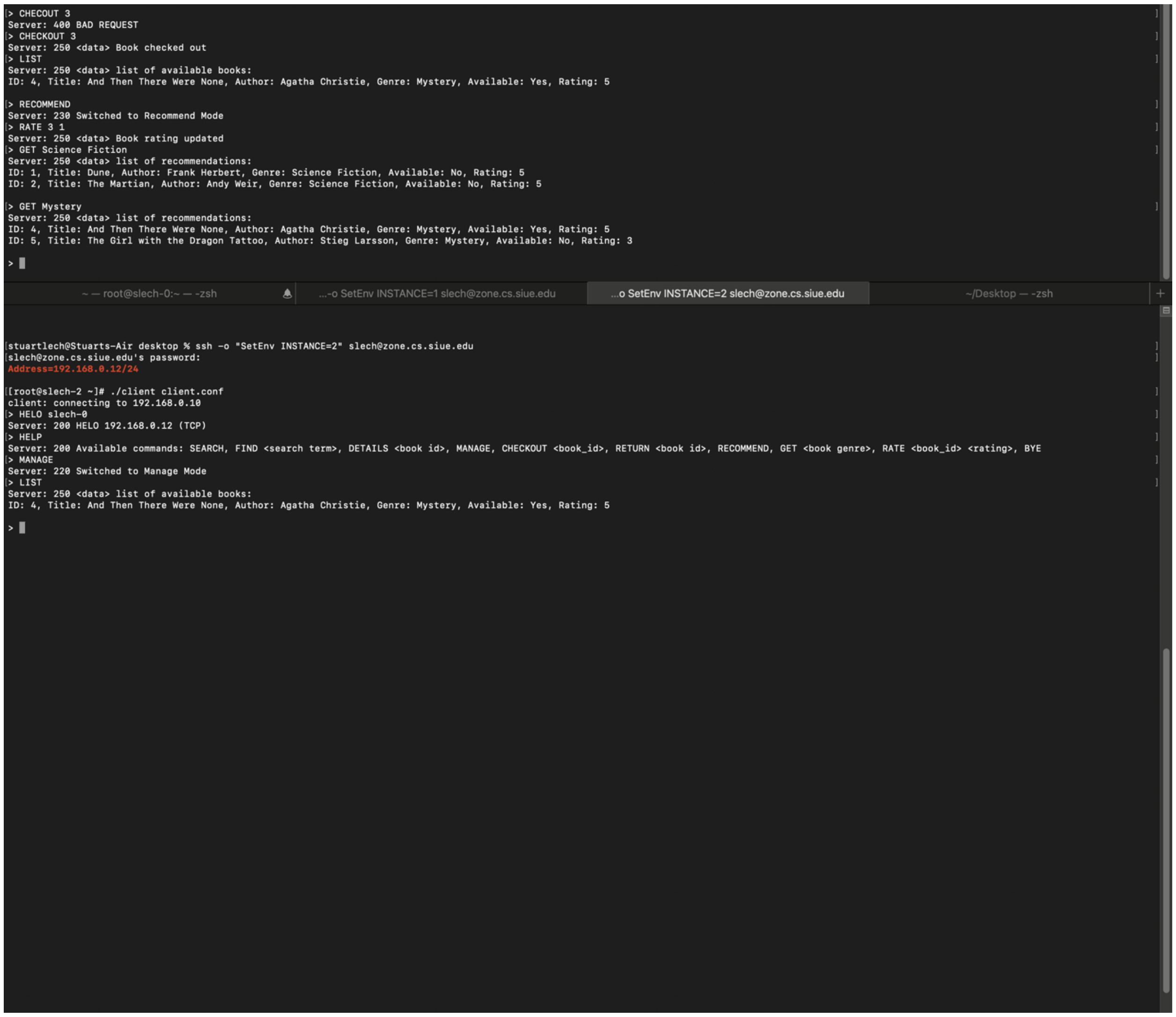
The use of fork() to manage concurrency and loading the book data from books.db into RAM for fast access are two important design choices. Updating the file synchronously guarantees that modifications will remain effective for upcoming client encounters. With its concise commands and server reply codes, the protocol is easy to use and efficient, giving clients quick feedback on their operations. Commands like BYE, for example, gracefully terminate the connection; the server responds with 200 BYE before the session ends. The architecture strikes a good mix between scalability, efficiency, and user-friendliness, which makes it ideal for managing several clients in a real-time, networked setting.
Sample Run:


~ — root@slech-0:~ — -zsh
...-o SetEnv INSTANCE=1 slech@zone.cs.siue.edu
...o SetEnv INSTANCE=2 slech@zone.cs.siue.edu
~/Desktop — -zsh

```
[root@slech-0 ~]# make
g++ -std=c++11 -Wall -c server.cpp
g++ -std=c++11 -Wall -o server server.o
g++ -std=c++11 -Wall -c client.cpp
g++ -std=c++11 -Wall -o client client.o
[root@slech-0 ~]# ./server server.conf
server: waiting for connections...
^[[O[2024-09-22 23:05:53] Connection from: 192.168.0.11
[2024-09-22 23:06:23] Connection from: 192.168.0.11
[2024-09-22 23:10:00] Connection from: 192.168.0.12
^[[Oclient_loop: send disconnect: Broken pipe
stuartlech@Stuarts-Air ~ %
```


~ — root@slech-0:~ — -zsh
...-o SetEnv INSTANCE=1 slech@zone.cs.siue.edu
...o SetEnv INSTANCE=2 slech@zone.cs.siue.edu
~/Desktop — -zsh

```
Server closed the connection.
[[root@slech-1 ~]# ./client client.conf
client: connecting to 192.168.0.10
[> HELO slech-0
Server: 200 HELO 192.168.0.11 (TCP)
[> SEARCH
Server: 210 Switched to Search Mode
[> DETAILS 1
Server: 250 <data> book details:
ID: 1, Title: Dune, Author: Frank Herbert, Genre: Science Fiction, Available: No, Rating: 5

[> DETAILS 3
Server: 250 <data> book details:
ID: 3, Title: Harry Potter and the Sorcerer's Stone, Author: J.K. Rowling, Genre: Fantasy, Available: Yes, Rating: 3

[> FIND Dune
Server: 250 <data> list of books:
ID: 1, Title: Dune, Author: Frank Herbert, Genre: Science Fiction, Available: No, Rating: 5

[> FIND The Martian
Server: 250 <data> list of books:
ID: 2, Title: The Martian, Author: Andy Weir, Genre: Science Fiction, Available: No, Rating: 5

[> MANAGE
Server: 220 Switched to Manage Mode
[> LIST
Server: 250 <data> list of available books:
ID: 3, Title: Harry Potter and the Sorcerer's Stone, Author: J.K. Rowling, Genre: Fantasy, Available: Yes, Rating: 3

[> RETURN 3
Server: 403 FORBIDDEN
[> RETURN 4
Server: 250 <data> Book returned
[> LIST
Server: 250 <data> list of available books:
ID: 3, Title: Harry Potter and the Sorcerer's Stone, Author: J.K. Rowling, Genre: Fantasy, Available: Yes, Rating: 3
ID: 4, Title: And Then There Were None, Author: Agatha Christie, Genre: Mystery, Available: Yes, Rating: 5

[> CHECOUT 3
Server: 400 BAD REQUEST
[> CHECKOUT 3
Server: 250 <data> Book checked out
[> LIST
Server: 250 <data> list of available books:
ID: 4, Title: And Then There Were None, Author: Agatha Christie, Genre: Mystery, Available: Yes, Rating: 5

[> RECOMMEND
Server: 230 Switched to Recommend Mode
[> RATE 3 1
Server: 250 <data> Book rating updated
[> GET Science Fiction
Server: 250 <data> list of recommendations:
ID: 1, Title: Dune, Author: Frank Herbert, Genre: Science Fiction, Available: No, Rating: 5
ID: 2, Title: The Martian, Author: Andy Weir, Genre: Science Fiction, Available: No, Rating: 5

[> GET Mystery
Server: 250 <data> list of recommendations:
ID: 4, Title: And Then There Were None, Author: Agatha Christie, Genre: Mystery, Available: Yes, Rating: 5
ID: 5, Title: The Girl with the Dragon Tattoo, Author: Stieg Larsson, Genre: Mystery, Available: No, Rating: 3

>
```



```
> CHECOUT 3
Server: 400 BAD REQUEST
> CHECKOUT 3
Server: 250 <data> Book checked out
> LIST
Server: 250 <data> list of available books:
ID: 4, Title: And Then There Were None, Author: Agatha Christie, Genre: Mystery, Available: Yes, Rating: 5

> RECOMMEND
Server: 230 Switched to Recommend Mode
> RATE 3 1
Server: 250 <data> Book rating updated
> GET Science Fiction
Server: 250 <data> list of recommendations:
ID: 1, Title: Dune, Author: Frank Herbert, Genre: Science Fiction, Available: No, Rating: 5
ID: 2, Title: The Martian, Author: Andy Weir, Genre: Science Fiction, Available: No, Rating: 5

> GET Mystery
Server: 250 <data> list of recommendations:
ID: 4, Title: And Then There Were None, Author: Agatha Christie, Genre: Mystery, Available: Yes, Rating: 5
ID: 5, Title: The Girl with the Dragon Tattoo, Author: Stieg Larsson, Genre: Mystery, Available: No, Rating: 3

> 
```

~ — root@slech-0:~ — -zsh
...-o SetEnv INSTANCE=1 slech@zone.cs.siue.edu
...o SetEnv INSTANCE=2 slech@zone.cs.siue.edu
~/Desktop — -zsh

```
stuartlech@Stuarts-Air desktop % ssh -o "SetEnv INSTANCE=2" slech@zone.cs.siue.edu
slech@zone.cs.siue.edu's password:
Address=192.168.0.12/24

[root@slech-2 ~]# ./client client.conf
client: connecting to 192.168.0.10
> HELO slech-0
Server: 200 HELO 192.168.0.12 (TCP)
> HELP
Server: 200 Available commands: SEARCH, FIND <search term>, DETAILS <book id>, MANAGE, CHECKOUT <book_id>, RETURN <book id>, RECOMMEND, GET <book genre>, RATE <book_id> <rating>, BYE
> MANAGE
Server: 220 Switched to Manage Mode
> LIST
Server: 250 <data> list of available books:
ID: 4, Title: And Then There Were None, Author: Agatha Christie, Genre: Mystery, Available: Yes, Rating: 5

> 
```

Summary:

The Library Book Management System is successfully implemented which allows multiple clients to search, manage, and recommend books concurrently. The system handles multiple connections efficiently by using the fork() system call, ensuring that each client session operates independently. Key functionalities like book searching, checkouts, returns, and recommendations work as intended, with clear protocol commands and server replies providing real-time feedback to the client. The integration of the provided books.db file ensures that the book data is managed consistently, with updates synchronized between memory and the database file after each modification. The only issues to raise are is the Library Book Management Systerm could include more user-friendly error messages or an expanded client-side interface for better usability. The system's design emphasizes concurrency, responsiveness, and ease of interaction, fulfilling the project's core objectives of practicing socket programming, concurrency management, and protocol design.