Documentation - ReadsProfiler (A)

Beschieru Dragos Marius, 2A5

January 15, 2024

1 Introduction

This paper has the primary goal of breaking down the structure of the ReadsProfiler project and highlighting important aspects of it, both in terms of implementation and organization.

ReadsProfiler is, in essence, an online library where each client can search for books using different criteria such as type, author, title, publication year, ISBN, and rating, and can download them. There will be multiple genres/subgenres, and each book will be in one or more genres. Each author will specify the genres their books belong to.

The server will store the searches and their results, as well as the downloads for each client. It will also have an algorithm that offers recommendations to clients based on multiple conditions like searches, ratings, authors, etc., scaling with clients' activity.

On the client side, each client will have a unique account with a username and password, ensuring that their data will not be lost in different sessions. Multiple instances of clients can exist simultaneously without interfering with each other.

Within the ReadsProfiler system, each client enjoys a personalized experience. The system is structured to allow multiple client instances simultaneous access, enabling users to engage independently in their activity without external interference. This approach mimics the experience of an online library, granting users the freedom to navigate and interact with the platform easily.

2 Technology

The project strategically incorporates a variety of technologies to ensure efficient implementation and seamless integration of features. These technologies are chosen to enhance system reliability, scalability, and overall performance.

2.1 TCP

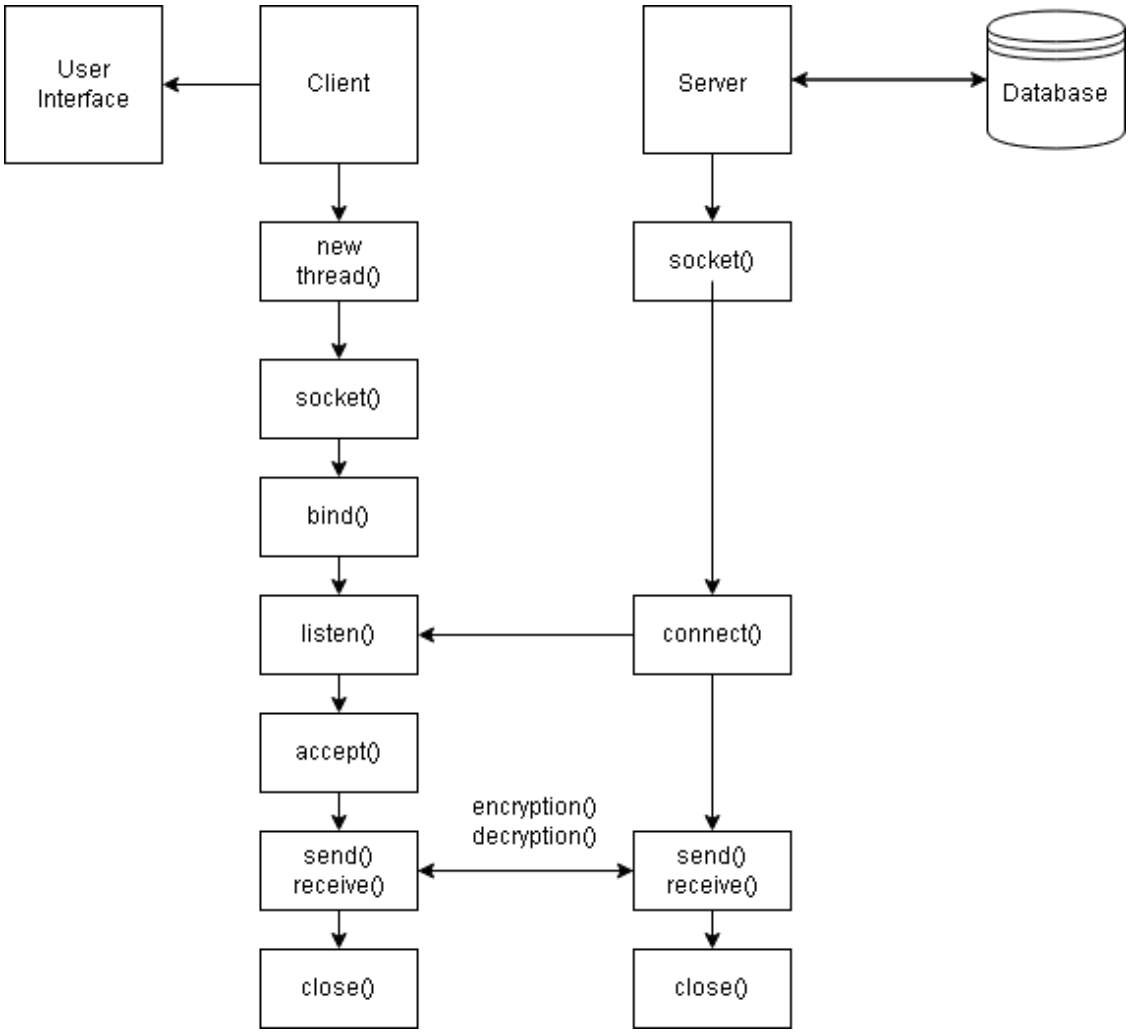
Transmission Control Protocol (TCP) plays a crucial role as a communication standard, facilitating the exchange of messages between application programs and computing devices over the network. In our project, TCP is fundamental for reliable data transmission, aligning with internet standards defined by the Internet Engineering Task Force (IETF). Its utilization ensures end-to-end data delivery, making it an ideal choice for maintaining data integrity during transmission between servers and clients.

2.2 DataBase

Using JSON files for data storage offers simplicity and flexibility, particularly beneficial for small-scale or lightweight applications. JSON's human-readable format simplifies the process of reading and writing

data, without the need for complex queries typical of traditional databases. This approach eliminates the overhead associated with database management systems, making it ideal for projects where such complexity and resource utilization are unnecessary. Moreover, JSON's language-agnostic nature ensures seamless data interchange across different programming environments, making it particularly suitable for applications with straightforward data storage needs or for storing configuration and temporary data.

3 Structure



3.1 Communication

- Facilitates reliable data transmission between servers and clients, ensuring smooth communication by using TCP.

3.2 User Management

- Ensures each user has a unique account with a username and password for secure access.

3.3 Search and Recommendation Engine

- Utilizes a sophisticated recommendation algorithm that analyzes user search history and activities to generate tailored book recommendations.

3.4 Database

- Utilizes the nlohmann JSON library to efficiently manage data storage and retrieval in a JSON format, providing a straightforward and flexible approach for handling user-related information. This method leverages the simplicity and accessibility of JSON files, ensuring an easy-to-use and maintain data management system.

4 Implementation

The code implementation covers a lot of functions, which I broke down into different categories.

4.1 TCP Connection

- `establish_tcp_connection()` - Initiates a TCP connection for communication.
- `start_listening()` - Starts listening for incoming client connections.
- `send_message(const std::string\`
  `&message)` - Sends a message to a connected client.
- `receive_message()` - Receives a message from a connected client.
- `handle_client()` - Handles communication with a connected client.

Dynamic binding is utilized in the implementation of these functions to allow for flexibility and extensibility in handling different types of TCP connections and communication scenarios. This approach enables the TCP connection class to be used as a base class, with derived classes overriding these methods to provide specific functionality for different use cases.

- `establish_tcp_connection()` - Initiates a TCP connection for communication.

4.2 User Management

- `login(const std::string &credentials)` - Authenticates a user to log in to an existing account.
- `newAccount(const std::string &credentials)` - Registers a new user account.
- `login_success(const std::string &username, const std::string &password)` - Handles successful login logic.
- `logout()` - Logs out the current user.
- `isLogged()` - Checks if a user is currently logged in.
- `getUsername()` - Retrieves the username of the current user.
- `getPassword()` - Retrieves the password of the current user.

4.3 Account Operations

- `verify_account(const std::string &filePath, const std::string &username, const std::stri &password)` - Verifies if an account exists.
- `new_account(const std::string &filePath, const std::string &username, const std::string &password)` - Creates a new account.
- `delete_account(const std::string &filePath, const std::string &username)` - Deletes an existing account.

4.4 Search, Recommendation, and Download

- `search_genre(const std::string &value)` - Searches for books by genre.
- `search_title(const std::string &value)` - Searches for books by title.
- `search_year(const std::string &value)` - Searches for books by year of publication.
- `search_rating(const std::string &value)` - Searches for books by user ratings.
- `search_author(const std::string &value)` - Searches for books by author.
- `search_ISBN(const std::string &value)` - Searches for books by ISBN.
- `rate_book(const double &rating, const int &bookId)` - Rates a book.
- `getDownloadLink(const int &bookId)` - Retrieves the download link for a book by its ID.
- `generateRecs()` - Generates personalized book recommendations.

4.5 Search History Management

- `updateSearchHistory(const std::string\ &filePath, const std::string\ &username, int bookID, double rating)` - Updates the search history for a user.
- `delete_history(const std::string\ &filePath, const std::string\ &username)` - Deletes the search history for a user.
- `delete_hstr()` - Deletes the current user's search history.

4.6 Data Storage and Retrieval

- `search_book(const std::string\ &filePath, RequestType criteria, const std::string\ &searchValue)` - Searches for books based on various criteria.
- `recommendBooks(const std::string\ &filePath, const std::string\ &currentUsername)` - Recommends books based on user preferences.

- `getDownloadLink(const std::string\`
  `&filePath, const int\`
  `&bookId)` - Retrieves the download link for a book.

4.7 Utilities

4.7.1 Server Utilities

- `getRequestType(const std::string& message)` - Determines the type of request from the message.
- `getInformation(const std::string& message)` - Extracts information from the message.
- `getAccount(const std::string& message)` - Parses account details from the message.
- `jsonToBook(const nlohmann::json& j)` - Converts JSON to a Book object.
- `booksToString(const std::vector<Book>& books)` - Converts a list of books to a string.
- `stringToBooks(const std::string& str)` - Converts a string to a list of books.
- `searchType(const std::string& message)` - Identifies the type of search from the message.
- `user_search(const std::string& message)` - Parses user search query from the message.
- `fuzzyMatch(const std::string& str1, const std::string& str2, int threshold)` - Performs a fuzzy match between two strings.
- `haveCommonGenres(const std::string& userGenres, const std::string& bookGenres)` - Checks if user genres and book genres have common elements.

4.7.2 Client Utilities

- `stringToBool(const std::string& myString)` - Converts a string to a boolean.
- `Interface_NotAuth()` - Interface for non-authenticated users.
- `getCredentials()` - Retrieves user credentials.
- `Interface_Auth()` - Interface for authenticated users.
- `showBooks(const std::vector<Book>& books)` - Displays a list of books.
- `showBook(const Book& book)` - Displays details of a single book.
- `nextexitrate()` - Handles the next, exit, and rate options.
- `exchange(const std::string& msg)` - Exchanges messages between client and server.

5 Conclusion

The implementation of ReadsProfiler incorporates a robust set of technologies, ensuring efficient communication, secure user management, and smooth database integration. The combination of TCP for network communication and a structured approach to handling user interactions, data security, and recommendation algorithms contributes to a reliable and user-friendly system. This design allows for effective handling of essential aspects such as user authentication, search functionality, and personalized book recommendations. The following ideeas can be further implemented to improve the quality of the project :

- Enhanced Book Search with Multiple Criteria
- Book Rating System (x)
- User Reviews
- User Activity Tracking (x)

6 Bibliography

- `https://www.fortinet.com/resources/cyberglossary/tcp-ip`
- `https://github.com/nlohmann/jsonl`