Book Recommendation System

Presented by: Group 1

Introduction

The Book Recommendation System is a command-line tool (CLI) developed in C++ designed to help users discover new books based on their interests. The system allows users to input preferences such as genre or author and receive recommendations accordingly. With a simple CLI interface, the Book Recommendation System provides a seamless experience for users to explore a wide range of books tailored to their tastes.

Objectives

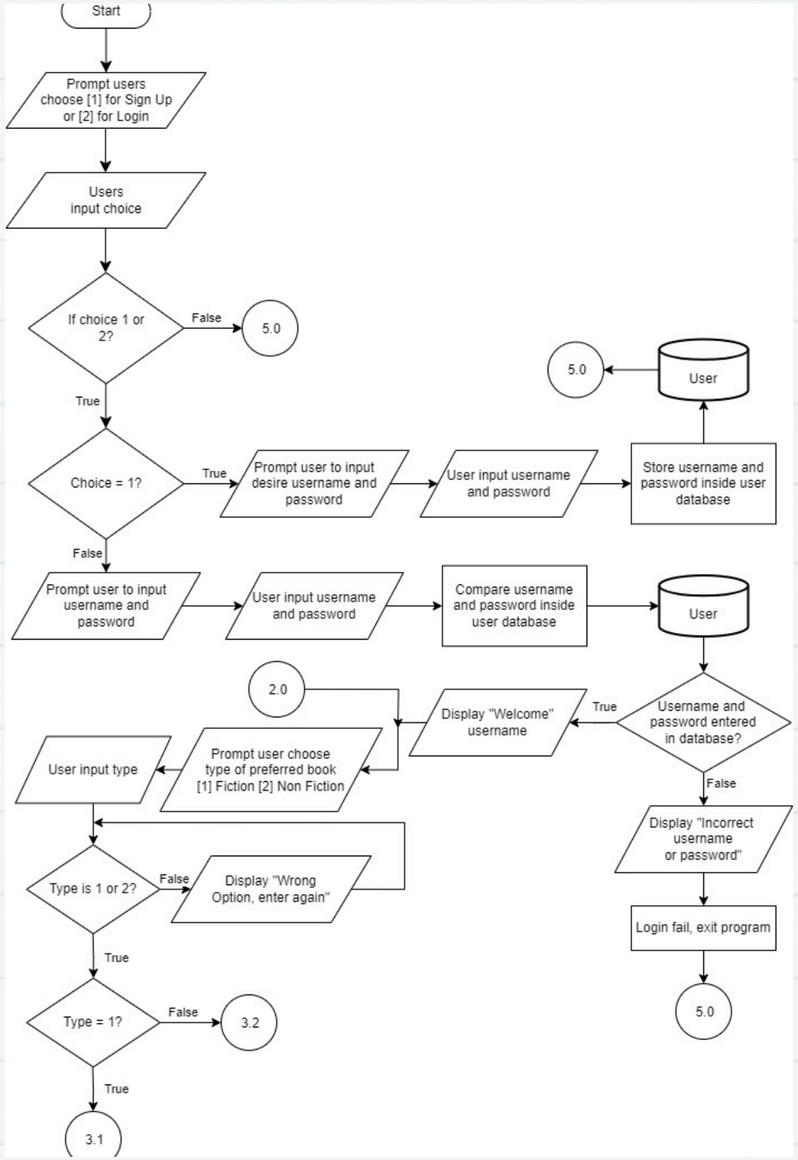
Assist users in discovering new books that match their preferences.

Provide personalized book recommendations tailored to each user's specific criteria.

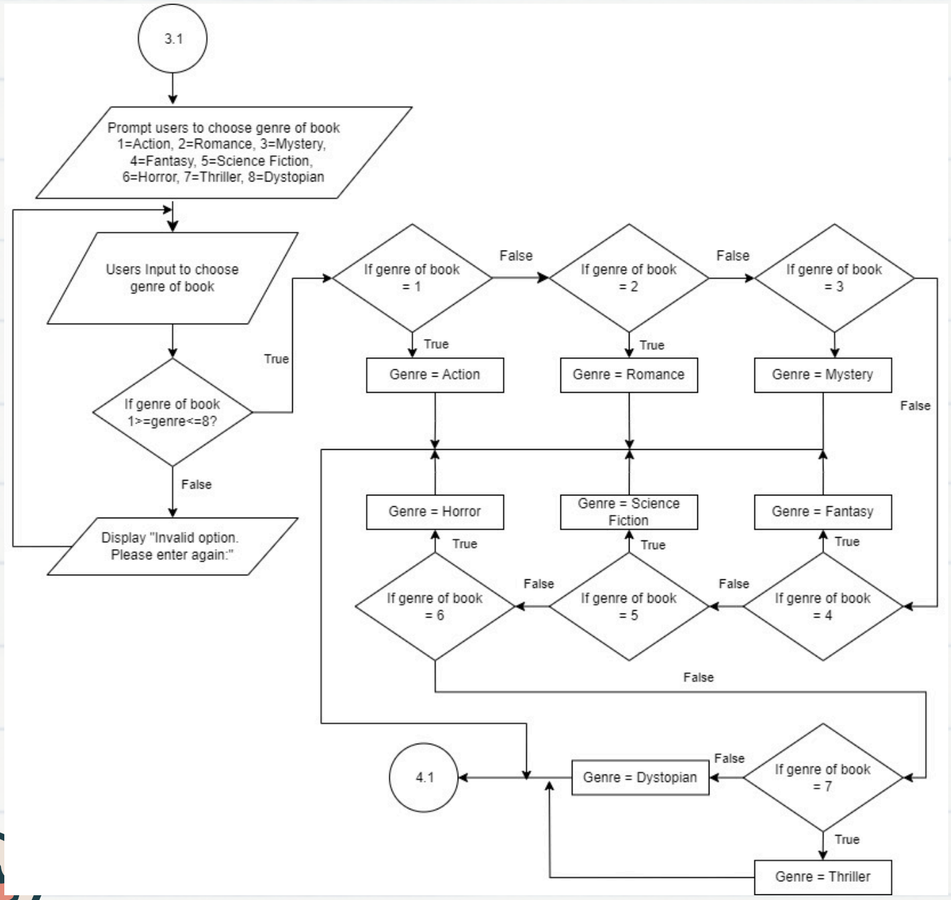
Enhance user engagement with books by suggesting titles they may enjoy

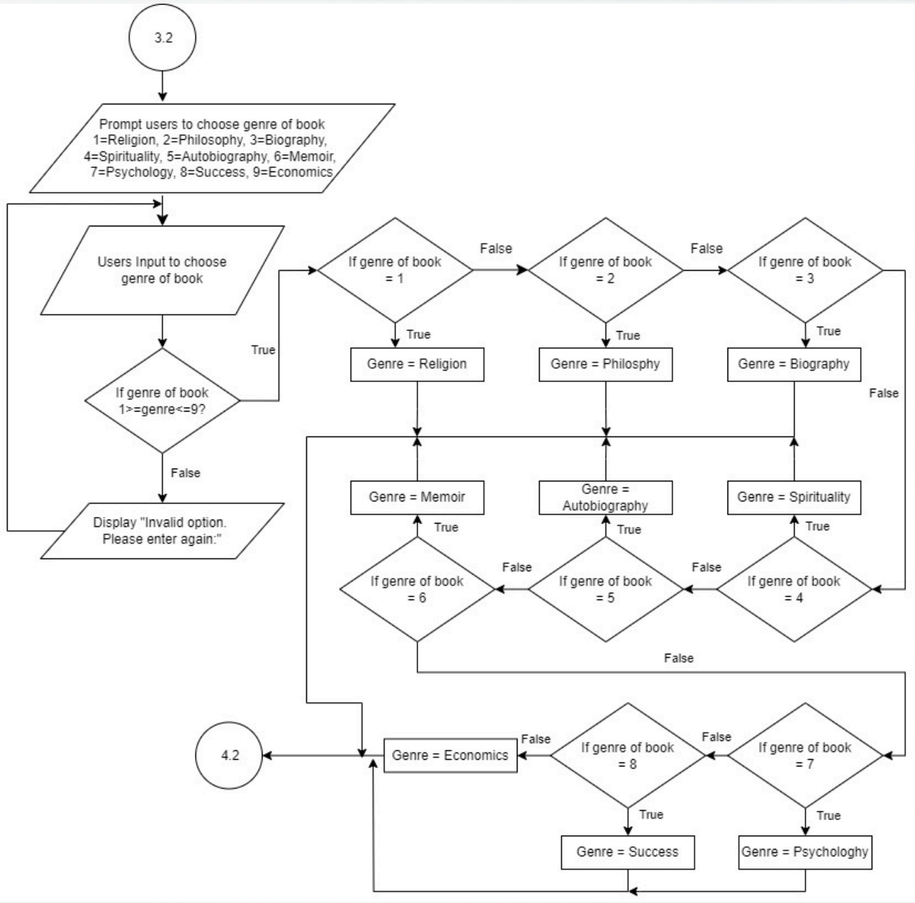
Foster a love for reading by introducing users to new genres and authors they may not have explored before

Flowchart

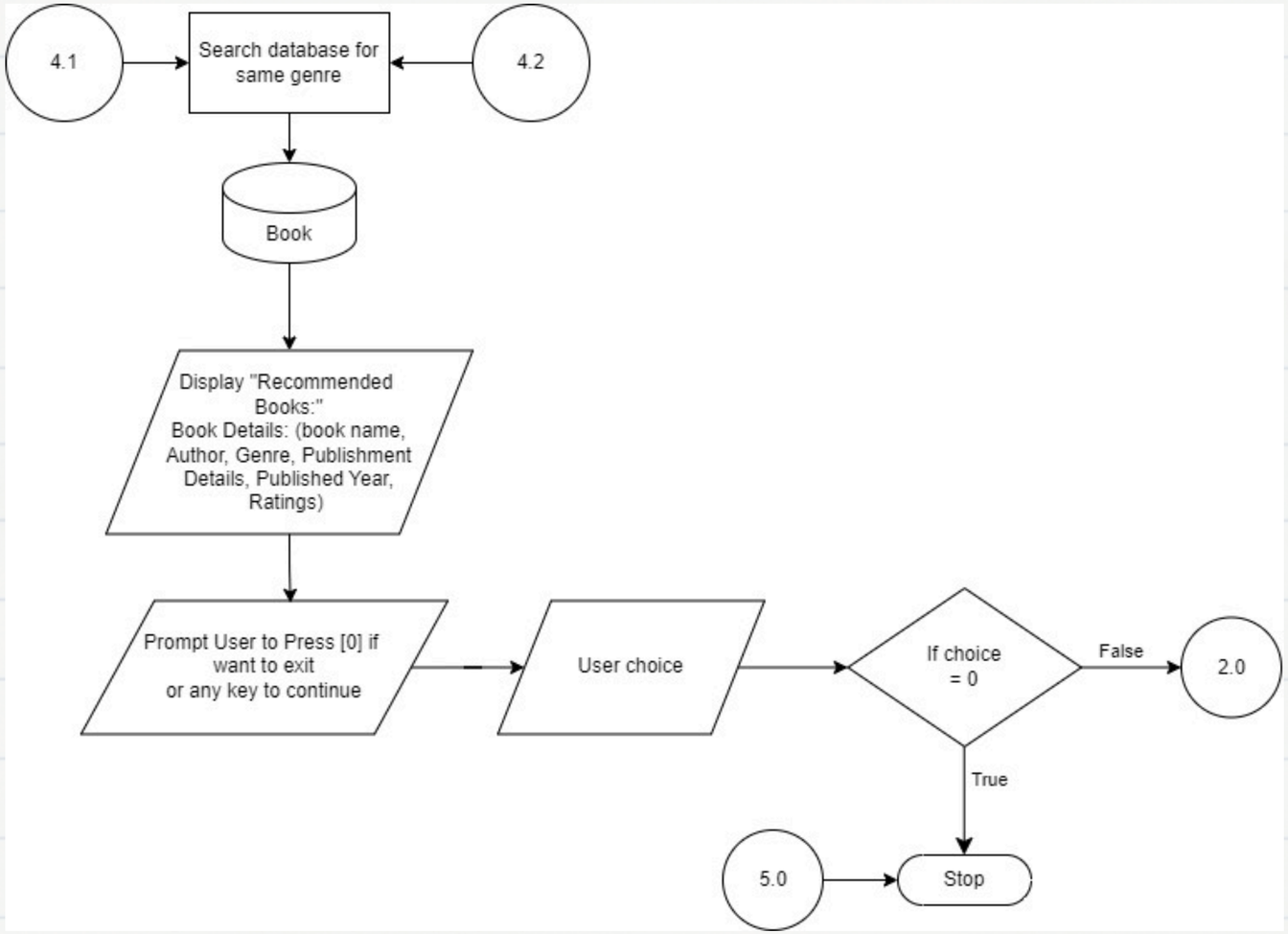

Start
Prompt users choose [1] for Sign Up or [2] for Login
Users input choice
If choice 1 or 2?
False
5.0
True
Choice = 1?
True
Prompt user to input desire username and password
User input username and password
Store username and password inside user database
User
5.0
False
Prompt user to input username and password
User input username and password
Compare username and password inside user database
User
Username and password entered in database?
True
Display "Welcome" username
2.0
Prompt user choose type of preferred book [1] Fiction [2] Non Fiction
User input type
False
Display "Incorrect username or password"
Login fail, exit program
5.0
Type is 1 or 2?
False
Display "Wrong Option, enter again"
True
Type = 1?
False
3.2
True
3.1

Flowchart

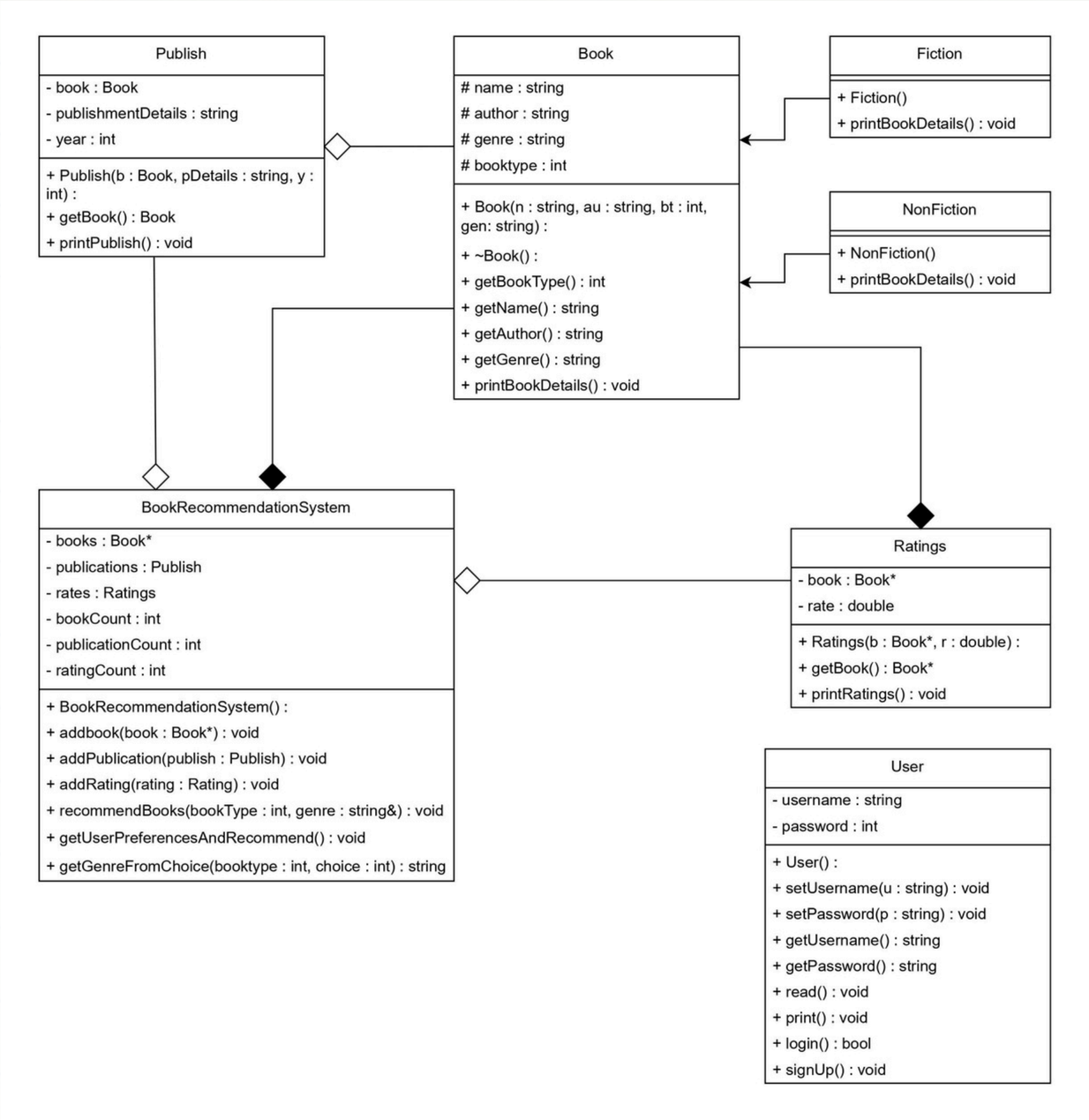

3.1
Prompt users to choose genre of book
1=Action, 2=Romance, 3=Mystery,
4=Fantasy, 5=Science Fiction,
6=Horror, 7=Thriller, 8=Dystopian
Users Input to choose genre of book
If genre of book 1>=genre<=8?
True
False
Display "Invalid option. Please enter again:"
If genre of book = 1
If genre of book = 2
If genre of book = 3
Genre = Action
Genre = Romance
Genre = Mystery
Genre = Horror
Genre = Science Fiction
Genre = Fantasy
If genre of book = 6
If genre of book = 5
If genre of book = 4
If genre of book = 7
Genre = Dystopian
Genre = Thriller
4.1


3.2
Prompt users to choose genre of book
1=Religion, 2=Philosophy, 3=Biography,
4=Spirituality, 5=Autobiography, 6=Memoir,
7=Psychology, 8=Success, 9=Economics
Users Input to choose genre of book
If genre of book 1>=genre<=9?
True
False
Display "Invalid option. Please enter again:"
If genre of book = 1
If genre of book = 2
If genre of book = 3
Genre = Religion
Genre = Philosphy
Genre = Biography
Genre = Memoir
Genre = Autobiography
Genre = Spirituality
If genre of book = 6
If genre of book = 5
If genre of book = 4
If genre of book = 8
If genre of book = 7
Genre = Economics
Genre = Success
Genre = Psychologhy
4.2

Flowchart


4.1
Search database for same genre
4.2
Book
Display "Recommended Books:"
Book Details: (book name, Author, Genre, Publishment Details, Published Year, Ratings)
Prompt User to Press [0] if want to exit
or any key to continue
User choice
If choice = 0
False
2.0
True
5.0
Stop

UML Class Diagram



Concept Implementation

User

```
bool login() {
    ifstream userFile("users.txt");
    if (!userFile) {
        cerr << "Unable to open user file 'users.txt'." << endl;
        return false;
    }

    string storedUsername;
    string storedPassword;
    bool loginSuccessful = false;

    while (userFile >> storedUsername >> storedPassword) {
        if (username == storedUsername && password == storedPassword) {
            loginSuccessful = true;
            break;
        }
    }

    userFile.close();
```

File Handling

Book

```cpp
#ifndef BOOK_H
#define BOOK_H
#include <iostream>
using namespace std;

class Book { //Parent class
protected:
    string name, author, genre;
    int booktype; // 1 for Fiction, 2 for Non-Fiction
public:
    Book() : name(""), author(""), genre(""), booktype(0) {} //Default constructor
    Book(string n, string au, int bt, string gen) : name(n), author(au), booktype(bt), genre(gen) {}
```

Encapsulation

Fiction & Non-Fiction

```cpp
#ifndef FICTION_H
#define FICTION_H
#include <iostream>
using namespace std;

class Fiction : public Book {
public:
    Fiction() {}
    Fiction(string n, string au, string gen) : Book(n, au, 1, gen) {}
```

```cpp
#ifndef NONFICTION_H
#define NONFICTION_H
#include <iostream>
using namespace std;

class NonFiction : public Book {
public:
    NonFiction() {}
    NonFiction(string n, string au, string gen) : Book(n, au, 2, gen) {}
```

Inheritance

Publish

```
#ifndef PUBLISH_H
#define PUBLISH_H
#include <iostream>
using namespace std;

class Publish {
private:
    Book book; // composition
    string publishmentDetails;
    int year;
public:
    Publish() : publishmentDetails(""), year(0) {}
    Publish(Book b, string pDetails, int y) : book(b), publishmentDetails(pDetails), year(y) {}

    Book getBook() const {
        return book;
    }
```

Composition

Ratings

```
 6   class Ratings {
 7   private:
 8       Book* book; // aggregation
 9       double rate;
10   public:
11       Ratings() : book(NULL), rate(0.0) {}
12       Ratings(Book* b, double r) : book(b), rate(r) {}
13
14       Book* getBook() const {
15           return book;
16       }
```

Aggregation

Book & Fiction & Non-Fiction

```
virtual void printBookDetails() const {
    cout << "Book Name: " << name << endl;
    cout << "Author: " << author << endl;
    cout << "Genre: " << genre << endl;
}

virtual ~Book() {}
```

```
void printBookDetails() const  {
    cout << "Fiction Book Details:" << endl;
    Book::printBookDetails();
}
```

```
void printBookDetails() const  {
    cout << "Non-Fiction Book Details:" << endl;
    Book::printBookDetails();
}
```

Polymorphism

BookRecommendationSystem

```
class BookRecommendationSystem {
private:
    Book* books[MAX_BOOKS];
    Publish publications[MAX_PUBLICATIONS];
    Ratings rates[MAX_RATINGS];
    int bookCount, publicationCount, ratingCount;

public:
    BookRecommendationSystem() : bookCount(0), publicationCount(0), ratingCount(0) {}

    void addBook(Book* book) {
        if (bookCount < MAX_BOOKS) {
            books[bookCount++] = book;
        }
    }

    void addPublication(Publish publish) {
        if (publicationCount < MAX_PUBLICATIONS) {
            publications[publicationCount++] = publish;
        }
    }

    void addRating(Ratings rating) {
        if (ratingCount < MAX_RATINGS) {
            rates[ratingCount++] = rating;
        }
    }
```

Array of Objects

Main

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include "User.h"
#include "Book.h"
#include "Fiction.h"
#include "NonFiction.h"
#include "Publish.h"
#include "Ratings.h"
#include "BookRecommendationSystem.h"
using namespace std;


int main() {
    BookRecommendationSystem bookSystem;

    // Adding Fiction Books
    bookSystem.addBook(new Fiction("The Great Gatsby", "F. Scott Fitzgerald", "Action"));
    bookSystem.addBook(new Fiction("1984", "George Orwell", "Mystery"));
    bookSystem.addBook(new Fiction("To Kill a Mockingbird", "Harper Lee", "Romance"));
    bookSystem.addBook(new Fiction("Pride and Prejudice", "Jane Austen", "Romance"));
    bookSystem.addBook(new Fiction("The Catcher in the Rye", "J.D. Salinger", "Action"));
    bookSystem.addBook(new Fiction("The Hobbit", "J.R.R. Tolkien", "Fantasy"));
```

Array of Objects

Thank You

Presented by:
Mervyn Bee Zheng Cheng
Avidian Dipesh Siva
Muhammad Aziidan Bin Muhd Azlan Ng