**Movie Ticket Booking System**

**TABLE OF CONTENTS**

Contents

[ Introduction 2](#_Toc2070626023)

[ App organization 3](#_Toc495313252)

[ About the Developer's Responsibilities 4](#_Toc1935021766)

[ Block scheme about the relationship between the files 5](#_Toc820263175)

# Introduction

|  |  |
| --- | --- |
| № | **Introduction** |
| 1 | **What is the product?**  The project is a console application written in C++ within the Visual Studio 2022 environment. The system allows users to register, search for movies, browse cinema programs, and book tickets by selecting specific seats. It also includes an administrative panel for content management. |
| 3 | **What technologies are used?**  The technologies used are **Visual Studio 2022** as the development environment, **C++** as the programming language, **JSON** for data storage, and **Git** and **GitHub** for collaborative work and version control. |

# App organization

|  |  |  |
| --- | --- | --- |
| **№** | **Component/Page** | **Description** |
| 1 | **App (app.cpp)** | The main class of the application, which manages the main loop and navigation between pages. |
| 2 | **PageHandler (pageHandler.cpp)** | A class that holds state flags and determines which page should currently be displayed. |
| 3 | **Dashboard (dashboard.cpp)** | The main menu after logging into the system. It provides access to all major functions. |
| 4 | **Register / Login** | Pages for user registration and login. |
| 5 | **pickCinema / pickMovie / pickShow** | A sequence of pages that guide the user through the process of selecting a cinema, movie, and projection. |
| 6 | **seatSelection (seatSelection.cpp)** | Displays a visual layout of the seats in the hall and allows the user to make their selection. |
| 7 | **Payment (payment.cpp)** | Simulates the final stage of payment by card or a cash reservation by a ticket agent. |
| 8 | **Search (search.cpp)** | A page that allows you to search for movies by title, genre, language, and date. |
| 9 | **Browse (browse.cpp)** | A "library" page that allows browsing the full program of a given cinema. |
| 10 | **MyBookings (myBookings.cpp)** | Displays the user's booking history and provides options for cancellation or payment confirmation. |
| 11 | **Admin (admin.cpp)** | A panel for administrators that allows for adding, deleting, and updating movies and projections. |
| 12 | **DAL (files.cpp)** | Data Access Layer. The only component that has direct access to the .json files for reading and writing. |

## **About the Developer's Responsibilities**

The developer's role in this project was to build the entire console application, covering all aspects from the user interface to data management.

1. **Data Design and Management**

* Structuring and creating JSON files (cities.json, accounts.json, bookings.json) to store all information.
* Defining a logical data structure for cinemas, movies, projections, seats, users, and bookings.
* Implementing functions in the **Data Access Layer (DAL)** for reading and writing to JSON files, ensuring data integrity.

1. **User Authentication and Security**

* Implementing functionality for registration and login.
* Developing methods for verifying user data against the records in accounts.json.
* Differentiating between regular users and administrators (ifAdmin.h) to grant access to specialized functions.

1. **Business Logic and Rules**

* Defining the logic for the entire booking process, including seat selection, price calculation, and changing seat status (**R5, R8, R9, R10, R11**).
* Implementing search algorithms based on multiple criteria (**R4**).
* Developing full CRUD (Create, Read, Update, Delete) functionality for administrators to manage movies and projections (**R12**).
* Implementing a system for cash and card reservations, as well as managing booking history (**R6, R7**).

1. **User Interface (Console)**

* Designing and building all menus and pages of the console application.
* Ensuring clear navigation and user feedback.
* Visualizing complex data such as the seating layout in the hall (**R13**).
* Implementing a user notification system (**R14**).

1. **Debugging and Testing**

* Conducting manual testing after the implementation of each new feature.
* Resolving logical and linker errors.
* Implementing robust data input (getIntegerInput) to prevent crashes from invalid user input.

# Block scheme about the relationship between the files

