**TICKET**

**NEXUS**

**OOP**  **PROJECT REPORT**

**Ticket Nexus:** CLI Based Ticket Booking system

**Made By: (Group#6)**

Muhammad Furqan Arshad (243583)

Muhammad Bin Nasir (243595)

**Submitted to:** Prof. Syed Burhanuddin

Contents

[**Introduction:** 3](#_Toc199204525)

[**Overview:** 3](#_Toc199204526)

[**Problem statement:** 4](#_Toc199204527)

[**Objectives:** 4](#_Toc199204528)

[**Scope (Features):** 4](#_Toc199204529)

[**Limitations:** 5](#_Toc199204530)

[**Significance:** 5](#_Toc199204531)

[Development Strategy: 6](#_Toc199204532)

[Programming Environment: 6](#_Toc199204533)

[Libraries/Dependencies: 6](#_Toc199204534)

[6. Classes Functionalities and Relationships: 7](#_Toc199204535)

[1.Transportation Domain 7](#_Toc199204536)

[A. Class Roles and Responsibilities 7](#_Toc199204537)

[B. Relationships 7](#_Toc199204538)

[2. Movie and Payment Domain 8](#_Toc199204539)

[A. Class Roles and Responsibilities 8](#_Toc199204540)

[B. Relationships 8](#_Toc199204541)

[3. User, Login, and Interface Utilities 8](#_Toc199204542)

[A. Class Roles and Responsibilities 8](#_Toc199204543)

[B. Relationships 9](#_Toc199204544)

[4. Validation, Menu, and Utility Classes 9](#_Toc199204545)

# **Introduction:**

## **Overview:**

**Ticket Nexus: A Command-Line Ticket Booking System** is a modular, object-oriented C++ application crafted to streamline transportation and entertainment ticketing processes. Designed for terminal-based environments, it facilitates seamless interactions between administrators and users through secure authentication, dynamic booking operations, and real-time seat management. The system ensures data consistency, validation integrity, and maintainability by organizing features into coherent modules—from login to ticket issuance—while supporting future extensibility into broader domains like airlines and events.

## **Problem statement:**

Manually managing bus and movie ticket reservations often results in booking conflicts, data redundancy, and limited accessibility. Without a centralized system, tracking seat availability, validating user inputs, and handling user roles becomes prone to errors and inefficiencies. To address these challenges, an automated, file-based solution is needed that supports secure role-based access, dynamic seat tracking, and seamless ticket management. This system ensures structured handling of booking operations, providing a reliable and scalable approach to multi-domain ticketing through a command-line interface.

## **Objectives:**

1. Develop a CLI-based ticket booking system with two distinct user roles: **Admin** and **User**.
2. Enable Admins to manage bus and movie data, including adding, editing, deleting, and searching records.
3. Allow Admins to monitor all bookings and view booking histories across users and services.
4. Provide Users with the ability to search, book, and cancel tickets for buses and movies.
5. Ensure robust authentication with role-based access using secure credential handling and validation checks.
6. Prevent double bookings by implementing dynamic seat maps and validation of availability during selection.
7. Maintain data persistence and consistency through structured, pipe-delimited text files and safe file update mechanisms.
8. Apply rigorous input validation to ensure accuracy in fields such as dates, times, phone numbers, and credentials.

## **Scope (Features):**

1. Role-based authentication for **Admin** and **User** with secure login and credential validation.
2. Multi-mode login support using **username, email, or phone number**, ensuring flexible access.
3. Ticket booking system for **bus and movie services** with real-time seat validation and overlap prevention.
4. Admin account is predefined and cannot be re-registered through the sign-up process.
5. Bookings are restricted to **current or future dates only**; past-dated bookings are automatically rejected.
6. **Admin capabilities** include adding/editing/deleting bus and movie records, viewing all bookings, and monitoring booking logs.
7. **User functionalities** include searching services, booking/canceling tickets, and viewing personal booking history.
8. Seat booking integrates **dynamic seat maps** with availability color-coding and input validation.
9. Data persistence is achieved through structured **text files**, with separate files for user data, bookings, seat availability, and records.
10. Each booking is uniquely timestamped and traceable to the user, ensuring **accountability and traceability**.

## **Limitations:**

* No GUI; all interactions occur through a command-line interface.
* Uses text-based file handling for data storage instead of a relational database system.
* Payment processing for bus and movie bookings is simulated; no real transaction
* integration.

## **Significance:**

1. Automates complex ticket booking operations with clear role segregation between Admin and User.
2. Strengthens data integrity through robust input validation and secure, hashed credential storage.
3. Prevents overbooking with real-time seat availability checks and dynamic seat maps.
4. Ensures clean removal of records (e.g., bus/movie entries) only when no active bookings exist.
5. Delivers a complete ticketing lifecycle—from user registration and service browsing to booking, cancellation, and history tracking.

# Development Strategy:

## Programming Environment:

**Development Environment:**

* Programming Language: C++ (Standard: C++13)
* IDE/Editors Used: Visual Studio 2019, Dev-C++,
* Compiler: MSVC (Visual Studio), MinGW (g++)
* Operating System: Windows 10 / Windows 11

## Libraries/Dependencies:

* <iostream> – Standard input/output operations
* <string> – String manipulations
* <fstream> – File I/O for persistent storage
* <windows.h> – Windows API functions for color and console manipulation
* <ctime> – Timestamp generation for bookings
* <sstream> – String stream conversions
* <conio.h> – Console input handling (e.g., \_getch() for password masking)
* <iomanip> – Output formatting (e.g., field width, precision)
* "Color.h" – Custom header for terminal text coloring
* "Validation.h" – Custom header for input validation (e.g., dates, names, formats)
* "menu.h" – Custom header for rendering and handling interactive CLI menus
* "wait.h" – Custom header for wait functions, cursor control, and loading animations
* **Build System:** Manual compilation via IDE or command-line using g++
* **Data Storage:** Text files using pipe ( “ | ” ) delimiters for structured persistence
* **Interface:** Command-Line Interface (CLI) only

# Classes Functionalities and Relationships:

## **1.Transportation Domain**

### A. Class Roles and Responsibilities

|  |  |
| --- | --- |
| **Class Name** | **Role/Responsibility** |
| Transport | Abstract base class for general transport-related properties. |
| Schedule | Abstract base class for scheduling-related data. |
| TransportOperator | Abstract base class for operator details. |
| Bus | Inherits from Transport, Schedule, and TransportOperator. Stores complete bus info. |
| BusDetailsInput | Provides static input functions. Uses setters of Bus to populate data (Association). |
| BusFile | Handles all file operations for bus data (Create/Read/Update/Delete). Uses BusRecord. |
| TransportDetailsDisplay | Displays transport data. Reads from file and formats output. |
| adminBusFunctionality | Static admin operations for buses (add/edit/remove). Relies on BusInterface. |
| UserBusFunctionality | Static user operations for buses (search/book/view). Relies on BusInterface. |
| BusInterface | User/Admin interface layer that routes options to the respective functionality classes. |

### B. Relationships

|  |  |  |  |
| --- | --- | --- | --- |
| **From Class** | **To Class** | **Type** | **Description** |
| Bus | Transport, Schedule, TransportOperator | Inheritance | Combines transport data and behaviors. |
| BusDetailsInput | Bus | Association | Uses setters of Bus to populate data. |
| BusFile | Bus, BusRecord | Aggregation | Uses for CRUD operations but does not own objects. |
| adminBusFunctionality | BusInterface | Association | Delegates to bus interface options. |
| UserBusFunctionality | BusInterface | Association | Delegates to bus interface options. |

## 2. Movie and Payment Domain

### A. Class Roles and Responsibilities

|  |  |
| --- | --- |
| **Class Name** | **Role/Responsibility** |
| MovieDetails | Core class containing all movie attributes. |
| MovieInput | Takes input and populates MovieDetails object using setters. |
| MovieFeatures | Contains logic for movie-related operations (add, find, delete, book, etc.). |
| MovieInterface | Interface for movie operations for both admin and user. |
| MovieTicket | Stores booked ticket information. |
| PaymentDetails | Stores payment data associated with ticket booking. |
| PaymentInputs | Handles input related to payment and populates PaymentDetails. |
| PaymentInterface | Coordinates payment operations. |

### B. Relationships

|  |  |  |  |
| --- | --- | --- | --- |
| **From Class** | **To Class** | **Type** | **Description** |
| MovieInput | MovieDetails | Aggregation | Owns and fills a MovieDetails object. |
| MovieFeatures | MovieDetails | Association | Uses MovieDetails for processing features like booking/searching. |
| MovieInterface | MovieFeatures | Association | Interfaces between user/admin and feature logic. |
| MovieTicket | MovieDetails, UserDetails, PaymentDetails | Association | Stores reference data for booking history. |
| PaymentInputs | PaymentDetails | Aggregation | Owns and fills a PaymentDetails object. |
| PaymentInterface | PaymentDetails, MovieTicket | Association | Facilitates payment logic tied to ticket booking. |

## 3. User, Login, and Interface Utilities

### A. Class Roles and Responsibilities

|  |  |
| --- | --- |
| **Class Name** | **Role/Responsibility** |
| UserDetails | Stores user-specific data. |
| UserInputs | Static input functions to populate UserDetails. |
| Login | Handles login and authentication process. Uses credentials from user/admin data. |
| Interface | Main user/admin interface controller. |

### B. Relationships

|  |  |  |  |
| --- | --- | --- | --- |
| **From Class** | **To Class** | **Type** | **Description** |
| UserInputs | UserDetails | Association | Populates UserDetails through setters. |
| Login | UserDetails | Association | Verifies credentials. |
| Interface | Login, UserBusFunctionality, MovieInterface, PaymentInterface | Association | Delegates control to relevant modules. |

## Validation, Menu, and Utility Classes

|  |  |
| --- | --- |
| **Class Name** | **Role/Responsibility** |
| Validation | Provides input validation utilities (e.g., CNIC, date format, etc.). |
| Print | Utility for formatted console printing. |
| Menu | Generates menu layouts for interface usage. |
| wait | Contains countdowns and timed lockout functionalities. |

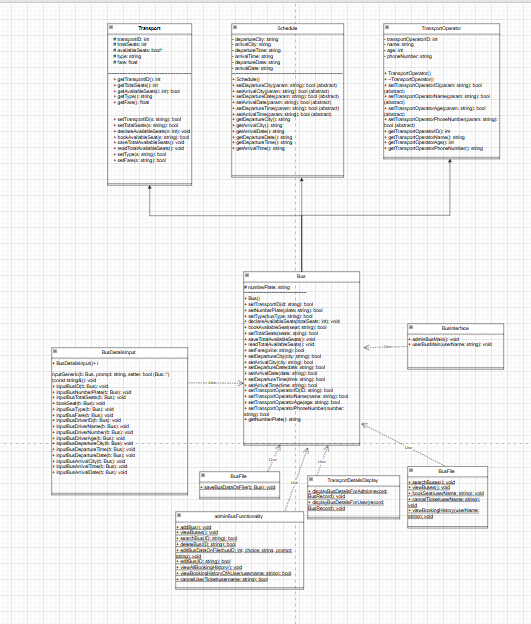
# UML

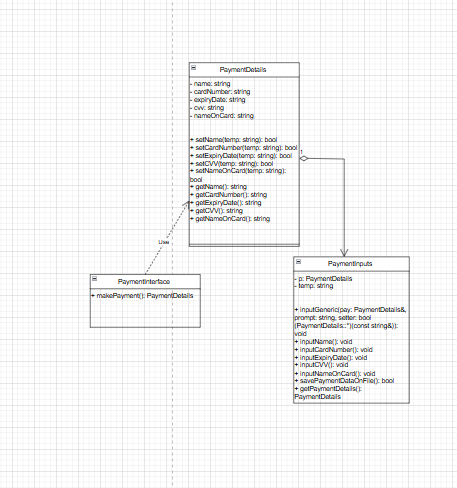
**UML Relationship Table:**

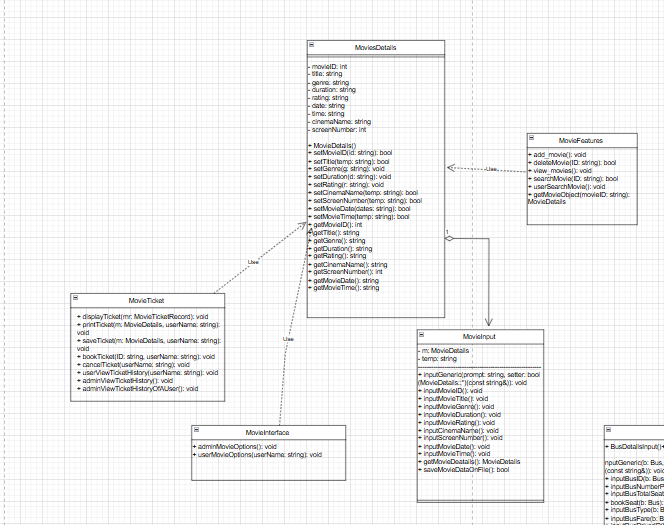
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Source Class** | **Relationship** | **Target Class** | **Type** | **Notes** |
| Bus | → | Transport | Association | Inherits (multiple inheritance) |
| Bus | → | TransportOperator | Association | Inherits |
| Bus | → | Schedule | Association | Inherits |
| BusDetailsInput | → | Bus | Association | Static functions use setters of Bus |
| BusFile | → | Bus | Association | Handles storage and retrieval |
| TransportDetailsDisplay | → | BusFile | Association | Reads and displays bus data |
| adminBusFunctionality | → | BusInterface | Association | Uses static methods |
| UserBusFunctionality | → | BusInterface | Association | Uses static methods |
| BusInterface | ◇ | adminBusFunctionality | Aggregation | Provides access to Admin functionalities |
| BusInterface | ◇ | UserBusFunctionality | Aggregation | Provides access to User functionalities |
| MovieInput | → | MovieDetails | Association | Uses object as parameter, sets data |
| MovieFeatures | → | MovieDetails | Association | Uses for find/book functionality |
| MovieInterface | ◇ | MovieFeatures | Aggregation | Uses feature-level methods |
| MovieInterface | ◇ | MovieInput | Aggregation | Invokes input routines |
| MovieTicket | → | MovieDetails | Association | Stores booked ticket info |
| PaymentInputs | → | PaymentDetails | Association | Populates data |
| PaymentInterface | ◇ | PaymentInputs | Aggregation | Uses for user input operations |
| PaymentInterface | ◇ | PaymentDetails | Aggregation | Manages payment records |
| UserInputs | → | UserDetails | Association | Static functions modify data |
| Login | → | UserDetails | Association | Used during auth |
| Interface | ◇ | Login | Aggregation | Manages login, acts as system entry |
| Interface | ◇ | MovieInterface | Aggregation | Connects all system parts |
| Interface | ◇ | BusInterface | Aggregation | Connects all system parts |
| Interface | ◇ | PaymentInterface | Aggregation | Connects all system parts |

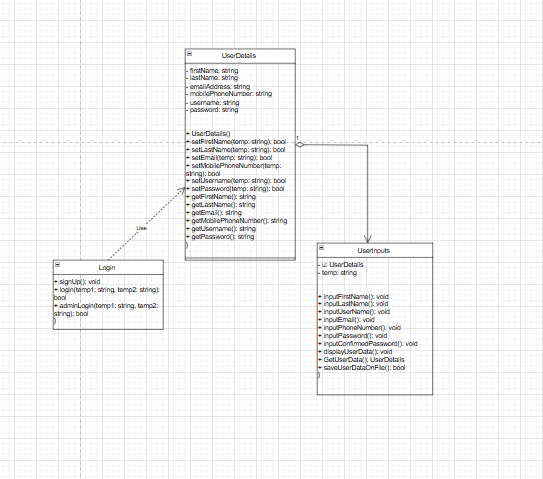
UML Screen Shot:

Bus classes









## Helper Classes

|  |  |
| --- | --- |
| **Helper Class** | **Purpose** |
| Validation | CNIC, date, and input validations |
| Print | Handles formatted output |
| Menu | Displays navigational menus |
| BusRecord, TicketRecord, UserRecord | Data structures for file operations |

UML Screen Shot:

