

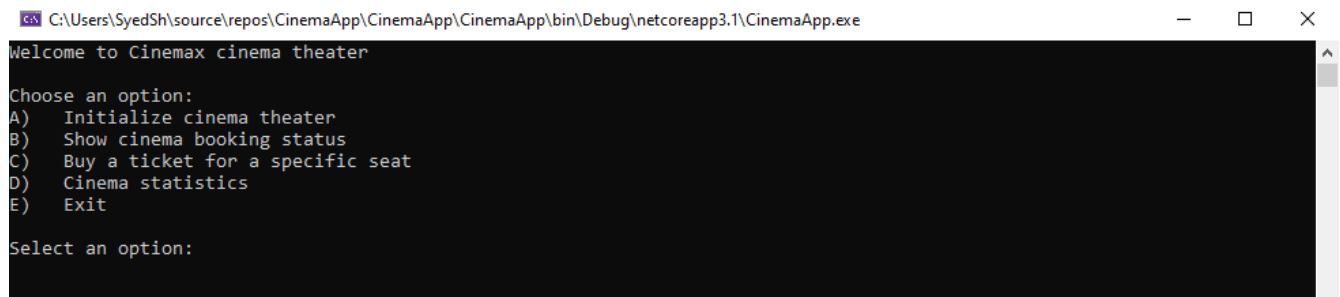
Solution design

The solution is divided into 3 projects

1. CinemaApp

Cinema app is .Net core console project. It is responsible for displaying available options for cinema hall and taking in option selected by user and passing the selected option to appropriate controller. Each menu option has an associated controller which is responsible for performing action based on the option selected. After completion the result is returned back and displayed by console app.

Main menu



```
C:\Users\SyedSh\source\repos\CinemaApp\CinemaApp\CinemaApp\bin\Debug\netcoreapp3.1\CinemaApp.exe
Welcome to Cinemax cinema theater
Choose an option:
A) Initialize cinema theater
B) Show cinema booking status
C) Buy a ticket for a specific seat
D) Cinema statistics
E) Exit
Select an option:
```

2. CinemaAppBackend

Cinema app backend is a .Net core class library project. It contains all the business logic for the cinema app. The project comprises of components like interfaces, repositories, services, validation handling, models, extensions and utility class. The project exposes one main backend interface `ICinemaAppBackendRepository` which is used by other projects. The backend repository contains a set of interfaces which are considered as use cases. The interface is responsible for calling appropriate use case. Each user action is associated with its own use case and all the logic for handling that use case is written in that

class. The CinemaAppBackendRepository calls the appropriate use case action based on the option selected.

For the purpose of this exercise the following options are provided and are handled as separate use cases.

Option A. Receive inputs for "number of rows" and "number of seats per row" for the given cinema room. All the logic for handling initializing cinema hall is provided by interface IInitializeCinemaHall. The class is responsible for initializing cinema hall object based on input "number of rows" and "number of seats per row". Cinema hall seats are initialized in terms of row and seats in each row.

```
C:\Users\SyedSh\source\repos\CinemaApp\CinemaApp\CinemaApp\bin\Debug\netcoreapp3.1\CinemaApp.exe
Enter "number of rows" and "number of seats per row" for the cinema hall
Enter no of rows: 3
Enter no of seats: 3
```

```
C:\Users\SyedSh\source\repos\CinemaApp\CinemaApp\CinemaApp\bin\Debug\netcoreapp3.1\CinemaApp.exe
Current booking status for cinema hall with 3 rows and 3 seats per row
Row: 1 Seat: 1 -> Status: Available
Row: 1 Seat: 2 -> Status: Available
Row: 1 Seat: 3 -> Status: Available
Row: 2 Seat: 1 -> Status: Available
Row: 2 Seat: 2 -> Status: Available
Row: 2 Seat: 3 -> Status: Available
Row: 3 Seat: 1 -> Status: Available
Row: 3 Seat: 2 -> Status: Available
Row: 3 Seat: 3 -> Status: Available
Press any key to go back to main menu !!!
```

Option B: Show an output of the seats and their status ("A" for available, "R" for reserved) This is considered as a case for displaying current booking status for the cinema hall. The logic is contained in ShowCinemaHallCurrentStatus use case and IShowCinemaHallCurrentStatus exposes the function for printing the cinema hall seat reservation status.

```
C:\Users\SyedSh\source\repos\CinemaApp\CinemaApp\CinemaApp\bin\Debug\netcoreapp3.1\CinemaApp.exe

Current booking status for cinema hall with 3 rows and 3 seats per row

Row: 1 Seat: 1 -> Status: Reserved
Row: 1 Seat: 2 -> Status: Available
Row: 1 Seat: 3 -> Status: Available

Row: 2 Seat: 1 -> Status: Available
Row: 2 Seat: 2 -> Status: Available
Row: 2 Seat: 3 -> Status: Available

Row: 3 Seat: 1 -> Status: Reserved
Row: 3 Seat: 2 -> Status: Available
Row: 3 Seat: 3 -> Status: Available

Press any key to go back to main menu !!!
```

Option C: Buy a ticket for a specific seat. The user should be able to choose which seat, however only available ones. This is considered as a case for buying cinema ticket. The user entered his desired seat number by entering row no and the seat in that row he wanted. the "BuyCinemaTicket" class has two function one for checking if the desired seat is available for reservation or not and then the other function is actually buying ticket. When the user buys the ticket the price is calculated and booking status is set to reserved.

```
C:\Users\SyedSh\source\repos\CinemaApp\CinemaApp\CinemaApp\bin\Debug\netcoreapp3.1\CinemaApp.exe

Current booking status for cinema hall with 3 rows and 3 seats per row

Row: 1 Seat: 1 -> Status: Available
Row: 1 Seat: 2 -> Status: Available
Row: 1 Seat: 3 -> Status: Available

Row: 2 Seat: 1 -> Status: Available
Row: 2 Seat: 2 -> Status: Available
Row: 2 Seat: 3 -> Status: Available

Row: 3 Seat: 1 -> Status: Available
Row: 3 Seat: 2 -> Status: Available
Row: 3 Seat: 3 -> Status: Available

Enter "seat number" for ticket reservation

Enter row number: 1
Enter seat number: 1
```

```
C:\Users\SyedSh\source\repos\CinemaApp\CinemaApp\CinemaApp\bin\Debug\netcoreapp3.1\CinemaApp.exe
```

```
Thank you for your reservation. Your seat is reserved at row 1 and seat number 1
```

```
Current booking status for cinema hall with 3 rows and 3 seats per row
```

```
Row: 1 Seat: 1 -> Status: Reserved
```

```
Row: 1 Seat: 2 -> Status: Available
```

```
Row: 1 Seat: 3 -> Status: Available
```

```
Row: 2 Seat: 1 -> Status: Available
```

```
Row: 2 Seat: 2 -> Status: Available
```

```
Row: 2 Seat: 3 -> Status: Available
```

```
Row: 3 Seat: 1 -> Status: Available
```

```
Row: 3 Seat: 2 -> Status: Available
```

```
Row: 3 Seat: 3 -> Status: Available
```

```
Do you want to reserve another seat? [y/n]
```

Option D: This is considered as the use case for getting statistics based on current situation for cinema hall. The logic is contained inside `GenerateCinemaHallStatistics` use case and it uses an object of type `CinemaHallStatistics` which has following properties. Output metrics for the cinema room, including:

- 1.Number of purchased tickets
- 2.Percentage occupied
- 3.Current income (sum of reserved tickets)
- 4.Potential total income (sum of all available and reserved tickets) After calculation the object is returned to display the statistics.

C:\Users\SyedSh\source\repos\CinemaApp\CinemaApp\CinemaApp\bin\Debug\netcoreapp3.1\CinemaApp.exe

Current booking status for cinema hall with 3 rows and 3 seats per row

Row: 1 Seat: 1 -> Status: Reserved
Row: 1 Seat: 2 -> Status: Available
Row: 1 Seat: 3 -> Status: Available

Row: 2 Seat: 1 -> Status: Available
Row: 2 Seat: 2 -> Status: Available
Row: 2 Seat: 3 -> Status: Available

Row: 3 Seat: 1 -> Status: Reserved
Row: 3 Seat: 2 -> Status: Available
Row: 3 Seat: 3 -> Status: Available

Current output metrics for the cinema room

Number of purchased tickets: 2 out of 9
Percentage occupied: 22.22%
Current income (sum of reserved tickets): 24.00
Potential total income (sum of all available and reserved tickets): 94.00

Press any key to go back to main menu !!!

Option E: Exist from cinema hall console App.

CinemaHallValidationService:

- 1.ValidateCinemaHallDimensions: Validates if user has entered valid dimensions for cinema hall i.e "number of rows" and "number of seats per row"
- 2.ValidateCinemaHall: Validates if cinema hall object is not null and if "number of rows" and "number of seats per row" are not zero.
- 3.ValidateSeatReservation: Validates if the row number and seat number entered by user fir buying ticket is valid.

Models:

1.**Cinema Hall**: Has properties like "number of rows" and "number of seats per row" and initializes cinema seat collection based on this input.

2.**Cinema Seat**: Has properties like Seat number, ticket price based on seat number and booking status for the seat (Available/Reserved)

3.**Cinema Hall Statistics**: Calculates different statistics based on current situation of cinema hall. the following things are calculated for now

- Number of purchased tickets
- Percentage occupied
- Current income (sum of reserved tickets)
- Potential total income (sum of all available and reserved tickets)

3. CinemaAppBackendUnitTest

The project is a .Net core NUnit test project responsible for testing functions and services provided by CinemaAppBackend.

Tools and Technologies:

.Net Core 3.0, C#, Serilog, Dependency injection, NUnit, Moq