Cinema Booking & Ticket Management System

1.Project Brief

Overview

a cinema management and booking system that models movie screenings, seating layouts, and ticket lifecycle operations, allowing users to book seats, generate tickets, and validate or redeem them through a structured and test- driven architecture.

Purpose

to provide a reliable and well-structured system for managing cinema halls, seat reservations, and ticket validation while ensuring data consistency, clear user feedback, and robust handling of all booking and ticket-related scenarios.

In One Sentence

A test-driven cinema booking system that manages seat reservations and ticket lifecycles from booking to validation

2. System Architecture Summary

Presentation Layer (GUI):

Handles user interactions and displays booking status and ticket information.

Services Layer:

Implements business logic for seat booking, ticket creation, validation, and redemption.

Domain Layer (Models):

Defines core entities such as Seat, Movie, CinemaHall, BookingRequest, and TicketInfo.

Data Layer:

Stores hall, movie, and seat data (in-memory or persistent

3. Key Features

Functional & Test-Driven:

Clean F# code with full unit test coverage.

Seat Booking Management:

Real-time seat availability and booking checks.

Ticket Lifecycle:

Create, validate, and redeem tickets with HTML output.

User Feedback:

Clear messages for success, errors, and validation states.

Structured Domain Models:

Robust representation of movies, halls, and bookings.

4. Testing Strategy (Detailed)

4.1 Cinema Models Tests

These tests verify the correctness and integrity of the core domain models used across the cinema management system. Their purpose is to ensure that all entities are created with valid structures and expected default behaviors.

- **SeatStatus enum values are correctly mapped:**
  - **Available equals 0.**
  - **Booked equals 1.**
- **Seat record correctly stores row, column, booking status, customer name, and booking time.**
- **Available seats correctly contain no booking information.**
- **Movie model correctly stores movie ID and title.**
- **PhysicalHall model correctly defines hall dimensions.**
- **CinemaHall correctly links movies, physical halls, time schedules, and seating layouts.**
- **Seats are stored as a two-dimensional array with correct width and height.**
- **BookingRequest correctly captures seat position and customer name.**
- **CinemaComplex correctly aggregates multiple cinema halls.**
- **TicketInfo model supports full ticket metadata and is compatible with serialization requirements.**

4.2 Booking Helpers Tests

These tests validate the logic responsible for seat booking interaction and user input validation. Their goal is to ensure consistent user feedback and prevent invalid booking requests.

- **Seat availability messages are correctly generated.**
- **Booked seats display the customer name when available.**
- **Booked seats without a customer name display a generic booked message.**
- **Booking input validation rejects missing seat selection.**
- **Booking input validation rejects empty customer names.**
- **Booking input validation rejects whitespace-only names.**
- **Valid booking input produces a correctly structured booking request.**

4.3 Ticket Helpers Tests

These tests validate ticket generation feedback and success messaging logic. Their purpose is to ensure that ticket creation results are communicated clearly to

the user.

- **Success messages include booking confirmation text.**
- **Successfully generated HTML tickets are reported with correct file names.**
- **HTML generation failures are gracefully handled and reported.**
- **Ticket identifiers are always included in success and failure messages.**
- **TicketInfo records are created with all required fields.**

4.4 Validation Helpers Tests

These tests ensure that ticket validation results are properly structured, formatted, and presented to the user.

- **Validation state correctly stores validity status, message, and optional ticket information.**
- **Validation logic handles missing ticket information gracefully.**
- **Ticket details are formatted with movie, hall, customer, seat, booking date, and ticket ID.**
- **Valid tickets display full validation messages and ticket details.**
- **Invalid tickets return only the validation error message.**
- **Ticket ID validation rejects empty input.**

4.5 Cinema Service Tests

These tests validate the core service-level domain behavior of the cinema booking system. Their purpose is to ensure that all service results and domain interactions are correctly modeled.

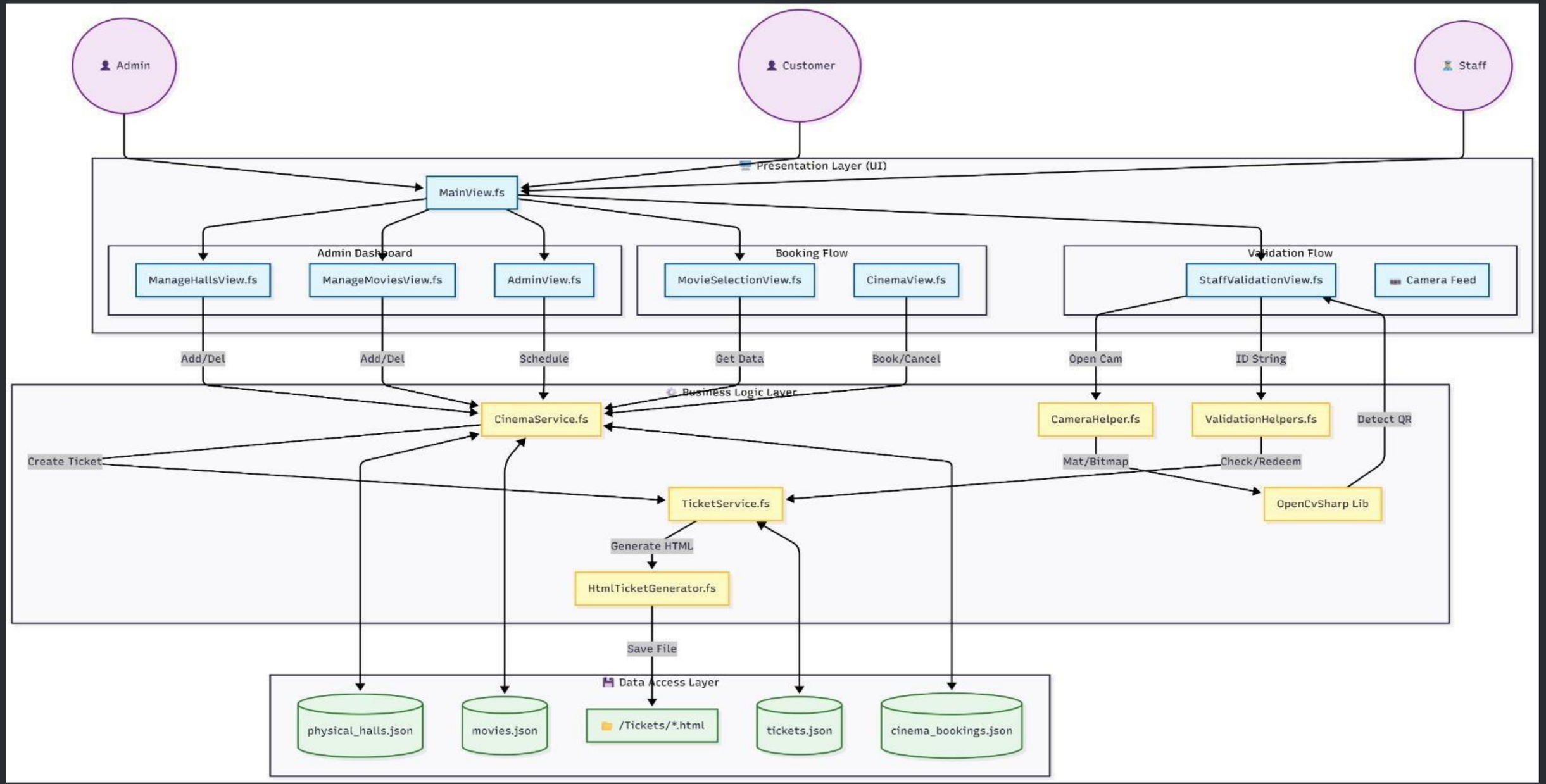
- Seat records correctly represent booking and availability states.
- Movie and PhysicalHall records correctly store identifying information.
- CinemaHall correctly represents movie scheduling data.
- Booking results support multiple outcomes:
  - Successful booking.
  - Successful booking with ticket generation.
  - Seat already booked.
  - Invalid seat selection.
  - General error handling.
- CinemaComplex correctly maintains a collection of cinema halls.

4.6 Ticket Service Tests

These tests validate ticket lifecycle operations including creation, validation, and redemption.

- TicketInfo records contain complete ticket metadata.
- Ticket validation correctly distinguishes between:
  - Valid tickets.
  - Invalid tickets with error messages.
  - Tickets not found.
  - Validation errors.
- Ticket operations support:
  - Ticket creation.
  - Ticket redemption.
  - Ticket operation error handling.
- All ticket-related results correctly encapsulate ticket information or error messages.

Diagram


Admin
Customer
Staff
Presentation Layer (UI)
MainView.fs
Admin Dashboard
ManageHallsView.fs
ManageMoviesView.fs
AdminView.fs
Booking Flow
MovieSelectionView.fs
CinemaView.fs
Validation Flow
StaffValidationView.fs
Camera Feed
Add/Del
Add/Del
Schedule
Get Data
Book/Cancel
Open Cam
ID String
Business Logic Layer
CinemaService.fs
CameraHelper.fs
ValidationHelpers.fs
Detect QR
Create Ticket
Mat/Bitmap
Check/Redeem
TicketService.fs
OpenCvSharp Lib
Generate HTML
HtmlTicketGenerator.fs
Save File
Data Access Layer
physical_halls.json
movies.json
/Tickets/*.html
tickets.json
cinema_bookings.json

6 Conclusion

The Cinema Booking & Ticket Management System demonstrates a clean, functional, and test-driven approach to managing cinema operations. It ensures reliable seat booking, structured ticket handling, and clear user feedback, while maintaining data integrity and extensibility for future growth. The layered architecture separates concerns effectively, making the system maintainable, scalable, and robust for real-world use.