# Perfectly-Viewable-Cinemas — Project Report

Version: 1.0.0  
Last updated: 2025-09-12

## Table of contents

* Executive Summary
* Goals and Scope
* System Overview and Architecture
* Data Model (CSV formats and examples)
* Core Components and Key Functions
* CLI: Walkthrough + Example Sessions (detailed)
* GUI (Streamlit): Screens, interactions, example output
* Testing and Validation
* Developer Notes: How to extend, add features, and run locally
* Troubleshooting & Common Errors
* Roadmap and Future Work
* Appendix: Sample files, API surface, and quick reference

## Executive summary

Perfectly-Viewable-Cinemas (PVC) is a lightweight movie booking system implemented in Python. It provides both a Command-Line Interface (CLI) and a Streamlit-based Graphical User Interface (GUI). The project stores data in CSV files (no external database required) and is suitable as a learning project or prototype for small cinemas.

This document serves as a detailed project report and developer guide. It documents the data model, expected CSV structures, examples of usage, internal function contracts, and diagnostics for common runtime errors. If you maintain or extend PVC, this README should be the single source of truth for expected file formats and runtime behavior.

## Goals and scope

* Provide a simple ticket booking flow (view movies -> select showing -> book seats).
* Allow theatre admins to add movies and showings, and to view bookings for their theatre.
* Provide a GUI using Streamlit for easy demonstration and a text-based CLI for quick testing.
* Keep persistence simple and portable using CSV files in the repository root.

## System overview and architecture

PVC is organized into the following layers:

* **Presentation**
  + cli.py — Single-file interactive CLI application.
  + gui.py — Streamlit application for a tab-based UI.
* **Business logic / Service layer**
  + handler.py — Core read/write logic, validation, and orchestration. All CSV operations should go through this module.
* **Data**
  + CSV files stored in the repository root. These are the source of truth: movies.csv, showings.csv, theatres.csv, users.csv, admins.csv, bookings.csv.

This architecture keeps the UI thin (presentation) and concentrates domain logic in handler.py. When adding new UIs, reuse the handler.py functions.

## Data model (CSV formats and examples)

All CSV files are comma-separated, include a header row, and use simple string fields. IDs are strings to preserve portability.

### 1) movies.csv

**Columns:** movie\_id,title,genre,duration

Example:

movie\_id,title,genre,duration  
1,Avatar: The Way of Water,Sci-Fi,192  
2,The Grand Budapest Hotel,Comedy,99

### 2) showings.csv

**Columns:** showing\_id,movie\_id,theatre\_id,showtime,available\_seats

Example:

showing\_id,movie\_id,theatre\_id,showtime,available\_seats  
1,1,1,18:00,50  
2,2,1,20:15,40

Notes: movie\_id must reference movies.csv. theatre\_id must exist in theatres.csv.

### 3) theatres.csv

**Columns:** theatre\_id,name,location,total\_seats

Example:

theatre\_id,name,location,total\_seats  
1,PVR Cinemas,Mall Road,100  
2,Central Screens,City Center,80

### 4) users.csv

**Columns:** user\_id,username,password,email

Example:

user\_id,username,password,email  
1,demo\_user,password123,demo@email.com

### 5) admins.csv

**Columns:** admin\_id,username,password,type,theatre\_id

Example:

admin\_id,username,password,type,theatre\_id  
1,system\_admin,admin123,system,  
2,theatre\_admin1,theatre123,theatre,1

### 6) bookings.csv

**Columns:** booking\_id,user\_id,showing\_id,seats\_booked,booking\_date

Example:

booking\_id,user\_id,showing\_id,seats\_booked,booking\_date  
1,1,1,2,2025-09-12T10:22:00

## Core components and key functions

Centralized in handler.py. Major functions:

* initialize\_csv\_files()
* get\_movies() -> list[dict]
* add\_movie(title, genre, duration) -> (bool, str)
* get\_showings() -> list[dict]
* add\_showing(movie\_id, theatre\_id, showtime, available\_seats) -> (bool, str)
* book\_ticket(user\_id, showing\_id, seats) -> (bool, str|booking\_id)
* get\_user\_bookings(user\_id) -> list[dict]
* get\_theatre\_bookings(theatre\_id) -> list[dict]
* get\_theatres(), get\_users(), get\_admins() — simple readers

## CLI: Walkthrough + Example Sessions

Start the CLI:

python cli.py

### Sample session: theatre admin adds a movie and a showing

=== MAIN MENU ===  
1. User Login  
2. User Registration  
3. Admin Login  
4. Exit  
Enter your choice (1-4): 3  
  
=== ADMIN LOGIN ===  
Username: theatre\_admin1  
Password: theatre123  
Welcome, theatre\_admin1! (theatre admin)  
  
=== THEATRE ADMIN MENU ===  
1. View Available Movies  
2. Add Movie  
3. View Theatre Bookings  
4. Logout  
Enter your choice (1-4): 2  
  
=== ADD MOVIE ===  
Movie Title: Lokah  
Genre: Sci fi  
Duration (minutes): 150  
Showtime (HH:MM): 18:00  
Available Seats: 50  
Movie 'Lokah' added successfully (id: 3).  
Showing added successfully (id: 3) for theatre 1.

### Booking example

Booking successful! Booking ID: 4

## GUI (Streamlit)

Run GUI:

streamlit run gui.py

Tabs:

* Login
* Register
* Movies (table of showings with details)
* Book Ticket
* My Bookings
* Admin: Movies
* Admin: Bookings

## Testing and validation

* Unit tests for handler.py.
* Manual smoke tests: run CLI and GUI end-to-end.

## Developer notes and how to run locally

**Prerequisites:** Python 3.8+

Install dependencies:

python -m venv .venv  
. .venv/bin/activate  
pip install -r requirements.txt

Run CLI:

python cli.py

Run GUI:

streamlit run gui.py

## Troubleshooting & common errors

1. **TypeError: add\_movie() takes 3 positional arguments but 6 were given**  
   Fix: separate add\_movie() and add\_showing() calls.
2. **Streamlit import errors**  
   Fix: pip install streamlit
3. **Pandas import errors**  
   Fix: pip install pandas
4. **CSV header mismatches**  
   Fix: delete and recreate with initialize\_csv\_files()

## Roadmap and future work

* Move from CSV to SQLite
* Use hashed passwords (bcrypt)
* Add unit tests and CI
* Add seat maps per theatre

## Appendix: sample CSV contents

### movies.csv

movie\_id,title,genre,duration  
1,Avatar: The Way of Water,Sci-Fi,192  
2,The Grand Budapest Hotel,Comedy,99  
3,Lokah,Sci-Fi,150

### showings.csv

showing\_id,movie\_id,theatre\_id,showtime,available\_seats  
1,1,1,18:00,50  
2,2,1,20:15,40  
3,3,1,18:00,50

### bookings.csv

booking\_id,user\_id,showing\_id,seats\_booked,booking\_date  
1,1,1,2,2025-09-12T10:22:00  
2,1,2,4,2025-09-12T10:25:00

## Contact and license

This project is licensed under the MIT License (see LICENSE file). For questions or contributions, open an issue or a PR on the project repository.