PERFECTLY-VIEWABLE-CINEMAS

Index

|  |  |  |
| --- | --- | --- |
| S. No. | Topic | Page No. |
| 1 | Certificate | 1 |
| 2 | Name of the Project | 2 |
| 3 | Index | 3 |
| 4 | Acknowledgement | 4 |
| 5 | Synopsis | 5 |
| 6 | Code | 6 |
| 7 | Output Screens | 7 |
| 8 | Bibliography | 8 |

ACKNOWLEDGEMENT

## We undertook this project as part of our class 12 Computer Science course. We had tried to use the best of our knowledge and experience to develop this software. Developing a software requires time, patience and systematic approach.

## We are thankful to Almighty for his immeasurable blessings.We would like to express our gratitude to our principal Ms.Prathiba.V and to our teacher Ms.Yasmin for their constant support and guidance.

## We extend our gratitude to all other staff members of Computer department and our parents for their moral support and patience. Last but not the least, we thank our school for providing a platform to develop our skills and help us gain knowledge.

Synopsis

Perfectly Viewable Cinemas is a modern cinema management and ticket booking system built with a CSV-based file system. It offers two interaction modes:

1. A command-line (CLI) interface for system administrators to manage platform-wide operations.
2. A class-based Streamlit GUI (CinemaGUI) providing visual interfaces for theatre admins and end-users.

### Core Features

* **Role-Based Access Control**: Separate privileges for system admin, theatre admin, and user roles
* **Enhanced Visual Experience**: Movie posters, emoji seat indicators, and modern icons
* **CSV Persistence**: File-based storage without external databases
* **Advanced Security**: Salted HMAC-SHA256 password hashing with enforcement of bans
* **Interactive Seat Selection**: Dynamic grid with real-time availability
* **Theatre Analytics**: Revenue tracking and booking aggregation

### User Types and Access Levels

#### 1. System Administrator

* **Access**: CLI only
* **Capabilities**:
  + Initialize/reset CSV data
  + Manage theatre admin accounts
  + Manage user accounts
  + Ban/unban users
  + Ensure data integrity
* **Authentication**: Username/password with elevated privileges

#### 2. Theatre Administrator

* **Access**: Streamlit GUI (admin dashboard)
* **Capabilities**:
  + Add new movie showings with metadata
  + View bookings for their theatre
  + Monitor revenue and analytics
  + Manage theatre inventory and schedules
* **Authentication**: Username/password tied to theatre ID

#### 3. Regular User (Customer)

* **Access**: Streamlit GUI (customer interface)
* **Capabilities**:
  + Browse movies across theatres
  + Select seats with real-time availability
  + Book tickets with conflict checks
  + View and cancel bookings
  + Register and manage account
* **Authentication**: Username/password with email verification

### Data Model (CSV Files)

CSV Data Model  
├── movies\_showings.csv  
│ ├── id  
│ ├── title  
│ ├── genre  
│ ├── duration  
│ ├── theatre\_id  
│ ├── showtime  
│ ├── available\_seats  
│ ├── price  
│ └── image\_url  
│  
├── users.csv  
│ ├── user\_id  
│ ├── username  
│ ├── password  
│ ├── salt  
│ ├── email  
│ └── status (active|banned)  
│  
├── admins.csv  
│ ├── admin\_id  
│ ├── username  
│ ├── password  
│ ├── salt  
│ ├── type (system|theatre)  
│ └── theatre\_id  
│  
└── bookings.csv  
 ├── booking\_id  
 ├── user\_id  
 ├── showing\_id  
 ├── seats\_booked  
 ├── seat\_numbers  
 ├── total\_price  
 └── booking\_date (ISO timestamp)

### Security Considerations

* **Password Security**: Salted HMAC-SHA256 hashes with secure random salts
* **Authentication Enforcement**: Banned users blocked from access
* **Session Management**: Streamlit session state preserves user context
* **Input Validation**: Validation and type-checking across CLI and GUI
* **Role Separation**: System admins restricted to CLI; GUI reserved for theatre admins and users

Code

### A. handler.py

Includes business logic: CSV management, core functions for movies, bookings, authentication.

import csv

import os

import datetime

from typing import List, Dict, Optional, Tuple

from crypto import hash\_password, verify\_password

# Global configuration

CSV\_FILES = {

    'movies\_showings': 'movies\_showings.csv',

    'users': 'users.csv',

    'admins': 'admins.csv',

    'bookings': 'bookings.csv'

}

def \_read\_csv(file\_key: str) -> List[Dict]:

    """Read a CSV file and return list of dictionaries."""

    data = []

    with open(CSV\_FILES[file\_key], 'r', newline='') as f:

        reader = csv.DictReader(f)

        data = list(reader)

    return data

def \_write\_csv(file\_key: str, data: List[Dict]):

    """Write list of dictionaries to CSV file."""

    headers = {

        'movies\_showings': ['id', 'title', 'genre', 'duration', 'theatre\_id', 'showtime', 'available\_seats', 'price', 'image\_url'],

        'users': ['user\_id', 'username', 'password', 'salt', 'email', 'status'],

        'admins': ['admin\_id', 'username', 'password', 'salt', 'type', 'theatre\_id'],

        'bookings': ['booking\_id', 'user\_id', 'showing\_id', 'seats\_booked', 'seat\_numbers', 'total\_price', 'booking\_date']

    }

    with open(CSV\_FILES[file\_key], 'w', newline='') as f:

        writer = csv.writer(f)

        # Always write headers

        writer.writerow(headers[file\_key])

        # Write data rows if any exist

        if data:

            dict\_writer = csv.DictWriter(f, fieldnames=headers[file\_key])

            dict\_writer.writerows(data)

def ensure\_csv\_files\_exist():

    """Initialize CSV files with headers and example data if they don't exist."""

    headers = {

        'movies\_showings': ['id', 'title', 'genre', 'duration', 'theatre\_id', 'showtime', 'available\_seats', 'price', 'image\_url'],

        'users': ['user\_id', 'username', 'password', 'salt', 'email', 'status'],

        'admins': ['admin\_id', 'username', 'password', 'salt', 'type', 'theatre\_id'],

        'bookings': ['booking\_id', 'user\_id', 'showing\_id', 'seats\_booked', 'seat\_numbers', 'total\_price', 'booking\_date']

    }

    # Example data to populate when creating new files

    example\_data = {

        'movies\_showings': [

            ['1', 'Oppenheimer', 'Biography/Drama', '180', '1', '19:00', '120', '15.00', 'https://m.media-amazon.com/images/M/MV5BMDBmYTZjNjUtN2M1MS00MTQ2LTk2ODgtNzc2M2QyZGE5NTVjXkEyXkFqcGdeQXVyNzAwMjU2MTY@.\_V1\_SX300.jpg'],

            ['2', 'Lokah Chapter 1: Chandra', 'Drama', '140', '1', '17:30', '80', '12.00', 'https://m.media-amazon.com/images/M/MV5BNjI2MGFkMTgtODJmMC00MjhmLTk2ZWEtYjczMTEzMzhkNzg5XkEyXkFqcGc@.\_V1\_.jpg'],

            ['3', 'Demon Slayer: Infinity Castle', 'Anime/Action', '117', '2', '20:30', '150', '10.00', 'https://upload.wikimedia.org/wikipedia/en/thumb/a/ae/Kimetsu\_No\_Yaiba\_Mugen\_Jyo-hen\_theatrical\_poster.jpg/250px-Kimetsu\_No\_Yaiba\_Mugen\_Jyo-hen\_theatrical\_poster.jpg'],

            ['4', 'La La Land', 'Romance/Musical', '128', '2', '18:45', '100', '13.00', 'https://m.media-amazon.com/images/M/MV5BMzUzNDM2NzM2MV5BMl5BanBnXkFtZTgwNTM3NTg4OTE@.\_V1\_SX300.jpg']

        ],

        'users': [

            # Regular user - username: demo, password: demo

            ['1', 'demo', 'bf46ada9f6237602b2e2818ecfd914bd18e822c7dcdaf4a835a80c870d4f5fe3', '22fe334fef8b2995d7cc1b5cf8f97880', 'demo@demo.com', 'active'],

            # Second regular user - username: john, password: user123

            ['2', 'john', '2cb8605e5e9330db6db5c9a810030403dad3d2ebc89a9ca57a1755f7874cefaa', '59947a5f5445a0ed45545bfd58d3bdae', 'john@cinema.com', 'active'],

            # Third regular user - username: alice, password: alice123

            ['3', 'alice', '44762a3a55f7ddfae12d806c9232a00229590ad2bb0072cfa8e4d30c03766ac1', 'a42d72101325ba5456e47d9aa3bc402f', 'alice@email.com', 'active']

        ],

        'admins': [

            # System admin - username: sysadmin, password: admin

            ['1', 'sysadmin', '9debf7cc5454c09f1ca385b50528164a0efd311b829afb0fd8052d294dbd1244', 'f132111b0b374f9cf1c6fc83d072e21a', 'system', ''],

            # Theatre admin for theatre 1 - username: theatre1, password: theatre1

            ['2', 'theatre1', 'feb2c5d2f340e320439353fc3561eceb747378e452d48ed41cad20807bc15396', 'd4ba216a203205f7e3f959c759eb9959', 'theatre', '1'],

            # Theatre admin for theatre 2 - username: theatre2, password: theatre2

            ['3', 'theatre2', 'a7014bc186c29cf0c2b5fd00250310fb2ad30f919600390b58fdcdbb471dc585', '372e733eab5eb9716a31478d545de037', 'theatre', '2']

        ],

        'bookings': []

    }

    for file\_key, filename in CSV\_FILES.items():

        if not os.path.exists(filename):

            with open(filename, 'w', newline='') as f:

                writer = csv.writer(f)

                writer.writerow(headers[file\_key])

                # Add example data if available for this file type

                if file\_key in example\_data:

                    writer.writerows(example\_data[file\_key])

def register\_user(username: str, password: str, email: str) -> bool:

    """Register a new user."""

    users = \_read\_csv('users')

    # Check if username or email already exists

    if any(u['username'] == username or u['email'] == email for u in users):

        return False

    # Hash password

    hashed\_pass, salt = hash\_password(password)

    # Generate new user ID

    user\_id = str(max([int(u['user\_id']) for u in users], default=0) + 1)

    new\_user = {

        'user\_id': user\_id,

        'username': username,

        'password': hashed\_pass,

        'salt': salt,

        'email': email,

        'status': 'active'

    }

    users.append(new\_user)

    \_write\_csv('users', users)

    return True

def authenticate\_user(username: str, password: str) -> Tuple[bool, Optional[Dict]]:

    """Authenticate a user or admin."""

    # Check users first

    users = \_read\_csv('users')

    for user in users:

        if user['username'] == username:

            # Check if user is banned

            if user.get('status', 'active') == 'banned':

                return False, None

            if verify\_password(password, user['password'], user['salt']):

                return True, {'type': 'user', 'id': user['user\_id']}

    # Check admins

    admins = \_read\_csv('admins')

    for admin in admins:

        if admin['username'] == username:

            if verify\_password(password, admin['password'], admin['salt']):

                return True, {'type': admin['type'], 'id': admin['admin\_id'],

                            'theatre\_id': admin['theatre\_id']}

    return False, None

def get\_movies\_showings(theatre\_id: Optional[str] = None) -> List[Dict]:

    """Get all movies and showings, optionally filtered by theatre."""

    movies = \_read\_csv('movies\_showings')

    if theatre\_id:

        return [m for m in movies if m['theatre\_id'] == theatre\_id]

    return movies

def add\_movie\_showing(title: str, genre: str, duration: int,

                     theatre\_id: str, showtime: str, seats: int, price: float) -> bool:

    """Add a new movie showing."""

    movies = \_read\_csv('movies\_showings')

    # Generate new ID

    movie\_id = str(max([int(m['id']) for m in movies], default=0) + 1)

    new\_movie = {

        'id': movie\_id,

        'title': title,

        'genre': genre,

        'duration': str(duration),

        'theatre\_id': theatre\_id,

        'showtime': showtime,

        'available\_seats': str(seats),

        'price': str(price),

        'image\_url': ''

    }

    movies.append(new\_movie)

    \_write\_csv('movies\_showings', movies)

    return True

def book\_tickets(user\_id: str, showing\_id: str, seat\_numbers: List[str]) -> Optional[str]:

    """Book tickets for a showing."""

    movies = \_read\_csv('movies\_showings')

    bookings = \_read\_csv('bookings')

    # Find the showing

    showing = None

    for m in movies:

        if m['id'] == showing\_id:

            showing = m

            break

    if not showing:

        return None

    # Check if seats are available

    available\_seats = int(showing['available\_seats'])

    if available\_seats < len(seat\_numbers):

        return None

    # Check if seats are already booked

    existing\_bookings = [b for b in bookings if b['showing\_id'] == showing\_id]

    booked\_seats = []

    for booking in existing\_bookings:

        booked\_seats.extend(booking['seat\_numbers'].split(','))

    if any(seat in booked\_seats for seat in seat\_numbers):

        return None

    # Create booking

    booking\_id = str(max([int(b['booking\_id']) for b in bookings], default=0) + 1)

    total\_price = len(seat\_numbers) \* float(showing['price'])

    new\_booking = {

        'booking\_id': booking\_id,

        'user\_id': user\_id,

        'showing\_id': showing\_id,

        'seats\_booked': str(len(seat\_numbers)),

        'seat\_numbers': ','.join(seat\_numbers),

        'total\_price': str(total\_price),

        'booking\_date': datetime.datetime.now().isofor()

    }

    # Update available seats

    showing['available\_seats'] = str(available\_seats - len(seat\_numbers))

    # Save changes

    bookings.append(new\_booking)

    \_write\_csv('bookings', bookings)

    \_write\_csv('movies\_showings', movies)

    return booking\_id

def get\_user\_bookings(user\_id: str) -> List[Dict]:

    """Get all bookings for a user."""

    bookings = \_read\_csv('bookings')

    return [b for b in bookings if b['user\_id'] == user\_id]

def cancel\_booking(booking\_id: str, user\_id: str) -> bool:

    """Cancel a booking and return seats to availability."""

    bookings = \_read\_csv('bookings')

    movies = \_read\_csv('movies\_showings')

    # Convert inputs to strings to ensure consistency

    booking\_id = str(booking\_id)

    user\_id = str(user\_id)

    # Find the booking

    booking = None

    for b in bookings:

        # Convert booking data to strings for comparison

        b\_booking\_id = str(b['booking\_id'])

        b\_user\_id = str(b['user\_id'])

        if b\_booking\_id == booking\_id and b\_user\_id == user\_id:

            booking = b

            break

    if not booking:

        # Debug: Print all bookings for troubleshooting

        print(f"DEBUG: Could not find booking {booking\_id} for user {user\_id}")

        print("Available bookings:")

        for b in bookings:

            print(f"  Booking {b['booking\_id']} for user {b['user\_id']}")

        return False

    # Update movie seats

    for movie in movies:

        if movie['id'] == booking['showing\_id']:

            current\_available = int(movie['available\_seats'])

            seats\_to\_restore = int(booking['seats\_booked'])

            movie['available\_seats'] = str(current\_available + seats\_to\_restore)

            print(f"DEBUG: Restored {seats\_to\_restore} seats to movie {movie['id']}")

            break

    # Remove booking - filter out the specific booking

    original\_count = len(bookings)

    bookings = [b for b in bookings if str(b['booking\_id']) != booking\_id]

    removed\_count = original\_count - len(bookings)

    print(f"DEBUG: Removed {removed\_count} booking(s) with ID {booking\_id}")

    print(f"DEBUG: Bookings remaining: {len(bookings)}")

    # Save changes

    \_write\_csv('bookings', bookings)

    \_write\_csv('movies\_showings', movies)

    # Verify the file was updated

    updated\_bookings = \_read\_csv('bookings')

    print(f"DEBUG: After save, CSV contains {len(updated\_bookings)} bookings")

    return True

def get\_theatre\_bookings(theatre\_id: str) -> List[Dict]:

    """Get all bookings for a specific theatre."""

    bookings = \_read\_csv('bookings')

    movies = \_read\_csv('movies\_showings')

    theatre\_movies = {m['id']: m for m in movies if m['theatre\_id'] == theatre\_id}

    return [b for b in bookings if b['showing\_id'] in theatre\_movies]

# Seat Layout and Visual Selection Functions

def get\_seat\_layout(showing\_id: str) -> Dict:

    """Get seat layout information for a showing."""

    movies = \_read\_csv('movies\_showings')

    bookings = \_read\_csv('bookings')

    # Find the showing

    showing = None

    for m in movies:

        if m['id'] == showing\_id:

            showing = m

            break

    if not showing:

        return {}

    # Calculate seat layout (assuming 10 seats per row)

    total\_seats = int(showing['available\_seats']) + sum(

        int(b['seats\_booked']) for b in bookings

        if b['showing\_id'] == showing\_id

    )

    rows = (total\_seats + 9) // 10  # Round up to get number of rows

    seats\_per\_row = min(10, total\_seats)

    # Get booked seats

    booked\_seats = []

    for booking in bookings:

        if booking['showing\_id'] == showing\_id:

            booked\_seats.extend(booking['seat\_numbers'].split(','))

    # Generate seat layout

    seat\_layout = {

        'rows': rows,

        'seats\_per\_row': seats\_per\_row,

        'total\_seats': total\_seats,

        'booked\_seats': booked\_seats,

        'available\_count': int(showing['available\_seats'])

    }

    return seat\_layout

def generate\_seat\_grid(total\_seats: int, seats\_per\_row: int = 10) -> List[List[str]]:

    """Generate a grid of seat identifiers."""

    rows = (total\_seats + seats\_per\_row - 1) // seats\_per\_row

    seat\_grid = []

    seat\_counter = 0

    for row in range(rows):

        row\_letter = chr(65 + row)  # A, B, C, etc.

        row\_seats = []

        for seat in range(seats\_per\_row):

            if seat\_counter < total\_seats:

                seat\_id = f"{row\_letter}{seat + 1}"

                row\_seats.append(seat\_id)

                seat\_counter += 1

            else:

                # Add empty placeholder to maintain grid structure

                row\_seats.append("")

        # Only add the row if it has at least one real seat

        if any(seat for seat in row\_seats):

            seat\_grid.append(row\_seats)

    return seat\_grid

def book\_tickets\_visual(user\_id: str, showing\_id: str, selected\_seats: List[str]) -> Optional[str]:

    """Book tickets using visual seat selection."""

    movies = \_read\_csv('movies\_showings')

    bookings = \_read\_csv('bookings')

    # Find the showing

    showing = None

    for m in movies:

        if m['id'] == showing\_id:

            showing = m

            break

    if not showing:

        return None

    # Check if selected seats are available

    existing\_bookings = [b for b in bookings if b['showing\_id'] == showing\_id]

    booked\_seats = []

    for booking in existing\_bookings:

        booked\_seats.extend(booking['seat\_numbers'].split(','))

    # Check if any selected seats are already booked

    if any(seat in booked\_seats for seat in selected\_seats):

        return None

    # Check if enough seats are available

    available\_seats = int(showing['available\_seats'])

    if available\_seats < len(selected\_seats):

        return None

    # Create booking

    booking\_id = str(max([int(b['booking\_id']) for b in bookings], default=0) + 1)

    total\_price = len(selected\_seats) \* float(showing['price'])

    new\_booking = {

        'booking\_id': booking\_id,

        'user\_id': user\_id,

        'showing\_id': showing\_id,

        'seats\_booked': str(len(selected\_seats)),

        'seat\_numbers': ','.join(selected\_seats),

        'total\_price': str(total\_price),

        'booking\_date': datetime.datetime.now().isoformat()

    }

    # Update available seats

    showing['available\_seats'] = str(available\_seats - len(selected\_seats))

    # Save changes

    bookings.append(new\_booking)

    \_write\_csv('bookings', bookings)

    \_write\_csv('movies\_showings', movies)

    return booking\_id

# Theatre Admin Management Functions

def create\_theatre\_admin(username: str, password: str, theatre\_id: str) -> bool:

    """Create a new theatre admin account."""

    admins = \_read\_csv('admins')

    # Check if username already exists

    if any(a['username'] == username for a in admins):

        return False

    # Hash password

    hashed\_pass, salt = hash\_password(password)

    # Generate new admin ID

    admin\_id = str(max([int(a['admin\_id']) for a in admins if a['admin\_id']], default=0) + 1)

    new\_admin = {

        'admin\_id': admin\_id,

        'username': username,

        'password': hashed\_pass,

        'salt': salt,

        'type': 'theatre',

        'theatre\_id': theatre\_id

    }

    admins.append(new\_admin)

    \_write\_csv('admins', admins)

    return True

def get\_all\_theatre\_admins() -> List[Dict]:

    """Get all theatre admin accounts."""

    admins = \_read\_csv('admins')

    return [a for a in admins if a['type'] == 'theatre']

def modify\_theatre\_admin(admin\_id: str, username: str = None,

                        password: str = None, theatre\_id: str = None) -> bool:

    """Modify a theatre admin account."""

    admins = \_read\_csv('admins')

    # Find the admin

    admin\_found = False

    for admin in admins:

        if admin['admin\_id'] == admin\_id and admin['type'] == 'theatre':

            admin\_found = True

            if username:

                # Check if new username already exists

                if any(a['username'] == username and a['admin\_id'] != admin\_id for a in admins):

                    return False

                admin['username'] = username

            if password:

                hashed\_pass, salt = hash\_password(password)

                admin['password'] = hashed\_pass

                admin['salt'] = salt

            if theatre\_id:

                admin['theatre\_id'] = theatre\_id

            break

    if not admin\_found:

        return False

    \_write\_csv('admins', admins)

    return True

def delete\_theatre\_admin(admin\_id: str) -> bool:

    """Delete a theatre admin account."""

    admins = \_read\_csv('admins')

    original\_count = len(admins)

    # Remove the admin

    admins = [a for a in admins if not (a['admin\_id'] == admin\_id and a['type'] == 'theatre')]

    if len(admins) == original\_count:

        return False

    \_write\_csv('admins', admins)

    return True

# User Account Management Functions

def get\_all\_users() -> List[Dict]:

    """Get all user accounts."""

    return \_read\_csv('users')

def modify\_user(user\_id: str, username: str = None,

               password: str = None, email: str = None) -> bool:

    """Modify a user account."""

    users = \_read\_csv('users')

    # Find the user

    user\_found = False

    for user in users:

        if user['user\_id'] == user\_id:

            user\_found = True

            if username:

                # Check if new username already exists

                if any(u['username'] == username and u['user\_id'] != user\_id for u in users):

                    return False

                user['username'] = username

            if password:

                hashed\_pass, salt = hash\_password(password)

                user['password'] = hashed\_pass

                user['salt'] = salt

            if email:

                # Check if new email already exists

                if any(u['email'] == email and u['user\_id'] != user\_id for u in users):

                    return False

                user['email'] = email

            break

    if not user\_found:

        return False

    \_write\_csv('users', users)

    return True

def delete\_user(user\_id: str) -> bool:

    """Delete a user account and their bookings."""

    users = \_read\_csv('users')

    bookings = \_read\_csv('bookings')

    movies = \_read\_csv('movies\_showings')

    # Check if user exists

    user\_exists = any(u['user\_id'] == user\_id for u in users)

    if not user\_exists:

        return False

    # Cancel all user's bookings and restore seats

    user\_bookings = [b for b in bookings if b['user\_id'] == user\_id]

    for booking in user\_bookings:

        # Restore seats to movie

        for movie in movies:

            if movie['id'] == booking['showing\_id']:

                movie['available\_seats'] = str(int(movie['available\_seats']) +

                                             int(booking['seats\_booked']))

                break

    # Remove user and their bookings

    users = [u for u in users if u['user\_id'] != user\_id]

    bookings = [b for b in bookings if b['user\_id'] != user\_id]

    # Save changes

    \_write\_csv('users', users)

    \_write\_csv('bookings', bookings)

    \_write\_csv('movies\_showings', movies)

    return True

# User Ban Management Functions

def ban\_user\_by\_email(email: str) -> bool:

    """Ban a user by their email address."""

    users = \_read\_csv('users')

    user\_found = False

    for user in users:

        if user['email'] == email:

            user\_found = True

            user['status'] = 'banned'

            break

    if not user\_found:

        return False

    \_write\_csv('users', users)

    return True

def unban\_user\_by\_email(email: str) -> bool:

    """Unban a user by their email address."""

    users = \_read\_csv('users')

    user\_found = False

    for user in users:

        if user['email'] == email:

            user\_found = True

            user['status'] = 'active'

            break

    if not user\_found:

        return False

    \_write\_csv('users', users)

    return True

def get\_banned\_users() -> List[Dict]:

    """Get all banned users."""

    users = \_read\_csv('users')

    return [u for u in users if u.get('status', 'active') == 'banned']

def find\_user\_by\_email(email: str) -> Optional[Dict]:

    """Find a user by their email address."""

    users = \_read\_csv('users')

    for user in users:

        if user['email'] == email:

            return user

    return None

# Initialize system when module is imported

ensure\_csv\_files\_exist()

### B. gui.py

Includes Streamlit interface: GUI tabs for login, movies, booking, admin control.

import streamlit as st

import handler

from datetime import datetime

class CinemaGUI:

    def \_\_init\_\_(self):

        pass

        if 'user' not in st.session\_state:

            st.session\_state.user = None

        if 'selected\_seats' not in st.session\_state:

            st.session\_state.selected\_seats = {}

        if 'current\_movie\_selection' not in st.session\_state:

            st.session\_state.current\_movie\_selection = None

    def force\_refresh\_seat\_data(self):

        """Clear any cached seat data to force refresh."""

        if 'seat\_layout\_cache' in st.session\_state:

            del st.session\_state.seat\_layout\_cache

        # Clear selected seats

        st.session\_state.selected\_seats = {}

        st.session\_state.current\_movie\_selection = None

    def login\_page(self):

        st.title("PVC - Cinema Management System")

        tab1, tab2 = st.tabs(["Login", "Register"])

        with tab1:

            with st.form("login\_form"):

                username = st.text\_input("Username")

                password = st.text\_input("Password", type="password")

                submitted = st.form\_submit\_button("Login")

                if submitted and username and password:

                    success, user\_info = handler.authenticate\_user(username, password)

                    if success:

                        if user\_info['type'] == 'system':

                            st.error("System admin must use CLI interface")

                        else:

                            st.session\_state.user = user\_info

                            st.rerun()

                    else:

                        st.error("Invalid credentials")

        with tab2:

            with st.form("register\_form"):

                new\_username = st.text\_input("Username")

                new\_password = st.text\_input("Password", type="password")

                email = st.text\_input("Email")

                submitted = st.form\_submit\_button("Register")

                if submitted and new\_username and new\_password and email:

                    if handler.register\_user(new\_username, new\_password, email):

                        st.success("Registration successful! Please login.")

                    else:

                        st.error("Registration failed. Username or email already exists.")

    def user\_interface(self):

        st.title("Movie Booking System")

        # Initialize page in session state if not exists

        if 'current\_page' not in st.session\_state:

            st.session\_state.current\_page = "Browse Movies"

        # Sidebar navigation

        with st.sidebar:

            st.markdown("### Navigation")

            if st.button("Browse Movies", icon=":material/theaters:", use\_container\_width=True):

                st.session\_state.current\_page = "Browse Movies"

                st.rerun()

            if st.button("My Bookings", icon=":material/confirmation\_number:", use\_container\_width=True):

                st.session\_state.current\_page = "My Bookings"

                st.rerun()

            st.markdown("---")

            if st.button("Logout", type="primary", icon=":material/logout:", use\_container\_width=True):

                st.session\_state.user = None

                st.rerun()

        # Display the selected page

        if st.session\_state.current\_page == "Browse Movies":

            self.show\_movies\_page()

        elif st.session\_state.current\_page == "My Bookings":

            self.show\_bookings\_page()

    def display\_seat\_selection(self, movie):

        """Display visual seat selection grid for a movie."""

        st.markdown("---")

        st.subheader(f":material/theater\_comedy: Select Seats for: {movie['title']}")

        # Get seat layout information

        seat\_layout = handler.get\_seat\_layout(movie['id'])

        if not seat\_layout:

            st.error("Unable to load seat layout")

            return False

        # Generate seat grid

        seat\_grid = handler.generate\_seat\_grid(

            seat\_layout['total\_seats'],

            seat\_layout['seats\_per\_row']

        )

        # Display seat information

        col1, col2, col3 = st.columns(3)

        with col1:

            st.metric("Available Seats", seat\_layout['available\_count'])

        with col2:

            st.metric("Total Seats", seat\_layout['total\_seats'])

        with col3:

            st.metric("Price per Seat", f"${float(movie['price']):.2f}")

        # Legend

        st.markdown("#### Legend:")

        col1, col2, col3 = st.columns(3)

        with col1:

            st.markdown("🟢 \*\*Available\*\*")

        with col2:

            st.markdown("🔴 \*\*Booked\*\*")

        with col3:

            st.markdown("🟡 \*\*Selected\*\*")

        # Initialize selected seats for this movie if not exists

        movie\_id = movie['id']

        if movie\_id not in st.session\_state.selected\_seats:

            st.session\_state.selected\_seats[movie\_id] = []

        # Display seat grid

        st.markdown("#### Select Your Seats:")

        selected\_seats = st.session\_state.selected\_seats[movie\_id].copy()

        for row\_index, row in enumerate(seat\_grid):

            # Create columns for the row - use max 10 seats plus row label

            max\_seats\_in\_row = max(len(r) for r in seat\_grid) if seat\_grid else 10

            cols = st.columns([1] + [1] \* max\_seats\_in\_row)  # Row label + seats

            # Row label

            with cols[0]:

                st.markdown(f"\*\*Row {chr(65 + row\_index)}\*\*")

            # Seats in this row

            for seat\_index, seat\_id in enumerate(row):

                with cols[seat\_index + 1]:

                    if seat\_id == "":

                        # Empty placeholder for consistent grid

                        st.markdown("&nbsp;")

                    else:

                        is\_booked = seat\_id in seat\_layout['booked\_seats']

                        is\_selected = seat\_id in selected\_seats

                        if is\_booked:

                            # Show booked seat (disabled)

                            st.markdown("🔴")

                            st.caption(f"{seat\_id}")

                            st.caption("Booked")

                        else:

                            # Show available seat with checkbox

                            checkbox\_key = f"seat\_{seat\_id}\_{movie\_id}"

                            checked = st.checkbox(

                                seat\_id,

                                value=is\_selected,

                                key=checkbox\_key,

                                label\_visibility="hidden",

                                help=f"Select seat {seat\_id}"

                            )

                            # Update selected seats

                            if checked and seat\_id not in selected\_seats:

                                selected\_seats.append(seat\_id)

                            elif not checked and seat\_id in selected\_seats:

                                selected\_seats.remove(seat\_id)

                            # Visual indicator

                            if checked:

                                st.markdown("🟡")

                            else:

                                st.markdown("🟢")

                            st.badge(f"{seat\_id}", color="primary")

        if st.button("Cancel",icon=":material/cancel:", key=f"cancel\_{movie\_id},"):

                    self.force\_refresh\_seat\_data()

                    st.rerun()

        # Update session state

        st.session\_state.selected\_seats[movie\_id] = selected\_seats

        # Booking summary and actions

        if selected\_seats:

            st.markdown("---")

            st.markdown("#### :material/assignment: Booking Summary")

            total\_price = len(selected\_seats) \* float(movie['price'])

            col1, col2 = st.columns(2)

            with col1:

                st.write(f"\*\*Selected Seats:\*\* {', '.join(selected\_seats)}")

                st.write(f"\*\*Number of Seats:\*\* {len(selected\_seats)}")

            with col2:

                st.write(f"\*\*Total Price:\*\* ${total\_price:.2f}")

            # Booking buttons

            col1, col2= st.columns(2)

            with col1:

                if st.button("Confirm Booking", type="primary",icon=":material/done\_outline:", key=f"confirm\_{movie\_id}"):

                    booking\_id = handler.book\_tickets\_visual(

                        st.session\_state.user['id'],

                        movie['id'],

                        selected\_seats

                    )

                    if booking\_id:

                        st.success(f"Booking successful! Booking ID: {booking\_id}")

                        self.force\_refresh\_seat\_data()

                        st.balloons()

                        st.rerun()

                    else:

                        st.error("Booking failed. Some seats may have been taken.")

            with col2:

                if st.button("Clear Selection",icon=":material/delete:", key=f"clear\_{movie\_id}"):

                    st.session\_state.selected\_seats[movie\_id] = []

                    st.rerun()

        return True

    def theatre\_admin\_interface(self):

        st.title("Theatre Admin Panel")

        # Sidebar navigation

        page = st.sidebar.selectbox(

            "Navigation",

            ["Add Movie/Showing", "Theatre Bookings"]

        )

        if page == "Add Movie/Showing":

            self.show\_add\_movie\_page()

        elif page == "Theatre Bookings":

            self.show\_theatre\_bookings\_page()

        if st.sidebar.button("Logout"):

            st.session\_state.user = None

            st.rerun()

    def show\_movies\_page(self):

        st.header("Available Movies and Showings")

        movies = handler.get\_movies\_showings()

        if not movies:

            st.info("No movies available at the moment.")

            return

        # Check if we're currently selecting seats for a movie

        if st.session\_state.current\_movie\_selection:

            selected\_movie = next(

                (m for m in movies if m['id'] == st.session\_state.current\_movie\_selection),

                None

            )

            if selected\_movie:

                self.display\_seat\_selection(selected\_movie)

                return

        # Display available movies

        for movie in movies:

            with st.expander(f"{movie['title']} - {movie['showtime']} | {movie['genre']}",):

                col1, col2, col3 = st.columns([2, 1, 1])

                with col1:

                    st.write(f"\*\*Genre:\*\* {movie['genre']}")

                    st.write(f"\*\*Duration:\*\* {movie['duration']} minutes")

                    st.write(f"\*\*Showtime:\*\* {movie['showtime']}")

                    st.write(f"\*\*Available Seats:\*\* {movie['available\_seats']}")

                    st.write(f"\*\*Price:\*\* ${float(movie['price']):.2f} per seat")

                with col2:

                    try:

                        if movie['image\_url'] and movie['image\_url'].strip():

                            st.image(movie['image\_url'], caption=movie['title'], width=200)

                        else:

                            st.write(f"🎬 {movie['title']}")

                    except Exception as e:

                        st.write(f"🎬 {movie['title']}")

                with col3:

                    if int(movie['available\_seats']) > 0:

                        # Visual seat selection button

                        if st.button(

                            "Select Seats",

                            key=f"visual\_select\_{movie['id']}",

                            type="primary",

                            icon=":material/comedy\_mask:"

                        ):

                            st.session\_state.current\_movie\_selection = movie['id']

                            st.rerun()

                    else:

                        st.error("No seats available")

                        st.write("This showing is fully booked.")

    def show\_bookings\_page(self):

        st.header("My Bookings")

        bookings = handler.get\_user\_bookings(st.session\_state.user['id'])

        if not bookings:

            st.info("You have no bookings")

        else:

            for index, booking in enumerate(bookings):

                with st.expander(f"Booking #{index + 1} - ID: {booking['booking\_id']}"):

                    col1, col2 = st.columns(2)

                    with col1:

                        st.write(f"\*\*Booking ID:\*\* {booking['booking\_id']}")

                        st.write(f"\*\*Movie ID:\*\* {booking['showing\_id']}")

                        st.write(f"\*\*Seats:\*\* {booking['seat\_numbers']}")

                    with col2:

                        st.write(f"\*\*Total Price:\*\* ${float(booking['total\_price']):.2f}")

                        st.write(f"\*\*Booking Date:\*\* {booking['booking\_date'][:19]}")

                    # Unique button key using both index and booking ID

                    button\_key = f"cancel\_booking\_{index}\_{booking['booking\_id']}"

                    if st.button("Cancel This Booking", key=button\_key, type="secondary", icon=":material/delete:"):

                        success = handler.cancel\_booking(

                            booking['booking\_id'],

                            st.session\_state.user['id']

                        )

                        if success:

                            st.success("Booking cancelled successfully!")

                            self.force\_refresh\_seat\_data()

                            st.rerun()

                        else:

                            st.error(f"Failed to cancel booking ID: {booking['booking\_id']}")

                    st.markdown("---")

    def show\_add\_movie\_page(self):

        st.header("Add New Movie/Showing")

        with st.form("add\_movie\_form"):

            title = st.text\_input("Movie Title")

            genre = st.text\_input("Genre")

            duration = st.number\_input("Duration (minutes)", min\_value=1, value=90)

            showtime = st.text\_input("Showtime (HH:MM)")

            seats = st.number\_input("Number of Seats", min\_value=1, value=50)

            price = st.number\_input("Ticket Price ($)", min\_value=0.0, value=10.0, step=0.5)

            image\_url = st.text\_input("Movie Poster URL", help="Enter the URL of the movie poster image")

            if st.form\_submit\_button("Add Movie/Showing"):

                if all([title, genre, showtime]):

                    if handler.add\_movie\_showing(

                        title, genre, duration,

                        st.session\_state.user['theatre\_id'],

                        showtime, seats, price

                    ):

                        st.success("Movie/Showing added successfully!")

                        st.rerun()

                    else:

                        st.error("Failed to add movie/showing")

                else:

                    st.error("Please fill all fields")

    def show\_theatre\_bookings\_page(self):

        st.header("Theatre Bookings")

        bookings = handler.get\_theatre\_bookings(st.session\_state.user['theatre\_id'])

        if not bookings:

            st.info("No bookings for your theatre")

        else:

            # Calculate total revenue

            total\_revenue = sum(float(b['total\_price']) for b in bookings)

            st.metric("Total Revenue", f"${total\_revenue:.2f}")

            for booking in bookings:

                with st.expander(f"Booking ID: {booking['booking\_id']}"):

                    st.write(f"User ID: {booking['user\_id']}")

                    st.write(f"Movie ID: {booking['showing\_id']}")

                    st.write(f"Seats: {booking['seat\_numbers']}")

                    st.write(f"Total Price: ${float(booking['total\_price']):.2f}")

                    st.write(f"Booking Date: {booking['booking\_date'][:19]}")

def main():

    # Set page configuration

    st.set\_page\_config(

        page\_title="PVC Cinema Management",

        page\_icon=":material/movie:",

        layout="wide"

    )

    # Custom CSS for better styling

    st.markdown("""

    <style>

    .seat-available {

        background-color: #90EE90;

        border: 1px solid #006400;

        border-radius: 5px;

        padding: 5px;

        text-align: center;

        margin: 2px;

    }

    .seat-booked {

        background-color: #FF6B6B;

        border: 1px solid #8B0000;

        border-radius: 5px;

        padding: 5px;

        text-align: center;

        margin: 2px;

        color: white;

    }

    .seat-selected {

        background-color: #FFD700;

        border: 1px solid #FFA500;

        border-radius: 5px;

        padding: 5px;

        text-align: center;

        margin: 2px;

    }

    .stButton > button {

        width: 100%;

    }

    .movie-card {

        border: 1px solid #ddd;

        border-radius: 10px;

        padding: 20px;

        margin: 10px 0;

        background-color: #f9f9f9;

    }

    </style>

    """, unsafe\_allow\_html=True)

    gui = CinemaGUI()

    if st.session\_state.user is None:

        gui.login\_page()

    else:

        if st.session\_state.user['type'] == 'user':

            gui.user\_interface()

        elif st.session\_state.user['type'] == 'theatre':

            gui.theatre\_admin\_interface()

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**C. cli.py**

Includes CLI interface: user/admin menus, booking flow, admin operations.

import os

from typing import Dict

import handler as handler

def clear\_screen():

    """Clear the terminal screen."""

    # os.system('cls' if os.name == 'nt' else 'clear')

def print\_menu(options: Dict[str, str]):

    """Print menu options."""

    print("\n" + "="\*50)

    for key, value in options.items():

        print(f"{key}. {value}")

    print("="\*50)

def get\_input(prompt: str, validate=None) -> str:

    """Get user input with optional validation."""

    while True:

        value = input(prompt).strip()

        if validate is None or validate(value):

            return value

        print("Invalid input, please try again.")

def login\_menu():

    """Display login menu and handle authentication."""

    while True:

        clear\_screen()

        print("Welcome to PVC - System Admin Interface")

        print\_menu({

            "1": "Login as System Admin",

            "0": "Exit"

        })

        choice = get\_input("Choose an option: ")

        if choice == "1":

            username = get\_input("Username: ")

            password = get\_input("Password: ")

            success, user\_info = handler.authenticate\_user(username, password)

            if success and user\_info['type'] == 'system':

                current\_user = user\_info

                current\_user['username'] = username

                system\_admin\_menu(current\_user)

            else:

                input("Invalid system admin credentials. Press Enter to continue...")

        elif choice == "0":

            print("Goodbye!")

            break

def system\_admin\_menu(current\_user: Dict):

    """Display system admin menu."""

    while True:

        clear\_screen()

        print(f"Welcome, System Admin!")

        print\_menu({

            "1": "Initialize/Reset CSVs",

            "2": "Manage Theatre Admins",

            "3": "Manage User Accounts",

            "4": "Ban/Unban Users",

            "0": "Logout"

        })

        choice = get\_input("Choose an option: ")

        if choice == "1":

            handler.ensure\_csv\_files\_exist()

            print("CSV files have been initialized/reset.")

            input("Press Enter to continue...")

        elif choice == "2":

            manage\_theatre\_admins()

        elif choice == "3":

            manage\_user\_accounts()

        elif choice == "4":

            manage\_user\_bans()

        elif choice == "0":

            break

def manage\_theatre\_admins():

    """Manage theatre admin accounts."""

    while True:

        clear\_screen()

        print("Theatre Admin Management")

        print\_menu({

            "1": "View All Theatre Admins",

            "2": "Create Theatre Admin",

            "3": "Modify Theatre Admin",

            "4": "Delete Theatre Admin",

            "0": "Back to Main Menu"

        })

        choice = get\_input("Choose an option: ")

        if choice == "1":

            view\_theatre\_admins()

        elif choice == "2":

            create\_theatre\_admin()

        elif choice == "3":

            modify\_theatre\_admin()

        elif choice == "4":

            delete\_theatre\_admin()

        elif choice == "0":

            break

def view\_theatre\_admins():

    """View all theatre admins."""

    clear\_screen()

    admins = handler.get\_all\_theatre\_admins()

    if not admins:

        print("No theatre admins found.")

    else:

        print("\nTheatre Admins:")

        print("-" \* 80)

        print(f"{'Admin ID':<10} {'Username':<20} {'Theatre ID':<15}")

        print("-" \* 80)

        for admin in admins:

            print(f"{admin['admin\_id']:<10} {admin['username']:<20} {admin['theatre\_id']:<15}")

    input("\nPress Enter to continue...")

def create\_theatre\_admin():

    """Create a theatre admin."""

    clear\_screen()

    print("Create New Theatre Admin")

    print("-" \* 50)

    username = get\_input("Username: ")

    password = get\_input("Password: ")

    theatre\_id = get\_input("Theatre ID: ")

    if handler.create\_theatre\_admin(username, password, theatre\_id):

        print("Theatre admin created successfully!")

    else:

        print("Failed to create theatre admin. Username may already exist.")

    input("\nPress Enter to continue...")

def modify\_theatre\_admin():

    """Modify a theatre admin."""

    clear\_screen()

    view\_theatre\_admins()

    admin\_id = get\_input("\nEnter Admin ID to modify (or 0 to go back): ")

    if admin\_id == "0":

        return

    print("\nLeave fields empty to keep current values:")

    username = get\_input("New Username (or press Enter to skip): ")

    password = get\_input("New Password (or press Enter to skip): ")

    theatre\_id = get\_input("New Theatre ID (or press Enter to skip): ")

    # Convert empty strings to None

    username = username if username else None

    password = password if password else None

    theatre\_id = theatre\_id if theatre\_id else None

    if handler.modify\_theatre\_admin(admin\_id, username, password, theatre\_id):

        print("Theatre admin modified successfully!")

    else:

        print("Failed to modify theatre admin. Admin ID not found or username already exists.")

    input("\nPress Enter to continue...")

def delete\_theatre\_admin():

    """Delete a theatre admin."""

    clear\_screen()

    view\_theatre\_admins()

    admin\_id = get\_input("\nEnter Admin ID to delete (or 0 to go back): ")

    if admin\_id == "0":

        return

    confirm = get\_input("Are you sure you want to delete this admin? (y/N): ")

    if confirm.lower() == 'y':

        if handler.delete\_theatre\_admin(admin\_id):

            print("Theatre admin deleted successfully!")

        else:

            print("Failed to delete theatre admin. Admin ID not found.")

    else:

        print("Deletion cancelled.")

    input("\nPress Enter to continue...")

def manage\_user\_accounts():

    """Manage user accounts."""

    while True:

        clear\_screen()

        print("User Account Management")

        print\_menu({

            "1": "View All Users",

            "2": "Modify User Account",

            "3": "Delete User Account",

            "0": "Back to Main Menu"

        })

        choice = get\_input("Choose an option: ")

        if choice == "1":

            view\_all\_users()

        elif choice == "2":

            modify\_user\_account()

        elif choice == "3":

            delete\_user\_account()

        elif choice == "0":

            break

def view\_all\_users():

    """View all users."""

    clear\_screen()

    users = handler.get\_all\_users()

    if not users:

        print("No users found.")

    else:

        print("\nAll Users:")

        print("-" \* 90)

        print(f"{'User ID':<10} {'Username':<20} {'Email':<30} {'Status':<10}")

        print("-" \* 90)

        for user in users:

            status = user.get('status', 'active')

            print(f"{user['user\_id']:<10} {user['username']:<20} {user['email']:<30} {status:<10}")

    input("\nPress Enter to continue...")

def modify\_user\_account():

    """Modify a user account."""

    clear\_screen()

    view\_all\_users()

    user\_id = get\_input("\nEnter User ID to modify (or 0 to go back): ")

    if user\_id == "0":

        return

    print("\nLeave fields empty to keep current values:")

    username = get\_input("New Username (or press Enter to skip): ")

    password = get\_input("New Password (or press Enter to skip): ")

    email = get\_input("New Email (or press Enter to skip): ")

    # Convert empty strings to None

    username = username if username else None

    password = password if password else None

    email = email if email else None

    if handler.modify\_user(user\_id, username, password, email):

        print("User account modified successfully!")

    else:

        print("Failed to modify user account. User ID not found or username/email already exists.")

    input("\nPress Enter to continue...")

def delete\_user\_account():

    """Delete a user account."""

    clear\_screen()

    view\_all\_users()

    user\_id = get\_input("\nEnter User ID to delete (or 0 to go back): ")

    if user\_id == "0":

        return

    confirm = get\_input("Are you sure you want to delete this user and all their bookings? (y/N): ")

    if confirm.lower() == 'y':

        if handler.delete\_user(user\_id):

            print("User account and all bookings deleted successfully!")

        else:

            print("Failed to delete user account. User ID not found.")

    else:

        print("Deletion cancelled.")

    input("\nPress Enter to continue...")

def manage\_user\_bans():

    """Manage user bans."""

    while True:

        clear\_screen()

        print("User Ban Management")

        print\_menu({

            "1": "View Banned Users",

            "2": "Ban User by Email",

            "3": "Unban User by Email",

            "4": "Check User Status by Email",

            "0": "Back to Main Menu"

        })

        choice = get\_input("Choose an option: ")

        if choice == "1":

            view\_banned\_users()

        elif choice == "2":

            ban\_user\_by\_email()

        elif choice == "3":

            unban\_user\_by\_email()

        elif choice == "4":

            check\_user\_status\_by\_email()

        elif choice == "0":

            break

def view\_banned\_users():

    """View banned users."""

    clear\_screen()

    banned\_users = handler.get\_banned\_users()

    if not banned\_users:

        print("No banned users found.")

    else:

        print("\nBanned Users:")

        print("-" \* 80)

        print(f"{'User ID':<10} {'Username':<20} {'Email':<30} {'Status':<10}")

        print("-" \* 80)

        for user in banned\_users:

            print(f"{user['user\_id']:<10} {user['username']:<20} {user['email']:<30} {user['status']:<10}")

    input("\nPress Enter to continue...")

def ban\_user\_by\_email():

    """Ban a user by email."""

    clear\_screen()

    print("Ban User by Email")

    print("-" \* 50)

    email = get\_input("Enter user email to ban: ")

    # First check if user exists

    user = handler.find\_user\_by\_email(email)

    if not user:

        print("No user found with that email address.")

    elif user.get('status', 'active') == 'banned':

        print("User is already banned.")

    else:

        if handler.ban\_user\_by\_email(email):

            print(f"User with email '{email}' has been banned successfully!")

        else:

            print("Failed to ban user.")

    input("\nPress Enter to continue...")

def unban\_user\_by\_email():

    """Unban a user by email."""

    clear\_screen()

    print("Unban User by Email")

    print("-" \* 50)

    email = get\_input("Enter user email to unban: ")

    # First check if user exists

    user = handler.find\_user\_by\_email(email)

    if not user:

        print("No user found with that email address.")

    elif user.get('status', 'active') == 'active':

        print("User is not currently banned.")

    else:

        if handler.unban\_user\_by\_email(email):

            print(f"User with email '{email}' has been unbanned successfully!")

        else:

            print("Failed to unban user.")

    input("\nPress Enter to continue...")

def check\_user\_status\_by\_email():

    """Check user status by email."""

    clear\_screen()

    print("Check User Status by Email")

    print("-" \* 50)

    email = get\_input("Enter user email to check: ")

    user = handler.find\_user\_by\_email(email)

    if not user:

        print("No user found with that email address.")

    else:

        status = user.get('status', 'active')

        print(f"\nUser Details:")

        print(f"User ID: {user['user\_id']}")

        print(f"Username: {user['username']}")

        print(f"Email: {user['email']}")

        print(f"Status: {status}")

    input("\nPress Enter to continue...")

if \_\_name\_\_ == '\_\_main\_\_':

    login\_menu()

**C. crypto.py**

**(only used for encryption of password)**

import hashlib

import hmac

import os

def generate\_salt():

    """Generate a random salt for password hashing."""

    return os.urandom(16).hex()

def hash\_password(password: str, salt: str = None) -> tuple[str, str]:

    """

    Hash a password using SHA256 with a salt.

    Returns tuple of (hashed\_password, salt).

    """

    if salt is None:

        salt = generate\_salt()

    # Convert password to bytes and combine with salt

    password\_bytes = password.encode('utf-8')

    salt\_bytes = bytes.fromhex(salt)

    # Create hash using HMAC with SHA256

    hashed = hmac.new(salt\_bytes, password\_bytes, hashlib.sha256).hexdigest()

    return hashed, salt

def verify\_password(password: str, stored\_hash: str, salt: str) -> bool:

    """Verify a password against its hash."""

    calculated\_hash, \_ = hash\_password(password, salt)

    return hmac.compare\_digest(calculated\_hash, stored\_hash)

Output

**cli.py**

Welcome to PVC - System Admin Interface

==================================================

1. Login as System Admin

0. Exit

==================================================

Choose an option: 1

Username: sysadmin

Password: admin

Welcome, System Admin!

==================================================

1. Initialize/Reset CSVs

2. Manage Theatre Admins

3. Manage User Accounts

4. Ban/Unban Users

0. Logout

==================================================

Choose an option: 1

CSV files have been initialized/reset.

Press Enter to continue...

Welcome, System Admin!

==================================================

1. Initialize/Reset CSVs

2. Manage Theatre Admins

3. Manage User Accounts

4. Ban/Unban Users

0. Logout

==================================================

Choose an option: 2

Theatre Admin Management

==================================================

1. View All Theatre Admins

2. Create Theatre Admin

3. Modify Theatre Admin

4. Delete Theatre Admin

0. Back to Main Menu

==================================================

Choose an option: 1

Theatre Admins:

--------------------------------------------------------------------------------

Admin ID Username Theatre ID

--------------------------------------------------------------------------------

2 theatre1 1

3 theatre2 2

Press Enter to continue...

Theatre Admin Management

==================================================

1. View All Theatre Admins

2. Create Theatre Admin

3. Modify Theatre Admin

4. Delete Theatre Admin

0. Back to Main Menu

==================================================

Choose an option: 2

Create New Theatre Admin

--------------------------------------------------

Username: PVR

Password: pvr123

Theatre ID: 3

Theatre admin created successfully!

Press Enter to continue...

Theatre Admin Management

==================================================

1. View All Theatre Admins

2. Create Theatre Admin

3. Modify Theatre Admin

4. Delete Theatre Admin

0. Back to Main Menu

==================================================

Choose an option: 3

Theatre Admins:

--------------------------------------------------------------------------------

Admin ID Username Theatre ID

--------------------------------------------------------------------------------

2 theatre1 1

3 theatre2 2

4 PVR 3

Press Enter to continue...3

Enter Admin ID to modify (or 0 to go back): 3

Leave fields empty to keep current values:

New Username (or press Enter to skip): PVR1

New Password (or press Enter to skip):

New Theatre ID (or press Enter to skip):

Theatre admin modified successfully!

Press Enter to continue...

Theatre Admin Management

==================================================

1. View All Theatre Admins

2. Create Theatre Admin

3. Modify Theatre Admin

4. Delete Theatre Admin

0. Back to Main Menu

==================================================

Choose an option: 1

Theatre Admins:

--------------------------------------------------------------------------------

Admin ID Username Theatre ID

--------------------------------------------------------------------------------

2 theatre1 1

3 PVR1 2

4 PVR 3

Press Enter to continue...

Theatre Admin Management

==================================================

1. View All Theatre Admins

2. Create Theatre Admin

3. Modify Theatre Admin

4. Delete Theatre Admin

0. Back to Main Menu

==================================================

Choose an option: 4

Theatre Admins:

--------------------------------------------------------------------------------

Admin ID Username Theatre ID

--------------------------------------------------------------------------------

2 theatre1 1

3 PVR1 2

4 PVR 3

Press Enter to continue...

Enter Admin ID to delete (or 0 to go back): 3

Are you sure you want to delete this admin? (y/N): y

Theatre admin deleted successfully!

Press Enter to continue...

Theatre Admin Management

==================================================

1. View All Theatre Admins

2. Create Theatre Admin

3. Modify Theatre Admin

4. Delete Theatre Admin

0. Back to Main Menu

==================================================

Choose an option: 1

Theatre Admins:

--------------------------------------------------------------------------------

Admin ID Username Theatre ID

--------------------------------------------------------------------------------

2 theatre1 1

4 PVR 3

Press Enter to continue...

Theatre Admin Management

==================================================

1. View All Theatre Admins

2. Create Theatre Admin

3. Modify Theatre Admin

4. Delete Theatre Admin

0. Back to Main Menu

==================================================

Choose an option: 0

Welcome, System Admin!

==================================================

1. Initialize/Reset CSVs

2. Manage Theatre Admins

3. Manage User Accounts

4. Ban/Unban Users

0. Logout

==================================================

Choose an option: 3

User Account Management

==================================================

1. View All Users

2. Modify User Account

3. Delete User Account

0. Back to Main Menu

==================================================

Choose an option: 1

All Users:

------------------------------------------------------------------------------------------

User ID Username Email Status

------------------------------------------------------------------------------------------

1 demo demo@demo.com active

2 john john@cinema.com active

3 alice alice@email.com active

Press Enter to continue...

User Account Management

==================================================

1. View All Users

2. Modify User Account

3. Delete User Account

0. Back to Main Menu

==================================================

Choose an option: 2

All Users:

------------------------------------------------------------------------------------------

User ID Username Email Status

------------------------------------------------------------------------------------------

1 demo demo@demo.com active

2 john john@cinema.com active

3 alice alice@email.com active

Press Enter to continue...1

Enter User ID to modify (or 0 to go back): 1

Leave fields empty to keep current values:

New Username (or press Enter to skip): main

New Password (or press Enter to skip):

New Email (or press Enter to skip):

User account modified successfully!

Press Enter to continue...

User Account Management

==================================================

1. View All Users

2. Modify User Account

3. Delete User Account

0. Back to Main Menu

==================================================

Choose an option: 1

All Users:

------------------------------------------------------------------------------------------

User ID Username Email Status

------------------------------------------------------------------------------------------

1 main demo@demo.com active

2 john john@cinema.com active

3 alice alice@email.com active

Press Enter to continue...

User Account Management

==================================================

1. View All Users

2. Modify User Account

3. Delete User Account

0. Back to Main Menu

==================================================

Choose an option: 3

All Users:

------------------------------------------------------------------------------------------

User ID Username Email Status

------------------------------------------------------------------------------------------

1 main demo@demo.com active

2 john john@cinema.com active

3 alice alice@email.com active

Press Enter to continue...

Enter User ID to delete (or 0 to go back): 2

Are you sure you want to delete this user and all their bookings? (y/N):

Deletion cancelled.

Press Enter to continue...

User Account Management

==================================================

1. View All Users

2. Modify User Account

3. Delete User Account

0. Back to Main Menu

==================================================

Choose an option: 3

All Users:

------------------------------------------------------------------------------------------

User ID Username Email Status

------------------------------------------------------------------------------------------

1 main demo@demo.com active

2 john john@cinema.com active

3 alice alice@email.com active

Press Enter to continue...2

Enter User ID to delete (or 0 to go back): 2

Are you sure you want to delete this user and all their bookings? (y/N): y

User account and all bookings deleted successfully!

Press Enter to continue...

User Account Management

==================================================

1. View All Users

2. Modify User Account

3. Delete User Account

0. Back to Main Menu

==================================================

Choose an option: 1

All Users:

------------------------------------------------------------------------------------------

User ID Username Email Status

------------------------------------------------------------------------------------------

1 main demo@demo.com active

3 alice alice@email.com active

Press Enter to continue...

User Account Management

==================================================

1. View All Users

2. Modify User Account

3. Delete User Account

0. Back to Main Menu

==================================================

Choose an option: 0

Welcome, System Admin!

==================================================

1. Initialize/Reset CSVs

2. Manage Theatre Admins

3. Manage User Accounts

4. Ban/Unban Users

0. Logout

==================================================

Choose an option: 4

User Ban Management

==================================================

1. View Banned Users

2. Ban User by Email

3. Unban User by Email

4. Check User Status by Email

0. Back to Main Menu

==================================================

Choose an option: 1

No banned users found.

Press Enter to continue...

User Ban Management

==================================================

1. View Banned Users

2. Ban User by Email

3. Unban User by Email

4. Check User Status by Email

0. Back to Main Menu

==================================================

Choose an option: 2

Ban User by Email

--------------------------------------------------

Enter user email to ban:

No user found with that email address.

Press Enter to continue...

User Ban Management

==================================================

1. View Banned Users

2. Ban User by Email

3. Unban User by Email

4. Check User Status by Email

0. Back to Main Menu

==================================================

Choose an option: 4

Check User Status by Email

--------------------------------------------------

Enter user email to check: demo@demo.com

User Details:

User ID: 1

Username: main

Email: demo@demo.com

Status: active

Press Enter to continue...

User Ban Management

==================================================

1. View Banned Users

2. Ban User by Email

3. Unban User by Email

4. Check User Status by Email

0. Back to Main Menu

==================================================

Choose an option: 0

Welcome, System Admin!

==================================================

1. Initialize/Reset CSVs

2. Manage Theatre Admins

3. Manage User Accounts

4. Ban/Unban Users

0. Logout

==================================================

Choose an option: 0

Welcome to PVC - System Admin Interface

==================================================

1. Login as System Admin

0. Exit

==================================================

Choose an option: 0

Goodbye!

**gui.py**

**User Panel**

A screenshot of a computer

Description automatically generatedA screen shot of a movie

Description automatically generatedA screen shot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Theatre Admin Panel**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Bibliography

- Python documentation: https://docs.python.org/3/  
- Streamlit documentation: https://docs.streamlit.io/  
- Pandas documentation: https://pandas.pydata.org/docs/  
- W3Schools: https://www.w3schools.com/python/  
- GeeksforGeeks: https://www.geeksforgeeks.org/python-programming-language

- Preeti Aurora – Computer science with python