class Project:

Movie Ticketing System

An Object-Oriented Python Implementation

Project Overview

This project simulates a real-world **Movie Ticketing System** using Python. It demonstrates key software engineering concepts including `Classes`, `Inheritance`, and `Encapsulation` to manage theatre inventory and user bookings dynamically.

// System Stack

Backend Logic

Python 3.x based core utilizing OOP principles for modularity.

Data Model

Structured List-based storage for transient data persistence.

CLI Interface

Interactive command-line loop for user input and booking.

// Architecture Design

`class Show:`

The data model representing a single screening entity.

- **State:** `title`, `time`, `seats_available`
- **Behavior:** `book(n)`, `cancel(n)`

`class Theatre:`

The controller managing multiple Show instances.

- **State:** `List[Show]` inventory
- **Behavior:** `find_show()`, `dispatch_booking()`

// The Controller

Centralized Logic

The `Theatre` class acts as the system's API. It abstracts the complexity of individual show management from the user.

When a request is made, it performs a linear search across the inventory to match the movie title (case-insensitive), ensuring a robust user experience even with imperfect inputs.

// Booking Algorithm

1. Input

User inputs movie title and quantity via CLI. Input is sanitized using `.strip().lower()`.

2. Validation

System checks: `if requested ⩽ current.available_seats`. If false, returns error.

3. Commit

If valid, state is updated: `booked += n`. Transaction is confirmed to user.

// Error Handling

Defensive Programming

To prevent runtime crashes, the system implements robust exception handling:

```
> try:
    count = int(input())
  except ValueError:
    print("Invalid Integer")
```

This ensures that if a user accidentally types text instead of a number, the program catches the error gracefully and prompts again.

// Live Inventory

SHOW ID	TITLE	DURATION	CAPACITY	STATUS
#001	Sholay	3h 48m	160	Available
#002	DDLJ	3h 26m	160	Filling Fast
#003	Border	2h 17m	160	Sold Out
#004	Coolie	3h 40m	160	Available

// Booking Metrics

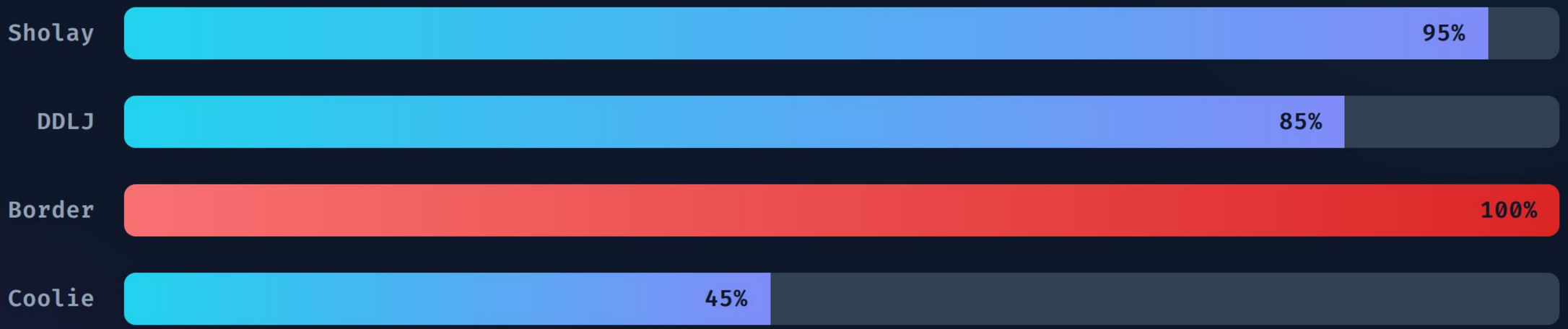

Figure 1: Real-time occupancy rates derived from booking simulation.

// Future Roadmap v2.0

Persistent Storage

Implement **SQLite** or **JSON** serialization to save booking state across sessions, moving away from in-memory volatile storage.

Dynamic Pricing Module

Add a pricing attribute to the `Show` class to calculate revenue: `total_cost = seats * price_per_ticket`.

"Talk is cheap.
Show me the code."

— Linus Torvalds

Q & A

System.exit(0)

// Image Sources

https://static.vecteezy.com/system/resources/previews/054/088/420/non_2x/computer-monitor-displaying-python-programming-language-code-on-blue-background-png.png

Source: www.vecteezy.com

https://media.istockphoto.com/id/1873575181/photo/real-film-for-cinema-background.jpg?s=612×612&w=0&k=20&c=peigWu_xj_NIgdK9ZuMWSgIFuEKHG432O37u03U8xHs=

Source: www.istockphoto.com

https://img.freepik.com/premium-psd/3d-render-cinema-ticket-popup-from-smartphone-with-booking-tickets-online_252008-535.jpg

Source: www.freepik.com

https://en.pimg.jp/104/919/945/1/104919945.jpg

Source: www.pixtastock.com

https://static.vecteezy.com/system/resources/previews/008/020/652/non_2x/abstract-technology-connection-data-concept-circuit-lines-board-with-nodes-and-geometric-elements-lighting-effect-on-blue-background-vector.jpg

Source: www.vecteezy.com