Movie Ticketing Kiosk System

Introduction

I will be talking about the movie ticketing kiosk system. It's a self-service solution that aims to simplify the movie ticket purchasing process for customers visiting theaters. Users can then browse the movies available, select their showtime, and pick their seats right on an interactive touch screen. The system must provide a user-friendly interface capable of handling confirmation of ticket details, selection verification, and payment transfers through several options, such as card payment (credit/debit) and digital wallets. Once the payment has been taken, the kiosk issues a paper ticket or a QR code to be used to gain access. The kiosk also lets you look up showtimes, modify your bookings, and get help if you need it.

3.1 External Interface Requirements

User Requirements:

Select a Movie: Users must have the ability to view a list of available movies.

Choose Showtimes: Users need to see and choose the showtime for their chosen movie.

Interactive Seat Selection: Users should be able to select their preferred seats by clicking on the available seat in the seating chart.

Payment Options: The system should accept different modes of payment, including credit/debit cards and e-wallets.

Ticket Generation: After finishing payment, users must receive a printed ticket or a digital QR code for theater entry.

Multi-Language Support: The system should have the ability to support multi-language access. Users should have access to help through the kiosk if they experience difficulties.

Change Bookings: Before confirming the payment, users must be allowed to cancel/change bookings.

System Requirements:

Database Management: The system will need to have a database that will contain movie information and its screening times, ticket availability, and booking details.

Payment Processing Integration: The mandatory integration of the kiosk with external payment gateways for transaction processing.

Real-Time Updates: The system should provide real-time updates to keep track of all the showtimes and seat availability like a new listing and also prevent double booking.

Ticket Printing: The system should have a ticket printing machine for physical tickets and also apply QR codes to tickets for digital ticket generation.

Kiosk Interaction: A touch-based interface to be used by users operating the kiosk.

Performance: The responses from the application should be within 3 seconds for individual requests (i.e., seat selection and payment).

Reliability: Your system should be as free of downtime as possible, and if the network fails, have a backup to rely on.

Security: User sensitive information (such as payment information) will be securely stored as per industry security standards.

Accessibility: Kiosks must be accessible, even for visually impaired users.

Scale: The system should scale to handle many requests during peak hours of the day (weekends, holiday seasons).

3.2 Functional Requirements

3.2.1.1 Introduction

The system will allow use case 3 (i.e., browsing movies).

3.2.1.2 Inputs

Movie catalog data (title, genre, showtimes, description, poster).

Direct user input through touch interface to search and select movies.

3.2.1.3 Processing

Display movie listings.

Select movies from indicators (for example, genre, showtime).

Select a movie and show more details about the movie.

3.2.1.4 Outputs

Movie catalog display.

Detailed movie information display.

3.2.1.5 Error Handling

Show an error message if the movie catalog cannot load.

Some grid or message to show if no movies matched the user filter criteria.

3.2.2 Choose Showtimes

3.2.2.1 Introduction

The system shall provide users to book a showtime of a selected movie.

3.2.2.2 Inputs

Movie showtime information (time, date, theater).

User input through the touch interface for a showtime selection.

3.2.2.3 Processing

Show available showtimes for selected movie.

Adjust seat availability on the fly.

3.2.2.4 Outputs

Confirmation for a showtime selection.

Showing available seats for a selected showtime.

3.2.2.5 Error Handling

Show message if selected showtime is fully booked.

Show a message when selected showtime is no longer available.

3.2.3 Interactive Seat Selection

3.2.3.1 Introduction

Users shall be able to choose seats with the help of an interactive seating chart.

3.2.3.2 Inputs

Seating chart data (layout, available/unavailable seats).

Select seat input through touch interface.

3.2.3.3 Processing

Show a seating map with available seats in real time.

Update availability for seats after selection.

Users should be able to select/deselect seats.

3.2.3.4 Outputs

Display selected seats.

Seat selection record.

3.2.3.5 Error Handling

Certain seats may not be available for selection.

Show alert if selected seats are not available before confirmation.

3.2.4 Payment Options

3.2.4.1 Introduction

The system must accept multiple payment options.

3.2.4.2 Inputs

Type of payment (credit/debit card, e-wallet).

Payment information (card number, expiry, etc.).

3.2.4.3 Processing

Payment to the chosen payment gateway.

Verify payment.

3.2.4.4 Outputs

Payment confirmation.

Transaction receipt.

3.2.4.5 Error Handling

Show message for failed payment.

Accept payment through alternate methods.

3.2.5 Ticket Generation

3.2.5.1 Introduction

The system will create tickets in printed or electronic form.

3.2.5.2 Inputs

Payment confirmation.

Reservation information (movie, show time, seats).

3.2.5.3 Processing

Create a unique ticket with QR code.

Print a physical ticket.

3.2.5.4 Outputs

Printed ticket.

QR code digital tickets.

3.2.5.5 Error Handling

Show message on ticket generation failure.

Re-send or re-print the ticket.

3.2.6 Multiple Languages

3.2.6.1 Introduction

The system should be multilingual (i18n).

3.2.6.2 Inputs

User language selection.

3.2.6.3 Processing

Translate the interface and content according to the user selection.

3.2.6.4 Outputs

Show the system in the chosen language.

3.2.6.5 Error Handling

Don't do anything if the selected language is not supported.

Use a primary language by default.

3.2.7 Help and Support

3.2.7.1 Introduction

The system will assist and support users.

3.2.7.2 Inputs

User request for help.

3.2.7.3 Processing

Display help information.

Get help contact information.

3.2.7.4 Outputs

Help information display.

Showing contact information.

3.2.7.5 Error Handling

Fallback to a message if could not get help info.

3.2.8 Change Bookings

3.2.8.1 Introduction

Users may change or cancel any bookings made before payment.

3.2.8.2 Inputs

Booking details.

Request by the user to update or cancel.

3.2.8.3 Processing

Handle changes or cancellations to bookings.

Update seat availability.

3.2.8.4 Outputs

Here are a few examples of these transactional messages:

Changes or cancellations in bookings.

3.2.8.5 Error Handling

Show a message if a booking change or cancellation fails.

3.3 Use Cases

Purchase Movie Tickets 3.3.1

Actors: Moviegoer

Description: Moviegoer selects a movie, selects showtime, selects seats, and makes a payment to book tickets.

3.3.2 Check Movie Showtimes

Actor: Moviegoer

Description: The moviegoer views the movies that are available and their showtimes.

3.3.3 Seek Assistance

Actor: Moviegoer

Description: The moviegoer asks for help & the kiosk support assists.

3.4 Classes / Objects

3.4.1 Movie

3.4.1.1 Attributes: Title, Genre, Showtimes, Availability, Description, Poster

3.4.1.2 Functions: GetMovieDetails(), GetShowtimes(), CheckAvailability().

3.4.2 Ticket

3.4.2.1 Attributes: Movie, Showtime, Seats, QR code, Price, TransactionID

3.4.2.2 Functions: GenerateTicket(), PrintTicket(), SendDigitalTicket().

3.4.3 Payment

3.4.3.1 Attributes: Amount, PaymentMethod, TransactionID, Status, PaymentGateway

3.4.3.2 Functions: ProcessPayment(), VerifyPayment(), GenerateReceipt().

3.5 Non-Functional Requirements

3.5.1 Performance

At least 95% of the transactions will be completed in less than 3 seconds.

The system shall support a maximum of 100 concurrent users in peak times.

3.5.2 Reliability

The system must not have any downtime longer than 1 minute a day.

You shall have a backup and recovery plan for network failures.

3.5.3 Availability

The system will be reasonably available 24 hours a day, 7 days a week, and no less than 50 weeks a year, excluding scheduled maintenance periods.

3.5.4 Security

Payment information of users should be encrypted and stored securely.

The system shall comply with industry security standards (e.g., PCI DSS).

3.5.5 Maintainability

The system will be motivated for simple updates and hand impression.

The code should be documented and modular.

3.5.6 Portability

There are different options for kiosk hardware and operating systems available in the market.

3.6 Inverse Requirements

The system will not permit double booking of seats.

The payment information of the user must not be stored in plain text in the system.

3.7 Design Constraints

For example, the system should conform to accessibility standards for users with navigation issues.

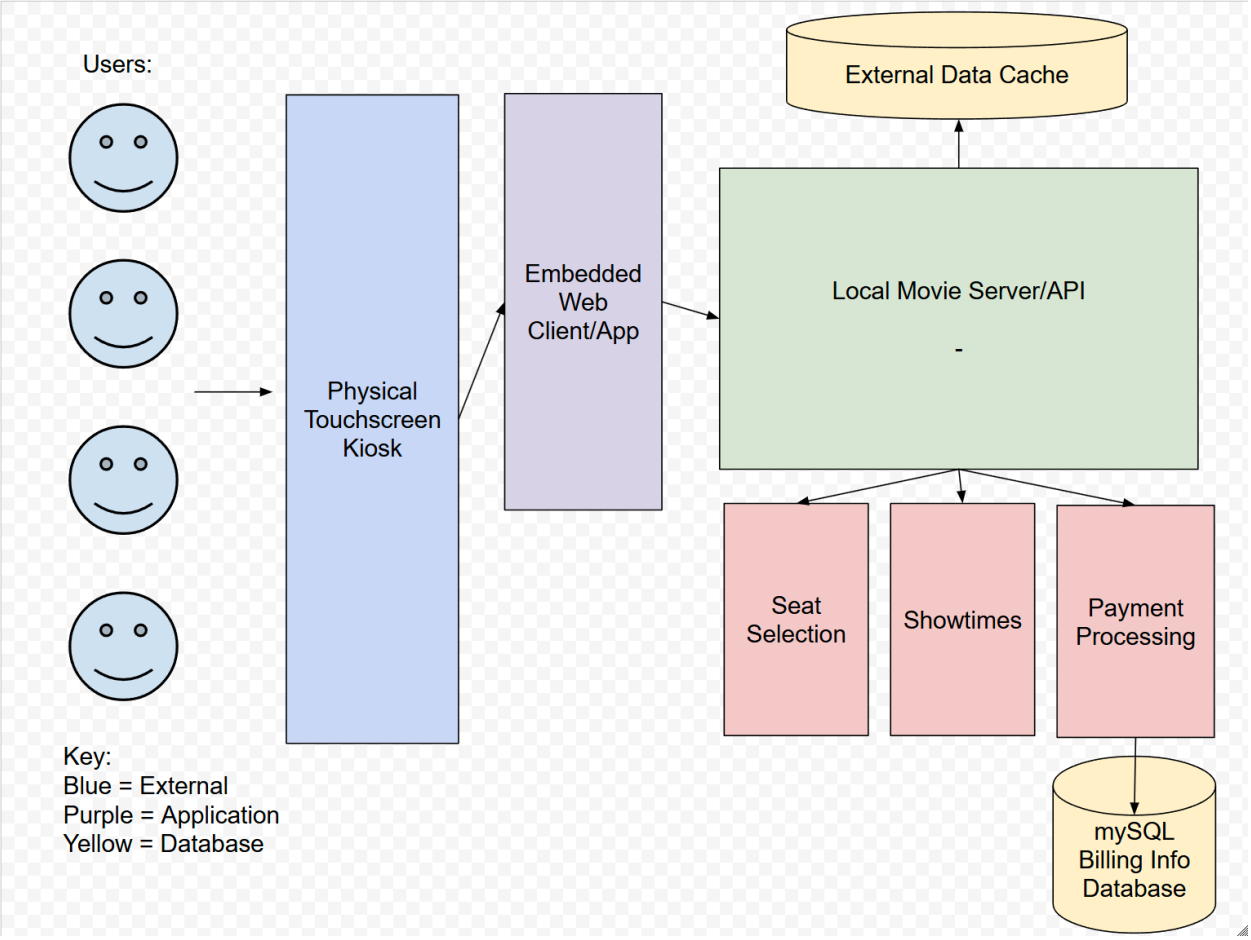
Integrate into the existing theater infrastructure.

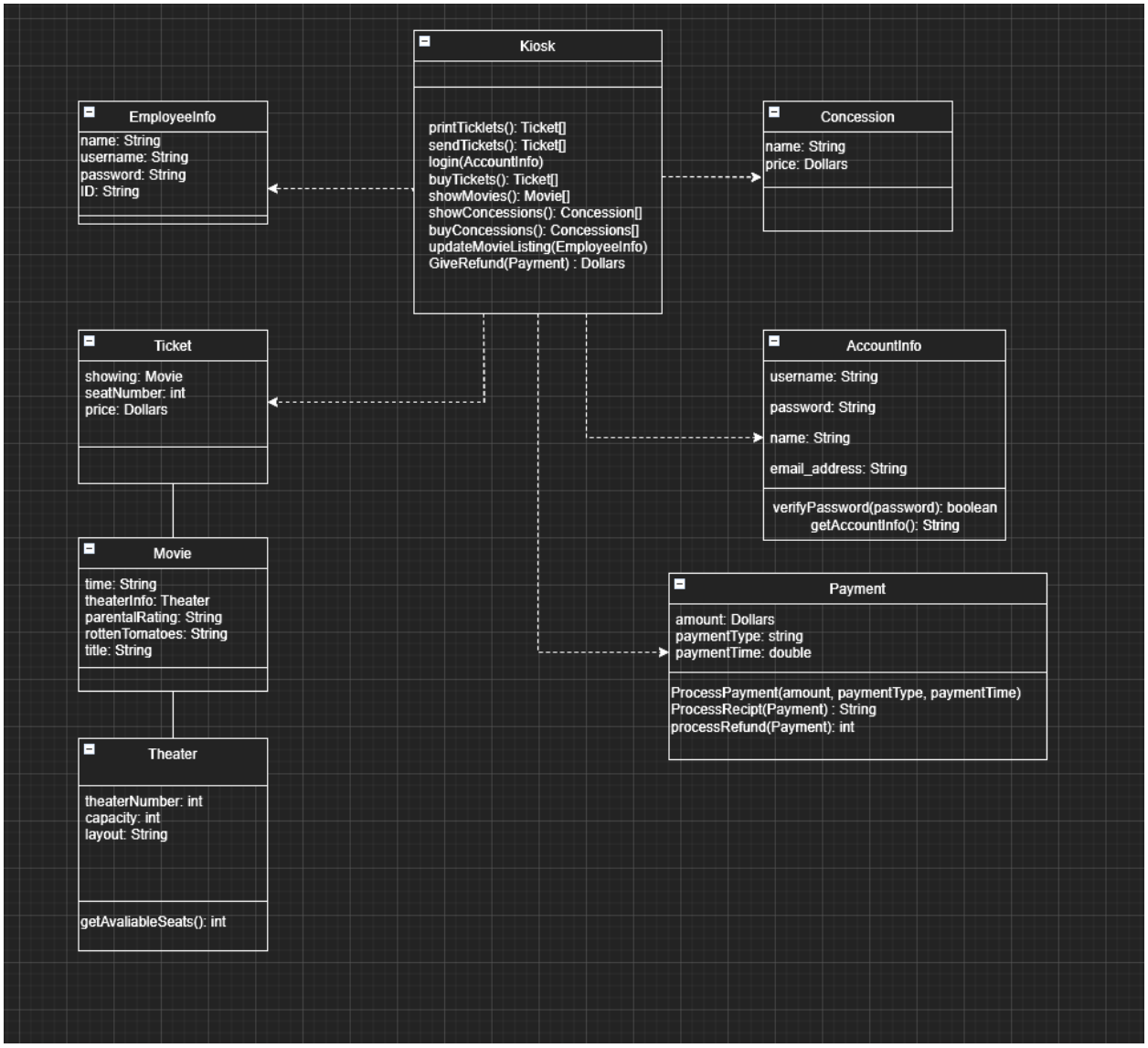
3.8 Logical Database Requirements

The duration in weeks will vary according to the movie show.

At least one year to keep data.

Make sure the database maintains data integrity and prevents data from being lost.


Users:
External Data Cache
Physical Touchscreen Kiosk
Embedded Web Client/App
Local Movie Server/API
-
Seat Selection
Showtimes
Payment Processing
mySQL Billing Info Database
Key:
Blue = External
Purple = Application
Yellow = Database



Descriptions:

Classes:

EmployeeInfo

Description: This is an employee of the theater.

Attributes:

- name: String - Employee full name

- username: String - The employee login username.

- example: String - Defines the login password for the employee.

- ID: String - The unique identifier of the employee.

Operations:

(No operations are mentioned in the diagram)

Kiosk

Description: The main class for the movie theater kiosk. It handles everything related to ticket sales, concessions, and movie information.

Note: (None specifically mentioned in the diagram)

Operations:

- printTickets(): Ticket[] - This function will generate and return an array of Ticket objects.

- sendTickets(): Ticket[] - Sends the generated tickets (to a printer or digitally).

- login(Accountinfo): Boolean (Implicit) - Input: Accountinfo Object | Task: Try to log in through the instance, returns boolean.

- buyTickets(): Ticket[] - Manages ticket purchasing and returns an array of Ticket objects.

- showMovies(): Movie[] - Returns a stream of Movie objects.

- showConcessions(): Concession[] - Fetches and returns a set of available Concession objects.

- buyConcessions(): Concession[] - This method is responsible for purchasing concessions and returns an array of Concession objects.

- updateMovieListing(EmployeeInfo): Boolean (Implicitly) - An EmployeeInfo object allows updating the movie listings. Returns true or false depending if successful or not.
- GiveRefund(Payment): Dollars - Behind the scenes, this function processes a refund for the given Payment object and returns the amount refunded.

Ticket

Description: A movie ticket.

Attributes:

- showing: Movie - The Movie object associated with the ticket.
- seatNumber: int - The ticket's assigned seat number.
- type: String - The custom type of the ticket e.g. Premium, Economy.

Movie

Description: Represents a movie currently playing in the theater.

Attributes:

- time: String - The showtime of the movie.
- theaterInfo: Theater - The Theater object at which the movie is showing.
- parentalRating: String - The parental rating of the movie (e.g., PG-13, PG, R).
- rottenTomatoes: String - The Rotten Tomatoes score or rating.
- title: String - The title of the movie.

Theater

Description: A theater (screen) for the cinema complex.

Attributes:

- theaterNumber: int - The unique number of the theater.

- capacity: int - The total seating capacity of the theater.

- layout: String - A description of the theater's seating layout.

Operations:

- GetAvailableSeats(): int - Determines the number of available seats in the theater.

Concession

Description: Represents a concession item sold at the theater.

Attributes:

- name: String - The name of the concession item.

- price: Dollars - Price of the item.

Accountinfo

Description: User account details for login.

Attributes:

- username: String - The username that will be used to log in.

- password: String - User login password.

- name: String - The full name of the user.

- email_address: String - User email.

Operations:

- verifyPassword(password): boolean - Verifies the password.

- getAccountinfo(): String - A string representation of the account information.

Payment

Description: Corresponds to a payment that a customer has made.

Attributes:

- amount: Dollars - The payment amount.

- paymentType: String - Type of payment (credit card, cash etc.)

- paymentTime: double - The timestamp of the payment.

Operations:

- ProcessPayment(amount, paymentType, paymentTime): Boolean - This method processes the payment with the provided values. Returns a boolean to indicate success or failure.

- ProcessReceipt(Payment): String - Produces a receipt string of the payment.

- processRefund(Payment): int - Processes a refund for the given payment.

Data Types:

- String: Used for text data.

- int: For integer numbers.

- double: Used for floating-point numbers (like timestamps).

- boolean: A type of value that can be true or false.

- Dollars: A custom type to denote currency values. This is likely a custom class or data type to represent money.

- Ticket[], Movie[], Concession[]: Array or list of respective objects.

SDS: Test Plan

TC-U-001	Unit	Verify Movie Search Function	Movie database loaded.	1. Enter a movie title in the search field. 2. Click "Search."	List of matching movies displayed.	
TC-U-002	Unit	Verify User Login Function	User account exists.	1. Enter valid username and password. 2. Click "Login."	User logged in successfully.	
TC-F-001	Functional	Rent a Movie	User logged in, movie available.	1. Select a movie. 2. Click "Rent." 3. Confirm rental.	Movie rented, rental confirmation displayed.	
TC-F-002	Functional	Return a Movie	User has a rented movie.	1. Select the rented movie. 2. Click "Return."	Movie returned, confirmation displayed.	
TC-S-001	System	Simultaneous User Access	Multiple users accessing the system.	1. Multiple users perform movie searches and rentals concurrently.	System responds without errors or delays.	
TC-S-002	System	Database Recovery	Simulate database failure.	1. Simulate a database crash. 2. Restart the system.	System recovers, data is intact.	
TC-U-003	Unit	Verify Movie details display	Movie is selected	1. Select a movie from the search results. 2. click "view details"	Correct movie details are displayed.	
TC-U-004	Unit	Verify password encryption	user creates account	1. user creates account. 2. check database.	password is encrypted in database.	
TC-F-003	Functional	Verify late return fee calculation	Movie is returned late.	1. rent a movie. 2. return the movie after due date.	Late fee is calculated and displayed.	

TC-F-004	Functional	Verify payment processing	user rents movie.	1. user selects a movie. 2. user enters payment info. 3. user confirms payment.	payment is processed successfully.	