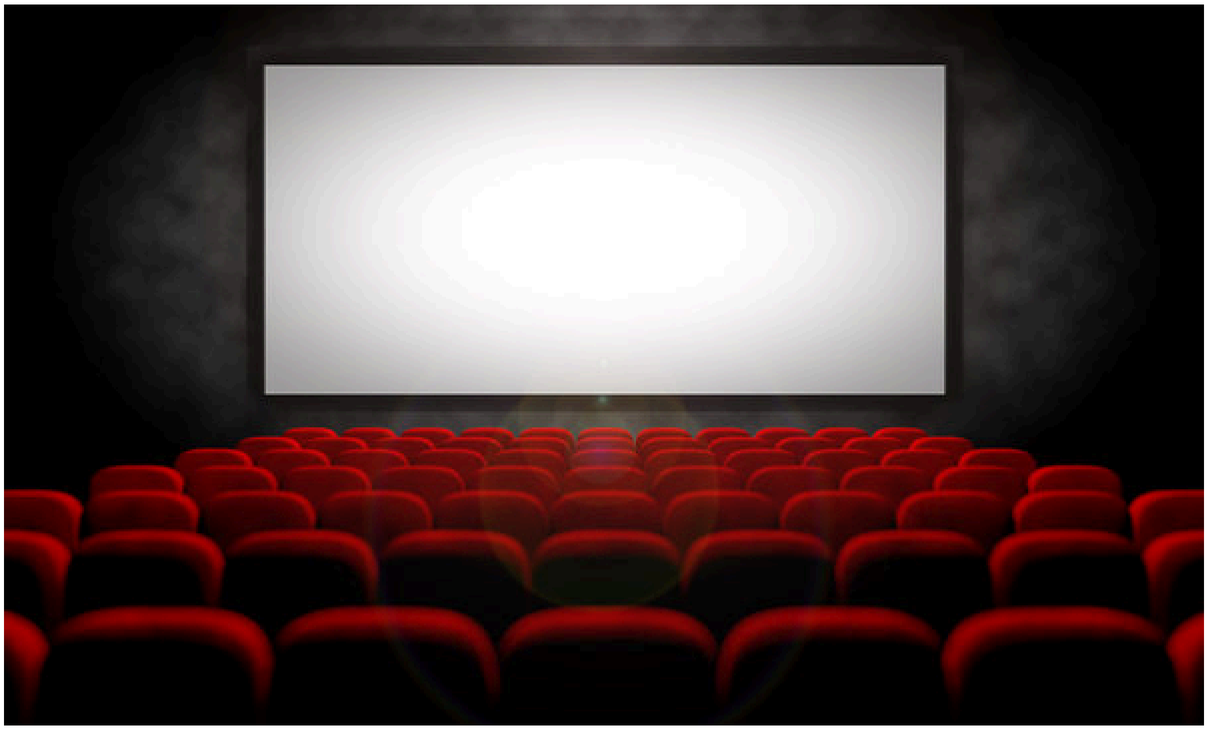
T.I.C.K.E.T. - Theater Integrated Comprehensive Kiosk and Entry Tool

Group 1 Software Design Specification

Riley Mathews, Brandon Mahdavi, Justin Nam



Git repo

06.02.2024

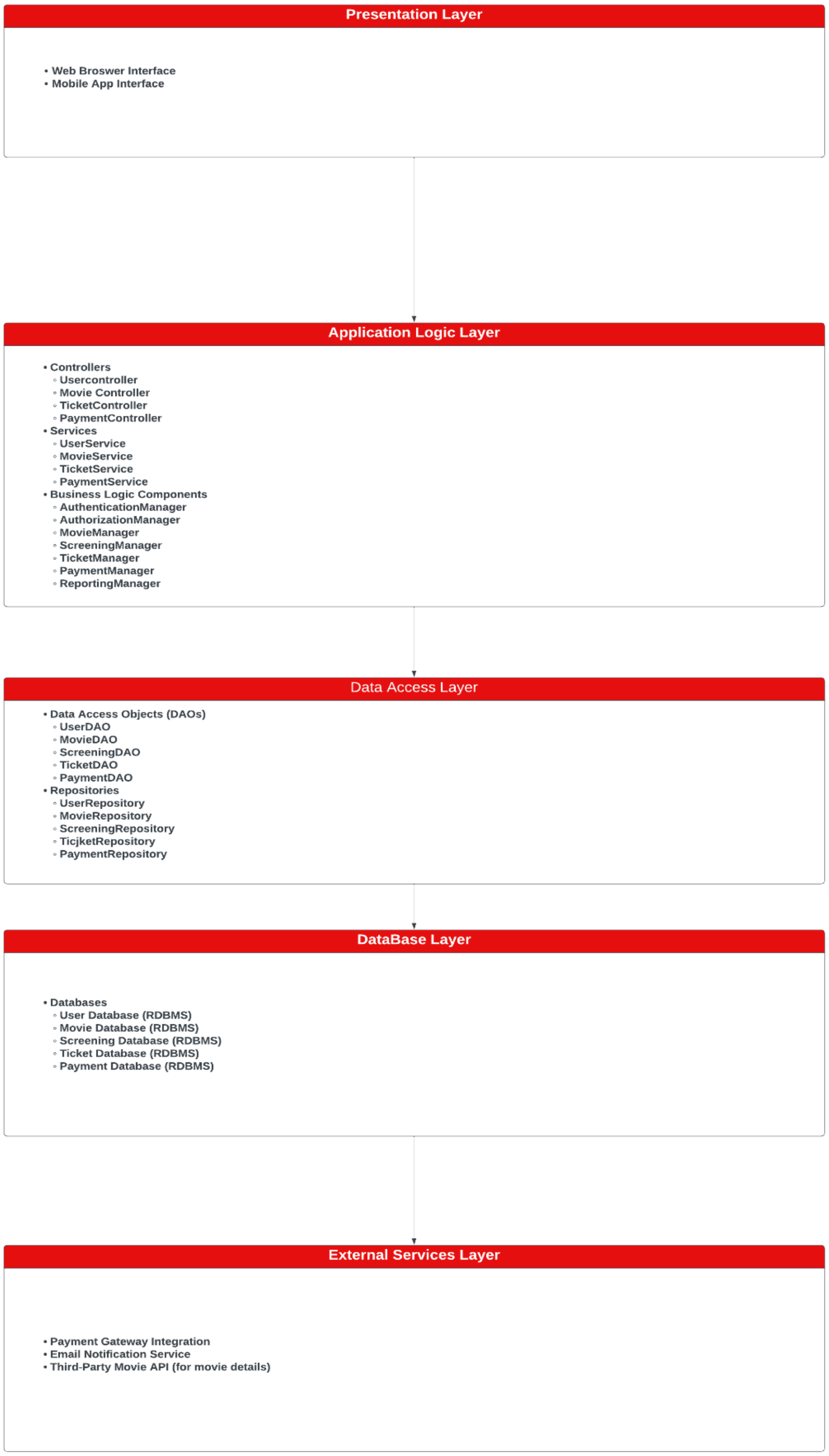
CS-250-Summer 2024

System Description

- The T.I.C.K.E.T System (Theater Integrated Comprehensive Kiosk and Entry Tool) is designed to facilitate online ticket booking for movie theaters. Users can browse movies, check showtimes, select seats, and purchase tickets. The system will also support theater staff in managing screenings, handling bookings, and generating sales reports.

Software Architecture Overview

- Architectural diagram of all major components status: done
    - Talked about this in the 05/28 lecture
    - Maybe we used layered diagram using the lucid software to design it?


Presentation Layer
• Web Broswer Interface
• Mobile App Interface
Application Logic Layer
• Controllers
◦ Usercontroller
◦ Movie Controller
◦ TicketController
◦ PaymentController
• Services
◦ UserService
◦ MovieService
◦ TicketService
◦ PaymentService
• Business Logic Components
◦ AuthenticationManager
◦ AuthorizationManager
◦ MovieManager
◦ ScreeningManager
◦ TicketManager
◦ PaymentManager
◦ ReportingManager
Data Access Layer
• Data Access Objects (DAOs)
◦ UserDAO
◦ MovieDAO
◦ ScreeningDAO
◦ TicketDAO
◦ PaymentDAO
• Repositories
◦ UserRepository
◦ MovieRepository
◦ ScreeningRepository
◦ TicjketRepository
◦ PaymentRepository
DataBase Layer
• Databases
◦ User Database (RDBMS)
◦ Movie Database (RDBMS)
◦ Screening Database (RDBMS)
◦ Ticket Database (RDBMS)
◦ Payment Database (RDBMS)
External Services Layer
• Payment Gateway Integration
• Email Notification Service
• Third-Party Movie API (for movie details)

Architectural diagram still needs to be created in the process of doing research on it

- UML Class Diagram status: done
- Description of classes status: done

Classes:

- Abstract user
  - Represents a generic user in the system.
  - Contains common attributes and operations for all types of users, such as registration, login, and profile updates.
- Admin
  - Represents an administrator in the system.
  - Has privileges to create and update movie times.
  - Inherits attributes from the Abstract User class but may have additional admin-specific functionalities.
- Employee
  - Represents an employee in the system, likely working at a movie theater.
  - Has attributes like name, password, and employee ID.
  - Performs tasks related to managing screenings, handling bookings, etc.
- Ticket
  - Represents a ticket for a movie screening.
  - Contains attributes like purchase number, time, date, seat number, and price.
  - Has operations for booking tickets, canceling tickets, and checking if a ticket has been scanned.
- Movie viewer
  - Represents a user who views movies and purchases tickets.
  - Inherits attributes from the Abstract User class.
  - Has specific functionalities such as login, registration, and purchasing tickets.
- Payment
  - Represents a payment made for a ticket.
  - Includes attributes such as payment ID, ticket details, amount, and payment method.
  - Provides operations for processing payments and refunding payments.

Individual Classes of Systems:

- Abstract user
  - Attributes

        - userID(int): Integer representing the user’s unique ID.
        - username (String): String representing the user’s username.
        - password (String):String representing the user’s password.
        - email (String): String representing the user’s email address.
    - Operations:
        - register(username: String, password: String, email: String): bool
        - login(username: String, password: String): bool
        - updateProfile(email: String): bool

- Admin
    - Attributes
        - name (String): String representing the admin's name.
        - password (String): String representing the admin's password.
        - userID (String): String representing the admin's unique ID.
    - Functions
        - createMovieTime (movieID, time)
        - updateMovieTime (movieID, time)
- Employee
    - Attributes
        - name (String): String representing the employee's name.
        - password (String): String representing the employee's password.
        - emplID (int): Integer representing the employee's unique ID.
- Movie viewer
    - Attributes
        - name (String): String representing the movie viewer's name.
        - email (String): String representing the movie viewer's email address.
        - password (String): String representing the movie viewer's password.
        - userID (int): Integer representing the movie viewer's unique ID.
    - Functions
        - login (bool)
        - register (void)

        - purchaseTicket (void)
- Ticket
    - Attributes
        - purchaseNumber (ticketID: int): Integer representing the ticket's purchase number or ID.
        - Time (screening: Screening): Object representing the screening time of the ticket.
        - Date (screening: Screening): Object representing the screening date of the ticket.
        - Seat (seatNumber: String): String representing the seat number associated with the ticket.
        - Price (price: float): Float representing the price of the ticket.
    - Functions
        - checkIfScanned (bool)
        - bookTicket(screeningID: int, seatNumber: String): Ticket
        - cancelTicket(ticketID: int): bool
- Payment
    - Attributes:
        - paymentID (int):Integer representing the payment's unique ID.
        - ticket (Ticket): Object representing the ticket associated with the payment.
        - amount (float): Float representing the amount paid.
        - paymentMethod (String): String representing the payment method used.
    - Operations:
        - processPayment(ticketID: int, paymentMethod: String): Payment
        - refundPayment(paymentID: int): bool

Description of attributes(work on these things above) status: done

Description of operations status: done

Push to git: status: done

Functions Available to Each Class

User Class**:**

- **register:** Registers a new user with a username, password, and email. Returns true if registration is successful.
- **login:** Logs in a user with the given username and password. Returns true if login is successful.
- **updateProfile:** Updates the user's email. Returns true if update is successful.

Movie Class**:**

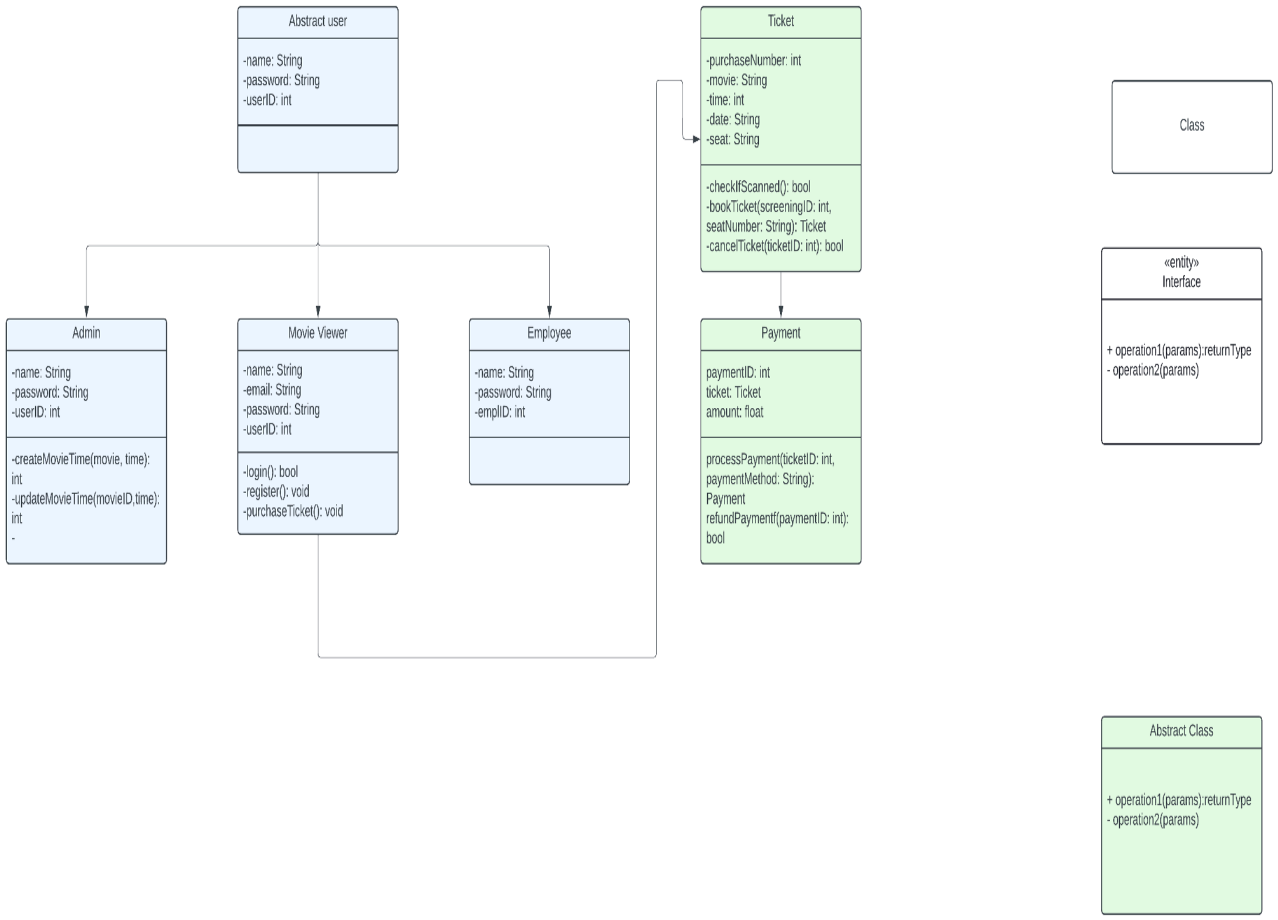
- **getMovieDetails:** Retrieves details of a movie by its ID.
- **searchMovies:** Searches for movies by title and returns a list of matching movies.

Ticket Class**:**

- **bookTicket:** Books a ticket for a specific screening and seat number. Returns the booked ticket.
- **cancelTicket:** Cancels a ticket by its ID. Returns true if cancellation is successful.

Payment Class**:**

- **processPayment:** Processes payment for a ticket using a specified payment method. Returns the payment details.
- **refundPayment:** Refunds a payment by its ID. Returns true if refund is successful.


Abstract user
-name: String
-password: String
-userID: int
Ticket
-purchaseNumber: int
-movie: String
-time: int
-date: String
-seat: String
-checkIfScanned(): bool
-bookTicket(screeningID: int, seatNumber: String): Ticket
-cancelTicket(ticketID: int): bool
Class
«entity»
Interface
+ operation1(params):returnType
- operation2(params)
Admin
-name: String
-password: String
-userID: int
-createMovieTime(movie, time): int
-updateMovieTime(movieID,time): int
-
Movie Viewer
-name: String
-email: String
-password: String
-userID: int
-login(): bool
-register(): void
-purchaseTicket(): void
Employee
-name: String
-password: String
-emplID: int
Payment
paymentID: int
ticket: Ticket
amount: float
processPayment(ticketID: int, paymentMethod: String): Payment
refundPaymentf(paymentID: int): bool
Abstract Class
+ operation1(params):returnType
- operation2(params)

Development plan and timeline

- Partitioning of tasks
- Team member responsibilities

Partitioning of Tasks**:**

1. Frontend Development**:**
    - Justin Nam: Design and implement the user interface for browsing movies and selecting screenings. Develop the user interface for ticket booking and payment processing.
2. Backend Development**:**
    - Riley Mathews: Implement the backend services for user management and movie information.
    - Brandon Mahdavi: Implement the backend services for screening management and ticket booking.

Team Member Responsibilities**:**

- Justin Nam**:**
    - Task: User Interface for Browsing Movies
    - Responsibility: Design and implement the interface for users to browse movies and view details.
- Justin Nam**:**
    - Task: User Interface for Booking and Payment
    - Responsibility: Develop the interface for users to select seats, book tickets, and make payments.
- Riley Mathews**:**
    - Task: User and Movie Management Services
    - Responsibility: Implement services for user registration, login, and movie details retrieval.
- Brandon Mahdavi**:**
    - Task: Screening and Ticket Management Services

    - Responsibility: Develop services for managing screenings, booking tickets, and processing payments.

Timeline**:**

- Week 1-2**:**
    - Frontend: Initial design and setup of user interfaces (.
    - Backend: Setup of database and basic user and movie management services.
- Week 3-4**:**
    - Frontend: Development of ticket booking and payment interfaces.
    - Backend: Implementation of screening management and ticket booking services.
- Week 5-6**:**
    - Frontend: Integration testing of user interfaces.
    - Backend: Testing and refinement of backend services.
- Week 7-8**:**
    - Final integration and system testing.
    - Deployment and preparation for presentation.