<CheepTIX.com>

Software Design Specification

<1.0>

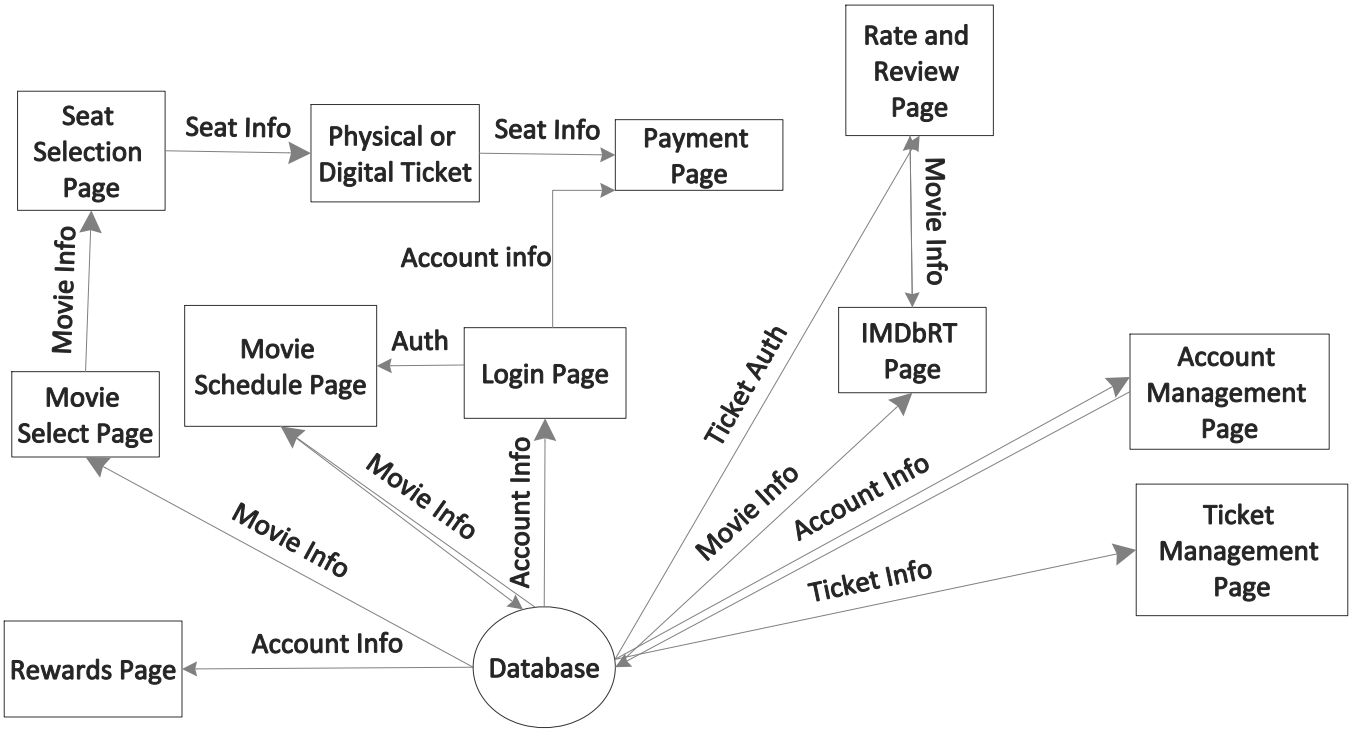
<10/04/2023>

<Group 11>
<Kalanikekai Tran>
<Elijah Agustin>
<Isaac Reveles>
<Jonathan Van>

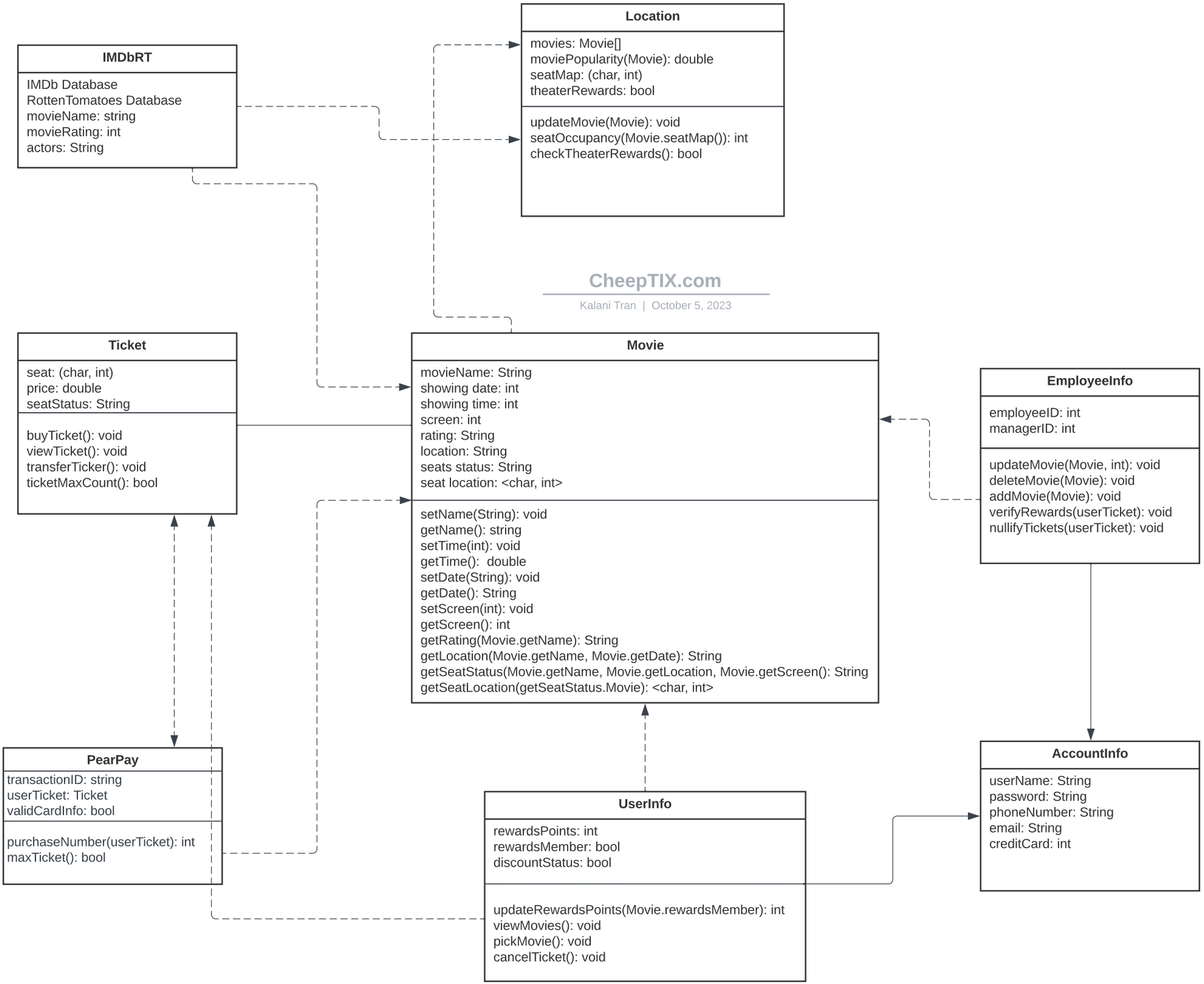
Prepared for
CS 250- Introduction to Software Systems
Instructor: Gus Hanna, Ph.D.
Fall 2023

Brief Overview

The user shall be able to purchase movie tickets through CheepTix.com. CheepTix will give the user the option to select a theatre based on their location. Ticket selection choices will be seating, movie time, movie type, and whether they want a physical or digital ticket. The user will be able to create an account to gain benefits such as: rating movies, managing their tickets, and discounted tickets through loyalty rewards and veteran/student/senior status. CheepTix will contain a database of movie times and ratings.



<showtimes, seat, prices, movie info, account info, seat info, ticket info, payment info, rewards points, member status, purchase history>


Location
movies: Movie[]
moviePopularity(Movie): double
seatMap: (char, int)
theaterRewards: bool
updateMovie(Movie): void
seatOccupancy(Movie.seatMap()): int
checkTheaterRewards(): bool
IMDbRT
IMDb Database
RottenTomatoes Database
movieName: string
movieRating: int
actors: String
CheepTIX.com
Kalani Tran | October 5, 2023
Ticket
seat: (char, int)
price: double
seatStatus: String
buyTicket(): void
viewTicket(): void
transferTicker(): void
ticketMaxCount(): bool
Movie
movieName: String
showing date: int
showing time: int
screen: int
rating: String
location: String
seats status: String
seat location: <char, int>
setName(String): void
getName(): string
setTime(int): void
getTime(): double
setDate(String): void
getDate(): String
setScreen(int): void
getScreen(): int
getRating(Movie.getName): String
getLocation(Movie.getName, Movie.getDate): String
getSeatStatus(Movie.getName, Movie.getLocation, Movie.getScreen(): String
getSeatLocation(getSeatStatus.Movie): <char, int>
EmployeeInfo
employeeID: int
managerID: int
updateMovie(Movie, int): void
deleteMovie(Movie): void
addMovie(Movie): void
verifyRewards(userTicket): void
nullifyTickets(userTicket): void
PearPay
transactionID: string
userTicket: Ticket
validCardInfo: bool
purchaseNumber(userTicket): int
maxTicket(): bool
UserInfo
rewardsPoints: int
rewardsMember: bool
discountStatus: bool
updateRewardsPoints(Movie.rewardsMember): int
viewMovies(): void
pickMovie(): void
cancelTicket(): void
AccountInfo
userName: String
password: String
phoneNumber: String
email: String
creditCard: int

Description of Classes:

Movie:

The Movie class holds all data regarding the specific movies showing in theaters, including: movie name, show dates, showtimes, screen numbers, ratings of the movies, what theaters are showing what movies, the seats positions at the theaters, and if there's open seating in the theaters. This data can be manipulated within the class to set and get any of the aforementioned info.

Ticket:

The Ticket class holds information regarding ticket info, such as the particular seat number indicated by a character and then an integer, ex. K19, the price of the ticket indicated by a double, ex. 12.99, and the status of the seat such as whether it's an open seat or a reserved purchase seat. This information can be manipulated within the class to buy the ticket which will call PearPay, view the ticket, transfer the ticket, and display whether the max amount of tickets, 10, are bought by one user.

Location:

The Location class holds all relevant data for the location of the theaters registered on CheepTIX.com. The class holds an array of all of the movies at a particular location, displays the ratings of the movies at a particular location, holds a seat map of the theaters at a particular location, and displays whether a theater is signed up for the rewards system compatible with CheepTIX.com. The Location class has functions including updating the movie at a location and checking the current seat occupancy of a location.

AccountInfo:

The AccountInfo class holds information regarding the account of a user, such as passwords, usernames, phone numbers, email addresses, and their credit card information. This class pertains more to logging in and verifying that the account exists within the system, and will also be implemented with security protocols to ensure only authorized accounts can log in to the system.

UserInfo:

The UserInfo class holds information for a user's account, but differs from the AccountInfo class in that the variation in user profiles are held in this class. This consists of the variable information such as whether a use is a rewards member, how many points they have if they are a rewards member, whether they are a student, senior, or in the military. This data has functions that can

EmployeeInfo:

The EmployeeInfo class holds all the information about employees at CheepTIX.com. This includes whether they are just an employee or have managerial status, and allows employees to update movies, delete movies, add movies, verify a user's rewards status via their sticker, and nullify user tickets if fraud is suspected.

IMDbRT:

The IMDbRT class contains the database information from IMDb and Rotten Tomatoes. These two movie review sites hold valuable information such as the movie's cast list, reviews from critics, synopses, and additional information about the movie. This class has been simplified for easy viewing of the UML and a simple call to the SQL database is contained within the class, but this class would contain its own expanded UML as well.

PearPay:

PearPay is the proprietary payment system for CheepTIX.com. The class contains information such as the transaction ID for a payment, the user's ticket information, and a valid credit/debit card. This information is authenticated to allow the user to purchase their ticket(s), and checks to see if the maximum number of tickets have been purchased.

Description of Attributes

Location:

- **movies**: Movie() String : Attribute holds the location of a specific movie name in the system.
- **moviePopularity**(Movie): double : Attribute gets popularity of movie and stores as moviePopularity type, once a specific movie is imputed.
- **seatMap**: char, int : Holds specific layout of seating for users to input a character or integer value (ex: B6).
- **theaterRewards**: bool : Stores a true or false value for user input if user is part of the rewards program on the system.

IMDbRT:

- **movieName**: string : Stores movie name input from Imdb data as a string.
- **movieRating**: int : Movie rating from IMDb data as an integer.
- **actors**: string : Name(s) of actors stored as string.

Ticket:

- **seat**: (char, int) : Location of a specific seat on the seating chart/map of each theater in the system labeled as character and integer value (ex: Seat H4)
- **price**: double : Price value for ticket at theater imputed as a double.
- **seatStatus**: string : String value that provides the user with status of current seat they have selected or would like to purchase.

Movie:

- **movieName:** string : Stores name of any movie as a string data type.
- **showingDate**: int : Provides movie showtime date as an integer (ex: 1/15/2023)
- **showingTime**: int : Shows movie showtime as an integer (ex: 10:45)
- **Screen**: int : Provides the number of screen corresponding to the theater in which the movie will be played (ex: Screen #2) however, this value is stored just as an integer therefore would just be (ex: 2).
- **rating**: string : user is allowed to see movie rating based on a written review shown as a string.
- **location**: string : Location of movie theater.

- **seatStatus**: string : Displayed if seat is occupied or empty.
- **seatLocation**: <char, int> : Provides location of seat on seat map (ex: Seat F3)

EmployeeInfo:

- **employeeID**: int : Displays employees ID #
- **managerID**: int : Displays manager's ID #

PearPay

- **transactionID**: string : ID# of a specific transaction
- **userTicket**: Ticket : Displays user ticket.
- **validCardInfo**: bool : Shows user's card info accepted or not.

UserInfo:

- **rewardsPoints**: int : Displays user rewards points as an int.
- **rewardsMember**: bool : Shows if user is a rewards member or not.
- **discountStatus**: bool : Displays if user is currently receiving a discount on ticket pricing.

AccountInfo:

- **userName**: string : Name user inputs to log into system and account.
- **password**: string : Password for user to log into system.
- **phoneNumber**: string : Phone number for user to receive information/tickets.
- **email**: string : Email address where user can receive info/tickets.
- **creditCard**: int : Saves card information for user to make recurring purchases.

Description of Operations:

Location:

- updateMovie(Movie) : void : Updates movie into system retuning void.
- seatOccupancy(Movie.seatMap()): int : Returns an integer describing the amount of seats a specific movie is showing.
- chekTheaterRewards(); bool : Checks whether the user is a rewards member at the theater, returns true or false.

Ticket:

- buyTicket(): void : Allows the user to purchase ticket, returns void.
- viewTicket(): void : Allows the user to view specific ticket they have purchased.
- transferTicket(): void : User can transfer ticket to another user in the system.
- ticketMaxCount(): bool : Specifies whether or not tickets are full or not.

Movie

- setName(String); void : Sets movie name as a string.
- getName(): string : Gets name of a specific movie and returns a string.
- setTime(int): void : Sets movie times as an int.
- getTime(): double : Gets movie time as a double.
- setDate(String): void : Sets movie date as a string.
- getDate(): String Gets movie date, returns value as a string.
- setScreen(int): void : Sets movie screen number.
- getScreen(): int : Gets movie screen number.
  getRating(Movie.getName): String : Gets movie rating based on movie name.
- getLocation(Movie.getName, Movie.getDate): String : Gets movie location based upon date and name of movie, shows available theaters.
- getSeatStatus(Movie.getName, Movie.getLocation, Movie.getScreen(): String : Gets seat status depending on the movie, theater location and screen number. Returns a string verification.
- getSeatLocation(getSeatStatus.Movie): <char, int> : Gets seat location based upon movie status.

Employee Info

- updateMovie(Movie, int): void : Updates movies
- deleteMovie(Movie): void : Deletes specific movie

- addMovie(Movie): void : Adds movie into system.
- verifyRewards(userTicket): void : Verifies rewards of user.
- nullifyTickets(userTicket): void : Movies user ticket.

PearPay

- purchaseNumber(userTicket): int : user can purchase ticket through PearPay
- maxTicket(): bool : Max tickets true or false.

UserInfo

- updateRewardsPoints(Movie.rewardsMember): int : updates user rewards
- viewMovies(): void : Allows user to view movies.
- pickMovie(): void : User can pick movies
- cancelTicket(): void : User can cancel order.

Development Plan and Timeline:

To make sure the development of the CheepTix software goes smoothly, we will be following a structured timeline as follows:

- Weeks 1-2: Project Planning, Location Verifications and Resource Requirements
- Weeks 3-4: Project Design and Beginning of Software Development
- Weeks 5-6: User and Employee Registration, Account Info and Login System Integration
- Weeks 7-8: Implementation of Payment System and PearPay Integration
- Weeks 9-10: Location, Theater and Seat Implementation
- Weeks 11-12: Ticket System Implementation and IMDbRT Integration
- Weeks 13-14: Security Implementation and Alpha Testing
- Weeks 15-16: Bug Fixes and Final Testing
- Week 17: Documentation and Beta Testing
- Week 18: Release Software

Team Member Responsibilities:

The following tasks will be assigned to team members as follows:

- Project Manager: Oversees that all objectives in the timeline are completed within parameters and that team members are completing their tasks efficiently.
- Business Analyst: Gathers information and scope out the best locations and theaters for Cheeptix and whether it can be verified there. Also verifies if the product is ready for commercial use.
- Engineering Manager: In charge with working with other team members to ensure a safe and working product. Responsible of analyzing any challenges in hardware and figuring out fixes for them.
- Software Architect: In charge of conducting the internal arrangement of the software and work with the Software Developers to perfect the product with optimal technical solutions.
- Software Developers: Responsible of implementing the internal arrangement and systems in the software by working with the Software Architect and in parallel with UI Designers and QA Engineers.

- UI Designers: In charge of the Project Design and UI system of the software. Makes sure the user experience is optimal, easy and convenient to use. Also work in parallel with Engineer and Software Developers.
- QA Engineer: In charge of quality assurance during testing, makes sure to point out any flaws in the system and automize the testing process for software quality.
- Testers: Required to test the software in the Alpha and Beta phase to see if it is ready for commercial use. Provides feedback before the release of the software.