**ENSF 619 Term Project**

**Design Phase**

**Group 14**

**Team Member:**

**Haixia Wu,**

**Jenny Tong Xu,**

**John Van Heurn,**

**Javier Vite**

Contents

[1. Systems use case diagram 3](bookmark://_Toc56893147#_Toc56893147)

[2. Systems activity diagram 5](bookmark://_Toc56893148#_Toc56893148)

[3. A state transition diagram 6](bookmark://_Toc56893149#_Toc56893149)

[3.1 Ticket object 6](bookmark://_Toc56893150#_Toc56893150)

[3.2 Payment object 6](bookmark://_Toc56893151#_Toc56893151)

[4. A detailed “Scenario” for each use case 7](bookmark://_Toc56893152#_Toc56893152)

[5. Systems interaction diagram 11](bookmark://_Toc56893153#_Toc56893153)

[5.1 Haixia Wu 11](bookmark://_Toc56893154#_Toc56893154)

[5.2 Jenny Tong Xu 12](bookmark://_Toc56893155#_Toc56893155)

[5.3 John Van Heurn 13](bookmark://_Toc56893156#_Toc56893156)

[5.4 Javier Vite 13](bookmark://_Toc56893157#_Toc56893157)

[6. A Design Level Class Specification 14](bookmark://_Toc56893158#_Toc56893158)

[6.1 A class diagram without attributes and behavior that only shows the class name and the relationships among them 14](bookmark://_Toc56893159#_Toc56893159)

[6.2 A class diagram with no relationships (no lines), only showing the class details: attributes and behaviors 14](bookmark://_Toc56893160#_Toc56893160)

[7. A Package diagram 18](bookmark://_Toc56893161#_Toc56893161)

[8. A Deployment diagram 19](bookmark://_Toc56893162#_Toc56893162)

# 1. Systems use case diagram

1.1 Actors

1. User

User are those people who use Movie Theater Ticket Reservation APP to buy movie’s tickets.

1. Registered User

Registered User is a subclass of User. The personal and credit-card information of the Registered User are saved on the database, making it easier for them to buy movie tickets using Movie Theater Ticket Reservation APP. They have some privileges (no movie cancellation fee, receive movie new by email, etc.) by paying annual fee.

1. Database Engine

Database Engine is the underlying software component that recognizes and interprets SQL commands to access a relational database and interrogate data. It’s also referred to as a SQL database engine or a SQL query engine. There is a relational database that store the information related with Movie Theater Ticket Reservation APP. Database Engine helps users to interact with the relational database.

1. Financial Institute

Financial Institute is referred to banks that offering banking and investment services to customers here. If User actors want to make a payment to a movie theater, the banks act as an intermediary between them.

# 2. Systems activity diagram

# 3. A state transition diagram

## 3.1 Ticket object

## 3.2 Payment object

# 4. A detailed “Scenario” for each use case

Notation: **candidate-objects candidate-operations**

1. Search Movie

This use case involves the user browsing the movie-catalogue. The user can choose search-movie and the program (booking-manager) will return a list of movies that matches the user’s criteria. This use case ends with user selecting a movie (front-end).

1. Select Theater

This use case begins when the user has already selected the movie. At this point the user will browse the theater-catalogue. The user can search-theaters-for-movie and the program (booking-manager) will return the list of movie theaters where the movie that the user has selected is playing. The user can also search-theater and the theater-catalogue will return a list of theaters where the selected movie is playing and that matches the user’s criteria. This use case ends with user selecting a theater (front-end).

1. View Showtime

This use case begins when the user has already selected the movie and the theater, the next step is to select a showtime. The front-end allows user to select a date for the upcoming week, and the booking-manager will get-showtimes for the user to select a showtime for the selected date (front-end).

1. Select Seat

This use case begins when the user has selected a showtime and decides to proceed. The booking-manager will display the seat-map for the theater-room where the movie will be played. The front-end allows users to select a seat, provided that the selected seat has not been booked.

1. Make Payment

This use case begins when the user has selected his seat and either created or retrieved his user profile (discussed in more detailed in the Scenario “Login”). The booking-manager get-ticket-price from showtime and process-payment for the movie-ticket. The booking-manager deduct ticket price from the user ‘s credit-card and increase revenue for theater. Once the transaction is approved by the financial institution, it book-seat (updates the seat-map and seat) and creates movie-ticket and print-ticket for user (via email).

1. Cancel Ticket

This use case begins when the user requests a cancellation and either created or retrieved his user profile (discussed in more detailed in the Scenario “Login”). The booking-manager will ask for information about the booking ID, validate-booking (check if booking exist) and verify-cancellation (check if the cancellation request is made up to 72 hours prior to the show). If cancellation can be processed, the booking-manager will cancel-seat.

1. Receive Credit

This use case begins after the user has completed his movie cancellation. The booking-manager will provide a Voucher to users at 85% of the movie cost with a one-year expiration date. For registered user, the system will provide a refund for 100% of the movie cost. This is done with a refund-credit function.

1. Pay Annual Fee

This use case begins after the user has logged-in to his account. The customer-manager, which tracks user information, check-fee-renewal-status for the registered-user. If payment is required, it will prompt-annual-fee which asks the user to pay a $20.00 annual account fee.

1. Receive Movie News

Registered-users have an operation to send-movie-news before movie announcements to users via email.

1. Login

The use case is used for booking tickets and occurs after book-seat and before process-payment. It is also used for cancelling ticket after validate-booking (check if booking exist) and verify-cancellation, and before cancel-seat. The user information is managed by customer-manager; He has the option of log-in (if he is a registered-user); otherwise, he can register-user or create-temp-account (user).

Candidate Objects

|  |  |  |
| --- | --- | --- |
| * Movie Catalogue | * Movie | * Theater Catalogue |
| * Theater | * Showtime | * Theater Room |
| * Seat Map | * Seat | * Movie Ticket |
| * Credit Card * Booking Manager | * User * Customer Manager | * Registered User * Voucher * process-payment |

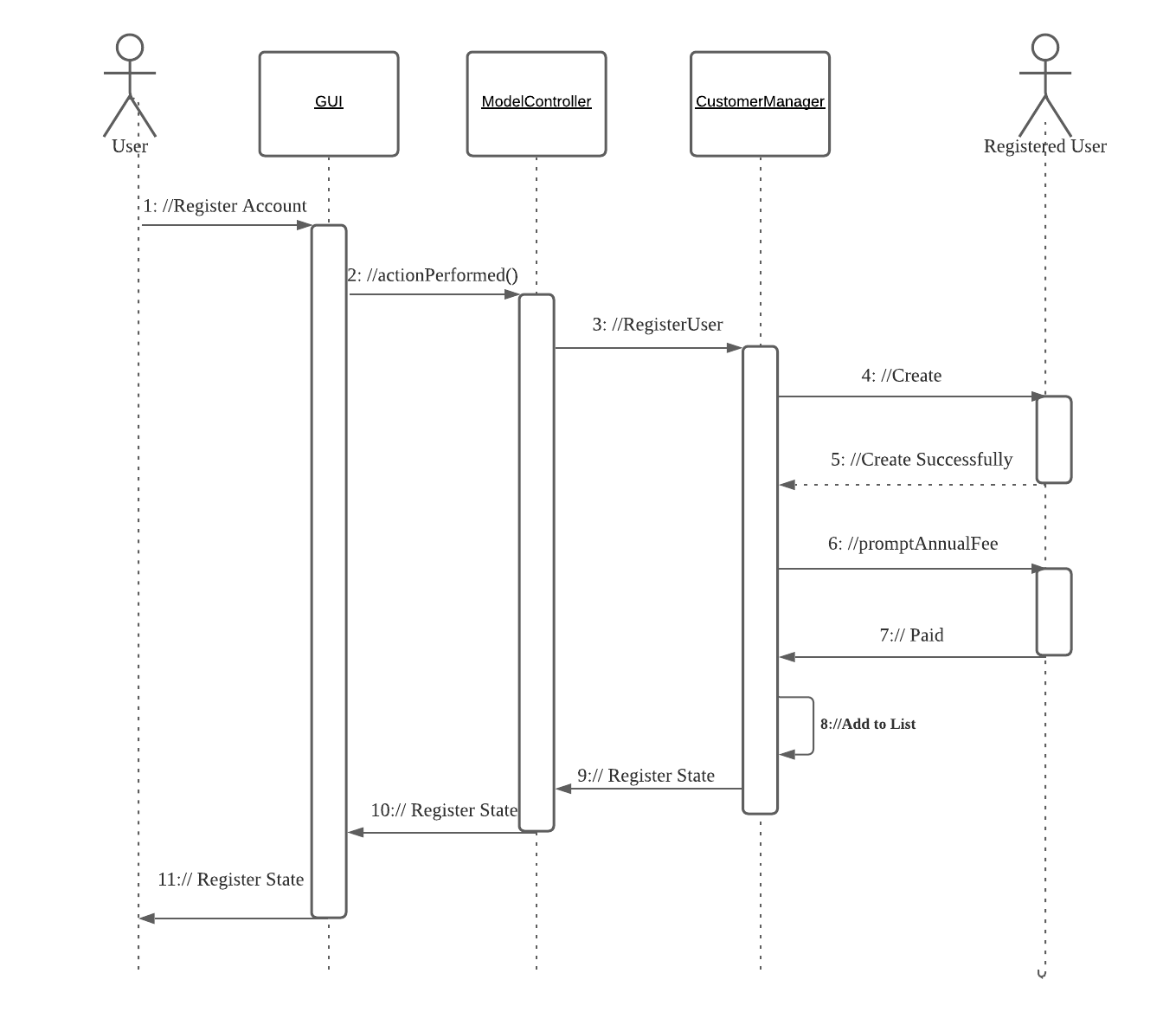
Candidate Operations

|  |  |  |
| --- | --- | --- |
| * view-all-movies * search-theater-by-name * get-ticket-price * print-ticket * cancel-seat * prompt-annual-fee * log-in * display | * search-movie * get-showtimes * process-payment * validate-booking * refund-credit * check-fee-renewal-status * create-user * create-temp account | * search-theaters-for-movie * Display-seats * book-seat * verify-cancellation * register-user * send-movie-news * search-theater |

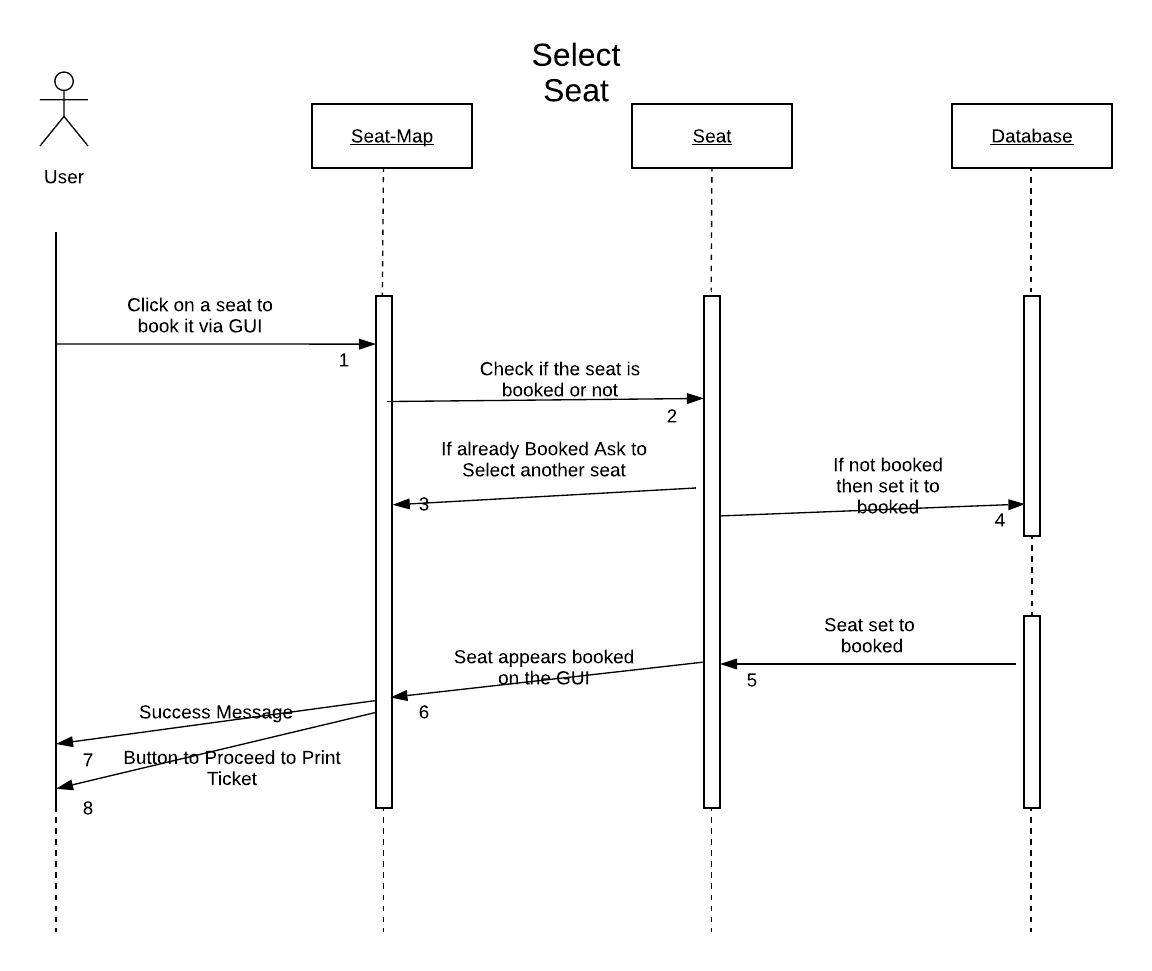
# 5. Systems interaction diagram

## 5.1 Haixia Wu

Register for Account



## 5.2 Jenny Tong Xu

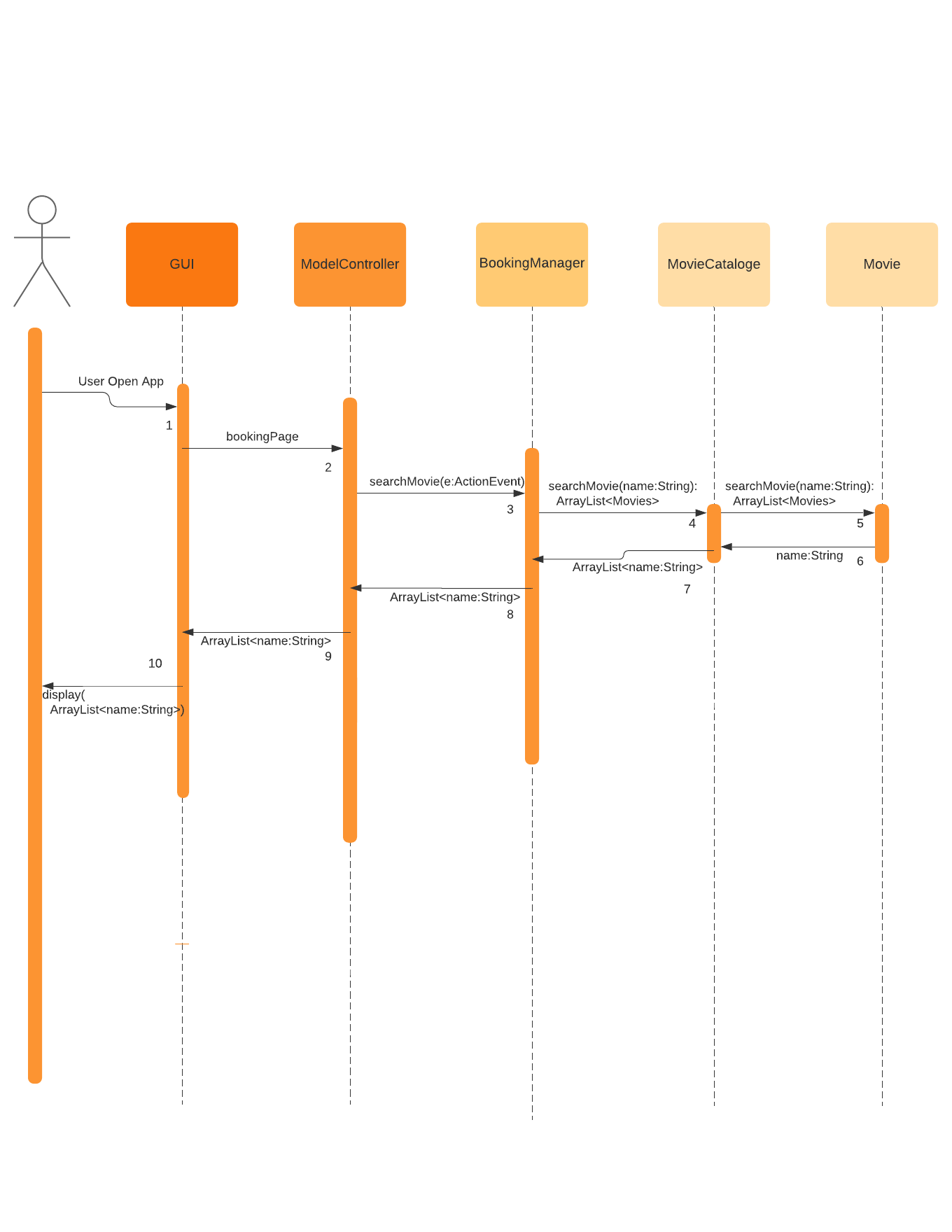


## 5.3 John Van Heurn

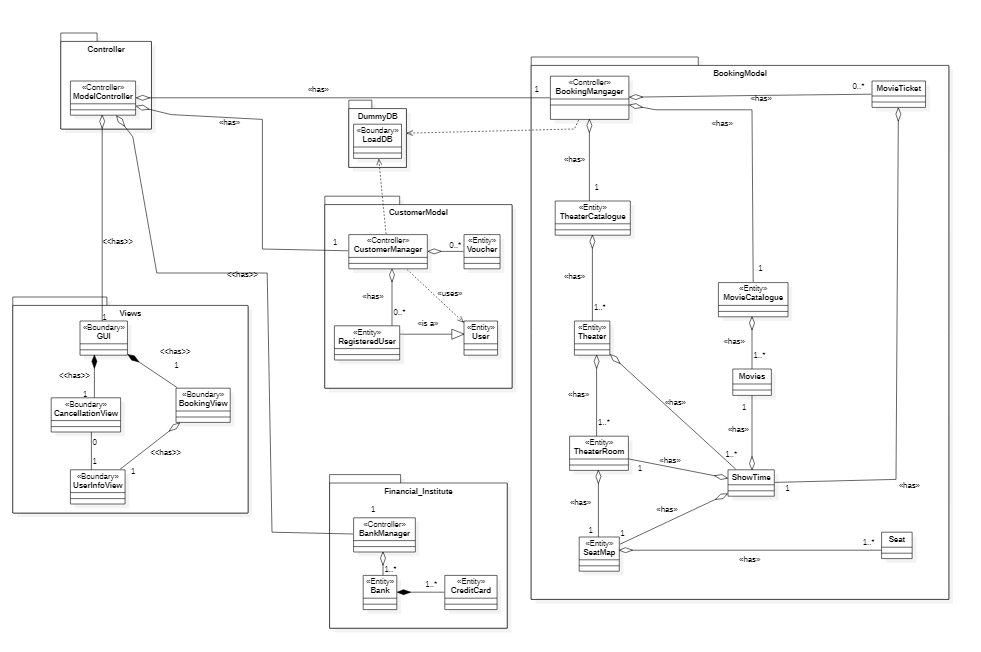
## 

## 5.4 Javier Vite

Search movie



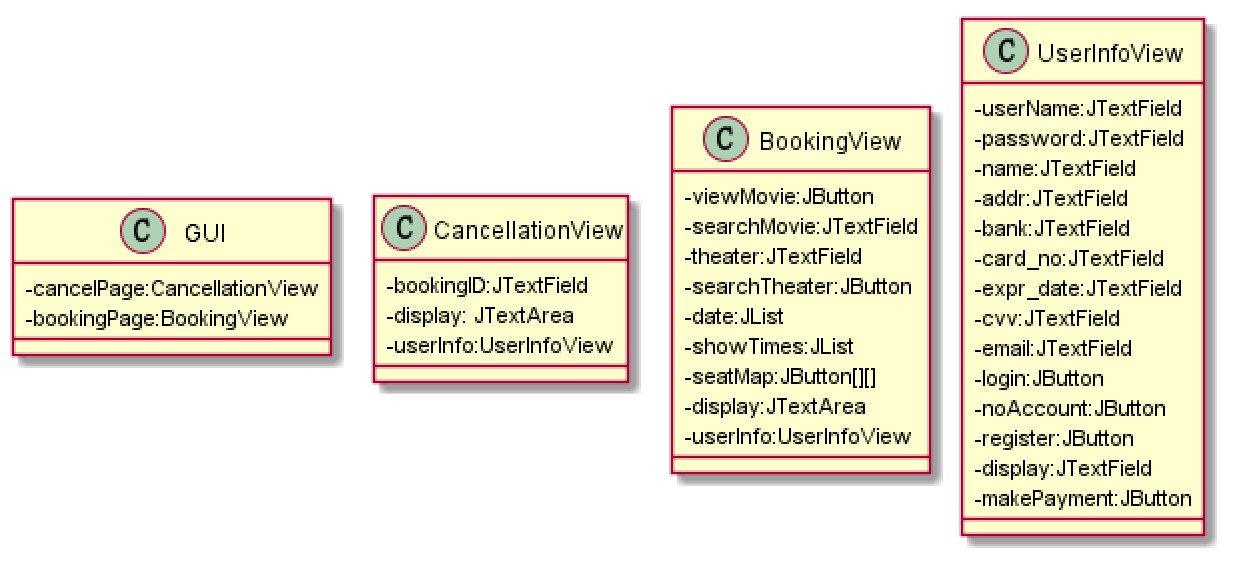
# 6. A Design Level Class Specification



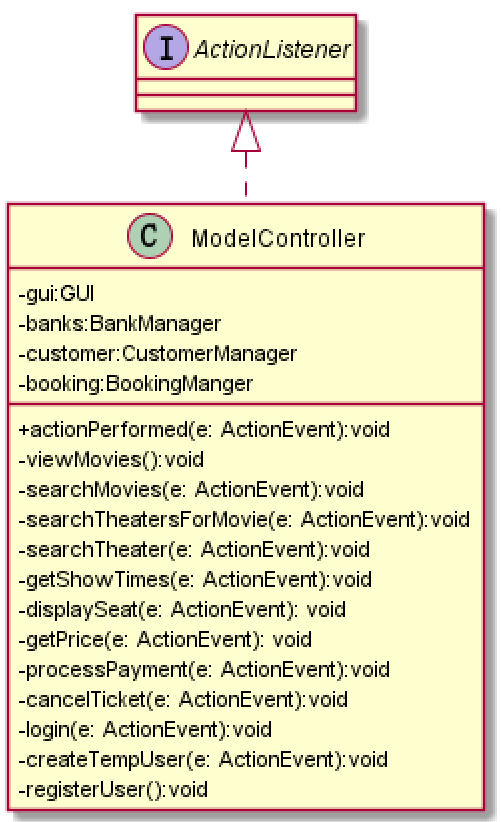
## 6.1 A class diagram without attributes and behavior that only shows the class name and the relationships among them

## 6.2 A class diagram with no relationships (no lines), only showing the class details: attributes and behaviors

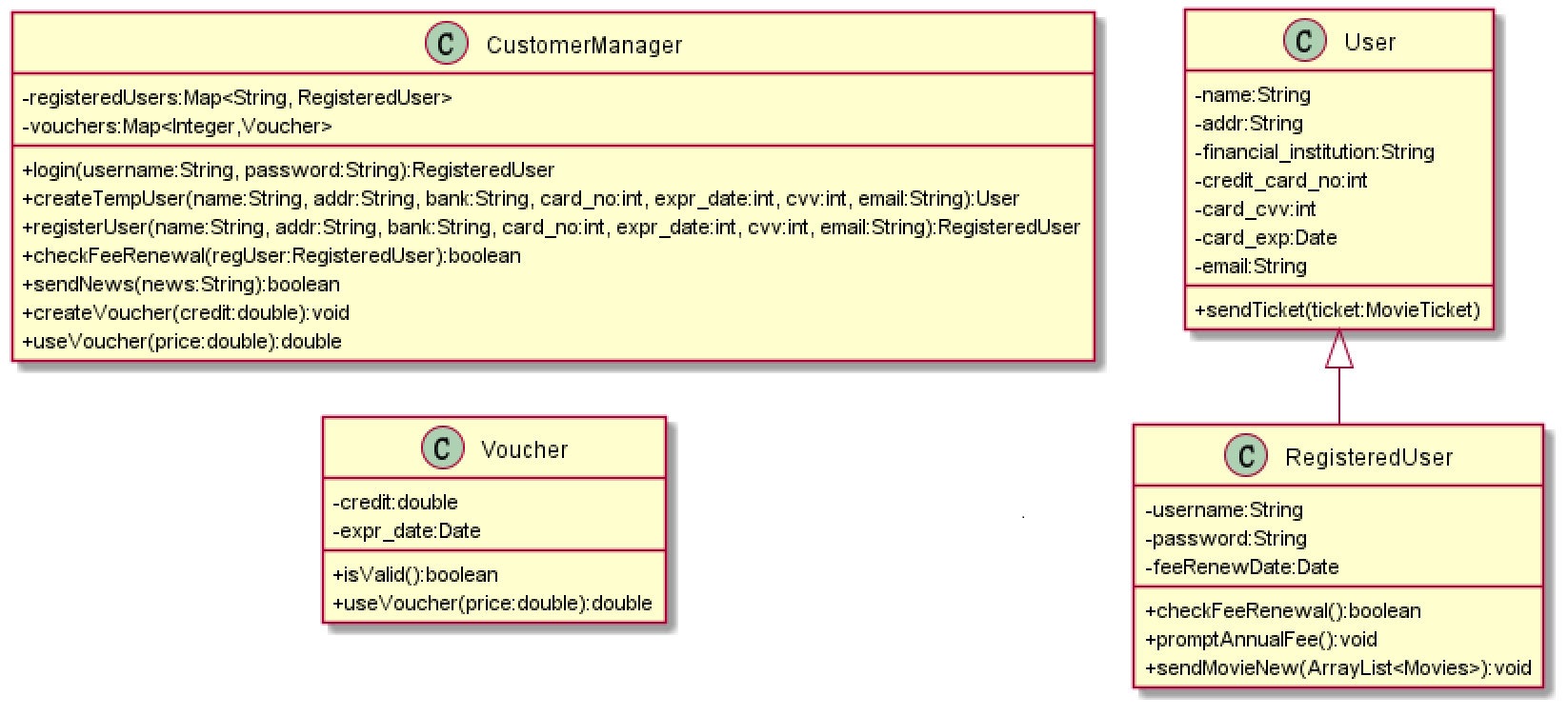
1. Views



1. Controller

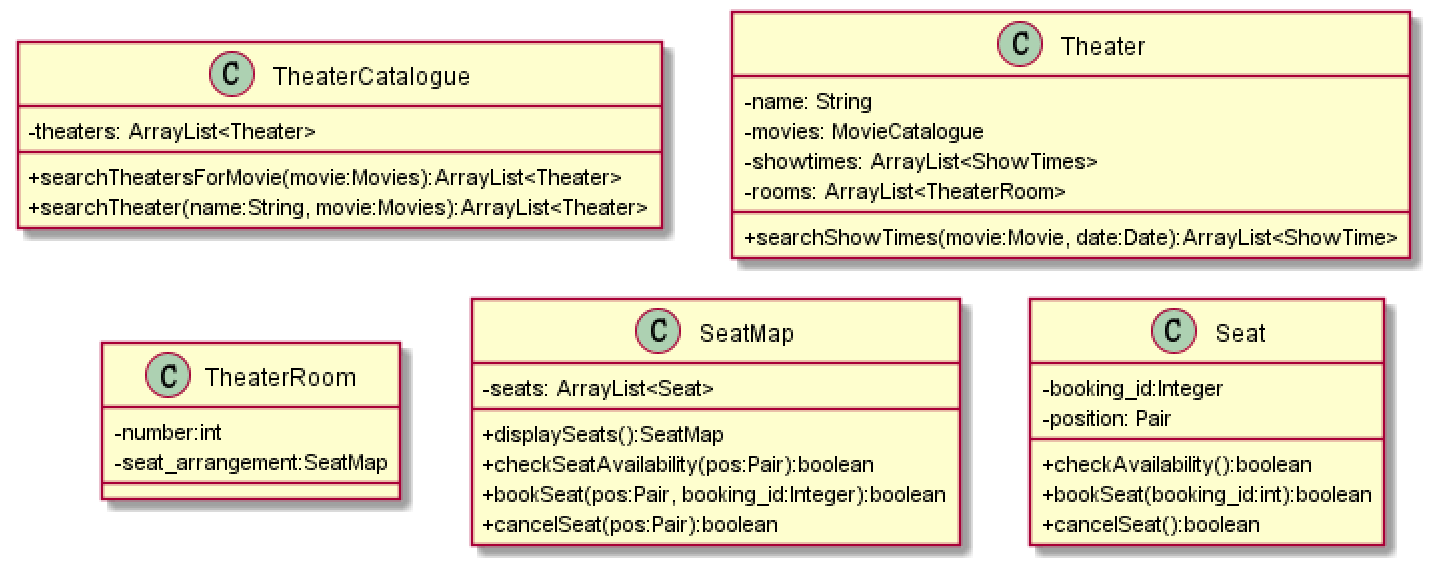


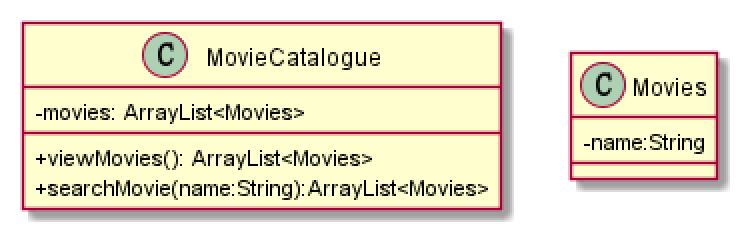
1. CustomerModel

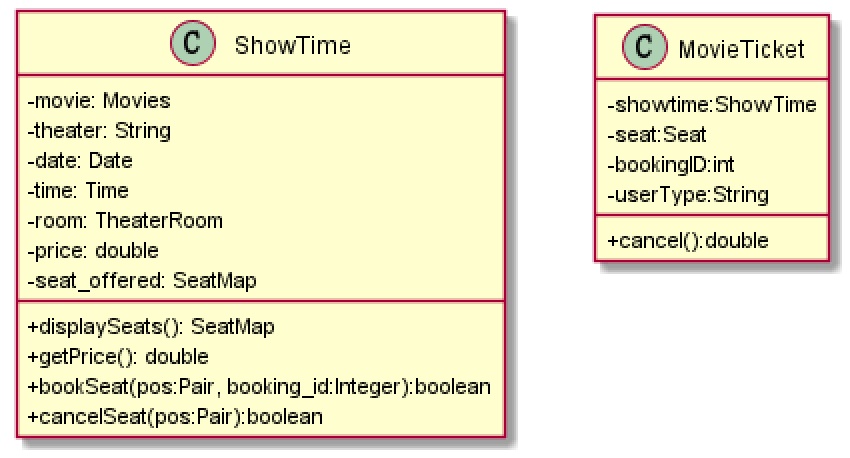


1. BookingModel

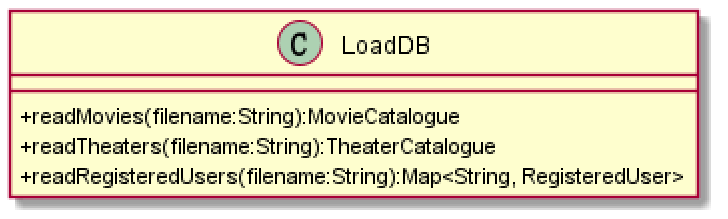




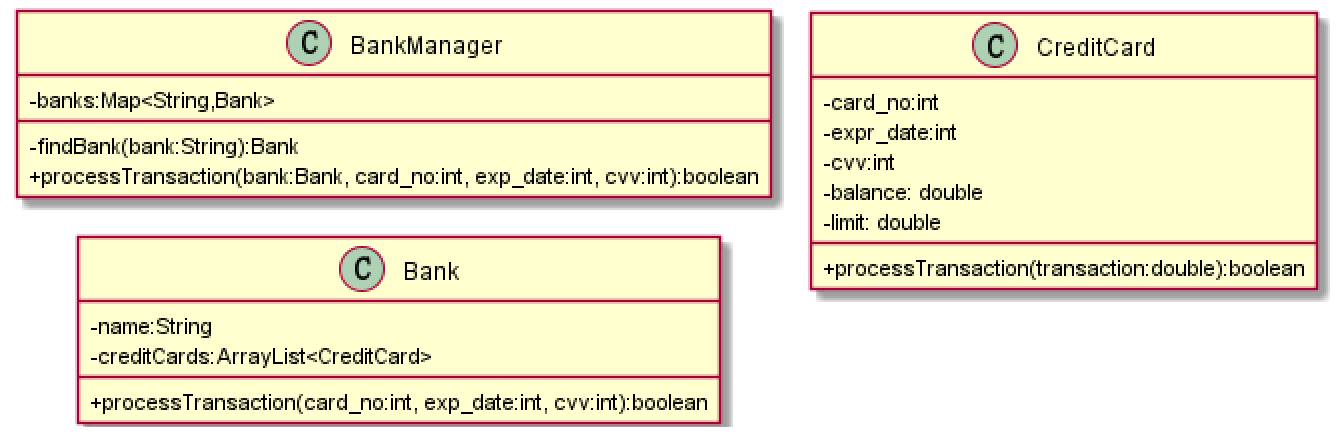




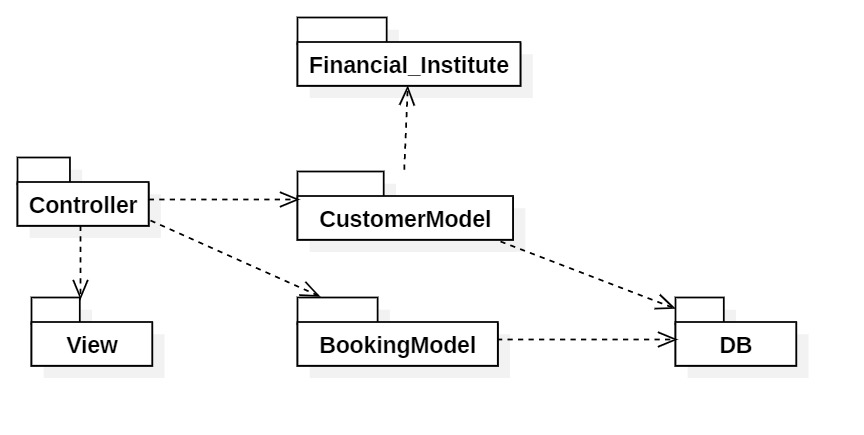
1. DummyDB



1. Financial\_Institute



# 7. A Package diagram



# 8. A Deployment diagram

