### **ENSF 614**

### Advanced System Analysis and Software Design Fall 2022

### Design Package

### Group 1:

Stewart Pratt - 30073940

Robert Njie - 30020243

Justin Nguyen - 30042258

Florian Bache - 10075304

Fizzah Malik - 10122276

### **Submission Date:**

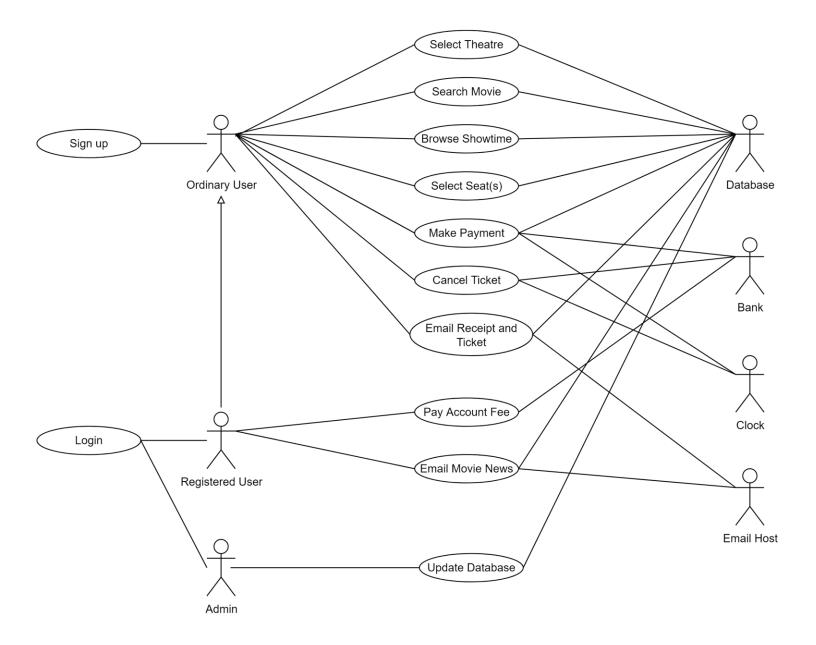
December 4, 2022

### **Table of Contents**

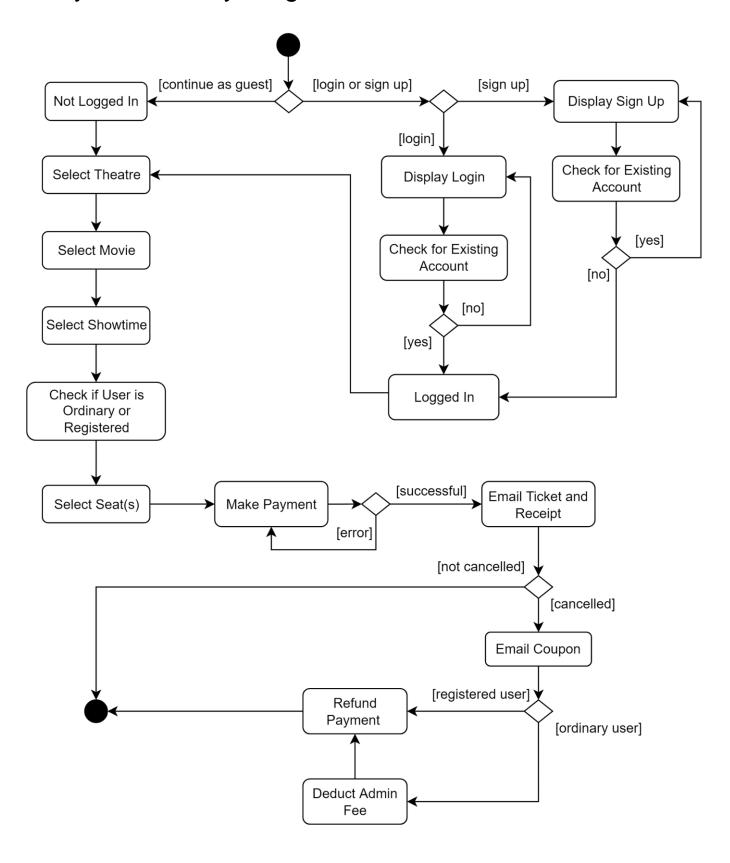
System Use Case Diagram	3
System Activity Diagram	4
Scenarios	5
Use Case 1: Login	5
Use Case 2: Sign up	5
Use Case 3: Select Theatre	6
Use Case 4: Search Movie	6
Use Case 5: Select Showtime	6
Use Case 6: Select Seat(s)	6
Use Case 7: Make Payment	7
Use Case 8: Cancel Ticket	7
Use Case 9: Email Receipt and Ticket	8
Use Case 10: Pay Account Fee	8
Use Case 11: Email Movie News	9
Use Case 12: Update Database	9
List of Candidate Objects from Use Case Scenarios	10
Nouns	10
Verbs	11
State Transition Diagrams	13
Ticket - Justin Nguyen	13
Payment - Stewart Pratt	14
Cancel Ticket - Fizzah Malik	15
Email Receipt and Ticket - Robbie Njie	15
Login - Florian Bache	16
System Interaction Diagrams	17
Login - Justin Nguyen	17
Sign Up - Stewart Pratt	18
Make Payment - Florian Bache	18
Cancel Ticket - Fizzah Malik	19
Select Seat(s) - Robbie Njie	19
Design Level Class Diagram	20
Class Diagram	21
Theatre classes	21

Showtime classes	21
Movie classes	22
SeatMap classes	22
Seat classes	23
User classes	23
Ticket classes	24
Payment classes	24
Email classes	25
Admin classes	25
Package Diagram	26
Deployment Diagram	27

### System Use Case Diagram



### System Activity Diagram



### **Scenarios**

### Use Case 1: Login

This scenario starts when a <u>registered\_user</u> or <u>admin</u> wants to login to the app by <u>clicking</u> on <u>Login</u>.

A <u>LoginView</u> window appears with two text fields for <u>EmailAddress</u> and <u>Password</u>. The window also displays two buttons, <u>Submit</u> and <u>Sign\_up</u>.

The <u>registered\_user</u> then enters their Email Address and Password and <u>clicks</u> the <u>Submit</u> button.

An <u>admin</u> will enter their Username and Password and <u>click</u> the <u>Submit</u> button.

The system then <u>checks</u> the given user information against the <u>Database</u> to see if the user exists. If successful, close the popup window, otherwise display an error message.

### Use Case 2: Sign up

This scenario starts when an <u>ordinary\_user</u> wants to sign up for an account by <u>clicking</u>on <u>Sign\_up</u>.

A <u>SignupView</u> window appears with five text fields for <u>FirstName</u>, <u>EmailAddress</u>, <u>Password</u>, <u>ConfirmPassword</u>, <u>CardholderName</u>, <u>CreditCardNo</u>, <u>ExpiryDate</u>, and <u>CVV</u>. The window also displays a button called <u>Submit</u>.

The  $\underline{ordinary\_user}$  then enters their information and  $\underline{clicks}$  the  $\underline{Submit}$  button.

The system then <u>checks</u> the given user information against the <u>Database</u> to see if the <u>registered user</u> already exists. If <u>registered user</u> does not exist, <u>charge</u> the <u>CreditCard</u> the annual fee, otherwise display an <u>error\_message</u>. If <u>payment</u> is successful, close the window, otherwise display an error\_message.

### Use Case 3: Select Theatre

This scenario starts when a <u>user selects</u> the <u>Theatres\_tab</u>.

Then the user can <u>scroll through</u> the <u>Theatre\_list</u> and <u>select</u> a <u>Theatre</u>.

A Movie\_list will appear on the screen and a user will select a Movie.

### Use Case 4: Search Movie

This scenario starts when a <u>user</u> <u>selects</u> a <u>Theatre</u>.

Then the <u>user can  $\underline{scroll\ through}$ </u> the  $\underline{Movie\_list}$  to  $\underline{\underline{search}}$  for a  $\underline{Movie\_and\ \underline{\underline{select}}}$  it.

A <u>Showtime list</u> will  $\underline{display}$  and the  $\underline{user}$  can  $\underline{\underline{select}}$  a <u>Showtime</u>.

### Use Case 5: Select Showtime

This scenario starts when a <u>user selects</u> a <u>Movie</u>.

Then the <u>user can scroll through</u> the <u>Showtime\_list</u> and <u>select</u> a <u>Showtime</u>.

A  $\underline{\text{Seat\_map}}$  will  $\underline{\textit{display}}$  and the  $\underline{\text{user\_will}}$  a number of  $\underline{\text{Seats}}$ .

### Use Case 6: Select Seat(s)

This scenario starts when a <u>user</u> has <u>selected</u> a <u>Theatre</u>, <u>Movie</u>, and a <u>Showtime</u>.

The user will then be shown a <u>Seat\_map</u> and the <u>Theatre</u>, <u>Movie</u> and <u>Showtime</u>.

The user will then <u>select</u> <u>available</u> <u>seats</u>.

The user will then <u>select</u> <u>proceed</u> button.

### Use Case 7: Make Payment

This scenario starts when a <u>user</u> has <u>selected</u> a <u>Theatre</u>, <u>Movie</u>, <u>Showtime</u>, and <u>available\_seats</u> and <u>clicked</u> <u>proceed</u>.

The <u>user</u> will be shown a <u>payment page</u> where the <u>Showtime</u>, <u>Theatre</u>, <u>Movie and Total cost</u> will be *displayed*.

The <u>user</u> will also be shown <u>CardholderName</u>, <u>CreditCardNo</u>, <u>ExpiryDate</u>, and <u>CVV</u> as input fields to make a <u>payment</u> and an input field <u>EmailAddress</u> to <u>receive</u> <u>confirmation</u>.

The <u>user</u> will <u>enter</u> their <u>information</u> in the <u>input fields</u>.

The <u>user</u> will then <u>select</u> <u>confirm and pay</u>.

The system will  $\underline{verify}$  that the  $\underline{payment}$  was successful.

The user will then be <u>displayed</u> a confirmation page of the <u>ticket information</u> and payment summary.

### Use Case 8: Cancel Ticket

This scenario can start when a <u>user selects</u> <u>cancel\_button</u>.

The <u>user will <u>enter</u> their <u>ticket\_information</u> into the <u>cancel\_window</u>. The <u>ticket\_information</u> will be <u>retrieved</u> from the <u>Database</u>. This includes <u>theatre</u>, <u>movie</u>, <u>showtime</u>, and <u>seat</u>.</u>

If the <u>cancel\_button</u> was <u>clicked</u> is more than 72 hours before <u>showtime</u>, a popup <u>confirmation window</u> will be <u>displayed</u> to the <u>user</u>. If <u>user <u>presses</u> decline or the hour to <u>showtime</u> is less than 72 hours the system will exit use case.</u>

If the <u>user will selects</u> confirm and cancel the <u>user group will be</u> <u>checked</u>.

If the <u>user</u> is an <u>ordinary\_user</u>, the system <u>checks</u> the <u>total\_cost</u> of the <u>Ticket</u> and 85% of full <u>payment</u> is <u>returned</u> to the <u>user</u> as <u>credit</u>.

If the <u>user</u> is an <u>registered\_user</u>, the system  $\underline{\underline{checks}}$  the <u>total\_cost</u> of the <u>Ticket</u> and the full <u>payment</u> is  $\underline{returned}$  to the <u>user</u> as <u>credit</u>.

The system will then  $\underline{\underline{update}}$  the  $\underline{\underline{Database}}$  to  $\underline{\underline{add}}$  the  $\underline{\underline{credit}}$  and  $\underline{\underline{Date}}$  of  $\underline{\underline{Cancellation}}$  as well as  $\underline{\underline{remove}}$  the  $\underline{\underline{Ticket}}$ .

### Use Case 9: Email Receipt and Ticket

This scenario starts when a <u>user has <u>selected</u> <u>confirm and pay</u> or <u>confirm and cancel.</u></u>

If the <u>user</u> has <u>selected</u> <u>confirm\_and\_pay.</u> The system <u>retrieves</u> the <u>user</u> <u>EmailAddress</u>, <u>Ticket</u>, and <u>Receipt</u> from the <u>Database</u>.

The <u>Ticket</u> and <u>Receipt</u> is then <u>emailed</u> to the <u>user EmailAddress</u>.

If the user has <u>selected</u> <u>confirm\_and\_cancel.</u> The system <u>retrieves</u> the <u>user EmailAddress</u>, <u>Receipt</u> from the <u>Database</u>.

The Receipt is then *emailed* to the user EmailAddress.

### Use Case 10: Pay Account Fee

This scenario starts when a new day begins. The <u>Database</u> is <u>checked</u> to see if it has been one year since a <u>registered\_user</u> first registered.

If such a  $\underline{\underline{registered\_user}}$  is  $\underline{\underline{found}}$ , their  $\underline{\underline{EmailAddress}}$ ,  $\underline{\underline{CardholderName}}$ ,  $\underline{\underline{CreditCardNo}}$ ,  $\underline{\underline{ExpiryDate}}$ , and  $\underline{\underline{CVV}}$  is  $\underline{\underline{retrieved}}$  from the  $\underline{\underline{Database}}$ .

Payment charges \$20 to their CreditCard.

The <u>EmailHost</u> sends an <u>Email</u> to the <u>registered user</u> containing the <u>Receipt</u>.

### Use Case 11: Email Movie News

This scenario starts when  $\underline{\text{MovieList}}$  is updated with a new  $\underline{\text{Movie}}$ .

The new Movie's information is written into an Email.

The <u>EmailAddresses</u> of all <u>registered users</u> are <u>retrieved</u> from the <u>Database</u>. They are <u>sent</u> the <u>Email</u> with the <u>Movie</u> news.

### Use Case 12: Update Database

This scenario starts when an <u>admin clicks</u> <u>add movie</u>, <u>remove movie</u>, <u>add registered user</u>, <u>remove registered user</u>, <u>add staff</u>, or <u>remove staff</u> on the <u>adminView</u>.

The appropriate <u>object</u> will be  $\underline{added}$  or  $\underline{removed}$  from the <u>Database</u>. The <u>admin</u> will then be notified of success.

### List of Candidate Objects from Use Case Scenarios

### Nouns

```
registered_user
                       (Use Case 1, 2, 8, 10, 11)
 admin
                       (Use Case 1, 12)
Login
                       (Use Case 1)

    LoginView

                       (Use Case 1)

    EmailAddress

                       (Use Case 1)

    Password

                       (Use Case 1)
Submit
                       (Use Case 1, 2)
Sign_up
                       (Use Case 1)
                       (Use Case 1, 2, 8, 9, 11, 12)

    Database

    ordinary user

                       (Use Case 2, 8)
                       (Use Case 2)
Sign_up
SignupView
                       (Use Case 2)

    FirstName

                       (Use Case 2)

    EmailAddress

                       (Use Case 2, 7, 9, 10, 11)

    Password

                       (Use Case 2)

    ConfirmPassword

                       (Use Case 2)

    CardholderName

                       (Use Case 2, 7, 10)

    CreditCardNo

                       (Use Case 2, 7, 10)

    ExpiryDate

                       (Use Case 2, 7, 10)
CVV
                       (Use Case 2, 7, 10)

    CreditCard

                       (Use Case 2, 10)

    error message

                       (Use Case 1, 2)
user
                       (Use Case 3, 4, 5, 6, 7, 8, 9)

    Theatres tab

                       (Use Case 3)
Theatre_list
                       (Use Case 3)

    Theatre

                       (Use Case 3, 4, 6, 7, 8)

    Movie List

                       (Use Case 3, 4, 11)
                       (Use Case 3, 4, 5, 6, 7, 8, 11)

    Movie

    Showtime List

                       (Use Case 4, 5)
                       (Use Case 4, 5, 6, 7, 8)

    Showtime

Seat_map
                       (Use Case 5, 6)
Seats
                       (Use Case 5, 8)
available_seats
                       (Use Case 3, 6, 7)
                       (Use Case 6, 7)
   proceed
   payment_page
                       (Use Case 7)
 Total cost
                       (Use Case 7, 8)
```

- Payment (Use Case 7, 8, 10)
- confirmation (Use Case 7)
- information (Use Case 7)
- input\_fields (Use Case 7)
- confirm\_and\_pay (Use Case 7, 9)
- ticket\_information (Use Case 7)
- payment\_summary (Use Case 7)
- cancel\_button (Use Case 8)
- ticket\_information (Use Case 8)
- confirm\_and\_cancel (Use Case 8, 9)
- credit (Use Case 8)
- Date of Cancellation (Use Case 8)
- Ticket (Use Case 8, 9)
- Receipt (Use Case 9, 10)
- Email (Use Case 10, 11)
- EmailHost (Use Case 10)
- add\_movie (Use Case 12)
- remove movie (Use Case 12)
- add staff (Use Case 12)
- remove staff (Use Case 12)
- add registered user (Use Case 12)
- remove\_registered\_user (Use Case 12)

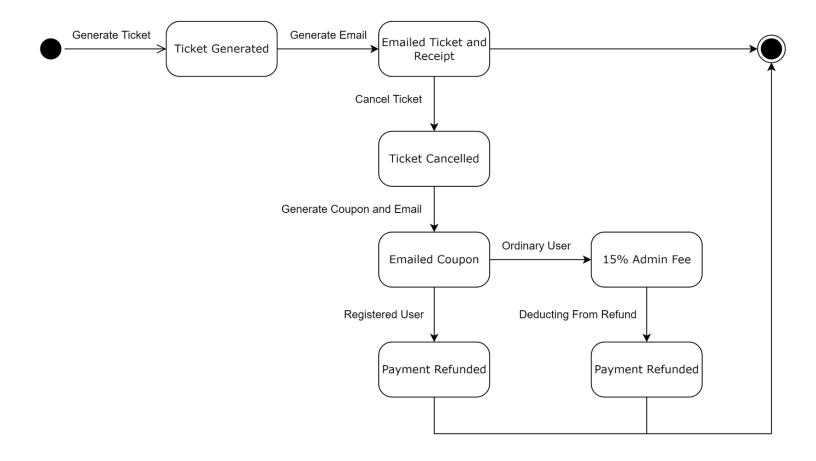
### Verbs

- click (Use Case 1, 2, 7, 8, 12)
- check (Use Case 1, 2, 8, 10)
- charge (Use Case 2, 10)
- select (Use Case 3, 4, 5, 6, 7, 8, 9)
- scroll through (Use Case 3, 4, 5)
- display (Use Case 3, 4, 5, 7, 8)
- search (Use Case 4)
- receive (Use Case 7)
- enter (Use Case 7, 8)
- verify (Use Case 7)
- retrieve (Use Case 8, 9, 10, 11)
- press (Use Case 8)
- return (Use Case 8)
- update (Use Case 8, 11)
- add (Use Case 8, 12)
- remove (Use Case 8, 12)

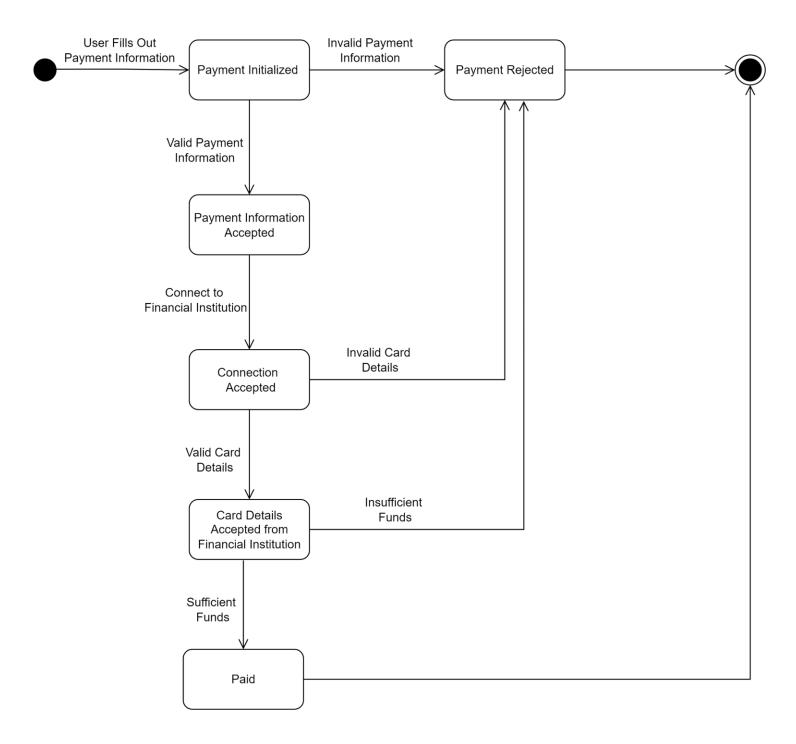
•	email	(Use Case 9)
•	charge	(Use Case 10)
•	write	(Use Case 11)
•	send	(Use Case 11)
•	notify	(Use Case 12)

### **State Transition Diagrams**

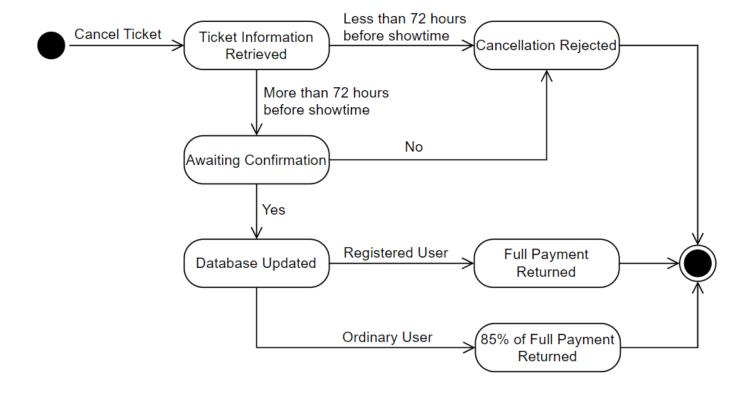
### Ticket - Justin Nguyen



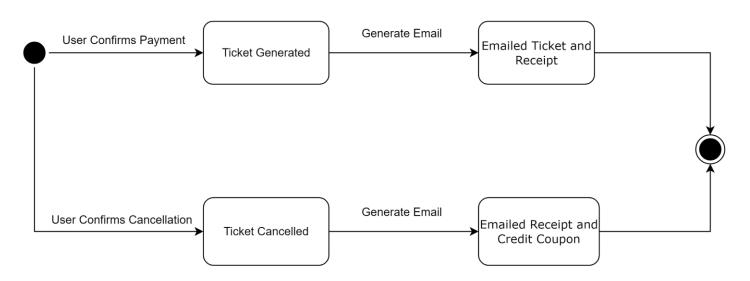
### Payment - Stewart Pratt



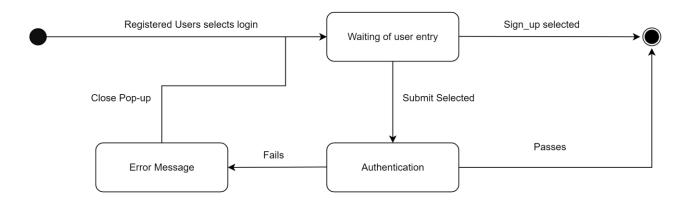
### Cancel Ticket - Fizzah Malik



### Email Receipt and Ticket - Robbie Njie

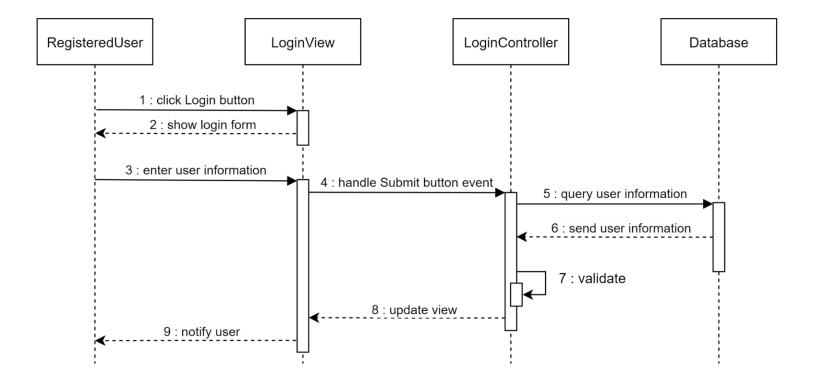


### Login - Florian Bache

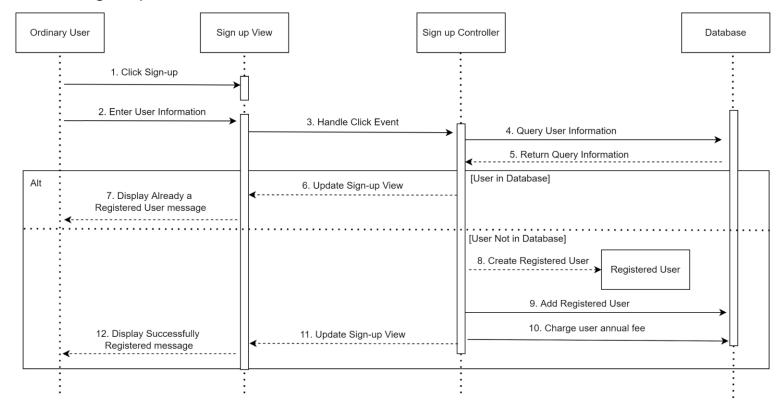


### **System Interaction Diagrams**

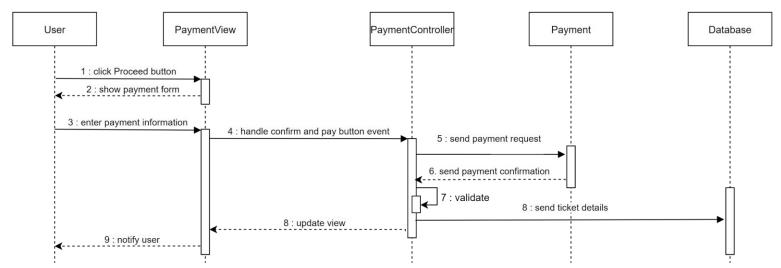
### Login - Justin Nguyen



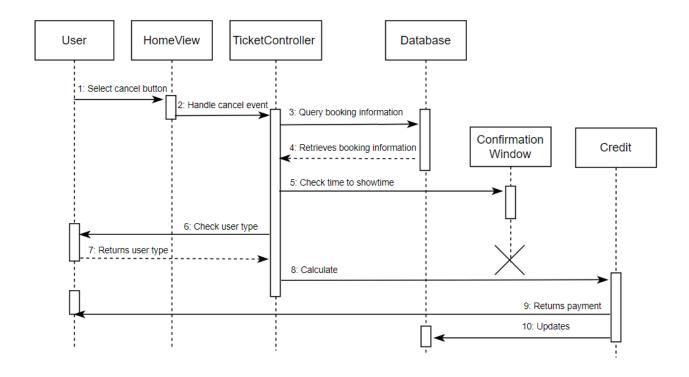
### Sign Up - Stewart Pratt



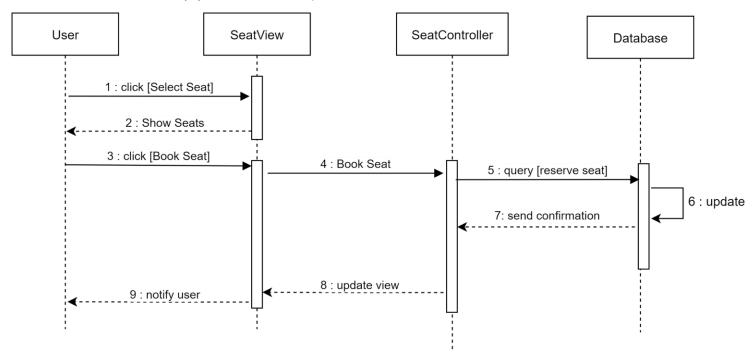
### Make Payment - Florian Bache



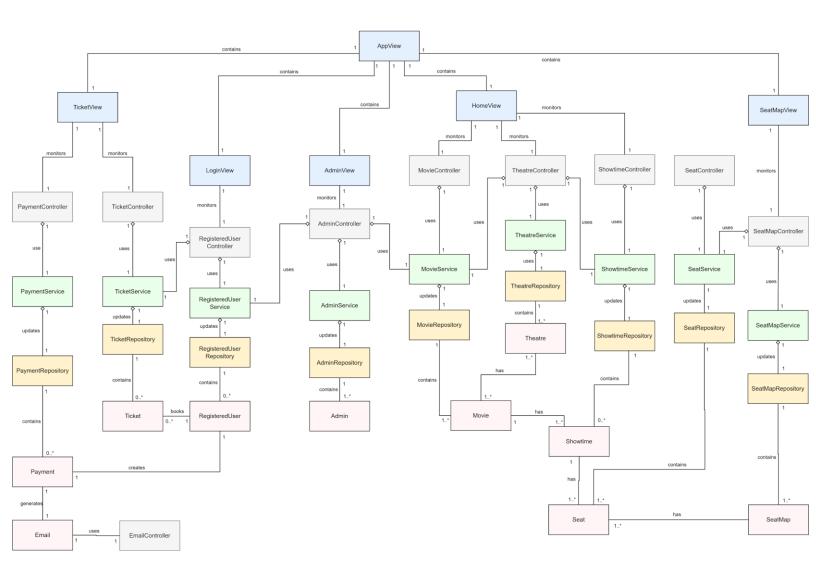
### Cancel Ticket - Fizzah Malik



### Select Seat(s) - Robbie Njie



### Design Level Class Diagram



### Class Diagram

### Theatre classes

### 

< <control>&gt; TheatreService</control>
- theatreRepository: TheatreRepository
+ getTheatres(): List <theatre> + addTheatre(theatre:Theatre):void + updateTheatre(theatre:Theatre):void + getTheatreByld(tld:Long):Theatre</theatre>

< <control>&gt; TheatreController</control>
- theatreService: TheatreService - showtimeService: ShowtimeService
+ addTheatre(theatre: Theatre): void + getMovieBasedOnTheatre(tld: Long): Set <movie> + getShowtimesBasedOnTheatreAndMovie(tld:Long,mld:Long) :Set<showtime> + getSeatsBasedOnShowtime(tld: Long): Set<seat></seat></showtime></movie>

### 

### Showtime classes

### <entity>> Showtime - stld: Long - showtime: LocalDateTime - seats: Set<Seat> - movie: Movie

<control>&gt; ShowtimeService</control>
- showtimeRepository: ShowtimeRepository
+ getShowtimes(): List <showtime> + addShowtime(showtime:Showtime): void + updateShowtime(showtime:Showtime): void + getShowtimeByld(stld:Long):Showtime + getMovieBasedOnShowtime(showtime: Set<showtime>) :Set<movie> + getShowtimeBasedOnMovie(showtime: Set<showtime> ,mld:Long):Set<showtime></showtime></showtime></movie></showtime></showtime>

< <control>&gt;</control>
ShowtimeController
- showtimeService: ShowtimeService
+ registerNewShowtime(showtime:Showtime) :ResponseEntity <string> + aetShowtimes(): List<showtime></showtime></string>

< <box>       <br <="" td=""/></box>
ShowtimeRepository
-
+ findByld(stld:Long) : Optional <showtime></showtime>

### Movie classes

### 

< <control>&gt;</control>
MovieService
- movieRepository: MovieRepository
+ getMovies(): List <movie> + addMovie(movie:Movie): void + updateMovie(movie:Movie): void</movie>

< <control>&gt;</control>	
MovieController	
movieService: TheatreService	

+ registerNewMovie(movie:Movie):ResponseEntity<String> + getMovies(): List<Movie> <<br/>boundary>><br/>MovieRepository

+ findByName(name:String)
: Optional<Movie>

### SeatMap classes

< <entity>&gt; SeatMap</entity>
- id: Long - theatreRoom: String - reserveCapacity: int - seats: Set <seat></seat>
+ addedSeats(seat: Seat): void + setAddedSeats(addedSeats:Set <seat>):void</seat>

< <control>&gt;</control>
SeatMapService
- seatMapRepository: SeatMapRepository
+ getAllSeats(): List <seatmap> + addNewSeatMap(seatMap:SeatMap): void + updateSeatMap(seatMap:SeatMap): void + getSeatMapById(seatMapId:Long):SeatMap + getAvailableSeats(seatMap: SeatMap):List<seat></seat></seatmap>

< <control>&gt;</control>
SeatMapController
- seatMapService: SeatMapService - seatService: SeatService
+ registerNewSeatMap(seatMap:SeatMap): void

+ registerNewSeatMap(seatMap): Void+ getAvailableSeats(seatMap: SeatMap): List<Seat>

<<br/>

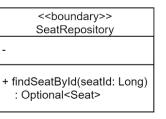
+ findByTheatreRoom(theatreRoom:String)
: Optional<SeatMap>

### Seat classes

### 

< <control>&gt; SeatService</control>
- seatRepository: SeatRepository
+ getAllSeats(): List <seat> + addNewSeat(seat:Seat): void + getSeatByld(seatId:Long):Seat + reserve(seat: Seat):void + reserveByld(seatId: Long):void + unreserveByld(seatId: Long):void</seat>

< <control>&gt;</control>
SeatController
- seatService: SeatService
+ registerNewSeat(seat:Seat): void + reserveSeat(seat: Seat): void + reserveSeatBvId(id: Long): void



### User classes

# <entity>> RegisteredUser - id: Long - email: String - password: String - name: String - address: String - accountBalance: double - dateRegistered: LocalDate - payments: Set<Payment> - cvv: int - cardNo: int - expiry: int - tickets: Set<Ticket>

+ addTickets(ticket:Ticket):void

< <control>&gt;</control>
RegisteredUserService
- registeredUserRepository: RegisteredUserRepository
+ getAllRegisteredUsers(): List <registereduser> + addNewUser(registeredUser:RegisteredUser): void + getUserByEmail(email:String):RegisteredUser + updateRegisteredUser(registeredUser:RegisteredUser):void + removeNewUser(registeredUser:RegisteredUser):void</registereduser>

### 

< <box>//&gt;&gt;&gt;//&gt;&gt;&gt;//&gt;&gt;&gt;//&gt;&gt;//&gt;&gt;//&gt;&gt;//&gt;&gt;//&gt;&gt;//&gt;&gt;//</box>
RegisteredUserRepository
_
+ findSeatByEmail(email: String)
: Optional <registereduser></registereduser>
+ findRvCardNo(cardNo:int):RegisteredLiser

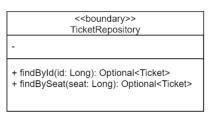
### Ticket classes

### <entity>> Ticket - id: Long - theatre: String - movie: String - showtime: String - price: double - seat: Long +

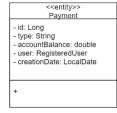
	< <control>&gt; TicketService</control>
- tick	etRepository:TicketRepository
+ ad + rer + ge	tAllTickets(): List <ticket> dNewTicket(ticket:Ticket): void moveTicket(id:Long):void tById(id:Ling):Ticket ncelTicket(if:Long):void</ticket>

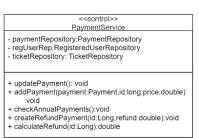
< <control>&gt; TicketController</control>
- seatService: SeatService - ticketService: TicketService - paymentService: PaymentService
+ getAllTickets(): List <ticket> + addNewTicket(ticket: Ticket): void + removeTicket(id: Long): void</ticket>

+ cancelTicket(id: Long): void

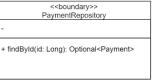


### Payment classes





< <control>&gt;</control>
PaymentController
- paymentService: SeatService
+ confirmPayment(amount:double): ResponseEntity <string> + registerNewPayment(paymnet:Payment,id:long,price:double) : void ResponseEntity<string> + refundPayment(id: Long,refundAmount:double) ResponseEntity<string></string></string></string>

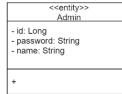


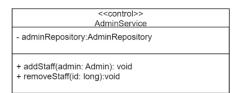
### **Email classes**

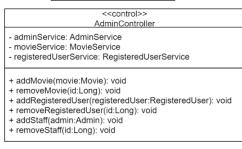
## <<entity>> Email - toEmail:String - contents: String - fromEmail:String - password: String +

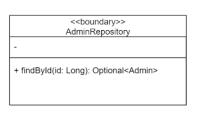
< <control>&gt;</control>
EmailService
-
+ createEmail(toEmail:String,contents:String): void + sendEmail(email:Email)

### Admin classes

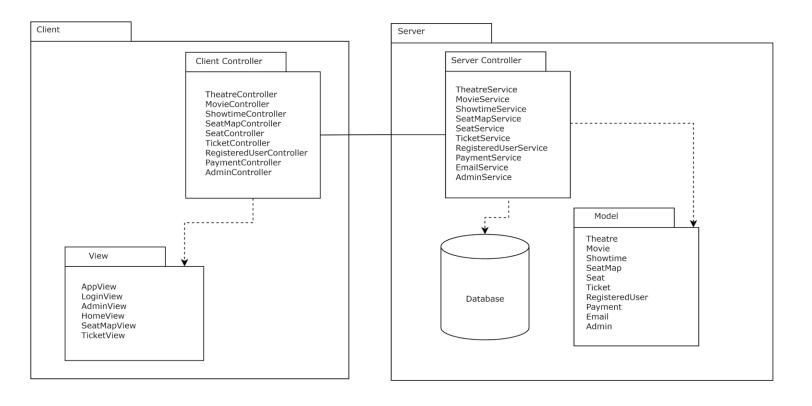








### Package Diagram



### **Deployment Diagram**

