ENSF614: Advanced System Analysis

Group 8 Final Project:

Course Dr. Moussavi

instructor:

TA Anja & Yobbahim

Date of November 29th

submission: 2021

By Alex Leakos

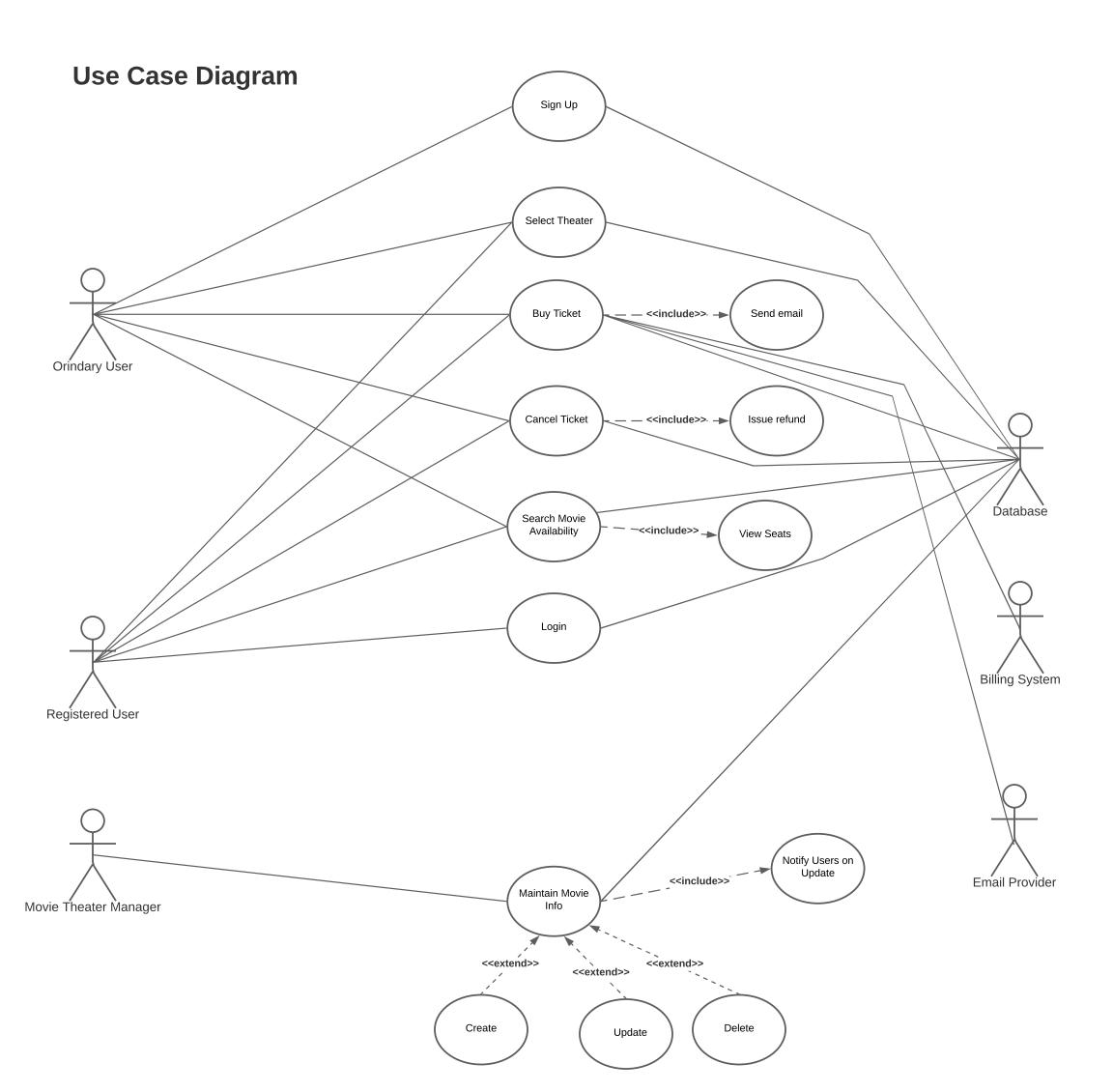
(10083280)

Melissa Liao (30139264)

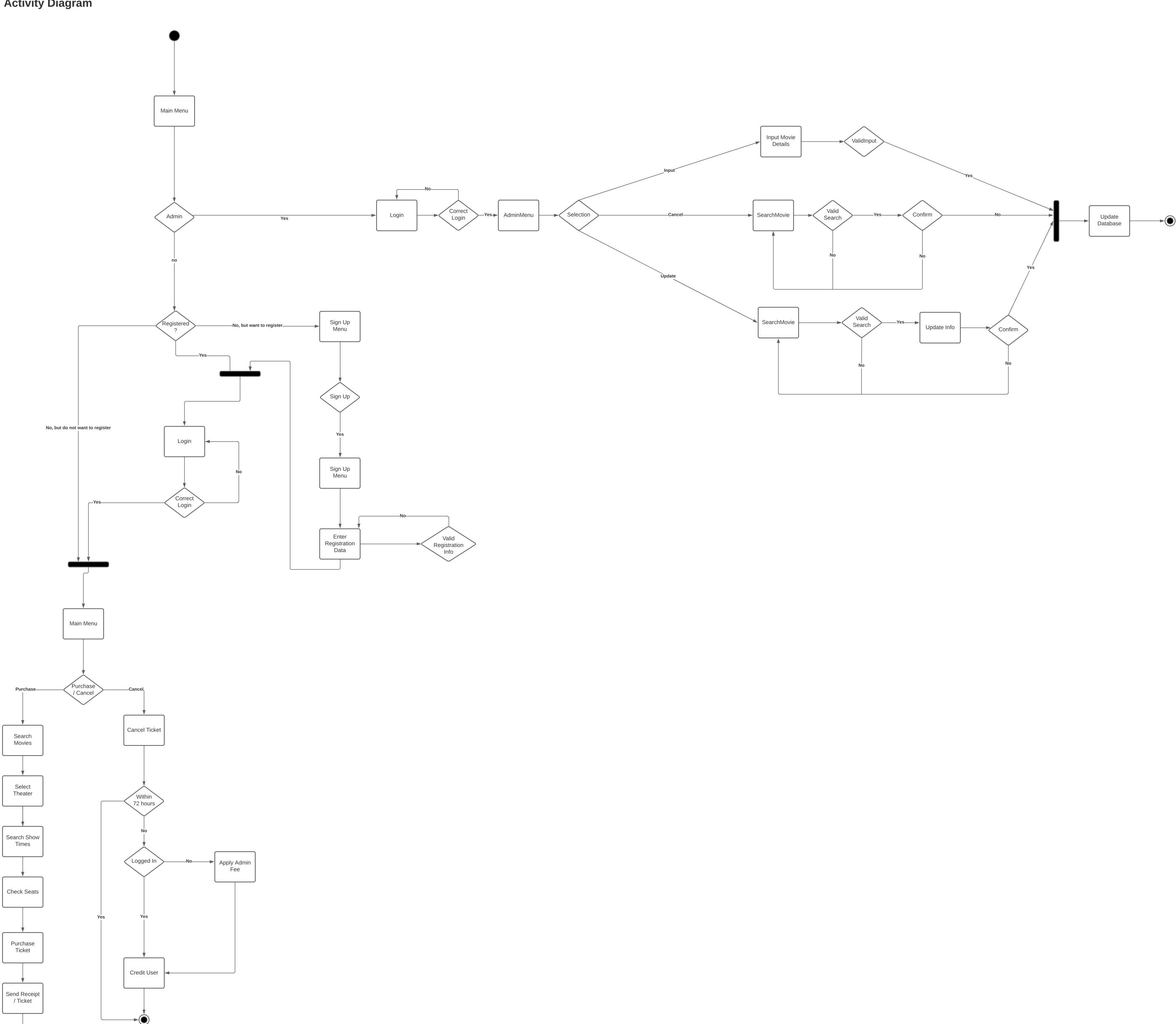
Michael Lee

Hamza Luqman (30144869)

Use Cases:



Activity	Diagrams:
----------	-----------



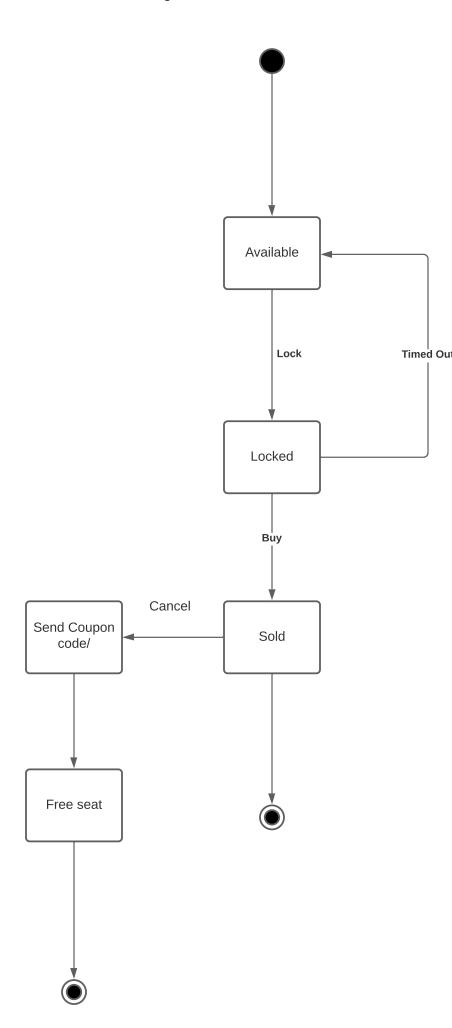
State	Transition:
State	Transition:

State Transition Diagram

Payment State

Diagram: Transaction Invoice and Not Approved Transaction **Email Invoice**

Ticket State Diagram:





Login:

<u>John</u> is a <u>registered user</u> who wants to <u>login</u> to the system. John enters the main <u>menu</u> and clicks on the <u>login</u> option and enters his username and password. The system <u>validates</u> the credentials input by John.

If <u>John's</u> credentials match the database, he is <u>granted access</u> to the system and enters the <u>main menu</u>. if <u>John's</u> credentials do not match the database, He Is <u>prompted</u> to re-enter his username and password.

Register:

John is currently an <u>ordinary user</u> (OU) and wants to be <u>registered</u>. John enters the <u>main menu</u> and clicks on the <u>sign-up</u> option. The system takes <u>John</u> to the <u>sign-up form</u> where <u>John</u> is required to input his first name, last name, email address, credit card or debit card information. The system display's that <u>John</u> will need to <u>pay</u> a \$20.0 annual membership fee where the first <u>payment</u> occurs at <u>registration</u>. Once John clicks <u>submit</u>, the system <u>verifies</u> that the required information is present and <u>debits</u> \$20.0 from <u>John's</u> credit/debit card. <u>John</u> is now <u>added</u> to the registered user's database.

Buy Ticket:

As an <u>ordinary user</u>, <u>Jane</u> enters directly to the <u>main menu</u> of the system. She <u>clicks</u> "<u>Select Movies</u>" and the system <u>views</u> her a <u>list of movies</u> available for the public. <u>Jane</u> looks through the <u>movie catalogue</u> and <u>selects</u> a <u>movie called</u> "<u>Titanic</u>". The system <u>views</u> her all the <u>theaters</u> available in the database for the <u>movie</u> "<u>Titanic</u>". <u>Jane selects</u> to watch the <u>movie</u> in the "<u>Blue Theater</u>" and the system <u>views</u> her back with the <u>show times</u> available in the <u>Blue Theater</u>. She <u>selects</u> the <u>show time</u> at "8:30 PM". The system <u>reads</u> her choice of movie "<u>Titanic in Blue Theater at 8:30 PM</u>. System <u>views</u> a graphical <u>layout of available seats</u> to choose from in the <u>Blue Theater</u>. <u>Jane clicks</u> on a <u>seat icon</u> and <u>selects seat "M15"</u>. To proceed with the <u>payment</u>, Jane <u>clicks</u> on "Proceed to Checkout" to <u>confirm</u> her choice.

The system <u>prompts Jane</u> to fill out her credit card information, email and any applicable <u>credit coupon code</u>. Jane remembers she has a <u>credit coupon code</u> from a <u>ticket</u> cancelled 6 months ago, so she <u>proceeds by entering</u> her <u>coupon code</u> and <u>confirm</u> if it is still usable. The system <u>validates</u> the <u>coupon code</u> in the database and <u>verifies</u> if it still valid from the time cancelled. The system <u>confirms</u> the <u>coupon code</u> is less than a year, so it <u>tells Jane</u> that her <u>coupon code</u> is valid. <u>Jane</u> is happy that the coupon is still usable and proceeds with completing her credit card information and email. Once all her <u>payment information</u> is filled out, she <u>clicks</u> "Confirm

<u>Payment"</u> to finish her <u>purchase</u>. The system immediately <u>validates</u> her credit card information with her bank and <u>validates</u> whether her email address is deliverable. The system <u>checks</u> all her <u>information</u> is correct. It <u>indicates</u> <u>Jane</u> that the transaction is complete and automatically <u>sends</u> a copy of the ticket and receipt to her email address.

Cancel Ticket:

<u>Jason</u> has bought a <u>ticket</u> and wants to have it <u>cancel</u> 5 days prior to the show. Jason <u>enters</u> to the <u>main menu</u> of the system, and <u>clicks</u> on <u>cancel ticket</u>. System then <u>prompts</u> Jason to enter the ticket# as well as credit card number from which it is paid. <u>Jason</u> enters the required information and press the confirm button to confirm the cancellation.

System <u>verifies</u> ticket# and credit card number against its <u>transaction records</u>. System, after successfully verified the entry, <u>calculates</u> the difference between the show date and the date of cancellation to make sure it is more than 72 hours. Once successful, the system <u>emails Jason</u> a <u>coupon code</u> that is <u>discounted</u> at 85% of the purchase value of the original ticket, as well as the expiration date, which is exactly 1 year after the date of refund.

Pre-sale ticket:

<u>System admin (manager) adds movie</u> "The Attack of the Observer Pattern" and system <u>sends</u> <u>email</u> to it's <u>registered users</u> that they can buy ticket for this movie prior to the official public announcement. <u>Michael</u>, a registered user, received this email, and quickly <u>access</u> the menu of the system.

<u>Michael press</u> login and the system <u>prompts</u> for username and password. Michael <u>enters</u> username and password and the system <u>validates</u> the input information against its records. The system <u>flows back</u> into the <u>Main Menu</u>, and Michael then <u>press</u> <u>"Select Movies"</u>. The System <u>returns</u> a view with a list of movies available, including the selection "The Attack of the Observer Pattern" that is not viewable by nonregistered users.

Michael <u>selects</u> "The Attack of the Observer Pattern" and system <u>shows</u> a list of <u>theaters</u> that this will be shown. Michael <u>selects</u> "Red Theater", and the system <u>shows</u> a list of available <u>showtimes</u> this movie is available in the particular <u>theater</u>. Michael <u>selects</u> December 10, 021 9pm, and system <u>shows</u> view with button for user to click to confirm <u>payment</u>.

Michael <u>clicks</u> confirm payment button, and system <u>process</u> payment using the stored credit card information. The system <u>indicates</u> that transaction is complete. The system <u>emails</u> Michael a copy of ticket and the receipt.

Adding a movie:

<u>Alex</u> is a <u>theatre manager</u> and needs to <u>enter</u> a new <u>movie</u> that is playing at his <u>theatre</u>. In his <u>theatre</u> movies only play on one <u>screen</u> and a <u>theatre</u> can only play one <u>movie</u> on a given date.

Alex receives news that his <u>theatre</u> will now be showing "<u>The Attack of the Observer Pattern"</u> and he decides this <u>movie</u> will be playing in <u>theatre 3</u>.

He <u>logs in</u> with his username and password. After the system <u>verifies</u> these Alex can proceed.

Alex opens up the <u>admin gui</u> and <u>selects "Input Movie"</u> and <u>inputs the movie info</u> into the <u>update movie gui</u>. He needs to <u>input</u> the Title of the Movie, length of the movie, for the movie, and the price of the ticket, he also needs to <u>select</u> which screen this movie is being played on.

He needs to <u>select</u> the start time of the first <u>showtime</u> as well as the start time of the last <u>movie</u> showing, and <u>input a number of dates</u> that the movie will run.

Based on the length of the movie above, as well as the start time of the first and last showings, show times are automatically generated and <u>input into the database</u>.

The system <u>displays</u> a confirmation that all the notice to <u>subscribers</u> has been sent.

Update a movie:

Alex is still the <u>theatre manager</u>. He needs to <u>make a change</u> to the start time of <u>"The Attack of the Observer Pattern"</u> on December 10, 2021. He's going to a birthday party that day and wants to change it so that the last show time is at 8pm.

He <u>logs in</u> with his username and password. After the system <u>verifies</u> these <u>Alex</u> can proceed.

Alex now needs to <u>open the gui</u> and <u>select</u> "<u>Update Movie</u>". Alex <u>searches</u> the <u>movie database</u> by <u>inputting the title</u> of the movie in a search bar. A result comes up for "<u>The Attack of the Observer Pattern"</u> showing an <u>editable list</u> of the dates playing, <u>theatre</u>, start time, and last show time.

Alex <u>updates</u> the row for December 10, 2021 and changes the "Last Show Time" column to 8pm.

Alex presses "Proceed" and updates the status of any movies after 8pm to "cancelled".

Cancel a movie:

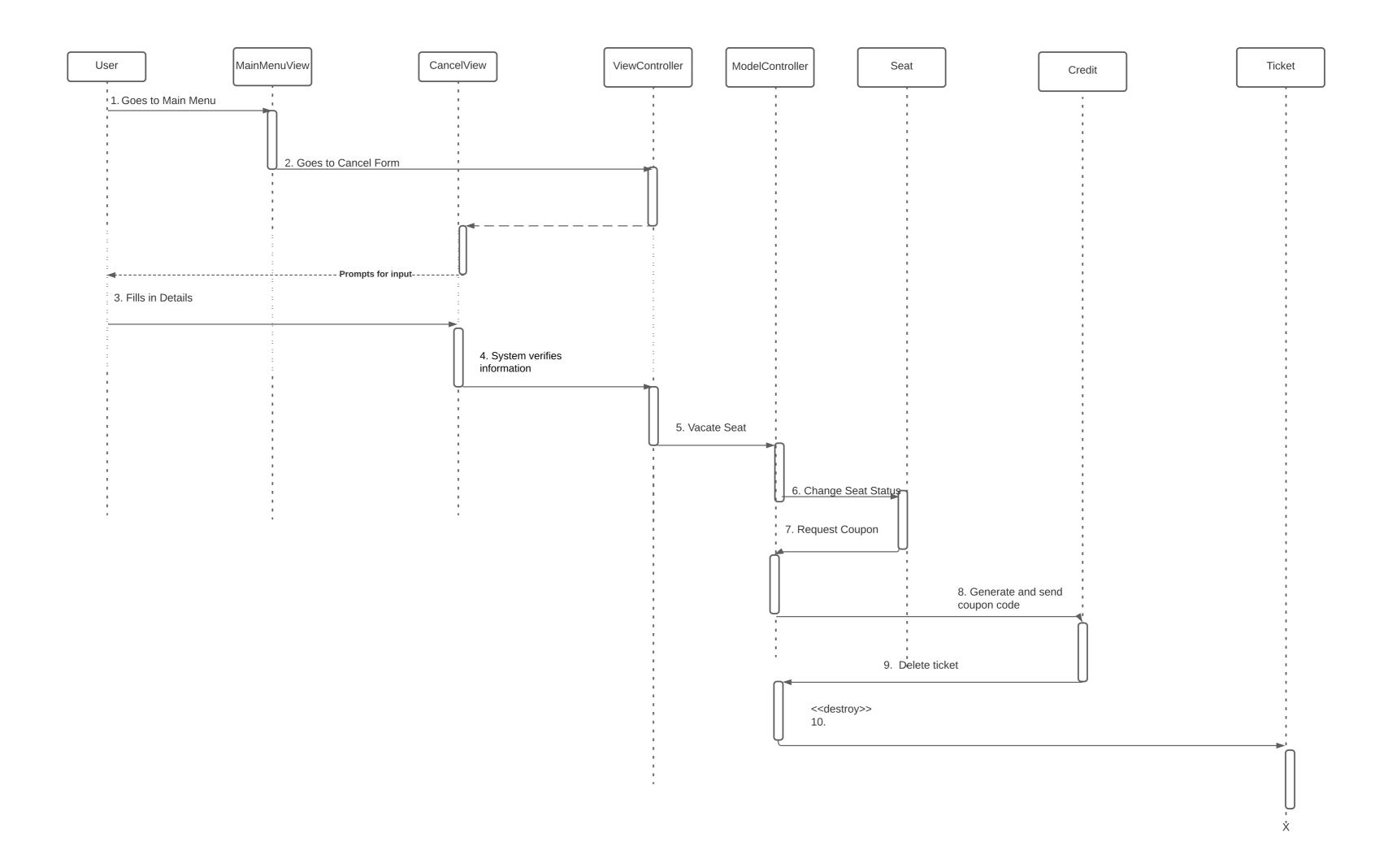
Once again, Alex is still the <u>theatre manager</u>. He has received news about <u>"The Attack of the Observer Pattern"</u> - apparently one of their stars made some-off colour remarks and now theatres across the country are pulling the movie.

He <u>logs in</u> with his username and password. After the system <u>verifies</u> these Alex can proceed.

Alex now needs to open the <u>gui</u> and <u>select</u> "<u>Cancel Movie</u>". Alex <u>searches</u> the movie database by <u>inputting the title of the movie in a search bar</u>. A result comes up for "<u>The Attack of the Observer Pattern</u>". <u>Alex selects</u> this movie and <u>clicks "Cancel"</u>.

The database <u>updates the status</u> of the movie from "Active" to "Cancelled"

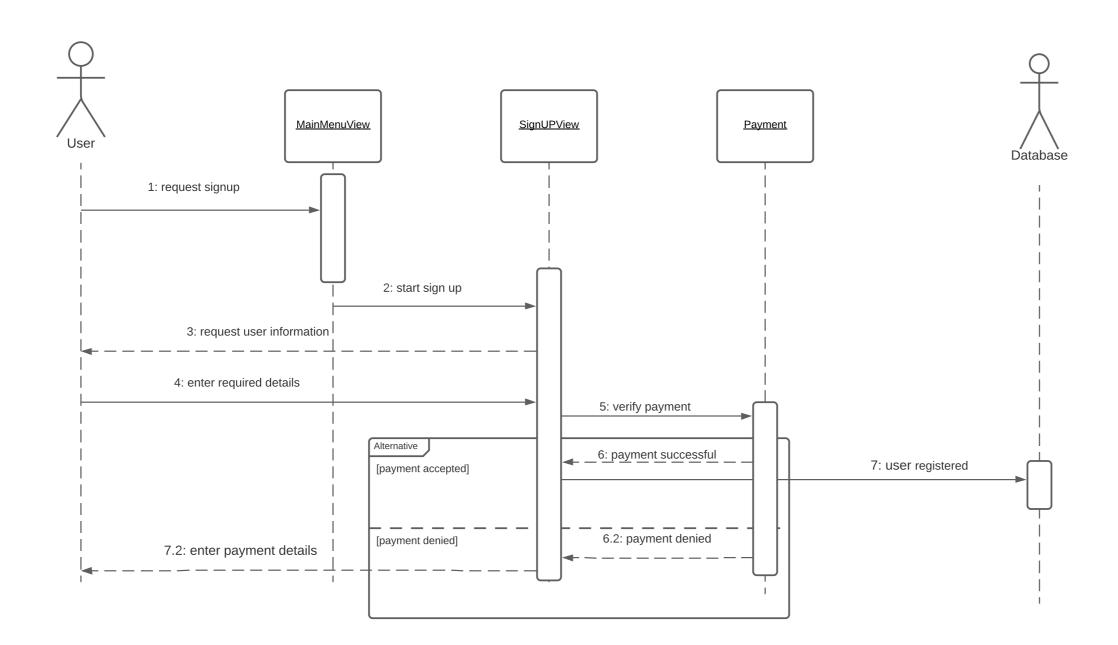
Sequence Diagrams:



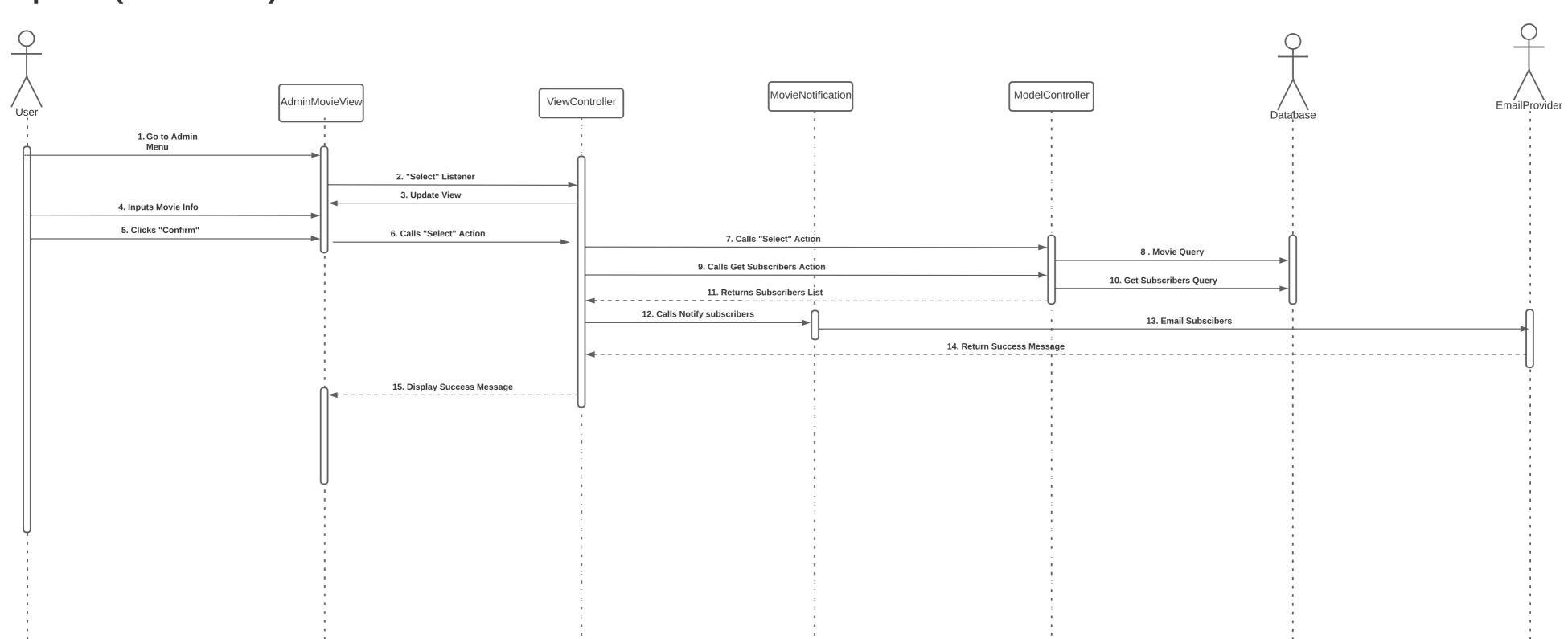
Buy Ticket - Melissa Liao Note: For every entity class, we access to the ViewController to retrieve the view for each class. The ViewController communicates with the ModelController to retrieve the list of movies, theater, show time and seat from the database. MainMenuView Theater MovieView Movie ShowTime Seat **Payment** CreditList Ticket 1: enters the main menu 2: selects to view movie catalogue 3: selects a movie from catalogue 4: view show times available and select a show time 5: view theaters available and select a theater 6: view available seats and 7: proceed to fill out credit card select seat(s) information, email address and coupon code (if applicable) 8: validate coupon code 9: return message for coupon status 10: once payment validation is complete

11: send copy of ticket an receipt to customer's email adrress

Register an Account - Hamza Luqman

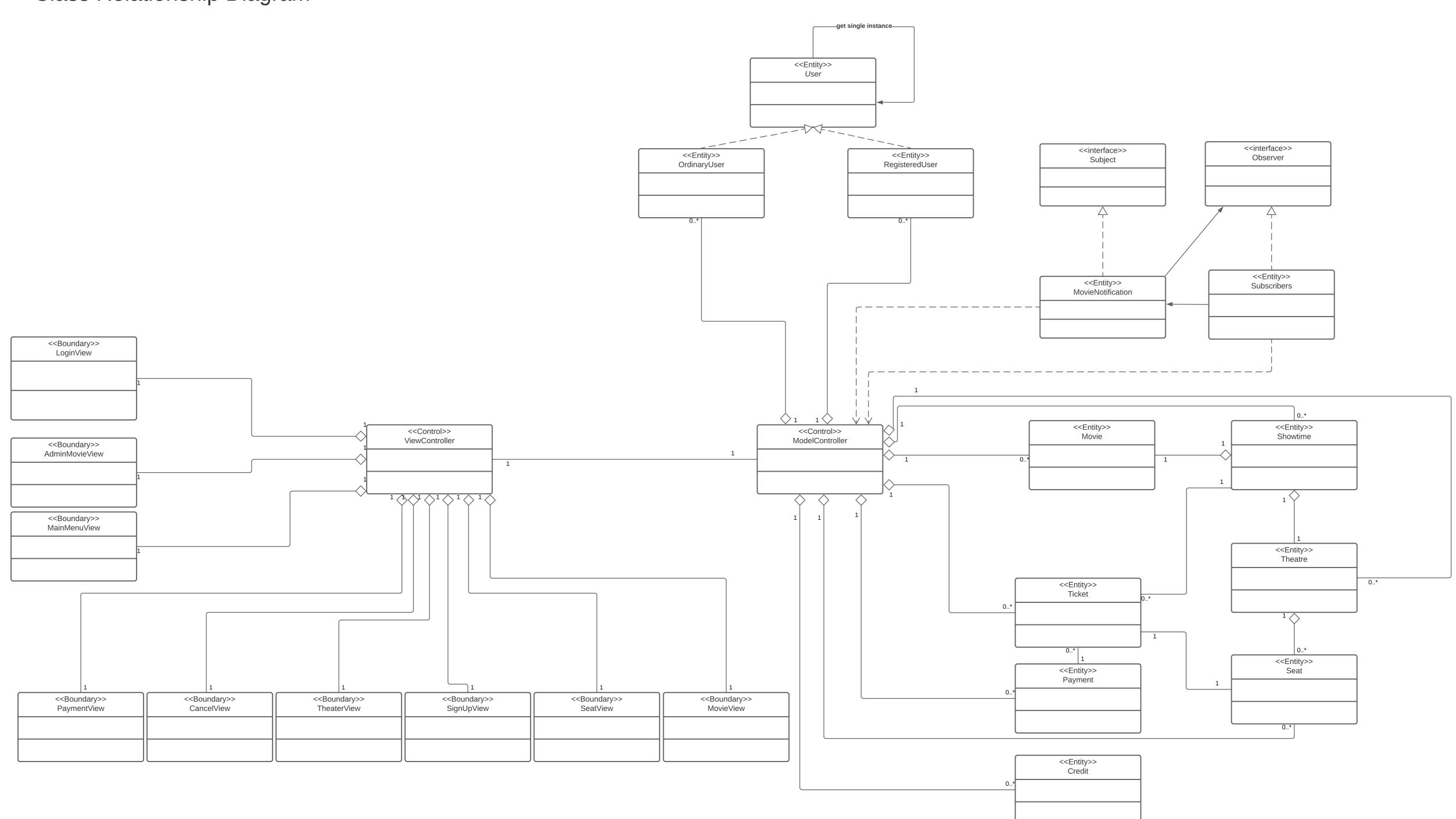


Update (Add/Delete) Movie - Alex Leakos





Class Relationship Diagram



Detailed Cla	asses:
--------------	--------

Class Diagrams - Attributes and Behaviours

<<interface>> Subject

+addObserver(Observer o): void +removeObserver(Observer o): void +notifyAllObservers(): void

<<interface>> Observer +update(String movieInfo): void

MovieNotification

-observers: ArrayList<Observer> -movie: ArrayList<Movie>

+MovieNotification(Movie m) +addMovie(Movie m): void +addObserver(Observer o): void +removeObserver(Observer o): void +notifyAllObservers(): void

Subscribers

-subject: Subject -subscriberList(): ArrayList<String>

+Subscribers()

+update(String movieInfo): void

ModelContoller

-userInstance:User -movieList: ArrayList<Movie> -theaterList: ArrayList<Theater> -showTimeList: ArrayList<Showtime> -ticket: ArrayList<Ticket>

+makeInputQuery(Int movieId): string +makeCancelQuery(int movield) : string +makeUpdateQuery(int movield , ShowTime updatedShowTimes, Date: starttime, Date: endtime): string +getAllSubscribers(): <String> +loadModelsQuery(): void +ModelController(User user) +getShowTime(String theater, String starttime): Showtime +addTicketQuery(Ticket ticket): void //calls payment as well +removeTicketQuery(Ticket ticket): string //outputs coupon code +createTicket(String movie, Showtime showtime, String seat, Date date): Ticket

+getUserInstance(); +addCouponCodeQuery(Credit coupon); void +removeCouponCodeQuery(string couponcode); void +getCouponCodeQuery(string couponcode); Credit +getMovies(); +getTheaters(); +getShowTimes(); +getTickets();

ViewController

- modelController: ModelContoller

 loginView: LoginView - adminView: AdminMovieView

- mainMenuView: MainMenuView

- cancelView: CancelView

- theatreView: TheatreView

- signUpView: SignUpView paymentView: PaymentView

seatView: SeatView

- movieView: MovieView

+ SystemController()

+ showSelectMovie():void

+ showSeatView():void

+ showSignUpView():void

+ showTheatreView():void

+ showCancelView():void + showPaymentView():void

+ showMainMenuView():void

+ showLoginView():void

+ showRegisterView():void

+ updateSelectMovieView(<Movie> movies):void

+ updateSeatView(<Seat> seats):void

+ updateSignUpView(string message):void

+ updateTheaterView(<Showtime> showtimes):void

+ updateCancelView(string message):void + updatePaymentView(string message):void

+ updateLoginView(string message):void

+ updateRegisterView(string message):void

+ addSelectMovieListener(ActionListener al):void

+ addSeatSelectListener(ActionListener al):void + addSignUpListener(ActionListener al):void

+ addNoSignUpListener(ActionListener al):void

+ addTheaterSelectListener(ActionListener al):void

+ addMakePaymentListener(ActionListener al):void

+ addCancelTicketListener(ActionListener al):void

+ addLoginListener(ActionListener al):void

+ addRegisterListener(ActionListener al):void

+ addMainMenuListener(ActionListener al):void

User

onlyinstance: User

- User()

+ getOnlyInstance(): User + getTicket(): Ticket

+ setTicket(Ticket): void

+ getMovie(): Movie + setMovie(Movie movie): void

+ makePayment(String creditCardNum, String exp, int cvv): void

OrdinaryUser

+ OrdinaryUser()

+ makePayment(String creditCardNum, String exp, int cvv): void

RegisteredUser

-id: int -firstName: String -lastName: String -email: String -address: String -cardFullName: String -cardNum: String -cardExp: String

-cardCVV: int -creditCodes: ArrayList<String> -password: String

+RegisteredUser(String firstName, String lastName, String email, String address, String cardName, String cardNum, String exp, int cvv)

+makePayment(String cardNum, String exp, int cvv): void

-hour: int -minute: int -movie: Movie -theater: Theater

+ShowTime(Movie m, Theater t)

ShowTime

-nameId: String -seatList: ArrayList<Seat>

+Theater(String name, int rows, int cols)

+getShowTimeList(): ArrayList<ShowTime>

+addSeat(int row, int col): void

+selectSeat(String id): Seat

Payment

-creditDiscount: Double

+pay(String creditCardNum, String exp, int cvv): void

-id: String -row: int

-col: int

+Seat(int row, int col)

-status: boolean

Theater

+createAllSeats(int rows, int cols): void

+removeSeat(int row, int col): void +setSeatStatus(boolean status): void

+changeSeatStatus(boolean status): void

-ticket: Ticket

+Payment(Ticket ticket, Double creditDiscount)

Movie

+Credit(Double balance)

Credit

Ticket

-name: String -releaseDate: String

-id: String

-user: User

-price: Double

-seat: Seat

-showtime: ShowTime

+getMovie(): Movie

-purchaseDate: LocalDateTime

+Ticket(User u, ShowTime st)

+setMovie(Movie m): void

+setTheater(Theater t): void

-code: String

-balance: Double

+getTheater(): Theater

+Movie(String name, Theater t, String releaseDate)

CancelView

+ confirmCancelListener(ActionListener al): void

+ menuButtonListener(ActionListener al): void

MainMenuView

-buyTicket: Jbutton -cancelTicket: Jbutton -searchMovie: Jbutton -login: Jbutton -register: Jbutton

+MainMenuView()

+addBuyTicketListener(ActionListener al): void +addCancelTicketListener(ActionListener al): void

+addSearchMovieListener(ActionListener al): void +addLoginListener(ActionListener al): void

AdminMovieView

+addRegisterListener(ActionListener al): void

-ConfirmUpdate:JButton -MainMenu:JButton -SearchBar: TextBox

-SearchButton:JButton -Showtimes:JTable

- InputButton : JButton - CancelButton : JButton

- UpdateButton : JButton

+AdminMovieView()

+ addSelectListener(ActionListener al):void

+ addInputListener(ActionListener al):void

+ addSearchListener(ActionListener al) :void

+ addCancelListener(ActionListener al):void - addUpdateListener(ActionListener al):void LoginView

emailAddress: JTextField password: JPasswordField emailAddressLabel: Jlabel passwordLabel: JLabel

login: JButton

cardNumber: JTextField

- cardHolderFName: JTextField

cardHolderLName: JTextField

cardHolderFNameLabel: JLabel

cardHolderLNameLabel: JLabel

+ addActionListener(ActionListener al):void

cardNumberLabel: JLabel

cardExpiryLabel: JLabel

cardCvvLabel: JLabel

submit: JButton

+ PaymentView()

cardExpiry: JTextField

cardCvv: JTextField

+ LoginView() - addActionListener(ActionListener al):void

PaymentView

firstName: JTextField lastName: JTextField

emailAddress: JTextField password: JPasswordField

firstNameLabel: JLabel

· lastNameLabel: JLabel

emailAddressLabel: Jlabel

passwordLabel: JLabel

enterPayment: JButton register: JButton

+ SignUpView()

SignUpView

+ addActionListener(ActionListener al): void

MovieView

· movieList: JList - movieScrollBar: JScrollBar - moviesLabel: Jlabel

+MovieView()

next: JButton

+ addActionListener(ActionListener al): void

TheaterView

-theaterList: ArrayList<Jbutton> -showtimeList:ArrayList<Jbutton>

-ConfirmCancel:JButton

-MainMenu:JButton

-Email: TextBox

+ CancelView()

-TicketNum: TextBox

+TheaterView(ShowTime st)

+setTheatersList(ArrayList<String> theaters): void

+getTheatersDB(ShowTime st): ArrayList<String> +addSelectTheaterListener(ActionListener al):void

+setShowTimeList(ArrayList<String> theaters): void +getShowTimeDB(ShowTime st): ArrayList<String> +addSelectShowTimeListener(ActionListener al):void

SeatView

-seats: ArrayList<ArrayList<String>> -rows: int

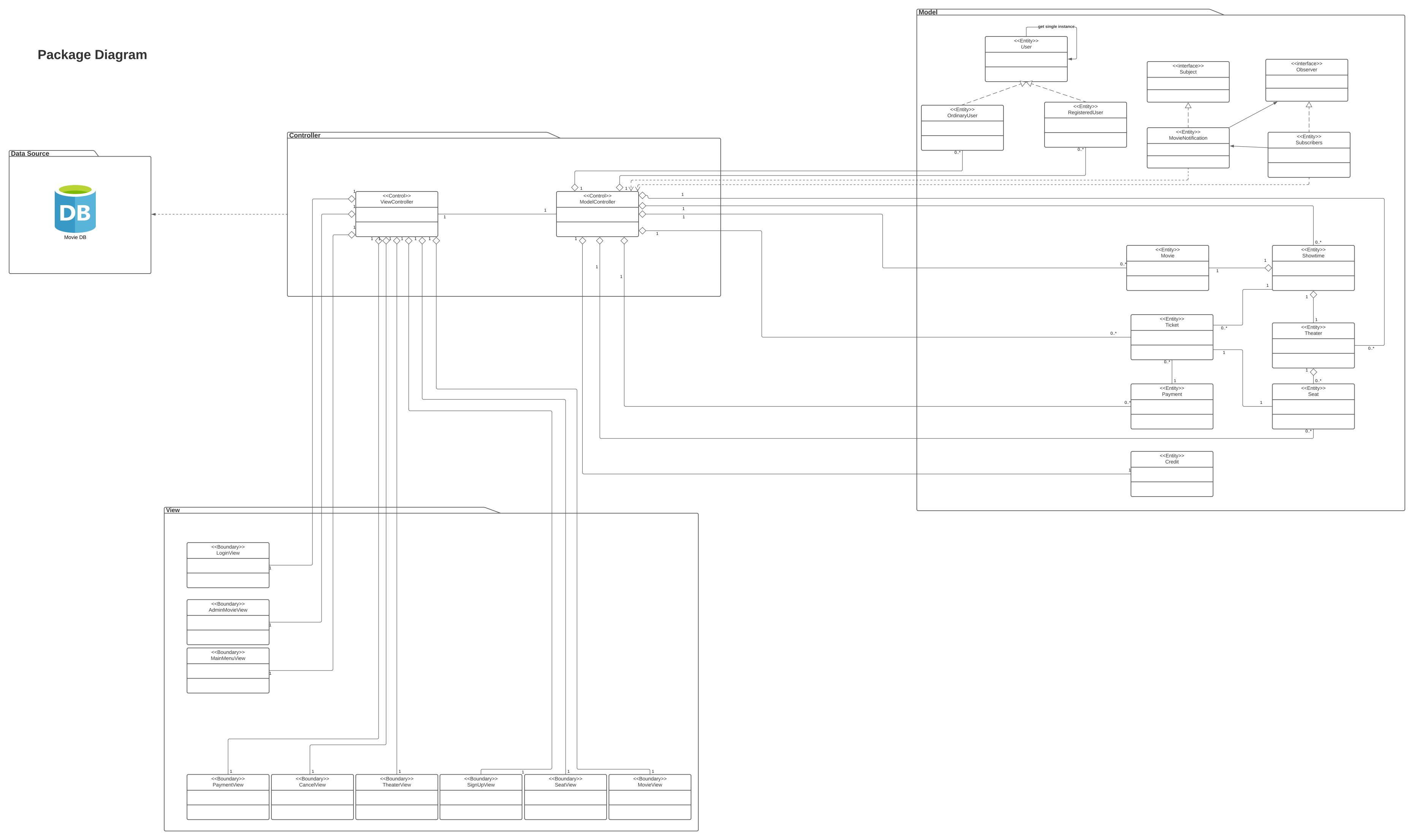
+SeatView(ShowTime st, Theater t)

-columns: int

-selectedSeat: ImageIcon -checkout: JButton

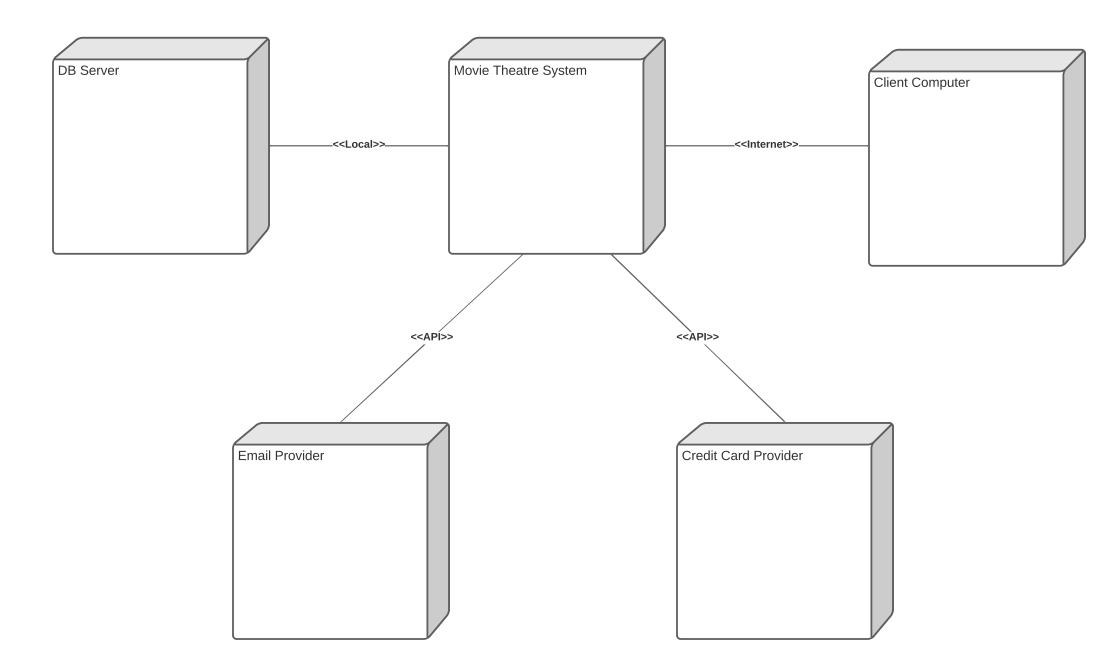
+getSeatsDB(ShowTime st, Theater t): ArrayList<ArrayList<String>> +addSelectSeatTheaterListener(ActionListener al):void

Package Diagram:



Deployment Diagram:

Deployment Diagram



Appendix - GUI Examples:

MainMenuView Snapshot



SeatView Snapshot

AVAILA	BLE SEATS					_ ×			
SCREEN									
A 1	A2	А3	A4	A 5	A 6	A7			
B1	B2	B 3	B4	B 5	B 6	B7			
C1	taken	C 3	C4	C 5	C6	C 7			
D1	D2	D 3	D4	D 5	D6	D 7			
E1	E2		E4	E 5	E 6	E 7			
F1	F2	F3	F4	taken	F 6	F 7			
G 1	G2	taken	G4	G 5	G6	G7			
H1	H2	Н3	H4	Н5	Н6	Н7			
I1	12	13			16	17			
J1	J2	J3	J4	J5	J6	J7			
Proceed to Checkout									