ENSF 614

FINAL PROJECT

Dec 4, 2022

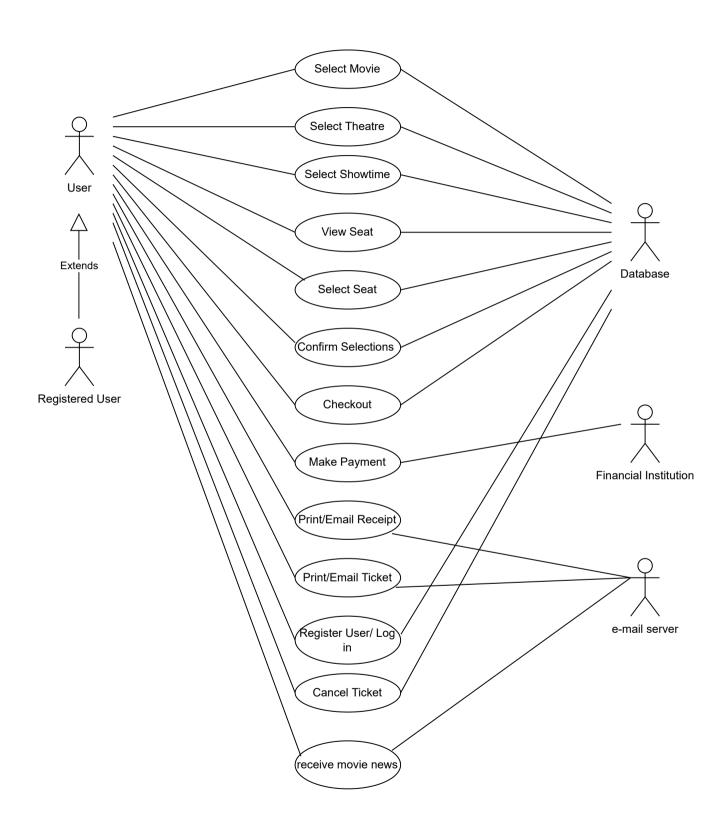
Group 5 Team Members:

Steve Andersen
Chinwe Ajieh
Muhammad Hussain
Mohammed Alhajjaj

Use Case Diagram

Movie Theatre Ticket Reservation App

Group 5



Use Case Scenarios

Select Movie:

The Home Page has a number of options – <u>The user can login</u> as a registered user, <u>sign up</u> to be a registered user, or <u>continue as a guest</u>

The user either <u>logs in or continues as a guest</u>.

<u>The user</u> can <u>search</u> for the movie he wants on the Home page and when he finds the one he wants, <u>The user</u> can click on the "<u>Buy Tickets"</u> button associated with the movie banner.

After The user selects <u>Buy Tickets</u>, the system proceeds to the "My Tickets" page

Select Theatre:

The user has either logged in or continues as a guest.

The user has clicked on the "Buy Tickets" button for the movie he wants to see.

The system takes <u>The user</u> to the "My Tickets" page which shows the <u>showtimes</u>, <u>locations</u> (<u>theatre</u>), <u>price</u>, available <u>seats</u>, and <u>offers</u> (if any).

The user selects the theatre location that he wishes to go to

Select Showtime:

The user has either <u>logged</u> in or <u>continues</u> as a guest.

The user has clicked on the "Buy Tickets" button for the movie he wants to see.

The system has taken <u>The user</u> to the "My Tickets" page which shows the <u>showtimes</u>, <u>locations</u> (theatre), price, available seats, and offers (if any).

The user selects the showtime that he wishes attend.

View Seat:

The user has either logged in or continues as a guest.

The user has clicked on the "Buy Tickets" button for the movie he wants to see.

The system has taken <u>The user</u> to the "My Tickets" page which shows the <u>showtimes</u>, <u>locations</u> (<u>theatre</u>), <u>price</u>, available <u>seats</u>, and <u>offers</u> (if any).

The user views the available seats for that specific theatre and showtime.

Select Seat:

The user has either <u>logged in or continues as a guest</u>.

<u>The user</u> has clicked on the "Buy Tickets" button for the <u>movie</u> he wants to see.

The system has taken <u>The user</u> to the "My Tickets" page which shows the <u>showtimes</u>, <u>locations</u> (<u>theatre</u>), <u>price</u>, available <u>seats</u>, and <u>offers</u> (if any).

The user <u>views</u> the available seats and then <u>selects</u> the specific seat he wants.

Confirm Selections:

After the user selects a movie, theatre, showtime and desired seat, the system asks the user to confirm selections. If the user chooses to proceed, the system indicates that the activity is complete and takes the user to the next step; else it takes the user back to the previous step if the user's choice is "not to proceed".

Checkout/ Make Payment:

After the <u>user confirms</u> his <u>selection</u>, the <u>user</u> enters details to checkout as a guest or log in as a registered <u>user</u> (if not logged in earlier). The <u>user selects checkout</u> and <u>make payment</u>, and the <u>system</u> proceeds to the <u>payment page</u>. The <u>user</u> is asked to enter his payment details which are transmitted to the <u>Billing Institution</u>. User <u>gets a confirmation message</u> to indicate if <u>payment</u> was successful or not. If payment was successful, <u>ticket and receipt</u> are <u>generated</u>.

Print/Email Receipt:

Once payment is <u>confirmed</u> by the <u>Billing Institution</u>, a <u>payment receipt</u> is <u>generated</u>. The <u>system communicates</u> with the <u>email server</u> and the <u>receipt</u> is automatically <u>sent</u> to the user's <u>email</u> provided during checkout or registration. The <u>user</u> can choose to <u>print the receipt</u>.

Print/Email Ticket:

<u>User</u> is <u>logged in</u> as a <u>registered user</u> or is a <u>guest</u>.

From a <u>view</u> of <u>available movies</u>, <u>User selects</u> the <u>movie</u> he <u>wishes to see</u> and <u>presses</u> the "<u>Buy</u> Tickets" button.

User <u>selects</u> his desired theatre from a list of available theatres.

User <u>selects</u> his desired showtime from a list of available showtimes.

User <u>selects</u> his desired seat from a view of seats, which <u>indicate</u> if they are available or not.

User confirms his selections and checks out.

User <u>makes</u> a payment for his tickets.

User can <u>select</u> if he would like to <u>receive</u> his tickets printed, emailed or both.

The system prints User's tickets if he selects printed/both.

If <u>User</u> has <u>selected emailed/both</u>, the <u>system</u> will <u>prompt</u> <u>User</u> for his <u>email</u> if he is a <u>guest</u>. The <u>system emails</u> <u>User's tickets</u> if he selects <u>emailed/both</u>.

Register User:

<u>User selects</u> the <u>option</u> to <u>register user</u> from the <u>Home Page</u>.

The <u>system</u> asks <u>User</u> to <u>input</u> his <u>email</u>, <u>name</u>, a <u>password</u>, <u>address</u> and <u>credit/debit card</u> <u>information</u>. <u>User inputs</u> this information.

The system sends User an email confirming his registration.

Login:

<u>User</u> selects the <u>option</u> to <u>login</u> from the <u>Home Page</u>.

The <u>system asks User</u> his <u>email</u> and <u>password</u>. <u>User inputs</u> this <u>information</u>.

The <u>system changes</u> the <u>view</u> of the <u>Home Page</u> to that of a <u>registered user</u>, and <u>loads User's</u> information.

Cancel Ticket:

<u>User selects</u> the <u>option</u> to <u>cancel ticket</u> from the <u>Home Page</u>.

The system asks User for his ticket number. User inputs the information.

The system checks if the show is in less than 72 hours.

If the show is less than 72 hours, displays error that ticket can't be cancelled.

If the <u>show</u> is not in less than <u>72 hours</u>, the <u>system checks</u> if the <u>ticket</u> was <u>purchased</u> by a <u>registered user</u>.

If the <u>ticket</u> was <u>purchased</u> by a <u>registered user</u>, the <u>system</u> asks <u>User</u> to <u>confirm</u> the <u>cancellation</u>. <u>User confirms it</u>.

If the <u>ticket</u> was not <u>purchased</u> by a <u>registered user</u>, the <u>system displays a warning</u> that <u>User</u> will only <u>receive</u> a <u>credit</u> valid for <u>one year</u> with an <u>administration fee</u> of 15%. The <u>system</u> asks <u>User</u> to <u>confirm</u> his <u>cancellation</u> and <u>User confirms it.</u>

Receive Movie News:

The <u>system sends movie news</u> to User's email.

The system displays movie news to User on the Home Page as a notification if User is logged in.

Candidate Object List:

The user

movie

Seats

showtime

theatre

receipt

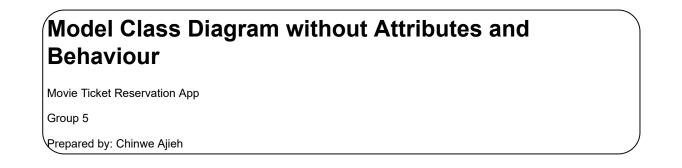
tickets

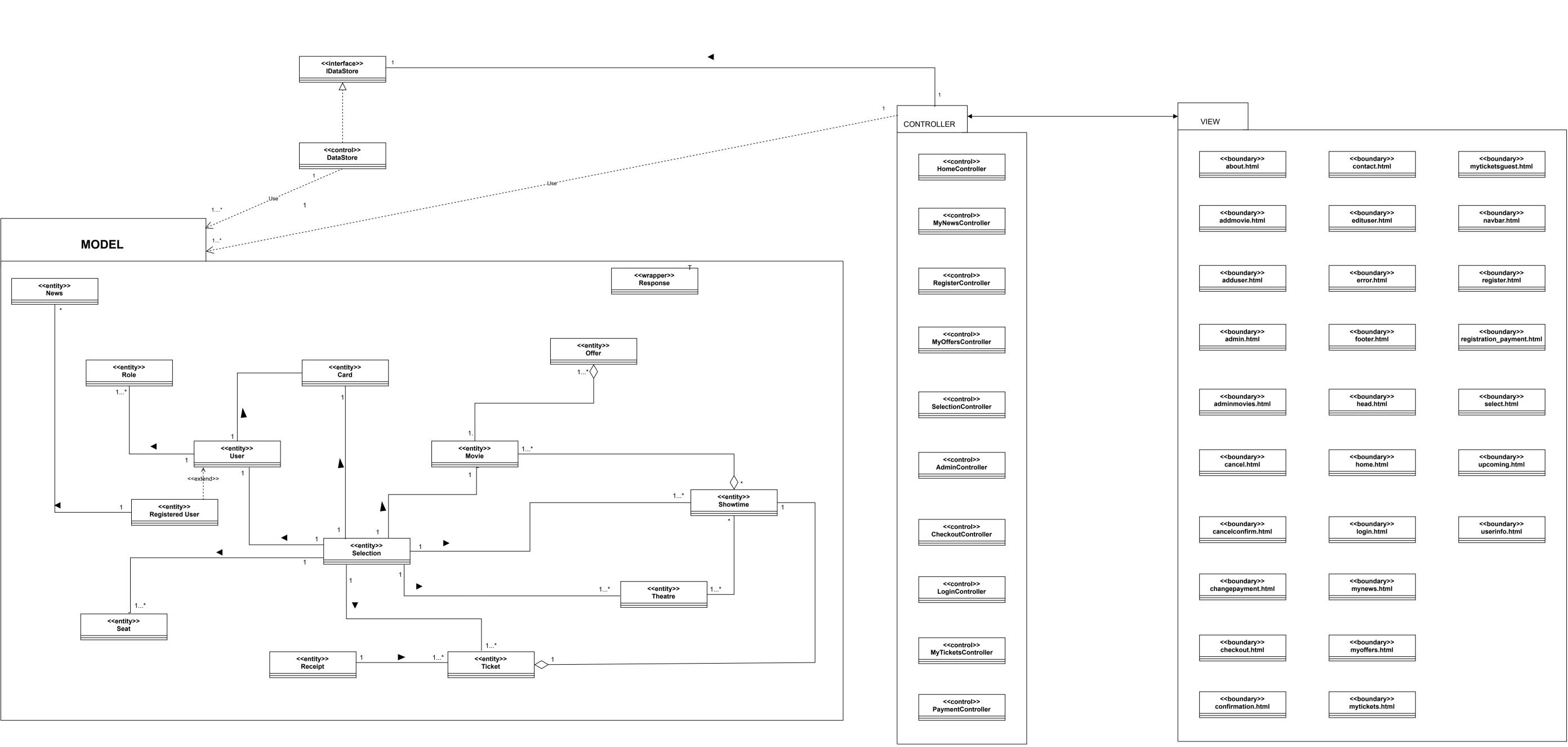
e-mail

billing institution

payment

movie news



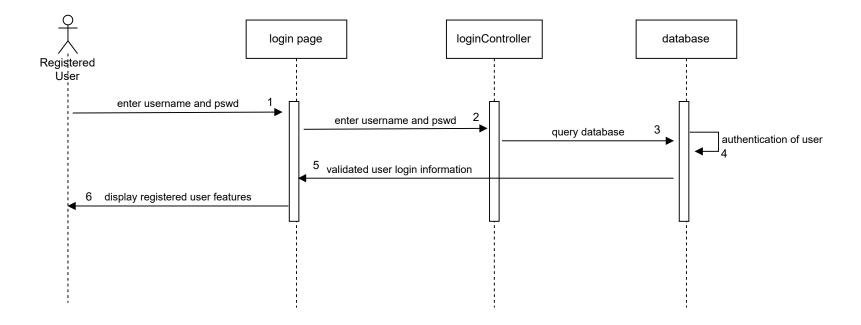


Model Class Diagram without Relationships, and with Attributes and Behaviour Movie Ticket Reservation App Group 5 <<entity>> <<entity>> <<entity>> <<entity>> <<entity>> <<entity>> Selection Theatre **Showtime** User Role Movie Seat -id: Integer -user: User -id: int -id: int -id: int -id: int -id: int | -title: String -date: String -firstName: String -selectedTheater: Theater -name: String -theaterId: int -name: String -lastName: String -selectedMovie: Movie -description: String -address: String -time: String -row number: int -selectedShowtime: Showtime -genre: String -theater: Theater -email: String -city: String -seat_number: int -selectedSeatsString: String[] -movie: Movie -address: String -rating: String -province: String -status: String -city: String -selectedTickets: ArrayList<Ticket> -releaseDate: String -postalCode: String ~selectedSeats: String[] -price: Double -card: Card -province: String -duration: String -phone: String +theaters: ArrayList<Theatre> -trailer: String email: String -postalCode: String -phone: String +showtimes: ArrayList<Showtime> -poster: String -website: String +seats: ArrayList<Seat> -password: String -status: String ~showTimes: ArrayList -defaultCard: Card -totalPrice: double -roles: HashSet<Role> -rolStrings: String[] +isEnabled(): boolean +getAuthorities(): Collection <<wrapper>> <<entity>> <<entity>> <<entity>> <<entity>> <<entity>> Response News **Ticket** Card Receipt -image: String -id: int -data: T -id: int -id: int -id: int -message: String -title: String -userld: int -userId: int -date: String -userId: int -success: boolean -theNews: String -price: Double -showtime: Showtime -cardNumber: String -time: String -offer: String -seat: Seat -cardHolderName: String -movieName: String -quantity: int -price: Double -expiryDate: String -theatreName: String -movie: Movie -status: String -cvv: String -userEmail: String -price: double ~tickets: ArrayList + method(type): type Controller Class Diagram without Relationships, and with Attributes and Behaviour <<control>> <<control>> <<control>> <<control>> **HomeController** MyNewsController RegisterController MyOffersController -dataStore: IDataStore ~dataStore: IDataStore ~dataStore: IDataStore ~dataStore: IDataStore +myNewsPage(model: Model): String +registerPage(model: Model): String +myOffersPage(model: Model): String +paymentPage(user: User, session: HttpSession, model: Model): String +home(model: Model): String +registerUser(myuser: User, session: HttpSession, model: Model): String +home2(model: Model): String +saverUser(model: Model, session: HttpSession, myselection: Selection): String +upcoming(model: Model): String +search(query: String, model: Model): String <<control>> <<control>> <<control>> <<control>> CheckoutController SelectionController AdminController LoginController ~dataStore: IDataStore ~dataStore: IDataStore ~dataStore: IDataStore ~dataStore: IDataStore ~emailService: EmailService +login(request: HttpServletRequest, model: Model): String +loginError(model: Model): String +showCheckoutPage(session: HttpSession, model: Model): String +buyTickets(session: HttpSession, model: Model, movield: int, mySelection: Selection): String +adminPage(model: Model): String +logout(model: Model): String +checkoutPage(session: HttpSession, model: Model): String +userInfo(session: HttpSession, model: Model, myselection: Selection): String +addUser(model: Model): String +addUser(user: User, model: Model): String +editUser(id: int, model: Model): String +editUser(user: User, model: Model): String +showAllMovies(model: Model): String +addMovie(model: Model): String +addMovie(movie: Movie, model: Model): String +deleteMovie(model: Model, id: int): String +deleteUser(model: Model, id: int): String +editRoles(model: Model, user: User): String <<control>> <<control>> MyTicketsController **PaymentController** ~dataStore: IDataStore ~dataStore: IDataStore +changePayment(model: Model, defaultCard: Card, session: HttpSession): String +myTicketsPage(model: Model): String +changePaymentPost(model: Model, card: Card, session: HttpSession): String Data Control Class Diagram without Relationships, and with Attributes and Behaviour <<interface>> <<control>> **IDataStore DataStore** -env: Environment -getStatement(): Statement -getPreparedStatement(query: String): PreparedStatement -getStatement(): Statement -checkSQLInjection(str: String): boolean -getPreparedStatement(query: String): PreparedStatement +registerUser(user: User): Response -checkSQLInjection(str: String): boolean +addUser(user: User): Response +registerUser(user: User): Response -checkEmailExists(email: String): boolean +addUser(user: User): Response +loginUser(email: String, password: String): Response -checkEmailExists(email: String): boolean +logoutUser(user: User): boolean +loginUser(email: String, password: String): Response +getAllMovies(): Response> +logoutUser(user: User): boolean +getAllShowingMovies(): Response> +getAllMovies(): Response> +getUpcomingMovies(): Response> +getAllShowingMovies(): Response> +getTheatersByMovie(movie: Movie): Response> +getUpcomingMovies(): Response> +getShowtimesByMovieAndTheater(movie: Movie, theater: Theater): Response> +getTheatersByMovie(movie: Movie): Response> +getSeatsByShowtime(showtime: Showtime): Response> +getShowtimesByMovieAndTheater(movie: Movie, theater: Theater): Response> +getUserByEmail(email: String): User +getSeatsByShowtime(showtime: Showtime): Response> -getUserRoles(user: User): Set +getUserByEmail(email: String): User +getMovie(movield: int): Response -getUserRoles(user: User): Set +getShowtimes(movield: int, theaterId: int): Response> +getMovie(movield: int): Response +getAllTheaters(movield: int): Response> +getShowtimes(movield: int, theaterId: int): Response> +getShowtime(showtimeId: int): Response +getAllTheaters(movield: int): Response> +getSeats(movield: int, showtimeld: int): Response> +getShowtime(showtimeld: int): Response +getSeat(seatId: int): Response +getSeats(movield: int, showtimeld: int): Response> +addTickets(tickets: ArrayList, userId: int): Response> +getSeat(seatId: int): Response +getTicketsByUserEmail(name: String): Response> +addTickets(tickets: ArrayList, userId: int): Response> +getMoviesNews(): Response> +getTicketsByUserEmail(name: String): Response> getDefaultCard(user: User): Response +getMoviesNews(): Response> +isEmailInUse(email: String): boolean +getDefaultCard(user: User): Response +checkSeats(selectedShowtimeId: int, selectedSeatsString: String[]): boolean +isEmailInUse(email: String): boolean rifSeatsExceedsTenPercentage(movield: int, theaterId: int, showtimeId: int, selectedSeatsString: String[]): boolean +checkSeats(selectedShowtimeld: int, selectedSeatsString: String[]): boolean +updateUserPayment(userId: int, card: Card): Response +ifSeatsExceedsTenPercentage(movield: int, theaterId: int, showtimeId: int, selectedSeatsString: String[]): boolean +searchMovies(query: String): Response> +updateUserPayment(userId: int, card: Card): Response +processPayment(selection: Selection): Response +searchMovies(query: String): Response> +addMovie(movie: Movie): Response +processPayment(selection: Selection): Response +deleteMovie(id: int): Response +addMovie(movie: Movie): Response +deleteUser(id: int): Response +deleteMovie(id: int): Response +getAllUsers(): Response> +deleteUser(id: int): Response +getDefaultCard(id: int): Response +getAllUsers(): Response> +editRoles(user: User): Response +getDefaultCard(id: int): Response +getAllRoles(): ArrayList +editRoles(user: User): Response +getUser(id: int): Response +getAllRoles(): ArrayList +updateUser(user: User): Response +getUser(id: int): Response +getOffers(): Response> +updateUser(user: User): Response +getOffers(): Response> View Class Diagram without Relationships, and with Attributes and Behaviour <<body>
<
boundary>></br/> <<body><<body><
 <<body><<body> <<body><<body> <<body><<body> <<body><<body><
<
 about.html contact.html myticketsguest.html navbar.html userinfo.html upcoming.html select.html <<body>
<
boundary>></br/> <<body><<body> <<body><<body> <<body><<body> <<body><<body> <<body><<body><
 <<body><<body><
 registration_payment.html register.html edituser.html mytickets.html myoffers.html mynews.html login.html <<body><<body><
 <<body><<body><
 <<body><<body> <<body><<body> <<body><<body> <<body><<body> <<body><<body><
<
 home.html head.html footer.html error.html addmovie.html confirmation.html checkout.html <<body><<body><
 <<body><<body><
 <<body><<body> <<body><<body> <<body><<body><
 <<body><<body> cancelconfirm.html cancel.html adminmovies.html admin.html adduser.html changepayment.html

System Interaction Diagram

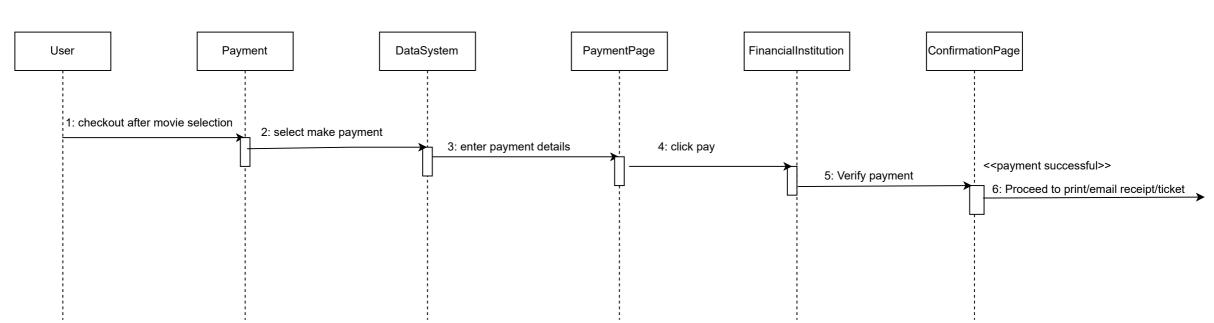
Registered User Login

Steve Andersen



System Interaction Diagram for Use Case- Make Payment

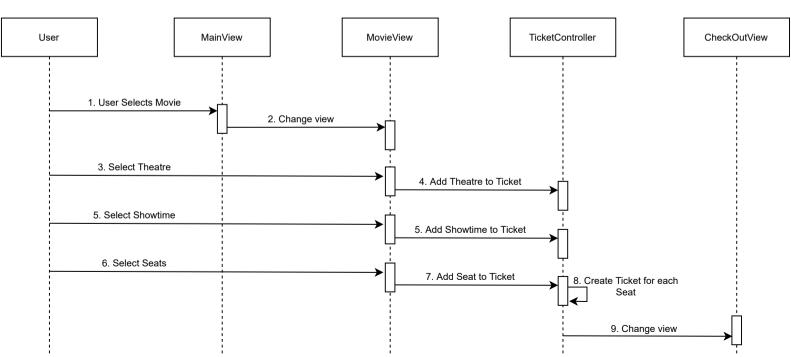
Created by: Chinwe Ajieh



System Interaction Diagram - Ticket Movie Ticket Reservation App Group 5

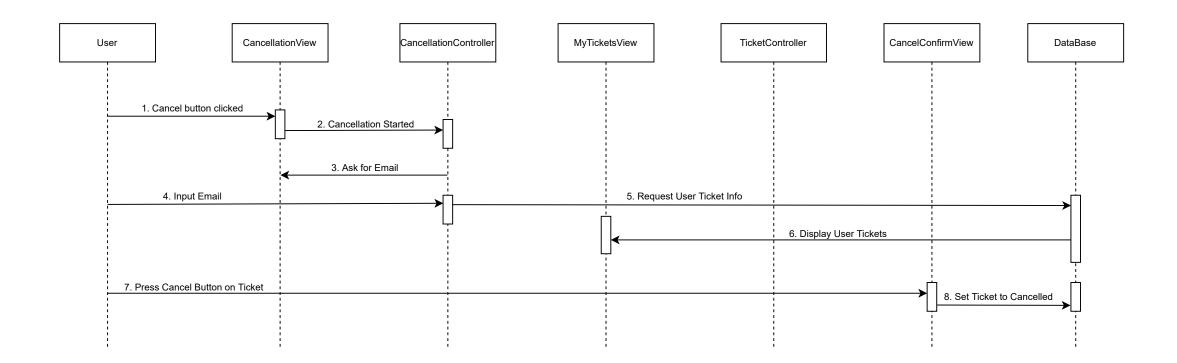
Prepared By:

Mohammed Alhajjaj



System Interaction Diagram - Cancellation Movie Ticket Reservation App Group 5 Prepared By:

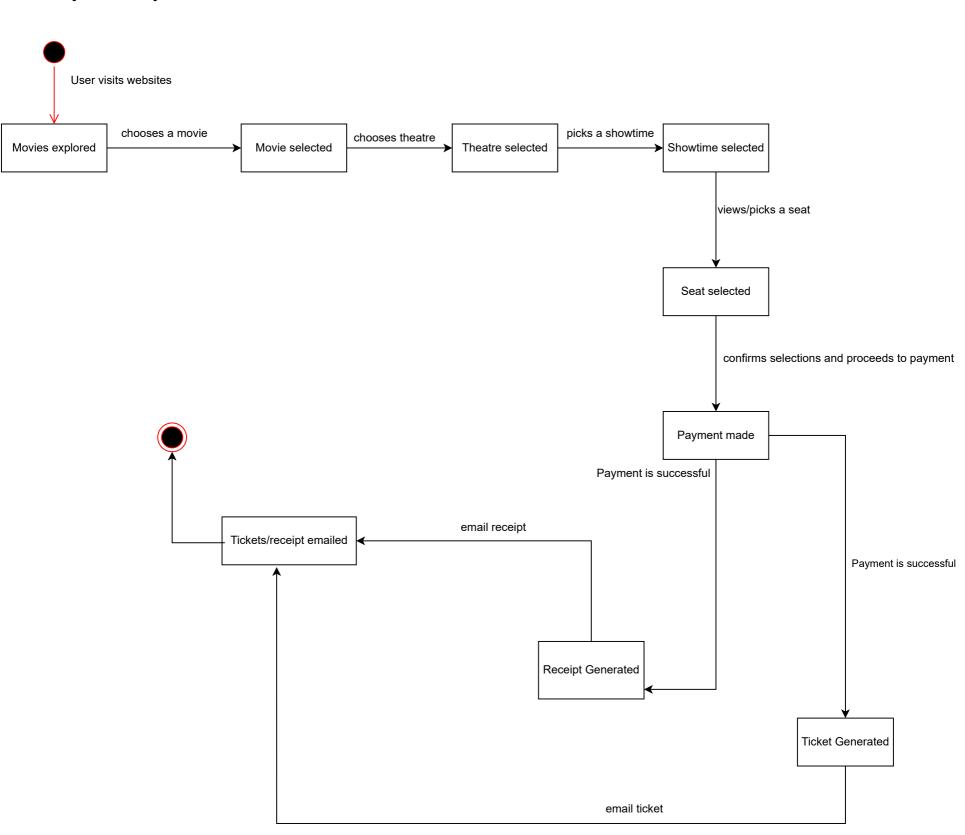
Muhammad Fahad Hussain

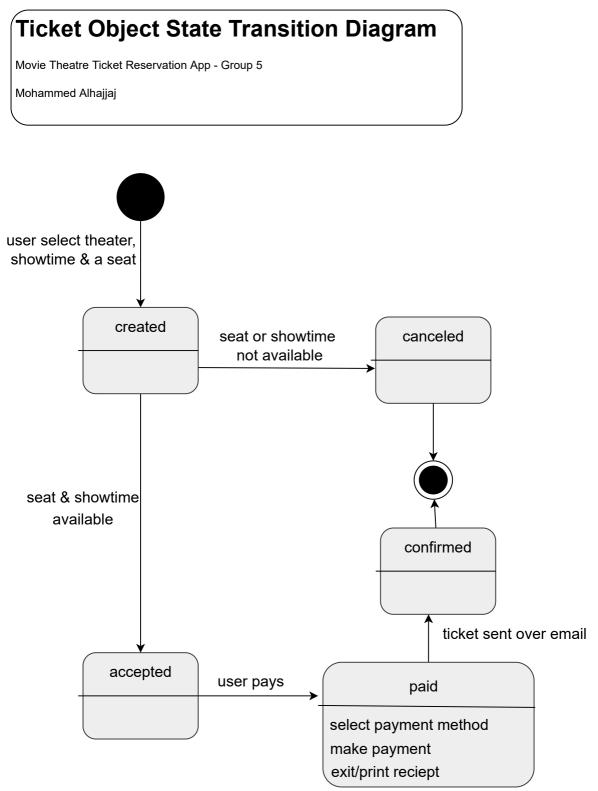


State Transition Diagram

for Use Case - Select Movies

Created by: Chinwe Ajieh

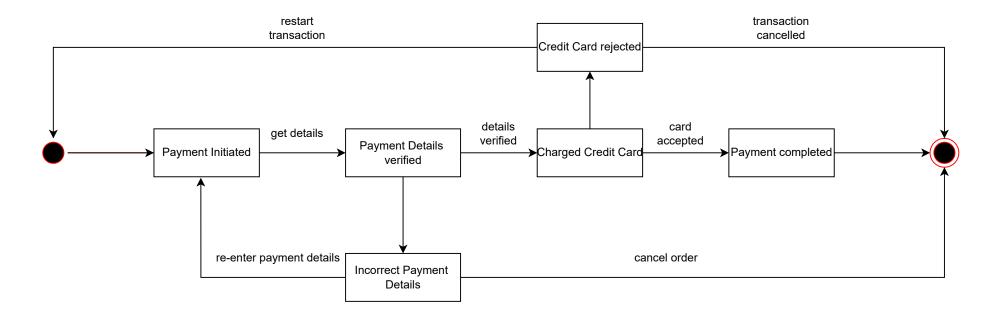




State Transition Diagram

Payment

Steve Andersen



State Transition Diagram - Ticket Cancellation Movie Ticket Reservation App Group 5 Prepared By: Muhammad Fahad Hussain *Please note that the states shown are for the ticket

