Course: ENSF 614 – Fall 2022

Final Project

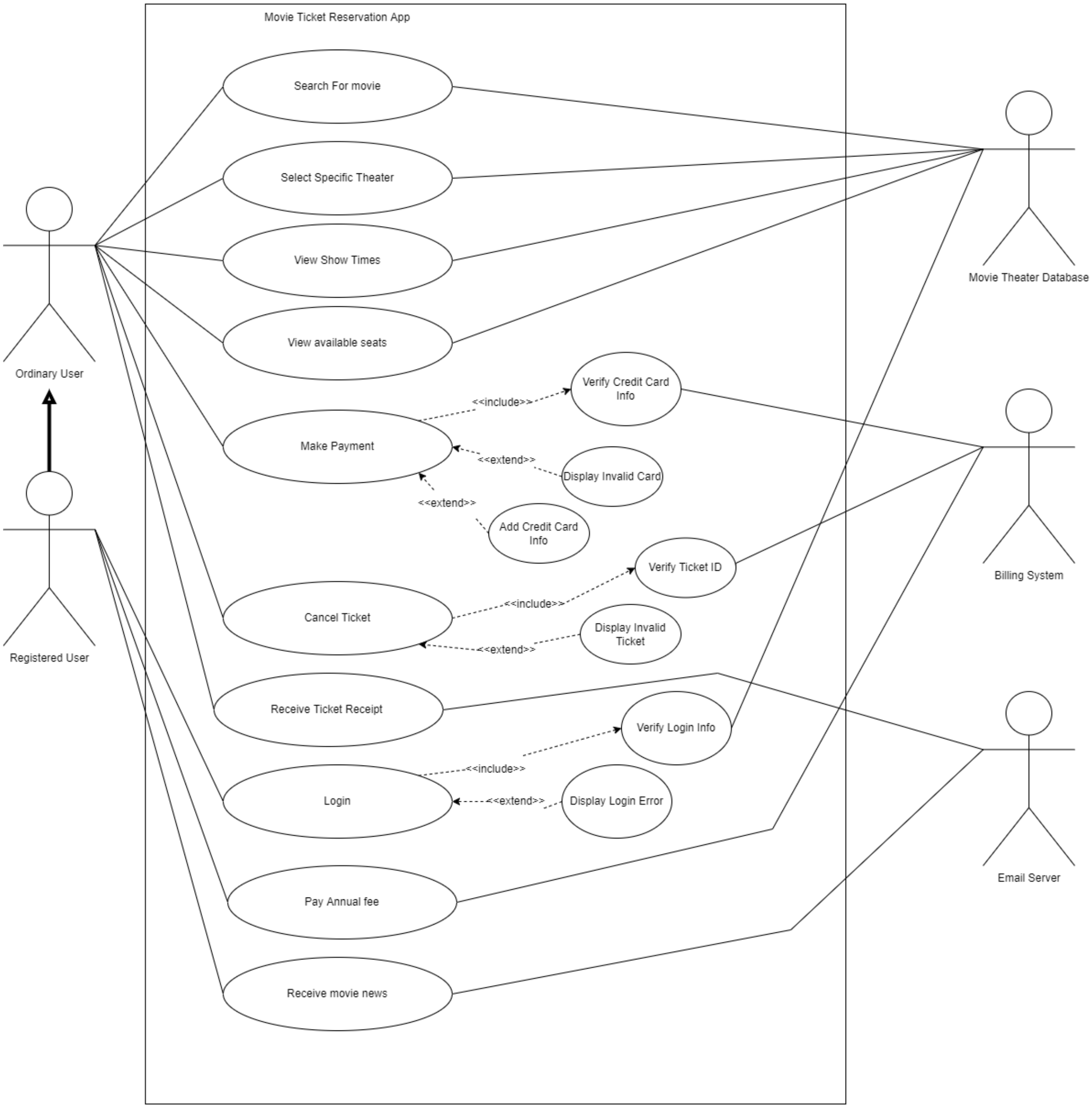
Instructor: Mahmood Moussavi

Student Names: Chansuck Lee, Temiloluwa Bakare, Feras Dahrooge, Okeoghenemarho Obuareghe

Submission Date: December 3, 2022

System Use-Case Diagram

With Extend And Include



Use-case Scenarios

Scenario: Use Case “Select Theatre”
The user will click the “Select Theatre” tab. The Select Theatre page will pull the list of theaters from the database server and display the list of theaters on the page. The user in this scenario can browse through the theaters and selects one of the theaters

Scenario: Use Case “Select Movie”
This scenario starts after the use case “Select Theatre” is completed. The Movie List Page will pull the list of movies from the database server and display the movies on the page. The user will browse through the movie list page and select one of the movies.

Scenario: Use Case “View ShowTimes”
This scenario starts after the use case “Select Movie” is completed. The View Showtimes Page will pull the list of showtimes from the database server and display the showtimes on the page. The user in this scenario can browse through the available showtimes and select one of the showtimes.

Scenario: Use Case “View Available Seats”
This scenario starts after the use case “View ShowTimes” is completed. The Available Seats Page will pull the list of available seats from the database server and display the available seats on the page. The user can browse through the Available Seats and select one of the seats.

Scenario: Use Case “Make Payment”
This scenario starts after the use case “Select Desired Seat” is completed. An ordinary user first adds their credit card information and the information will be validated by the Financial System Server and then the transaction is confirmed and finalized. A registered user will use the saved credit card info and the transaction will be finalized by the Financial System Server.

Scenario: Use Case “Receive Ticket Receipt”
The scenario starts after the use case “Make Payment” has been Terminated. The payment receipt and movie ticket are sent to the User’s email address by the Email Server.

Scenario: Use Case “Cancel Ticket”
This scenario could be executed only after ticket payment is successful. The User will select Cancel_Ticket_Tab. The user will input the ticket ID that they want to cancel. The ticket_ID will be verified by the Database Server. The Ordinary Users get a credit with a 15% administration fee, for future purchase, whereas, the Registered Users get full credit with no fee.

Scenario: Use Case “Login”
This scenario starts when a registered user clicks the login button. The user will input their login information and it will be checked if it exists in the Movie Theatre Database. If it exists, it will login to the person’s account and display the main page. If it does not exist, it will display the login page again.

Scenario: Use Case “Pay Annual Fee”
This scenario starts when an ordinary user becomes a registered user . The user will pay an annual fee automatically through the Financial System Server using their credit card information.

Scenario: Use Case “Receive Movie News”
This scenario starts after the use case “Make Payment” is completed. The registered user receives an email about upcoming movies from the Email Server. The user can look through the email and see if they want to watch an upcoming movie.

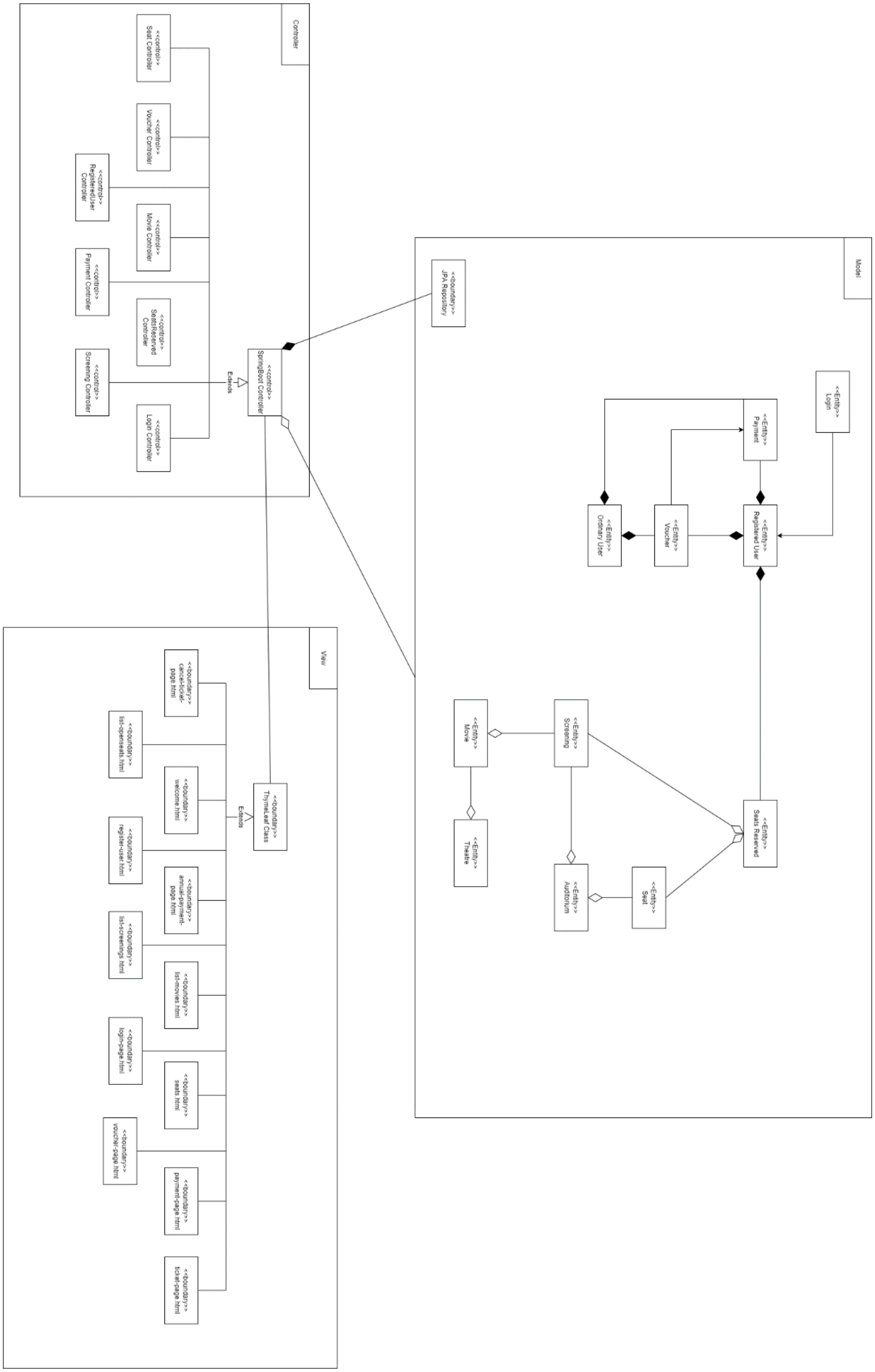
List of Candidate Objects

Noun Filtering Decision

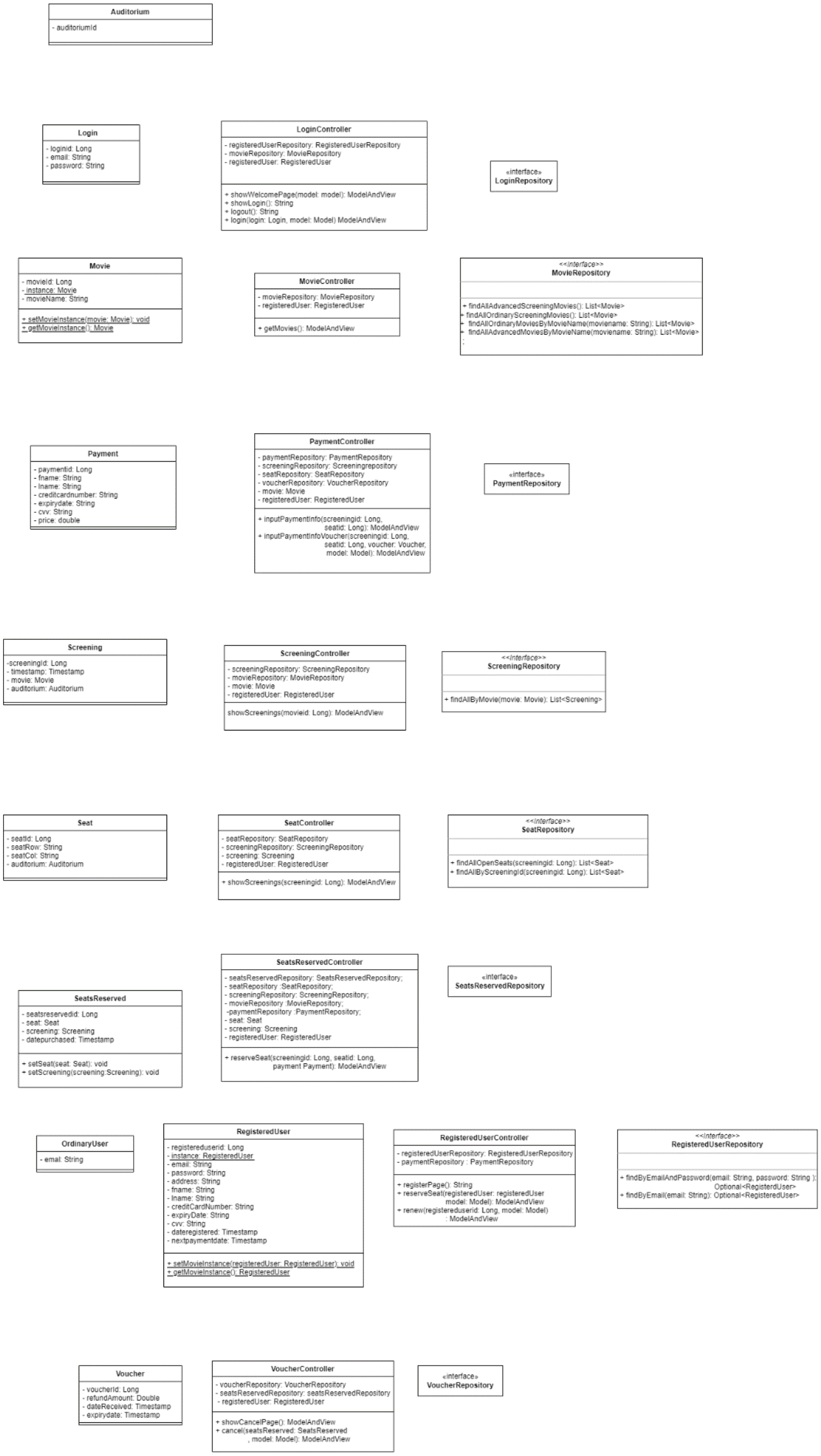
Noun	**Filtering Decision**
User	Filtered(actor)
Theatres	Candidate Object
Database Server	Filtered (actor)
List of Theatres	Candidate object
Movie List	Candidate object
Movie	Candidate object
Showtime	Candidate Object
Available seats	Candidate Object
Ordinary user	Type of User
Registered user	Type of user
Credit-card	Property of user
Financial system server	actor
Payment receipt	Candidate Object
Movie ticket	Candidate object
Email server	actor
Ticket id	Property fo movie ticket
Movie theatre DB	actor
Email server	actor
Email	Property of user

Movie theatre DB	actor
User account	object
Annual Fee	Property of user account

Class Diagram with Relationships

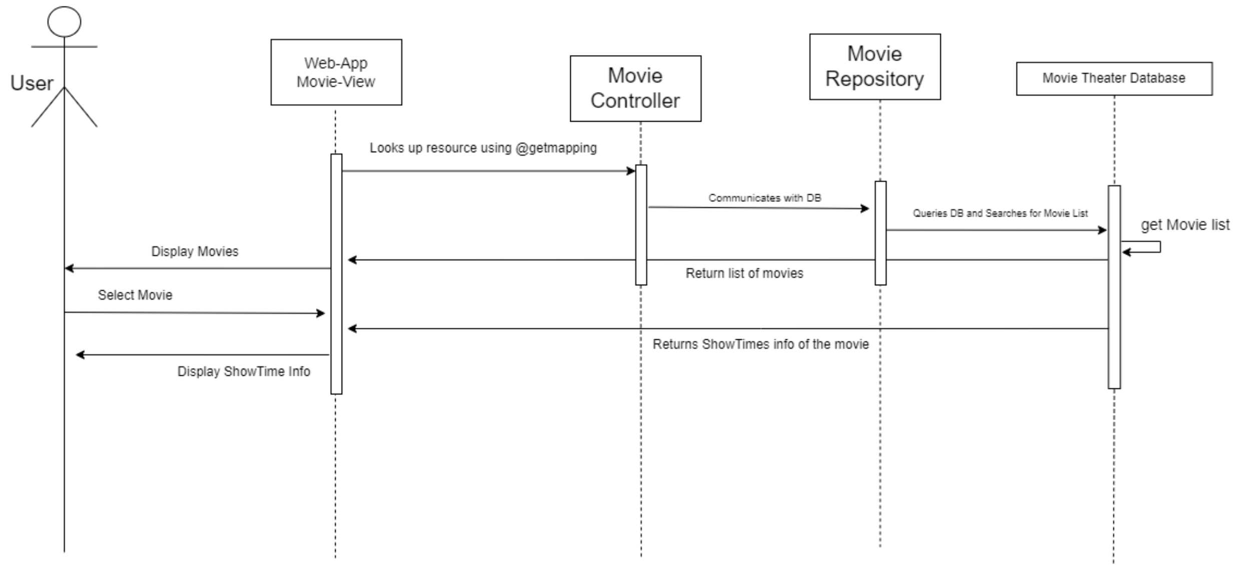


Class Diagram Without Relationships


Auditorium
- auditoriumId
Login
- loginid: Long
- email: String
- password: String
LoginController
- registeredUserRepository: RegisteredUserRepository
- movieRepository: MovieRepository
- registeredUser: RegisteredUser
+ showWelcomePage(model: model): ModelAndView
+ showLogin(): String
+ logout(): String
+ login(login: Login, model: Model) ModelAndView
«interface»
LoginRepository
Movie
- movieId: Long
- instance: Movie
- movieName: String
+ setMovieInstance(movie: Movie): void
+ getMovieInstance(): Movie
MovieController
- movieRepository: MovieRepository
- registeredUser: RegisteredUser
+ getMovies(): ModelAndView
<<Interface>>
MovieRepository
+ findAllAdvancedScreeningMovies(): List<Movie>
+ findAllOrdinaryScreeningMovies(): List<Movie>
+ findAllOrdinaryMoviesByMovieName(moviename: String): List<Movie>
+ findAllAdvancedMoviesByMovieName(moviename: String): List<Movie>
;
Payment
- paymentid: Long
- fname: String
- lname: String
- creditcardnumber: String
- expirydate: String
- cvv: String
- price: double
PaymentController
- paymentRepository: PaymentRepository
- screeningRepository: Screeningrepository
- seatRepository: SeatRepository
- voucherRepository: VoucherRepository
- movie: Movie
- registeredUser: RegisteredUser
+ inputPaymentInfo(screeningid: Long,
seatid: Long): ModelAndView
+ inputPaymentInfoVoucher(screeningid: Long,
seatid: Long, voucher: Voucher,
model: Model): ModelAndView
«interface»
PaymentRepository
Screening
-screeningId: Long
- timestamp: Timestamp
- movie: Movie
- auditorium: Auditorium
ScreeningController
- screeningRepository: ScreeningRepository
- movieRepository: MovieRepository
- movie: Movie
- registeredUser: RegisteredUser
showScreenings(movieid: Long): ModelAndView
<<Interface>>
ScreeningRepository
+ findAllByMovie(movie: Movie): List<Screening>
Seat
- seatId: Long
- seatRow: String
- seatCol: String
- auditorium: Auditorium
SeatController
- seatRepository: SeatRepository
- screeningRepository: ScreeningRepository
- screening: Screening
- registeredUser: RegisteredUser
+ showScreenings(screeningid: Long): ModelAndView
<<Interface>>
SeatRepository
+ findAllOpenSeats(screeningid: Long): List<Seat>
+ findAllByScreeningId(screeningid: Long): List<Seat>
SeatsReserved
- seatsreservedid: Long
- seat: Seat
- screening: Screening
- datepurchased: Timestamp
+ setSeat(seat: Seat): void
+ setScreening(screening:Screening): void
SeatsReservedController
- seatsReservedRepository: SeatsReservedRepository;
- seatRepository :SeatRepository;
- screeningRepository: ScreeningRepository;
- movieRepository :MovieRepository;
-paymentRepository :PaymentRepository;
- seat: Seat
- screening: Screening
- registeredUser: RegisteredUser
+ reserveSeat(screeningid: Long, seatid: Long,
payment Payment): ModelAndView
«interface»
SeatsReservedRepository
OrdinaryUser
- emal: String
RegisteredUser
- registereduserid: Long
- instance: RegisteredUser
- email: String
- password: String
- address: String
- fname: String
- lname: String
- creditCardNumber: String
- expiryDate: String
- cvv: String
- dateregistered: Timestamp
- nextpaymentdate: Timestamp
+ setMovieInstance(registeredUser: RegisteredUser): void
+ getMovieInstance(): RegisteredUser
RegisteredUserController
- registeredUserRepository: RegisteredUserRepository
- paymentRepository : PaymentRepository
+ registerPage(): String
+ reserveSeat(registeredUser: registeredUser
model: Model): ModelAndView
+ renew(registereduserid: Long, model: Model)
: ModelAndView
<<Interface>>
RegisteredUserRepository
+ findByEmailAndPassword(email: String, password: String):
Optional<RegisterdUser>
+ findByEmail(email: String): Optional<RegisteredUser>
Voucher
- voucherId: Long
- refundAmount: Double
- dateReceived: Timestamp
- expirydate: Timestamp
VoucherController
- voucherRepository: VoucherRepository
- seatsReservedRepository: seatsReservedRepository
- registeredUser: RegisteredUser
+ showCancelPage(): ModelAndView
+ cancel(seatsReserved: SeatsReserved
, model: Model): ModelAndView
«interface»
VoucherRepository

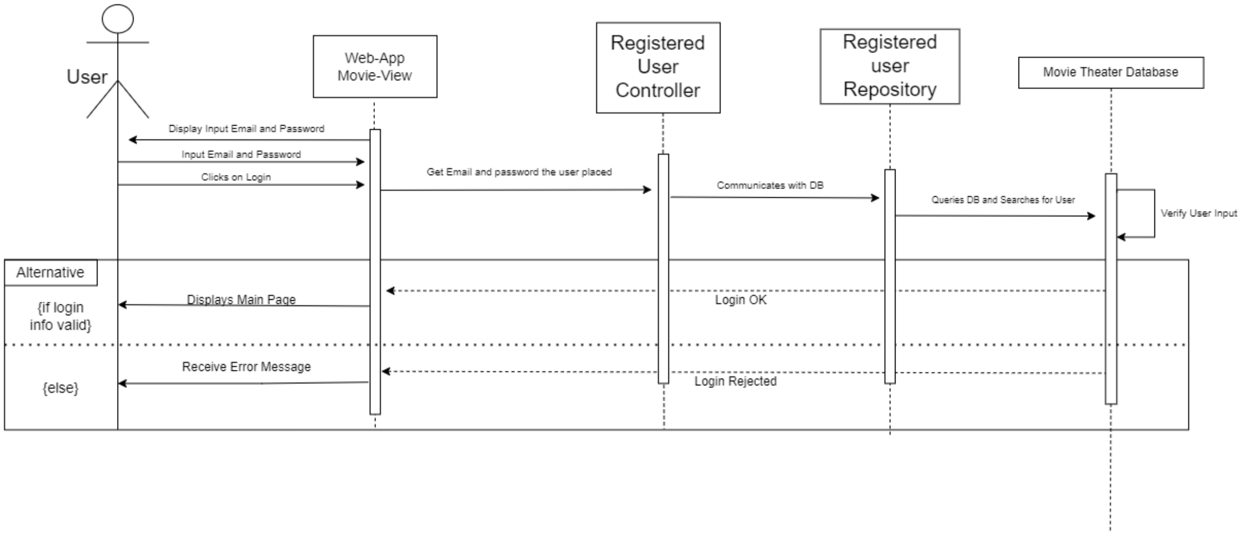
Interaction Diagrams

Movie Selection



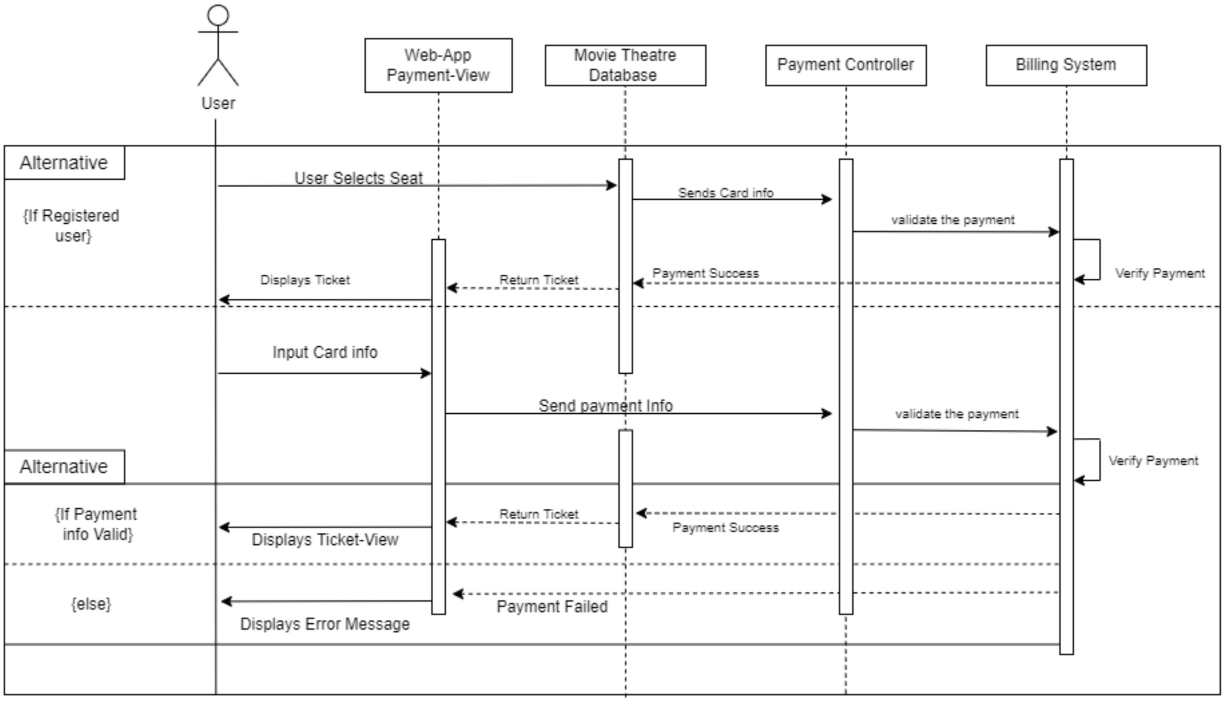
Chansuck Lee

Login



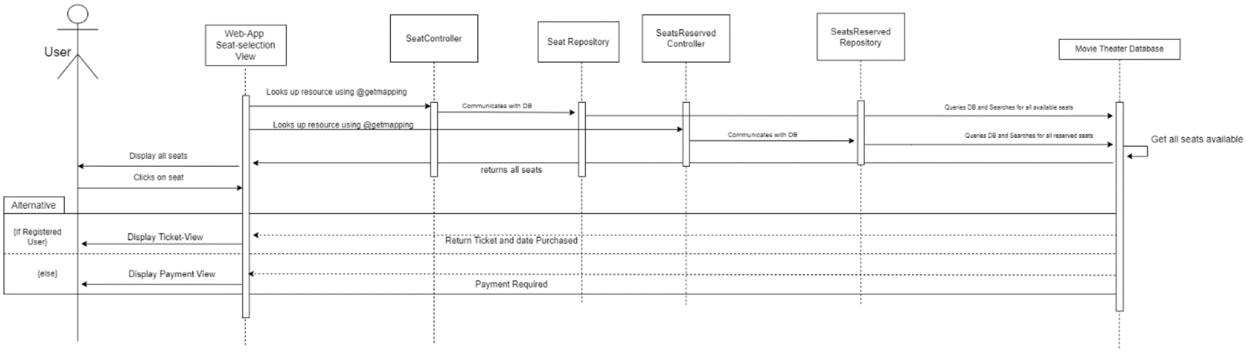
Temiloluwa Bakare

Ticket Payment



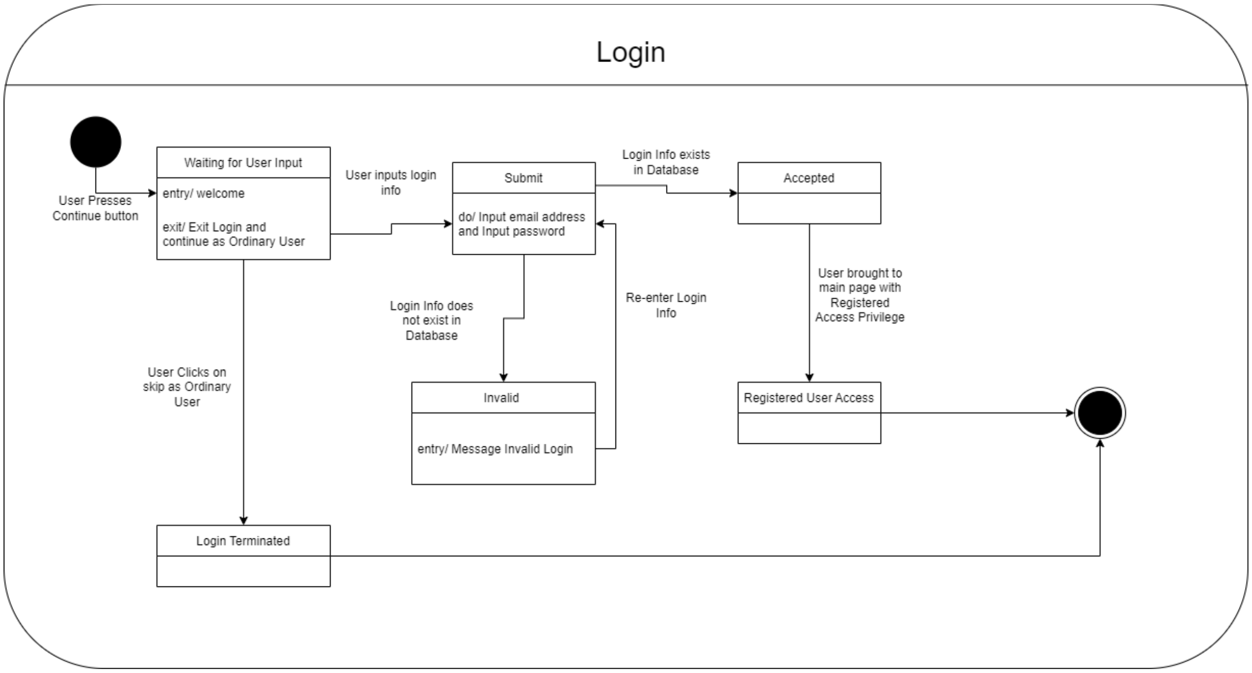
Feras Dahrooge

Seat Selection



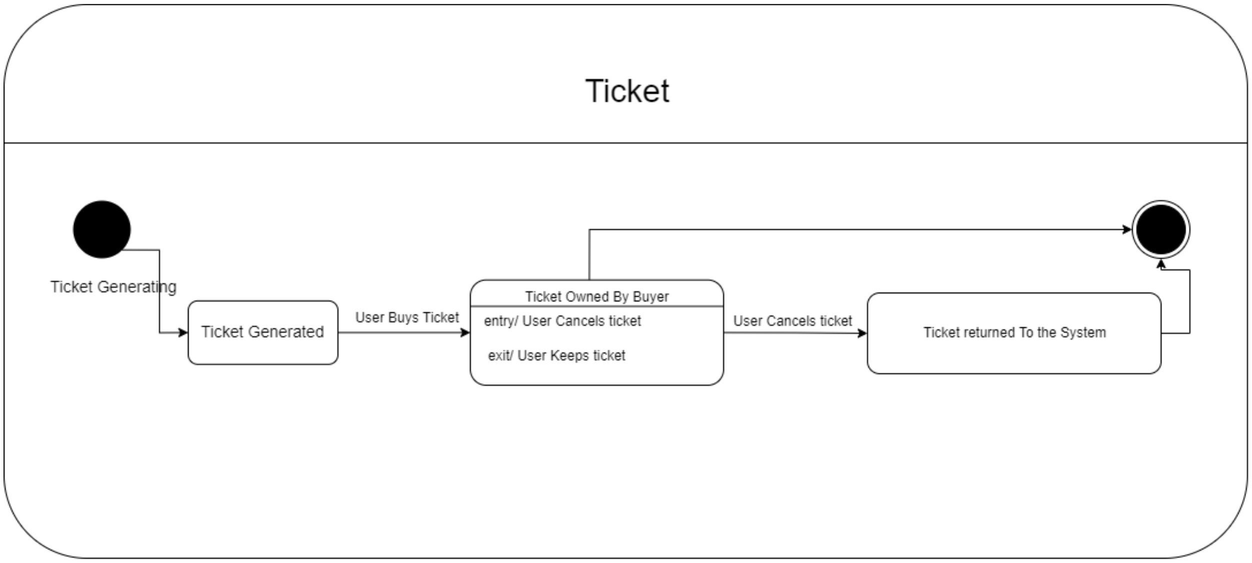
Okeoghenemarho Obuareghe

State Transition Diagrams

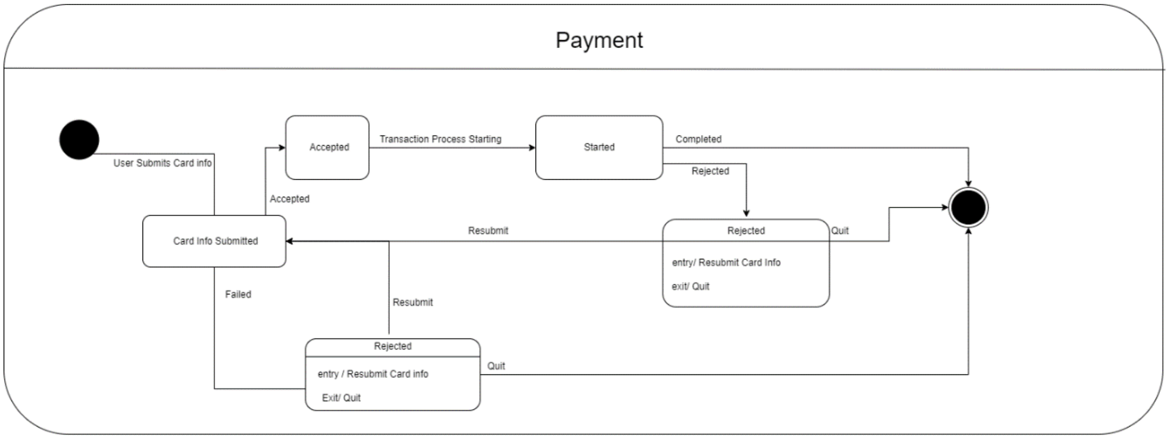


Login
User Presses Continue button
Waiting for User Input
entry/ welcome
exit/ Exit Login and continue as Ordinary User
User inputs login info
Submit
do/ Input email address and Input password
Login Info exists in Database
Accepted
User brought to main page with Registered Access Privilege
Registered User Access
Login Info does not exist in Database
Re-enter Login Info
Invalid
entry/ Message Invalid Login
User Clicks on skip as Ordinary User
Login Terminated

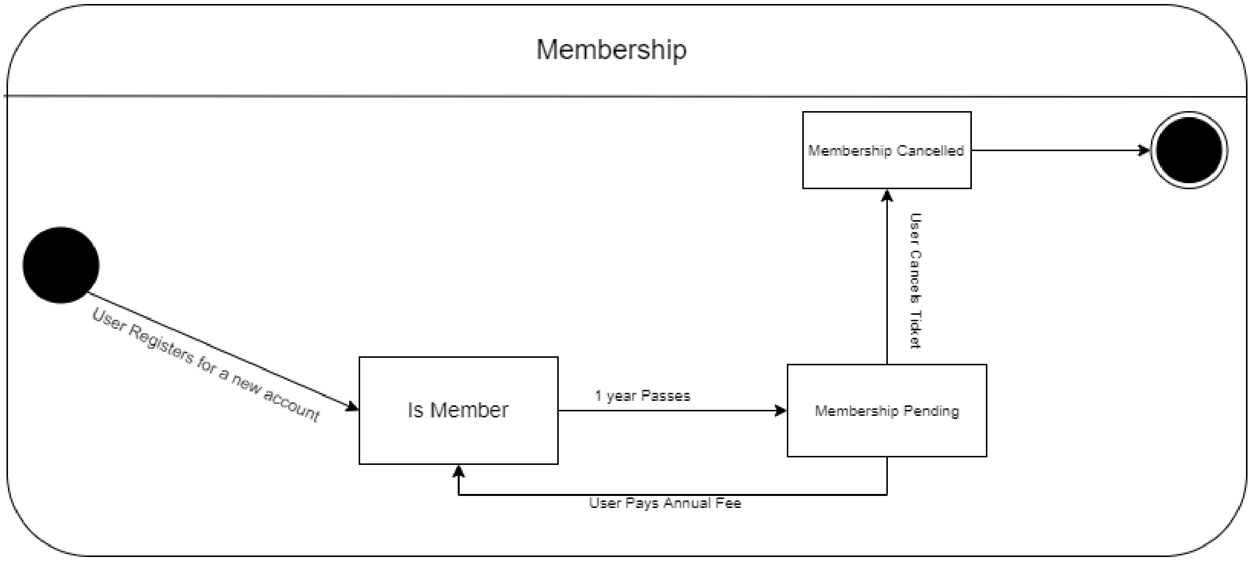
Chansuck Lee



Temiloluwa Bakare

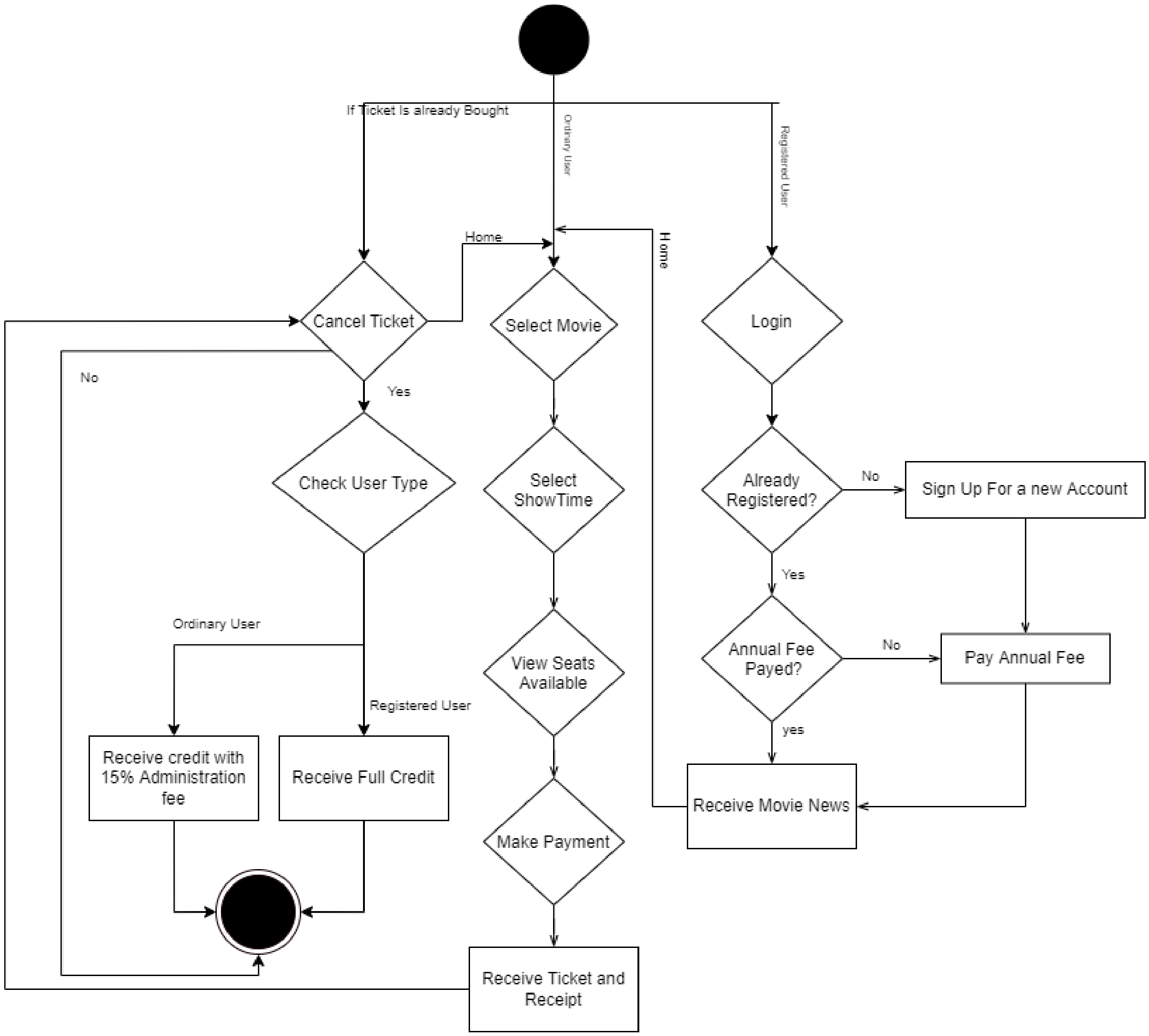


Feras Dahrooge



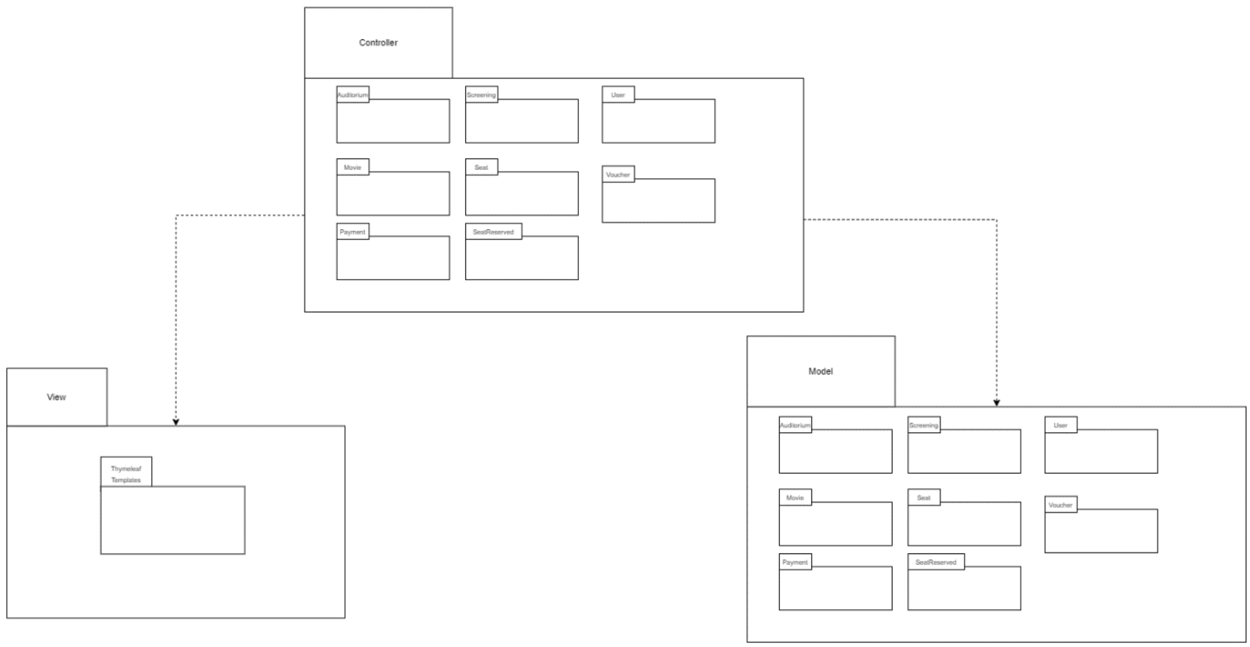
Okeoghenemarho Obuareghe

System Activity Diagrams

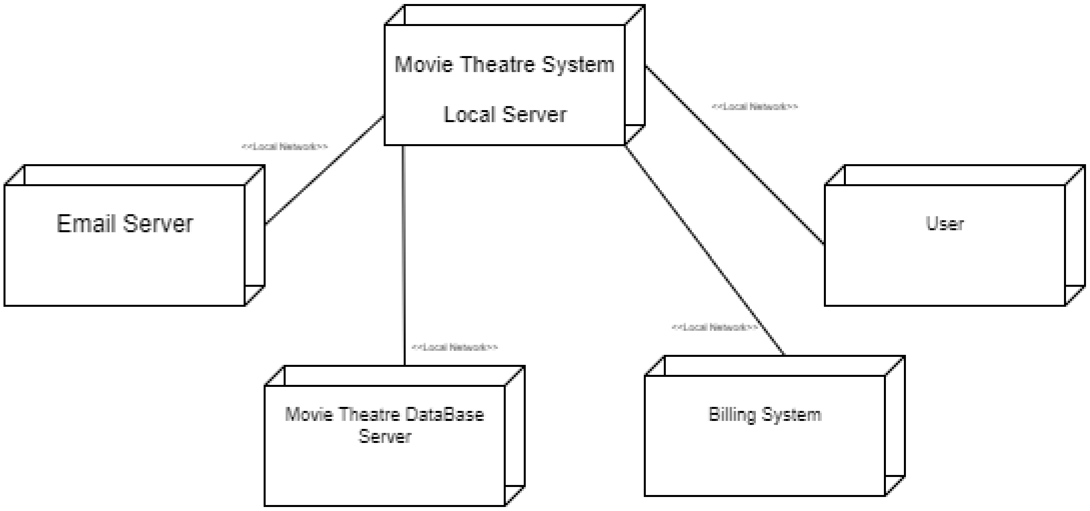


System Package Diagram

Our code is MVC that follows an Observer Design Pattern



System Deployment Diagram


Movie Theatre System
Local Server
<<Local Network>>
<<Local Network>>
Email Server
User
<<Local Network>>
<<Local Network>>
Movie Theatre DataBase Server
Billing System