Course: ENSF 614 – Fall 2022

Term Project – Design Documents

Instructor: Dr. Mahmood Moussavi

Group: 2

Alexander Kruger

Derek Walz

Graeme Folk

Leo Wei

Submission Date: December 4, 2022

Use Case Diagram

Use Case Scenarios and Candidate Objects

Scenario: Use case "select movie"

This scenario starts when user *selects* movies tab. Then he can *browse* the movie list and *select* one of the movies. A theatre list will be *displayed* and the user *selects* the theatre. A show time list will be *displayed* and user *selects* the show. Representation of theatre seats will be *displayed* and user selects the seat(s) A payment prompt will be *displayed* and the user will *complete* the payment. The user will *receive email confirmation* and ticket with the selected options.

This applies to User and Registered User and Database.

Candidate Objects:

Noun	Filtering Decision
Movies tab	Candidate Object
He	Filtered(actor)
Movie list	Candidate Object
Movies	Candidate Object
Theatre list	Candidate Object
User	Candidate Object
Theatre	Candidate Object
Show Time List	Candidate Object
Show	Candidate Object
Theatre Seats	Filtered(same as seats)
Seats	Candidate Object
Payment Prompt	Filtered(Display)
Payment	Candidate Object
Ticket	Candidate Object

Scenario: Use case "Payment with Financial Institution"

This scenario starts when a user *enters* their financial information. The credit card details are *sent* to the financial institution. Financial institution *verifies* details and payment is completed for the ticket cost.

This applies to User and Financial Institution.

Candidate Objects:

Noun	Filtering Decision
User	Candidate Object
Financial Information	Filtered(attribute of user)
Credit Card Details	Filtered(attribute of user)
Financial Institution	Candidate Object
Details	Filtered(attribute of user)
Payment	Candidate Object
Ticket Cost	Filtered(attribute of ticket)

Scenario: Use case "Payment with Financial Institution registered user"

This scenario starts when a registered user confirms ticket purchase. The credit card details are *retrieved* from the database. The credit card details are *sent* to the financial institution. Financial institution *verifies* details and payment is completed for the ticket cost.

This applies to Registered User and Financial Institution and Database.

Candidate Objects:

Noun	Filtering Decision
User	Candidate Object
Ticket Purchase	Filtered(Same as ticket)
Credit Card Details	Filtered(attribute of user)
Database	Candidate Object
Financial Institution	Candidate Object
Details	Filtered(attribute of user)
Ticket Cost	Filtered(attribute of ticket)

Scenario: Use case "Payment with store credit"

This scenario starts when a user confirms ticket purchase. The credit balance is *retrieved* from the database. If it is a registered user, the credit balance is *retrieved* from their profile. If it is a regular user, the credit balance is *retrieved* against their cancelled ticket number. The credit card details are *sent* to the financial institution if store credit balance is insufficient for the ticket cost. Financial institution *verifies* details and payment is completed for the outstanding balance, if required. The credit balance is *updated* in the database.

This applies to User and Financial Institution and Database.

Candidate Objects:

Noun	Filtering Decision
User	Candidate Object
Ticket Purchase	Filtered(Same as Ticket)
Credit Balance	Candidate Object
Database	Candidate Object
Registered User	Candidate Object
Profile	Candidate Object
Regular User	Filtered(same as user)
Ticket Number	Filtered(attribute of ticket)
Credit Card Details	Filtered(attribute of user)
Financial Institution	Candidate Object
Ticket Cost	Filtered(attribute of ticket)
Details	Filtered (attribute of user)
Payment	Candidate Object
Outstanding Balance	Filtered(attribute of Payment)

Scenario: Use case "Registration"

This scenario starts when a user *selects* the registration option. The user *enters* financial information (credit card) and personal information (name, address, email, password). Database *verifies* personal information is correct and unique. All user information is *stored* in the database after verification.

This applies to User and Database.

Candidate Objects:

Noun	Filtering Decision
User	Candidate Object
Registration Option	Filtered(Visual Prompt)
Financial Information	Filtered(attribute of user)
Personal Information	Filtered(attribute of user)
Database	Candidate Object
Information	Filtered(attribute of user)

Scenario: Use case "Login"

This scenario starts when a registered user *selects* the login option. The registered user *enters* their credentials. Credentials are *sent* to the database for *verification*. User is *given* login success output.

This applies to Registered User and Database.

Candidate Objects:

Noun	Filtering Decision
User	Candidate Object
Login Option	Filtered(GUI element)
Registered User	Candidate Object
Credentials	Filtered(attribute of user)
Database	Candidate Object
Login Success Output	Filtered(visual output)

Scenario: Use case "Cancel" as Registered User

This scenario starts when a registered user *chooses* a movie to cancel. The user *confirms cancelation*. The cancellation request is *sent* to the database. The database changes the seat availability. The database *credits* the registered user the full balance.

This applies to Registered User and Database.

Candidate Objects:

Noun	Filtering Decision
Registered User	Candidate Object
Movie	Candidate Object
User	Candidate Object
Cancellation Request	Candidate Object
Database	Candidate Object
Seat Availability	Filtered(attribute of seat)

Scenario: Use case "Cancel" as Regular User

Commented [GF1]: Could we merge the two cancel scenarios? The only difference is the admin fee, we could possibly just specify that.

This scenario starts when a user *chooses* a movie to cancel. The user *confirms cancelation*. The cancellation request is *sent* to the database. The database changes the seat availability. The database *credits* the ticket number after removal of admin fee.

This applies to User and Database.

Candidate Objects:

Noun	Filtering Decision
User	Candidate Object
Movie	Candidate Object
Database	Candidate Object
Seat Availability	Filtered(attribute of seat)
Ticket Number	Filtered(attribute of ticket)
Admin Fee	Filtered(attribute of Payment)

Scenario: Use case "Pay Annual Fee"

This scenario starts when a registered user confirms annual participation. The credit card details are *retrieved* from the database. The credit card details are *sent* to the financial institution. Financial institution *verifies* details and payment is completed. Annual payment confirmation is *sent* to the database. Database *updates* the registered users status.

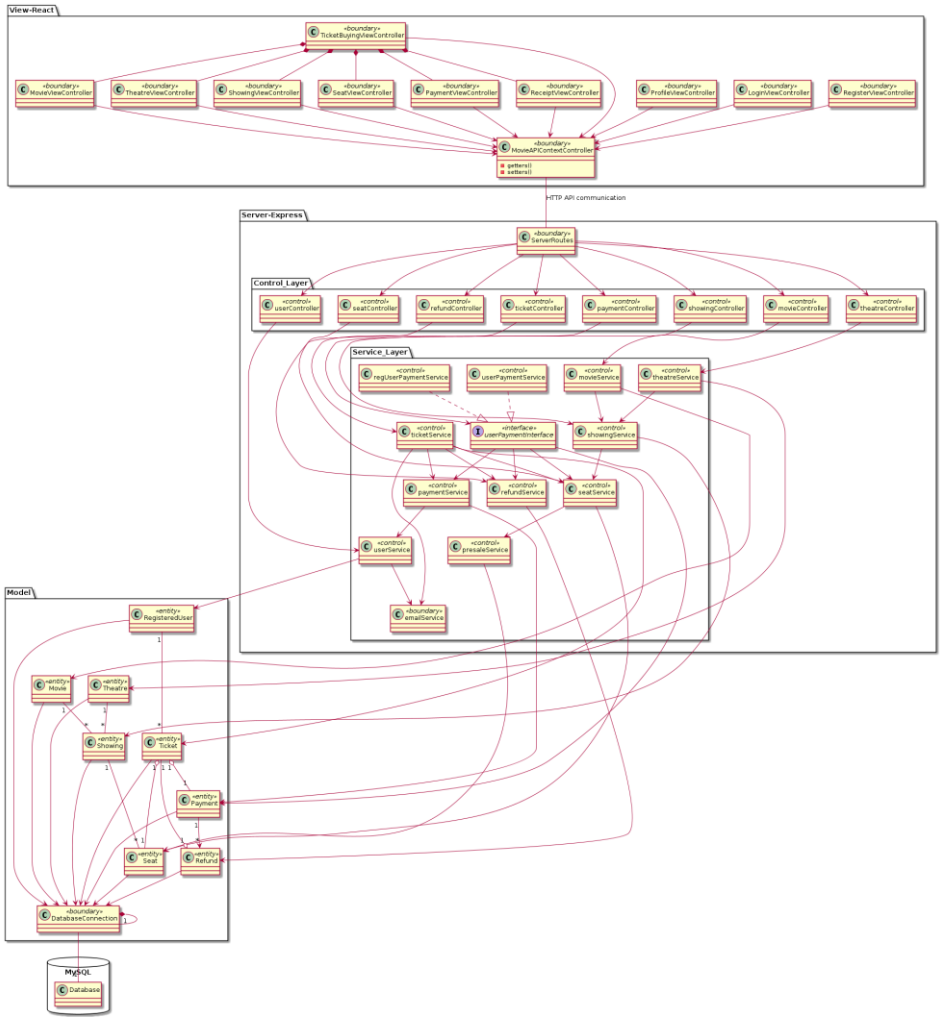
Candidate Objects:

Noun	Filtering Decision
Registered User	Candidate Object
Credit Card Details	Filtered(attribute of user)
Database	Candidate Object
Financial Institution	Candidate Object
Payment	Candidate Object
Annual Payment Confirmation	Filtered(Visual Element)
Status	Filtered(Attribute of user)

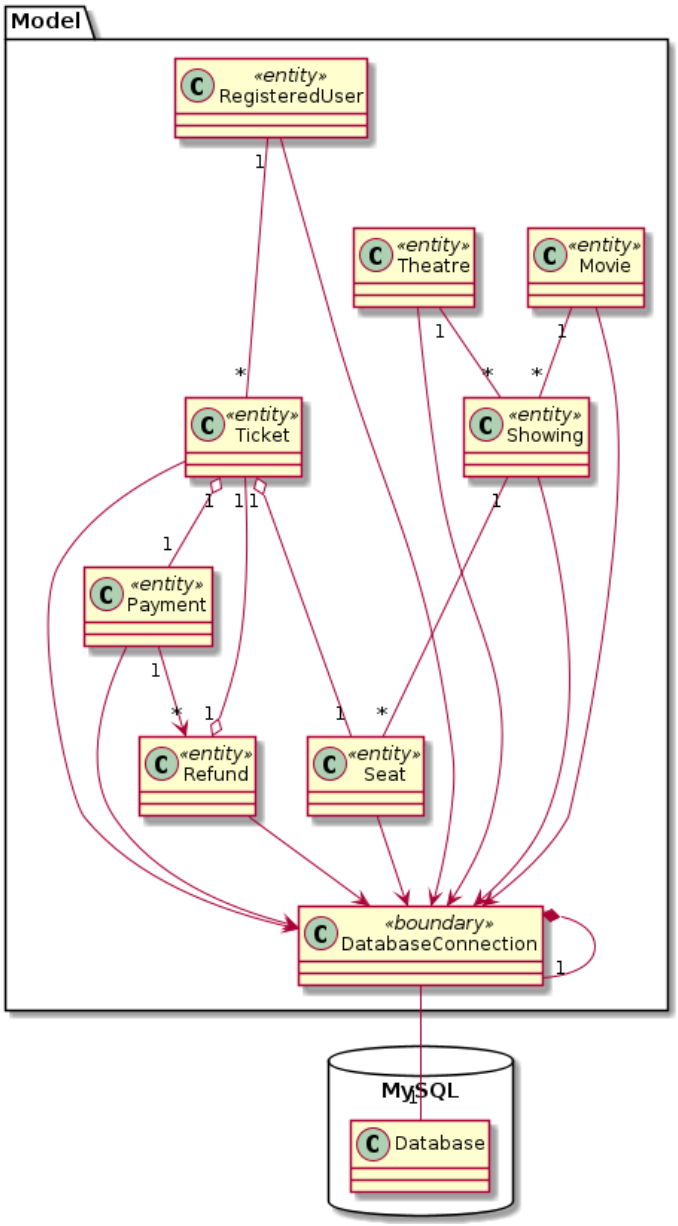
All Unique Candidate Objects:

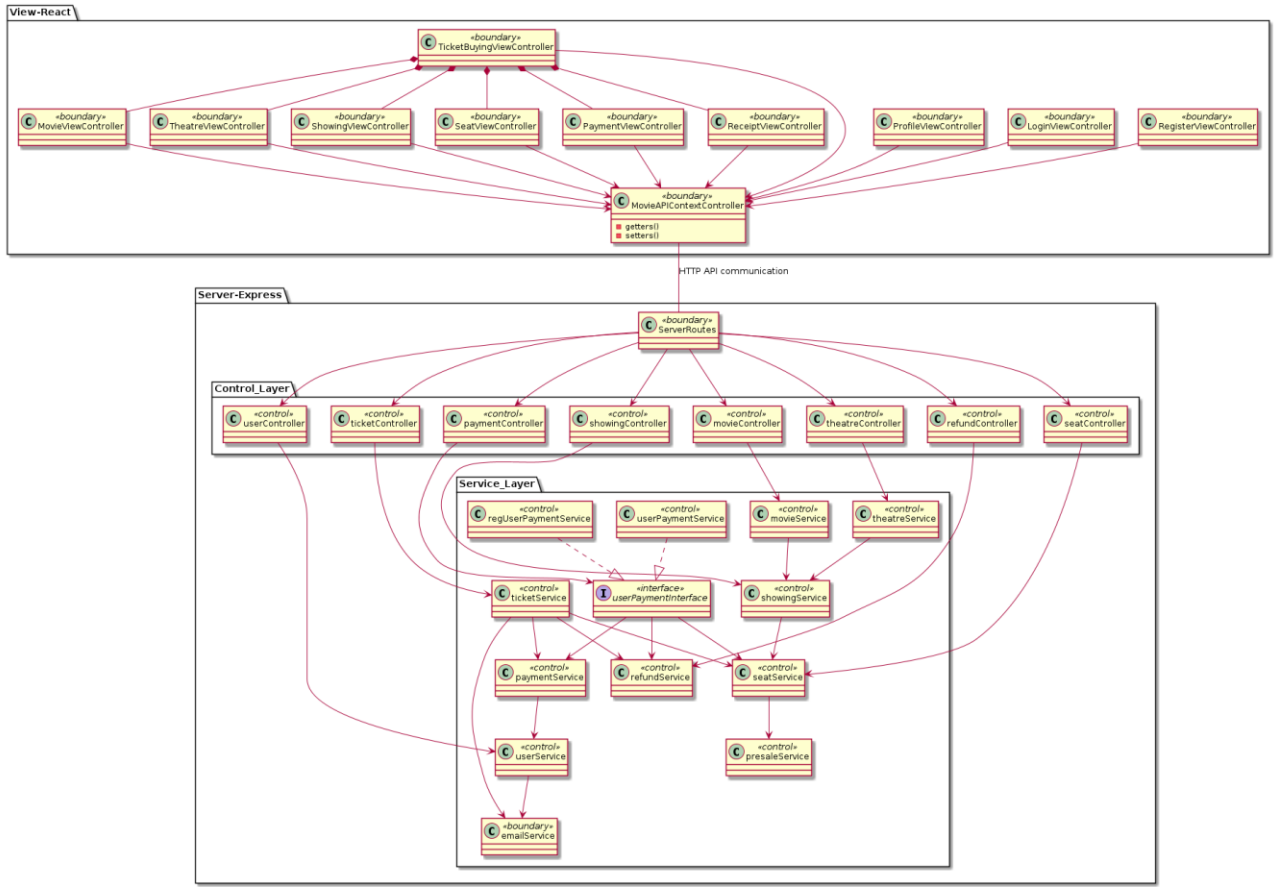
Movies Tab
Movies
Theatre List
User
Theatre
Show Time List
Show
Seats
Payment
Ticket
Financial Institution
Database
Credit Balance
Profile
Cancellation Request

Design Class Diagram (Relationships Only)



Zoomed in:

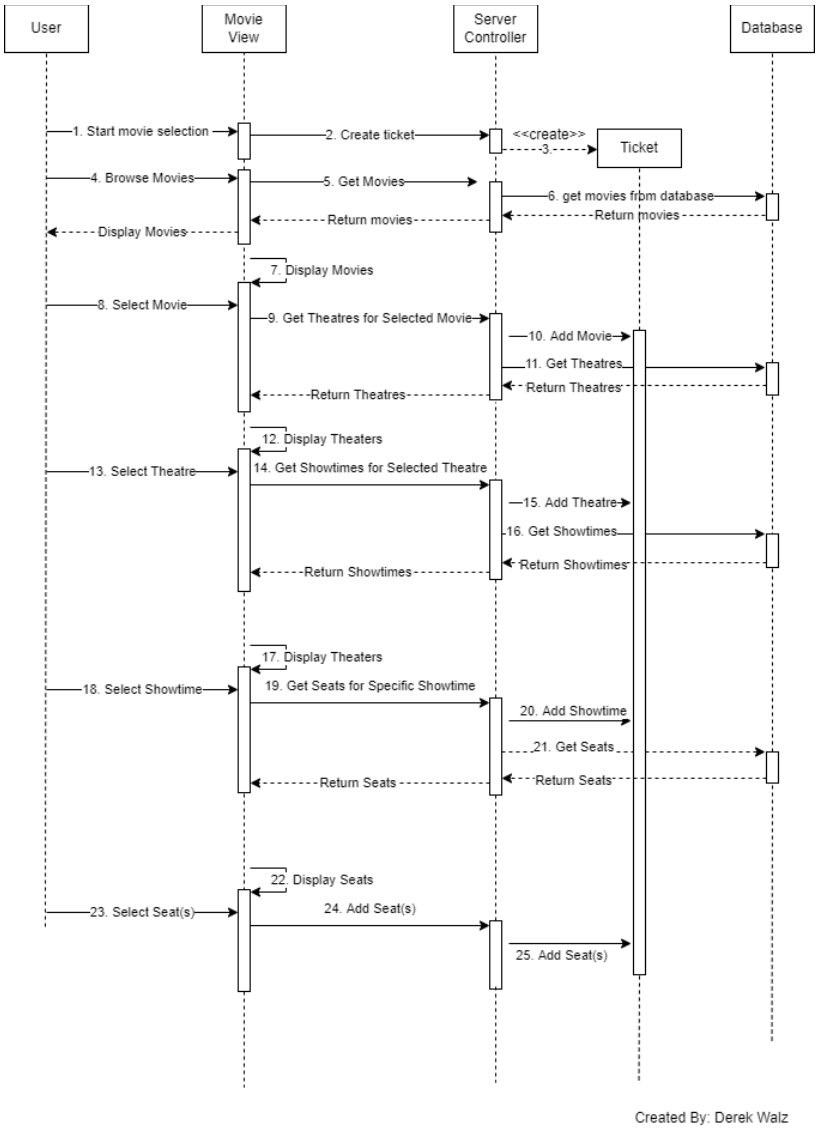

Model
«entity»
RegisteredUser
«entity»
Theatre
«entity»
Movie
«entity»
Ticket
«entity»
Showing
«entity»
Payment
«entity»
Refund
«entity»
Seat
«boundary»
DatabaseConnection
MySQL
Database


View-React
«boundary»
TicketBuyingViewController
«boundary»
MovieViewController
«boundary»
TheatreViewController
«boundary»
ShowingViewController
«boundary»
SeatViewController
«boundary»
PaymentViewController
«boundary»
ReceiptViewController
«boundary»
ProfileViewController
«boundary»
LoginViewController
«boundary»
RegisterViewController
«boundary»
MovieAPIContextController
getters()
setters()
HTTP API communication
Server-Express
«boundary»
ServerRoutes
Control_Layer
«control»
userController
«control»
ticketController
«control»
paymentController
«control»
showingController
«control»
movieController
«control»
theatreController
«control»
refundController
«control»
seatController
Service_Layer
«control»
regUserPaymentService
«control»
userPaymentService
«control»
movieService
«control»
theatreService
«control»
ticketService
«interface»
userPaymentInterface
«control»
showingService
«control»
paymentService
«control»
refundService
«control»
seatService
«control»
userService
«control»
presaleService
«boundary»
emailService

Design Class Diagram (Attributes and Operations)

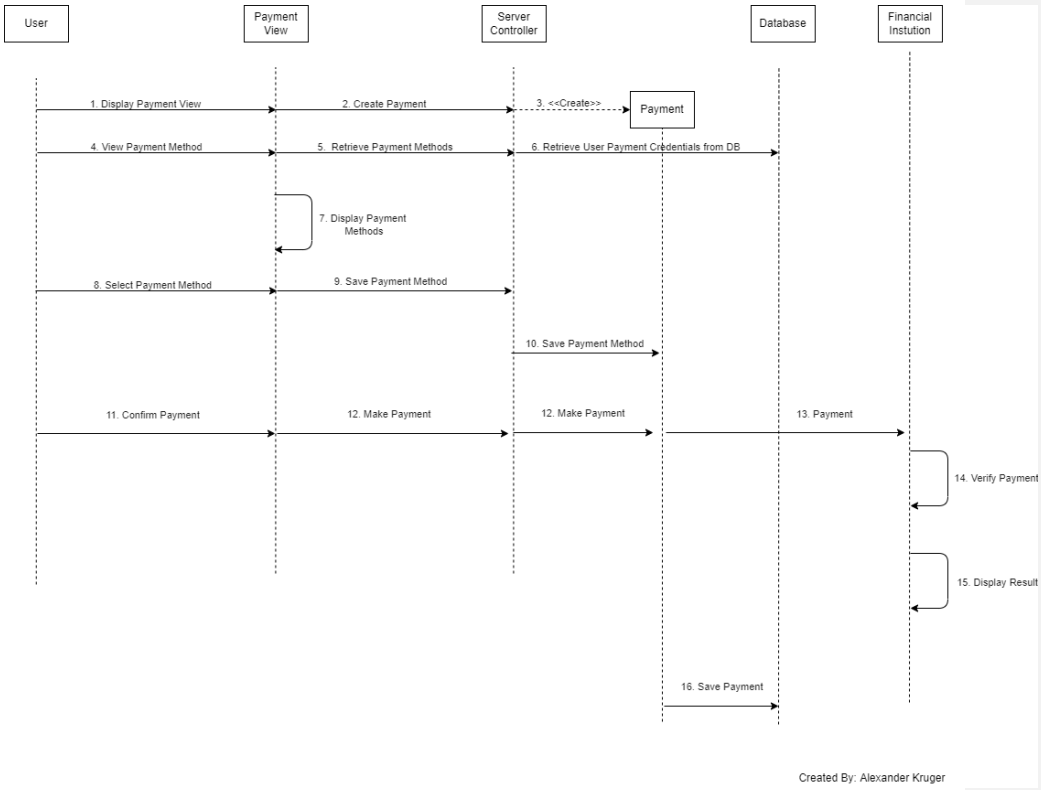
Interaction Diagrams

Movie



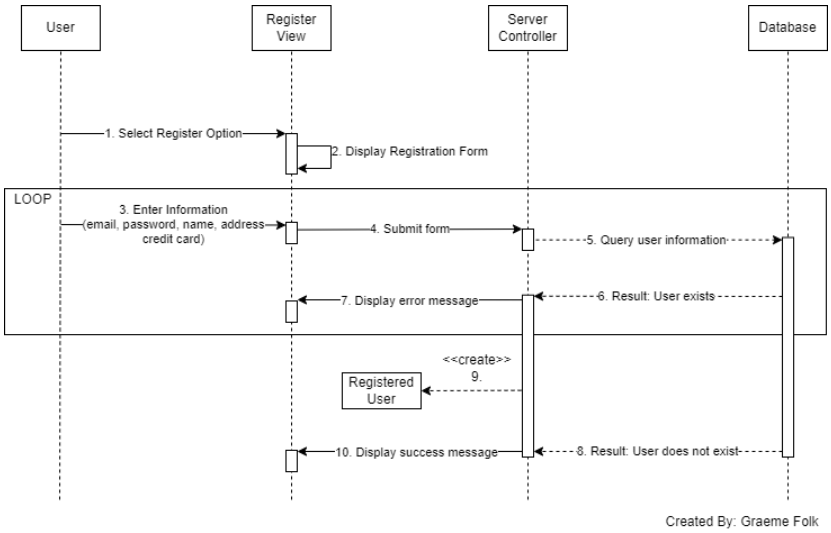
Created By: Derek Walz

Payment



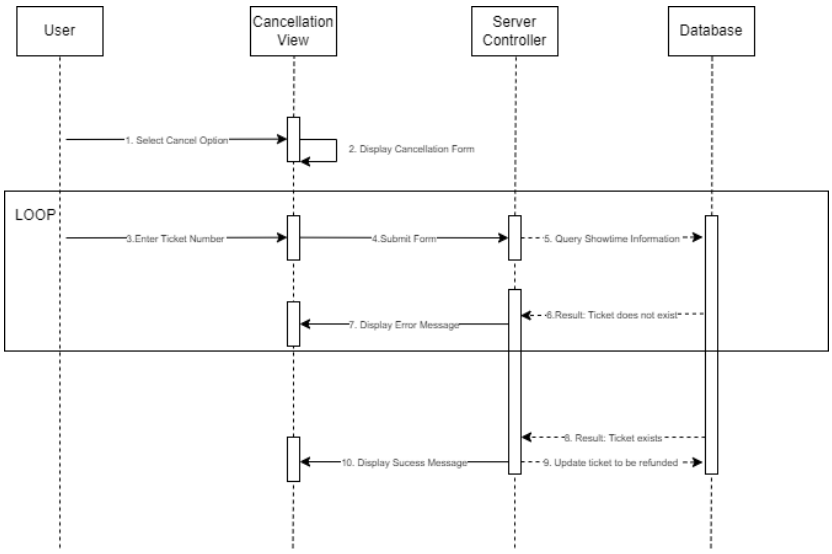
Created By: Alexander Kruger

Register



Created By: Graeme Folk

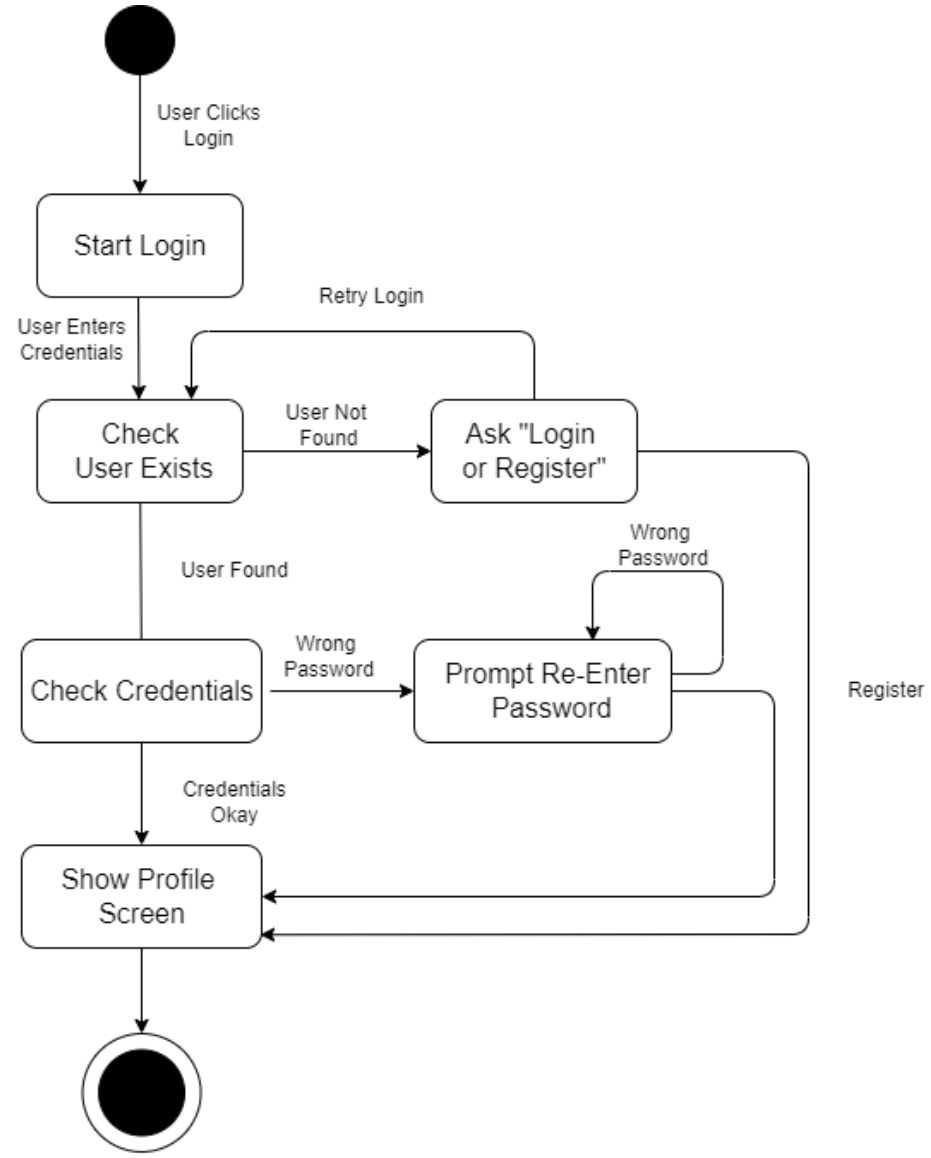
Cancel



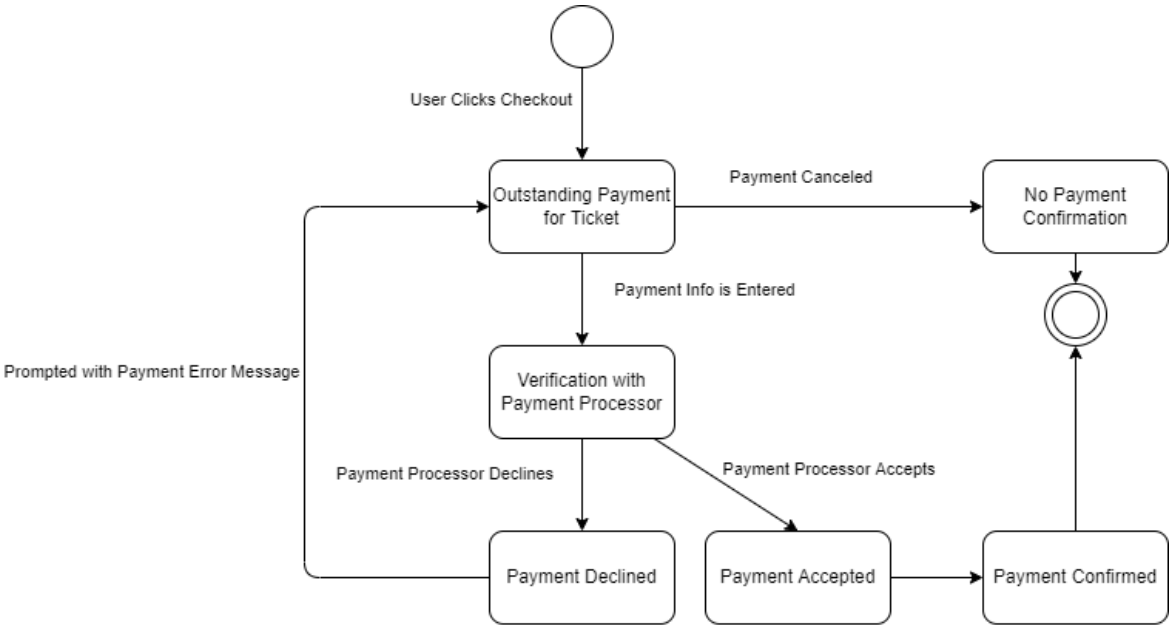
Created By: Leo Wei

State Transition Diagrams

Login



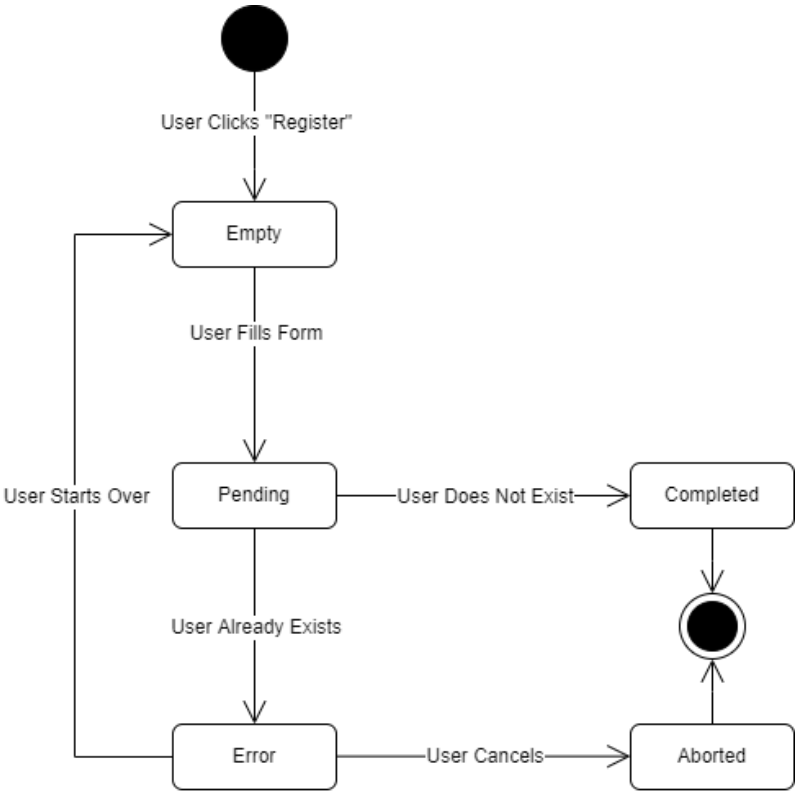
Created By: Alexander Kruger



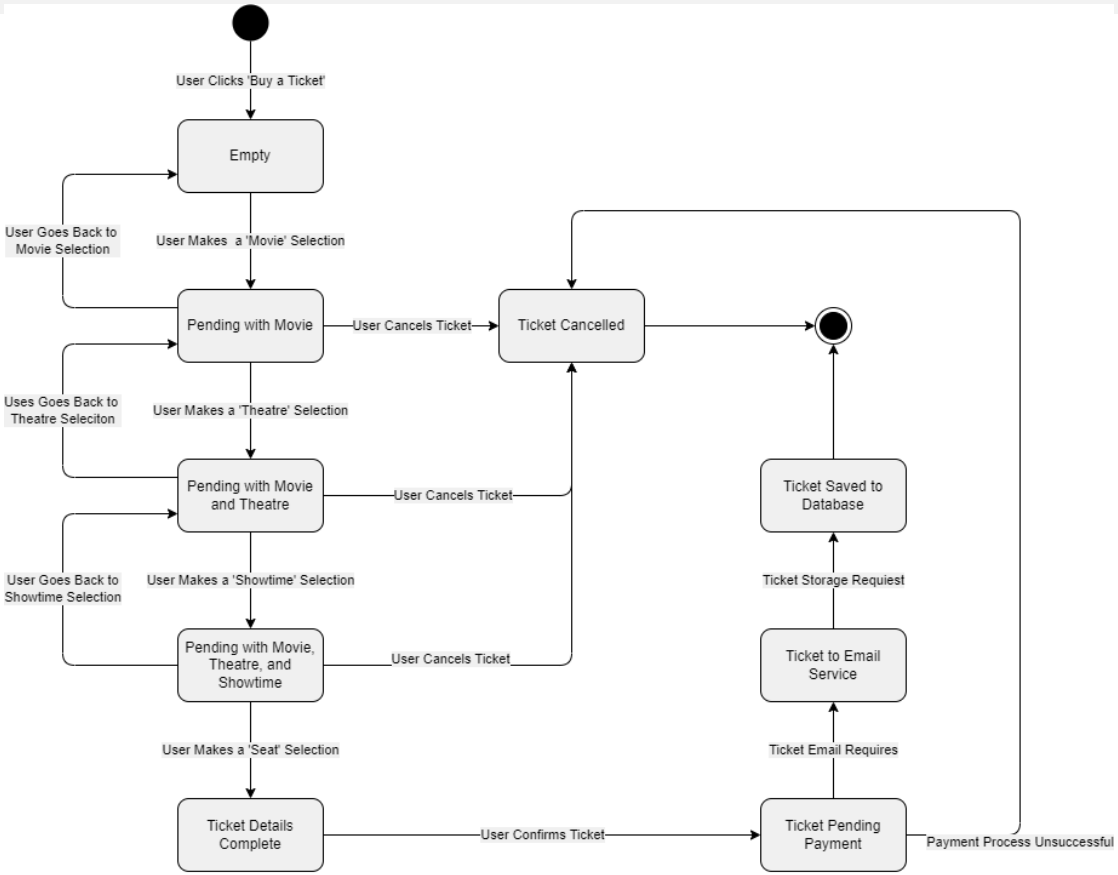
Payment

Created By: Leo Wei

Registration



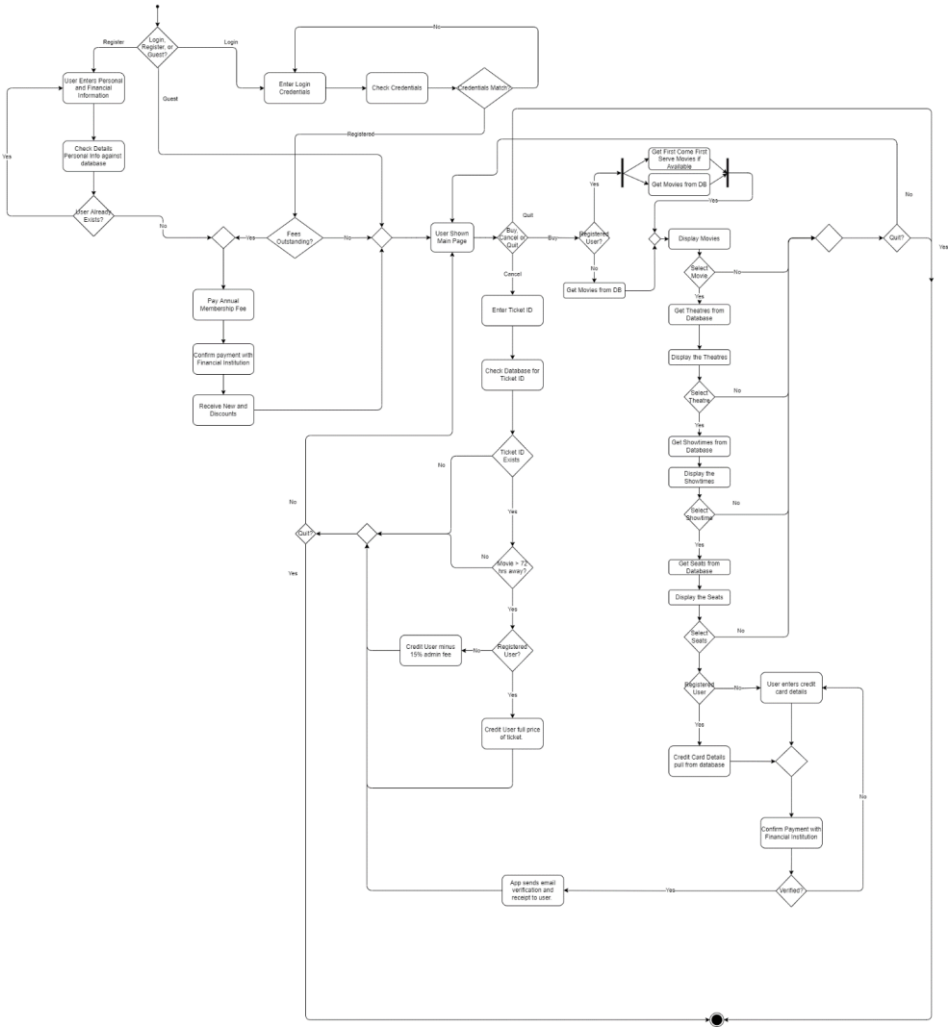
Created By: Graeme Folk



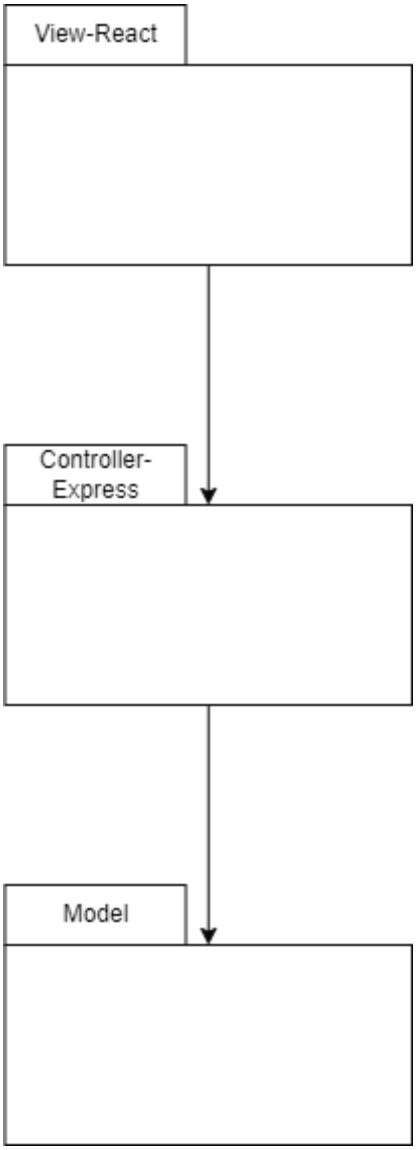
Ticket

Created By: Derek Walz

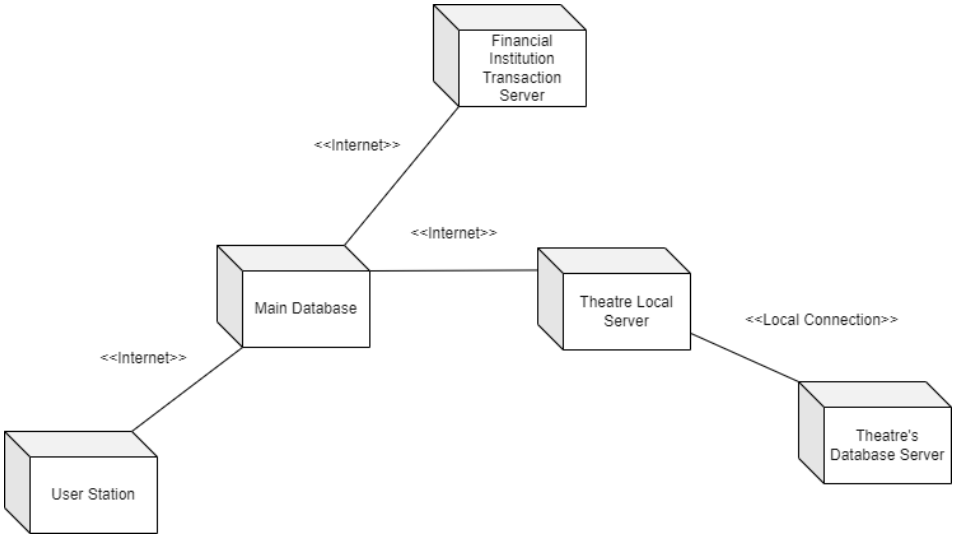
System Activity Diagram



System Package Diagram

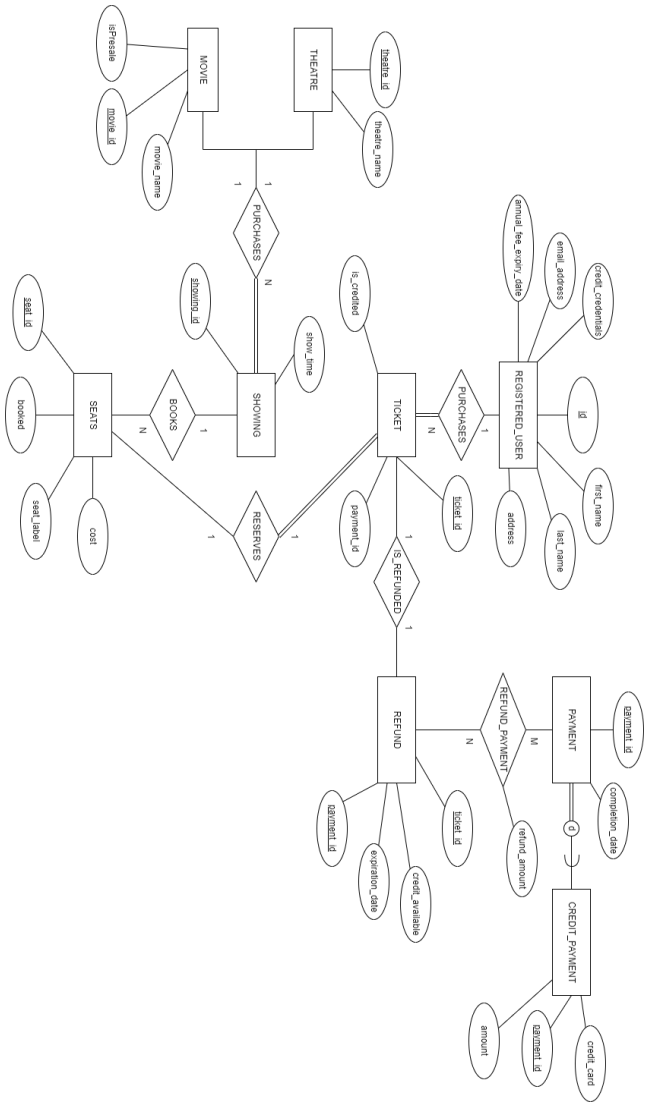


System Deployment Diagram

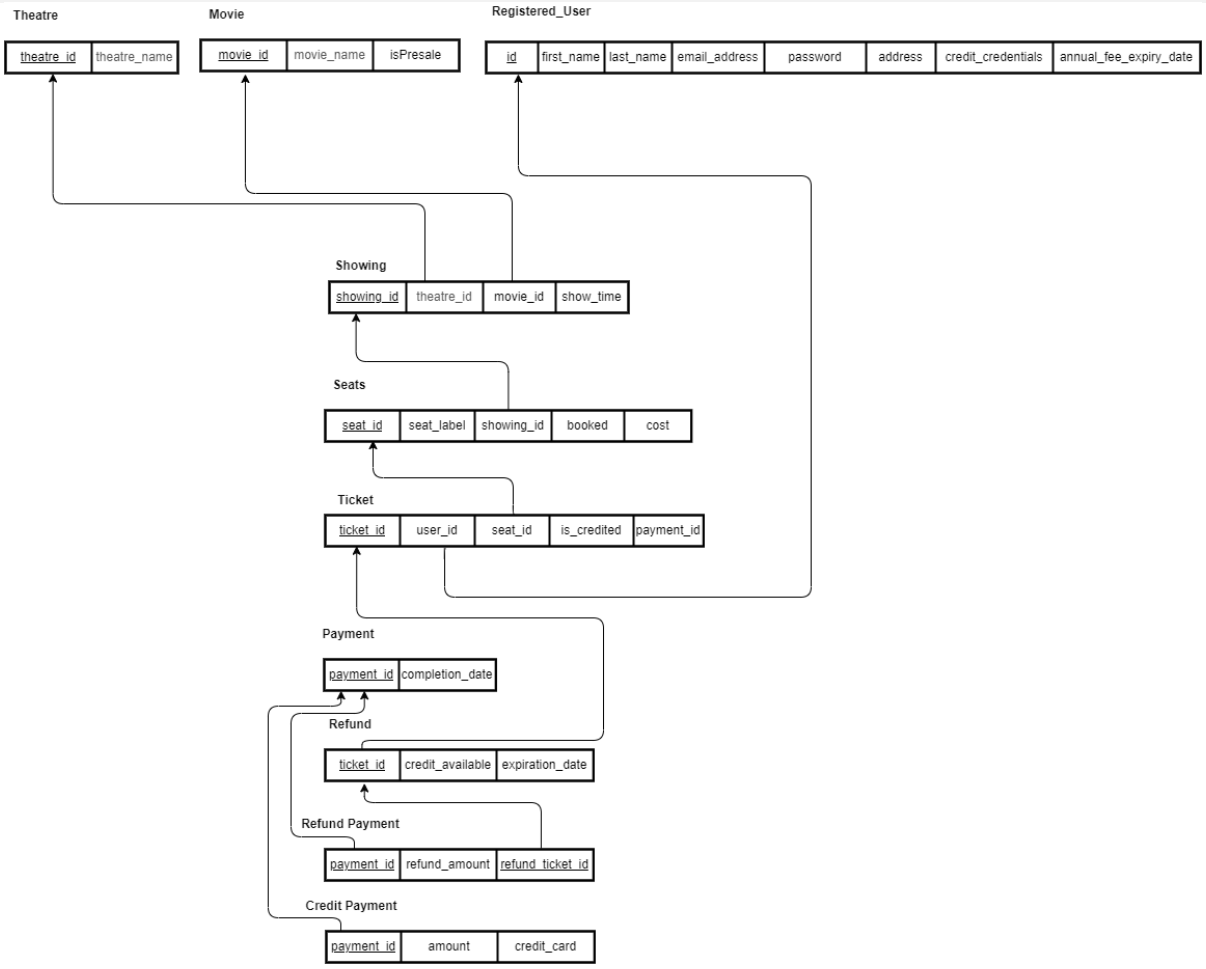


Database Diagrams (Extra)

EER Diagram



Relational Model



Seat Layout Reference