ENSF 619 - Group 8 - Ticket Reservation App

Design Phase Report

Submitted on: November 22, 2020

Submitted by: Victor Tuah Kumi, Patrick Kwan, Oluwapelumi Laditan, Michael Lasby

README:

Please note that some of these diagrams are best viewed as image / pdf files. These diagrams are included in the enclosed `./lib/` folder and file path names are included below.

Use Case Diagram

See `./lib/UseCaseDiagram/UseCaseDiagram.png`

Scenarios and Candidate Objects (see next page)

Use Case Scenarios

Please note the following formatting styles:

- Candidate Object: Object
- Candidate Operation: operation

Login:

John McClane selects login from the Main Menu. The GUI Controller displays the Login View. John enters his Username and Password in the Login View. The Login View Controller registers John's input and passes the data to the Database Controller. The Database Controller queries the Database for a Registered User matching the provided Username and Password. If the query is unsuccessful, the Login View will prompt John to re-enter their Username and Password until a successful query is returned or John returns to the Main Menu. Since John has correctly entered his Username and Password, he is returned to the Main Menu as a Registered User.

Noun	Filter	Class Type
John McClane	Filtered (User Actor)	N/A
Login	Filtered (Attribute of Main Menu)	N/A
Main Menu	Candidate Object	Boundary
GUI Controller	Candidate Object	Control
Login View	Candidate Object	Boundary
Login View Controller	Candidate Object	Control
Username	Filtered (Attribute of Registered User)	N/A
Password	Filtered – (Attribute of Registered User)	N/A
Database Controller	Candidate Object	Control
Database	Filtered (Database Actor)	N/A
Registered User	Candidate Object	Entity

Select Ticket:

John selects select ticket from the Main Menu. The GUI Controller displays the Movie View The Movie View displays all Movie options, including both Publicly-Announced Movies and Early-Access Movies. John browses Movie options before selecting the Die Hard Movie.

The Theatre View displays the available Theatres that are showing Die Hard and waits for John to make a selection. John selects the Country Hills Theatre. Upon selection of the Theatre, the Showtime View displays Die Hard's Showtimes and prompts John to make a selection. John selects Dec. 27th at 5 PM. Since this Showtime is for an Early-Access Movie, the Showtime View Controller checks whether or not 10% of the Seats have already been reserved by Registered Users. Since only 3% of the Seats have been reserved thus far, the Showtime View Controller accepts John's Showtime selection. Once a Showtime is selected, the Seats View displays the Seats with a colour hue to indicate each Seat's Availability Status.

John selects Seat D7 from the remaining available Seats. With all the necessary Ticket information selected, a Ticket is added to John's Cart. John is then prompted to the either return to the Main Menu or proceeds to the Payment View with his selected Ticket information (Movie, showtime, etc.) to complete his transaction. John elects to proceed to the Payment View, so the GUI Controller displays that view.

Noun	Filter	Class Type
John McClane	Filtered (User Actor)	N/A
Main Menu	Candidate Object	Boundary
GUI Controller	Candidate Object	Control
Movie View	Candidate Object	Boundary
Movie	Candidate Object	Entity
Publically-Announced Movie	Filtered (Attribute of Movie)	N/A
Early-Access Movie	Filtered (Attribute of Movie)	N/A
Theatre View	Candidate Object	Boundary
Theatre	Candidate Object	Entity
Showtime View	Candidate Object	Boundary
Showtime	Candidate Object	Entity
Showtime View Controller	Candidate Object	Control
Registered User	Candidate Object	Entity
Seat View	Candidate Object	Boundary
Seat	Candidate Object	Entity
Availability Status	Filtered (Attribute of Seat)	N/A
Payment View	Candidate Object	Boundary
Ticket	Candidate Object	Entity
Cart	Candidate Object	Control

Checkout:

Since John is logged in as a Registered User, the Payment View Controller queries the Database Controller and retrieves John's Payment Information. The Payment View Controller pre-populates the Payment View form with John's Payment Information, including Card Type and Card Number. John enters his Pin Number and reviews his Payment Information before pressing submit. The Payment View Controller passes the Payment Information to the Financial Controller so the Financial Institution may verify the Payment Information.

With the Payment Information verified, the Database Controller records the Ticket and Receipt data to the Database before emailing John a copy of the Ticket and Ticket Receipt. John is flashed a success message before being returned to the Main Menu by the GUI Controller.

Noun	Filter	Class Type
John McClane	Filtered (User Actor)	N/A
Registered User	Candidate Object	Entity
Payment View Controller	Candidate Object	Control
Database Controller	Candidate Object	Boundary

Payment Information	Candidate Object	Entity
Payment View	Candidate Object	Boundary
Card Type	Filtered (Attribute of Payment Information)	N/A
Card Number	Filtered (Attribute of Payment Information)	N/A
Pin Number	Filtered (Attribute of Payment Information)	N/A
Financial Controller	Candidate Object	Boundary
Financial Institution	Filtered (Financial Institution Actor)	N/A
Ticket	Candidate Object	Entity
Ticket Receipt	Candidate Object	Entity
Database	Filtered (Database Actor)	N/A
Main Menu	Candidate Object	Boundary
GUI Controller	Candidate Object	Control

Refund Ticket:

John selects refund ticket from the Main Menu. The GUI Controller displays the Refund View. John enters his Receipt Number into the form and submits the data. The Refund View Controller passes the data to the Database Controller which queries the Database for a Ticket Receipt matching the Receipt Number entered by John. Since John has correctly entered an existing Receipt Number and the Ticket's Showtime is more than 72 hours away, the previously reserved Seat is changed to an Available Status and John's Credit Card is refunded the full price of the Ticket since he is logged in as a Registered User. An email of the Refund Receipt is sent to John's email account. The Refund View Controller flashes a success message before returning John to the Main Menu.

If John had not been logged in as a Registered User, a Coupon worth 85% of the original Ticket Price would have been included in the Refund Receipt emailed to John.

Noun	Filter	Class Type
John McClane	Filtered (User Actor)	N/A
Main Menu	Candidate Object	Boundary
GUI Controller	Candidate Object	Control
Refund View	Candidate Object	Boundary
Receipt Number	Candidate Object	Entity
Database	Filtered (Database Actor)	N/A
Ticket Receipt	Candidate Object	Entity
Showtime	Candidate Object	Entity
Seat	Candidate Object	Entity
Available Status	Filtered (Attribute of Seat)	N/A
Credit Card	Filtered (Attribute of Payment Information)	N/A
Ticket	Candidate Object	Entity
Registered User	Candidate Object	Entity

Refund Receipt	Candidate Object	Entity
Financial Controller	Candidate Object	Boundary
Refund View Controller	Candidate Object	Control
Coupon	Candidate Object	Entity

Register Account

Sally wants to register as a Registered User. She selects register user from the Main Menu. The GUI Controller displays the Register View. Sally enters her information and presses the submit button. The Register View Controller verifies that all required information is provided before passing the information to the Database. The Database Controller confirms that the requested username is available before committing the new Register User information to the Database.

Sally is flashed a success message that reminds her to pay her Annual Dues before reserving any Early-Access Movies. Sally accepts the message and is returned to the Main Menu where pay annual dues is now available as an option to Sally.

Noun	Filter	Class Type
Sally	Filtered (User Actor)	N/A
Registered User	Candidate Object	Entity
Main Menu	Candidate Object	Boundary
GUI Controller	Candidate Object	Control
Register View	Candidate Object	Boundary
Register View Controller	Candidate Object	Control
Database Controller	Candidate Object	Boundary
Database	Filtered (Database Actor)	N/A
Annual Dues	Filtered (Attribute of Registered User)	N/A

Pay Annual

Sally selects pay annual from the Main Menu. Since Sally is logged in as a Registered User, the Payment View Controller queries the Database Controller and retrieves Sally's Payment Information The Payment View Controller pre-populates the Payment View form with Sally's Payment Information, including Card Type and Card Number. Sally enters his Pin Number and reviews his Payment Information before pressing submit. The Payment View Controller passes the Payment Information to the Financial Controller so the Financial Institution may verify the Payment Information.

With the Payment Information verified, the Database Controller records the Annual Due payment and Receipt data to the Database before emailing Sally a copy of the Annual Dues Receipt. Sally is flashed a success message before being returned to the Main Menu by the GUI Controller.

Noun	Filter	Class Type
Sally	Filtered (User Actor)	N/A
Registered User	Candidate Object	Entity

Payment View Controller	Candidate Object	Control
Database Controller	Candidate Object	Boundary
Payment Information	Candidate Object	Entity
Payment View	Candidate Object	Boundary
Card Type	Filtered (Attribute of Payment Information)	N/A
Card Number	Filtered (Attribute of Payment Information)	N/A
Pin Number	Filtered (Attribute of Payment Information)	N/A
Financial Controller	Candidate Object	Boundary
Financial Institution	Filtered (Financial Institution Actor)	N/A
Annual Dues Receipt	Candidate Object	Entity

Issue Movie News

Bill, a manager at Country Hills Theatre, is logged in as a Manager and selects issue movie news from the Main Menu. The GUI Controller displays the Issue News View. Bill selects the upload button and adds the new Movie News file he intends to publish. The Issue News View Controller paints the Issue News View with the uploaded Movie News text so Bill may review before issuing. Bill selects the submit button and the Issue News View Controller passes the new Movie News to the Boss Controller. The Boss Controller queries the Database Controller for the Emails of all Registered Users before emailing the new Movie News to all Registered Users.

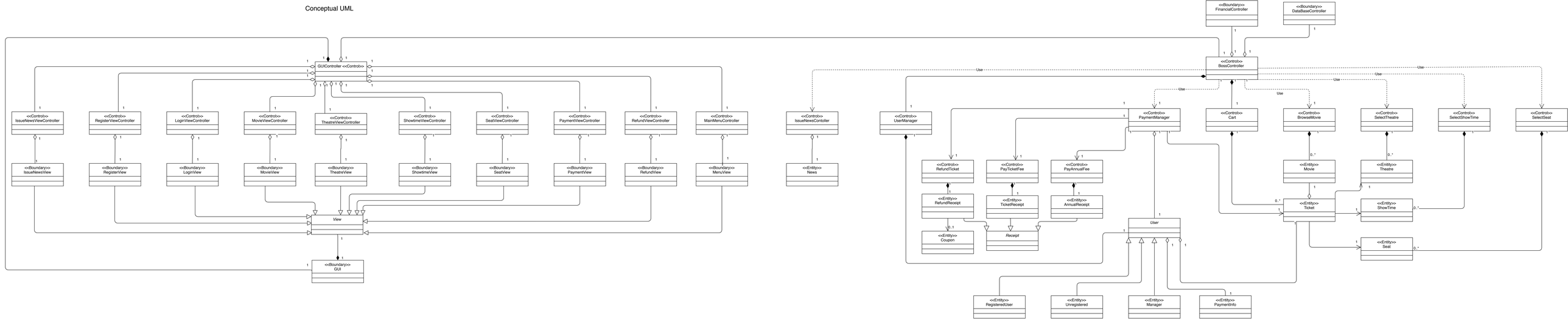
Bill is flashed a success message before being returned to the Main Menu.

Noun	Filter	Class Type
Bill	Filtered (User Actor)	N/A
Manager	Candidate Object	Entity
Theatre	Candidate Object	Entity
Main Menu	Candidate Object	Boundary
GUI Controller	Candidate Object	Controller
Issue News View	Candidate Object	Boundary
Movie News	Candidate Object	Entity
Issue News View Controller	Candidate Object	Controller
Boss Controller	Candidate Object	Controller
Database Controller	Candidate Object	Boundary
Email	Filtered (Attribute of Registered User)	N/A
Registered User	Candidate Object	Entity

Conceptual Class Design Diagram (see next page)

See `./lib/ConceptualUML/ConceptualUML.png` or `/lib/ConceptualUML/ConceptualUML.pdf`

Conceptual UML



Detailed Class Design Diagram (see next 3 pages)

See `./lib/DetailedUML/DetailedUML.pdf`. `.png` files also available in the same folder.

Controller

BossController

- cart: Cart
- userMgmt: UserManager
- financialController: FinancialController
- dbController: DatabaseController

- browseMovie(): JsonObject
- selectMovie(name: Movie): JsonObject
- selectTheatre(theatreId: int): JsonObject
- selectShowTime(date: String): JsonObject
- selectSeat(seatRow: int, seatCol: int): JsonObject
- registerUser(name: String, password: String, cardType: String, cardNum: String, email: String)
- refundTicket(receiptNum: int): JsonObject
- checkout(): JsonObject
- verifyPayment(cardType: char, cardNum: int, cardPin: int): JsonObject
- payAnnual(userId: int): JsonObject
- issueMovieNews(news: File): JsonObject

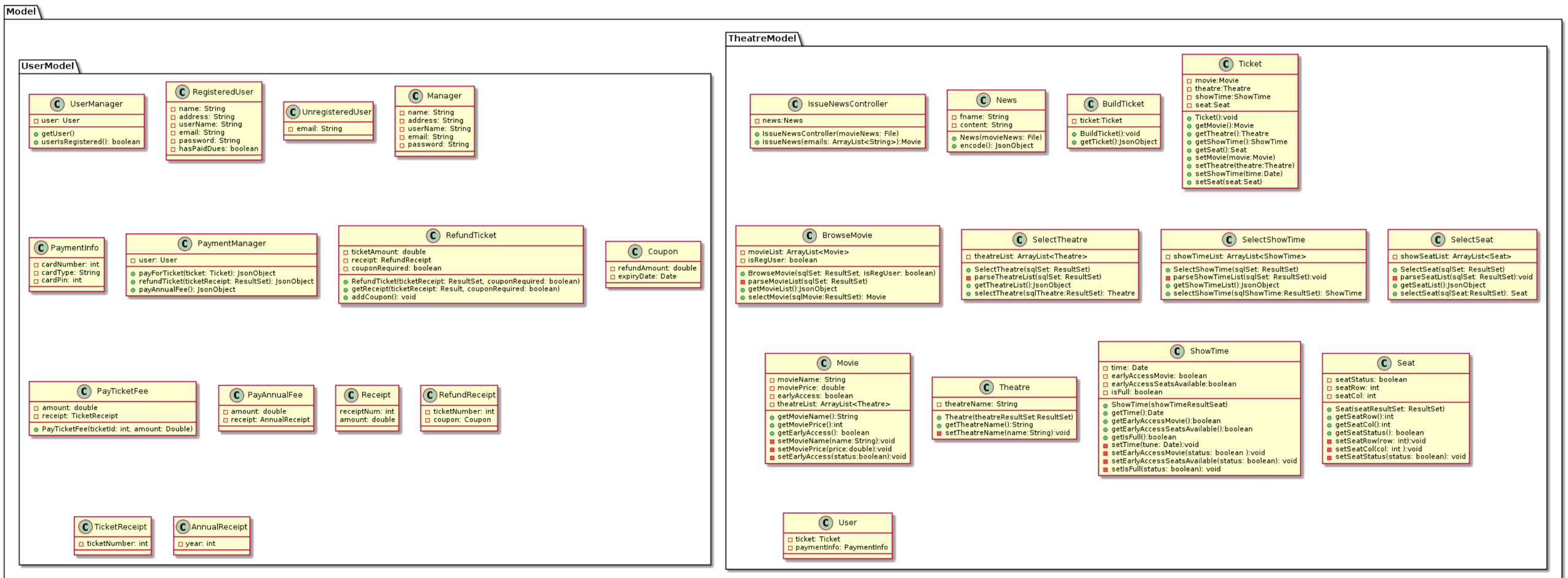
FinancialController

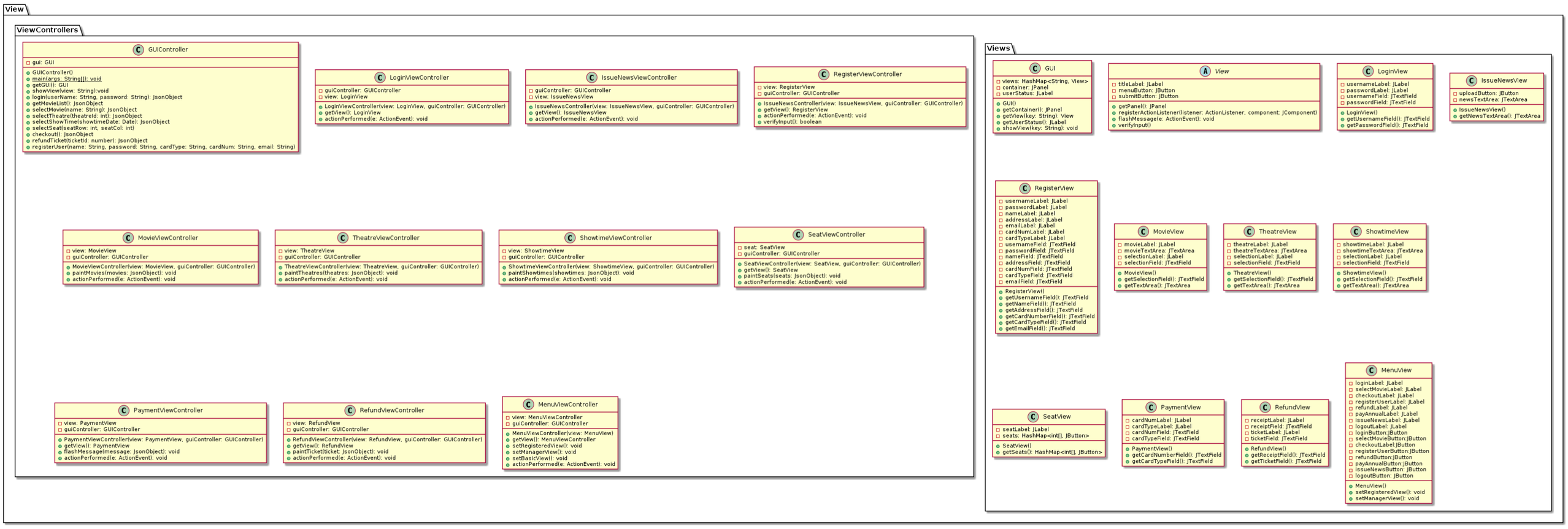
- FinancialController()
- processPayment(paymentInfo: JsonObject): boolean
- processRefund(paymentInfo: JsonObject): boolean

DataBaseController

- connection: Connection
- connectionInfo: String
- username: String
- password: String

- getUser(userId: int): ResultSet
- getRegisteredUserList(): ResultSet
- updateRegisteredUserPayment(userId: int, cardNumber: int, cardType: char): boolean
- registerUser(name: String, password: String, cardType: String, cardNum: String, email: String): void
- getRegisteredUserEmails():ResultSet
- getMovieList(): ResultSet
- getMovie(movieName: String): ResultSet
- getTheatreList(): ResultSet
- getTheatre(movieName: String, theatreId: int): ResultSet
- getShowTimeList(): ResultSet
- getShowtime(movieName: String, theatreId: int, showtimeDate: date): ResultSet
- updateShowtimeAvailability(ticket: JsonObject): void
- getSeatList(): ResultSets
- getSeat(movieName: String, theatreId: int, showtimeDate: Date, seatNum: int): ResultSet
- insertTicket(ticket: JsonObject): void
- getReceipt(receiptId: int): ResultSet
- returnTicket(receiptId: int): void
- insertReceipt(receipt: JsonObject): void


Model
UserModel
UserManager
user: User
getUser()
userIsRegistered(): boolean
RegisteredUser
name: String
address: String
userName: String
email: String
password: String
hasPaidDues: boolean
UnregisteredUser
email: String
Manager
name: String
address: String
userName: String
email: String
password: String
PaymentInfo
cardNumber: int
cardType: String
cardPin: int
PaymentManager
user: User
payForTicket(ticket: Ticket): JsonObject
refundTicket(ticketReceipt: ResultSet): JsonObject
payAnnualFee(): JsonObject
RefundTicket
ticketAmount: double
receipt: RefundReceipt
couponRequired: boolean
RefundTicket(ticketReceipt: ResultSet, couponRequired: boolean)
getReceipt(ticketReceipt: Result, couponRequired: boolean)
addCoupon(): void
Coupon
refundAmount: double
expiryDate: Date
PayTicketFee
amount: double
receipt: TicketReceipt
PayTicketFee(ticketId: int, amount: Double)
PayAnnualFee
amount: double
receipt: AnnualReceipt
Receipt
receiptNum: int
amount: double
RefundReceipt
ticketNumber: int
coupon: Coupon
TicketReceipt
ticketNumber: int
AnnualReceipt
year: int
TheatreModel
IssueNewsController
news:News
IssueNewsController(movieNews: File)
issueNews(emails: ArrayList<String>):Movie
News
fname: String
content: String
News(movieNews: File)
encode(): JsonObject
BuildTicket
ticket:Ticket
BuildTicket():void
getTicket():JsonObject
Ticket
movie:Movie
theatre:Theatre
showTime:ShowTime
seat:Seat
Ticket():void
getMovie():Movie
getTheatre():Theatre
getShowTime():ShowTime
getSeat():Seat
setMovie(movie:Movie)
setTheatre(theatre:Theatre)
setShowTime(time:Date)
setSeat(seat:Seat)
BrowseMovie
movieList: ArrayList<Movie>
isRegUser: boolean
BrowseMovie(sqlSet: ResultSet, isRegUser: boolean)
parseMovieList(sqlSet: ResultSet)
getMovieList():JsonObject
selectMovie(sqlMovie:ResultSet): Movie
SelectTheatre
theatreList: ArrayList<Theatre>
SelectTheatre(sqlSet: ResultSet)
parseTheatreList(sqlSet: ResultSet)
getTheatreList():JsonObject
selectTheatre(sqlTheatre:ResultSet): Theatre
SelectShowTime
showTimeList: ArrayList<ShowTime>
SelectShowTime(sqlSet: ResultSet)
parseShowTimeList(sqlSet: ResultSet):void
getShowTimeList():JsonObject
selectShowTime(sqlShowTime:ResultSet): ShowTime
SelectSeat
showSeatList: ArrayList<Seat>
SelectSeat(sqlSet: ResultSet)
parseSeatList(sqlSet: ResultSet):void
getSeatList():JsonObject
selectSeat(sqlSeat:ResultSet): Seat
Movie
movieName: String
moviePrice: double
earlyAccess: boolean
theatreList: ArrayList<Theatre>
getMovieName():String
getMoviePrice():int
getEarlyAccess(): boolean
setMovieName(name:String):void
setMoviePrice(price:double):void
setEarlyAccess(status:boolean):void
Theatre
theatreName: String
Theatre(theatreResultSet:ResultSet)
getTheatreName():String
setTheatreName(name:String):void
ShowTime
time: Date
earlyAccessMovie: boolean
earlyAccessSeatsAvailable:boolean
isFull: boolean
ShowTime(showTimeResultSeat)
getTime():Date
getEarlyAccessMovie():boolean
getEarlyAccessSeatsAvailable():boolean
getIsFull():boolean
setTime(tune: Date):void
setEarlyAccessMovie(status: boolean):void
setEarlyAccessSeatsAvailable(status: boolean): void
setIsFull(status: boolean): void
Seat
seatStatus: boolean
seatRow: int
seatCol: int
Seat(seatResultSet: ResultSet)
getSeatRow():int
getSeatCol():int
getSeatStatus(): boolean
setSeatRow(row: int):void
setSeatCol(col: int):void
setSeatStatus(status: boolean): void
User
ticket: Ticket
paymentInfo: PaymentInfo


View
ViewControllers
GUIController
- gui: GUI
+ GUIController()
+ main(args: String[]): void
+ getGUI(): GUI
+ showView(view: String):void
+ login(userName: String, password: String): JsonObject
+ getMovieList(): JsonObject
+ selectMovie(name: String): JsonObject
+ selectTheatre(theatreId: int): JsonObject
+ selectShowTime(showtimeDate: Date): JsonObject
+ selectSeat(seatRow: int, seatCol: int)
+ checkout(): JsonObject
+ refundTicket(ticketId: number): JsonObject
+ registerUser(name: String, password: String, cardType: String, cardNum: String, email: String)
LoginViewController
- guiController: GUIController
- view: LoginView
+ LoginViewController(view: LoginView, guiController: GUIController)
+ getView(): LoginView
+ actionPerformed(e: ActionEvent): void
IssueNewsViewController
- guiController: GUIController
- view: IssueNewsView
+ IssueNewsController(view: IssueNewsView, guiController: GUIController)
+ getView(): IssueNewsView
+ actionPerformed(e: ActionEvent): void
RegisterViewController
- view: RegisterView
- guiController: GUIController
+ IssueNewsController(view: IssueNewsView, guiController: GUIController)
+ getView(): RegisterView
+ actionPerformed(e: ActionEvent): void
+ verifyInput(): boolean
MovieViewController
- view: MovieView
- guiController: GUIController
+ MovieViewController(view: MovieView, guiController: GUIController)
+ paintMovies(movies: JsonObject): void
+ actionPerformed(e: ActionEvent): void
TheatreViewController
- view: TheatreView
- guiController: GUIController
+ TheatreViewController(view: TheatreView, guiController: GUIController)
+ paintTheatres(theatres: JsonObject): void
+ actionPerformed(e: ActionEvent): void
ShowtimeViewController
- view: ShowtimeView
- guiController: GUIController
+ ShowtimeViewController(view: ShowtimeView, guiController: GUIController)
+ paintShowtimes(showtimes: JsonObject): void
+ actionPerformed(e: ActionEvent): void
SeatViewController
- seat: SeatView
- guiController: GUIController
+ SeatViewController(view: SeatView, guiController: GUIController)
+ getView(): SeatView
+ paintSeats(seats: JsonObject): void
+ actionPerformed(e: ActionEvent): void
PaymentViewController
- view: PaymentView
- guiController: GUIController
+ PaymentViewController(view: PaymentView, guiController: GUIController)
+ getView(): PaymentView
+ flashMessage(message: JsonObject): void
+ actionPerformed(e: ActionEvent): void
RefundViewController
- view: RefundView
- guiController: GUIController
+ RefundViewController(view: RefundView, guiController: GUIController)
+ getView(): RefundView
+ paintTicket(ticket: JsonObject): void
+ actionPerformed(e: ActionEvent): void
MenuViewController
- view: MenuViewController
- guiController: GUIController
+ MenuViewController(view: MenuView)
+ getView(): MenuViewController
+ setRegisteredView(): void
+ setManagerView(): void
+ setBasicView(): void
+ actionPerformed(e: ActionEvent): void
Views
GUI
- views: HashMap<String, View>
- container: JPanel
- userStatus: JLabel
+ GUI()
+ getContainer(): JPanel
+ getView(key: String): View
+ getUserStatus(): JLabel
+ showView(key: String): void
View
- titleLabel: JLabel
- menuButton: JButton
- submitButton: JButton
+ getPanel(): JPanel
+ registerActionListener(listener: ActionListener, component: JComponent)
+ flashMessage(e: ActionEvent): void
+ verifyInput()
LoginView
- usernameLabel: JLabel
- passwordLabel: JLabel
- usernameField: JTextField
- passwordField: JTextField
+ LoginView()
+ getUsernameField(): JTextField
+ getPasswordField(): JTextField
IssueNewsView
- uploadButton: JButton
- newsTextArea: JTextArea
+ IssueNewsView()
+ getNewsTextArea(): JTextArea
RegisterView
- usernameLabel: JLabel
- passwordLabel: JLabel
- nameLabel: JLabel
- addressLabel: JLabel
- emailLabel: JLabel
- cardNumLabel: JLabel
- cardTypeLabel: JLabel
- usernameField: JTextField
- passwordField: JTextField
- nameField: JTextField
- addressField: JTextField
- cardNumField: JTextField
- cardTypeField: JTextField
- emailField: JTextField
+ RegisterView()
+ getUsernameField(): JTextField
+ getNameField(): JTextField
+ getAddressField(): JTextField
+ getCardNumberField(): JTextField
+ getCardTypeField(): JTextField
+ getEmailField(): JTextField
MovieView
- movieLabel: JLabel
- movieTextArea: JTextArea
- selectionLabel: JLabel
- selectionField: JTextField
+ MovieView()
+ getSelectionField(): JTextField
+ getTextArea(): JTextArea
TheatreView
- theatreLabel: JLabel
- theatreTextArea: JTextArea
- selectionLabel: JLabel
- selectionField: JTextField
+ TheatreView()
+ getSelectionField(): JTextField
+ getTextArea(): JTextArea
ShowtimeView
- showtimeLabel: JLabel
- showtimeTextArea: JTextArea
- selectionLabel: JLabel
- selectionField: JTextField
+ ShowtimeView()
+ getSelectionField(): JTextField
+ getTextArea(): JTextArea
SeatView
- seatLabel: JLabel
- seats: HashMap<int[], JButton>
+ SeatView()
+ getSeats(): HashMap<int[], JButton>
PaymentView
- cardNumLabel: JLabel
- cardTypeLabel: JLabel
- cardNumField: JTextField
- cardTypeField: JTextField
+ PaymentView()
+ getCardNumberField(): JTextField
+ getCardTypeField(): JTextField
RefundView
- receiptLabel: JLabel
- receiptField: JTextField
- ticketLabel: JLabel
- ticketField: JTextField
+ RefundView()
+ getReceiptField(): JTextField
+ getTicketField(): JTextField
MenuView
- loginLabel: JLabel
- selectMovieLabel: JLabel
- checkoutLabel: JLabel
- registerUserLabel: JLabel
- refundLabel: JLabel
- payAnnualLabel: JLabel
- issueNewsLabel: JLabel
- logoutLabel: JLabel
- loginButton:JButton
- selectMovieButton:JButton
- checkoutLabel:JButton
- registerUserButton:JButton
- refundButton:JButton
- payAnnualButton:JButton
- issueNewsButton: JButton
- logoutButton: JButton
+ MenuView()
+ setRegisteredView(): void
+ setManagerView(): void

Sequence Diagrams (see next 4 pages)

Use Case: Refund Ticket - By: Victor Tuah Kumi

See `./lib/SequenceDiagrams/KumiRefundTicketSequenceDiagram.pdf`

Use Case: Select Ticket - By: Patrick Kwan

See `./lib/SequenceDiagrams/KwanSelectTicketSeqeunceDiagram.pdf`

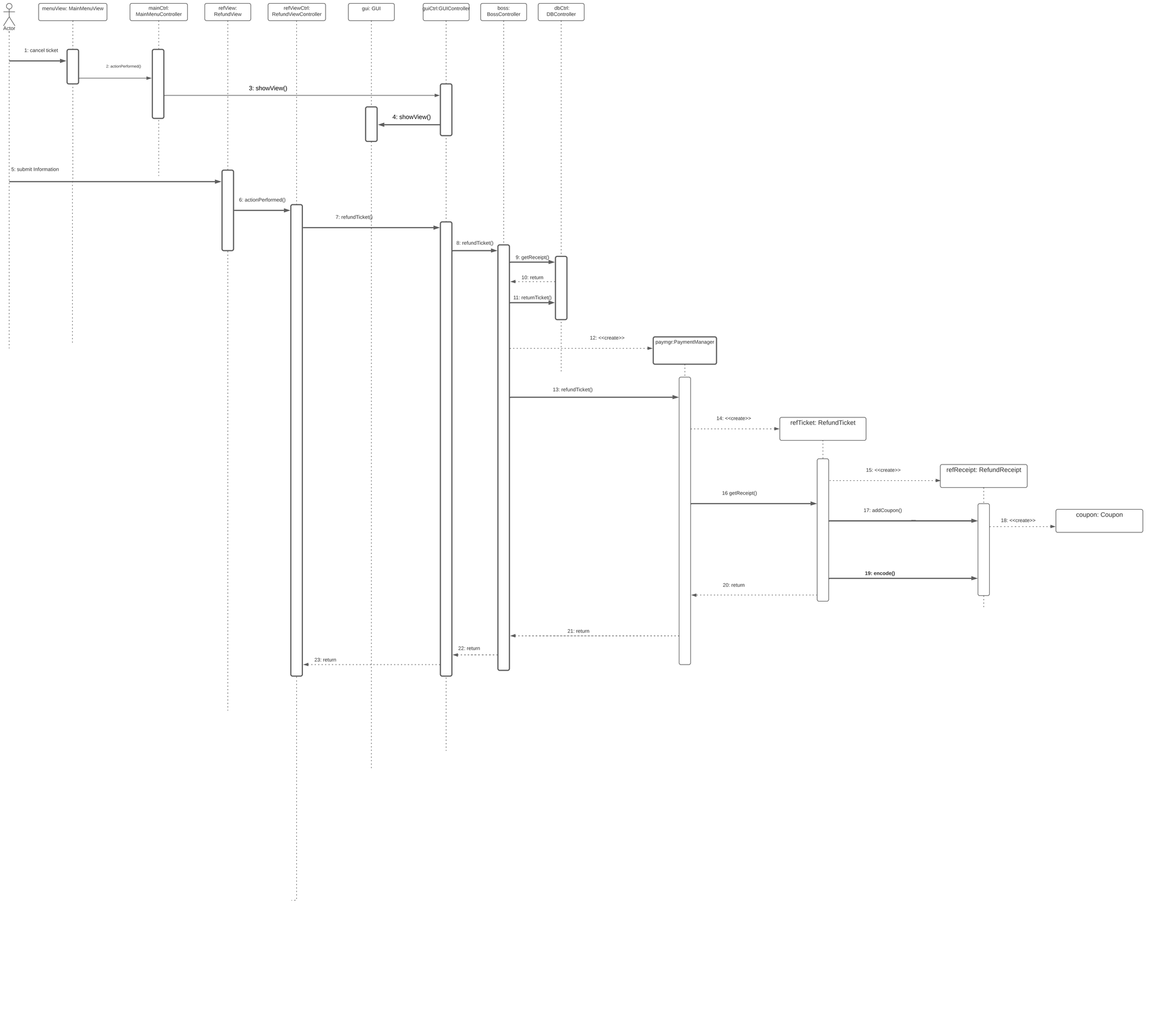
Use Case: Checkout - By: Oluwapelumi Laditan

See `./lib/SequenceDiagrams/LaditanCheckoutSequenceDiagram.pdf`

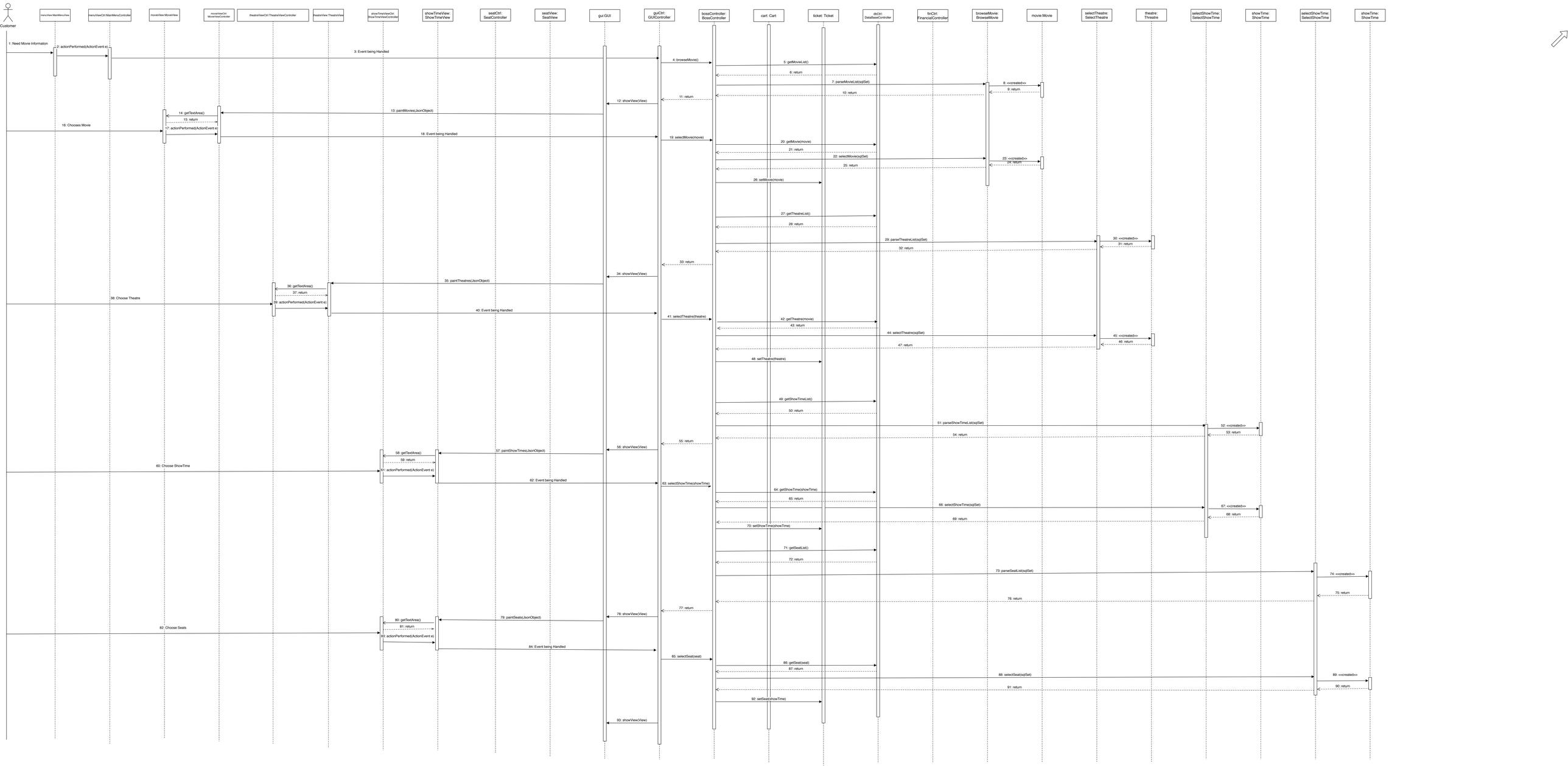
Use Case: Register User - By: Michael Lasby

See `./lib/SequenceDiagrams/LasbyRegisterUserSequenceDiagram.pdf`

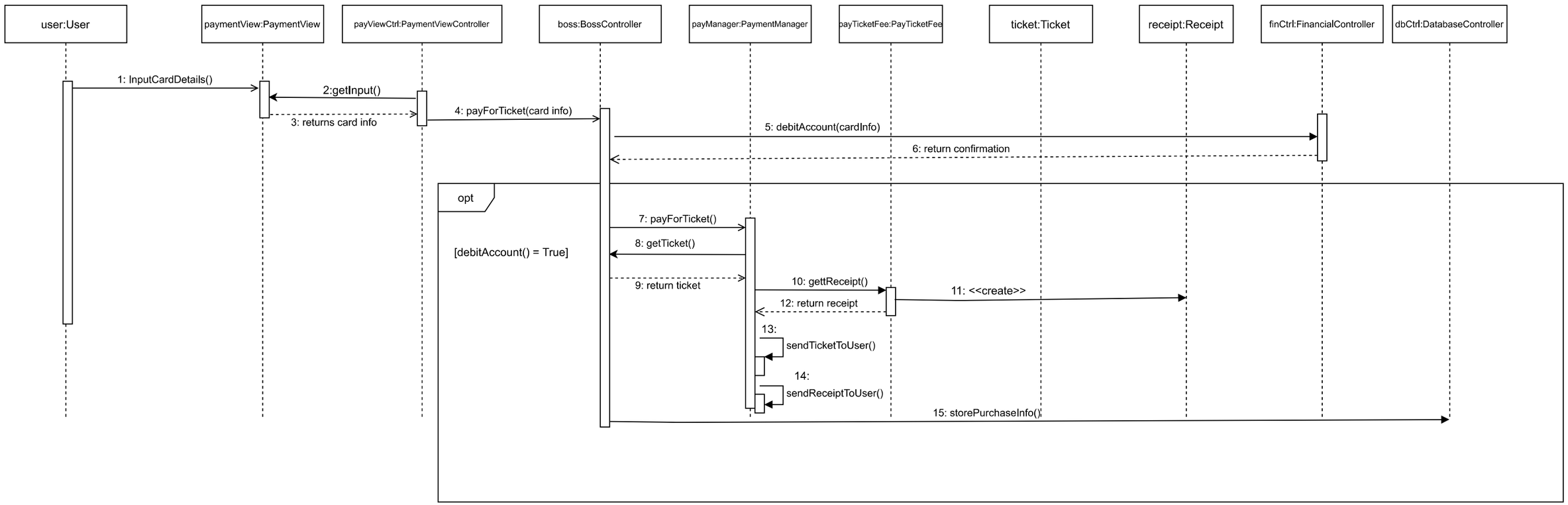
Refund Ticket Sequence Diagram- By: Victor Tuah Kumi

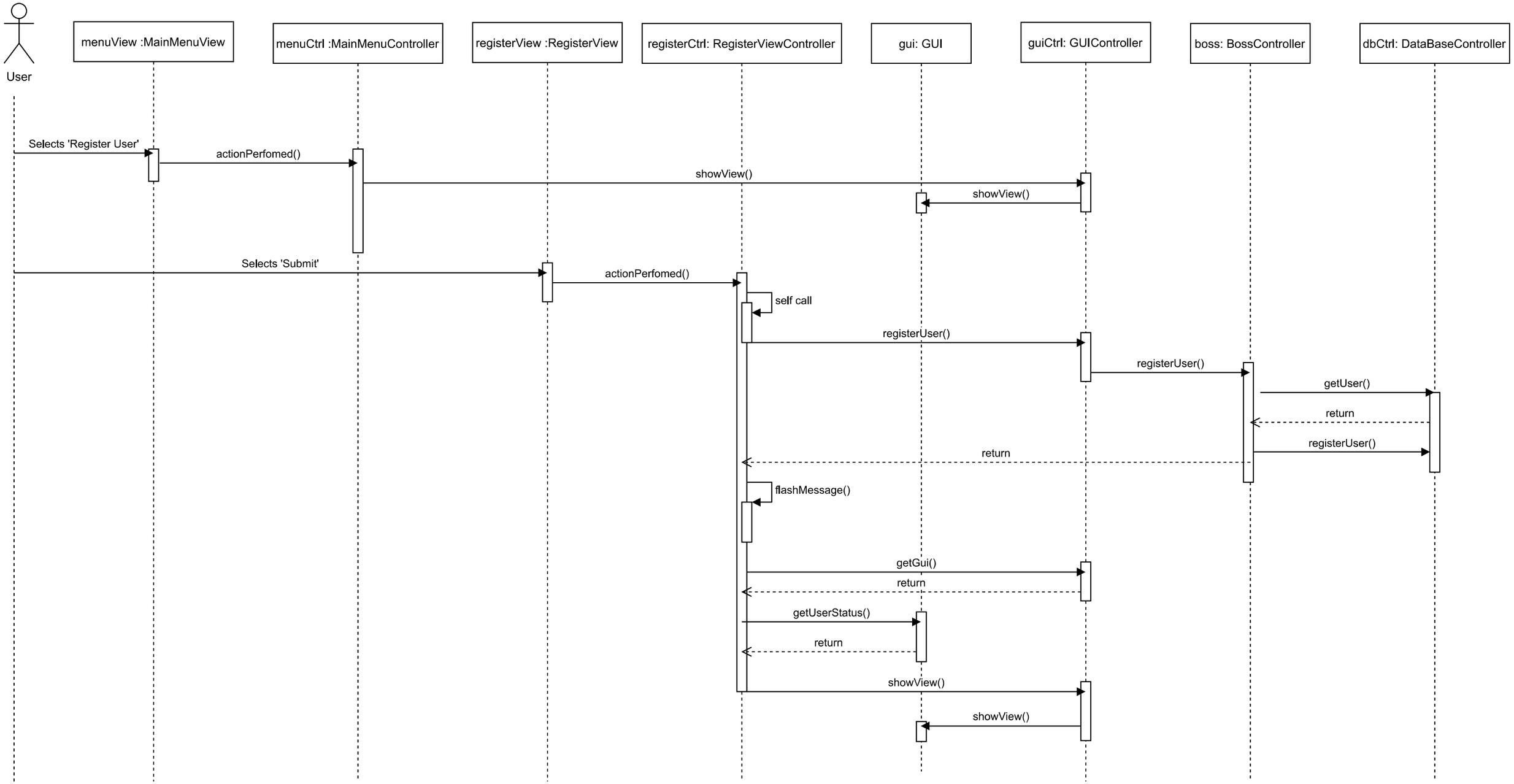


Select Ticket Sequence Diagram - By: Patrick Kwan



Checkout Sequence Diagram - By: Oluwapelumi Laditan

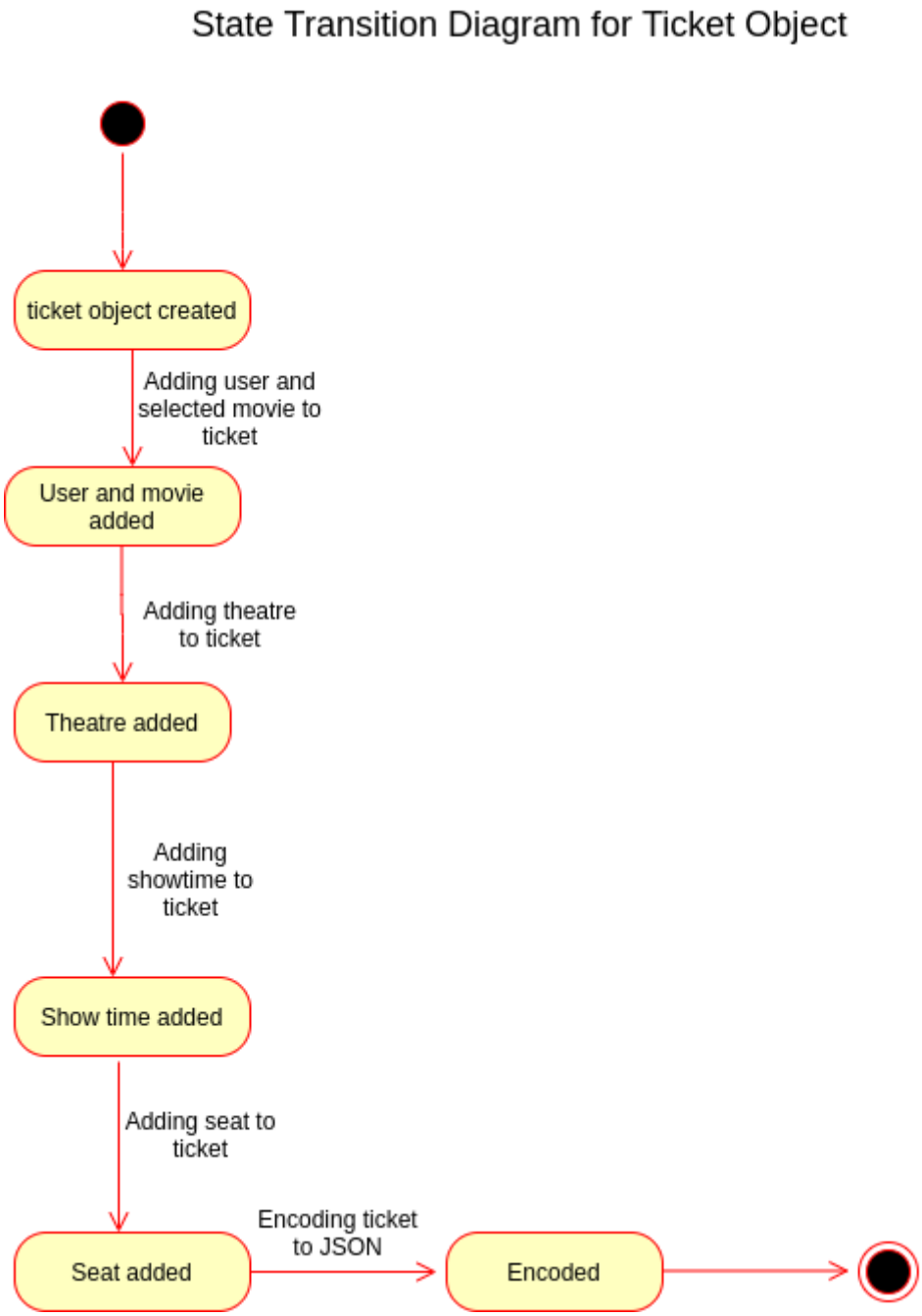



User
menuView :MainMenuView
menuCtrl :MainMenuController
registerView :RegisterView
registerCtrl: RegisterViewController
gui: GUI
guiCtrl: GUIController
boss: BossController
dbCtrl: DataBaseController
Selects 'Register User'
actionPerfomed()
showView()
showView()
Selects 'Submit'
actionPerfomed()
self call
registerUser()
registerUser()
getUser()
return
registerUser()
return
flashMessage()
getGui()
return
getUserStatus()
return
showView()
showView()

State Transition Diagrams

Ticket

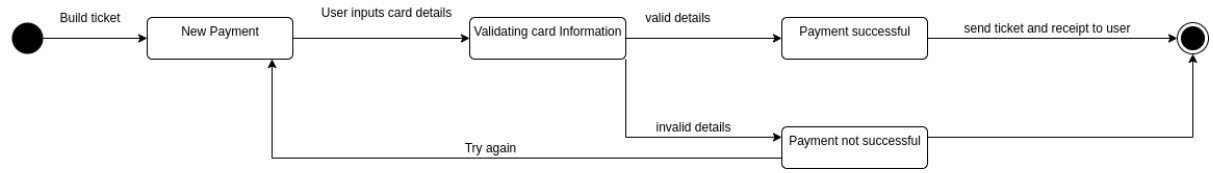
See `./lib/StateTransition/TicketStateTransition.png`



Payment

See `./lib/StateTransition/PaymentStateTransition.png`

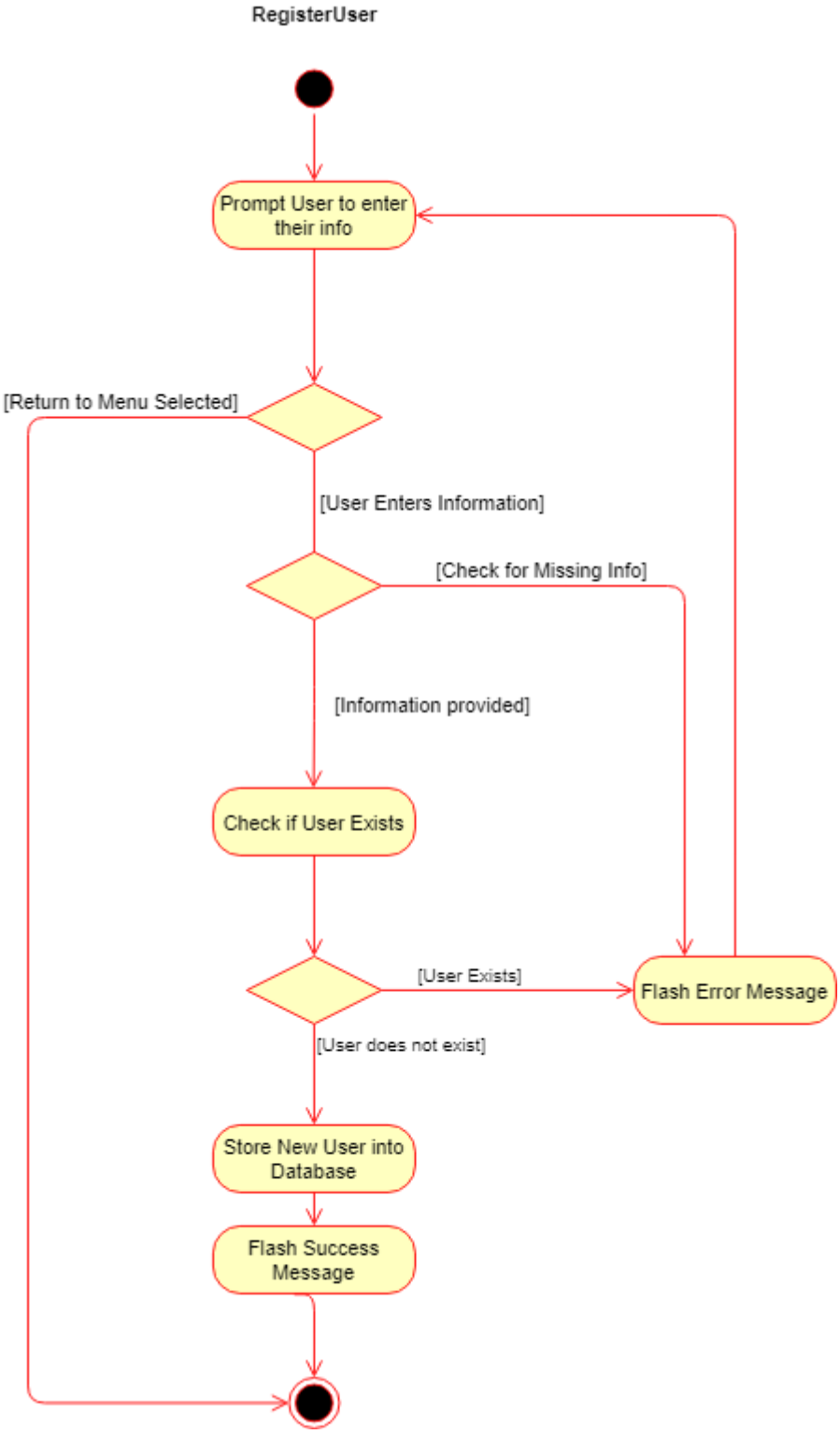
State Transition Payment



Activity Diagrams

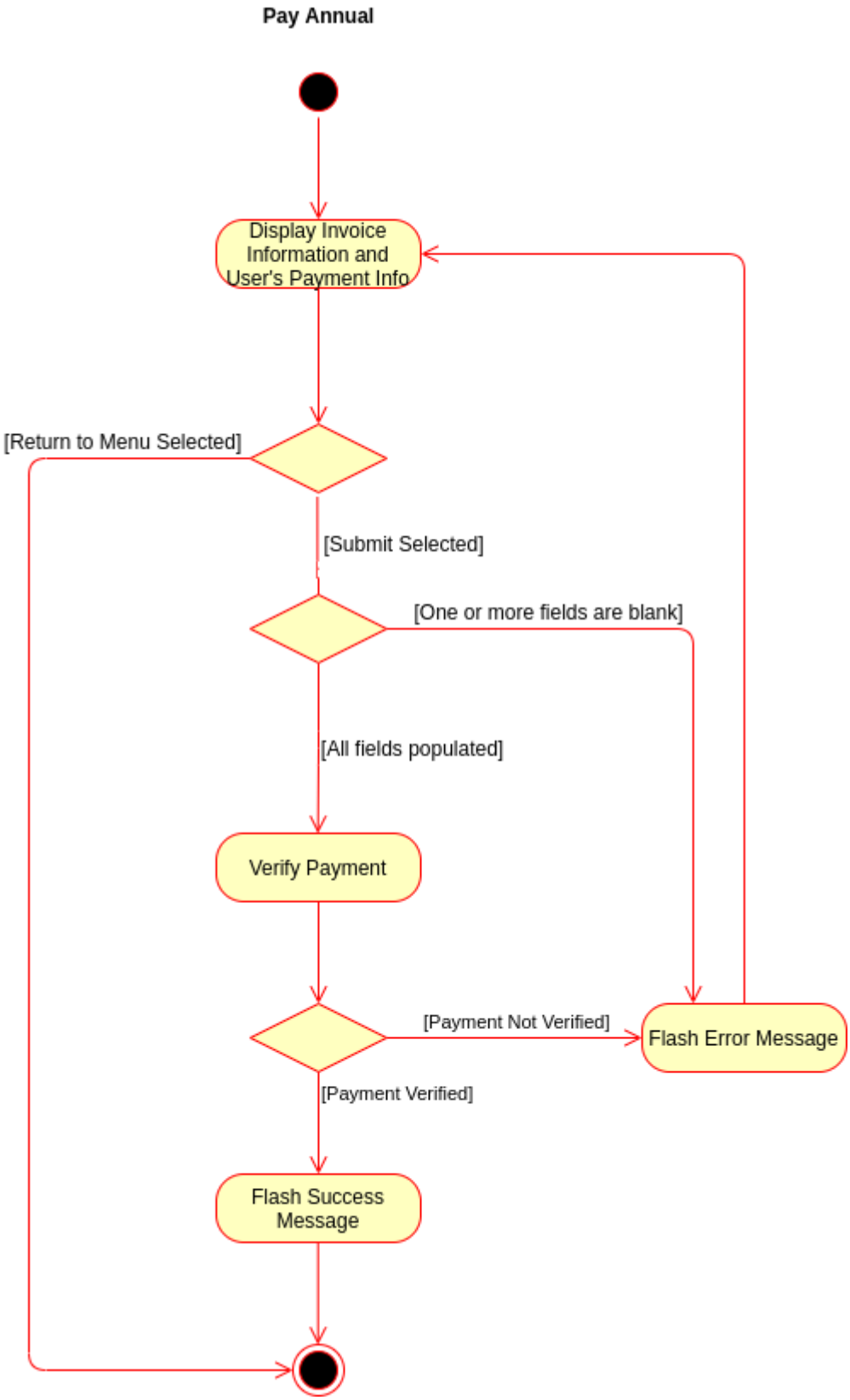
Register Account

See `./lib/ActivityDiagrams/RegisterUserActivityDiagram.png)`



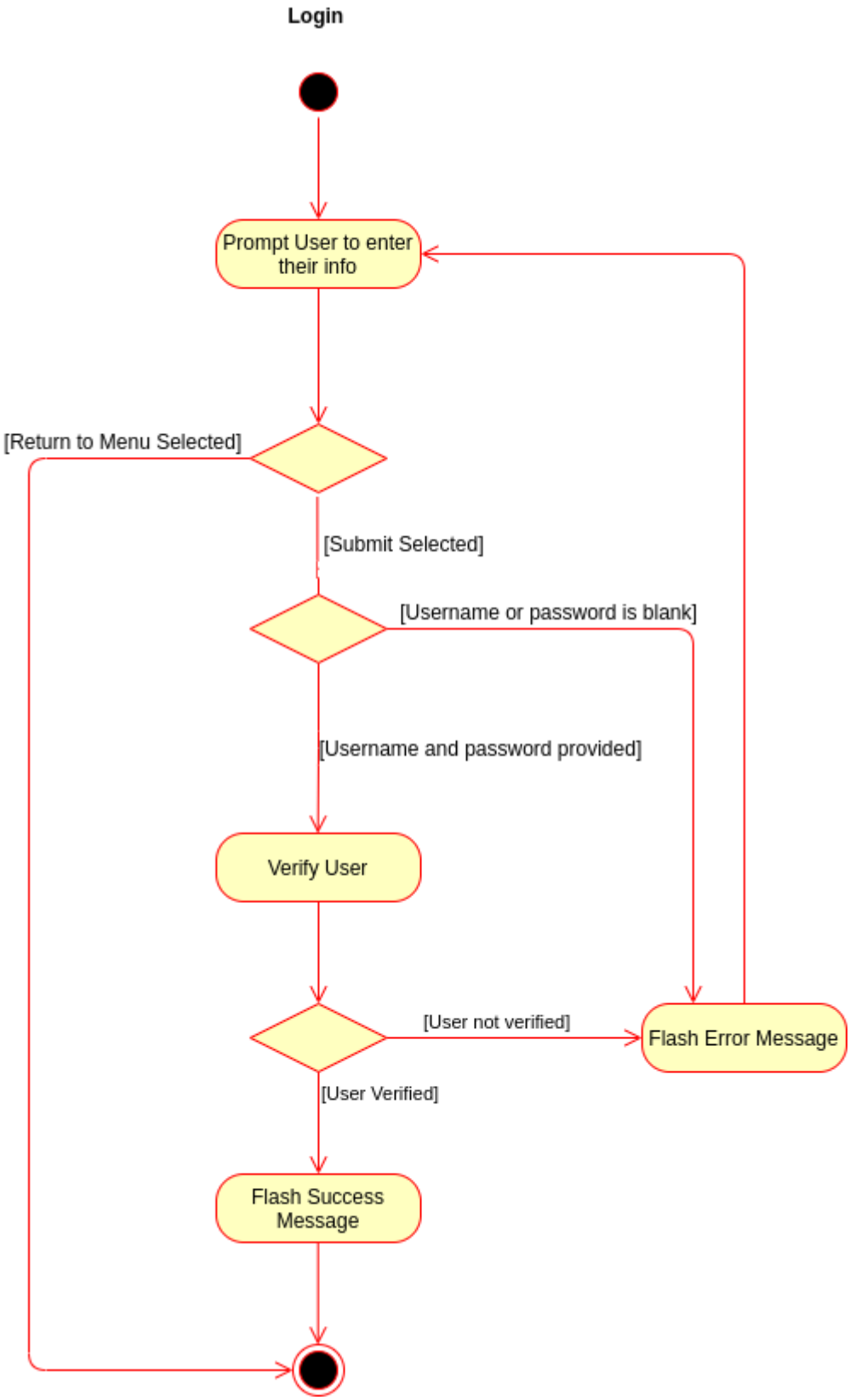
Pay Annual

See `./lib/ActivityDiagrams/RegisterUserActivityDiagram.png)`



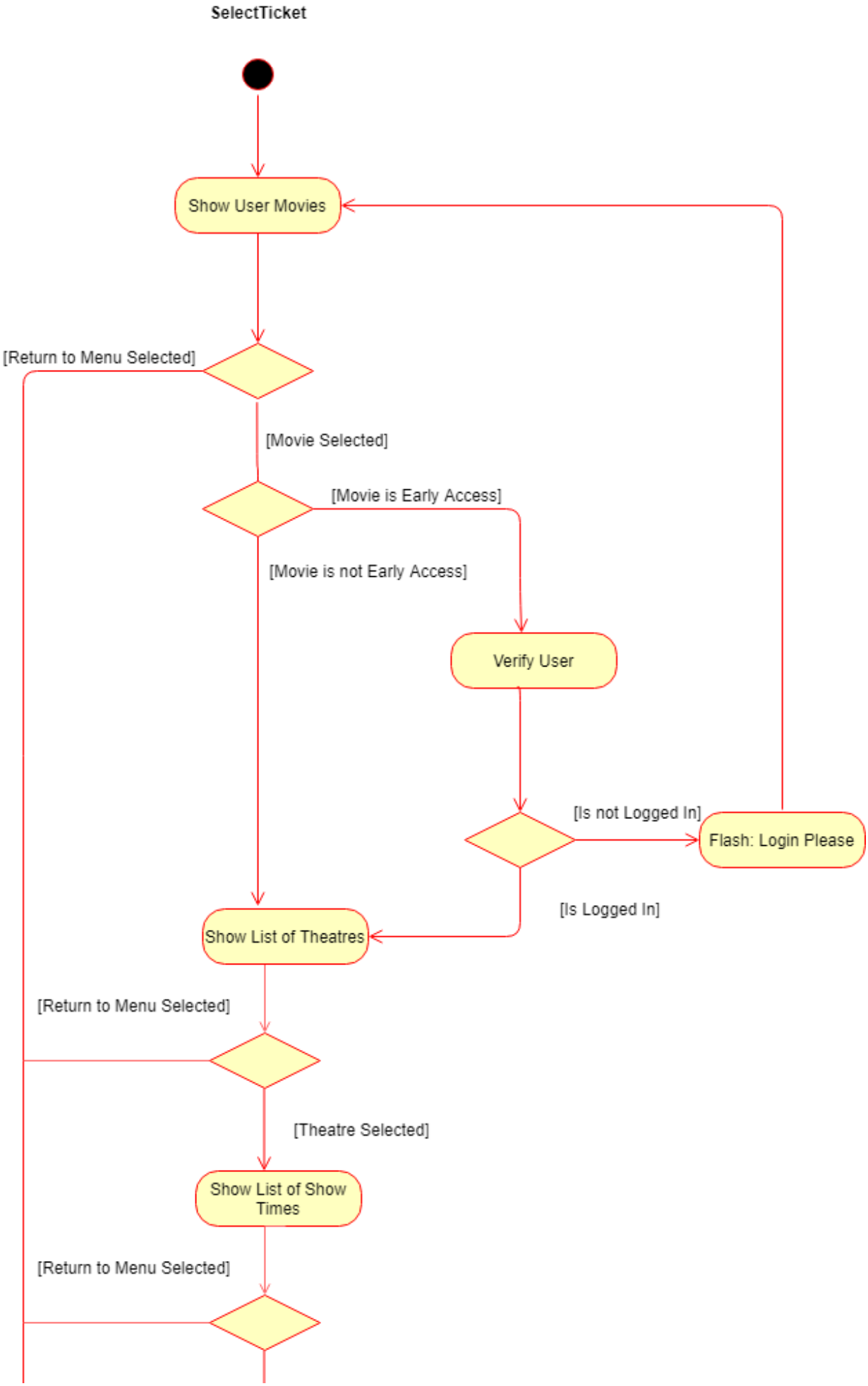
Login

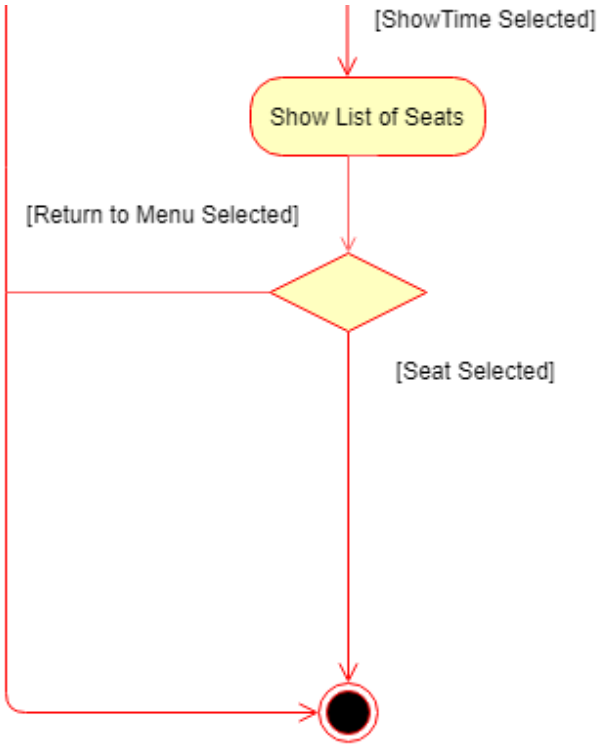
See `./lib/ActivityDiagrams/LoginActivityDiagram.png)`



Select Ticket

See `./lib/ActivityDiagrams/TicketSelectActivityDiagram.png)`

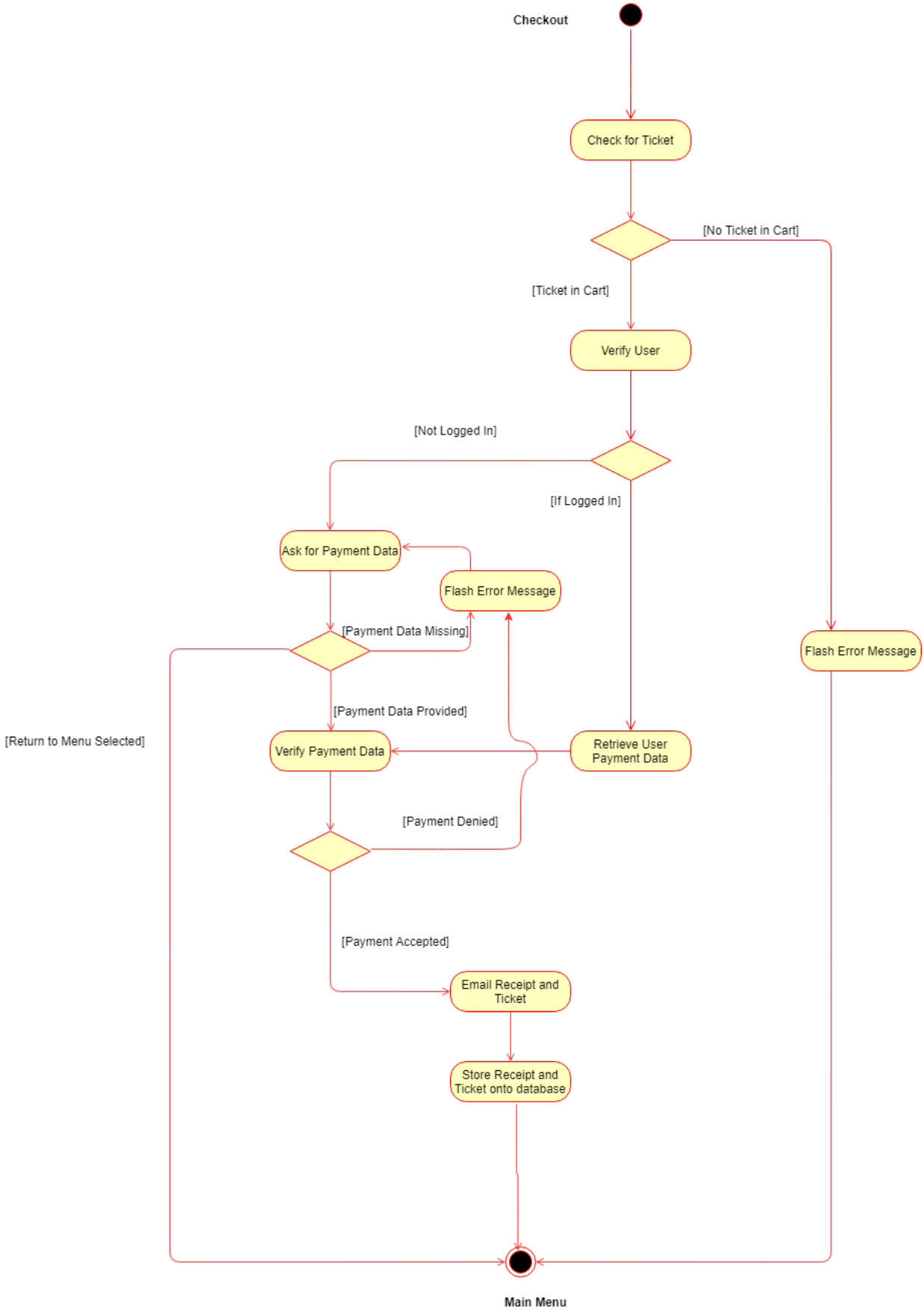



[ShowTime Selected]
Show List of Seats
[Return to Menu Selected]
[Seat Selected]

Main Menu

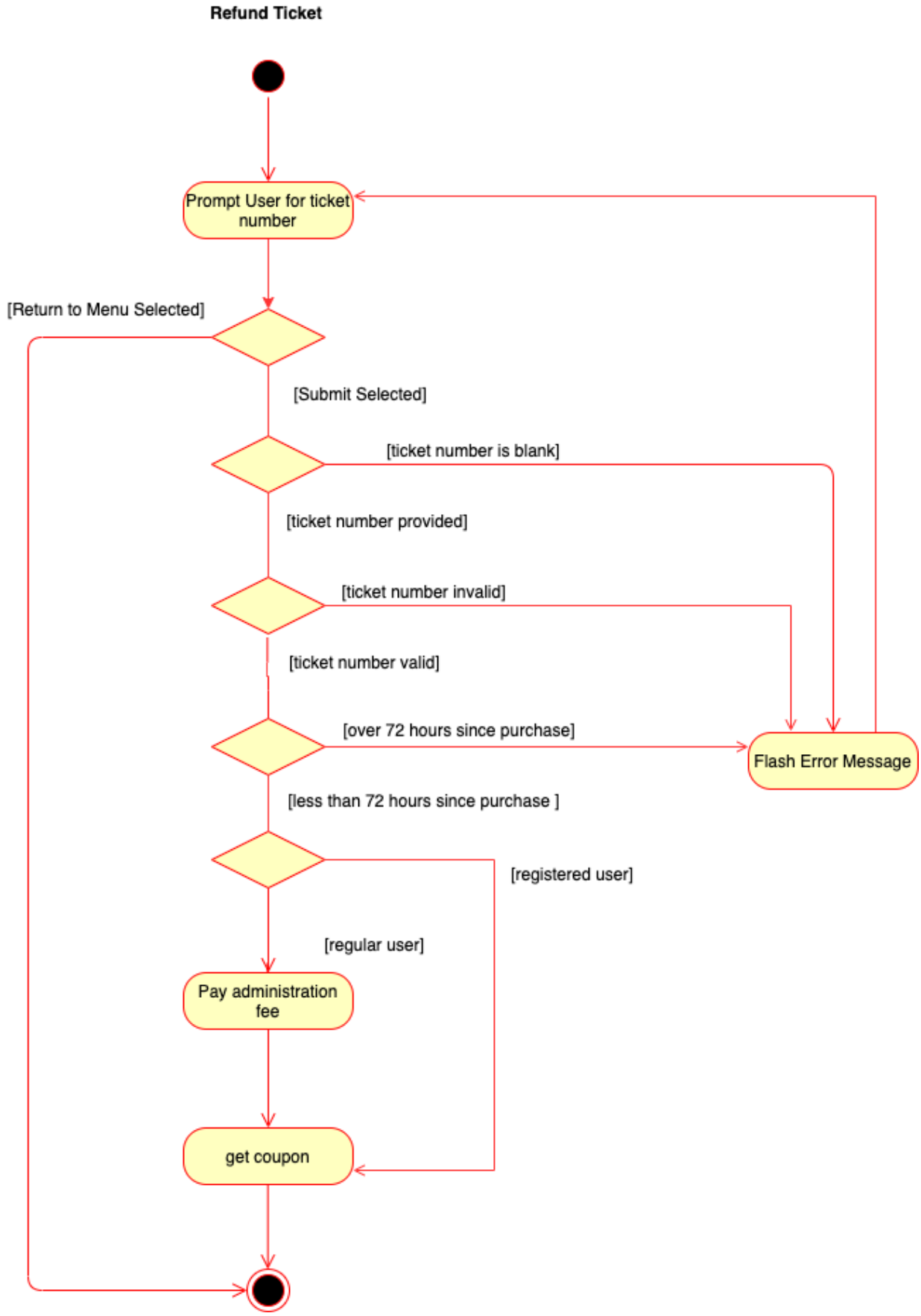
Checkout

See `./lib/ActivityDiagrams/CheckoutActivityDiagram.png)`



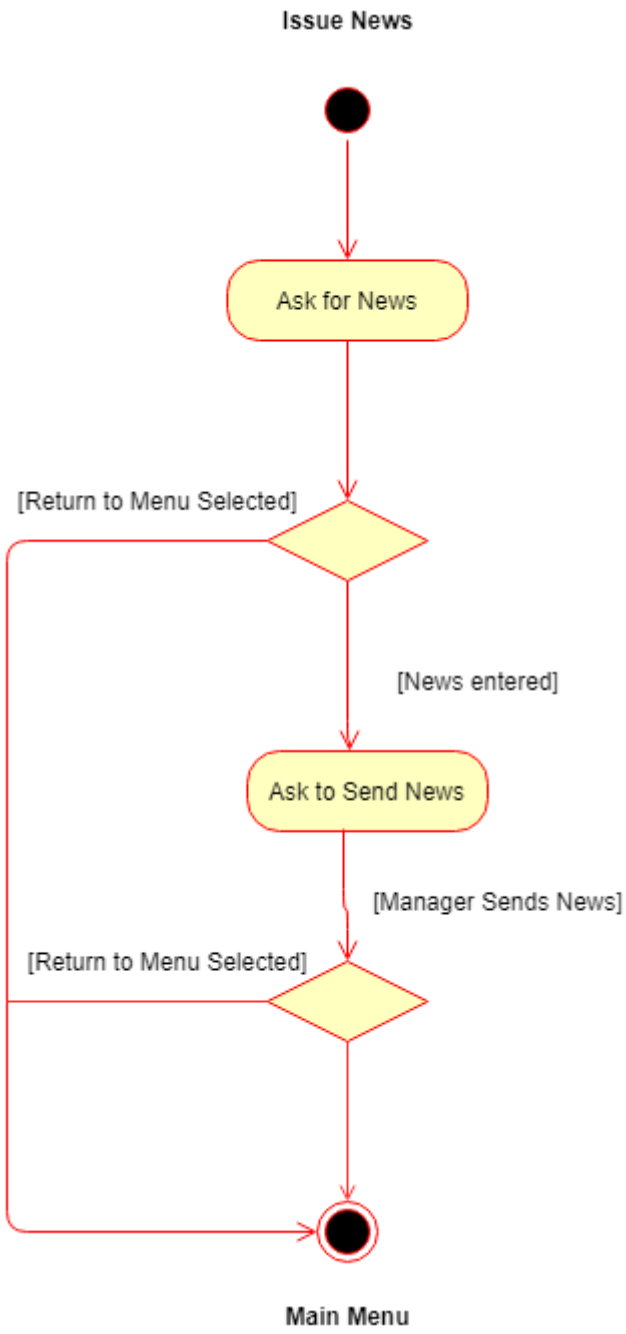
Refund Ticket

See `./lib/ActivityDiagrams/RefundTicketActivityDiagram.png)`



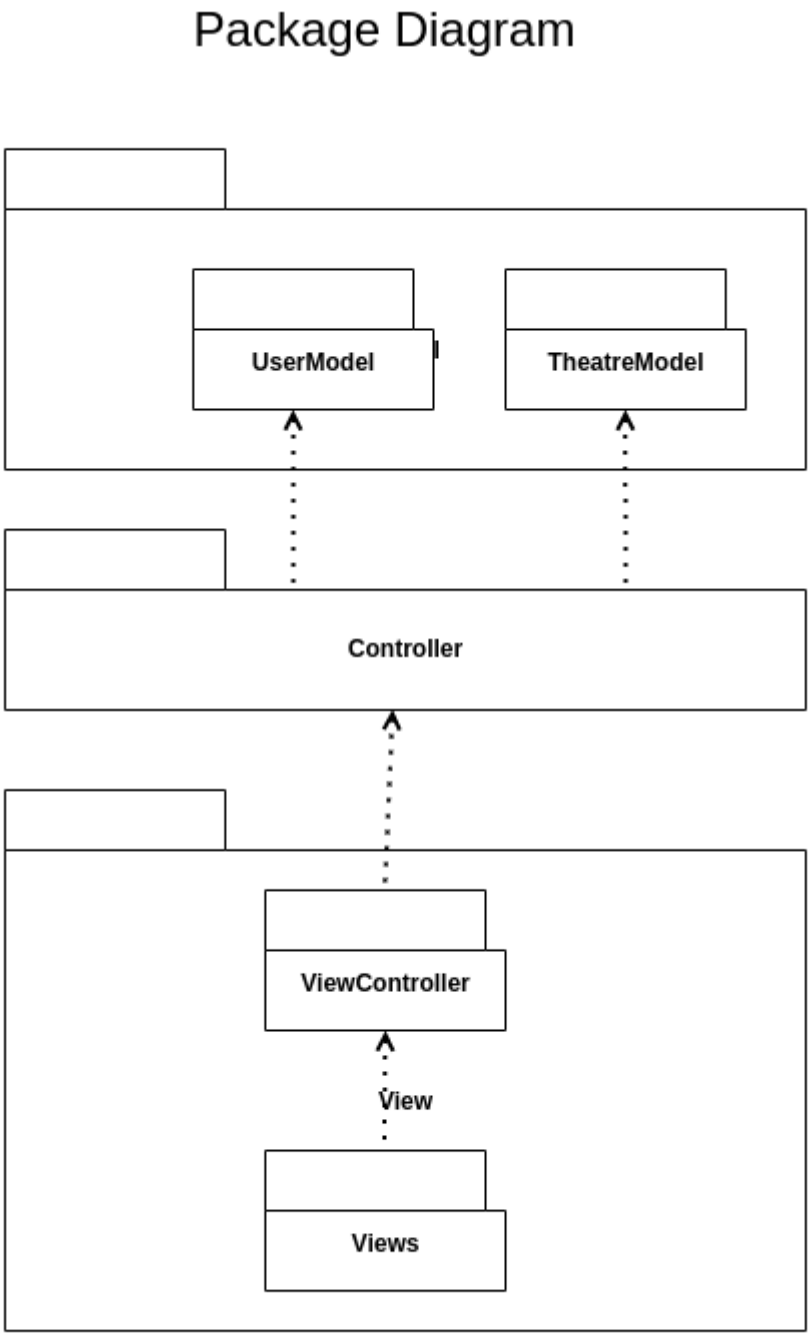
Issue Movie News

See `./lib/ActivityDiagrams/IssueNewsActivityDiagram.png)`



Package Diagram

See ./lib/PackageDiagram/PackageDiagram.png)



Deployment Diagram

See ./lib/PackageDiagram/DeploymentDiagram.png)