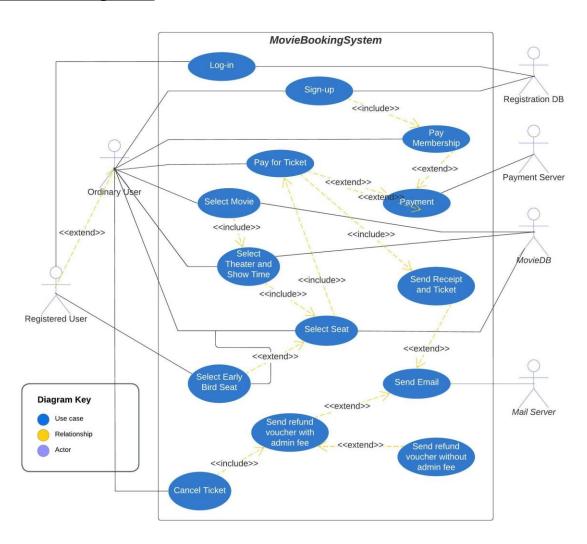
Design Documents

Use-case Diagram:



Use-case Scenarios:

Sign up

- o Ordinary User presses register button
- System asks for account information
 - Id, password, name, address, credit/debit card
- o <u>Validate</u> id and password
 - If not validated, <u>asks</u> the <u>User</u> for a different <u>Username</u> and/or <u>password</u>
- o <u>User presses</u> next button
- o System contacts database to store account info
- o System asks User for membership payment
- User <u>inputs</u> <u>payment info</u> and <u>server contacts</u> the <u>payment server</u> to complete the <u>payment</u>
- o <u>Program contacts mail server</u> to <u>send Confirmation email</u>
- o New page showing User registration is complete

Login

- o Ordinary User presses login button
- o System asks for Username/email and password
- o System contacts User database to validate Username and password
 - If <u>Username</u> and <u>password</u> are valid, <u>System displays</u> a <u>success message</u> and <u>returns registered User</u> to <u>home page</u>
 - If <u>Username</u> and/or <u>password</u> are invalid, <u>displays</u> an <u>error message</u> and <u>prompts</u> for the <u>Username</u> and <u>password</u> again

Buy Movie Ticket

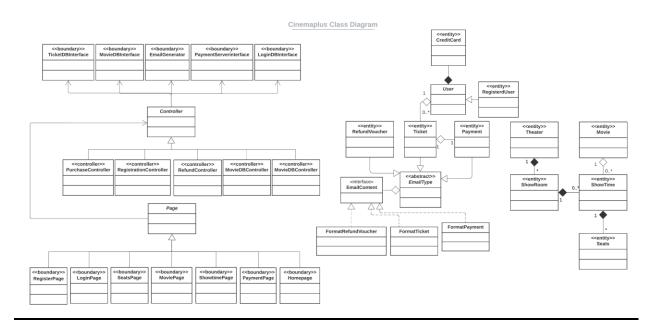
- User opens theatre tab
- System contacts theatre database
- System displays theatres
- o User selects theatre
- o User opens movie tab
- o System contacts movie database
- o System displays movies
- o <u>User selects movie</u>
- System contacts showtimes database
- o System displays showtimes
- User <u>selects</u> showtime
- o System contacts seats availability database
- System <u>displays</u> seat availability graphically
 - If <u>User</u> is registered then <u>show</u> <u>early bird seats</u>
- User selects seat
- o System opens payment tab
- System asks for payment
 - If ordinary <u>User</u>, <u>inputs</u> <u>credit card details</u> and <u>email address</u> for <u>tickets</u> and <u>receipt</u>
 - If registered <u>User</u>, <u>contacts</u> <u>User database</u> for <u>payment details</u> and <u>contact details</u>
- o System contacts mail server to send email with tickets and receipt

Refund Movie Ticket

- User presses refund button
- System asks for ticket details
- o System checks if the movie time is within the 72 hours period

- If so, <u>reject</u> <u>refund</u>
- System checks if the User is a registered User
 If so, exempt 15% admin fee
 System contacts email server to send the refund voucher with 1 year expiry date to the User

Design Class-diagram:

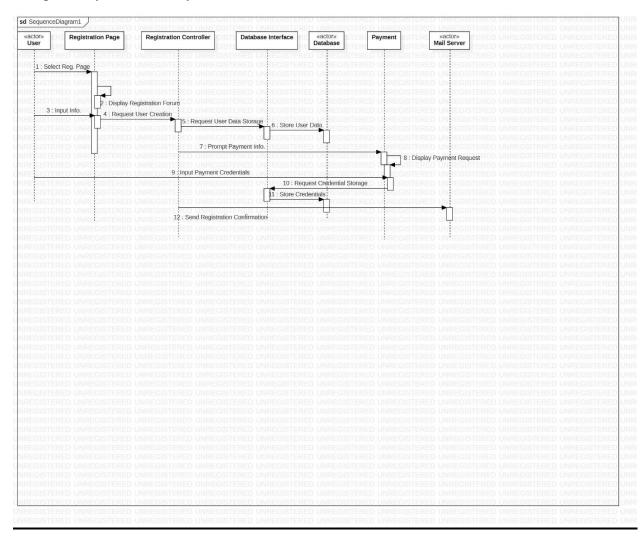


Attributes Class-diagram:

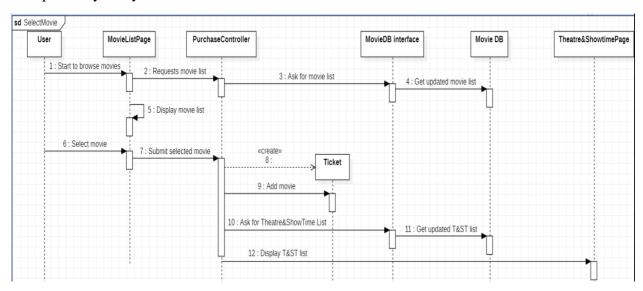
< boundary>> MovieDBInterface	< boundary>> EmailGenerator	< seatsPage	< <entity>> RefundVoucher</entity>	< <controller>> RefundController</controller>
-onlyInstance: MovieDBInterface	-mailClient: String	-seats: Arraylist <int></int>	-voucherAmount: float	
-MovieDBInterface() +getInstance(): MovieDBInterface	-type: EmailType +generateEmail(): void	+displaySeats(): void	-voucherID: int	+processRefund(user: User): void
+setOnlyInstance(onlyInstance: MovieDBInterface) +getMovies(): ArrayList-String> +findShowtimes(movie: String): ArrayList-String> +getSeats(selectedST: String): (seatsGUI??) + getShowtime(selectedST: String): +freeSeats(ticket: Ticket): void	+sendEmail(): void	< <abstract>> EmailType</abstract>	< <box> < < <b< td=""><td><<entity>> Theater</entity></td></b<></box>	< <entity>> Theater</entity>
	< <box><box> PaymentServerInterface</box></box>	-contentType: EmailContent	RegisterPage -user: User	-name: String -movies: Arraylist <movie> -showRooms: ArrayList<showroom></showroom></movie>
		+formatContent:void		
< <box> LoginDBInterface</box>		< boundary>> MoviePage		+addMovie(movie: Movie): void -removeMovie(movie: Movie): void
	+processPayment(): boolean	-movieList: Arraylist <movie></movie>	FormatTicket	
-onlyInstance: LoginDBInterface			-ticket: Ticket	< <entity>> Movie</entity>
	< <entity>></entity>		+formatInfo():String	
-LoginDBInterface() +getInstance(): LoginDBInterface +setOnlyInstance(onlyInstance: LoginDBInterface) +validateID(){ +validatePW(}	Cardinfo	< <entity>> RegisterdUser</entity>		-showrooms: ArrayList <showroom> -title: String</showroom>
	-email: String -cardNumber: String -expiryDate: String -cvv: String	#count: int = 0	«interface» EmailContent	7
		#userID: String #password: String	+formatInfo():String	< <box><box </box dary>></box>
	-postalCode: String	#cardNumber: String #expiryDate: String	FormatPayment	ShowtimePage
< toundary>> TicketDBInterface		#cvv: String #phone: String	-payment: Payment	-showtimes: ArrayList <string> -pc: PurchaseController</string>
	< <bod>< RefundPage</bod>	+addTicket(t: Ticket): boolean	+formatInfo():String	-user: User
-onlyinstance: TicketDBInterface	-TicketNumberEntry: JTextField	< boundary>> PaymentPage	< <entity>> Payment</entity>	
	-pc: PurchaseCotroller -user: User	-price: double -purchasedOn: Date	-card: CardInfo -amountDue: float	< <entity>> ShowTime</entity>
-TicketDBInterface() +getOnlyInstance(): TicketDBInterface +setOnlyInstance(onlyInstance: TicketDBInterface) -findTicket(ticketN: String): Ticket		-paymentStatus: bool	-status: bool	-date: Calendar
	< <entity>></entity>	-transactionID: int	-email: String	-movie: Movie -seats: ArrayList <seat></seat>
	Ticket			
	-movie: String -showroom: String	FormatRefundVoucher	< <controller>></controller>	< <entity>></entity>
< <entity>></entity>	-showTime: Showtime -seat: String	-refundVoucher: RefundVoucher	RegistrationController	Seats
User	-payment: Payment -price: float	+formatInfo():String	+validateID(): void	-seatNum: String -occupied: boolean
-tickets: ArrayList <ticket></ticket>	-ticketID: int	< <bod>< boundary>></bod>	+validatePW(): void	
+addTicket(showTime: Showtime, nOfSeats: int): void +storeSeats(seats: ArrayList <string>): void</string>	+checkValidity(): bool +freeMovieSeat(): void	LoginPage -user: User	+register(): void +login(): void	
		-textField_Username:		
< <controller>> PurchaseController</controller>	< <entity>> ShowRoom</entity>	JTextField -textField_1_Password: JTextField	< Homepage	
user: User	-roomNumber: int	-pc: PurchaseController	-user: User -movies: ArrayList <string></string>	
+retrieveMovies(): void	ArrayList <showtime></showtime>		-pc: PurchaseController	
+retrieveShowtime(movie: String): void +retriveSeats(showtime: Showtime, nOfSeats: int): void +requestPaymentInfo(seats: ArrayList <string>): void +processPayment(card: CardInfo): void</string>	-capacity: int -addShowtime(time,	Page	-goToBook(movie: String): voi	d
	movie): void	+display(): void		
		ruispidy(). Voiu		

Sequence Diagrams:

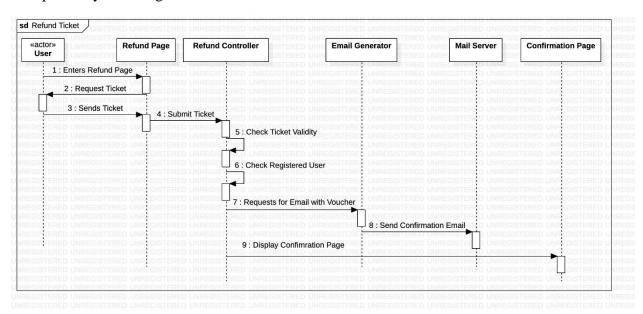
Completed by Braden Foley



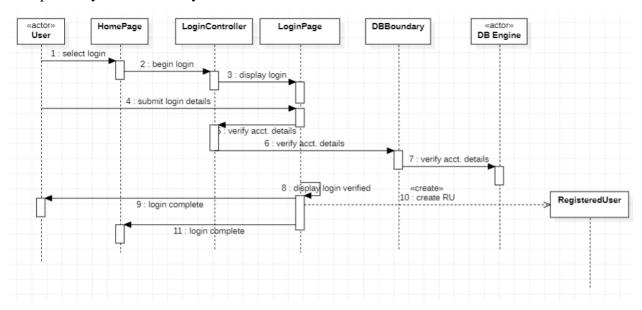
Completed by Mary Mo



Completed by Mark Ngu

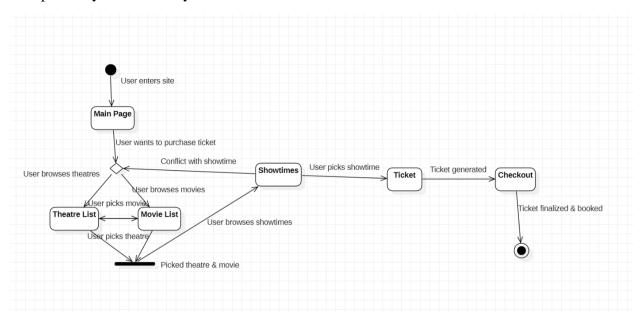


Completed by Max Thorniley

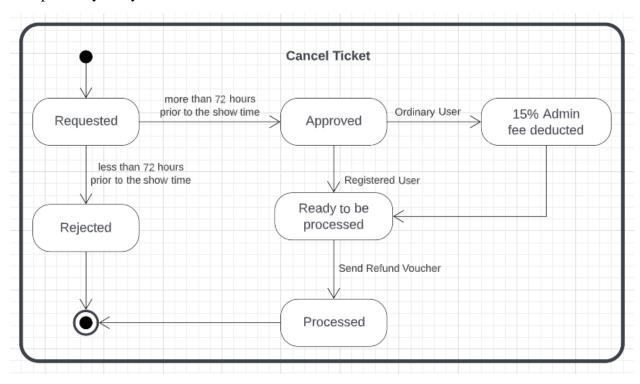


State Transition Diagram:

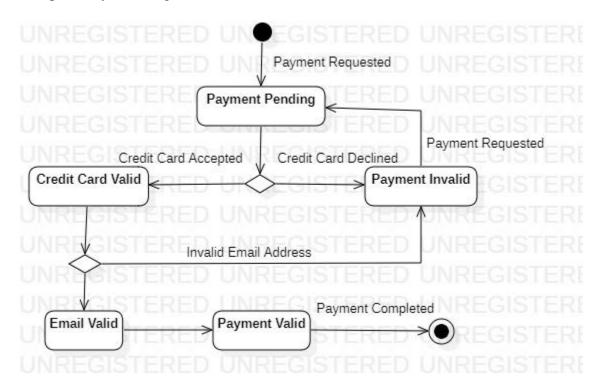
Completed by Braden Foley



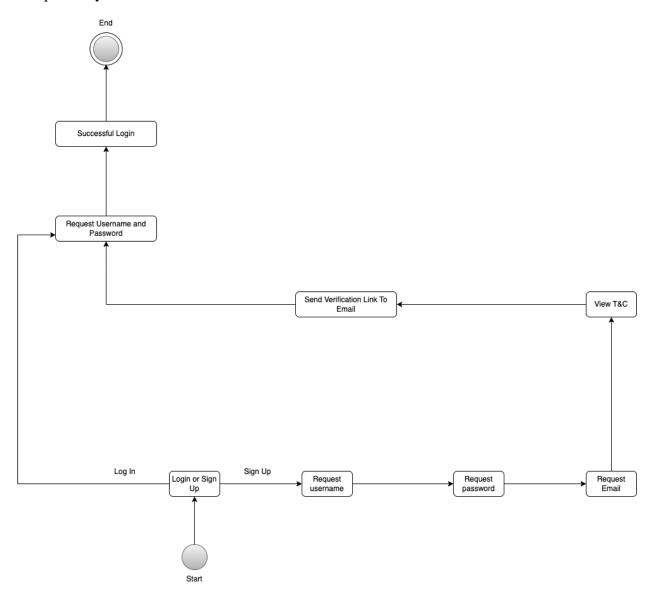
Completed by Mary Mo



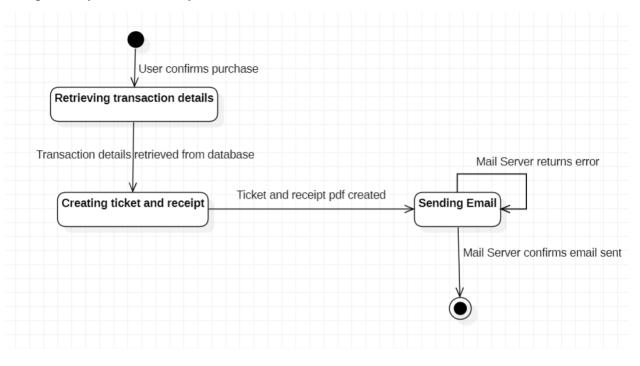
Completed by Mark Ngu



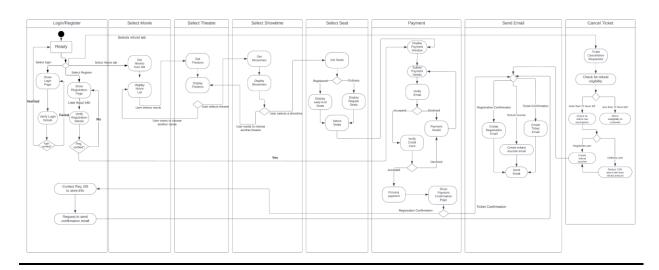
Completed by Sadat Mohtasim



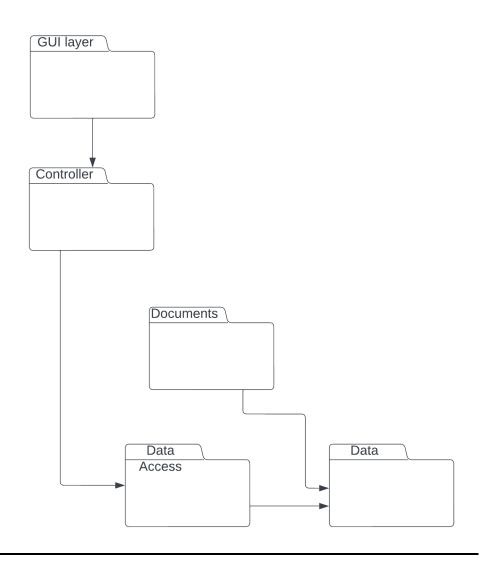
Completed by Max Thorniley



Activity Diagram:



Package Diagram:



Deployment Diagram:

