

Movie Show-Time Finder

Movietickets.com

Refactored-Guacamole

Group 8

Royan Hanson

Micheal Smith

Chad Smith

Yash Shah

Paul Lee

Spring '19

Work Breakdown Structure

Assignee Name	Email	Task	Duration (Hours)	Dependency	Due date	Evaluation
Micheal Smith	msmith303@student.gsu.edu	-4 Implementation	12 hours	None	03/02/19	100%
Paul Lee (Team Coordinator)	plee30@student.gsu.edu	-1 Planning Scheduling -2 Revise and Refine -5 Testing	4 hours	None	03/02/19	100%
Royan Hanson	rhanson3@student.gsu.edu	-Assistance in task 5: Testing -6 Communication and Collaboration	3 hours	None	03/02/19	100%
Yash Shah	yshah4@student.gsu.edu	-3 System Modeling	4.5 hours	None	03/02/19	100%
Chad Smith	csmith324@student.gsu.edu	-4 Implementation	12 hours	None	03/02/19	100%

2. Revise and Refine System

Problem Statement

With an increase of streaming movies at home with a push of a button, purchasing movies should be the same too. Our product is aimed to make it as simple for customers to purchase a movie ticket online. On a high level, our product is a collection of large databases that will be used to collect information. The information gathered from these databases will be used to keep track of current movie times and available tickets at each movie theater. Our website is for customers on both sides of the movie buying spectrum.

Some people are hardcore movie fans that already know what they want to watch and need a simple way of purchasing movie tickets. Then there are people who go out once in a while, who do not know what is available in theaters and want to see trailers of what is showing, which might lead to ticket sales through us. The problem that our website solves is sometimes customers find themselves in need of a simple way to find and buy tickets to their favorite movies. Our website entertains, informs and guides movie fans with must-see trailers and movie clips.

We strive to make it a fun experience to find and buy tickets for the right movie at the right time. Movie Show-Time Finder is available online and through our mobile web app. We have given a lot of thought about making a showtime finder and think it is worth developing because there is a need for it. In the age we live in, many people who are in their 20s or younger do not watch cable television, so they usually do not know what movies are out in theaters. This project will fix that problem by showcasing all the top films that are being shown in theaters, show what movies will be released and will also allow them to purchase tickets with ease.

Our primary objective with creating this project is to give our customers an alternative to buying movie tickets, while also showcasing all the current movies and upcoming movie releases. Our target customers are those who want a more simplistic way of finding showtimes and purchasing tickets for a movie. There are a couple of competitors in this space already, such as Fandango, Atomticket, and Movietickets. What makes us different from our competitors is our website, which is user-friendly and offers a place to watch trailers, find showtimes and purchase tickets. Our system can be built because there are other websites and apps that have already done this, we can use those sites as resources and data for our project.

Use Cases

Use Case 1: Login

Actors: User

Description:

- The system will ask the user to enter their registered username and password.
- The user enters their username and password.
- The system will verify the entered information and log the user in.

Exception Path: If the user is not a registered member then the user will be unable to login.

Alternate Path: If the user decides not to login then the user can exit the login area.

Pre-condition: User must be a registered member.

Post-condition: If the user was successful then the user will be logged in. If not the user will be asked to try again.

Use Case 2: File support ticket

Actors: User

Description:

- The user navigates to support page .
- The user will enter their login credentials and click submit.
- The user will select a support topic from the drop down menu.
- The user will enter a detailed explanation in the body section of the ticket .
- The user will then click submit and take note of the displayed confirmation number.

Exception Path: If the user is not logged in or have given a detailed explanation, the user will be prompted to login or complete the explanation.

Alternate Path: If the user is not a registered user then the user can register on that screen.

Pre-condition: The Login use case must be completed.

Post-condition: User ticket will be submitted and will have a response within 24 hours.

Use Case 3: Search for Theater

Actors: User

Description:

- The user will enter the zip code of where the theater is located.
- The user will choose a search radius from the drop down menu.
- The user will choose a theater from the search results.

Exception Path: If the user does not provide a zip code then the user will not be able to continue the search.

Alternate Path: At anytime the user can go back and change zip code to view available theaters.

Pre-condition: The Login use case must be completed.

Post-condition: A list of theaters will be displayed from the search results.

Use Case 4: Search for Movie**Actors:** User**Description:**

- The user will enter the title of the movie.
- The user will choose the preferred movie title from the search results.

Exception Path: If the user enters a movie currently not in theaters then the user will be prompted to search for another movie.**Alternate Path:** At any time the user can cancel the search and return to the homepage.**Pre-condition:** The Login use case must be completed.**Post-condition:** A list of movies will be displayed after entering the title of the movie.**Use Case 5: View Theaters****Actors:** User**Description:**

- The user will be shown a list of theaters.
- The user will select the theater they want to view the movie in.

Exception Path: None**Alternate Path:** None**Pre-condition:** The user must have completed the Search for Movies use case.**Post-condition:** A list of theaters will be displayed for the user to choose from.**Use Case 6: View Movie Showtimes****Actors:** User**Description:**

- The user will see different showtimes for the movie they have entered.
- The user will select the showtime for the movie they have searched.
- Once the movie showtime has been selected they can proceed to view available seats.

Exception Path: None**Alternate Path:** The user can go back and choose another movie to view a different showtime.**Pre-condition:** The user must have completed the View Theaters use case.**Post-condition:** The user will be able to select the movie showtime they want.

Use Case 7: View Available Seating

Actors: User

Description:

- The user will be shown available seating.
- The user will select the seat they want.
- Once completed the user will be taken to a new screen to reserve that seat.

Exception Path: If the user select a seat that has been purchased already they will prompted with an error and asked to select another seat.

Alternate Path:

Pre-condition: The View Movie Showtimes use case needs to be completed.

Post-condition: The user will be able to select available seats.

Use Case 8: Reserve Seating

Actors: User

Description:

- The completion of use case view available seating.
- The user will be able to select their seats and reserve them.
- Now the user will be taken to the next screen to purchase tickets.

Exception Path: If the user does not select their seats within the time limit an error will appear taking them back to the view available seating screen.

Alternate Path: The user can cancel the seat selection process and go back to the view available seating screen.

Pre-condition: The View Available Seating use case needs to be completed.

Post-condition: The user seats will be selected and now be taking to the next screen to purchase tickets.

Use Case 9: Purchase Tickets

Actors: User

Description:

- The user will need select the category of ticket they will be purchasing (adult, senior and child).
- Now they will select the number of tickets they will be purchasing for each category.
- Once finished selecting the number of tickets they will be prompted to pay now.
- Use will now enter their credit card detail and hit pay now.

Exception Path: None

Alternate Path: The user can skip the reserve seating screen if they have selected their seats and still available in their cart.

Pre-condition: The Reserve Seating use case needs to be successfully be completed.

Post-condition: The user will be taken to the ticket confirmation screen to print out their ticket or not.

Use Case 10: Receive Confirmation Ticket**Actors:** User**Description:**

- User will see a new screen showing their purchased tickets.
- They will have the option to print their ticket or save it as a pdf and a copy will be sent to their email.
- Once they have picked their option they can close the browser or go to the home screen to view available movies.

Exception Path: None**Alternate Path:** None**Pre-condition:** The purchase tickets use case must be successfully completed.**Post-condition:** The user will receive a confirmation email containing their ticket and receipt.**Use Case 11: View Transaction Logs****Actors:** Admin**Description:**

- The Admin will be prompted for username and password.
- Admin login successfully with username and password.
- Now the admin will be able to view transaction logs.

Exception Path: None**Alternate Path:** None**Pre-condition:** None**Post-condition:** The admin can view the transaction logs if there is a problem on the user side when purchasing the ticket.**Use Case 12: View Support Ticket****Actors:** Admin**Description:**

- The Admin will be prompted for username and password.
- Admin login successfully with username and password.
- Now the admin will be able to view support tickets.

Exception Path: None**Alternate Path:** None**Pre-condition:** The user will need to successfully fill out a support ticket.**Post-condition:** The admin will be able to view the support the user has sent in.

System Requirements

Requirement 1: Use Case 1: Name: Login

Introduction: A user can log in to access their purchased tickets or purchase more tickets.

Rationale: In order to log in, the user must input their valid account information to access the system functions and capabilities.

Input: User ID and password

Requirement Description:

1.1 An interface must be made to allow the user to login by inputting their credentials.

1.2 If a user forgets their user ID or password, a recovery function must be provided.

1.3 The system must detect suspicious hacking attempts to accounts.

Output: Access to the movie show-time finding functions and capabilities.

Test Cases: TBD

Requirement 2: Use Case 2: Name: File Support Ticket

Introduction: A user may file a ticket to send their concern to an admin.

Rationale: The user will navigate to the “File Support Ticket” button and will be prompted to input their concern as a message.

Input: String message

Requirement Description:

1.1 The support ticket page requires the user to enter their name and email.

1.2 The text box where the customer enters their concerns will take up to a maximum of 300 characters.

1.4 The text must be encoded to prevent system hacking.

1.5 The user must receive confirmation by the system that their ticket has been submitted.

1.6 The user must be provided an automated response message that tells the user their ticket will be processed and addressed in due time.

Output: Support ticket confirmation

Test Cases: TBD

Requirement 3: Use Case 3: Name: Search for Theater

Introduction: A user will search for a particular theater with a keyword and the results will display all related theaters relevant to the search performed.

Rationale: The user will access the theater search bar by clicking on its text box and type in their search keywords to find information on the theaters related to the keywords.

Input: String message

Requirement Description:

1.1 The search box will search for theater matches based upon keywords

1.2 There will be a search within “x” radius option where x ranges from 25 miles, 50 miles, and 100 miles.

1.3 The results must display in descending order from closest theaters to farthest

1.4 If no results can be found related to the keywords, a text prompt saying “No matches found” must be displayed.

Output: List of available theaters if matches found

Test Cases: TBD

Requirement 4: Use Case 4: Name: Search for Movie

Introduction: A user will search for a particular movie with a keyword and the results will display all related movies relevant to the search performed.

Rationale: The user will access the movie search bar by clicking on its text box and typing in their search keywords to find information on the theaters related to the keywords.

Input: String message

Requirement Description:

1.1 The search box will search for movie matches based upon keywords

1.2 If no results can be found related to the keywords, a text prompt saying “No matches found” must be displayed.

1.3 There will be a drop down menu which can sort the movies in descending order by categories such as new arrivals, most popular, etc.

Output: List of available movies if matches found.

Test Cases: TBD

Requirement 5: Use Case 5: Name: View Theaters

Introduction: After a search is performed, the user will be shown all theaters and its corresponding showtimes related to the search performed.

Rationale: The user will be provided a list of theaters related to the search and have the option to select one of the theaters.

Input: System request to view

Requirement Description:

1.1 The results will display in descending order from closest theaters to farthest.

1.2 After clicking on the desired theater, a list of movies which are currently being shown are displayed along with their showtimes.

Output: List of available theaters if matches found

Test Cases: TBD

Requirement 6: Use Case 6: Name: View Movie Showtimes

Introduction: After a search is performed, the user will be shown all movies and its corresponding showtimes related to the search performed.

Rationale: The user will be provided a list of movies related to the search and have the option to select a showtime pertaining to that movie.

Input: System request to view

Requirement Description:

1.1 The results will display in descending order from closest to farthest based on each showtime’s theater location.

1.2 Movie showtimes will be displayed in descending order from movies currently being played and soon to be played.

1.3 Image of movie poster pertaining to the movie and the movie name will be displayed for the user.

Output: List of available movies and their showtimes

Test Cases: TBD

Requirement 7: Use Case 7: Name: View Available Seating

Introduction: Once a user selects a movie showtime, they will be redirected to a page that displays available seating and the option to reserve seats.

Rationale: The user gains the option to reserve seats based on the remaining available seating or simply see how many seats are available at that time.

Input: System request to view

Requirement Description:

1.1 After selecting a movie showtime, a new tab will be opened which displays a chart of both available and reserved seating.

1.2 The available seating chart will be color coded to indicate seats that are available and seats that are already reserved.

Output: Chart of available and reserved seating

Test Cases: TBD

Requirement 8: Use Case 8: Name: Reserve Seating

Introduction: A user will select and reserve seating pertaining to how many seats they want to reserve.

Rationale: The user will select the seat or seats they want to reserve on the available seating chart and then click on the “Reserve Seating” button to reserve the seating.

Input: User’s desired seating numbers

Requirement Description:

1.1 The user only has the option to reserve available seating. The user cannot reserve seats that are already reserved.

1.2 The user is required to state how many seats they are reserving.

1.3 If a particular seat is taken by another user by chance of multiple users viewing the available seating at the same time, the user who was later will be prompted with a message that the seat has been already reserved.

1.4 After reserving seating, the user will have an on-screen alert announcing that their reservation was successful.

Output: Seat reservation confirmation

Test Cases: TBD

Requirement 9: Use Case 9: Name: Purchase Tickets

Introduction: The user will purchase tickets based on the number of seats they want to reserve for the movie showtime they selected.

Rationale: After receiving reservation confirmation, the user will be automatically redirected to a page where they are required to purchase tickets. The user must provide valid payment information and must pay the full amount required to reserve all of the seating they wish to reserve.

Input: Customer and payment information

Requirement Description:

1.1 The user is required to purchase the same number of tickets as the number of seats reserved.

1.2 The user is required to enter personal information such as name and email.

1.3 The user is required to enter valid payment information.

1.4 The payment system accepts only card (master and visa).

1.5 After the payment is processed, the user is sent a payment confirmation via email.

1.6 After the payment is processed, the transaction is logged into the database.

Output: Confirmation ticket

Test Cases: TBD

Requirement 10: Use Case 10: Name: Receive Confirmation Ticket

Introduction: The user will receive confirmation and related info to the ticket they purchased.

Rationale: The user is provided all the necessary information related to their purchase including the movie showtime they purchased the ticket for and the seats they reserved.

Input: Client information

Requirement Description:

1.1 The confirmation ticket contains a receipt for the processed payment.

1.2 The confirmation ticket must display the time and date of ticket purchase.

1.3 The confirmation ticket displays the user's ticket code(s) and seat assignments.

1.4 The confirmation ticket displays the chosen theater's address and movie showtime.

Output: Confirmation Ticket

Test Cases: TBD

Requirement 11: Use Case 11: Name: View Transaction Logs

Introduction: An admin is able to view the logs related to ticket purchases.

Rationale: An admin logs into their account and has the option to view information related to purchases made through GuacAgain.com.

Input: System request to view

Requirement Description:

1.1 The transaction log must display the date, time, and amount processed.

1.2 There must be an option to sort the list by date and time in descending order.

1.3 There will be a link which opens a new tab that displays simple statistics such as total revenue and number of tickets sold.

Output: Transaction logs

Test Cases: TBD

Requirement 12: Use Case 12: Name: View Support Tickets

Introduction: An admin is able to view all support tickets submitted and reply accordingly.

Rationale: An admin logs into their account and has the option to view all support tickets and also send a reply to the user who submitted the ticket.

Input: System request to view

Requirement Description:

1.1 The admin must be able to view all support tickets that were submitted.

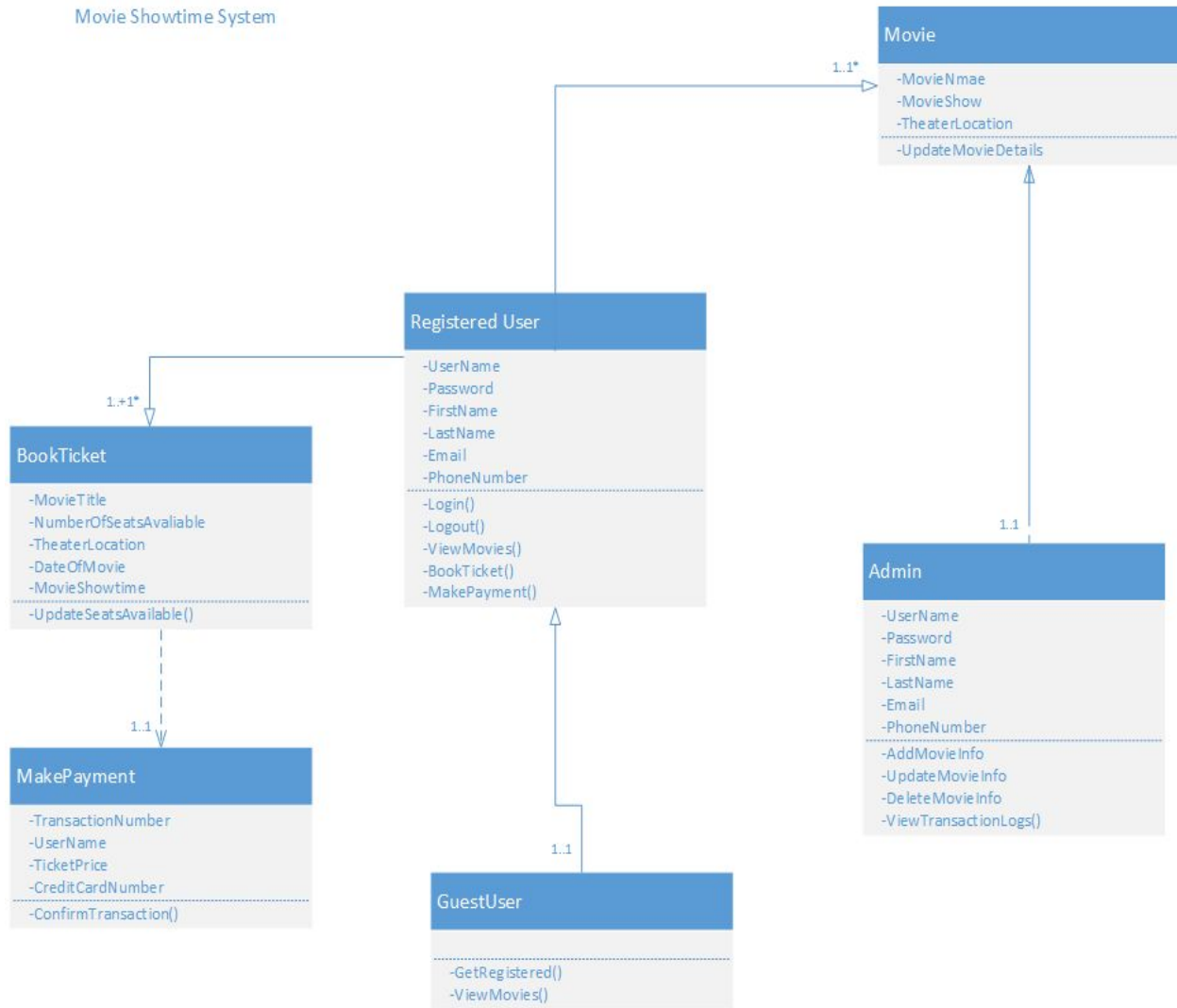
1.2 Each support ticket must show the time and date that it was submitted.

1.3 The support ticket must be formatted to be easily eligible.

Output: Support tickets

Test Cases: TBD

Movie Showtime System

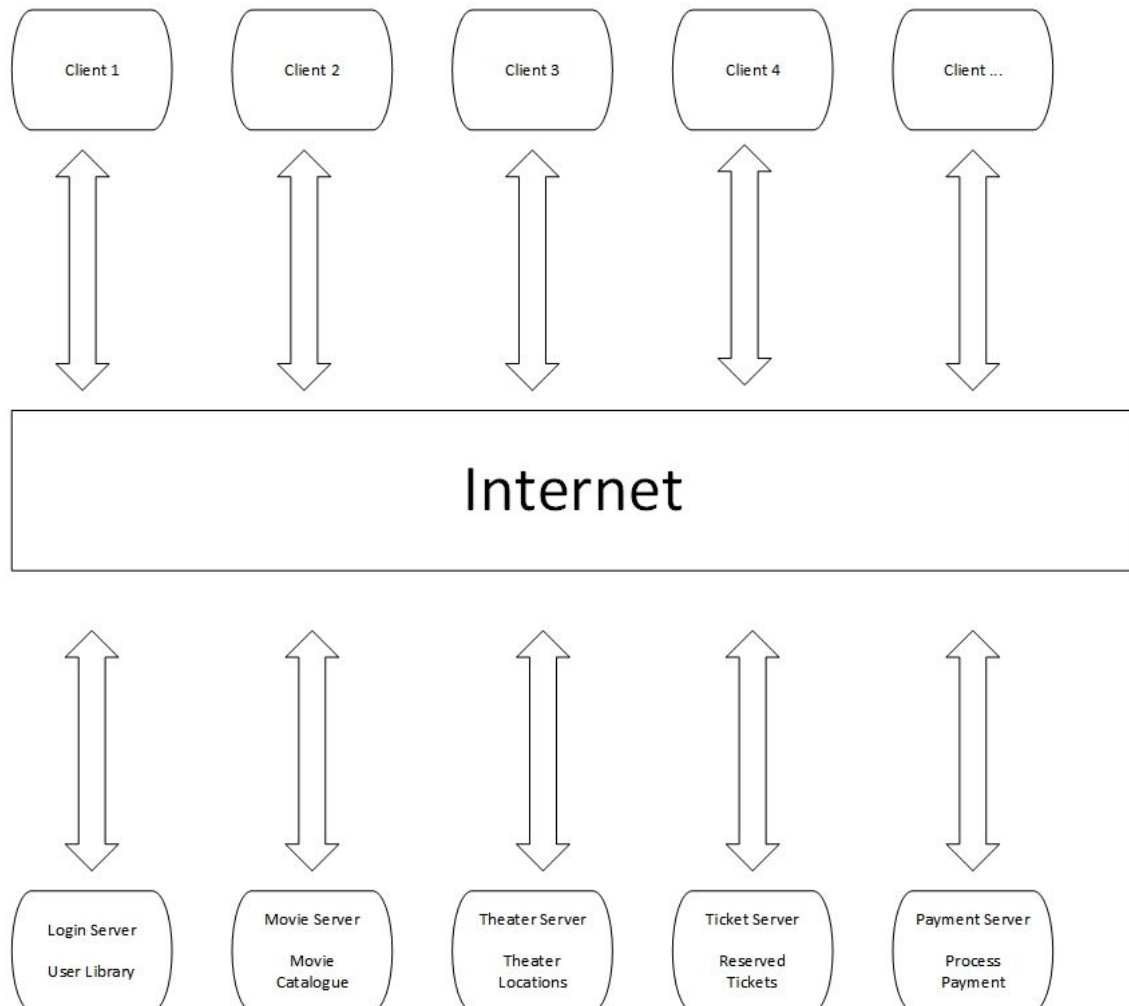


3. System modeling and analysis

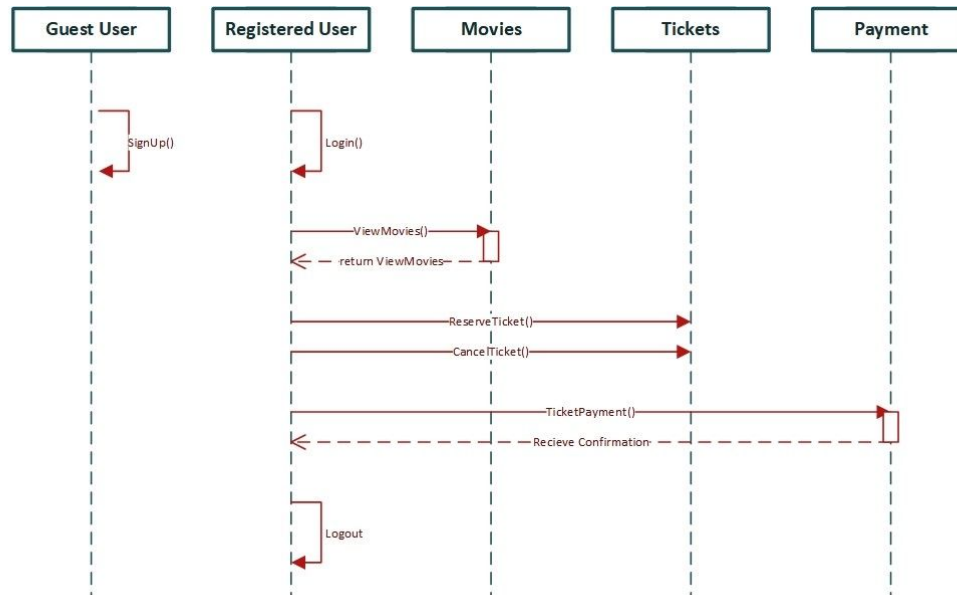
1. Architecture Modeling

Client- Server Architecture

The Clients could be admins, registered user or guest user



2. Behavioral Modeling



4. Implementation

1. Go to <https://c9.io/login>
2. User name: chad_smith95@yahoo.com
Password: blockbuster1
3. To start the website run the command “ npm install “ in the terminal.
4. Then run “ node app.js “ in the terminal
5. Select preview in the navigation bar then press preview running application.
6. Copy <https://block-buster-chad95.c9users.io> in your address bar browse live site.

5. Testing

Test Cases:

Test Case ID: 1.1

Description: Attempt logging in with valid account information.

Test Inputs: Valid username and password corresponding to account.

Expected Results: Successful login and redirect to homepage.

Dependencies: None.

Initialization: The user's account must have already been created.

Test Steps: 1. Type in user's username and password in corresponding boxes.

2. Attempt login.

3. Verify login was successful

Test Case ID: 1.2

Description: Attempt logging in with invalid account information.

Test Inputs: Invalid username and password corresponding to account.

Expected Results: Unsuccessful login. User is told to try again or go to forgot password.

Dependencies: None.

Initialization: Database with correct user information to check against

Test Steps: 1. Type in invalid user's username and password in corresponding boxes.

2. Attempt login.

3. Verify login was unsuccessful, and the correct exception path is taken.

Test Case ID: 2.1

Description: Attempt submitting a support ticket.

Test Inputs: Support ticket title and paragraph for support ticket.

Expected Results: Ticket is submitted and user is redirected to a ticket confirmation page.

Dependencies: Test Case Id 1.1.

Initialization: User must be logged in and ticket must contain content consisting of strings.

Test Steps: 1. Press "Submit Support Ticket" and navigate to ticket submission page.

2. Type in a title and body for the support ticket.

3. Attempt ticket submission.

Test Case ID: 3.1

Description: Attempt a search for nearby theaters.

Test Inputs: Valid zip code and desired radius

Expected Results: The search will be completed and the user will be shown a list of theaters closest to the user zip code.

Dependencies: None.

Initialization: Databases of valid zip codes, and movie theaters.

Test Steps: 1. Type in zip code

2. Select the radius to search.

3. Select movie theater.

Test Case ID: 3.2

Description: Attempt a unsuccessful search for theater.

Test Inputs: Invalid zip code and desired radius

Expected Results: The search will be unsuccessful and the user won't see any theaters. User will be asked to try again.

Dependencies: None.

Initialization: User must type in at least one search keyword to perform a search.

Test Steps: 1. Type in invalid zip code (zip code doesn't exist, or user didn't type in 5 digit number.)

2. Select the radius to search

3. No theaters show up, User is asked to increase radius or change zip code and try again.

Test Case ID: 4.1

Description: Attempt a search for movies.

Test Inputs: Valid movie keyword or keywords.

Expected Results: The search will be completed and the user will be shown a list of movies closest to the inputted keyword.

Dependencies: None.

Initialization: Databases of movie names and all movie related information.

Test Steps: 1. Type in movie keyword.

2. Select movie.

Test Case ID: 4.2

Description: Attempt a search for movies that are currently not in theaters.

Test Inputs: Valid old movie keyword or keywords.

Expected Results: The search will be completed and the user will be shown a list of movies closest to the inputted keyword, if the movie selected isn't in theaters the user will be prompted to search for another movie.

Dependencies: None.

Initialization: Databases of movie names and all movie related information.

Test Steps: 1. Type in movie keyword.

2. Select movie.

3. Movie isn't available in theaters so user is prompted to search for a different movie.

Test Case ID: 5.1

Description: Attempt to view a theater.

Test Inputs: Click on a theater.

Expected Results: Theater information will be shown to the user including the movies being shown and the showtimes available.

Dependencies: None.

Initialization: Movie theater information.

Test Steps: 1. Click on a theater.

2. Observe all available movies and their showtimes.

Test Case ID: 6.1

Description: Attempt to view a movie.

Test Inputs: Click on a movie.

Expected Results: Movie name and showtimes will be shown to the user.

Dependencies: None.

Initialization: Movie showtime data.

Test Steps: 1. Click on a movie.

2. Observe all showtimes available for that movie.

Test Case ID: 7.1

Description: Attempt to view available seating.

Test Inputs: Click on a showtime.

Expected Results: The user will be shown all available seating for that movie time.

Dependencies: Test Case ID 6.1.

Initialization: Available seating data.

Test Steps: 1. Click on a showtime.

2. Observe available seating.

3. Reserve designated seating.

Test Case ID: 7.2

Description: Attempt to view available seating and selecting a reserved seating.

Test Inputs: Click on a showtime.

Expected Results: The user will be prompted with an error stating that the seat they have chosen has already been reserved. User will then be asked to select another seat.

Dependencies: Test Case ID 6.1.

Initialization: Available seating data.

Test Steps: 1. Click on a showtime.

2. Observe available seating

3. Select a seat that has already been reserved by another user.

4. Error, user is prompted to select a seat that is not already reserved.

Test Case ID: 8.1

Description: Attempt to reserve seating.

Test Inputs: Click on desired seating from available seating spots and click on reserve.

Expected Results: The seating will be held and the user will be given the option to purchase tickets to finalize the reservation.

Dependencies: None.

Initialization: Seating must be available.

Test Steps: 1. Click on the desired seatings.

2. Click on reserve.

Test Case ID: 9.1

Description: Attempt to purchase ticket.

Test Inputs: Valid payment method and information.

Expected Results: The ticket will be purchased and the seating reservation will be finalized.

Dependencies: Test Case ID 8.1

Initialization: None.

Test Steps: 1. Input valid payment information.

2. Click on purchase tickets.

3. Notifies user that ticket has been purchased and they will receive a confirmation in their email.

Test Case ID: 9.2

Description: Attempt to purchase ticket with invalid card information.

Test Inputs: Invalid payment method and information.

Expected Results: User will be notified card declined or card information isn't correct.

Dependencies: Test Case ID 8.1

Initialization: None.

Test Steps: 1. Input invalid payment information.

2. Click on purchase tickets.

3. User is notified that the payment has failed, try another card or review the card information that was inputted.

Test Case ID: 10.1

Description: Attempt to receive ticket confirmation.

Test Inputs: None.

Expected Results: User will be provided ticket confirmation after purchasing a ticket.

Dependencies: Test Case ID 9.1

Initialization: Confirmation data.

Test Steps: 1. Purchase ticket.

2. Observe ticket confirmation being provided to user.

Test Case ID: 11.1

Description: Attempt to view transaction logs.

Test Inputs: Click on “View Transaction Logs”.

Expected Results: Transaction logs will be shown to the user.

Dependencies: None.

Initialization: Transaction data.

Test Steps: 1. Click on view transaction logs.

2. Observe transaction logs provided.

Test Case ID: 12.1

Description: Attempt to view support ticket.

Test Inputs: Click on “View Support Tickets”.

Expected Results: Submitted support tickets will be shown to the user.

Dependencies: Test Case ID 2.1

Initialization: Support ticket data.

Test Steps: 1. Click on view support tickets.

2. Observe submitted support tickets.

3. Alternatively observe the “No tickets have been submitted.” message if no support tickets have been submitted.

6. Communication and Collaboration

