

Movie Show-Time Finder

Movietickets.com

Refactored-Guacamole

Group 8

Royan Hanson

Micheal Smith

Chad Smith

Yash Shah

Paul Lee

Work Breakdown Structure

Assignee Name	Email	Task	Duration (Hours)	Dependency	Due date	Evaluation
Micheal Smith	msmith303@student.gsu.edu	Task 5: Implementation	20 Hours	None	4/13/19	100%
Paul Lee	plee30@student.gsu.edu	Task 6: Testing	6 hours	None	4/13/19	100%
Royan Hanson	rhanson3@student.gsu.edu	Task 1: Planning/Peer Evaluation Task 2: Communication and Collaboration Task 6: Testing	3 hours	None	4/13/19	100%
Yash Shah (Team Coordinator)	yshah4@student.gsu.edu	Task 4: System Modeling	4 Hours	None	4/13/19	100%
Chad Smith	csmith324@student.gsu.edu	Task 5: Implementation	20 hours	None	4/13/19	100%

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: Submit review

Actors: User

Description:

- The user navigates to movie page .
- The user will enter their review details and click submit..
- 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 review confirmation..

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 review will be submitted and displayed for data.

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 3: 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 4: 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 4: 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 5: 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 6: 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 7: 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 the 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 8: View Transaction History

Actors: Admin

Description:

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

Exception Path: None

Alternate Path: None

Pre-condition: None

Post-condition: The user can view the transaction logs if there is a problem on the system 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: Attempt logging in with valid account information.

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: Attempt submitting a support ticket.

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: Attempt a search for nearby theaters.

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: Attempt a search for movies.

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: Attempt to view a theater.

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: Attempt to view a movie.

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: Attempt to view available seating.

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: Attempt to reserve seating.

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: Attempt to purchase ticket.

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: Attempt to receive ticket confirmation.

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: Attempt to view transaction logs.

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: Attempt to view support ticket.

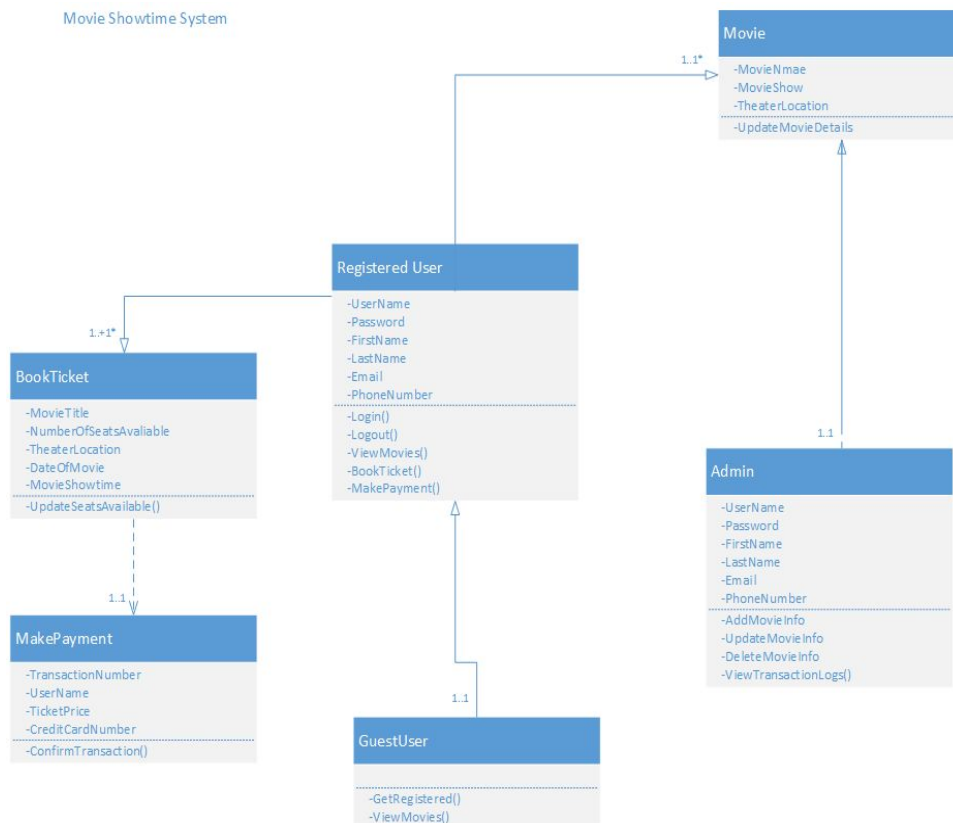
System Modeling (Analysis)

1. Architectural Modeling 4+1

- Movie show time finder Scenarios
 - Search for movies
 - Find a local theater
 - Reserve tickets
 - Purchase tickets

Logical View

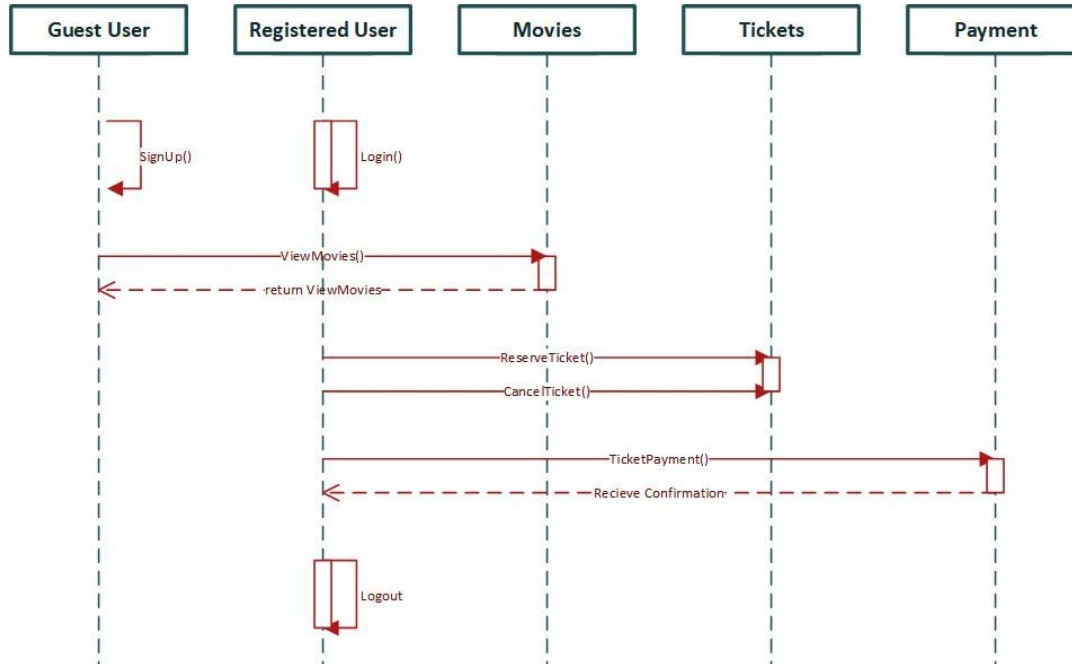
- Decompose the system structure into software components and connectors
- Map functionality (use cases) onto the components
 - Concern: Functionality
 - Target Audience: Developers and Users



Process View

- Model the dynamic aspects of the architecture and the behavior its parts
 - Concern: Functionality, Performance

- Target Audience: Developers



Development View

- Static organization of the software code artifacts
- Mapping between the elements in the logical view and the code artifacts
 - Concern: Reuse, Portaility, Build
 - Target Audience: Developers

Movie-Composite

Field Name	Field Description
MovieName	The movie title (pk)
MovieShow	Movie's show time
TheaterLocation	Theater's location

RegisteredUser-Simple



Field Name	Field Description
UserName	User name login (pk)
Password	Password for login
FirstName	User's First name
LastName	User's Last name
Email	User's email address
PhoneNumber	User's phone number

Admin-Simple

Field Name	Field Description
UserName	Admin name login (pk)
Password	Password for Admin's login
FirstName	Admin's First name
LastName	Admin's Last name
Email	Admin's email address
Phonenumber	Admin's phone number

BookTicket-Multivalued

Field Name	Field Description
MovieTitle	Title of Movie(pk)
NumberOfSeatsAvailable	Number of the seats that haven't been booked
TheaterLocation	Location of theater(fk)
DateOfMovie	Date of the ticket for the movie
MovieShowtime	Time of movie

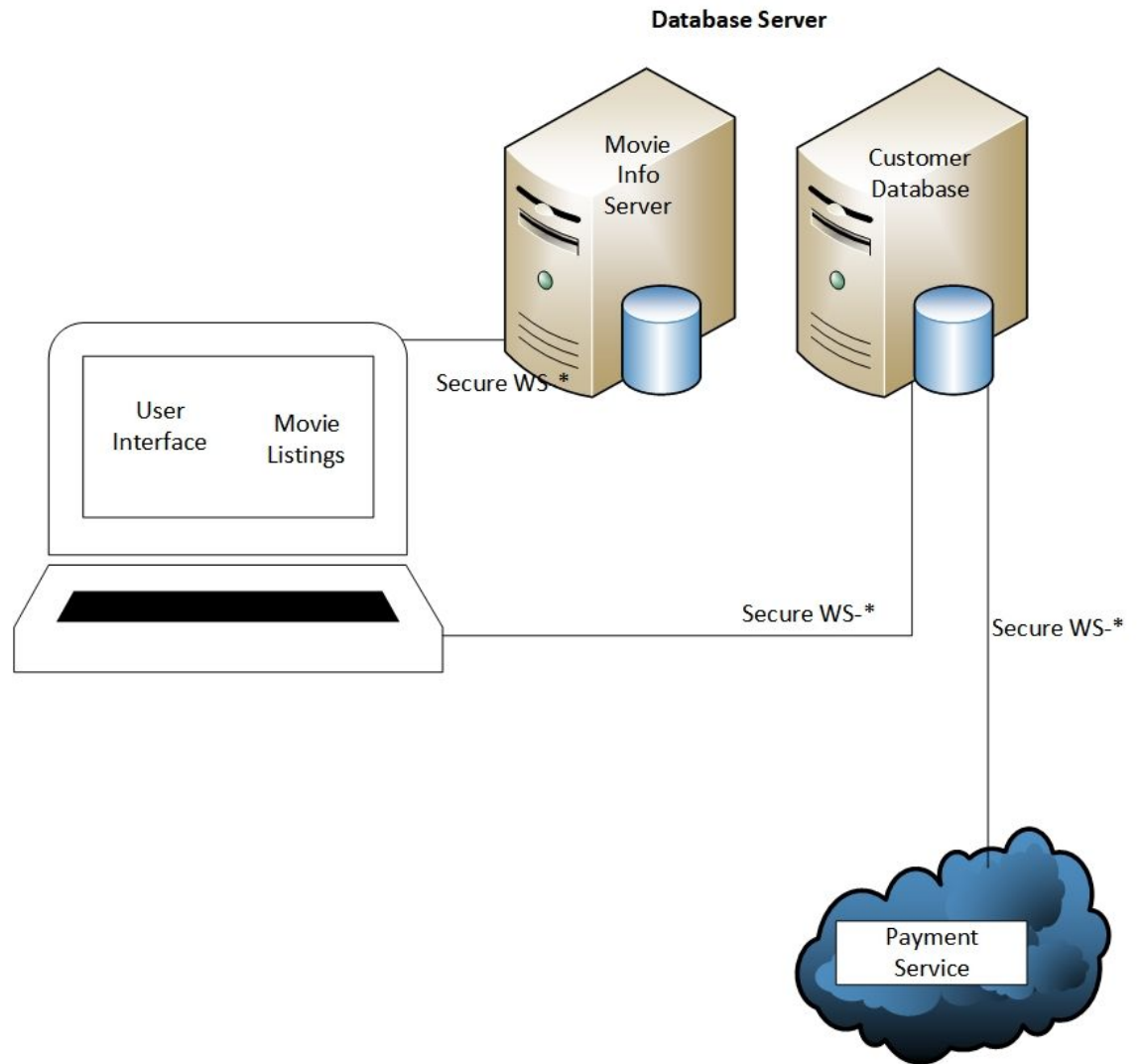
MakePayment-Composite

Field Name	Field Description
TransactionNumber	Transaction number(pk)
UserName	Customer's Name(fk)
TicketPrice	Price of Ticket
CreditCardNumber	Customer's card number

Movie show-time finder will use MySQL as the database management system, since the databases for the movies and the times are being provided from the open sources that already available.

Physical View

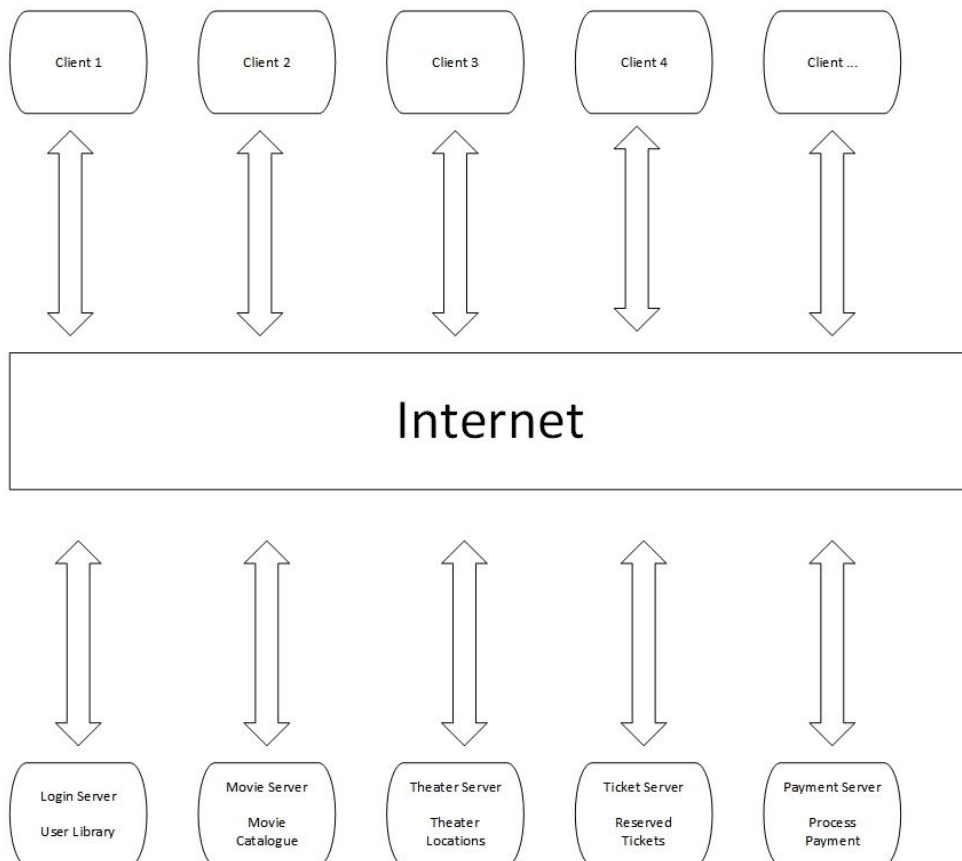
- Hardware environment where the software will be deployed
- Mapping between logical and physical entities
 - Concern: Quality attributes
 - Target Audience: Operations



Architecture Model (i+1 Model)

Client- Server Architecture

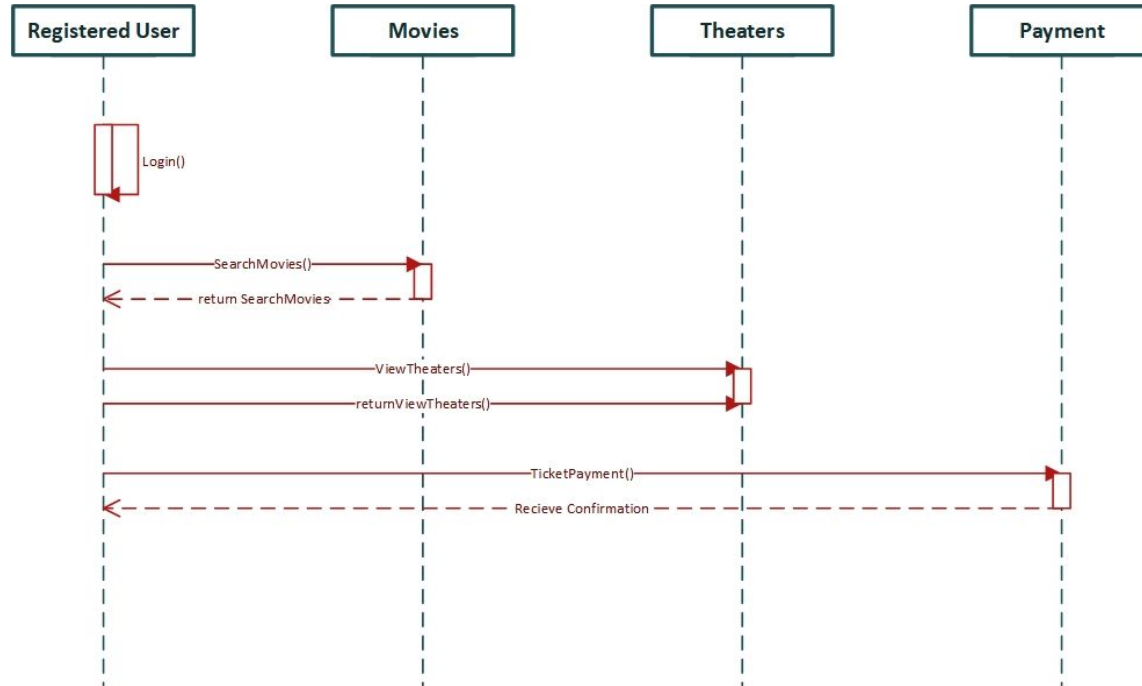
The Clients could be admins, registered user or guest user

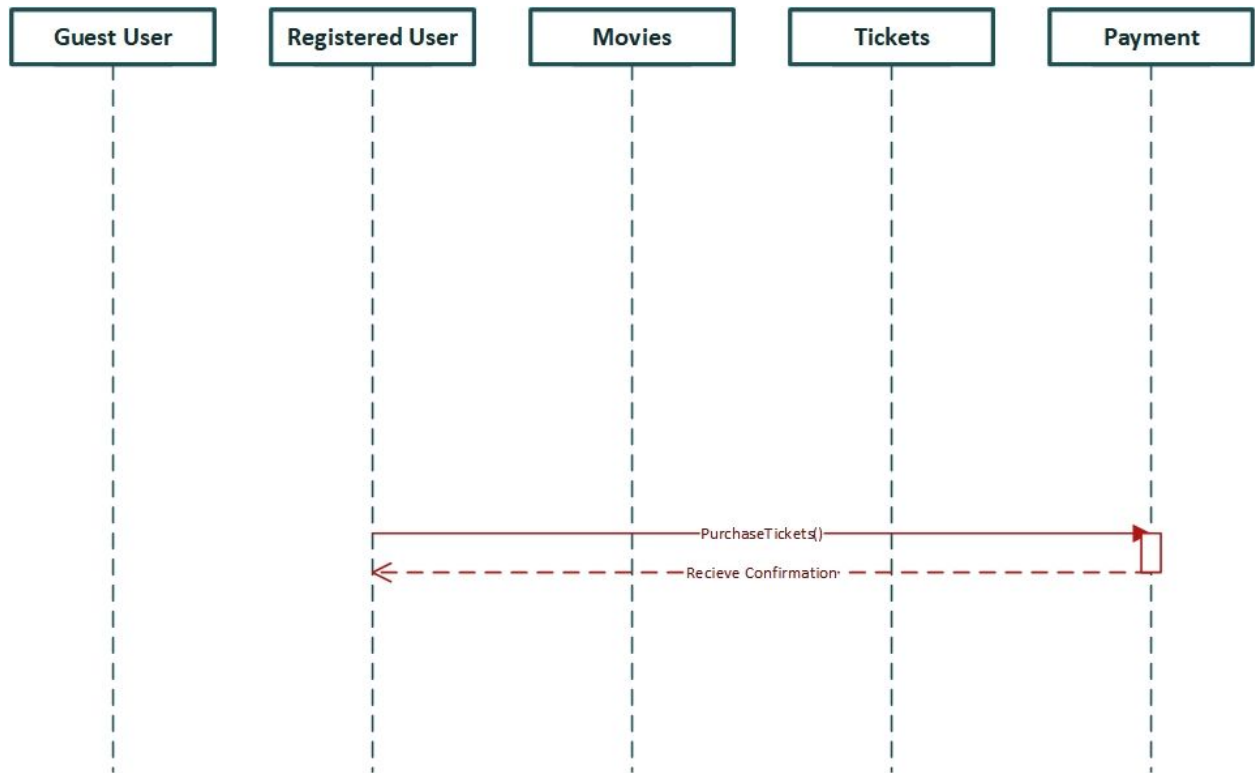


Behavioral Modeling

Use Case 4: SearchMovies()

Use Case 5: ViewTheaters()





Use Case 11: Purchase Tickets

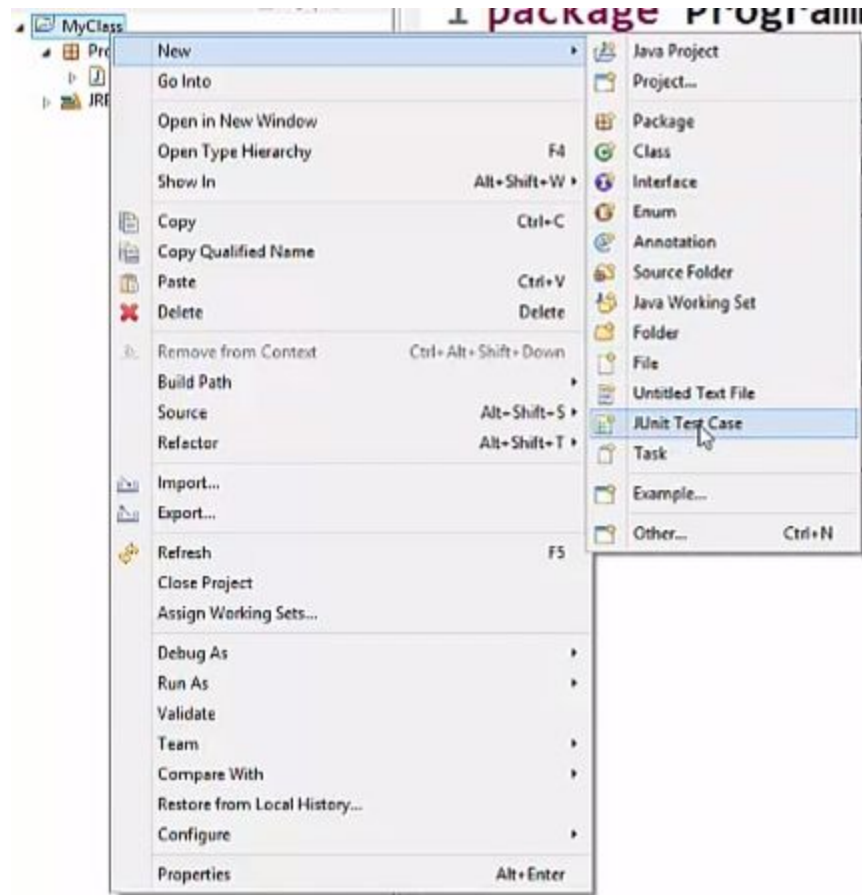
Use Case 10: Receive Confirmation Ticket

Implementation

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

Testing

a. Steps to install framework



New JUnit Test Case

JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

☐ New JUnit 3 test ☐ New JUnit 4 test ☒ New JUnit Jupiter test

Source folder:

Package:

Name:

Superclass:

Which method stubs would you like to create?

☐ setUpBeforeClass() ☐ tearDownAfterClass()
☐ setUp() ☐ tearDown()
☐ constructor

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Class under test:

New JUnit Test Case

JUnit 4 is not on the build path. Do you want to add it?

☐ Not now
☐ Open the build path property page
☒ Perform the following action:

☒ Add JUnit 4 library to the build path

b.

Test ID	1.1 Attempt logging in with valid credentials.
Purpose of Test	Check the response of the system when valid email and password is entered.
Test Environment	Test should be conducted in an environment that isolates username and password verification.
Test Steps	1. Enter valid username/email. 2. Enter valid password.
Test Input	Username() Password()
Expected Result	Login will be successful
Likely Problems/Bugs Revealed	System does not recognize username or password even though it is valid.

Test ID	2.1 Attempt to file a support ticket.
Purpose of Test	Check to see whether the function of submitting a support ticket is working properly. i.e. logging the ticket, server receiving the ticket.
Test Environment	Jasmine Javascript testing environment based on Windows OS. Assumption of server capable of receiving ticket.
Test Steps	1. Click on the option to submit ticket. 2. Type text into textbox and press submit. 3. Verify the ticket has gone through.

Test Input	Any text combination to place inside text box.
Expected Result	System will accept submitted ticket and save the ticket to view later.
Likely Problems/Bugs Revealed	Ticket may not be accepted by system or duplicate tickets may be stored instead of just one ticket.

Test ID	3.1 Attempt a search for nearby theaters.
Purpose of Test	To receive accurate search results that are relevant to your theater related keyword search.
Test Environment	Jasmine Javascript testing environment based on Windows OS. Assumption of theater data existing in servers already.
Test Steps	1. In the search textbox, input a relevant theater related keyword. 2. Search results and check if results are accurate.
Test Input	Theater related keyword for search textbox.
Expected Result	System will return information related to theaters relevant to keywords.
Likely Problems/Bugs Revealed	The search results may return random information that are unrelated to the designated keywords.

Test ID	4.1 Attempt searching for movie.
---------	----------------------------------

Purpose of Test	See whether a movie search provides proper search results.
Test Environment	Test should be conducted and tested between database and keywords to compare string results.
Test Steps	1. Enter movie related keyword. 2. Search
Test Input	Valid keyword for movie.
Expected Result	System will successfully find and display all movies related to keyword used for searching.
Likely Problems/Bugs Revealed	The system may display incorrect or irrelevant movies.

Test ID	5.1 Attempt to view a theater.
Purpose of Test	To check to see whether the function of viewing a theater is working as intended.
Test Environment	Jasmine Javascript testing environment based on Windows OS. Assumption of theater data existing in servers already.
Test Steps	1. In results page, click on a theater that has showed up due to a search performed.
Test Input	A click performed by user on a theater.
Expected Result	System will return information about the theater selected.

Likely Problems/Bugs Revealed	Incorrect theater info may be displayed to the user or the wrong theater info may appear.
-------------------------------	---

Test ID	6.1 Attempt to view a movie.
Purpose of Test	Check to see whether the function of viewing a movie is working as intended.
Test Environment	Test should be conducted in an environment that isolates the file retrieval of particular movies.
Test Steps	1. Click on a movie found through search result.
Test Input	A user click.
Expected Result	System will show information on movie and showtimes to the user.
Likely Problems/Bugs Revealed	A link may not work or the link will show a different movie instead.

Test ID	7.1 Attempt to view available seating.
Purpose of Test	To verify that seating reservation system is properly displaying the correct available seating.
Test Environment	Jasmine Javascript testing environment based on Windows OS. Assumption of seating data existing in servers already.
Test Steps	1. Select a showtime for any movie.
Test Input	A click performed by user on a movie showtime.
Expected Result	System will return <i>correct</i> information regarding available seating arrangements.
Likely Problems/Bugs Revealed	The shown available seating may not be accurate showings.

Test ID	8.1 Attempt to reserve seating.
Purpose of Test	To check to see whether the function of reserving seating is properly working. Should block any attempts of duplicate reservations.
Test Environment	Jasmine Javascript testing environment based on Windows OS. Assumption of available seating function working properly.
Test Steps	1. Select an available seat from the seating selection page. 2. Press submit and verify reservation.
Test Input	An available seat selection and a click on reserve button.

Expected Result	System will reserve seat as intended and notify if already taken.
Likely Problems/Bugs Revealed	Reserving a seat that has already been taken may cause the original seatholder to be overwritten.

Test ID	9.1 Attempt to purchase a ticket.
Purpose of Test	To verify that tickets are properly distributed and paid for with the purchase ticket function.
Test Environment	Jasmine Javascript testing environment based on Windows OS. Assumption of payment system already set up.
Test Steps	1. After reserving a seat, enter payment info. 2. Attempt ticket payment.
Test Input	A valid payment information set.
Expected Result	System will verify the payment info is valid and will issue ticket.
Likely Problems/Bugs Revealed	Tickets may be purchasable with incorrect payment information.

Test ID	10.1 Attempt to receive ticket confirmation.
Purpose of Test	To verify that ticket confirmation is properly issued after purchasing a ticket.

Test Environment	Jasmine Javascript testing environment based on Windows OS. Assumption of ticket properly being purchased.
Test Steps	1. Purchase ticket. 2. View whether ticket confirmation is received.
Test Input	Purchasing of ticket.
Expected Result	System will send out the ticket confirmation to the user.
Likely Problems/Bugs Revealed	Ticket confirmation may not be sent or sent to the wrong receiver.

Test ID	11.1 Attempt to view a transaction logs.
Purpose of Test	To check to see whether the proper transaction logs open up when attempted to be viewed.
Test Environment	Jasmine Javascript testing environment based on Windows OS. Assumption of transactions existing.
Test Steps	1. Click on button to view transaction logs.
Test Input	A click performed by user to view transaction logs.
Expected Result	System will return information regarding transactions and display it for the user.

Likely Problems/Bugs Revealed	Transaction logs may not properly display or they may display incorrectly.
-------------------------------	--

Test ID	12.1 Attempt to view support ticket.
Purpose of Test	To check to see whether the function of viewing a support ticket is working as intended.
Test Environment	Jasmine Javascript testing environment based on Windows OS. Assumption of submitted tickets existing in servers already.
Test Steps	1. Press on view support ticket to move to ticket page.
Test Input	A click performed by user on view support ticket button.
Expected Result	System will return information regarding all submitted support tickets.
Likely Problems/Bugs Revealed	Submitted support tickets may not be appearing properly to the user.

Bugs found to report:	N/A	N/A	N/A
-----------------------	-----	-----	-----

c. 1. Attempting Login

4 specs, 0 failures, randomized with seed 58158

```
Player
  #resume
    • should throw an exception if id and pw do not match
    •
    •
    • should be able to login
```

Ran 1 of 4 specs - run all

1 spec, 0 failures, randomized with seed 58158

```
Player
  • should be able to login
```

1 spec, 0 failures, randomized with seed 58158

```
Player
  #resume
    • should throw an exception if id and pw do not match
```



```

describe("Login", function() {
  var id;
  var pw;
  beforeEach(function() {
    id = new id();
    pw = new pw();
  });
  it("should be able to login", function() {
    id.match(pw);
    expect(id.currentlymatched).toEqual(pw);

    //demonstrates use of custom matcher
    expect(id).toBeMatching(pw);
  });
  describe("#resume", function() {
    it("should throw an exception if id and pw do not match", function() {
      id.match(pw);
      expect(function() {
        id.resume();
      }).toThrowError("should throw an exception if id and pw do not match");
    });
  });
});

```

2. Search for movie

4 specs, 0 failures, randomized with seed 21893

```
Player
  • should be able search for movie

when keywords are entered
  • should display movie related to keywords
  • should display movies, index, and movie

#resume
  • should throw an exception if invalid keywords inputted
```

1 spec, 0 failures, randomized with seed 08328

```
Player
  • should be able search for movie
```

2 specs, 0 failures, randomized with seed 77434

```
Player

when keywords are entered
  • should display movies, index, and movie
  • should display movie related to keywords
```

1 spec, 0 failures, randomized with seed 18533

```
Player

when keywords are entered
  • should display movies, index, and movie
```

1 spec, 0 failures, randomized with seed 25354

```
Player

when keywords are entered
  • should display movie related to keywords
```

1 spec, 0 failures, randomized with seed 00867

```
Player
  • should be able search for movie
```

```

describe("Movie", function() {
  var Movie;
  var index;

  beforeEach(function() {
    Movie = new Movie();
    index = new index();
  });

  it("should be able search for movie", function() {
    Movie.display(index);
    expect(Movie.currentlyDisdisplayingindex).toEqual(index);

    //demonstrates use of custom matcher
    expect(Movie).toBeShowing(index);
  });

  describe("when keywords are entered", function() {
    beforeEach(function() {
      Movie.display(index);
      Movie.pause();
    });

    it("should disdisplay movie related to keywords", function() {
      expect(Movie.isDisplaying).toBeFalsy();

      // demonstrates use of 'not' with a custom matcher
      expect(Movie).not.toBeShowing(index);
    });

    it("should disdisplay movies, index, and movie", function() {
      Movie.resume();
      expect(Movie.isDisplaying).toBeTruthy();
      expect(Movie.currentlyDisdisplayingindex).toEqual(index);
    });
  });
});

```

```
//demonstrates use of expected exceptions
describe("#resume", function() {
  it("should throw an exception if invalid keywords inputted", function() {
    Movie.display(index);

    expect(function() {
      Movie.resume();
    }).toThrowError("should throw an exception if invalid keywords inputted");
  });
});
```

3. Displaying Movie

4 specs, 0 failures, randomized with seed 46731

```
Player
  • should display movie info

#resume
  • should throw an exception if movie details are unavailable

when movie is clicked on
  • should display movie info
  • should display title, description
```

1 spec, 0 failures, randomized with seed 76235

```
Player
  • should display movie info
```

1 spec, 0 failures, randomized with seed 26499

```
Player
  #resume
    • should throw an exception if movie details are unavailable
```

1 spec, 0 failures, randomized with seed 30451

```
Player
  when movie is clicked on
    • should display title, description
```

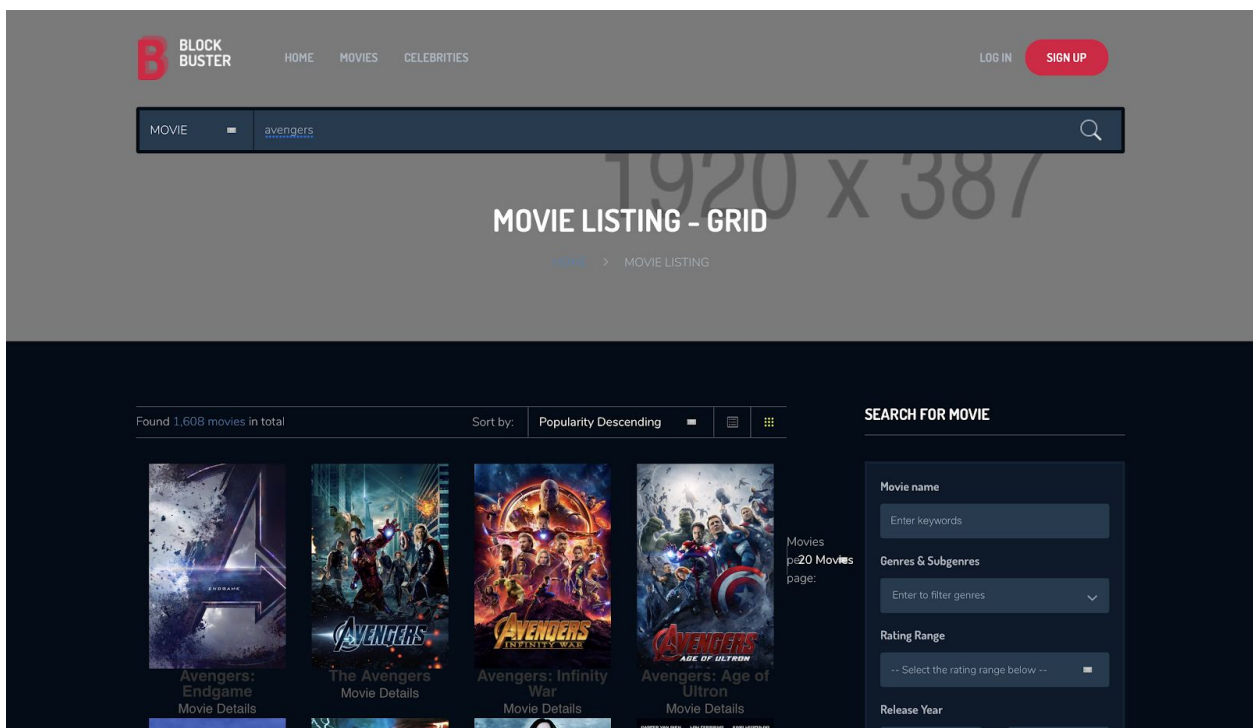
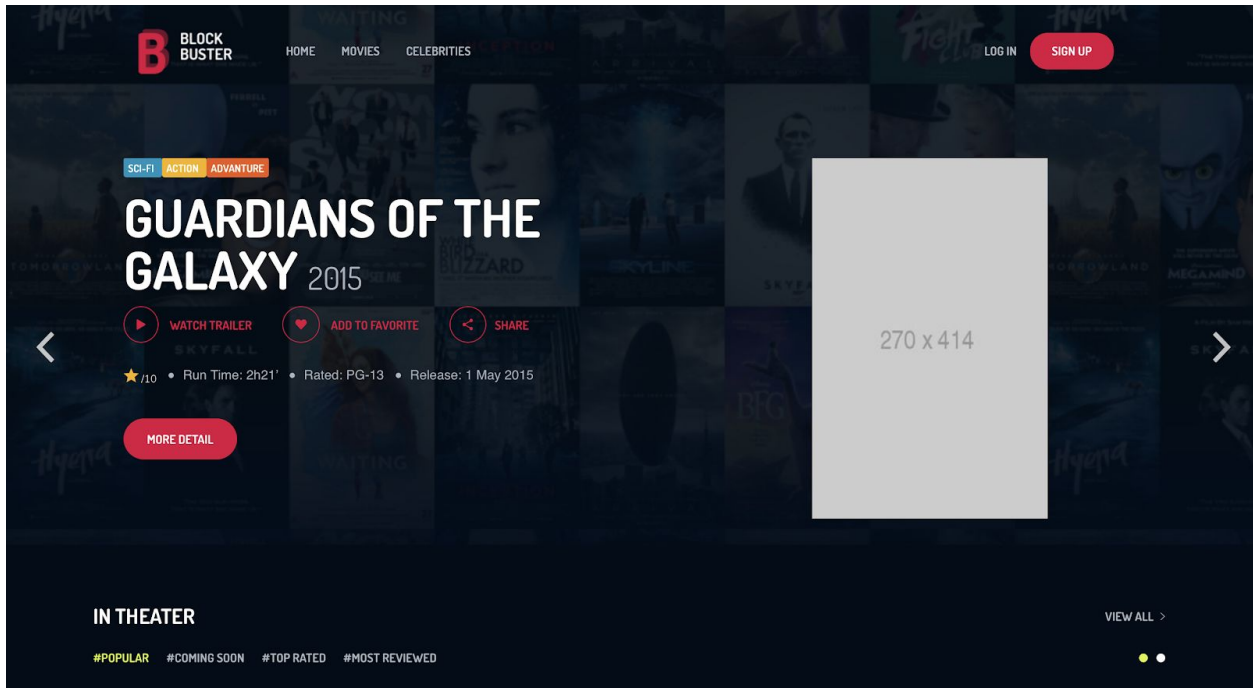
1 spec, 0 failures, randomized with seed 42064

Player
when movie is clicked on
• should display movie info

```
describe("Title", function() {  
  var Title;  
  var Description;  
  
  beforeEach(function() {  
    Title = new Title();  
    Description = new Description();  
  });  
  
  it("should display movie info", function() {  
    Title.display(Description);  
    expect(Title.currentlyDisplayingDescription).toEqual(Description);  
  
    //demonstrates use of custom matcher  
    expect(Title).toShowDetailed(Description);  
  });  
  
  describe("when movie is clicked on", function() {  
    beforeEach(function() {  
      Title.display(Description);  
      Title.pause();  
    });  
  
    it("should display movie info", function() {  
      expect(Title.isDisplaying).toBeFalsy();  
  
      // demonstrates use of 'not' with a custom matcher  
      expect(Title).not.toShowDetailed(Description);  
    });  
  
    it("should display title, description", function() {  
      Title.resume();  
      expect(Title.isDisplaying).toBeTruthy();  
      expect(Title.currentlyDisplayingDescription).toEqual(Description);  
    });  
  });  
});
```

```
describe("#resume", function() {  
  it("should throw an exception if movie details are unavailable", function() {  
    Title.display(Description);  
  
    expect(function() {  
      Title.resume();  
    }).toThrowError("should throw an exception if movie details are unavailable");  
  });  
});
```


Screenshot



gsuproject2019 / CSC-SWE-REFACTOREDGUACAOLE

Watch1

Star0

Fork0

<> Code

Issues0

Pull requests0

Projects2

Wiki

Insights

Settings

CSC-SWE-REFACTOREDGUACAOLE

Updated 4 days ago

Filter cards

Add cards

Fullscreen

Menu

5 Group Members

☑ Royan Hanson (Team Coordinator)

Added by gsuproject2019

☑ Paul Lee

Added by gsuproject2019

☑ Micheal Smith

Added by gsuproject2019

☑ Yash Shah

Added by gsuproject2019

☑ Chad Smith

Added by gsuproject2019

0 TO DO

0 In progress

21 Done

☑ A4_TASK 1: Planning Scheduling and Peer Evaluation

Added by gsuproject2019

☑ A4_TASK 2 : Communication and Collaboration

Added by gsuproject2019

☑ A4_TASK3: Revise and Refine your System

Added by gsuproject2019

☑ A4_TASK4: System Modeling (Analysis)

Added by gsuproject2019

☑ A4_TASK5: Implementation

Added by gsuproject2019

☑ A4_TASK6 - Testing

Added by gsuproject2019

39