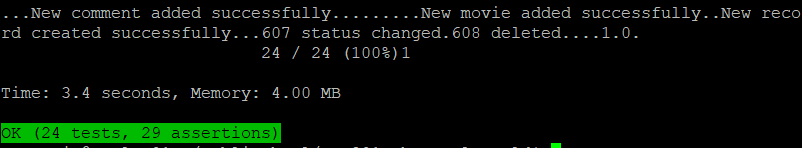
**The Reel World Testing** Results

## Unit Tests

**Test:** MovieManager, UserManager, RequestManager, and CommentManager unit tests

**Result:** All unit tests passed

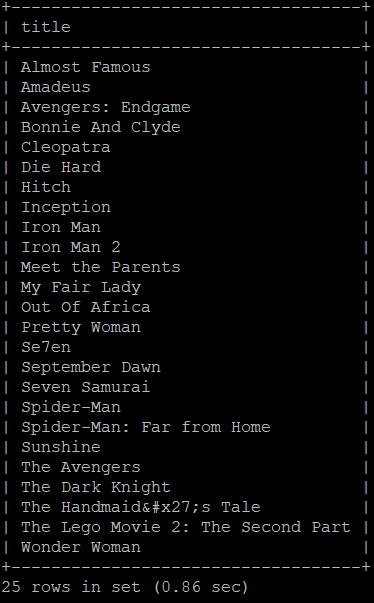


## Integration Tests

**Test 1:** The same movies appear in the UI on the Home page as the results for querying the database for all movies

**SQL Query:** SELECT title FROM Movies WHERE isDeleted = 0 ORDER BY title;

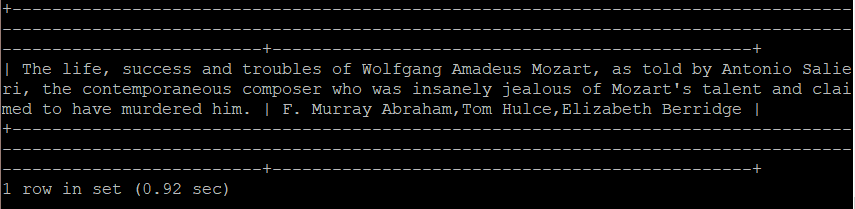
**Result:** The movies viewable on the interface match the query results:

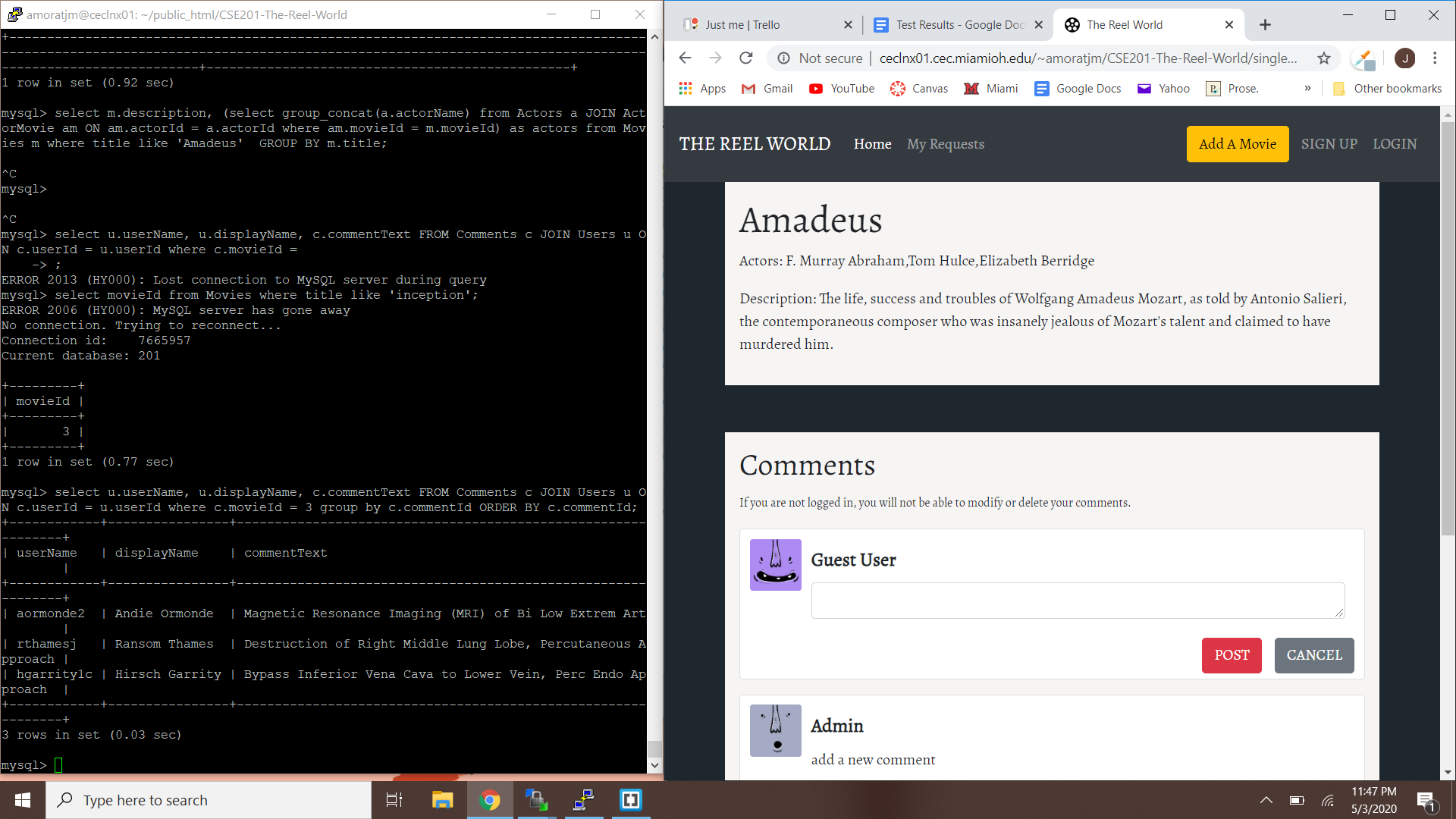
****

**Test 2:** The “Amadeus” movie page UI displays the same information as querying the database for the actors and description for “Amadeus”

**SQL Query:** SELECT m.description, (SELECT GROUP\_CONCAT(a.actorName) FROM Actors a JOIN ActorMovie am ON am.actorId = a.actorID WHERE am.movieId = m.movieId) AS actors FROM Movies m WHERE title LIKE ‘Amadeus’ GROUP BY m.title;

**Result:** The actors and description viewable on the interface match the query results:

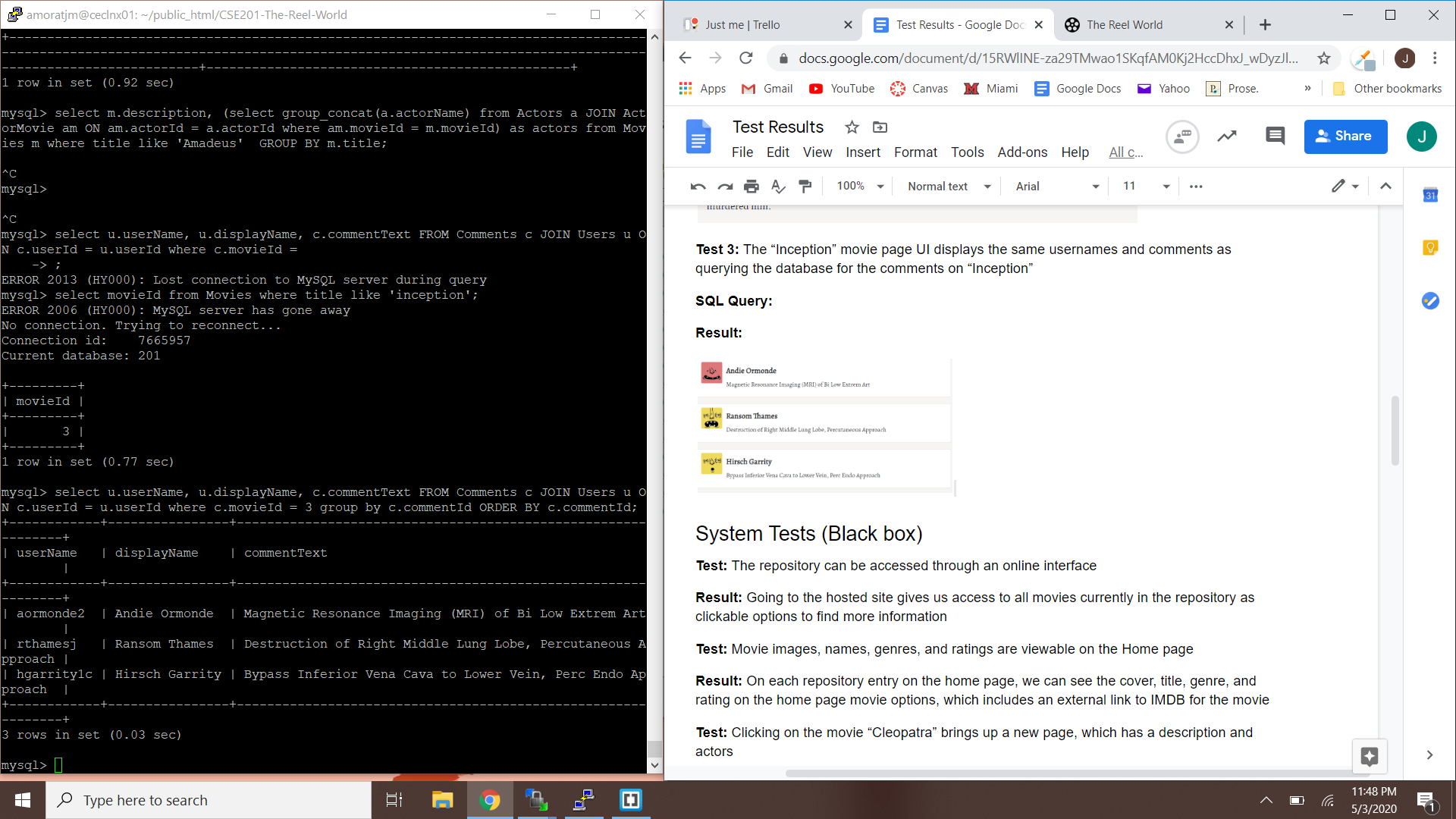
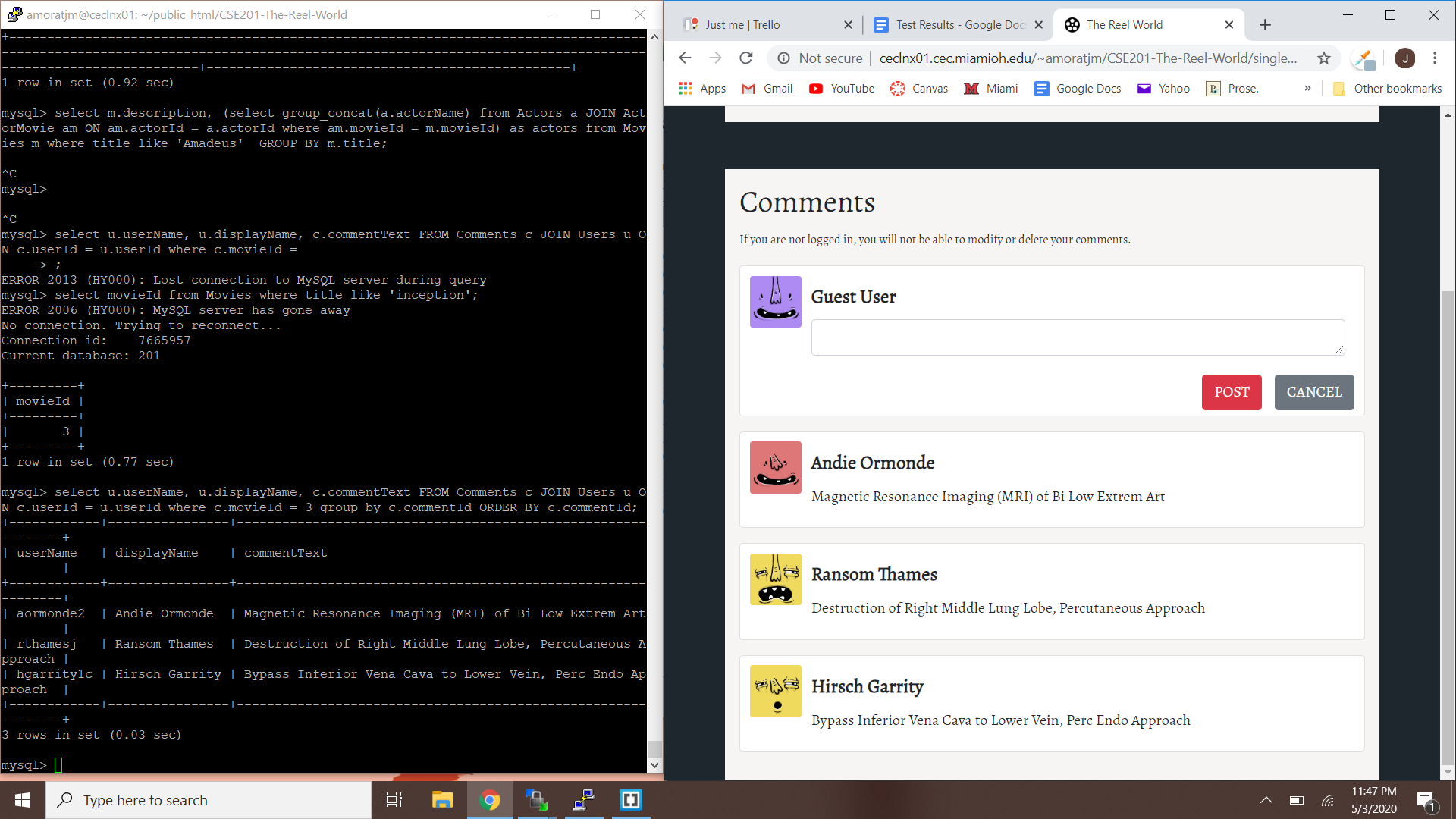
****

****

**Test 3:** The “Inception” movie page UI displays the same usernames and comments as querying the database for the comments on “Inception” (movieId = 3)

**SQL Query:** SELECT u.userName, u.displayName, c.commentText FROM Comments c JOIN Users u ON c.userId = u.userId WHERE c.movieId = 3 GROUP BY c.commentId ORDER BY c.commentId;

**Result:** The comments viewable on the interface match the query results:

****

## System Tests (Black box)

**Test:** The repository can be accessed through an online interface

**Result:** Going to the hosted site gives us access to all movies currently in the repository as clickable options to find more information

**Test:** Movie images, names, genres, and ratings are viewable on the Home page

**Result:** On each repository entry on the home page, we can see the cover, title, genre, and rating on the home page movie options, which includes an external link to IMDB for the movie

**Test:** Clicking on the movie “Cleopatra” brings up a new page, which has a description and actors

**Result:** The “Cleopatra” page show the title, actors list, and description for the correct movie entry

**Test:** The sort button can be used to sort movies alphabetically by title

**Result:** The movies start sorted alphabetically, so sorting by rating and then sorting again by title gives an alphabetical list of all movies in the repository

**Test:** The sort button can be used to sort movies by rating

**Result:** Choosing “by rating” in the sort button sorts all the movies on the home page repository by highest rating to lowest, and if they have the same rating, the movies are sorted alphabetically

**Test:** The filter button can be used to see movies only of the genres “Adventure” and “Thriller”

**Result:** Only movies that had adventure or thriller listed as a genre show up and state which genre the movie entry is there for, and clicking on the movie after filtering takes you to the correct movie page with correct information.

**Test:** It is possible to create an account and log in

**Result:** Signed up using name “John Smith,” user name “jsmith,” and password “password”. The fields are all checked for input accordingly to make sure the user put something in for each required field. An alert confirming account creation is shown at the top. You can then log in and enter the movie request page (demonstrating you’ve been logged in since it’s a page not previously accessible). The comment section also now shows your name, further demonstrating John Smith is logged in.

**Test:** A logged in user can submit a movie request through the movie request form

**Result:** If you’re not logged in, you’re not able to enter the movie request form, but once logged in you can. Submitting the request clears the fields and shows an alert at the top with submission confirmation. The request shows up on the user’s “My Requests” page with all the information that was input. It handled two new actors and a newline in the description input well. The option to cancel the movie request works, however the button disappears after a request has already been approved or denied.

**Test:** Admin can approve or deny movies and leave a related comment

**Result:** Admins can leave a comment and approve or deny the request. After doing so, the user who submitted the request can see the updated comment and the movie shows up on the homepage as part of the repository. The IMDB link is included for admins to manually check the validity of the submission.

**Test:** The search bar can be used to search the movies by titles containing “man”

**Result:** All titles that included the word man in the name came up on the home page and clicking on them took you to their respective page.

**Test:** A logged in user can leave a comment on a movie

**Result:** Leaving a comment shows the correct username based on who is currently logged in and it saves the state after leaving the page as well. Guest users can leave comments but no unique username is possible for them.

**Test:** Logged in users can edit or delete their own comments, but not others

**Result:** The logged in user can see the comment they made when posted with their full name above the comment. They then can see two buttons labelled “Edit” or “Delete.” When editing the comment, the new comment is posted at the bottom of the comment section as if it was a new comment. The delete button removes the comment on that entry and the state is saved for both instances. These buttons do not show up for other users, only the logged in one.

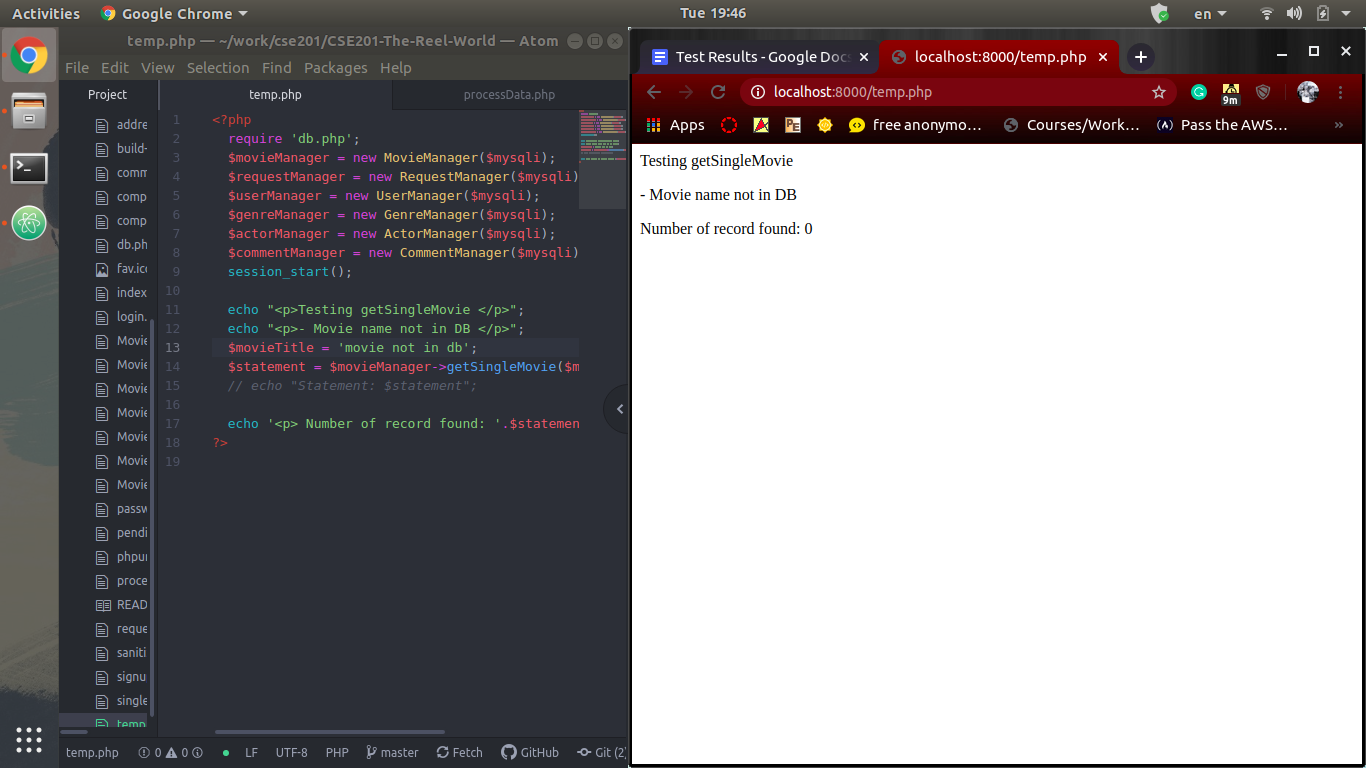
**Test:** Moderators have the power of normal users but can edit or delete any user comment

**Result:** Moderator roles can delete any comment left under a movie and it stays deleted as a result.

## White Box Tests

**Test:** The program doesn’t crash if getSingleMovie is passed a movie that doesn’t exist

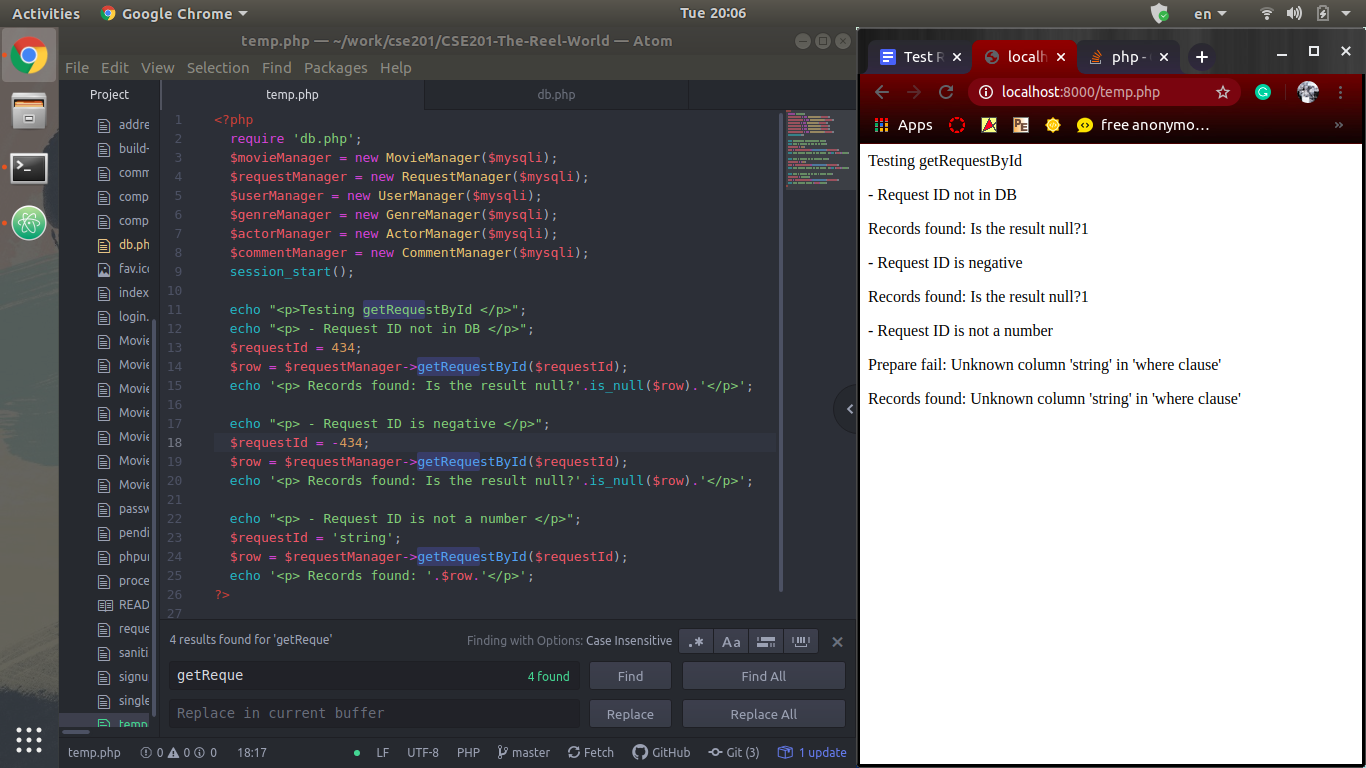
**Result:** In case the movie title is not found in the DB, the method will return 0 record but it will not crash. The single movie page will then not show any information of the movie since the movie is not in the DB. However, this situation will rarely happen since the singleMovie page is called directly from the link in the homepage with the movie title included.

****

**Test:** The program doesn’t crash if getRequestById is passed a request id that doesn’t exist

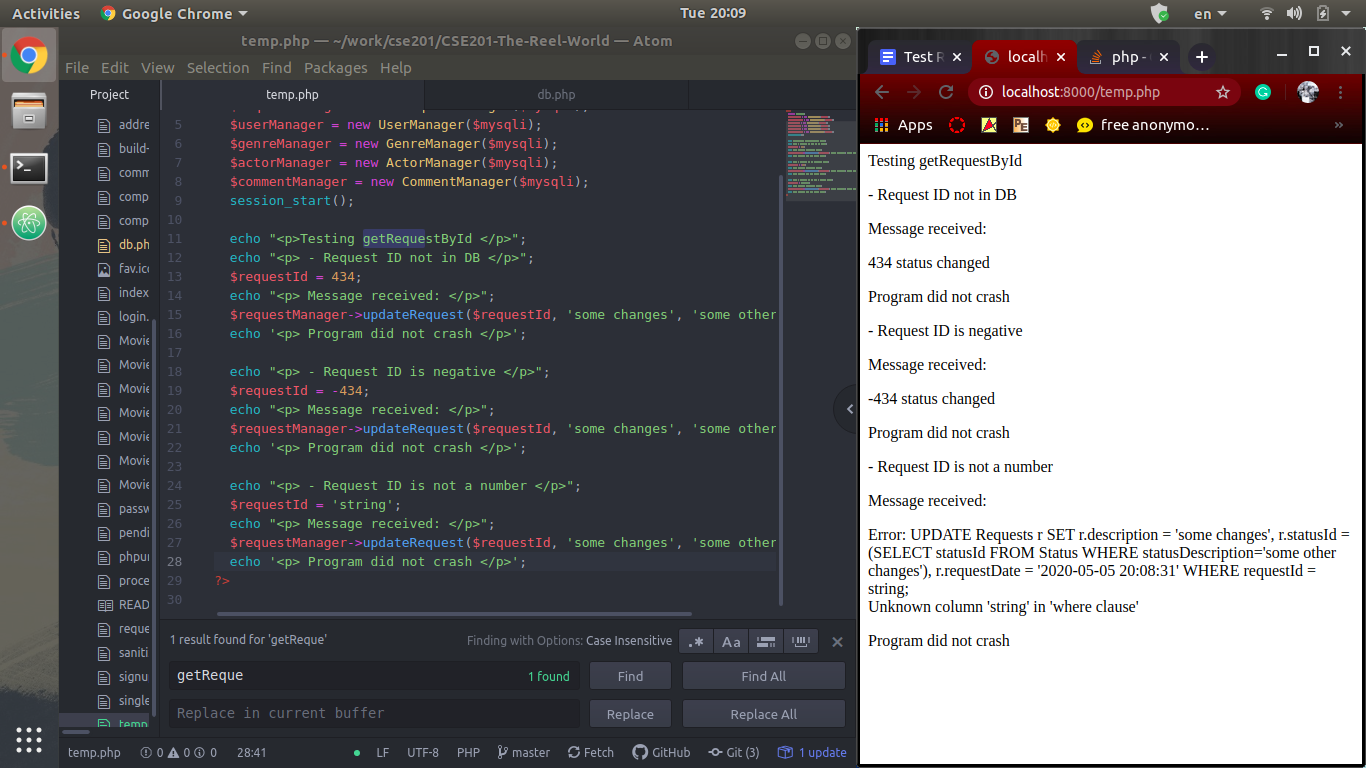
**Result:** If the requestId is not in the database, the result returned will be null and thus the request details will not be displayed. However, when the requestId is not a number, the program will not crash but an error message will be displayed. This is the expected behavior of the method because if the requestId is not a number, we cannot construct a valid MySQL request and thus an error message should be displayed.

Consider the fact that this method is only called internally, it’s hardly the case that an invalid (i.e. not a number) requestId will be passed in

****

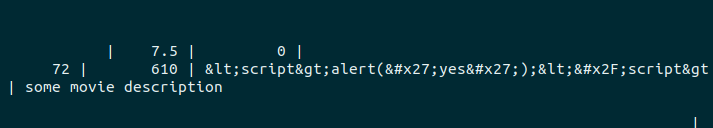
**Test:** The program doesn’t crash if updateRequest is passed a request that doesn’t exist

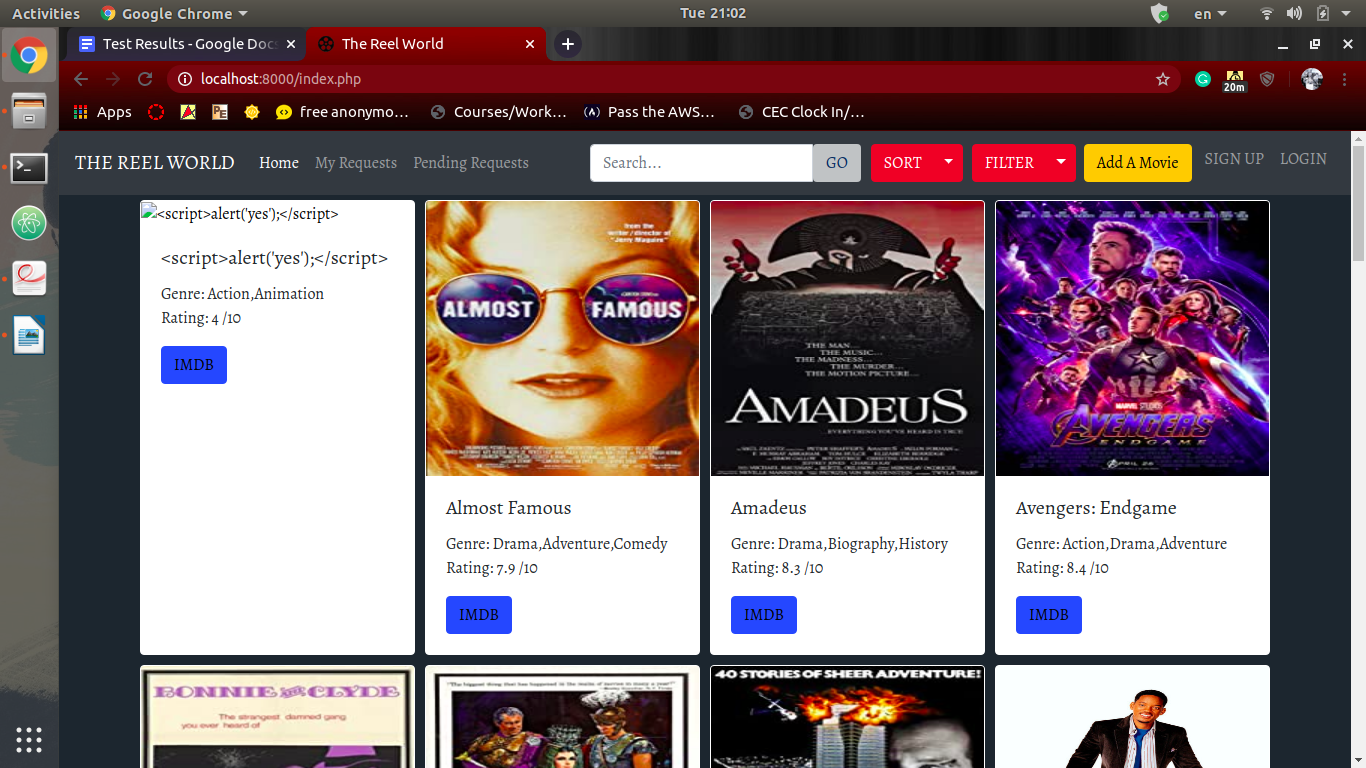
**Result:** Similar to getRequestById, the program did not crash when an invalid requestId is passed in. The message returned shows that some request has been updated, but since there is no request with such ID, no request is actually updated in the DB. If we pass in anything other than a number, an error message will be sent back, but the program will not crash.

****

**Test:** Users cannot input malicious code into the movie request form (injections)

**Result:** We have sanitize() method to escape all special characters and all input from the request form is encoded as json before being added to the database. Hence, the malicious input cannot affect the database. The following pictures depict an attempt to put some JavaScript alert code to the page. All special characters in the alert script are escaped, thus the script shows up as text but doesn’t modify the page behavior.

****

****