The Knowledge Hub Universities

Coventry University

School of Computing

KH6005CEM

Security

Coursework 2

Report

Module Leader:

Dr Ayman Taha

Eng. Kareem ElDebassy

By:

Abdallah Ibrahim

CU1900008

GitHub Repo:

GitHub Link

Table of Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Scope

In recent years, where many corporations depend highly on computer systems, and storing highly sensitive information in these systems, security has become a must for all computer systems. There has also been a rapid increase in the number of attackers on the internet, which is directly caused by the introduction of many new cyberattack tools which can be easily used to attack systems (Tunggal, 2022).

This report aims to analyse a system without any security features implemented, conduct a risk analysis for it to create a security recommendation, and finally, implement the recommended security features, so that the system can be fully secure.

1.2 Time Plan Activities

To begin with, a concrete time plan must be created so that the project can have a well-structured layout. The total amount of time for this project is three weeks, so the project must be divided accordingly.

The first phase for this project is to and define its features and purpose of the system and develop the system itself following the Software Development Lifecycle (SDLC) (Leau et al., 2012). To start with, planning the structure of the system will be done. Afterwards analysis and designing will help outline the key features. The next stage is to implement the features and finally, the system needs to be tested to make sure it is working. The purpose of the system is to allow customers to be able to create accounts and login with said accounts, to create and view restaurants in the system and post reviews for restaurant. The system aims to have full Create, Read, Update and Delete (CRUD) functionalities.

With two weeks remaining for the project, the next step is to implement the security features which will be needed to secure the system. These features include input validation, password integrity, permissions and many more. This phase will take one week, leaving the last week to test the security features for the system and make sure it is all working as intended. **Table 1** shows a Gantt chart which illustrates the time plan for the development of the secure system

	Week 1	Week 2	Week 3
System Planning			
SDLC			
Implementing Security Features			
Testing Software			
Writing Report			

Table 1 - Gantt chart showing time plan

2 System Design

2.1 System Description

The system which was developed for this coursework is a restaurant review system, where the expected outcome for customers is that they can view and post reviews for restaurants after logging in. The system has many features including, signing up, logging in, logging out, adding a restaurant, deleting a restaurant, editing a restaurant, viewing restaurant, posting reviews for a restaurant, deleting reviews, and viewing the review. The system is built using NodeJS for the back end, EJS for the front-end, and MySQL for the database. **Figure 1** below shows the activity diagram for the proposed system and the functionalities that it aims to have. The development phase for the system plans to take one week.

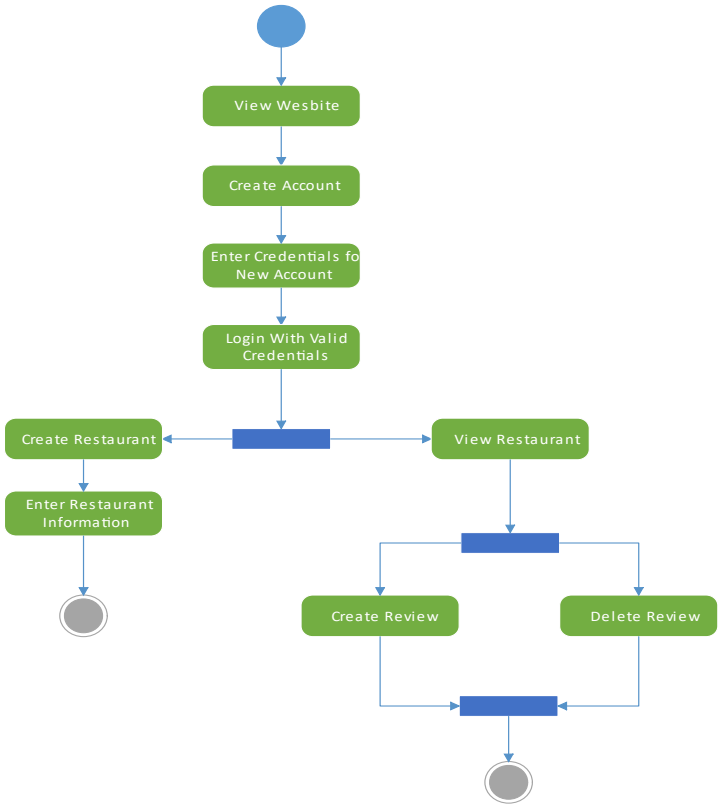

Figure 1 - Activity Diagram for System

The mentioned features of the system need to be secured to prevent customer data being leaked, to not allow any review data being manipulated, and to prevent any attacks on the server.

2.2 Potential Security Issues

The implemented system contains many assets, such as user information, and the data on the website showing the reviews for the restaurants. The system has many potential threats since there are no security measures in place to ensure that no attacks happen. A risk assessment will be made based on the current system to understand the possible risks, which will allow an opportunity to arise where the risks can be mitigated. **Table 2** below shows the risk assessment for the current system.

System Characterization	**Risk summary**	**Overall Risk**
User account passwords	There is no password policy being enforced to ensure that all users have strong password to prevent them from being hacked. There is also no hashing being used to protect the passwords.	Major
Access rights	Anyone can edit the data which is on the website, there are no access rights, so even if someone is not logged in or authenticated, they can make changes to the data on the website.	Major
User sessions	There is no session expiry, leaving the sessions and cookies vulnerable to attackers.	Medium

Input validation	Any data can be input into the textboxes, which will affect the database if for example a user enters a string when the database expects an integer.	Minor
Sending requests through postman	Some requests may not be directly accessed through the system. But attackers can try to bypass this by sending requests through applications like postman.	Medium
SQL Injection	Attackers can inject queries into the database to manipulate the database and attack it.	Major

Table 2 - Risk Assessment

Constructing this risk assessment has helped in defining all the possible threats and vulnerabilities which could occur for the system. Now with this, security measures and recommendations can be implemented to ensure that these risks are mitigated.

2.3 Recommendations and Proposed Security Features

To begin with, a password policy must be created and enforced, to ensure that users have strong passwords, and that attackers are unable to infiltrate user accounts. The password policy to be enforced is ensuring that passwords contain, at least one lower case letter, at least one uppercase letter, at least one number, and a minimum of eight characters. Enforcing said policy can help prevent user accounts from being attacked.

Password hashing will also improve security as they are designed to be a one-way function, making them difficult to reverse and attack (Sharan, 2017). Password hashing will help make the passwords in the system more secure, but they can still be attacked through brute force. To make the passwords in the system more secure, adding a salt to the hash will help. A salt is a random data string which is added to the password before it is hashed, which would further help with security and mitigate any attacks.

The next step is to define the access rights for the system. This will ensure that only when a user is authenticated and authorized, they can make changes to the data on the website. **Table 3** below shows the access control matrix for the system and shows the proposed access rights for the different actors in the system.

	View restaurant information and reviews	**CRUD for Restaurants**	**CRUD for reviews**
Logged-in User	View-only	Author for Restaurant can Access	Author for Review can Access
Visitor	View-only	No Access	No Access

Table 3 - Recommended Access Control Matrix

Another security feature which needs to be added to the system is making sure that all inputs which enter the database are valid. This will prevent any errors from happening and prevent any malicious input from entering the database. All text boxes in the system will validate if the correct data type is being entered and will stop any wrong inputs from entering the database.

The user sessions in the system do not expire, which causes many security issues. A way to mitigate this threat is to handle the user cookies and sessions to expire after 1 hour. This way, the user's data will still be stored for a short period of time to not cause a disturbance for the user, but still guarantee that the data is secure.

To combat the threat of SQL injections, the system will switch from using an SQL database, to a NoSQL database, like MongoDB.

3 Implementation

The first thing implemented into the system to improve security is to enforce the password policy. This will be done by editing the html form and add a pattern which is required for the password using regular expression. Afterwards, adding JavaScript and CSS, dynamic text can be added to the page to alert the user if they have met the requirements for the password or not. The form for the registration page will not be submitted until the user meets the requirements for the password.

To make sure the password is even more protected, hashing and salt will be added. The 'passport' library in JavaScript will be used to help hash the password when it is sent to the database, as well as adding a salt to the hash to make it even more secure. This way, the system has secured passwords by implementing password protection policies and hashing the password.

Next, is to implement the access control. In the system, users can create restaurants and edit them, so the system must make sure that only the author of the restaurant can edit the restaurant they created, and the same should be applied for the reviews for the restaurant. This is implemented by checking if the user who is logged in and checking the author of the restaurant/review from the database, and if they match, the user has the permission to be able to edit/delete the review. There also needs to be a check to see if the user is logged in so that they can post a review. The system checks if there is a current user logged in, and if there is not, adding the review is hidden and the visitor cannot access it. If a user tries to access something they do not have permission for, they are alerted and redirected to the login page. The requests are also protected if they do not have permissions, so they cannot bypass the authorization by attempting to send requests through 'Postman'. **Figure 2** shows the implementation of this, which checks if the user is logged in, and the id of the user matches the one for the author, they can delete their review.

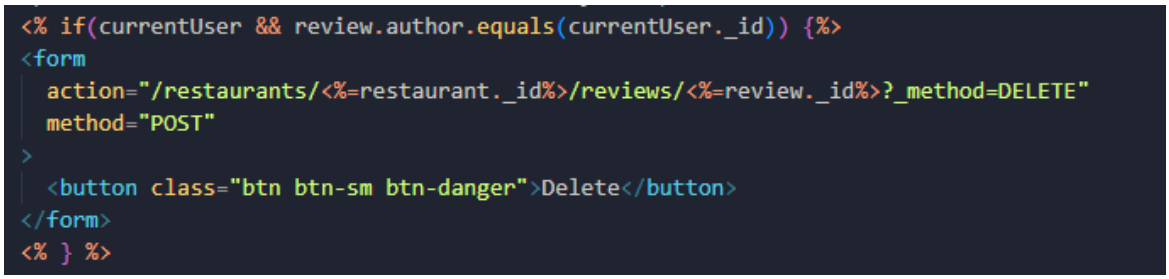

Figure 2 - Authorization check if user is author for review

To implement user sessions and cookies and make sure they are protected and expire, the JavaScript library, 'express-session' will be used since our server is running on the 'express' library. The user session will track the user's data for when they are logged into the web application, but when the session expires, the cookies will be deleted, and the user will need to log in again. The session is implemented to be deleted after 1 hour. Having this feature implemented into the system protects the web application from session hijacking, since each user request has a session secret which is stored securely in the browser (Dacosta et al., 2012). **Figure 3** shows the code configuration for the session.

```
const sessionConfig = {
  secret: "thisshouldbeabettersecret!",
  resave: false,
  saveUninitialized: true,
  cookie: {
    httpOnly: true,
    expires: Date.now() + 1000 * 60 * 60,
    maxAge: 1000 * 60 * 60,
  },
};
```

Figure 3 - Session and cookies configuration

The text boxes have all been implemented to accept only their respective valid inputs. For example, the text box where it is required to enter the price for dining at the restaurant, if someone tries to enter as string, the database will reject it and ask the user to enter a valid input (number). This is done using the library 'Joi', and the implementation of the input validation can be seen in **Figure 4**.

```
module.exports.restaurantSchema = Joi.object({
  restaurant: Joi.object({
    title: Joi.string().required(),
    price: Joi.number().required().min(0),
    image: Joi.string().required(),
    location: Joi.string().required(),
    description: Joi.string().required(),
  }).required(),
});

module.exports.reviewSchema = Joi.object({
  review: Joi.object({
    rating: Joi.number().required().min(1).max(5),
    body: Joi.string().required(),
  }).required(),
});
```

Figure 4 - Input validation using Joi

The final security implementation which is added to the system is preventing SQL injections. The way this is handled, is the system switched to using MongoDB, which is a NoSQL database, which makes it more secured than an SQL database, and being able to mitigate SQL injection attacks.

4 Testing

4.1 Test Plan

Now that the system has been implemented with the security features, the next phase is to test that these security features are working as intended. The test plan which will be used is to create test cases for

each of the different security features which have been implemented. The tables below show the different test cases for these features.

4.1.1 Password Test Plan

Test ID	Test Case	Pre-Condition	Test Steps	Expected Output
1	Verify that if user password does not meet password policy user cannot register	User is registering for account	1. User fills in registration form 2. User enters a password which does not meet password policy (At least 8 characters, 1 uppercase, 1 lowercase, 1 number)	Form should not be submitted, and user should be alerted that password is invalid
2	Verify that if user password meets password policy, user can register	User is registering for account	1. User fills in registration form 2. User enters a password which meets password policy	Form should be submitted, and user should be redirected to home page
3	Verify that user password should be hashed and salted in the database	User should have registered for an account	1. User submits registration form with valid information 2. Check database for password information	Database should store password with salt and hashed

Table 4 - Password test cases

4.1.2 Authorization Test Plan

Test ID	Test Case	Pre-Condition	Test Steps	Expected Output
4	Verify that user who is not logged in cannot create restaurant	User is not logged in	1. User attempts to access create restaurant route	User is redirected to login page
5	Verify that user who is not logged in cannot post review	User is not logged in	1. User attempts to access create review route	Review textbox should be hidden, and user should be redirected to login page
6	Verify that user who is not the author of restaurant is not able to edit restaurant information	User is not author of restaurant	1. Open restaurant show page 2. Try to access edit route for restaurant	System will alert user that they are unauthorized and redirect them to previous page
7	Verify that user who is not the author of review is not able to delete review	User is not author of restaurant	1. Open restaurant show page 2. Try to access delete route for review	System will alert user that they are unauthorized and redirect them to previous page

Table 5 - Authorization test cases

4.1.3 User Session Test Plan

Test ID	Test Case	Pre-Condition	Test Steps	Expected Output
8	Verify that user session is created and	User should be logged into the web application	1. User sends a new request to system 2. Inspect application storage and look at session and cookie details	Session should be created and set to expire after 1 hour

	reset after each request			
9	Verify that sessions expire after 1 hour of inactivity	User should be logged into the web application and inactive	1. User sends a request to the system 2. User is inactive and does not send requests for 1 hour	User session and cookies should be expired

Table 6 - User Session Test Cases

4.1.4 Input Validation Test Plan

Test ID	Test Case	Pre-Condition	Test Steps	Expected Output
10	Verify that all input fields are required	User should be entering data into field	1. User leaves input field empty 2. User submits form	System should alert user that there is missing data
11	Verify that email field accepts valid email	User should be registering for an account	1. User enters an invalid email 2. User submits form	System should alert user that the email is invalid
12	Verify that price field accepts a number that is a minimum of 0	User should be creating restaurant	1. User tries to enter a string or a number smaller than 0 2. User submits form	System should alert user that the price is invalid

Table 7 - Input Validation Test Cases

4.2 Test output

After creating the test plan, the final step in the testing process is to execute these tests and verify whether the tests have passed or not and provide evidence of the system passing these tests. The system will be manually tested to ensure that everything is working as intended, and that the system is being secured.

4.2.1 Password Test Outputs

Test ID	Test Case	Pre-Condition	Expected Output	Status
1	Verify that if user password does not meet password policy user cannot register	User is registering for account	Form should not be submitted, and user should be alerted that password is invalid	Pass

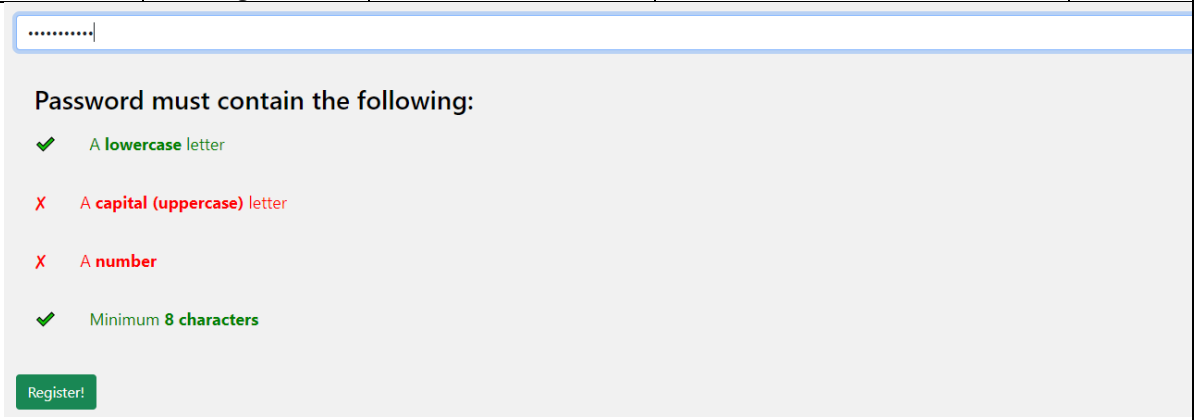

Figure 5 - Weak password test

Test ID	Test Case	Pre-Condition	Expected Output	Status
2	Verify that if user password meets	User is registering for account	Form should be submitted, and user should be redirected to home page	Pass

	password policy, user can register			

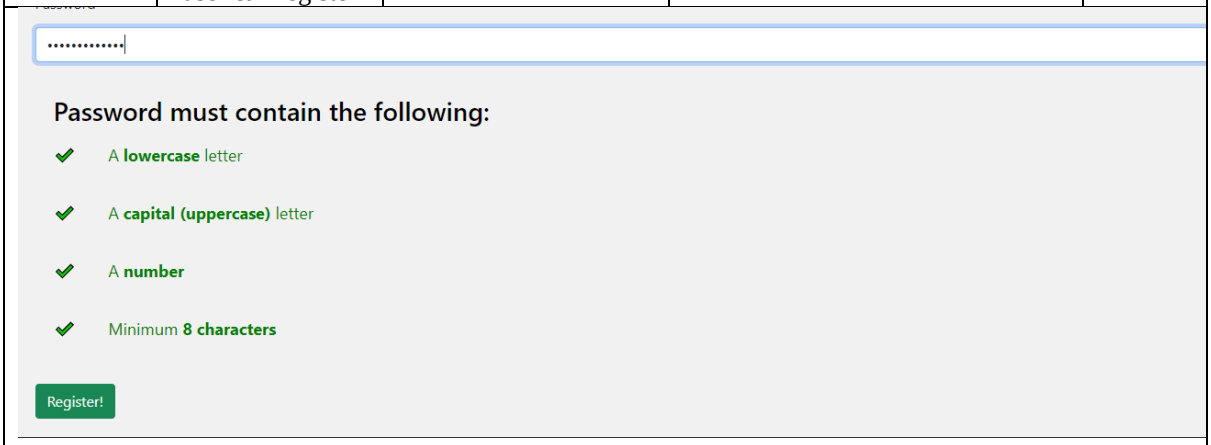

Figure 6 - Strong password test

Test ID	Test Case	Pre-Condition	Expected Output	Status
3	Verify that user password should be hashed and salted in the database	User should have registered for an account	Database should store password with salt and hashed	Pass

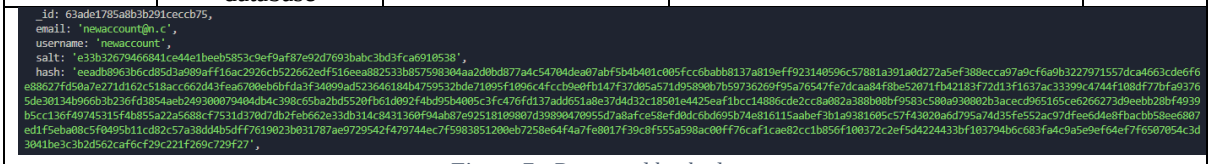

Figure 7 - Password hashed test

Table 8 - Password test outputs

4.2.2 Authorization Test Outputs

Test ID	Test Case	Pre-Condition	Expected Output	Status
4	Verify that user who is not logged in cannot create restaurant	User is not logged in	User is redirected to login page	Pass

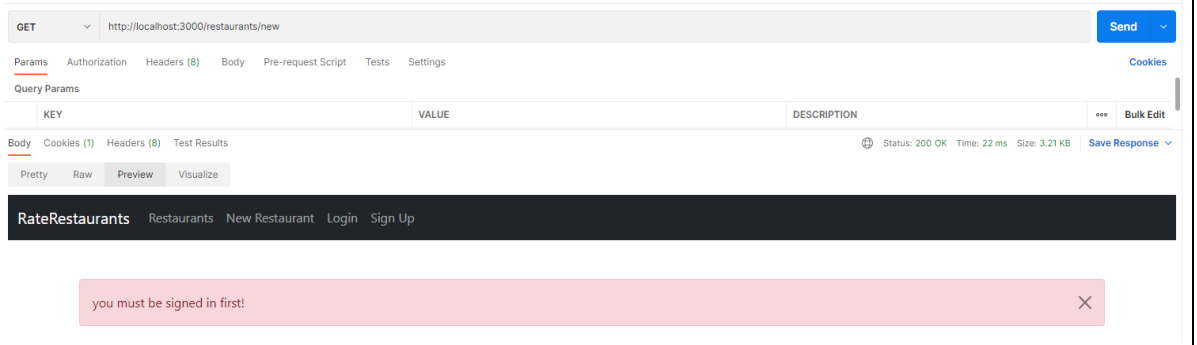

Figure 8 - Not logged in user create restaurant test

Test ID	Test Case	Pre-Condition	Expected Output	Status
5	Verify that user who is not logged in cannot post review	User is not logged in	Review textbox should be hidden, and user should be redirected to login page	Pass

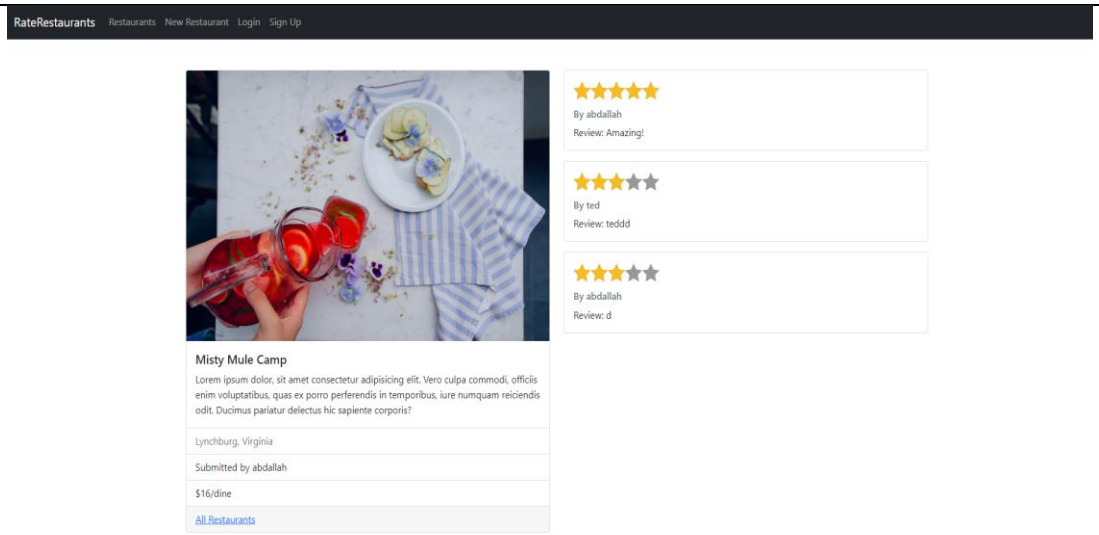

Figure 9 - Not logged in user cannot post review test

Test ID	Test Case	Pre-Condition	Expected Output	Status
6	Verify that user who is not the author of restaurant is not able to edit restaurant information	User is not author of restaurant	System will alert user that they are unauthorized and redirect them to previous page	Pass

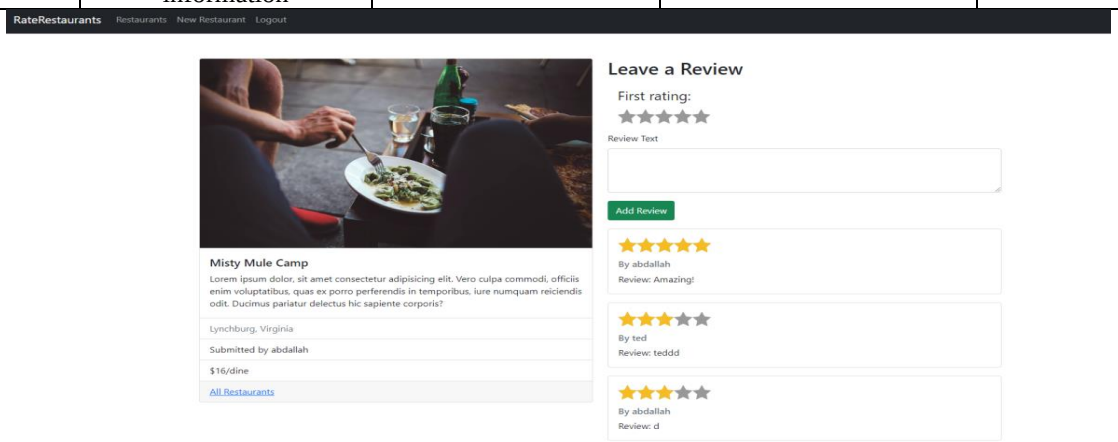

Figure 10 - User who is not restaurant author cannot edit details test

Test ID	Test Case	Pre-Condition	Expected Output	Status
7	Verify that user who is not the author of review is not able to delete review	User is not author of restaurant	System will alert user that they are unauthorized and redirect them to previous page	Pass

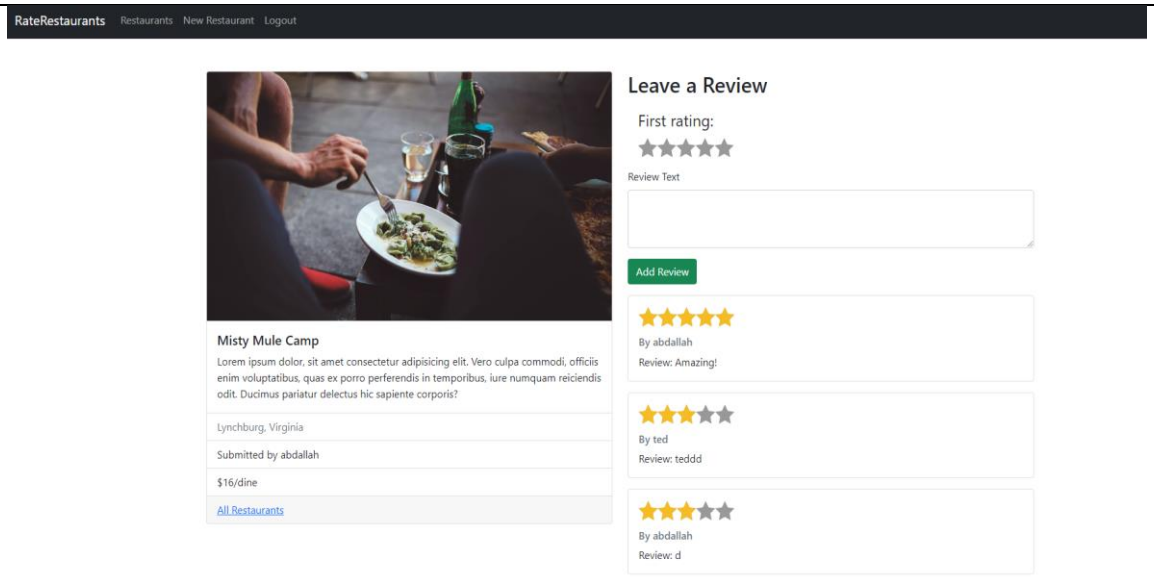

Figure 11 - User who is not review author cannot delete review test

Table 9 - Authorization test outputs

4.2.3 User Session Test Outputs

Test ID	Test Case	Pre-Condition	Expected Output	Status
8	Verify that user session is created and reset after each request	User should be logged into the web application	Session should be created and set to expire after 1 hour	Pass

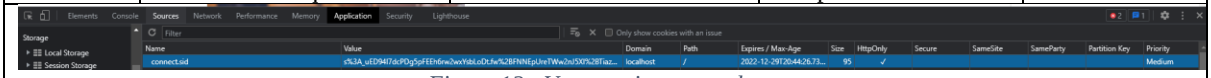

Figure 12 - Very session created test

Test ID	Test Case	Pre-Condition	Expected Output	Status
9	Verify that sessions expire after 1 hour of inactivity	User should be logged into the web application and inactive	User session and cookies should be expired	Pass

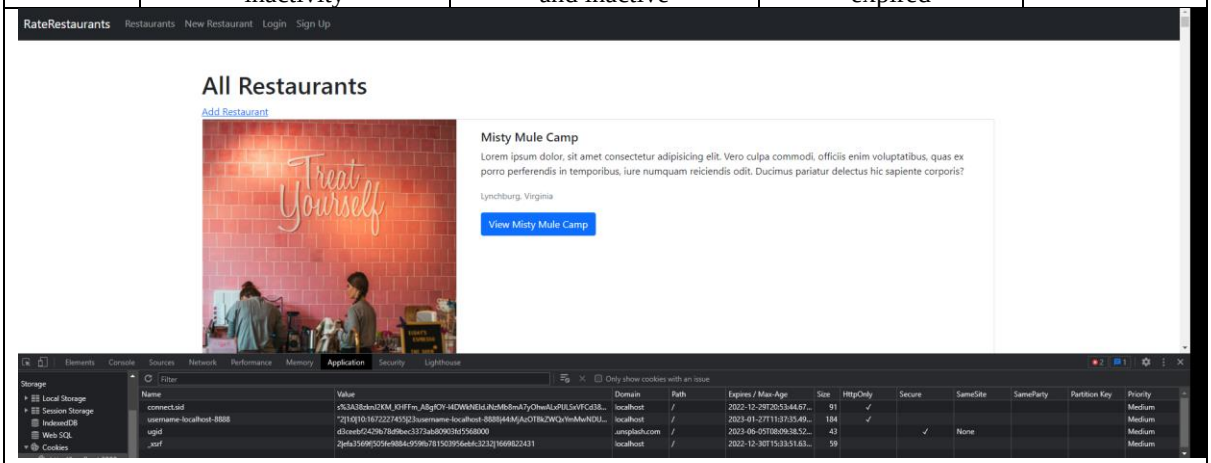

Figure 13 - Verify session expired test

Table 10 - User session test outputs

4.2.4 Input Validation Test Outputs

Test ID	Test Case	Pre-Condition	Expected Output	Status
10	Verify that all input fields are required	User should be entering data into field	System should alert user that there is missing data	Pass

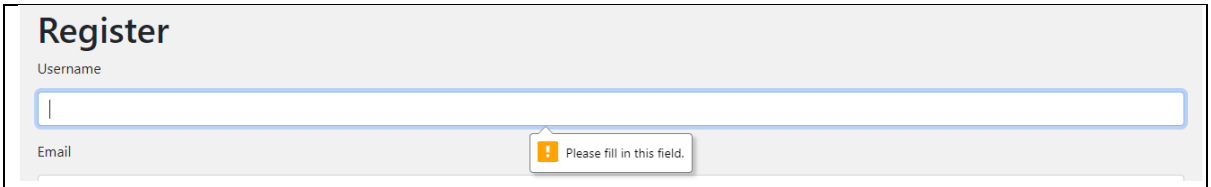

Figure 14 - Verify inputs are required test

Test ID	Test Case	Pre-Condition	Expected Output	Status
11	Verify that email field accepts valid email	User should be registering for an account	System should alert user that the email is invalid	Pass

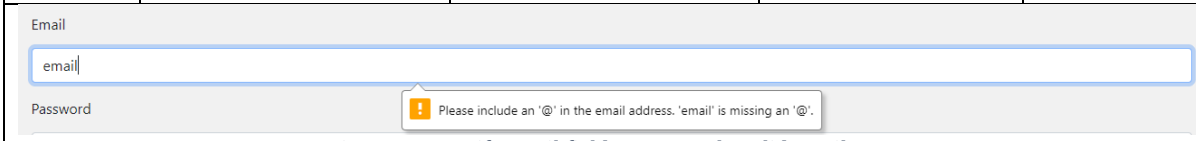

Figure 15 - Verify email field accepts only valid email test

Test ID	Test Case	Pre-Condition	Expected Output	Status
12	Verify that price field accepts a number that is a minimum of 0	User should be creating restaurant	System should alert user that the price is invalid	Pass

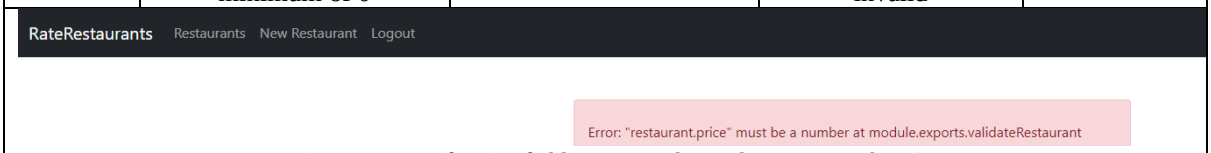

Figure 16 - Verify price field accepts only numbers greater than 0

Table 11 - Input validation test outputs

From these test outputs, the system has been fully tested and has passed all the tests.

5 Evaluation

5.1 Vulnerability Scan

Now that the system has been developed and tested, a vulnerability scanner will be used to make note of the vulnerabilities that the system has, to be able to give an evaluation of the system. The 'Nessus' vulnerability scanner will be used for this. **Figure 17** shows the results of the vulnerability scan.

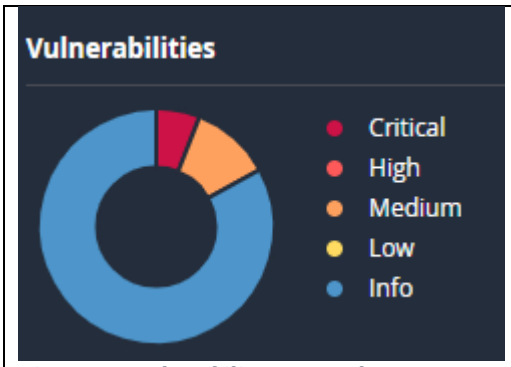

Figure 17 - Vulnerability scan results

From this, it can be seen that overall, the system is secured, with only having 1 critical vulnerability, which is related to the MongoDB authentication, which will be fixed in the future.

5.2 Conclusion

The RateRestaurants system which has been implemented made use of a variety of different security features and demonstrated their importance in securing a system. The system at hand was also able to follow the CERT guidelines for secure coding practices, which include features like, input validation, principle of least privilege, and adopting a secure coding standard (Seacord, 2018), which are practices that have been followed throughout the development of this system.

It can also be concluded that the RateRestaurants system is able to mitigate the threats that were identified in the risk assessment, therefore showing that there is an understanding of the security threats which could arise in a security system. The system was also fully tested to prove that it could mitigate the mentioned security threats, hence making it a fully secure system, protecting from possible attacks.

6 References

Dacosta, I., Chakradeo, S., Ahamad, M., & Traynor, P. (2012). One-time cookies: Preventing session hijacking attacks with stateless authentication tokens. *ACM Transactions on Internet Technology*, *12*(1), 1–24. https://doi.org/10.1145/2220352.2220353

Leau, Y. B., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). *Software Development Life Cycle AGILE vs Traditional Approaches*.

Seacord, R. (2018). *Top 10 Secure Coding Practices—CERT Secure Coding—Confluence.* https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices

Sharan, A. (2017, May 5). Passwords and Cryptographic hash function. *Passwords and Cryptographic Hash Function*. https://www.geeksforgeeks.org/passwords-and-cryptographic-hash-function/

Tunggal, A. (2022). *Why is Cybersecurity Important? | UpGuard*. https://www.upguard.com/blog/cybersecurity-important