



Bee Design Award Competition System - Security Report

Global Design

V1.0

27th June 2021

WONG WEI JIAN & ADRIEL GOH WEI EN

Global Design Security Report

Terms of Use

The information provided in this document is strictly for viewing only. This document must not be reproduced in whole or in part or used for any purposes except under written consent with the authors of this documents.

© Global Design 2021

Document Log

Title	Version	Release Type	Author	Date
Competition System - Security Report	0.1	Draft	Wong Wei Jian	4 th June 2021
Competition System - Security Report	0.2	Draft	Adriel Goh Wei En & Wong Wei Jian	11 th June 2021
Competition System - Security Report	0.3	Draft	Adriel Goh Wei En & Wong Wei Jian	15 th June 2021
Competition System - Security Report	0.4	Draft	Adriel Goh Wei En & Wong Wei Jian	17 th June 2021
Competition System - Security Report	0.5	Draft	Adriel Goh Wei En & Wong Wei Jian	19 th June 2021
Competition System - Security Report	0.6	Draft	Adriel Goh Wei En & Wong Wei Jian	22 nd June 2021
Competition System - Security Report	0.7	Draft	Adriel Goh Wei En & Wong Wei Jian	24 th June 2021
Competition System - Security Report	0.8	Draft	Adriel Goh Wei En & Wong Wei Jian	26 th June 2021
Competition System - Security Report	1.0	Final Release	Adriel Goh Wei En & Wong Wei Jian	27 th June 2021

Table of Contents

1	Executive Summary	5
1.1.	Overview	5
1.2.	Our Team	5
2	Key Findings	6
2.1.	Introduction	6
2.2.	Terminologies	8
2.3.	Tools List	9
2.4.	Key Issues.....	9
3	Detailed Findings	11
	Vulnerability 1 – SQL Injection	11
	Vulnerability 2 – Broken Authentication (Input Validation)	19
	Vulnerability 3 – Logging and Monitoring.....	23
	Vulnerability 4 – Sanitize JSON Results	25
	Vulnerability 5 – Broken Authentication (Restricted Pages Access)	28
	Vulnerability 6 – Broken Access Control	36
	Vulnerability 7 – Cross-site Scripting Attacks (XSS)	42
	Vulnerability 8 – Sensitive Data Exposure (localStorage).....	48
	Vulnerability 9 – Sensitive Data Exposure (HTTP)	54
	Other Code Improvements – Adding Regular Expressions to Check Inputs	57
	Other Code Improvements – Wrong Redirection	58
	Other Code Improvements – Code Reusability.....	58
	Other Code Improvements – Salted Passwords	59

1 Executive Summary

1.1. Overview

Global Design approached Singapore Polytechnic's Enterprise System Development (ESDE) Students to conduct a code check of its Bee Design Award Competition System. The purpose of this engagement was to get the ESDE Team to evaluate the security of the system to validate its security mechanisms and identify possible vulnerabilities. The possible vulnerabilities found will be documented and with remedial advice provided.

This report also details the scope of the testing conducted, with all significant findings with detailed remedial advice documented. This summary has been written such that it provides a non-technical audience with the key findings and associate it to the impact on the business.

1.2. Our Team

Role	Name
Project Leader	Wong Wei Jian (P1908365)
Project Member	Adriel Goh Wei En (P1907872)

2 Key Findings

2.1. Introduction

During our initial scan of the codes, we found that:

- (a) The program uses multiple callbacks (refer to Terminologies) when performing an action. This could lead to the system hanging like in the login page where if someone spams the login, the system will end up hanging after a few minutes. To avoid the usage of nested callbacks, there could be implementation of using Asynchronous functions or JavaScript promises.
- (b) There were inconsistent JSON response data in several JavaScript files. To improve readability as well as ensure that the entire system is more streamlined and consistent, the JSON returns should be standardized to one style rather than multiple styles which indicate that there are more than 1 developer working the codes. Some JSON returns are put directly inline, while some are crafted into a variable before passing the variable.

Original Codes	Fixed Codes
backEnd > src > controllers > authController.js (exports.processLogin)	
<pre>exports.processLogin = (req, res, next) => { let email = req.body.email; let password = req.body.password; try { auth.authenticate(email, function (error, results) { if (error) [let message = 'Credentials are not valid.'; //return res.status(500).json({ message: message }); //If the following statement replaces the above statement //to return a JSON response to the client, the SQLMap or //any attacker (who relies on the error) will be very happy //because they relies a lot on SQL error for designing how to do //attack and anticipate how much "rewards" after the effort. //Rewards such as sabotage (seriously damage the data in database), //malicious sql injection, etc. return res.status(500).json({ message: error });] else { if (results.length == 1) { if ((password == null) (results[0] == null)) { return res.status(500).json({ message: 'Login failed' }); } if (bcrypt.compareSync(password, results[0].user_password) == true) { let data = { user_id: results[0].user_id, role_name: results[0].role_name, token: jwt.sign({ id: results[0].user_id }, config.JWTKey, { expiresIn: 86400 //Expires in 24 hrs }) }; //End of data variable setup return res.status(200).json(data); } else { //return res.status(500).json({ message: 'Login failed.' }); return res.status(500).json({ message: error }); } } } } catch (error) { return res.status(500).json({ message: error }); } } //End of try };</pre>	<pre>try { auth.authenticate(email, function (error, results) { if (error) [let message = 'Credentials are not valid.'; //return res.status(500).json({ message: message }); //If the following statement replaces the above statement //to return a JSON response to the client, the SQLMap or //any attacker (who relies on the error) will be very happy //because they relies a lot on SQL error for designing how to do //attack and anticipate how much "rewards" after the effort. //Rewards such as sabotage (seriously damage the data in database), //malicious sql injection, etc. let jsonResult = { message: 'Credentials are not valid.' }; return res.status(500).json(jsonResult);] else { if (results.length == 1) { if ((password == null) (results[0] == null)) { let jsonResult = { message: 'Login Failed' }; return res.status(500).json(jsonResult); } if (bcrypt.compareSync(password, results[0].user_password) == true) { let jsonResult = { user_id: results[0].user_id, role_name: results[0].role_name, token: jwt.sign({ id: results[0].user_id, uRole: results[0].role_name }, config.JWTKey, { expiresIn: 7200 //Expires in 2 hrs }) }; //End of jsonResult variable setup return res.status(200).json(jsonResult); } else { let jsonResult = { message: 'Login has failed..' }; return res.status(500).json(jsonResult); } } } } catch (error) { let jsonResult = { message: 'An error has occurred!' }; return res.status(500).json(jsonResult); } };</pre>

As seen in the above code (highlighted yellow), you can see that throughout the page, there is inconsistent JSON object returns... Some returning the error directly to the frontEnd, some returning customized message to the frontEnd, and some returning data to the frontEnd

To have a more consistent, standardized and streamlined view, we have decided to separate the returns into a variable (JSON Object) declaration first, before sending the variable (JSON Object) back to the frontEnd

This allows more readability of the codes, as well as more clarity on what you are sending back to the frontEnd.

Original Codes	Fixed Codes
backEnd > src > controllers > userController.js (exports.processGetUserData, exports.processGetOneUserData)	
<pre> exports.processGetUserData = async(req, res, next) => { let pageNumber = req.params.pagenumber; let search = req.params.search; try { let results = await userManager.getUserData(pageNumber, search); console.log('Inspect result variable inside processGetUserData code\n', results); if (results) { var jsonResult = { 'number_of_records': results[0].length, 'page_number': pageNumber, 'userdata': results[0], 'total_number_of_records': results[2][0].total_records } return res.status(200).json(jsonResult); } } catch (error) { let message = 'Server is unable to process your request.'; return res.status(500).json({ message: error }); } }; //End of processGetUserData </pre>	<pre> if (userId == decodedId && decodedRole == "admin") { try { let results = await userManager.getUserData(pageNumber, search); console.log('Inspect result variable inside processGetUserData code\n', results); if (results) { var jsonResult = { 'number_of_records': results[0].length, 'page_number': pageNumber, 'userdata': results[0], 'total_number_of_records': results[2][0].total_records } return res.status(200).json(jsonResult); } } catch (error) { console.log("Error Occured: " + error); } let jsonResult = { message: 'Server is unable to process your request' } return res.status(500).json(jsonResult); } </pre>
<pre> exports.processGetOneUserData = async(req, res, next) => { let recordId = req.params.recordId; try { let results = await userManager.getOneUserData(recordId); console.log('Inspect result variable inside processGetOneUserData code\n', results); if (results) { var jsonResult = { 'userdata': results[0], } return res.status(200).json(jsonResult); } } catch (error) { let message = 'Server is unable to process your request.'; return res.status(500).json({ message: error }); } }; //End of processGetOneUserData </pre>	<pre> exports.processGetOneUserData = async (req, res, next) => { console.log("====="); console.log("userController.js > processGetOneUserData is called and running!"); console.log("====="); let recordId = req.params.recordId; try { let results = await userManager.getOneUserData(recordId); console.log('Inspect result variable inside processGetOneUserData code\n', results); if (results) { var jsonResult = { 'userdata': results[0], } return res.status(200).json(jsonResult); } } catch (error) { console.log(error); let jsonResult = { message: 'Server is unable to process your request' }; return res.status(500).json(jsonResult); } }; //End of processGetOneUserData </pre>

2.2. Terminologies

Below are the terminologies that our groups will be using for the following report. These explanations will be able to guide you into understanding the different terms that professionals use for doing security checks for the entire system:

Terminologies	Meanings
Callbacks / Callback Functions	Callbacks / Callback Functions is known as a “Call-After” function whereby the other function is passed as a parameter in the original function, and the other function is expected to call back the original function once it is complete.
CRUD	CRUD or Create, Retrieve, Update and Delete, is a terminology or shortcut used by developers to modify the database.
Query Statement	Query statements are statements that are sent to the database to retrieve any data based on the statements sent, from the database.
SQL Injection	SQL is the type of database that the companies use to store their application’s data in. SQL Injection is where an attacker can send an input into the database (directly or indirectly) to tamper with the data in the database.
Authentication	Authentication is the action of validating the user’s identity in the system before it can perform any action.
Authorization	Authorization is the action of checking the user’s authorization level within a system, determining the type of access the user is granted.
Input Validation	Input validation is the code that exists in the client’s side (Front-End) that will do the checking of the inputs that the user enters to ensure that it met the requirements set out by the developer or owner.
Output Sanitization	Output sanitizations is the code that exists in the developer’s side (Back-End) that will, based on the outputs of the system (whether success, failure, and dependent on the HTTP Status Code), sends the appropriate response back to the user without revealing too much details about the outputs
Broken Access Control	Broken Access Control is when a user can act outside their intended permissions and perform actions or functions that are not intended for them.
Stored Cross-site Scripting (XSS)	Cross-site Scripting (XSS) is when a piece of JavaScript code, typically a script, is inserted and stored in the database and runs when that particular data is queried

Regular Expressions	Regular expressions (Regex) are a “String-of-Conditions” that has special meanings when processed as a regular expression. This “String-of-Conditions” processes text fields to determine if the text matches the “String-of-Conditions”. If it does not match the conditions set out, the text is invalid, otherwise it will be valid.
---------------------	---

2.3. Tools List

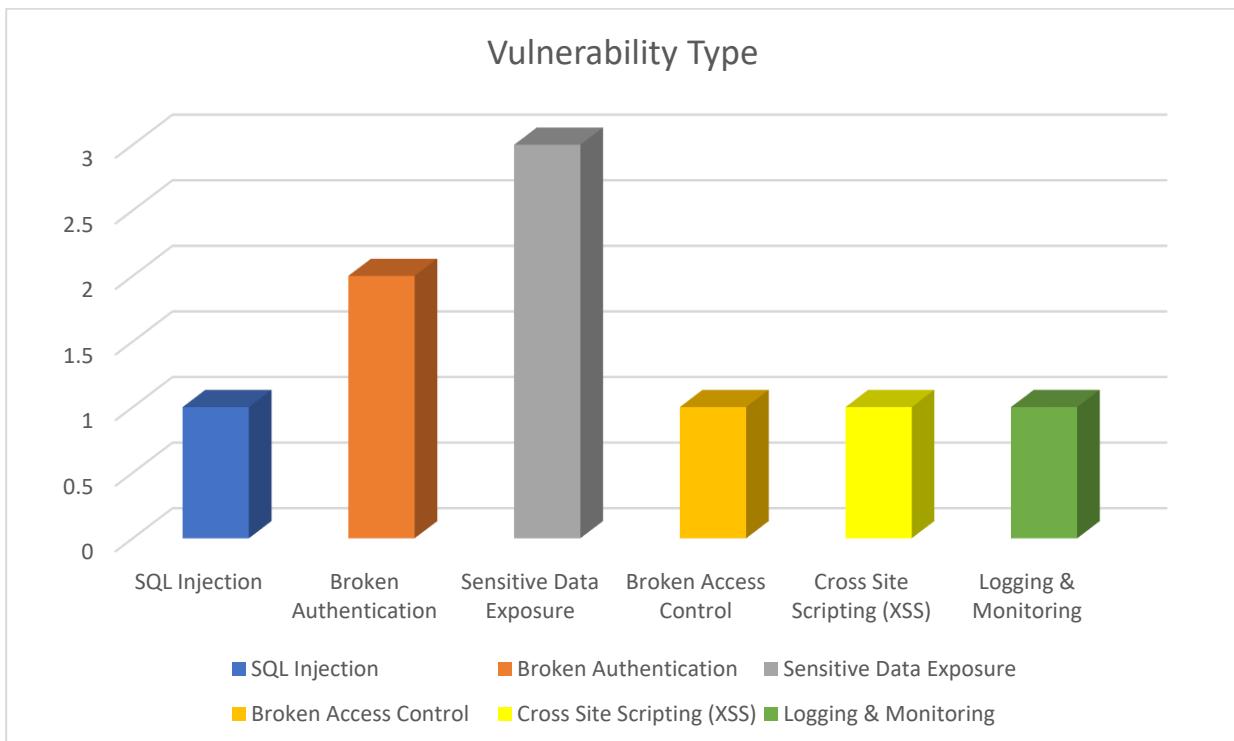
Below are the tools and Integrated Development Environment (IDE) that we have used to do the testing of the Bee Design Award Competition System. Most of these tools are used by professionals in the industry to test out different aspects of their systems.

Tools Used	Targeted Test Area
Burp Suite	Used to intercept the HTTP requests and/or any traffic made by the users
Postman	Used to target the Back-End server to see the responses that are received back from the server
Visual Studio Code (VSC)	This is one of the many professional Integrated Development Environment (IDE) that professionals use to write their codes in. It provides a nice, formatted view of the codes meant for developers to develop their codes in

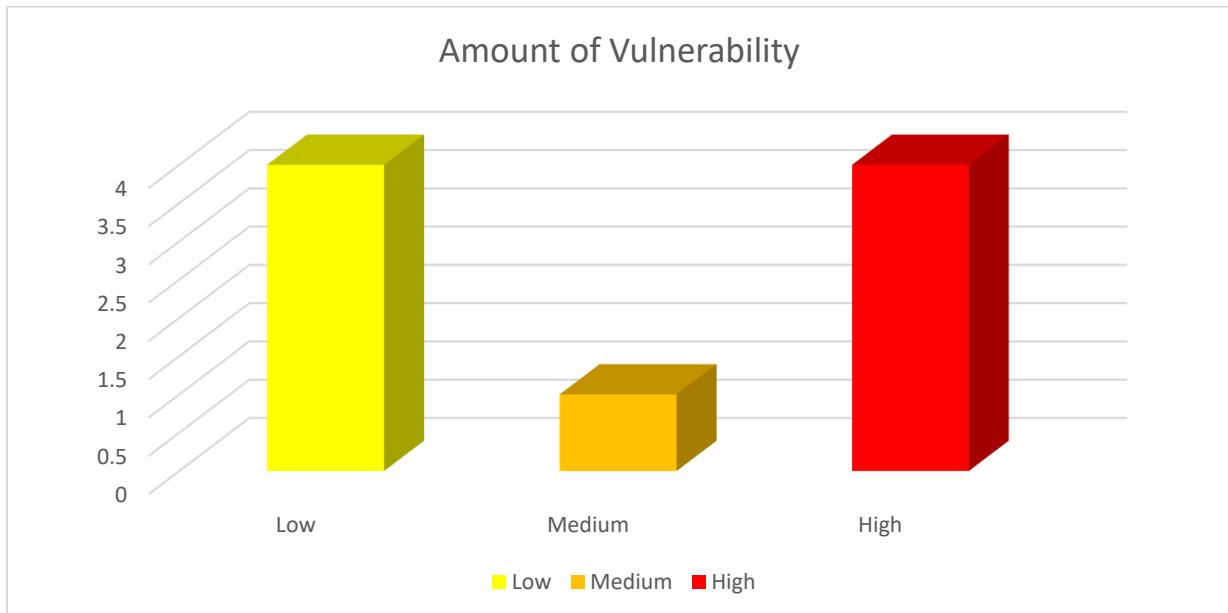
2.4. Key Issues

Through our thorough examination of Bee Design Award Competition System’s security, our team has found that there are inadequate amounts of security tools and features implemented and as such, has led to numerous vulnerabilities found in the code.

Our team has categorized the vulnerabilities found to 6 different types and 3 levels of impact it will have on the system:



Graph above shows the different vulnerability types our team has found.



Graph above shows the different vulnerability impacts it will have on the system.

In short, we have found that the system is prone to easy and simple XSS and SQL injection that makes the system very vulnerable. It is possible for a novice hacker to hack into and attack the system along with the database without much difficulty. Due to the lack of input validation, users can input malicious code and queries to tamper with the backend database as well as inject scripts into the database, harming the whole integrity of the system as well as potentially exposing danger to the administrators of the system.

Additionally, there has been little to no user authentication and authorization when functions are called from the website, leaving all pages accessible by anyone and with any authority level.

The following highlights the areas that are of upmost importance when dealing with web security as well as what the system is prone to be attacked by if the system is targeted.

Key Issues	Issue Area	Impact
A significant number of query statements were found to be prone to SQL Injection attacks. Any attacker who attacks this system will be able to gain access to the database, modify the database as well as retrieve and delete data (CRUD) from the database.	SQL Injection	High
A significant number of pages were found to have no user authorization and authentication. This means that low-privileged users can gain access to administrator pages of the website. Besides that, a significant number of functions were also found to have no authorization checks leading to low-privileged users being able to run administrative functions like editing a user's permission level.	Broken Authentication & Access Control	High
A significant number of functions were assessed to be vulnerable to Cross-site Scripting (XSS) attacks. This allows attackers to steal information by targeting at users rather than the system itself, and hence able to portray themselves as the targeted user to run commands within the system	XSS	High

3 Detailed Findings

Vulnerability 1 – SQL Injection

Vulnerability ID	Impact	Ease of Exploit	Recommendations
V-01 SQL Injection	When any user uploads a file into the database, they can craft and insert any SQL statements into the text field and when submitted, it will tamper the data stored. This will result in having a possible database exposure.	In the login.html page, the email input is vulnerable to SQL injection. When a SQL statement is inserted, it can tamper with the database and delete users as seen in the demo screenshots on page12.	By simply changing the SQL statement in fileService.js to a prepared statement, we can prevent SQL injection as the user input will be handled as a string for the parameter instead of a SQL statement. In the code changes screenshots provided below, we have changed the "\${}" to a "?" placeholder. Hence, when the values are called it prevents a SQL injection attack in the fileService.js Furthermore, the user input should be filtered as an attacker could still attempt a SQL injection attack.

Demo Screenshots (Before)

- Uploading of New Files & Loggin In

Your email
 rita'; DELETE FROM file WHERE email = 'bob@designer.com

Your password
 ••••••••

SUBMIT

- Public can register as a user
- The database has been **seeded** with user test data.
- user:rita@designer.com password:password

The above screenshot is used to test to see if there are any SQL response received based on the input

	user_id	fullname	email	user_password	role_id
▶	112	ahtan	ahtan@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2
	108	amy	amy@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2
	109	anita	anita@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2
	107	ashley	ashley@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2
	102	bob	bob@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2
	103	braddy	braddy@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2
	110	eddy	eddy@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2
	140	ewrwrwer	erwrwer@wewe.wewe	\$2b\$10\$csEQVzdRDGM4t.eOMUJ3NOqQqecsOBPQ0zFr0RVjyV4jO3SAwcqgW	2
	106	fred	fred@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2
	114	gabby	gabby@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2
	142	hashtest	hashtest@gmail.com	\$2b\$10\$sf.AHTWCsRJCS1N7AQs2E.fan0BOck1Y2/UNYICsXSOfD9WxtyMZC	2
	153	hasht...	hashtest10@gmail.com	\$2b\$10\$wugeCnj7hFmS9eLJAQq8luBVJGUbtT/yhKR5YmqZqzG4yuWgSIyS.Z.	2
	144	hashtest2	hashtest2@gmail.com	\$2b\$10\$4f/M6zw9Q93ir5AISC6wUekCYJktRDzDh4Weh7vmjhTbnVEk3R6cC	2
	145	hashtest3	hashtest3@gmail.com	\$2b\$10\$PbZ2dQlkxHUBapVmsj0.xeEDN/p40kzPQ5L5SoC38MPdH6O292Fc6	2
	146	hashtest4	hashtest4@test.com	\$2b\$10\$s0/4qiE4dcGWz.mOBpbhT.PEPWKt4rYw/A6So0bIpDxz6NWXY7kqi	2
	147	hashtest5	hashtest5@test.com	\$2b\$10\$fJ1H42rwNJT//ELALOq7Zegp6KqgT/YDYmpbbhSiWYG3QCB1zdsae	2
	148	hashtest6	hashtest6@test.com	\$2b\$10\$pDjMQ4uYTawn40pUtbumqn.jauK/h047.FaDI28/h2GNyjTRg9xRza	2
	149	hashtest7	hashtest7@test.com	\$2b\$10\$aQeAdkCBh,60s8oD6Xtth.HTy2rcKt1bjkw5TAJid8fxnd3PbSwq6	2
	150	hashtest8	hashtest8@test.com	\$2b\$10\$OhCKYqA8nmMfDmSfOqXAjO7dRbJ4/oJkP7O4C3i76qVOcPnAsm1bi	2
	151	hashtest8	hashtest9@test.com	\$2b\$10\$pwAQ.OY3cy6QPLO1pRNGMeaOafJzmZm4D5NoxFQ0X7vm6otWgRSGC	2
	152	hashtest9	hashtest9@gmail.com	\$2b\$10\$WmSIE9B/uG2CxdwXJubUz.LQhDNytXeZVIhvdD4eWhS5BtPz3y/bu	2

user 1 ×

The above screenshot shows the original data inside the user table, and to show that the user "bob" (user_id: 102) is still in the table

The screenshot shows a browser window with a login form. In the 'email' input field, there is a SQL command: 'rita'; DELETE FROM file WHERE email = 'bob@designer.com'. The developer tools' console tab shows an error response from an axios request, with a red box highlighting the 'ER_BAD_FIELD_ERROR' message.

The screenshot above shows that the SQL statement was successfully ran in the database as it gives a SQL Error message in response. This shows that SQL injection was successful

Result Grid					Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
user_id	fullname	email	user_password	role_id				
100	rita	rita@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
101	robert	robert@admin.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	1				
103	braddy	braddy@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
104	josh	josh@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
105	john	john@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
106	fred	fred@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
107	ashley	ashley@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
108	amy	amy@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
109	anita	anita@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
110	eddy	eddy@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
111	larry	larry@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
112	ahtan	ahtan@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
113	joe	joe@admin.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
114	gabby	gabby@designer.com	\$2b\$10\$K.0Hwps0PDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6	2				
140	erwrrwer	erwrrwer@wewe.wewe	\$2b\$10\$csEQVzdRDGM4t.eOMUJ3NOqQqeecsOBPQ0zFr0RvJyV4jO3SAwcqqW	2				
141	Test	test@test.com	\$2b\$10\$oEodoR2DY4q4O6Jdb2syu.Fakyl33btZbkfVkr44dVZvuvCelW	2				
142	hashtest	hashtest@gmail.com	\$2b\$10\$sf.AHTWCsRJCS1N7AQs2E.fan0BOCK1Y2/UNYICsXSOfD9WxtMzC	2				
144	hashtest2	hashtest2@gmail.com	\$2b\$10\$4F/M6zw9Q93ir5AISC6wUekCYJktrDzDh4Weh7vmjHTbnVEk3R6cC	2				
145	hashtest3	hashtest3@gmail.com	\$2b\$10\$PbZ2dQlkxHUBapVmsjO.xeEDN/p40kzPQ5L5SoC38MPdH6O292Fc6	2				
146	hashtest4	hashtest4@test.com	\$2b\$10\$s0/4qiE4dcGWz.mOBpbhT.PEPWKt4rYw/A6So0bIpDxz6NWxy7kqi	2				
147	hashtest5	hashtest5@test.com	\$2b\$10\$fJ1H42rnNJT//ELALoq7Zegp6Kqgt/YDYmpbbhSiWYG3QCB1zdsae	2				

After editing the SQL statement, based on the response in the previous picture, the screenshot above shows that the SQL statement was successfully ran in the database and that SQL Injection was successful as the user "bob" (user_id: 102) was deleted from the table

About this interface: The administrator role can make changes to the role so that he can set another person as an

For updating of a user inside the /admin/update_user.html page, Changing the ID number inside the HTML document's hidden input, as shown above, will allow an SQL Injection to take place

```
s:46:28
      at Ping.onOperationComplete (D:\SP\AY21_22_Sem_1\ESDE\Assignment_1_original\experimentsecuritywithcompetitionsystem\node_modules\mysql\lib\pool.js:110:5)
      at Ping.<anonymous> (D:\SP\AY21_22_Sem_1\ESDE\Assignment_1_original\experimentsecuritywithcompetitionsystem\node_modules\mysql\lib\Connection.js:526:10)
      at Ping._onSocket (D:\SP\AY21_22_Sem_1\ESDE\Assignment_1_original\experimentsecuritywithcompetitionsystem\node_modules\mysql\lib\Connection.js:488:15)
      at Ping.Sequence.end (D:\SP\AY21_22_Sem_1\ESDE\Assignment_1_original\experimentsecuritywithcompetitionsystem\node_modules\mysql\lib\protocol\sequences\Sequence.js:83:24)
      at Ping.Sequence.OKPacket (D:\SP\AY21_22_Sem_1\ESDE\Assignment_1_original\experimentsecuritywithcompetitionsystem\node_modules\mysql\lib\protocol\sequences\Sequence.js:78:12)
      at Parser._parsePacket (D:\SP\AY21_22_Sem_1\ESDE\Assignment_1_original\experimentsecuritywithcompetitionsystem\node_modules\mysql\lib\protocol\Protocol.js:291:23)
      at Parser._parsePacket (D:\SP\AY21_22_Sem_1\ESDE\Assignment_1_original\experimentsecuritywithcompetitionsystem\node_modules\mysql\lib\protocol\Parser.js:433:10)
    code: 'ER_PARSE_ERROR',
    errno: 1064,
    sqlMessage: "You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' DELETE FROM file WHERE email = '' at line 1",
    sqlState: '42000',
    index: 0
sql: "DELETE user SET role_id = 1 WHERE user_id=101"; DELETE FROM file WHERE email = "
}
```

In the screenshot above, you could see that there is a SQL response from the input, this shows that the above HTML document's hidden input is susceptible to an SQL Injection

Preventing Future Vulnerabilities

To prevent future vulnerability, please make use of the parameterized query so that you can prevent any future vulnerabilities due to SQL Injection.

The codes below are still susceptible to SQL injection if left untouched and as such, our team has taken measures to prevent SQL injection from occurring.

Original Codes	Fixed Codes
backEnd > src > services > authService.js (module.exports.authenticate)	
<pre>module.exports.authenticate = (email, callback) => { pool.getConnection((err, connection) => { if (err) { if (err) throw err; } else { try { connection.query(`SELECT user.user_id, fullname, email, user_password, role_name, user.role_id FROM user INNER JOIN role ON user.role_id=role.role_id AND email='\${email}'`, {}, (err, rows) => { if (err) { if (err) return callback(err, null); } else { if (rows.length == 1) { console.log(rows); return callback(null, rows); } else { return callback('Login has failed', null); } } connection.release(); }); } catch (error) { return callback(error, null); } } }); //End of getConnection }); //End of authenticate</pre>	<pre>module.exports.authenticate = (email, callback) => { console.log("====="); console.log("authService.js > authenticate is called and running!"); console.log("====="); pool.getConnection((err, connection) => { if (err) { if (err) throw err; } else { try { connection.query(`SELECT user.user_id, fullname, email, user_password, role_name, user.role_id FROM user INNER JOIN role ON user.role_id=role.role_id AND email=?`, [email], (err, rows) => { if (err) { if (err) return callback(err, null); } else { if (rows.length == 1) { console.log(rows); return callback(null, rows); } else { return callback('Login has failed', null); } } connection.release(); }); } catch (error) { return callback(error, null); } } }); //End of getConnection }); //End of authenticate</pre>
backEnd > src > services > userService.js (module.exports.updateUser)	
<pre>module.exports.updateUser = (recordId, newRoleId) => { return new Promise((resolve, reject) => { //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection //to prepare the following code pattern which does not use callBack technique (uses Promise technique) pool.getConnection((err, connection) => { if (err) { console.log('Database connection error ', err); resolve(err); } else { connection.query(`UPDATE user SET role_id = \${newRoleId} WHERE user_id = \${recordId}`, (err, rows) => { if (err) { reject(err); } else { resolve(rows); } }); } }); }); //End of new Promise object creation }; //End of updateUser</pre>	<pre>module.exports.updateUser = (recordId, newRoleId) => { console.log("====="); console.log("userService.js > updateUser is called and running!"); console.log("====="); return new Promise((resolve, reject) => { //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection //to prepare the following code pattern which does not use callBack technique (uses Promise technique) pool.getConnection((err, connection) => { if (err) { console.log('Database connection error ', err); resolve(err); } else { connection.query(`UPDATE user SET role_id = ? WHERE user_id = ?`, [newRoleId, recordId], (err, rows) => { if (err) { reject(err); } else { resolve(rows); } }); } }); }); //End of new Promise object creation }; //End of updateUser</pre>

Original Codes	Fixed Codes
backEnd > src > services > userService.js (module.exports.getOneUserData)	
<pre>module.exports.getOneUserData = function (recordId) { console.log('getOneUserData method is called.'); console.log('Prepare query to fetch one user record'); userDataQuery = 'SELECT user_id, fullname, email, user.role_id, role_name FROM user INNER JOIN role ON user.role_id = role.role_id WHERE user_id=' + recordId; return new Promise((resolve, reject) => { //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection //to prepare the following code pattern which does not use callback technique (uses Promise technique) pool.getConnection((err, connection) => { if (err) { console.log('Database connection error ', err); resolve(err); } else { connection.query(userDataQuery, (err, results) => { if (err) { reject(err); } else { resolve(results); } connection.release(); }); } }); }); //End of new Promise object creation } //End of getOneUserData</pre>	<pre>module.exports.getOneUserData = function (recordId) { console.log('====='); console.log('userService.js > getOneUserData is called and running====='); console.log('====='); console.log('Prepare query to fetch one user record'); return new Promise((resolve, reject) => { //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection //to prepare the following code pattern which does not use callback technique (uses Promise technique) pool.getConnection((err, connection) => { if (err) { console.log('Database connection error ', err); resolve(err); } else { connection.query(userDataQuery, [recordId], (err, results) => { if (err) { reject(err); } else { // console.log("results: " + results[0].user_id); resolve(results); } connection.release(); }); } }); }); //End of new Promise object creation } //End of getOneUserData</pre>
backEnd > src > services > userService.js (module.exports.getOneUserDataByEmail)	
<pre>module.exports.getOneUserDataByEmail = function (search) { console.log('getOneUserDataByEmail method is called.'); console.log('Prepare query to fetch one user record'); userDataQuery = 'SELECT user_id, fullname, email, user.role_id, role_name FROM user INNER JOIN role ON user.role_id = role.role_id WHERE email=' + search + ''; return new Promise((resolve, reject) => { //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection //to prepare the following code pattern which does not use callback technique (uses Promise technique) pool.getConnection((err, connection) => { if (err) { console.log('Database connection error ', err); resolve(err); } else { connection.query(userDataQuery, (err, results) => { if (err) { reject(err); } else { resolve(results); } connection.release(); }); } }); }); //End of new Promise object creation } //End of getOneUserDataByEmail</pre>	<pre>module.exports.getOneUserDataByEmail = function (search) { console.log('====='); console.log('userService.js > getOneUserDataByEmail is called and running====='); console.log('====='); console.log('Prepare query to fetch one user record'); return new Promise((resolve, reject) => { //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection //to prepare the following code pattern which does not use callback technique (uses Promise technique) pool.getConnection((err, connection) => { if (err) { console.log('Database connection error ', err); resolve(err); } else { userDataQuery = 'SELECT user_id, fullname, email, user.role_id, role_name FROM user INNER JOIN role ON user.role_id = role.role_id WHERE email=' + search; connection.query(userDataQuery, [search], (err, results) => { if (err) { reject(err); } else { resolve(results); } connection.release(); }); } }); }); //End of new Promise object creation } //End of getOneUserDataByEmail</pre>
backEnd > src > services > userService.js (module.exports.getOneDesignData)	
<pre>module.exports.getOneDesignData = function (recordId) { console.log('getOneDesignData method is called.'); console.log('Prepare query to fetch one design record'); userDataQuery = 'SELECT file_id,cloudinary_file_id,cloudinary_url,design_title,design_description FROM file WHERE file_id=' + recordId; return new Promise((resolve, reject) => { //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection //to prepare the following code pattern which does not use callback technique (uses Promise technique) pool.getConnection((err, connection) => { if (err) { console.log('Database connection error ', err); resolve(err); } else { connection.query(userDataQuery, (err, results) => { if (err) { reject(err); } else { resolve(results); } connection.release(); }); } }); }); //End of new Promise object creation } //End of getOneDesignData</pre>	<pre>module.exports.getOneDesignData = function (recordId) { console.log('====='); console.log('userService.js > getOneDesignData is called and running====='); console.log('====='); console.log('Prepare query to fetch one design record'); var userDataQuery = 'SELECT file_id,cloudinary_file_id,cloudinary_url,design_title,design_description,created_by_id FROM file WHERE file_id=' + recordId; return new Promise((resolve, reject) => { //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection //to prepare the following code pattern which does not use callback technique (uses Promise technique) pool.getConnection((err, connection) => { if (err) { console.log('Database connection error ', err); resolve(err); } else { connection.query(userDataQuery, [recordId], (err, results) => { if (err) { reject(err); } else { resolve(results); } connection.release(); }); } }); }); //End of new Promise object creation } //End of getOneDesignData</pre>

Original Codes	Fixed Codes
backEnd > src > services > userService.js (module.exports.updateDesign)	
<pre>module.exports.updateDesign = (recordId, title, description) => { return new Promise((resolve, reject) => { //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection //to prepare the following code pattern which does not use callback technique (uses Promise technique) pool.getConnection((err, connection) => { if (err) { console.log('Database connection error ', err); reject(err); } else { connection.query(`UPDATE file SET design_title = ? , design_description=? WHERE file_id=?`, [title, description, recordId], (err, rows) => { if (err) { reject(err); } else { resolve(rows); } connection.release(); }); } }); }); }; //End of updateDesign</pre>	<pre>module.exports.updateDesign = (recordId, title, description) => { console.log("====="); console.log("userService.js > updateDesign is called and running!"); console.log("====="); return new Promise((resolve, reject) => { //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection //to prepare the following code pattern which does not use callback technique (uses Promise technique) pool.getConnection((err, connection) => { if (err) { console.log('Database connection error ', err); reject(err); } else { var sqlQuery = `UPDATE file SET design_title = ? , design_description = ? WHERE file_id = ?`; connection.query(sqlQuery, [title, description, recordId], (err, rows) => { if (err) { reject(err); } else { resolve(rows); } connection.release(); }); } }); }); }; //End of new Promise object creation</pre>
backEnd > src > services > fileService.js (module.exports.createFileData)	
<pre>module.exports.createFileData = (imageURL, publicId, userId, designTitle, designDescription) => { console.log('createFileData method is called.'); return new Promise((resolve, reject) => { //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection //to prepare the following code pattern which does not use callback technique (uses Promise technique) pool.getConnection((err, connection) => { if (err) { console.log('Database connection error ', err); reject(err); } else { console.log('Executing query'); let query = `INSERT INTO file (cloudinary_file_id, cloudinary_url, design_title, design_description, created_by_id) VALUES ('\${publicId}', '\${imageURL}', '\${designTitle}', '\${designDescription}', '\${userId}')`; connection.query(query, [], (err, rows) => { if (err) { console.log('Error on query on creating record inside file table', err); reject(err); } else { resolve(rows); } connection.release(); }); } }); }); }; //End of new Promise object creation</pre>	<pre>module.exports.createFileData = (imageURL, publicId, userId, designTitle, designDescription) => { console.log("====="); console.log("fileService.js > createFileData is called and running!"); console.log("====="); return new Promise((resolve, reject) => { //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection //to prepare the following code pattern which does not use callback technique (uses Promise technique) pool.getConnection((err, connection) => { if (err) { console.log('Database connection error ', err); reject(err); } else { console.log('Executing query'); let query = `INSERT INTO file (cloudinary_file_id, cloudinary_url, design_title, design_description, created_by_id) VALUES (?, ?, ?, ?, ?)`; connection.query(query, [publicId, imageURL, designTitle, designDescription, userId], (err, rows) => { if (err) { console.log('Error on query on creating record inside file table', err); reject(err); } else { resolve(rows); } connection.release(); }); } }); }); }; //End of new Promise object creation</pre>

Demo Screenshot (After)

- Logging In

The screenshot shows a login form with a blue header bar containing 'Home' and 'Login Register'. The main area has a light gray background with a centered 'Login' title. Below it is a 'Your email' field with the value 'rita@designer.com'; DELETE FROM file WHERE file_id = '100';--' and a 'Your password' field with the value '.....'. A blue 'SUBMIT' button is at the bottom.

• Public can register as a user

In our testing, we tried to delete the file with file_id of 100 from the database with the patched SQL injection codes

Result Grid						
file_id	cloudinary_file_id	cloudinary_url	design_title	design_description	created_by_id	
100	Design/hbukmdugiatdbmgobrdq	http://res.cloudinary.com/wongwj/image/upload/	rita design ...	rita design 1 description text 1 text 2 text 3 text 4	100	
101	Design/jecbaawpweki0ru0dq2c	http://res.cloudinary.com/wongwj/image/upload/	rita design 2	rita design 2 description text 1 text 2 text 3 tex...	100	
102	Design/bjwej9alijhcvzgdre	http://res.cloudinary.com/wongwj/image/upload/	rita design 3	rita design 3 description text 1 text 2 text 3 tex...	100	
103	Design/hinkgxzmduzieqabdmf	http://res.cloudinary.com/wongwj/image/upload/	rita design 4	rita design 4 description text 1 text 2 text 3 tex...	100	
104	Design/kwjkdf4rbfw1j4a1bj2	http://res.cloudinary.com/wongwj/image/upload/	rita design 5	rita design 5 description text 1 text 2 text 3 tex...	100	
105	Design/gttdb6ydsqeqlqaiansvp	http://res.cloudinary.com/wongwj/image/upload/	rita design 6	rita design 6 description text 1 text 2 text 3 tex...	100	
106	Design/wul3phulzmz6ekk8j68	http://res.cloudinary.com/wongwj/image/upload/	rita design 7	rita design 7 description text 1 text 2 text 3 tex...	100	
107	Design/dlihtck8cimougx7hw1	http://res.cloudinary.com/wongwj/image/upload/	rita design 8	rita design 8 description text 1 text 2 text 3 tex...	100	
108	Design/pb0r2sbqegtkxbv6gbvj	http://res.cloudinary.com/wongwj/image/upload/	rita design 9	rita design 9 description text 1 text 2 text 3 tex...	100	
109	Design/c096nxvfgg43waomrms	http://res.cloudinary.com/wongwj/image/upload/	rita design 10	rita design 10 description text 1 text 2 text 3 te...	100	
110	Design/jwxgo5fkjwdwdr3ma9t	http://res.cloudinary.com/wongwj/image/upload/	rita design 11	rita design 11 description text 1 text 2 text 3 te...	100	
111	Design/oxkkumijs08rvq3wv8on	http://res.cloudinary.com/wongwj/image/upload/	rita design 12	rita design 12 description text 1 text 2 text 3 te...	100	
112	Design/alyphhtik9soskk6cngo	http://res.cloudinary.com/wongwj/image/upload/	Bob Design 1	Bob Design Description 1	102	
115	Design/smuryoptiwbgraam2zs1	http://res.cloudinary.com/wongwj/image/upload/	Bob Design 2	Bob Design 2 Description	102	
117	Design/r5semil6krvlsztjo5osj	http://res.cloudinary.com/wongwj/image/upload/	Bob Design 3	Bob Design Description 3	102	
118	Design/ijojg07t2caosy6bfpb	http://res.cloudinary.com/wongwj/image/upload/	!	!	102	
119	Design/trivgbeghdurlx4m1sftn	http://res.cloudinary.com/wongwj/image/upload/	Bob Design 6	Details	102	
120	Design/mt69kbtvou9aztqzh2qa	http://res.cloudinary.com/wongwj/image/upload/	Braddy 1	BRaddy 1 D	103	
121	Design/zo7mpazjawscguqzq0cxe	http://res.cloudinary.com/wongwj/image/upload/	Braddy 2	BRaddy 2 D	103	

As seen in our database, our patched codes worked and file_id 100 still exists, preventing SQL injection in our login.html

Vulnerability 2 – Broken Authentication (Input Validation)

V-02 Broken Authenticati on (Input Validation)	Input Validation was not done for registration, therefore any user can register without an email or password.	Any user can go the registration page and sign up without any name, email or password entered in the text field.	Recommend using regular expression checks as front-end user input validation to ensure that user does not input any malicious JavaScript codes or empty fields. This also helps fix the SQL injection issue as mentioned in Vulnerability 1
	Impact – Medium	Ease of exploit - Easy	

Demo Screenshot (Before)

The screenshot shows a registration form with three input fields: 'Your name', 'Your email', and 'Your password'. Below the fields is a blue 'SUBMIT' button.

You have registered. Please [Login](#)

Sign up

Your name

Your email

Your password

SUBMIT

The screenshot above shows that users can register without any input.

113	joe	joe@admin.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg... 2
114	gabby	gabby@designer.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg... 2
121			\$2b\$10\$QLk8kffqSNaGQrNfRa.yTusM7HJVg6zZ... 2

As seen in the screenshot above, a new role is created in the user database where there is no name, email, or password in the database. The password will still be automatically created as the empty password is hashed.

Profile

BACK

The nameless user has created an account inside the database with no information and has normal user access.

frontEnd > public > js > register.js

```

1 let $registerFormContainer = $('#registerFormContainer');
2 if ($registerFormContainer.length != 0) {
3   console.log('Registration form detected. Binding event handling logic to form elements.');
4   //If the jQuery object which represents the form element exists,
5   //the following code will create a method to submit registration details
6   //to server-side api when the #submitButton element fires the click event.
7   $('#submitButton').on('click', function(event) {
8     event.preventDefault();
9     const baseUrl = 'http://localhost:5000';
10    let fullName = $('#fullNameInput').val();
11    let email = $('#emailInput').val();
12    let password = $('#passwordInput').val();
13    let webFormData = new FormData();
14    webFormData.append('fullName', fullName);
15    webFormData.append('email', email);
16    webFormData.append('password', password);
17    axios({
18      ...
19    })
20      .then(function(response) { ...
21    })
22      .catch(function(response) { ...
23    });
24  });
25 });
26 });
27 } //End of checking for $registerFormContainer jQuery object

```

The above codes extracted from the frontEnd: register.js shows that the input, once received, is not validated in client-side codes using JavaScript and is immediately sent to the Backend.

```

let $registerFormContainer = $('#registerFormContainer');
if ($registerFormContainer.length != 0) {
  console.log('Registration form detected. Binding event handling logic to form elements.');
  //the following code will create a method to submit registration details
  //to server-side api when the #submitButton element fires the click event.
  $('#submitButton').on('click', function(event) {
    event.preventDefault();
    const baseUrl = 'https://localhost:5000';
    let fullName = $('#fullNameInput').val();
    let email = $('#emailInput').val();
    let password = $('#passwordInput').val();

    let fullName_regex = new RegExp("[a-zA-Z0-9\\s]+");
    let email_regex = new RegExp("(^([a-zA-Z0-9]+[a-zA-Z0-9]*[a-zA-Z0-9]+)[@][a-zA-Z0-9]+([a-zA-Z0-9]{2,})$)");
    let pwd_regex = new RegExp("(^([a-zA-Z]{2,})(?=.*[0-9])(?=.*[a-zA-Z0-9]{8,})$)");

    // console.log(fullName_regex.test(fullName));
    // console.log(email_regex.test(email));
    // console.log(pwd_regex.test(password));

    if (fullName_regex.test(fullName) && pwd_regex.test(password) && email_regex.test(email)) {
      ...
    } else {
      console.log('test Failed');
      new Noty({
        title: 'Error',
        type: 'error',
        layout: 'topCenter',
        theme: 'sunset',
        text: 'Unable to register',
      }).show();
    }
  });
} //End of checking for $registerFormContainer jQuery object

```

To ensure that the input is validated, we included regex check for all 3 components of the input form: username, email as well as password.

This ensures that all 3 components of the input form are not being submitted with malicious codes that could potentially cause data leakage or harm to the database.

backEnd > src > controllers > authController.js (exports.processRegister)

```

exports.processRegister = (req, res, next) => {
  console.log('processRegister running');
  let fullName = req.body.fullName;
  let email = req.body.email;
  let password = req.body.password;

  bcrypt.hash(password, 10, async (err, hash) => {
    if (err) {
      console.log('Error on hashing password');
      return res.status(500).json({ statusMessage: 'Unable to complete registration' });
    } else {
      results = user.createUser(fullName, email, hash, function (results, error) {
        if (results != null) {
          console.log(results);
          return res.status(200).json({ statusMessage: 'Completed registration.' });
        }
        if (error) {
          console.log('processRegister method : callback error block section is running.');
          console.log(error, "=====");
          return res.status(500).json({ statusMessage: 'Unable to complete registration' });
        }
      }); //End of anonymous callback function
    }
  });
};

}; // End of processRegister

```

As you can see in the above screenshot, the values received are not checked in anyway, and immediately sent to database through the call of "user.createUser" to insert into the database.

This is a call for concern as the fullName may not be a nice input and email may not be email at all. Attackers could input scripts or malicious codes which will get stored inside the database and executed when it is being called. This leads to something called a Stored XSS attack.

```

exports.processRegister = (req, res, next) => {
  console.log('=====');
  console.log('authController.js - processRegister is called and running!');
  console.log('=====');

  let fullName = req.body.fullName;
  let email = req.body.email;
  let password = req.body.password;

  let fullName_regex = new RegExp("[a-zA-Z0-9\\s]+");
  let pwd_regex = new RegExp("(^([a-zA-Z]{2,})(?=.*[0-9])(?=.*[a-zA-Z0-9]{8,})$)");

  const saltRounds = 10;

  if (fullName_regex.test(fullName) && pwd_regex.test(password) && validator.isEmail(email)) {
  } else {
    console.log('regex check unsuccessful in backend');
    let data = {
      message: 'Unable to complete registration'
    }
    return res.status(500).json(data);
  }
}; // End of processRegister

```

Hence, to solve that issue, we make use of Regular Expressions (regex) to test the input based on the regex that we created to ensure that the input is validated before sending it to the database, otherwise it will be kicked out of the system.

Demo screenshot (After)

The screenshot shows a sign-up form on a website. At the top, there is a blue header bar with a logo, a "Home" button, and "Login" and "Register" links. Below the header is a red error message box containing the text "Unable to register.". The main form area has three input fields: "Your name" (with placeholder text "<script>alert('Hello')</script>"), "Your email" (with placeholder text "SSSSSSSSSS"), and "Your password" (with placeholder text "*****"). A blue "SUBMIT" button is located at the bottom right of the form.

After adding the regular expression check, the website no longer registers users if they have no input

```
=====
authController.js > processRegister is called and running!
=====
regex check unsuccesful in backend
```

The back endconsole displays that the regular expression check for user registration is not successful

This screenshot is identical to the one above, showing the same sign-up form with the same error message and invalid input. It serves as a visual confirmation that the regular expression check is now effective in preventing such inputs from being registered.

Additionally, if users use special characters in their name, email or password (e.g. < > () { } “ ”) they will not be able to make an account. (Excluding the usage of @ and . for email)

Vulnerability 3 – Logging and Monitoring

V-06 Logging and monitoring	<p>Insufficient logging throughout the system can cause difficulty when troubleshooting issues in the future. Logging can also help provide real time warnings for admin and developers.</p>	<p>One example is that hackers can simply use a brute force attack when they obtain the account's email to try and login with commonly used passwords.</p> <p>Without logging, the hacking attempt will be unknown to the administrator, and they cannot warn the user that their account has been compromised</p>	<p>Recommend to log failed login attempts as well as display on console so that suspicious activity can be monitored as well as logged and account can be locked to prevent brute force attacks.</p> <p>Automatically locking out accounts after certain number of failed attempts to block brute force attacks would help but also could be ineffective as hackers can simply bypass the lockout by attempting to login before the lockout occurs.</p> <p>Other backend errors should be logged as well to help future administrators and programmers troubleshoot easily when a future issue arises.</p>
Impact - Low	Ease of exploit - Easy		

Fixed Codes

backEnd > src > controllers > authController.js
(exports.processRegister) ← e.g.

```
34 |     const loggerformat = printf(({ level, message, timestamp }) => {
35 |         return `${timestamp} ${level}: ${message}`;
36 |     });
37 |     const logger = createLogger({
38 |         format: format.combine(
39 |             format.timestamp({ format: 'YYYY-MM-DD HH:mm:ss' }),
40 |             loggerformat
41 |         ),
42 |         transports: [
43 |             new transports.Console(),
44 |             new transports.File({ filename: './Logs/FailedLogins.log', level: 'warn' }),
45 |         ],
46 |     });
47 |     logger.warn('Login for ' + email + ' has failed!\n IP for ' + email + ' login: ' + req.ip)
48 |     let data = {
49 |         message: 'Credentials are not valid'
50 |     };
51 |     return res.status(500).json(data);
52 | }
```

In our codes, we have installed the module Winston to help us in our logging. It displays the warning in the console and furthermore logs it in a sperate folder called Logs

Demo screenshot (After)

```
2021-06-27 13:39:26 warn: Login for bob@designer.com has failed!
IP for bob@designer.com login: ::1
```

In the console, there is a warning for failed logins for administrators to monitor from winston

 FailedLogins.log - Notepad

File Edit Format View Help

```
2021-06-27 13:39:26 warn: Login for bob@designer.com has failed!
IP for bob@designer.com login: ::1
```

In this case, the time of failed login and email has been logged in a file called FailedLogs.log. The IP for the failed login is shown as ::1 because it is being ran in localhost.

Vulnerability 4 – Sanitize JSON Results

V-03 Sensitive data exposure (Sanitize JSON response)	SQL Error messages are directly printed for the Frontend users to see, which can be a vulnerability as this allows the attacker to make use of the SQL errors to craft their responses	When an error is made logging into the website, the google chrome console displays the error which attackers can fine tune their SQL injection statements based on the error messages that are getting sent to the user.	Recommend to change all the JSON response to return customised error messages rather than printing out the SQL error directly to the user. This is to prevent the attacker who has strong knowledge on SQL statements from relying heavily on SQL return statements to craft their next attack and send it back through the endpoints.
Impact - Low			

Demo screenshot (Before)

The screenshot shows a web browser window with a login form. The URL is `localhost:3001/login.html`. The developer tools console is open, specifically the 'Logs' tab, which displays the following error message:

```

columnNumber: 6414
> config: Object { url: "http://localhost:5000/api/user/login", method: "post", timeout: 0, ... }
  file: "http://localhost:3001/js/axios.js"
  isAxiosError: true
  lineNumber: 8
  message: "Request failed with status code 500"
  request: XMLHttpRequest { readyState: 4, timeout: 0, ... }
  withCredentials: false, ...
  response: Object { data: {}, status: 500, statusText: "Internal Server Error", ... }
  > config: Object { url: "http://localhost:5000/api/user/login", method: "post", timeout: 0, ... }
    file: "http://localhost:3001/js/axios.js"
    isAxiosError: true
    lineNumber: 8
    message: Object { code: "ER_BAD_FIELD_ERROR", ... }
    errno: 1054, sqlMessage: "Unknown column 'email' in 'where clause'", ...
    > __proto__: Object { ... }
  > headers: Object { "content-length": "369", "content-type": "application/json; charset=utf-8" }
  > request: XMLHttpRequest { readyState: 4, timeout: 0, ... }
  withCredentials: false, ...
  status: 500
  statusText: "Internal Server Error"
  > __proto__: Object { ... }

```

Original Codes	Fixed Codes
backEnd > src > controllers > authController.js (exports.processRegister)	
<pre>exports.processLogin = (req, res, next) => { let email = req.body.email; let password = req.body.password; try { auth.authenticate(email, function (error, results) { if (error) { let message = 'Credentials are not valid.'; //return res.status(500).json({ message: message }); //If the following statement replaces the above statement //to return a JSON response to the client, the SQLMap or //any attacker (who relies on the error) will be very happy //because they relies a lot on SQL error for designing how to do //attack and anticipate how much "rewards" after the effort. //Rewards such as sabotage (seriously damage the data in database), //data theft (grab and sell). return res.status(500).json({ message: error }); } else { if (results.length == 1) { if ((password == null) (results[0] == null)) { return res.status(500).json({ message: 'login failed' }); } if (bcrypt.compareSync(password, results[0].user_password) == true) { let data = { user_id: results[0].user_id, role_name: results[0].role_name, token: jwt.sign({ id: results[0].user_id }, config.JWTKey, { expiresIn: 86400 //Expires in 24 hrs }) }; //End of data variable setup return res.status(200).json(data); } else { // return res.status(500).json({ message: 'Login has failed.' }); return res.status(500).json({ message: error }); } //End of password comparison with the retrieved decoded password. } //End of checking if there are returned SQL results } }) } catch (error) { return res.status(500).json({ message: error }); } //end of try }; </pre>	<pre>try { auth.authenticate(email, function (error, results) { if (error) { let message = 'Credentials are not valid.'; let data = { message: 'Credentials are not valid' }; return res.status(500).json(data); } else { if (results.length == 1) { if ((password == null) (results[0] == null)) { let jsonResult = { message: 'Login Failed' }; return res.status(500).json(jsonResult); } if (bcrypt.compareSync(password, results[0].user_password) == true) { let data = { user_id: results[0].user_id, role_name: results[0].role_name, token: jwt.sign({ id: results[0].user_id, uRole: results[0].role_name }, config.JWTKey, { expiresIn: 7200 //Expires in 2 hrs }) }; //End of data variable setup return res.status(200).json(data); } else { let data = { message: 'Login has failed.' }; return res.status(500).json(data); } //End of password comparison with the retrieved decoded password. } //End of checking if there are returned SQL results } }) } catch (error) { console.log('error: ' + error); let data = { message: 'An error has occurred!' }; return res.status(500).json(data); } //end of try }; </pre>

In the above codes, you can see that when there is an error of when the login has failed, the SQL error is immediately packaged into a JSON object and sent back to the client (frontEnd).

This leads to a potential sensitive data exposure as if the endpoint called is to retrieve user data, it will display the data of that user.

To prevent or minimize the risk of attackers getting the SQL statement responses, we decided to package a custom error message to send to the client so that they will only see what we want them to see while the SQL statement responses is printed in the console in the backEnd server.

Original Codes	Fixed Codes
backEnd > src > controllers > userController.js (exports.processGetUserData)	
<pre> exports.processGetUserData = async(req, res, next) => { let pageNumber = req.params.pagenumber; let search = req.params.search; try { ... } catch (error) { let message = 'Server is unable to process your request.'; return res.status(500).json({ message: error }); } }; //End of processGetUserData </pre>	<pre> exports.processGetUserData = async (req, res, next) => { console.log("=====UserController.js > processGetUserData is called and running====="); let pageNumber = req.params.pagenumber; let search = req.params.search; let userId = req.body.userId; //req.body.userId is retrieved from the headers as done in the previous middleware function (checkUserFn.js) let decodedId = req.decodedId; let decodedRole = req.decodedRole; console.log("userId: " + userId); console.log("decodedId: " + decodedId); if (userId != decodedId) { ... if (userId == decodedId && decodedRole == "admin") { try { } catch (error) { console.log("Error Occurred: " + error); let jsonResult = { message: 'Server is unable to process your request' } return res.status(500).json(jsonResult); } else { } } } }; //End of processGetUserData </pre>

The above code sends the error that is received from the server directly to the client, which is not good as it leads to potential sensitive data leakage. Moreover, there is a message statement declared, but it was never read/sent to the client, hence the message variable declared was useless

The above codes address the issues mentioned and minimizes the risk of the user getting the data.

The jsonResult object will send back the customized message to the client so that the client is well aware of the current state of their request.

Demo screenshot (After)

The screenshot shows a browser window with a login form. The form has fields for 'Your email' (test@wewe.wewe) and 'Your password' (redacted). Below the form is a 'SUBMIT' button. To the right of the browser is the Chrome developer tools Network tab. A POST request to 'https://localhost:5000/api/user/login' is selected, showing a status of 500 (Internal Server Error). The response body is a JSON object with the message 'Credentials are not valid'.

- Public can register as a user
- The database has been **seeded** with user test data.
- user:rita@designer.com password:password normal user role

As seen in the google chrome console, the JSON result has been sanatized

Vulnerability 5 – Broken Authentication (Restricted Pages Access)

V-02 Broken Authenticati on (Restricted Pages Access)	<p>Users can access administrator only pages by changing the URL.</p> <p>To ensure that the request can be made when the logged in user is admin, we must send authorization headers to the Backend to be successfully authorized.</p>	<p>The URL can simply be changed to an administrator only page and can be accessed by all users. In this case as the URL of the pages are easy to guess, the hacker can change to /admin/manage_user.html. This allows hackers to access administrator only pages with a normal user account and perform administrator only.</p>	<p>Recommend validating the user's jwt token and user id to lock unauthorized users out of administrator pages using a middleware function to ensure that privileges can only be performed by an administrator and not by any user.</p>
	Impact – High	Ease of exploit - Easy	

Demo screenshots (Before)

localhost:3001/user/manage_submission.html

Manage my submissions

Search de... **SEARCH**

Application	
Manifest	Key
Service Worker	Value
Storage	token eyJhbGciOiJI
Storage	
Local Storage	user_id 100
Session Storage	role_name user
IndexedDB	
Web SQL	
Cookies	
Trust Token	
Cache	
Cache Storage	
Application Cache	
Background Services	
Background Fetch	
Background Sync	
Notification	
Payment Handler	
Periodic Background Task	
Push Message	

Currently, the user is logged in as rita@designer.com as seen in the localStorage on the right

localhost:3001/admin/manage_users.html

Manage users

Search by ... bob **SEARCH**

bob

MANAGE

bob@designer.com

Application	
Manifest	Key
Service Worker	Value
Storage	token eyJhbG
Storage	
Local Storage	user_id 100
Session Storage	role_name user
IndexedDB	
Web SQL	
Cookies	
Trust Token	
Cache	
Cache Storage	
Application Cache	
Background Services	
Background Fetch	
Background Sync	
Notification	
Payment Handler	
Periodic Background Task	
Push Message	

By simply changing the URL to /admin/manage_users.html as seen above, rita even though a normal user is able to access administrator pages as there is no authentication being made.

Added a script for the following codes that will run everytime the webpage is called. This script will send the token to the endpoint in the backEnd to check if the user has the proper authentication and authorization for that page. If not, the user is directed out of the page to the login page, indicating that the user does not have access to that particular page

Original Codes	Fixed Codes
<pre>public > admin > o profile.html > ... 1 [!DOCTYPE html] 2 <html lang="en"> 3 4 <head> 5 <title>B.D.S</title> 6 <!-- Font Awesome--> 7 <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.2/css/all.css"> 8 9 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&amp;display=sans"> 10 <!-- Bootstrap core CSS--> 11 <link href="https://cdn.jsdelivr.net/npm/twbs/bootstrap@4.5.0/dist/css/bootstrap.min.css" rel="stylesheet"> 12 <!-- Material Design Bootstrap--> 13 <link href="https://cdn.jsdelivr.net/npm/mdb-ui-kit@3.1.4/dist/mdb.min.css" rel="stylesheet"> 14 <!-- Noty CSS--> 15 <link href="https://cdn.jsdelivr.net/npm/noty@3.1.4/noty.css" rel="stylesheet"> 16 <!-- Noty CSS Map--> 17 <link href="https://cdn.jsdelivr.net/npm/noty@3.1.4/noty.css.map" rel="stylesheet"> 18 <!-- Noty CSS Map--> 19 <link href="https://cdn.jsdelivr.net/npm/noty@3.1.4/noty.css.map" rel="stylesheet"> 20 21 </head> 22 23 > <body> 24 <div class="container-fluid">... 25 </div> 26 </div> 27 </div> 28 29 <!-- jQuery--> 30 <script type="text/javascript" src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.min.js"></script> 31 <!-- Bootstrap tooltips--> 32 <script type="text/javascript" src="https://cdn.jsdelivr.net/npm/popper.js@1.14.4/dist/umd/popper.min.js"></script> 33 <!-- Bootstrap core JavaScript--> 34 <script type="text/javascript" src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.0/dist/js/bootstrap.min.js"></script> 35 <!-- MDB core JavaScript--> 36 <script type="text/javascript" src="https://cdn.jsdelivr.net/npm/mdb-ui-kit@3.1.4/dist/mdb.min.js"></script> 37 38 <!-- Web App JS--> 39 <script src="js/axios.js"></script> 40 <script src="js/profile.js"></script> 41 <script src="js/global.js"></script> 42 43 </body> 44 45 </html></pre>	<pre>public > user > o profile.html > ... 30 <script type="text/javascript" 31 src="https://cdnjs.cloudflare.com/ajax/libs/mdbootstrap/4.19.1/js/mdb.min.js"></script> 32 <!-- Noty JS--> 33 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.min.js"></script> 34 <!-- Web App JS--> 35 <script src="js/axios.js"></script> 36 37 38 const token = sessionStorage.getItem("token"); 39 const role = sessionStorage.getItem("role_name"); 40 const loggedInUserID = parseInt(sessionStorage.getItem("user_id")); 41 const baseURL = "https://localhost:5000"; 42 43 // ----- 44 // Checking if the user has logged in or not 45 // ----- 46 if (token === null !isValidLoggedInUserID) { 47 console.log("no token so outta here"); 48 window.location.href = "/login.html"; 49 } else { 50 // ----- 51 // Backend call to check token validity & authorisation 52 // ----- 53 axios({ 54 method: 'post', 55 url: baseURL + '/api/user/authentication-check', 56 //data: webFormData, 57 headers: { 58 authorization: "Bearer " + token, 59 "user": loggedInUserID 60 } 61 }) 62 .then(function (response) { 63 if (response.data.userRole == "user") { 64 console.log("Access Granted!"); 65 } else { 66 console.log("Access Denied!"); 67 sessionStorage.clear(); 68 window.location.href = "/home.html"; 69 } 70 }) 71 .catch(function (response) { 72 console.log("Error!"); 73 console.log(response); 74 75 if (response.response.status == 403) { 76 sessionStorage.clear(); 77 window.location.href = "/login.html"; 78 } else { 79 window.location.href = "/home.html"; 80 } 81 }); 82 } 83 </script> 84 </head> 85 86 <body> 87 <div class="container-fluid">... 88 </div> 89</pre>

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html>
3  <html lang="en">
4
5      <head>
6          <title>B.O.S</title>
7          <!-- Font Awesome -->
8          <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.2/css/all.css">
9          <!-- Google Fonts -->
10         <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Emily+Robot:300,400,300,400&display=swap">
11         <!-- Material Design Bootstrap -->
12         <link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.0/css/bootstrap.min.css" rel="stylesheet">
13         <!-- Material Design Bootstrap -->
14         <link href="https://cdnjs.cloudflare.com/ajax/libs/mdbbootstrap/4.19.1/css/mdb.min.css" rel="stylesheet">
15         <!-- Noty CSS -->
16         <link href="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.css" rel="stylesheet">
17         <!-- Noty CSS Map -->
18         <link href="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.css.map" rel="stylesheet">
19         <!-- Noty CSS Map -->
20         <link href="/css/site.css" rel="stylesheet">
21     </head>
22
23     <body>
24         <div class="container-fluid">...
25         </div>
26
27         <!-- jQuery -->
28         <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
29
30         <!-- Bootstrap tooltips -->
31         <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.min.js"></script>
32         <!-- Twitter Bootstrap -->
33         <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.0/js/bootstrap.min.js"></script>
34
35         <!-- MDB Core JavaScript -->
36         <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mdbbootstrap/4.19.1/js/mdb.min.js"></script>
37
38         <!-- Noty App JS -->
39         <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.min.js"></script>
40
41         <script src="/js/saxion.js"></script>
42         <script src="https://s3.us-east-1.amazonaws.com/check_user_submission.js"></script>
43         <script src="/js/global.js"></script>
44
45     </body>
46
47     </html>
```

```
public > admin > check_user_submission.html > html > head
37 <script>
38
39     const token = sessionStorage.getItem("token");
40     const role = sessionStorage.getItem("role_name");
41     const loggedInUserID = parseInt(sessionStorage.getItem("user_id"));
42     const baseUrl = "https://localhost:5000";
43
44
45     // -----
46     // Checking if the user has logged in or not
47     // -----
48     if (token === null || isNaN(loggedInUserID)) {
49         console.log("no token so gotta here");
50         window.location.href = "/login.html";
51     } else {
52         // -----
53         // Backend call to check token validity & authorisation
54         // -----
55         axios({
56             method: 'post',
57             url: baseUrl + '/api/user/authentication-check',
58             //data: webformData,
59             headers: {
60                 authorization: "Bearer " + token,
61                 'user': loggedInUserID
62             }
63         })
64         .then(function (response) {
65             if (response.data.userRole == "admin") {
66                 console.log("Access Granted!");
67             } else {
68                 console.log("Access Denied!");
69                 sessionStorage.clear();
70                 window.location.href = "/home.html";
71             }
72         })
73         .catch(function (response) {
74             console.log("Error!!");
75             console.log(response);
76
77             if (response.response.status == 403) {
78                 sessionStorage.clear();
79                 window.location.href = "/login.html";
80             } else {
81                 window.location.href = "/home.html";
82             }
83         });
84     }
85 </script>
86
87 </head>
88 > <body>...
89 </body>
90 </body>
91
92 </html>
```

frontEnd > admin > manage_users.html

```
public admin > $ ./manage_users.html ...  
[1] [DOCTYPE html]  
[2] [html lang="en"]  
[3] [head]  
[4]   [title]8.0.S</title]  
[5]   [<!-- Font Awesome -->  
[6]     [link rel="stylesheet" href="https://use.typekit.net/awesomeness.css"/>  
[7]     [link rel="stylesheet" href="https://use.typekit.net/awesomeness.css"/>  
[8]     [link rel="stylesheet" href="https://use.typekit.net/awesomeness.css"/>  
[9]     [link rel="stylesheet" href="https://use.typekit.net/awesomeness.css"/>  
[10]    [<!-- Bootstrap core CSS -->  
[11]      [link href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.0/dist/css/bootstrap.min.css" rel="stylesheet"]  
[12]      [link href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.0/dist/css/bootstrap.min.css" rel="stylesheet"]  
[13]      [link href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.0/dist/css/bootstrap.min.css" rel="stylesheet"]  
[14]      [<!-- Material Design Bootstrap -->  
[15]        [link href="https://cdn.jsdelivr.net/npm/mdbreactor@3.1.4/dist/mdbreactor.min.css" rel="stylesheet"]  
[16]        [link href="https://cdn.jsdelivr.net/npm/mdbreactor@3.1.4/dist/mdbreactor.min.css" rel="stylesheet"]  
[17]        [link href="https://cdn.jsdelivr.net/npm/mdbreactor@3.1.4/dist/mdbreactor.min.css" rel="stylesheet"]  
[18]        [<!-- Noty CSS -->  
[19]          [link href="https://cdn.jsdelivr.net/npm/noty@3.1.4/noty.css" rel="stylesheet"]  
[20]          [link href="https://cdn.jsdelivr.net/npm/noty@3.1.4/noty.css.map" rel="stylesheet"]  
[21]          [<!-- Site CSS -->  
[22]            [link href="https://use.typekit.net/awesomeness.css" rel="stylesheet"]  
[23]          ]</head>  
[24] <body>  
[25]   [<div class="container-fluid"> ...  
[26]   [</div>]  
[27]   [<!-- Jquery -->  
[28]     [script type="text/javascript" src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.min.js"]</script>  
[29]     [script type="text/javascript" src="https://cdn.jsdelivr.net/npm/popper.js@1.14.4/dist/umd/popper.min.js"]</script>  
[30]     [<!-- Bootstrap core JavaScript -->  
[31]       [script type="text/javascript" src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.0/dist/js/bootstrap.min.js"]</script>  
[32]       [script type="text/javascript" src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.0/dist/js/bootstrap.min.js"]</script>  
[33]       [script type="text/javascript" src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.0/dist/js/bootstrap.min.js"]</script>  
[34]       [<!-- MDB core JavaScript -->  
[35]         [script type="text/javascript" src="https://cdn.jsdelivr.net/npm/mdbreactor@3.1.4/dist/mdbreactor.min.js"]</script>  
[36]         [script type="text/javascript" src="https://cdn.jsdelivr.net/npm/mdbreactor@3.1.4/dist/mdbreactor.min.js"]</script>  
[37]         [script type="text/javascript" src="https://cdn.jsdelivr.net/npm/mdbreactor@3.1.4/dist/mdbreactor.min.js"]</script>  
[38]         [<!-- Noty JS -->  
[39]           [script src="https://cdn.jsdelivr.net/npm/noty@3.1.4/noty.js"]</script>  
[40]           [script src="https://cdn.jsdelivr.net/npm/noty@3.1.4/noty.js.map"]</script>  
[41]           [<!-- Global JS -->  
[42]             [script src="https://cdn.jsdelivr.net/npm/mdbreactor@3.1.4/dist/mdbreactor.min.js"]</script>  
[43]             [script src="https://cdn.jsdelivr.net/npm/mdbreactor@3.1.4/dist/mdbreactor.min.js"]</script>  
[44]             [script src="https://cdn.jsdelivr.net/npm/mdbreactor@3.1.4/dist/mdbreactor.min.js"]</script>  
[45]           ]</script>  
[46]         ]</script>  
[47]       ]</script>  
[48]     ]</script>  
[49]   ]</script>  
[50] ]</body>  
[51] </html>
```

```
36 </head>
37
38 <script>
39
40     const token = sessionStorage.getItem("token");
41     const role = sessionStorage.getItem("role_name");
42     const loggedInUserID = parseInt(sessionStorage.getItem("user_id"));
43     const baseUrl = "https://localhost:5000";
44
45     // -----
46     // Checking if the user has logged in or not
47     // -----
48     if (token === null || isNaN(loggedInUserID)) {
49         console.log("no token so outta here!");
50         window.location.href = "/login.html";
51     } else {
52         // -----
53         // Backend call to check token validity & authorisation
54         // -----
55         axios({
56             method: 'post',
57             url: baseUrl + '/api/user/authentication-check',
58             //data: webFormData,
59             headers: {
60                 authorization: "Bearer " + token,
61                 'user': loggedInUserID
62             }
63         })
64             .then(function (response) {
65                 if (response.data.userRole === "admin") {
66                     console.log("Access Granted!");
67                 } else {
68                     console.log("Access Denied!");
69                     sessionStorage.clear();
70                     window.location.href = "/home.html";
71                 }
72             })
73             .catch(function (response) {
74                 console.log("Error!");
75                 console.log(response);
76                 console.log(response);
77                 if (response.response.status === 403) {
78                     sessionStorage.clear();
79                     window.location.href = "/login.html";
80                 } else {
81                     window.location.href = "/home.html";
82                 }
83             });
84         }
85     });
86 </script>
87
88 > <body> ...
89 </body>
90
91 </html>
```

Original Codes	Fixed Codes
frontEnd > admin > update_user.html	
<pre>public > admin > update_user.html ... [DOCTYPE html] 1 <!DOCTYPE html> 2 <html lang="en"> 3 4 <head> 5 <title>B.O.S</title> 6 <!-- Font Awesome--> 7 <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.0.2/css/all.css"> 8 <!-- Google Fonts--> 9 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap"> 10 <!-- Bootstrap core CSS--> 11 <link href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/4.5.0/css/bootstrap.min.css" rel="stylesheet"> 12 <!-- MDB core CSS--> 13 <link href="https://cdnjs.cloudflare.com/ajax/libs/mdbbootstrap/4.19.1/css/mdb.min.css" rel="stylesheet"> 14 <!-- Noty CSS--> 15 <link href="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.css" rel="stylesheet"> 16 <!-- Noty CSS Map--> 17 <link href="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.css.map" rel="stylesheet"> 18 <!-- Noty JS--> 19 <link href="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.min.js" rel="stylesheet"> 20 </head> 21 22 <body> 23 <div class="container-fluid">... 24 </div> 25 </body> 26 </html></pre>	<pre>public > admin > update_user.html > HTML > head 35 36 37 <script> 38 39 const token = sessionStorage.getItem("token"); 40 const role = sessionStorage.getItem("role_name"); 41 const loggedInUserID = parseInt(sessionStorage.getItem("user_id")); 42 const baseUrl = "https://localhost:5000"; 43 44 45 // ----- 46 // Checking if the user has logged in or not 47 48 if (token === null isNaN(loggedInUserID)) { 49 console.log("no token so outta here"); 50 window.location.href = "/login.html"; 51 } else { 52 53 // ----- 54 // Backend call to check token validity & authorisation 55 56 axios({ 57 method: "post", 58 url: baseUrl + '/api/user/authentication-check', 59 headers: { 60 authorization: "Bearer " + token, 61 'user': loggedInUserID 62 } 63 }) 64 .then(function (response) { 65 if (response.data.userRole == "admin") { 66 console.log("Access Granted!"); 67 } else { 68 console.log("Access Denied!"); 69 sessionStorage.clear(); 70 window.location.href = "/home.html"; 71 } 72 }) 73 .catch(function (response) { 74 console.log("Error!!!"); 75 console.log(response); 76 77 if (response.response.status == 403) { 78 sessionStorage.clear(); 79 window.location.href = "/login.html"; 80 } else { 81 window.location.href = "/home.html"; 82 } 83 }); 84 </script> 85 86 </head>... 87 </body> 88 </html></pre>
frontEnd > user > manage_invite.html	
<pre>public > user > manage_invite.html ... 1 [DOCTYPE html] 2 <!DOCTYPE html> 3 4 <head> 5 <title>B.O.S</title> 6 <!-- Font Awesome--> 7 <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.0.2/css/all.css"> 8 <!-- Google Fonts--> 9 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap"> 10 <!-- MDB core CSS--> 11 <link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.0/css/bootstrap.min.css" rel="stylesheet"> 12 <!-- Material Design Bootstrap--> 13 <link href="https://cdnjs.cloudflare.com/ajax/libs/mdbbootstrap/4.19.1/css/mdb.min.css" rel="stylesheet"> 14 <!-- Noty CSS--> 15 <link href="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.css" rel="stylesheet"> 16 <!-- Noty CSS Map--> 17 <link href="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.css.map" rel="stylesheet"> 18 <!-- Noty JS--> 19 <link href="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.min.js" rel="stylesheet"> 20 </head> 21 22 <body> 23 <div class="container-fluid">... 24 </div> 25 </body> 26 </html></pre>	<pre>public > user > manage_invite.html > HTML > head 39 40 41 <script> 42 const token: string 43 const token = sessionStorage.getItem("token"); 44 const role = sessionStorage.getItem("role_name"); 45 const loggedInUserID = parseInt(sessionStorage.getItem("user_id")); 46 const baseUrl = "https://localhost:5000"; 47 48 49 // ----- 50 // Checking if the user has logged in or not 51 52 if (token === null isNaN(loggedInUserID)) { 53 console.log("no token so outta here"); 54 window.location.href = "/login.html"; 55 } else { 56 57 // ----- 58 // Backend call to check token validity & authorisation 59 60 axios({ 61 method: "post", 62 url: baseUrl + '/api/user/authentication-check', 63 headers: { 64 authorization: "Bearer " + token, 65 'user': loggedInUserID 66 } 67 }) 68 .then(function (response) { 69 if (response.data.userRole == "user") { 70 console.log("Access Granted!"); 71 } else { 72 console.log("Access Denied!"); 73 sessionStorage.clear(); 74 window.location.href = "/home.html"; 75 } 76 }) 77 .catch(function (response) { 78 console.log("Error!!!"); 79 console.log(response); 80 81 if (response.response.status == 403) { 82 sessionStorage.clear(); 83 window.location.href = "/login.html"; 84 } else { 85 window.location.href = "/home.html"; 86 } 87 }); 88 </script> 89 90 </head>... 91 </body> 92 </html></pre>

Original Codes	Fixed Codes
frontEnd > user > manage_submission.html	
<pre>public > user > manage_submission.html > ... 1 <!DOCTYPE html> 2 <html lang="en"> 3 4 <head> 5 <title>4.0.5</title> 6 <!-- Font Awesome --> 7 <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.2/css/all.css"> 8 <!-- Google Fonts--> 9 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap"> 10 <!-- Bootstrap core CSS--> 11 <link href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.0/dist/css/bootstrap.min.css" rel="stylesheet"> 12 <!-- Material Design Bootstrap--> 13 <link href="https://cdn.jsdelivr.net/npm/mdbbootstrap@4.19.1/css/mdb.min.css" rel="stylesheet"> 14 <!-- Noty CSS--> 15 <link href="https://cdn.jsdelivr.net/npm/noty@3.1.4/noty.css" rel="stylesheet"> 16 <!-- Noty CSS Map--> 17 <link href="https://cdn.jsdelivr.net/npm/noty@3.1.4/noty.css.map" rel="stylesheet"> 18 <!-- Noty CSS Map--> 19 <link href="/css/site.css" rel="stylesheet"> 20 21 </head> 22 <body> 23 <div class="container-fluid">... 24 <div> 25 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script> 26 <!-- Bootstrap tooltips--> 27 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.4/umd/popper.min.js"></script> 28 <!-- Bootstrap core JavaScript--> 29 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.0/js/bootstrap.min.js"></script> 30 <!-- MDB core JavaScript--> 31 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mdbbootstrap/4.19.1/js/mdb.min.js"></script> 32 <!-- Noty JS--> 33 <script type="text/javascript" src="https://cdn.jsdelivr.net/npm/noty@3.1.4/noty.min.js"></script> 34 <!-- Web App JS--> 35 <script src="/js/exicon.js"></script> 36 <script src="/js/user_manage_submission.js"></script> 37 <script src="/js/global.js"></script> 38 </div> 39 </body> 40 </html></pre>	<pre>public > user > manage_submission.html > html > head 40 <script> 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 259 260 261 262 263 264 265 266 267 268 269 269 270 271 272 273 274 275 276 277 278 279 279 280 281 282 283 284 285 286 287 287 288 289 289 290 291 292 293 294 295 296 297 297 298 299 299 300 301 302 303 304 305 306 307 307 308 309 309 310 311 312 313 313 314 315 315 316 316 317 317 318 318 319 319 320 320 321 321 322 322 323 323 324 324 325 325 326 326 327 327 328 328 329 329 330 330 331 331 332 332 333 333 334 334 335 335 336 336 337 337 338 338 339 339 340 340 341 341 342 342 343 343 344 344 345 345 346 346 347 347 348 348 349 349 350 350 351 351 352 352 353 353 354 354 355 355 356 356 357 357 358 358 359 359 360 360 361 361 362 362 363 363 364 364 365 365 366 366 367 367 368 368 369 369 370 370 371 371 372 372 373 373 374 374 375 375 376 376 377 377 378 378 379 379 380 380 381 381 382 382 383 383 384 384 385 385 386 386 387 387 388 388 389 389 390 390 391 391 392 392 393 393 394 394 395 395 396 396 397 397 398 398 399 399 400 400 401 401 402 402 403 403 404 404 405 405 406 406 407 407 408 408 409 409 410 410 411 411 412 412 413 413 414 414 415 415 416 416 417 417 418 418 419 419 420 420 421 421 422 422 423 423 424 424 425 425 426 426 427 427 428 428 429 429 430 430 431 431 432 432 433 433 434 434 435 435 436 436 437 437 438 438 439 439 440 440 441 441 442 442 443 443 444 444 445 445 446 446 447 447 448 448 449 449 450 450 451 451 452 452 453 453 454 454 455 455 456 456 457 457 458 458 459 459 460 460 461 461 462 462 463 463 464 464 465 465 466 466 467 467 468 468 469 469 470 470 471 471 472 472 473 473 474 474 475 475 476 476 477 477 478 478 479 479 480 480 481 481 482 482 483 483 484 484 485 485 486 486 487 487 488 488 489 489 490 490 491 491 492 492 493 493 494 494 495 495 496 496 497 497 498 498 499 499 500 500 501 501 502 502 503 503 504 504 505 505 506 506 507 507 508 508 509 509 510 510 511 511 512 512 513 513 514 514 515 515 516 516 517 517 518 518 519 519 520 520 521 521 522 522 523 523 524 524 525 525 526 526 527 527 528 528 529 529 530 530 531 531 532 532 533 533 534 534 535 535 536 536 537 537 538 538 539 539 540 540 541 541 542 542 543 543 544 544 545 545 546 546 547 547 548 548 549 549 550 550 551 551 552 552 553 553 554 554 555 555 556 556 557 557 558 558 559 559 560 560 561 561 562 562 563 563 564 564 565 565 566 566 567 567 568 568 569 569 570 570 571 571 572 572 573 573 574 574 575 575 576 576 577 577 578 578 579 579 580 580 581 581 582 582 583 583 584 584 585 585 586 586 587 587 588 588 589 589 590 590 591 591 592 592 593 593 594 594 595 595 596 596 597 597 598 598 599 599 600 600 601 601 602 602 603 603 604 604 605 605 606 606 607 607 608 608 609 609 610 610 611 611 612 612 613 613 614 614 615 615 616 616 617 617 618 618 619 619 620 620 621 621 622 622 623 623 624 624 625 625 626 626 627 627 628 628 629 629 630 630 631 631 632 632 633 633 634 634 635 635 636 636 637 637 638 638 639 639 640 640 641 641 642 642 643 643 644 644 645 645 646 646 647 647 648 648 649 649 650 650 651 651 652 652 653 653 654 654 655 655 656 656 657 657 658 658 659 659 660 660 661 661 662 662 663 663 664 664 665 665 666 666 667 667 668 668 669 669 670 670 671 671 672 672 673 673 674 674 675 675 676 676 677 677 678 678 679 679 680 680 681 681 682 682 683 683 684 684 685 685 686 686 687 687 688 688 689 689 690 690 691 691 692 692 693 693 694 694 695 695 696 696 697 697 698 698 699 699 700 700 701 701 702 702 703 703 704 704 705 705 706 706 707 707 708 708 709 709 710 710 711 711 712 712 713 713 714 714 715 715 716 716 717 717 718 718 719 719 720 720 721 721 722 722 723 723 724 724 725 725 726 726 727 727 728 728 729 729 730 730 731 731 732 732 733 733 734 734 735 735 736 736 737 737 738 738 739 739 740 740 741 741 742 742 743 743 744 744 745 745 746 746 747 747 748 748 749 749 750 750 751 751 752 752 753 753 754 754 755 755 756 756 757 757 758 758 759 759 760 760 761 761 762 762 763 763 764 764 765 765 766 766 767 767 768 768 769 769 770 770 771 771 772 772 773 773 774 774 775 775 776 776 777 777 778 778 779 779 780 780 781 781 782 782 783 783 784 784 785 785 786 786 787 787 788 788 789 789 790 790 791 791 792 792 793 793 794 794 795 795 796 796 797 797 798 798 799 799 800 800 801 801 802 802 803 803 804 804 805 805 806 806 807 807 808 808 809 809 810 810 811 811 812 812 813 813 814 814 815 815 816 816 817 817 818 818 819 819 820 820 821 821 822 822 823 823 824 824 825 825 826 826 827 827 828 828 829 829 830 830 831 831 832 832 833 833 834 834 835 835 836 836 837 837 838 838 839 839 840 840 841 841 842 842 843 843 844 844 845 845 846 846 847 847 848 848 849 849 850 850 851 851 852 852 853 853 854 854 855 855 856 856 857 857 858 858 859 859 860 860 861 861 862 862 863 863 864 864 865 865 866 866 867 867 868 868 869 869 870 870 871 871 872 872 873 873 874 874 875 875 876 876 877 877 878 878 879 879 880 880 881 881 882 882 883 883 884 884 885 885 886 886 887 887 888 888 889 889 890 890 891 891 892 892 893 893 894 894 895 895 896 896 897 897 898 898 899 899 900 900 901 901 902 902 903 903 904 904 905 905 906 906 907 907 908 908 909 909 910 910 911 911 912 912 913 913 914 914 915 915 916 916 917 917 918 918 919 919 920 920 921 921 922 922 923 923 924 924 925 925 926 926 927 927 928 928 929 929 930 930 931 931 932 932 933 933 934 934 935 935 936 936 937 937 938 938 939 939 940 940 941 941 942 942 943 943 944 944 945 945 946 946 947 947 948 948 949 949 950 950 951 951 952 952 953 953 954 954 955 955 956 956 957 957 958 958 959 959 960 960 961 961 962 962 963 963 964 964 965 965 966 966 967 967 968 968 969 969 970 970 971 971 972 972 973 973 974 974 975 975 976 976 977 977 978 978 979 979 980 980 981 981 982 982 983 983 984 984 985 985 986 986 987 987 988 988 989 989 990 990 991 991 992 992 993 993 994 994 995 995 996 996 997 997 998 998 999 999 1000 1000 1001 1001 1002 1002 1003 1003 1004 1004 1005 1005 1006 1006 1007 1007 1008 1008 1009 1009 1010 1010 1011 1011 1012 1012 1013 1013 1014 1014 1015 1015 1016 1016 1017 1017 1018 1018 1019 1019 1020 1020 1021 1021 1022 1022 1023 1023 1024 1024 1025 1025 1026 1026 1027 1027 1028 1028 1029 1029 1030 1030 1031 1031 1032 1032 1033 1033 1034 1034 1035 1035 1036 1036 1037 1037 1038 1038 1039 1039 1040 1040 1041 1041 1042 1042 1043 1043 1044 1044 1045 1045 1046 1046 1047 1047 1048 1048 1049 1049 1050 1050 1051 1051 1052 1052 1053 1053 1054 1054 1055 1055 1056 1056 1057 1057 1058 1058 1059 1059 1060 1060 1061 1061 1062 1062 1063 1063 1064 1064 1065 1065 1066 1066 1067 1067 1068 1068 1069 1069 1070 1070 1071 1071 1072 1072 1073 1073 1074 1074 1075 1075 1076 1076 1077 1077 1078 1078 1079 1079 1080 1080 1081 1081 1082 1082 1083 1083 1084 1084 1085 1085 1086 1086 1087 1087 1088 1088 1089 1089 1090 1090 1091 1091 1092 1092 1093 1093 1094 1094 1095 1095 1096 1096 1097 1097 1098 1098 1099 1099 1100 1100 1101 1101 1102 1102 1103 1103 1104 1104 1105 1105 1106 1106 1107 1107 1108 1108 1109 1109 1110 1110 1111 1111 1112 1112 1113 1113 1114 1114 1115 1115 1116 1116 1117 1117 1118 1118 1119 1119 1120 1120 1121 1121 1122 1122 1123 1123 1124 1124 1125 1125 1126 1126 1127 1127 1128 1128 1129 1129 1130 1130 1131 1131 1132 1132 1133 1133 1134 1134 1135 1135 1136 1136 1137 1137 1138 1138 1139 1139 1140 1140 1141 1141 1142 1142 1143 1143 1144 1144 1145 1145 1146 1146 1147 1147 1148 1148 1149 1149 1150 1150 1151 1151 1152 1152 1153 1153 1154 1154 1155 1155 1156 1156 1157 1157 1158 1158 1159 1159 1160 1160 1161 1161 1162 1162 1163 1163 1164 1164 1165 1165 1166 1166 1167 1167 1168 1168 1169 1169 1170 1170 1171 1171 1172 1172 1173 1173 1174 1174 1175 1175 1176 1176 1177 1177 1178 1178 1179 1179 1180 1180 1181 1181 1182 1182 1183 1183 1184 1184 1185 1185 1186 1186 1187 1187 1188 1188 1189 1189 1190 1190 1191 1191 1192 1192 1193 1193 1194 1194 1195 1195 1196 1196 1197 1197 1198 1198 1199 1199 1200 1200 1201 1201 1202 1202 1203 1203 1204 1204 1205 1205 1206 1206 1207 1207 1208 1208 1209 1209 1210 1210 1211 1211 1212 1212 1213 1213 1214 1214 1215 1215 1216 1216 1217 1217 1218 1218 1219 1219 1220 1220 1221 1221 1222 1222 1223 1223 1224 1224 1225 1225 1226 1226 1227 1227 1228 1228 1229 1229 1230 1230 1231 1231 1232 1232 1233 1233 1234 1234 1235 1235 1236 1236 1237 1237 1238 1238 1239 1239 1240 1240 1241 1241 1242 1242 1243 1243 1244 1244 1245 1245 1246 1246 1247 1247 1248 1248 1249 1249 1250 1250 1251 1251 1252 1252 1253 1253 1254 1254 1255 1255 1256 1256 1257 1257 1258 1258 1259 1259 1260 1260 1261 1261 1262 1262 1263 1263 1264 1264 1265 1265 1266 1266 1267 1267 1268 1268 1269 1269 1270 1270 1271 1271 1272 1272 1273 1273 1274 1274 1275 1275 1276 1276 1277 1277 1278 1278 1279 1279 1280 1280 1281 1281 1282 1282 1283 1283 1284 1284 1285 1285 1286 1286 1287 1287 1288 1288 1289 1289 1290 1290 1291 1291 1292 1292 1293 1293 1294 1294 1295 1295 1296 1296 1297 1297 1298 1298 1299 1299 1300 1300 1301 1301 1302 1302 1303 1303 1304 1304 1305 1305 1306 1306 1307 1307 1308 1308 1309 1309 1310 1310 1311 1311 1312 1312 1313 1313 1314 1314 1315 1315 1316 1316 1317 1317 1318 1318 1319 1319 1320 1320 1321 1321 1322 1322 1323 1323 1324 1324 1325 1325 1326 1326 1327 1327 1328 1328 1329 1329 1330 1330 1331 1331 1332 1332 1333 1333 1334 1334 1335 1335 1336 1336 1337 1337 1338 1338 1339 1339 1340 1340 1341 1341 1342 1342 1343 1343 1344 1344 1345 1345 1346 1346 1347 1347 1348 1348 1349 1349 1350 1350 1351</pre>

Original Codes	Fixed Codes
frontEnd > user > submit_design.html	
<pre>public > user > submit_design.html 1 <!DOCTYPE html> 2 <html lang="en"> 3 4 <head> 5 <title>B.D.S</title> 6 <!-- Font Awesome--> 7 <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.2/css/all.css"> 8 <!-- Google Fonts--> 9 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap"> 10 <!-- Bootstrap Core CSS--> 11 <link href="https://cdn.jsdelivr.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.0/css/bootstrap.min.css" rel="stylesheet"> 12 <link href="https://cdn.jsdelivr.cloudflare.com/ajax/libs/mdbootstrap/4.19.1/css/mdb.min.css" rel="stylesheet"> 13 <!-- Material Design Bootstrap--> 14 <link href="https://cdmjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.css" rel="stylesheet"> 15 <!-- Noty CSS Map--> 16 <link href="https://cdmjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.css.map" rel="stylesheet"> 17 <!-- Noty CSS Map--> 18 <link href="/css/site.css" rel="stylesheet"> 19 </head> 20 21 <body> 22 <div class="container-fluid">... 23 24 <!-- jQuery--> 25 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script> 26 <!-- Bootstrap tooltips--> 27 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.4/umd/popper.min.js"></script> 28 <!-- Bootstrap core JavaScript--> 29 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.0/js/bootstrap.min.js"></script> 30 <!-- MDB core JavaScript--> 31 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mdbbootstrap/4.19.1/js/mdb.min.js"> 32 </script> 33 <!-- Web App JS--> 34 <script src="/js/axios.js"></script> 35 <script src="/js/submit_design.js"></script> 36 <script src="/js/global.js"></script> 37 </body> 38 </html></pre>	<pre>public > user > submit_design.html 1 <!DOCTYPE html> 2 <html lang="en"> 3 4 <head> 5 <script> 6 const token = sessionStorage.getItem("token"); 7 const role = sessionStorage.getItem("role_name"); 8 const loggedInUserID = parseInt(sessionStorage.getItem("user_id")); 9 const baseUrl = "https://localhost:5000"; 10 11 // ----- 12 // Checking if the user has logged in or not 13 // ----- 14 if (token === null isNaN(loggedInUserID)) { 15 console.log("no token so outta here"); 16 window.location.href = "/login.html"; 17 } else { 18 // ----- 19 // Backend call to check token validity & authorisation 20 // ----- 21 axios({ 22 method: "post", 23 url: baseUrl + '/api/user/authentication-check', 24 //data: webFormData, 25 headers: { 26 authorization: "Bearer " + token, 27 'user': loggedInUserID 28 } 29 }) 30 .then(function (response) { 31 if (response.data.userRole == "user") { 32 console.log("Access Granted!"); 33 } else { 34 console.log("Access Denied!"); 35 sessionStorage.clear(); 36 window.location.href = "/home.html"; 37 } 38 }) 39 .catch(function (response) { 40 console.log("Error!!!"); 41 console.log(response); 42 43 if (response.response.status == 403) { 44 sessionStorage.clear(); 45 window.location.href = "/login.html"; 46 } else { 47 window.location.href = "/home.html"; 48 } 49 }); 50 } 51 </script> 52 </head> 53 54 <body>... 55 </body> 56 </html></pre>
frontEnd > user > update_design.html	
<pre>public > user > update_design.html 1 <!DOCTYPE html> 2 <html lang="en"> 3 4 <head> 5 <title>B.D.S</title> 6 <!-- Font Awesome--> 7 <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.2/css/all.css"> 8 <!-- Google Fonts--> 9 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap"> 10 <!-- Bootstrap Core CSS--> 11 <link href="https://cdn.jsdelivr.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.0/css/bootstrap.min.css" rel="stylesheet"> 12 <link href="https://cdn.jsdelivr.cloudflare.com/ajax/libs/mdbootstrap/4.19.1/css/mdb.min.css" rel="stylesheet"> 13 <!-- Material Design Bootstrap--> 14 <link href="https://cdmjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.css" rel="stylesheet"> 15 <!-- Noty CSS Map--> 16 <link href="https://cdmjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.css.map" rel="stylesheet"> 17 <!-- Noty CSS Map--> 18 <link href="/css/site.css" rel="stylesheet"> 19 </head> 20 21 <body> 22 <div class="container-fluid">... 23 24 <!-- jQuery--> 25 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script> 26 <!-- Bootstrap tooltips--> 27 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.4/umd/popper.min.js"></script> 28 <!-- Bootstrap core JavaScript--> 29 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.0/js/bootstrap.min.js"></script> 30 <!-- MDB core JavaScript--> 31 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mdbbootstrap/4.19.1/js/mdb.min.js"> 32 </script> 33 <!-- Web App JS--> 34 <script src="/js/axios.js"></script> 35 <script src="/js/update_design.js"></script> 36 <script src="/js/global.js"></script> 37 </body> 38 </html></pre>	<pre>public > user > update_design.html 1 <!DOCTYPE html> 2 <html lang="en"> 3 4 <head> 5 <script> 6 const token = sessionStorage.getItem("token"); 7 const role = sessionStorage.getItem("role_name"); 8 const loggedInUserID = parseInt(sessionStorage.getItem("user_id")); 9 const baseUrl = "https://localhost:5000"; 10 11 // ----- 12 // Checking if the user has logged in or not 13 // ----- 14 if (token === null isNaN(loggedInUserID)) { 15 console.log("no token so outta here"); 16 window.location.href = "/login.html"; 17 } else { 18 // ----- 19 // Backend call to check token validity & authorisation 20 // ----- 21 axios({ 22 method: "post", 23 url: baseUrl + '/api/user/authentication-check', 24 //data: webFormData, 25 headers: { 26 authorization: "Bearer " + token, 27 'user': loggedInUserID 28 } 29 }) 30 .then(function (response) { 31 if (response.data.userRole == "user") { 32 console.log("Access Granted!"); 33 } else { 34 console.log("Access Denied!"); 35 sessionStorage.clear(); 36 window.location.href = "/home.html"; 37 } 38 }) 39 .catch(function (response) { 40 console.log("Error!!!"); 41 console.log(response); 42 43 if (response.response.status == 403) { 44 sessionStorage.clear(); 45 window.location.href = "/login.html"; 46 } else { 47 window.location.href = "/home.html"; 48 } 49 }); 50 } 51 </script> 52 </head> 53 54 <body>... 55 </body> 56 </html></pre>

Demo Screenshot (After)

The screenshot shows a web browser window with the URL `localhost:3001/user/manage_submission.html`. The page title is "Manage my submissions". A search bar with placeholder "Search design title" and a "SEARCH" button is visible. In the developer tools' Application tab, the Local Storage section shows a key "token" with a value of "ey...". The developer console displays several error messages related to CORB and file loading issues.

```
Navigated to https://localhost:3001/user/manage_submission.html
Cross-Origin Read Blocking (CORB) blocked cross-origin manage_submission.html:17 response https://cdnjs.cloudflare.com/ajax/libs/motyv/3.1.4/motyv.css.map with MIME type application/octet-stream. See https://www.chromestatus.com/feature/60297092492768 for more details.
Search design form detected in user manage submission interface. Binding event handling logic to form elements.
Access Granted!
DevTools failed to load source map: Could not load content for https://localhost:3001/user/manage_submission.html:165 net::ERR_HTTP_RESPONSE_CODE_FAILURE
```

In the developer console, there is confirmation that access has been granted as the logged in user is an administrator

The screenshot shows a web browser window with the URL `localhost:3001/home.html`. The page title is "Welcome" and features a logo for "Bee Design Award Competition". Below the logo, it says "Design to inspire dreams". A note at the bottom states "Design submission period is from 1st of November to 31 December". The developer tools' Application tab shows the Local Storage section. The developer console displays several error messages related to CORB and file loading issues.

```
Navigated to https://localhost:3001/user/manage_submission.html
Cross-Origin Read Blocking (CORB) blocked cross-origin manage_submission.html:17 response https://cdnjs.cloudflare.com/ajax/libs/motyv/3.1.4/motyv.css.map with MIME type application/octet-stream. See https://www.chromestatus.com/feature/60297092492768 for more details.
Search design form detected in user manage submission interface. Binding event handling logic to form elements.
Access Granted!
DevTools failed to load source map: Could not load content for https://localhost:3001/user/manage_submission.html:165 net::ERR_HTTP_RESPONSE_CODE_FAILURE
Navigated to https://localhost:3001/admin/profile.html
Cross-Origin Read Blocking (CORB) blocked cross-origin response profile.html:17 response https://cdnjs.cloudflare.com/ajax/libs/motyv/3.1.4/motyv.css.map with MIME type text/plain. See https://www.chromestatus.com/feature/60297092492768 for more details.
Profile page is detected. Binding event handling logic to form elements.
Access Granted!
DevTools failed to load source map: Could not load content for https://localhost:3001/admin/profile.html:166 net::ERR_HTTP_RESPONSE_CODE_FAILURE
Navigated to https://localhost:3001/home.html
Cross-Origin Read Blocking (CORB) blocked cross-origin response http://home.html:17 response https://cdnjs.cloudflare.com/ajax/libs/motyv/3.1.4/motyv.css.map with MIME type text/plain. See https://www.chromestatus.com/feature/60297092492768 for more details.
```

In this case, the user has been redirected to the home page and their localStorage is cleared as the script has detected that the user does not have the privilege to visit that page.

Vulnerability 6 – Broken Access Control

V-04 Broken Access Control	<p>Users can run any of the admin functions regardless of user roles/permissions such updating the user role.</p>	<p>After gaining access to the administrator only pages by changing the URL, the administrator only functions can also be called. One such example is changing the user's role in the system. By going to /admin/manage_user.html, any user can simply change their role in the system with the dropdown box and submit the change.</p>	<p>Recommend including a validation middleware to validate any user's token before any function is called. This ensures that the user's identity is validated before any changes to the database can be made.</p>
	<p>Impact - High</p>	<p>Ease of exploit - Easy</p>	

Demo Screenshot (Before)

user_id	fullname	email	user_password	role_id
100	rita	rita@designer.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg...	2
101	robert	robert@admin.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg...	1
102	bob	bob@designer.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg...	2
103	braddy	braddy@designer.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg...	2
104	josh	josh@designer.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg...	2
105	john	john@designer.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg...	2

This screenshot shows the database connected to the system where role_id 1 is an administrator and role_id 2 is a normal user. Currently bob is a normal user.

```

<!DOCTYPE html>
<html lang="en">
  <head></head>
  <body>
    <div class="container-fluid">
      <!-- Navbar -->
      <nav class="mb-1 navbar navbar-expand-lg navbar-dark default-color"></nav>
      <!-- /Navbar -->
      <main>
        <h2 class="text-center font-up font-bold blue-text py-4">Update user</h2>
        <div class="row"> <!-- Grid column-->
          <div class="offset-md-2 col-md-8">
            <div class="card"> <!-- Form register-->
              <form id="updateUserFormContainer">
                <div class="card-body">
                  <h4 class="card-title">...</h4>
                  <p class="card-text">...</p>
                  <div class="text-right">...</div>
                </div>
              </form>
            </div>
          </div>
        </div>
        <hr class="my-4">
      </main>
    </div>
  </body>
</html>

```

About this interface: The administrator role can make changes to the role so that he can set another person as an administrator.

In the screenshot above, rita@designer.com a normal user can change the roles of any members as there is no authentication in the front or back end. Once submit is entered, the role of bob changes to administrator.

	user_id	fullname	email	user_password	role_id
▶	100	rita	rita@designer.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg...	2
	101	robert	robert@admin.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg...	1
	102	bob	bob@designer.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg...	1
	103	braddy	braddy@designer.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg...	2
	104	josh	josh@designer.com	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg...	2

As seen in the database, the role id of bob@designer.com has been changed to 1 from 2, making him an administrator in the system.

Code Fixes for Vulnerability 6

Original Codes	Fixed Codes
backEnd > src > routes.js	
<pre> 1 // routes.js ... 2 // Import controllers 3 const authController = require('./controllers/authController'); 4 const userController = require('./controllers/userController'); 5 const checkUserFn = require('./middlewares/checkUserFn'); 6 const checkUserFnSolution = require('./middlewares/checkUserFnSolution'); 7 8 // Match URL's with controllers 9 exports.routes = router => { 10 11 router.post('/api/user/login', authController.processLogin); 12 router.post('/api/user/register', authController.processRegister); 13 router.post('/api/user/processSubmission', checkUserFn.getClientUserId, userController.processDesignSubmission); 14 router.put('/api/user/', userController.processUpdateOneUser); 15 router.put('/api/user/design/', userController.processUpdateOneDesign); 16 router.post('/api/user/processInvitation', checkUserFn.getInvitationId, userController.processSendInvitation); 17 18 router.get('/api/user/process-search-design/:pageNumber/:search?', checkUserFn.getCompanyId, userController.processGetSubmissionData); 19 router.get('/api/user/process-search-user/:idNumber/:search?', checkUserFn.getClientUserId, userController.processGetUserData); 20 router.get('/api/user/process-search-user/:idNumber/:search?', checkUserFn.getOneDesign/:idNumber, userController.processGetSubmissionsByEmail); 21 router.get('/api/user/recommend', userController.processGetTheUserBots); 22 router.get('/api/user/design/:fileId', userController.processGetOneDesign/:fileId); 23 24 } </pre>	<pre> 1 // routes.js ... 2 // Import controllers 3 const authController = require('./controllers/authController'); 4 const userController = require('./controllers/userController'); 5 const checkUserFn = require('./middlewares/checkUserFn'); 6 const checkUserFnSolution = require('./middlewares/checkUserFnSolution'); 7 8 // Match URL's with controllers 9 exports.routes = router => { 10 11 router.post('/api/user/login', authController.processLogin); 12 router.post('/api/user/register', authController.processRegister); 13 router.post('/api/user/processSubmission', checkUserFn.getClientUserId, userController.processDesignSubmission); 14 15 // User: Multi Design 16 router.get('/api/user/process-search-design/:pageNumber/:search?', checkUserFn.getCompanyId, userController.accessDesignSubmission); 17 18 // User: Single Submission 19 router.get('/api/user/process-search-user/:idNumber/:search?', checkUserFn.getClientUserId, userController.accessGetSubmissionData); 20 21 // User: Single Invitation 22 router.get('/api/user/process-search-user/:idNumber/:search?', checkUserFn.getInvitationId, userController.accessSendInvitation); 23 24 // User: Check User Submission 25 router.get('/api/user/process-search-user/:design/:invitations?', checkUserFn.getCompanyId, userController.accessGetSubmission); 26 27 // User: Single Profile 28 router.get('/api/user/process-search-user/:idNumber/:search?', checkUserFn.getClientUserId, userController.accessGetUserData); 29 30 // User: Single Recommendation 31 router.get('/api/user/process-search-user/:idNumber/:search?', checkUserFn.getOneDesign/:idNumber, userController.accessGetSubmissionsByEmail); 32 33 // User: Single User 34 router.get('/api/user/recommend', userController.accessGetTheUserBots); 35 36 // User: Single Design 37 router.get('/api/user/design/:fileId', userController.accessGetOneDesign/:fileId); 38 39 } </pre>
Before: No middleware function was added to verify the user's identity	Added middleware function to check user's session validity
Original Codes	Fixed Codes
backEnd > controllers > authController.js	
<pre> exports.processLogin = (req, res, next) => { let email = req.body.email; let password = req.body.password; try { auth.authenticate(email, function (error, results) { if (error) {...} } else { if (results.length == 1) { if ((password == null) (results[0] == null)) { return res.status(500).json({ message: 'login failed' }); } if (bcrypt.compareSync(password, results[0].user_password) == true) { let data = { user_id: results[0].user_id, role_name: results[0].role_name, token: jwt.sign({ id: results[0].user_id }, config.JwtKey, { expiresIn: 86400 //Expires in 24 hrs }) }; //End of data variable setup return res.status(200).json(data); } else { // return res.status(500).json({ message: 'Login has failed.' }); return res.status(500).json({ message: error }); } //End of password comparison with the retrieved decoded password. } //End of checking if there are returned SQL results } }); } catch (error) { return res.status(500).json({ message: error }); } //end of try } </pre>	<pre> exports.processLogin = (req, res, next) => { console.log("====="); console.log("authController.js > processLogin is called and running!"); console.log("====="); let email = req.body.email; let password = req.body.password; if (validator.isEmail(email)) { try { auth.authenticate(email, function (error, results) { if (error) {...} } else { if (results.length == 1) { if ((password == null) (results[0] == null)) {...} if (bcrypt.compareSync(password, results[0].user_password) == true) { let jsonResult = { user_id: results[0].user_id, role_name: results[0].role_name, token: jwt.sign({ id: results[0].user_id, uRole: results[0].role_name }, config.JwtKey, { expiresIn: 7200 //Expires in 2 hrs }) }; //End of jsonResult variable setup return res.status(200).json(jsonResult); } else { let jsonResult = { message: 'Login has failed.' } return res.status(500).json(jsonResult); } //End of password comparison with the retrieved decoded password. } //End of checking if there are returned SQL results } }); } catch (error) { console.log("error: " + error); let jsonResult = { message: 'An error has occurred!' } return res.status(500).json(jsonResult); } //end of try } </pre>
Added user role into the token so that it is accessible for all functions that require user role authentication	

backEnd > src > middlewares > checkUserFnSolution.js

```
src > middlewares > JS checkUserFnSolution.js > ...
1 const config = require('../config/config');
2 const jwt = require('jsonwebtoken');
3 module.exports.checkForValidUserRoleUser = (req, res, next) => {
4     //If the token is valid, the logic extracts the user id and the role information.
5     //If the role is not user, then response 403 Unauthorized
6     //The user id information is inserted into the request.body.userId
7     console.log('http header - user ', req.headers['user']);
8     if (typeof req.headers.authorization !== "undefined") {
9         // Retrieve the authorization header and parse out the
10        // JWT using the split function
11        let token = req.headers.authorization.split(' ')[1];
12        //console.log('Check for received token from frontend : \n');
13        //console.log(token);
14        jwt.verify(token, config.JWTKey, (err, data) => {
15            console.log('data extracted from token \n',data);
16            if (err) {
17                console.log(err);
18                return res.status(403).send({ message: 'Unauthorized access' });
19            }
20            else {
21                req.body.userId = data.id;
22                next();
23            }
24        })
25    }
26    else{
27        res.status(403).send({ message: 'Unauthorized access' });
28    }
29 }
30 } //End of checkForValidUserRoleUser
```

```
src > middlewares > JS checkUserFnSolution.js > checkForValidUserRoleUser > checkForValidUserRoleUser > jwt
1 const config = require('../config/config');
2 const jwt = require('jsonwebtoken');
3 const { send } = require('process');
4 const { json } = require('body-parser');
5 module.exports.checkForValidUserRoleUser = (req, res, next) => {
6     console.log("=====");
7     console.log("checkUserFnSolution.js is called and running!");
8     console.log("=====");
9
10    //If the token is valid, the logic extracts the user id and the role information.
11    //If the role is not user, then response 403 Unauthorized
12    //The user id information is inserted into the request.body.userId
13    console.log('http header - user ', req.headers['user']);
14    if (typeof req.headers.authorization !== "undefined") {
15        // Retrieve the authorization header and parse out the
16        // JWT using the split function
17        let token = req.headers.authorization.split(' ')[1];
18        //console.log('Check for received token from frontend : \n');
19        //console.log(token);
20        jwt.verify(token, config.JWTKey, (err, data) => {
21            console.log('data extracted from token \n', data);
22            if (err) {
23                console.log(err);
24
25                let jsonResult = {
26                    message: 'Unauthorized access'
27                };
28                return res.status(403).send(jsonResult);
29            }
30            else {
31                // console.log("-----");
32                // console.log("data id: " + data.id);
33                // console.log("data uRole: " + data.uRole);
34                // console.log("-----");
35
36                reqdecodedId = data.id;
37                reqdecodedRole = data.uRole
38
39                // console.log("\n-----");
40                // console.log("reqdecodedId: " + reqdecodedId);
41                // console.log("reqdecodedRole: " + reqdecodedRole);
42                // console.log("-----");
43                next();
44            }
45        })
46    }
47    else {
48        console.log("here!!!!");
49        var jsonResult = {
50            message: 'Unauthorized access'
51        };
52        res.status(403).send(jsonResult);
53    }
54 }
55 } //End of checkForValidUserRoleUser
```

Added a req.decodedRole to store the role that is retrieved from the token

Original Codes

Fixed Codes

backEnd > src > middlewares > accessCheckFrontEnd.js

N/A

```
src > middlewares > JS accessCheckFrontEnd.js > ...
1 module.exports.accessCheckFrontEnd = (req, res, next) => {
2     // If this middleware function is called, it will pass the data that was stored previously
3     // back to the FrontEnd.
4
5     // This module is just a lightweight module that passes the information back to FrontEnd and
6     // nothing else
7     console.log("=====");
8     console.log("accessCheckFrontEnd.js is called and running!");
9     console.log("=====");
10    console.log("");
11
12    //console.log("reqdecodedId: " + reqdecodedId);
13    //console.log("reqdecodedRole: " + reqdecodedRole);
14
15    var sendBack = {
16        'userId': reqdecodedId,
17        'userRole': reqdecodedRole
18    };
19
20    res.status(200).json(sendBack);
21
22 } //End of accessCheckFrontEnd
```

Added a new middleware function that will pass the data back to the frontEnd if the frontEnd needs the data to do page authentication

frontEnd > public > js > admin_manage_user.js

```
public > js > admin_manage_user.js > @ on(click) callback > @ then() callback
1 let $searchDesignFormContainer = $('#searchUserFormContainer');
2 if ($searchDesignFormContainer.length != 0) {
3   console.log('Search user form detected in manage user interface. Binding event handling logic to form elements.');
4   //the following code will create a method to submit search parameters
5   //to server-side api when the #submitButton element fires the click event.
6   $('#submitButton').on('click', function(event) {
7     event.preventDefault();
8     const baseU1 = 'http://localhost:5000';
9     let searchInput = $('#searchInput').val();
10    let userId = localStorage.getItem('user_id');
11    axios({
12      headers: {
13        'user': userId,
14      },
15      method: 'get',
16      url: baseU1 + '/api/user/process-search-user/1/' + searchInput,
17    }).then(function(response) {
18      //using the following to inspect the response data data structure ...
19    })
20  })
21 }

public > js > admin_manage_user.js > @ on(click) callback > @ then() callback
1 let $searchDesignFormContainer = $('#searchUserFormContainer');
2 if ($searchDesignFormContainer.length != 0) {
3   console.log('Search user form detected in manage user interface. Binding event handling logic to form elements.');
4   //the following code will create a method to submit search parameters
5   //to server-side api when the #submitButton element fires the click event.
6   $('#submitButton').on('click', function(event) {
7     event.preventDefault();
8     const baseU1 = 'http://localhost:5000';
9     let searchInput = $('#searchInput').val();
10    let userId = localStorage.getItem('user_id');
11    let token = localStorage.getItem('token');
12    axios({
13      headers: {
14        'user': userId,
15        'Authorization': 'Bearer ' + token
16      },
17      method: 'get',
18      url: baseU1 + '/api/user/process-search-user/1/' + searchInput,
19    })
20  })
21 .then(function(response) {
22   //use the following to inspect the response data data structure ...
23 })


```

Added Line 12 and Line 17 to enable authorization check for the webpage by sending the token to the endpoint

frontEnd > public > js > admin_manage_user_submission.js

```
public > js > admin_check_user_submission.js > @ on(click) callback > @ then() callback
1 let $searchUserFormContainer = $('#searchUserFormContainer');
2 if ($searchUserFormContainer.length != 0) {
3   console.log('Search user form detected in user manage submission interface. Binding event handling logic to form elements.');
4   //the following code will create a method to do record searching
5   //if the #submitButton object exists
6   //the following code will create a method to send key-value pair information to do record searching
7   //to server-side api when the #submitButton element fires the click event.
8   $('#submitButton').on('click', function(event) {
9     event.preventDefault();
10    const baseU1 = 'http://localhost:5000';
11    let searchInput = $('#searchInput').val();
12    let userId = localStorage.getItem('user_id');
13    axios({
14      headers: {
15        'user': userId
16      },
17      method: 'get',
18      url: baseU1 + '/api/user/process-search-user-design/1/' + searchInput,
19    }).then(function(response) {
20      //use the following to inspect the response data data structure ...
21    })
22  })
23 }

public > js > admin_check_user_submission.js > @ on(click) callback > @ then() callback
1 let $searchUserFormContainer = $('#searchUserFormContainer');
2 if ($searchUserFormContainer.length != 0) {
3   console.log('Search user form detected in user manage submission interface. Binding event handling logic to form elements.');
4   //the following code will create a method to do record searching
5   //if the #submitButton object exists
6   //the following code will create a method to send key-value pair information to do record searching
7   //to server-side api when the #submitButton element fires the click event.
8   $('#submitButton').on('click', function(event) {
9     event.preventDefault();
10    const baseU1 = 'https://localhost:5000';
11    let searchInput = $('#searchInput').val();
12    let token = sessionStorage.getItem('token');
13    axios({
14      headers: {
15        'user': userId
16        'Authorization': 'Bearer ' + token
17      },
18      method: 'get',
19      url: baseU1 + '/api/user/process-search-user-design/1/' + searchInput,
20    })
21  })
22 .then(function(response) {
23   //use the following to inspect the response data data structure ...
24 })


```

Added Line 13 and Line 17 to enable authorization check for the webpage by sending the token to the endpoint

backEnd > src > services > userService.js (module.exports.getOneUserDataByEmail)

```
module.exports.getOneUserDataByEmail = function (search) {
  console.log('getOneUserDataByEmail method is called.');
  console.log('Prepare query to fetch one user record');
  userDataQuery = 'SELECT user_id, fullname, email, user.role_id, role_name
  FROM user INNER JOIN role ON user.role_id = role.role_id WHERE email="' + search + '"';
  return new Promise((resolve, reject) => {
    //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection
    //to prepare the following code pattern which does not use callback technique (uses Promise technique)
    pool.getConnection((err, connection) => {
      if (err) {
        console.log('Database connection error ', err);
        resolve(err);
      } else {
        connection.query(userDataQuery, (err, results) => {
          if (err) {
            reject(err);
          } else {
            resolve(results);
          }
          connection.release();
        });
      }
    });
  });
}

//End of new Promise object creation
} //End of getOneUserDataByEmail

module.exports.getOneUserDataByEmail = function (search) {
  console.log('getOneUserDataByEmail method is called and running');
  console.log('=====');

  console.log('Prepare query to fetch one user record');
  console.log('Search parameter: ' + search);
  return new Promise((resolve, reject) => {
    //I referred to https://www.codota.com/code/javascript/functions/mysql/Pool/getConnection
    //to prepare the following code pattern which does not use callback technique (uses Promise technique)
    pool.getConnection((err, connection) => {
      if (err) {
        console.log('Database connection error ', err);
        resolve(err);
      } else {
        userDataQuery = 'SELECT user_id, fullname, email, user.role_id, role_name FROM user INNER JOIN role ON user.role_id = role.role_id WHERE email = "' + search + '"';
        connection.query(userDataQuery, [search], (err, results) => {
          if (err) {
            reject(err);
          } else {
            resolve(results);
          }
          connection.release();
        });
      }
    });
  }) //End of new Promise object creation
} //End of getOneUserDataByEmail
```

Demo Screenshots (After)

The screenshot shows a POST request in Postman to `https://localhost:5000/api/user/`. The request body contains three fields: `roleId` (value 2), `recordId` (value 102), and `userId` (value 100). The response status is 403 Forbidden, with the message "message": "Forbidden".

KEY	VALUE	DESCRIPTION
<code>roleId</code>	2	
<code>recordId</code>	102	
<code>userId</code>	100	

Body Cookies Headers (9) Test Results Status: 403 Forbidden

```
1 {  
2   "message": "Forbidden"  
3 }
```

To prevent any user from querying someone else's data, we protected that endpoint such that the user is not able to change any user's role in the database if the user is not authorized

Vulnerability 7 – Cross-site Scripting Attacks (XSS)

V-05 Cross-site Scripting (XSS)	<p>When a stored XSS attack occurs, the script will be called every time the html page is accessed. In this case when the administrator searches for the user's submission, the script will be called in the database and ran on the administrator page.</p> <p>Impact - High</p>	<p>Any user can simply submit a design or edit an existing submission with a script in the title or the description and submit it into the database. This script will then become a stored XSS attacked in the database and will target the administrators of the system.</p> <p>Ease of exploit - Easy</p>	<p>Recommend doing a regular expression(regex) check on the design title and design description input to remove any special characters to be inputted, making script injections impossible.</p>
--	--	--	---

Demo Screenshot (Before)

Submit design

Design title

```
<script>alert("Hacked")</script>
```

Design description

```
<script>alert("Hacked")</script>
```



Choose Files No file chosen

SUBMIT

In submit_design.html, a script can be inserted into the fields of design title and design description and will be stored in the database when submitted.

file_id	cloudinary_file_id	cloudinary_url	design_title	design_description	created_by_id
105	Design/zrl65auuz9zkv25k4cw	http://res.cloudinary.com/deuevi29r/image/upl...	rita design 6	rita design 6 description text 1 text 2 text 3 tex...	100
106	Design/zruza6htdsgeltumxp	http://res.cloudinary.com/deuevi29r/image/upl...	rita design 7	rita design 7 description text 1 text 2 text 3 tex...	100
107	Design/uhymyes5yp1qdtowokq	http://res.cloudinary.com/deuevi29r/image/upl...	rita design 8	rita design 8 description text 1 text 2 text 3 tex...	100
108	Design/ku4htzthqb5ainmzaj	http://res.cloudinary.com/deuevi29r/image/upl...	rita design 9	rita design 9 description text 1 text 2 text 3 tex...	100
109	Design/eswxqolriyysoo7gdvui	http://res.cloudinary.com/deuevi29r/image/upl...	rita design 10	rita design 10 description text 1 text 2 text 3 te...	100
110	Design/t23po97jwj8nfksbeymi	http://res.cloudinary.com/deuevi29r/image/upl...	rita design 11	rita design 11 description text 1 text 2 text 3 te...	100
111	Design/ouxgn5ursdmbf1hrdrj	http://res.cloudinary.com/deuevi29r/image/upl...	rita design 12	rita design 12 description text 1 text 2 text 3 te...	100
112	Design/wxe1k2hs0wfjg7fdg2mr	http://res.cloudinary.com/deuevi29r/image/upl...	asdads	adsadsa	100
113	Design/qdoh94bzxcceypft8uta	http://res.cloudinary.com/deuevi29r/image/upl...	asdads	adsadsa	100
114	Design/f02e1jg9tv4rkztaq	http://res.cloudinary.com/deuevi29r/image/upl...	asdads	DELETE *	100
115	Design/skj7acegugtexlvssdei	http://res.cloudinary.com/deuevi29r/image/upl...	<script>alert("Hacked")</script>	<script>alert("Hacked")</script>	100
116	Design/zaotexn3jq5arelzw	http://res.cloudinary.com/deuevi29r/image/upl...	<script>alert("Hacked")</script>	asdadasd" DELETE *	100

As seen in the database, the script is stored in the design title and description.

localhost:3001 says

Hacked

OK

Check submission

Search by email

rita@designer.com

SEARCH

When the administrator checks the submission of rita, the script will then load from the database and display the script, harming the administrator.

Design title
rita design 1

Design description
<script>alert("Rita design 1 script!")</script>

BACK **SUBMIT**

As for update deisgn, inserting a script also works for the design title and design description.

	file_id	cloudinary_file_id	cloudinary_url	design_title	design_description	created_by_id
▶	100	Design/gz0g0sdy3hwgqhpqnvf	http://res.cloudinary.com/deuevi29r/image/upl...	rita design 1	<script>alert("Rita design 1 script!")</script>	100
	101	Design/yjevf0kf5ryv4yf290i	http://res.cloudinary.com/deuevi29r/image/upl...	rita design 2	rita design 2 description text 1 text 2 text 3 tex...	100

In the database, the design description of rita design 1 is the script from the update design

localhost:3001 says
Rita design 1 script!

OK

Check submission

Search by email
rita@designer.com

SEARCH

And when the administrator checks rita submission, the script is ran from the database.

Original Codes	Fixed Codes
frontEnd > public > js > submit_design.js	
<pre>public > js > submit_design.js 1 let \$submitDesignFormContainer = \$('#submitDesignFormContainer'); 2 if (\$submitDesignFormContainer.length != 0) { 3 console.log('Submit design form detected. Binding event handling logic to form elements.'); 4 //If the jQuery object which represents the form element exists, 5 //the following code will create a method to submit design details 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'http://localhost:5000'; 10 let userId = localStorage.getItem('user_id'); 11 let designTitle = \$('#designTitleInput').val(); 12 let designDescription = \$('#designDescriptionInput').val(); 13 let webFormData = new FormData(); 14 webFormData.append('designTitle', designTitle); 15 webFormData.append('designDescription', designDescription); 16 // HTML file input, chosen by user 17 webFormData.append('file', document.getElementById('fileInput').files[0]); 18 }) 19 .method('post', 20 url: baseUrl + '/api/user/process-submission', 21 data: webFormData, 22 headers: { 'Content-Type': 'multipart/form-data', 'user': userId } 23) 24 .then(function(response) { 25 Noty.overrideDefaults({ 26 callbacks: { 27 onShow: function() { 28 if (this.options.type === 'systemresponse') { 29 this.barDom.innerHTML += '<div class="my-custom-template noty_body">'; 30 this.barDom.innerHTML += '<div class="noty-header">Message</div>'; 31 this.barDom.innerHTML += '<p class="noty-message-body">' + this.options.text + '</p>'; 32 this.barDom.innerHTML += ''; 33 this.barDom.innerHTML += '</div>'; 34 } 35 } 36 }) 37 }) 38 })</pre>	<pre>public > js > submit_design.js 1 let \$submitDesignFormContainer = \$('#submitDesignFormContainer'); 2 if (\$submitDesignFormContainer.length != 0) { 3 console.log('Submit design form detected. Binding event handling logic to form elements.'); 4 //If the jQuery object which represents the form element exists, 5 //the following code will create a method to submit design details 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'https://localhost:5000'; 10 let userId = sessionStorage.getItem('user_id'); 11 let designTitle = \$('#designTitleInput').val(); 12 let designDescription = \$('#designDescriptionInput').val(); 13 let webFormData = new FormData(); 14 let token = sessionStorage.getItem('token'); 15 16 // regex check to ensure that the designTitle and designDescription does not contain malicious codes 17 let designTitle_regex = new RegExp("[a-zA-Z0-9 ,!]*\$"); 18 let designDescription_regex = new RegExp("[a-zA-Z0-9 ,!]*\$"); 19 20 if (designTitle_regex.test(designTitle) && designDescription_regex.test(designDescription)) { 21 webFormData.append('designTitle', designTitle); 22 webFormData.append('designDescription', designDescription); 23 // HTML file input, chosen by user 24 webFormData.append('file', document.getElementById('fileInput').files[0]); 25 axios({ 26 method: 'post', 27 url: baseUrl + '/api/user/process-submission', 28 data: webFormData, 29 headers: { 30 'Content-Type': 'multipart/form-data', 31 'user': userId, 32 'authorization': 'Bearer ' + token 33 } 34 }) 35 } 36 }) 37 })</pre>
Added regular expression checks on all input field by the user to ensure that the input is clean and does not contain malicious codes	
backEnd > src > controllers > userController.js (exports.processDesignSubmission)	
<pre>exports.processDesignSubmission = (req, res, next) => { let designTitle = req.body.designTitle; let designDescription = req.body.designDescription; let userId = req.body.userId; let file = req.body.file; fileDataManager.uploadFile(file, async function(error, result) { console.log('check result variable in fileDataManager.upload code block\n', result); console.log('check error variable in fileDataManager.upload code block\n', error); let uploadResult = result; if (error) { let message = 'Unable to complete file submission.'; res.status(500).json({ message: message }); res.end(); } else {</pre>	<pre>exports.processDesignSubmission = (req, res, next) => { console.log("====="); console.log("userController.js - processDesignSubmission is called and running!"); console.log("====="); let designTitle = req.body.designTitle; let designDescription = req.body.designDescription; var userId = req.body.userId; let file = req.body.file; // ===== Start of Changes ===== let decodedId = req.decodedId; userId = checkUserID(userId, decodedId); // regex check to ensure that the designTitle and designDescription does not contain malicious codes let designTitle_regex = new RegExp("[a-zA-Z0-9 ,!]*\$"); let designDescription_regex = new RegExp("[a-zA-Z0-9 ,!]*\$"); // if regex passes, then store the files in cloudinary if (designTitle_regex.test(designTitle) && designDescription_regex.test(designDescription)) { fileDataManager.uploadFile(file, async function (error, result) { console.log('check result variable in fileDataManager.upload code block\n', result); console.log('check error variable in fileDataManager.upload code block\n', error); let uploadResult = result; if (error) { let jsonResult = { message: 'Unable to complete file submission' }; res.status(500).json(jsonResult); res.end(); } else {</pre>
Added regular expression checks on all input field in the backEnd, just in case the user has JavaScript disabled, or it is a Man-in-the-middle attacker which bypasses frontEnd validation checks, to ensure that the input is clean and does not contain malicious codes	

Original Codes	Fixed Codes
frontEnd > public > js > update_design.js	
<pre>public > js > update_design.js > ... 1 let \$updateDesignFormContainer = \$('#updateDesignFormContainer'); 2 if (\$updateDesignFormContainer.length != 0) { 3 console.log('Update Design form is detected. Binding event handling logic to form elements.'); 4 //If the jQuery object which represents the form element exists, 5 //the following code will create a method to submit design details 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'http://localhost:5000'; 10 //Collect fileId value from the input element, fileIdInput (hidden input element) 11 let fileId = \$('#fileIdInput').val(); 12 //Obtain user id from local storage 13 let userId = localStorage.getItem('user_id'); 14 //Collect design title and description input 15 let designTitle = \$('#designTitleInput').val(); 16 let designDescription = \$('#designDescriptionInput').val(); 17 //Create a FormData object to build key-value pairs of information before 18 //making a PUT HTTP request. 19 let webFormData = new FormData(); 20 webFormData.append('designTitle', designTitle); 21 webFormData.append('designDescription', designDescription); 22 webFormData.append('fileId', fileId); 23 axios({ 24 method: 'put', 25 url: baseUrl + '/api/user/design/', 26 data: webFormData, 27 headers: { 'Content-Type': 'multipart/form-data', 'user': userId } 28 }) 29 .then(function(response) { 30 new Noty({ 31 type: 'success', 32 layout: 'topCenter', 33 theme: 'sunset', 34 timeout: '5000', 35 text: 'Updated design information.', 36 }).show(); 37 }) 38 .catch(function(error) { 39 }) 40 </pre>	<pre>public > js > update_design.js > ... 1 let \$updateDesignFormContainer = \$('#updateDesignFormContainer'); 2 if (\$updateDesignFormContainer.length != 0) { 3 console.log('Update Design form is detected. Binding event handling logic to form elements.'); 4 //The following code will create a method to submit design details 5 //to server-side api when the #submitButton element fires the click event. 6 \$('#submitButton').on('click', function(event) { 7 event.preventDefault(); 8 const baseUrl = 'https://localhost:5000'; 9 //Collect fileId value from the input element, fileIdInput (hidden input element) 10 let fileId = \$('#fileIdInput').val(); 11 //Obtain user id from local storage 12 let userId = sessionStorage.getItem('user_id'); 13 //Collect design title and description input 14 let designTitle = \$('#designTitleInput').val(); 15 let designDescription = \$('#designDescriptionInput').val(); 16 // ----- 17 // ----- Start of changes ----- 18 let designTitle_regex = new RegExp("[a-zA-Z0-9_.]*\$"); 19 let designDescription_regex = new RegExp("[a-zA-Z0-9_.]*\$"); 20 // ----- 21 if (designTitle_regex.test(designTitle) && designDescription_regex.test(designDescription)) { 22 //Create a FormData object to build key-value pairs of information before 23 //making a PUT HTTP request. 24 let webFormData = new FormData(); 25 webFormData.append('designTitle', designTitle); 26 webFormData.append('designDescription', designDescription); 27 webFormData.append('fileId', fileId); 28 // ----- 29 // ----- Start of changes ----- 30 let token = sessionStorage.getItem('token'); 31 axios({ 32 </pre>
Added regular expression checks on all input field by the user to ensure that the input is clean and does not contain malicious codes	
backEnd > src > controllers > userController.js (exports.processUpdateOneDesign)	
<pre>exports.processUpdateOneDesign = async(req, res, next) => { console.log('processUpdateOneDesign running'); //Collect data from the request body let fileId = req.body.fileId; let designTitle = req.body.designTitle; let designDescription = req.body.designDescription; try { results = await userManager.updateDesign(fileId, designTitle, designDescription); console.log(results); return res.status(200).json({ message: 'Completed update' }); } catch (error) { console.log('processUpdateOneUser method : catch block section code is running'); console.log(error, '====='); return res.status(500).json({ message: 'Unable to complete update operation' }); } }; //End of processUpdateOneDesign</pre>	<pre>exports.processUpdateOneDesign = async (req, res, next) => { console.log("=====UserController.js ====="); console.log("UserController.js processUpdateOneDesign is called and running!"); console.log("====="); //Collect data from the request body let fileId = req.body.fileId; let designTitle = req.body.designTitle; let designDescription = req.body.designDescription; // ----- // ----- Start of Changes ----- let userId = req.body.userId; let decodedId = req.decodedId; if (userId != decodedId) { // regex check to ensure that it will not accept any unclean inputs from the user let dTitle_regex = new RegExp("[a-zA-Z0-9_.]*\$"); let dDesc_regex = new RegExp("[a-zA-Z0-9_.]*\$"); console.log("fileId: " + fileId); //console.log(dTitle_regex.test(designTitle)); //console.log(dDesc_regex.test(designDescription)); // below if check ensures that both regex is successful before updating if (dTitle_regex.test(designTitle) && dDesc_regex.test(designDescription)) { try { userCheck = await userManager.getUserIdFromFileId(fileId); //console.log("userCheck: " + userCheck); if (userCheck != decodedId) { console.log("The TokenId does not match the UserID of the file that needs updating"); console.log("Terminating update operation"); let jsonResult = { message: 'Unable to complete update operation' } return res.status(500).json(jsonResult); } else { </pre>
Added regular expression checks on all input field in the backEnd, just in case the user has JavaScript disabled, or it is a Man-in-the-middle attacker which bypasses frontEnd validation checks,to ensure that the input is clean and does not contain malicious codes	

Demo Screenshot (After)

The screenshot shows a web application interface for updating design details. On the left, there's a preview of a design featuring a pink cloud-like logo with a smiling face, the text "RITA DESIGN 1", and "Design File for testing". Below the preview are two input fields: "Design title" containing "rita design 1<script>alert('hi')</script>" and "Design description" containing "rita design 1 description text 1 text 2 text 3 text 4". At the bottom are "BACK" and "SUBMIT" buttons. A red banner at the top says "Unable to update." To the right, the browser's developer tools are open, specifically the Storage and Console tabs. The Storage tab shows session storage for the current domain. The Console tab displays several error messages related to CORS, Axios, and mixed content, with one message highlighted in orange: "regex check for frontEnd has failed".

tered user can make changes to the design title and the description. The file through this interface. The vulnerable system does not have delete file.

In our codes, we have used a regular expression check to validate the user's input to make sure that there is no chance of storing a script in the database. In this screenshot, when the user tries to update any design title or design description that has special characters, the update will fail as it has failed the regular expression check.

Vulnerability 8 – Sensitive Data Exposure (localStorage)

V-03 Sensitive Data Exposure (localStorage)	The downside of localStorage is that the data stored in localStorage does not expire, leaving their token to be targeted by hackers. The change to session storage alleviates this problem as the data in session storage is cleared when the session ends (e.g. Tab closed)	If hackers can gain access to the user's computer, their information might be unknowingly stolen from them as local storage will not be wiped when the browser is closed. They can then use their token to login with their account.	Recommend making use of sessionStorage instead of localStorage so that once the browser is closed, the data gets wiped from the browser rather than being persistent in the storage. Furthermore, the JWT token in the system is set to expire in 2 hours to strengthen security if the token is leaked.
	Impact - Low	Ease of exploit - Moderate	

Demo Screenshot (Before)	
Key	Value
user_id	121
token	eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJpZCI6MTIxLC...
role_name	user

Website uses local storage to store the user_id, token and role_name, thus we have decided to switch over from localStorage to sessionStorage so that the storage will be cleared the moment the tab is closed while the data in localStorage will still be stored until it is called to clear.

The code below shows the switch from localStorage to sessionStorage in the front end.

Original Codes	Fixed Codes
frontEnd > public > js > admin_check_user_submission.js	
<pre>public > js > admin_check_user_submission.js > ⚡ on(click) callback > [o] userd 1 let \$searchUserFormContainer = \$('#searchUserFormContainer'); 2 if (\$searchUserFormContainer.length != 0) { 3 //If the \$Query object which represents the form element exists, 4 //the following code will create a method to send key-value pair information to do record searching 5 //to server-side api when the #submitButton element fires the click event. 6 \$('#submitButton').on('click', function(event) { 7 event.preventDefault(); 8 const baseUrl = 'http://localhost:5000'; 9 let searchInput = \$('#searchInput').val(); 10 let userId = localStorage.getItem('user_id'); 11 axios({ 12 ... 13 }) 14 .then(function(response) { ... 15 ... 16 }) 17 .catch(function(response) { ... 18 }); 19 }); 20 // have hard code 3 buttons For the manage-submission interface (user role) 21 //to cut down the JavaScript code for this file. 22 //If the \$Query object which represents the form element exists, 23 //the following code will create a method to make a HTTP GET 24 //to server-side api. 25 function clickHandlerForPageButton(event) { 26 event.preventDefault(); 27 const baseUrl = 'http://localhost:5000'; 28 let userId = localStorage.getItem('user_id'); 29 let pageNumber = \$(event.target).text().trim(); 30 let searchInput = \$('#searchInput').val(); 31 console.log(pageNumber); 32 axios({ 33 ... 34 }) 35 .then(function(response) { ... 36 }) 37 .catch(function(response) { ... 38 }); 39 } 40 //End of clickHandlerForPageButton 41 } //End of checking for \$searchUserFormContainer \$Query object</pre>	<pre>public > js > admin_check_user_submission.js > clickHandlerForPageButton > [o] userd 1 let \$searchUserFormContainer = \$('#searchUserFormContainer'); 2 if (\$searchUserFormContainer.length != 0) { 3 //If the \$Query object which represents the form element exists, 4 //the following code will create a method to send key-value pair information to do record searching 5 //to server-side api when the #submitButton element fires the click event. 6 \$('#submitButton').on('click', function(event) { 7 event.preventDefault(); 8 const baseUrl = 'https://localhost:5000'; 9 let searchInput = \$('#searchInput').val(); 10 let userId = sessionStorage.getItem('user_id'); 11 let token = sessionStorage.getItem('token'); 12 axios({ 13 ... 14 }) 15 .then(function(response) { ... 16 ... 17 }) 18 .catch(function(response) { ... 19 }); 20 }); 21 // have hard code 3 buttons For the manage-submission interface (user role) 22 //to cut down the JavaScript code for this file. 23 //If the \$Query object which represents the form element exists, 24 //the following code will create a method to make a HTTP GET 25 //to server-side api. 26 function clickHandlerForPageButton(event) { 27 event.preventDefault(); 28 const baseUrl = 'https://localhost:5000'; 29 let userId = sessionStorage.getItem('user_id'); 30 let pageNumber = \$(event.target).text().trim(); 31 let searchInput = \$('#searchInput').val(); 32 let token = sessionStorage.getItem('token'); 33 console.log(pageNumber); 34 axios({ 35 ... 36 }) 37 .then(function(response) { ... 38 }) 39 .catch(function(response) { ... 40 }); 41 } 42 //End of clickHandlerForPageButton 43 } //End of checking for \$searchUserFormContainer \$Query object</pre>
frontEnd > public > js > admin_manage_user.js	
<pre>public > js > admin_manage_user.js > ... 1 let \$searchDesignFormContainer = \$('#searchUserFormContainer'); 2 if (\$searchDesignFormContainer.length != 0) { 3 console.log('Search user form detected in manage user interface. Binding event handling logic to form elements.'); 4 //If the \$Query object which represents the form element exists, 5 //the following code will create a method to submit search parameters 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'http://localhost:5000'; 10 let searchInput = \$('#searchInput').val(); 11 let userId = localStorage.getItem('user_id'); 12 axios({ 13 ... 14 }) 15 .then(function(response) { ... 16 ... 17 }) 18 .catch(function(response) { ... 19 }); 20 }); 21 function clickHandlerForPageButton(event) { 22 event.preventDefault(); 23 const baseUrl = 'http://localhost:5000'; 24 let userId = localStorage.getItem('user_id'); 25 let pageNumber = \$(event.target).text().trim(); 26 let searchInput = \$('#searchInput').val(); 27 let token = sessionStorage.getItem('token'); 28 console.log('Checking the button page number which raised the click event : ', pageNumber); 29 axios({ 30 ... 31 }) 32 .then(function(response) { ... 33 ... 34 }) 35 .catch(function(response) { ... 36 }); 37 }; 38 //End of clickHandlerForPageButton 39 } //End of checking for \$searchUserFormContainer \$Query object</pre>	<pre>public > js > admin_manage_user.js > clickHandlerForPageButton > [o] userd 1 let \$searchDesignFormContainer = \$('#searchUserFormContainer'); 2 if (\$searchDesignFormContainer.length != 0) { 3 console.log('Search user form detected in manage user interface. Binding event handling logic to form elements.'); 4 //If the \$Query object which represents the form element exists, 5 //the following code will create a method to submit search parameters 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'https://localhost:5000'; 10 let searchInput = \$('#searchInput').val(); 11 let userId = sessionStorage.getItem('user_id'); 12 let token = sessionStorage.getItem('token'); 13 axios({ 14 ... 15 }) 16 .then(function(response) { ... 17 ... 18 }) 19 .catch(function(response) { ... 20 }); 21 }); 22 function clickHandlerForPageButton(event) { 23 event.preventDefault(); 24 const baseUrl = 'https://localhost:5000'; 25 let userId = sessionStorage.getItem('user_id'); 26 let pageNumber = \$(event.target).text().trim(); 27 let searchInput = \$('#searchInput').val(); 28 let token = sessionStorage.getItem('token'); 29 console.log('Checking the button page number which raised the click event : ', pageNumber); 30 axios({ 31 ... 32 }) 33 .then(function(response) { ... 34 ... 35 }) 36 .catch(function(response) { ... 37 }); 38 } 39 //End of clickHandlerForPageButton 40 } //End of checking for \$searchUserFormContainer \$Query object</pre>

Original Codes	Fixed Codes
frontEnd > public > js > admin_update_user.js	
<pre>public > js > JS admin_update_user.js > ... 1 let \$updateUserFormContainer = \$('#updateUserFormContainer'); 2 if (\$updateUserFormContainer.length != 0) { 3 console.log('Update User form is detected. Binding event handling logic to form elements.'); 4 //If the jQuery object which represents the form element exists, 5 //the following code will create a method to submit user role data 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'http://localhost:5000'; 10 //Collect role id value from the input element, roleIdInput 11 let roleId = \$('#roleIdInput').val(); 12 //Obtain user id from local storage 13 let userId = localStorage.getItem('user_id'); 14 //There is a hidden textbox input, userRecordIdInput 15 let recordId = \$('#userRecordIdInput').val(); 16 let webFormData = new FormData(); 17 webFormData.append('roleId', roleId); 18 webFormData.append('recordId', recordId); 19 axios({ 20 }) 21 .then(function(response) { ... 22 }) 23 .catch(function(response) { ... 24 }); 25 }); 26 \$('#backButton').on("click", function(e) { ... 27 }); 28 29 function getOneUser() { 30 31 const baseUrl = 'http://localhost:5000'; 32 var query = window.location.search.substring(1); 33 let arrayData = query.split("="); 34 let recordIdToSearchUserRecord = arrayData[1]; 35 let userId = localStorage.getItem('user_id'); 36 axios({ ... 37 }) 38 .then(function(response) { ... 39 }) 40 .catch(function(response) { ... 41 }); 42 }); 43 44 //End of getOneUser 45 //Call getOneUser function to do a GET HTTP request on an API to retrieve one user record 46 getOneUser(); 47 } //End of checking for \$updateUserFormContainer JQuery object</pre>	<pre>public > js > JS admin_update_user.js > ... 1 let \$updateUserFormContainer = \$('#updateUserFormContainer'); 2 if (\$updateUserFormContainer.length != 0) { 3 console.log('Update User form is detected. Binding event handling logic to form elements.'); 4 //If the jQuery object which represents the form element exists, 5 //the following code will create a method to submit user role data 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'https://localhost:5000'; 10 //Collect role id value from the input element, roleIdInput 11 let roleId = \$('#roleIdInput').val(); 12 //Obtain user id & token from local storage 13 let userId = sessionStorage.getItem('user_id'); 14 let token = sessionStorage.getItem('token'); 15 //There is a hidden textbox input, userRecordIdInput 16 let recordId = \$('#userRecordIdInput').val(); 17 let webFormData = new FormData(); 18 webFormData.append('roleId', roleId); 19 webFormData.append('recordId', recordId); 20 21 console.log(roleId + " : " + roleId); 22 console.log(recordId + " : " + recordId); 23 24 axios({ 25 }) 26 .then(function(response) { ... 27 }) 28 .catch(function(response) { ... 29 }); 30 }); 31 \$('#backButton').on("click", function(e) { ... 32 }); 33 34 function getOneUser() { 35 36 const baseUrl = 'https://localhost:5000'; 37 var query = window.location.search.substring(1); 38 let arrayData = query.split("="); 39 let recordIdToSearchUserRecord = arrayData[1]; 40 let userId = sessionStorage.getItem('user_id'); 41 let token = sessionStorage.getItem('token'); 42 axios({ ... 43 }) 44 .then(function(response) { ... 45 }) 46 .catch(function(response) { ... 47 }); 48 }); 49 50 //End of getOneUser 51 //Call getOneUser function to do a GET HTTP request on an API to retrieve one user record 52 getOneUser(); 53 } //End of checking for \$updateUserFormContainer JQuery object</pre>
frontEnd > public > js > global.js	
<pre>public > js > JS global.js > ... 1 \$('#logoutButton').on('click', function(event) { 2 event.preventDefault(); 3 localStorage.clear(); 4 window.location.replace('/home.html'); 5 });</pre>	<pre>public > js > JS global.js > ... 1 \$('#logoutButton').on('click', function(event) { 2 event.preventDefault(); 3 sessionStorage.clear(); 4 window.location.replace('/home.html'); 5 });</pre>

Original Codes	Fixed Codes
frontEnd > public > js > login.js	
<pre>public > js > login.js > ... 1 let \$loginFormContainer = \$('#loginFormContainer'); 2 if (\$loginFormContainer.length != 0) { 3 console.log('Login Form detected. Binding event handling logic to form elements.'); 4 //If the jQuery object which represents the form element exists, 5 //the following code will create a method to submit registration details 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'http://localhost:5000'; 10 let email = \$('#emailInput').val(); 11 let password = \$('#passwordInput').val(); 12 let webFormData = new FormData(); 13 webFormData.append('email', email); 14 webFormData.append('password', password); 15 axios({ 16 ... 17 }) 18 .then(function(response) { 19 //Inspect the object structure of the response object. 20 //console.log('Inspecting the response object returned from the login web api'); 21 //console.dir(response); 22 userData = response.data; 23 if (userData.role_name == 'user') { 24 localStorage.setItem('token', userData.token); 25 localStorage.setItem('user_id', userData.user_id); 26 localStorage.setItem('role_name', userData.role_name); 27 window.location.replace('user/manage_submission.html'); 28 return; 29 } 30 if (response.data.role_name == 'admin') { 31 localStorage.setItem('token', userData.token); 32 localStorage.setItem('user_id', userData.user_id); 33 localStorage.setItem('role_name', userData.role_name); 34 window.location.replace('admin/manage_users.html'); 35 return; 36 } 37 }) 38 .catch(function(response) { 39 ... 40 }); 41 }); 42 } 43 } //End of checking for \$loginFormContainer jQuery object</pre>	<pre>public > js > login.js > ... 42 let \$loginFormContainer = \$('#loginFormContainer'); 43 if (\$loginFormContainer.length != 0) { 44 console.log('Login form detected. Binding event handling logic to form elements.'); 45 //The jQuery object which represents the form element exists, 46 //the following code will create a method to submit registration details 47 //to server-side api when the #submitButton element fires the click event. 48 \$('#submitButton').on('click', function(event) { 49 event.preventDefault(); 50 const baseUrl = 'https://localhost:5000'; 51 let email = \$('#emailInput').val(); 52 let password = \$('#passwordInput').val(); 53 let webFormData = new FormData(); 54 ... 55 //Regex check for the email to ensure that there is no chance for a attack through the email field 56 const regex = new RegExp('^(a-zA-Z0-9)+(:\\w[a-zA-Z0-9-]+)*.(a-zA-Z0-9)\$'); 57 if (email.match(regex)) { 58 webFormData.append('email', email); 59 webFormData.append('password', password); 60 } 61 ... 62 }) 63 .then(function(response) { 64 //Inspect the object structure of the response object. 65 //console.log('Inspecting the response object returned from the login web api'); 66 //console.dir(response); 67 userData = response.data; 68 if (userData.role_name == 'user') { 69 sessionStorage.setItem('token', userData.token); 70 sessionStorage.setItem('user_id', userData.user_id); 71 sessionStorage.setItem('role_name', userData.role_name); 72 ... 73 //localStorage.setItem('token', userData.token); 74 //localStorage.setItem('user_id', userData.user_id); 75 //localStorage.setItem('role_name', userData.role_name); 76 window.location.replace('user/manage_submission.html'); 77 return; 78 } 79 if (response.data.role_name == 'admin') { 80 sessionStorage.setItem('token', userData.token); 81 sessionStorage.setItem('user_id', userData.user_id); 82 sessionStorage.setItem('role_name', userData.role_name); 83 ... 84 //localStorage.setItem('token', userData.token); 85 //localStorage.setItem('user_id', userData.user_id); 86 //localStorage.setItem('role_name', userData.role_name); 87 ... 88 //localStorage.setItem('token', userData.token); 89 //localStorage.setItem('user_id', userData.user_id); 90 //localStorage.setItem('role_name', userData.role_name); 91 window.location.replace('admin/manage_users.html'); 92 return; 93 } 94 }) 95 .catch(function(response) { 96 ... 97 }); 98 } 99 } else { 100 ... 101 } 102 ... 103 }); 104 } //End of checking for \$loginFormContainer jQuery object</pre>
frontEnd > public > js > manage_invite.js	
<pre>public > js > manage_invite.js > ... 1 let \$manageInviteFormContainer = \$('#manageInviteFormContainer'); 2 if (\$manageInviteFormContainer.length != 0) { 3 console.log('Manage invite form detected. Binding event handling logic to form elements.'); 4 //If the jQuery object which represents the form element exists, 5 //the following code will create a method to submit registration details 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'http://localhost:5000'; 10 let fullName = \$('#fullNameInput').val(); 11 let email = \$('#emailInput').val(); 12 let userId = localStorage.getItem('user_id'); 13 let webFormData = new FormData(); 14 webFormData.append('recipientName', fullName); 15 webFormData.append('recipientEmail', email); 16 ... 17 axios({ 18 ... 19 }) 20 .then(function(response) { 21 ... 22 }) 23 .catch(function(response) { 24 ... 25 }); 26 }); 27 } 28 } //End of checking for \$manageInviteFormContainer jQuery object</pre>	<pre>public > js > manage_invite.js > ... 1 let \$manageInviteFormContainer = \$('#manageInviteFormContainer'); 2 if (\$manageInviteFormContainer.length != 0) { 3 console.log('Manage invite form detected. Binding event handling logic to form elements.'); 4 //If the jQuery object which represents the form element exists, 5 //the following code will create a method to submit registration details 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'https://localhost:5000'; 10 let fullName = \$('#fullNameInput').val(); 11 let email = \$('#emailInput').val(); 12 let userId = sessionStorage.getItem('user_id'); 13 let webFormData = new FormData(); 14 webFormData.append('recipientName', fullName); 15 webFormData.append('recipientEmail', email); 16 ... 17 axios({ 18 ... 19 }) 20 .then(function(response) { 21 ... 22 }) 23 .catch(function(response) { 24 ... 25 }); 26 }); 27 } 28 } //End of checking for \$manageInviteFormContainer jQuery object</pre>
frontEnd > public > js > profile.js	
<pre>public > js > profile.js > ... 1 let \$profileContainer = \$('#profileContainer'); 2 if (\$profileContainer.length != 0) { 3 console.log('Profile page is detected. Binding event handling logic to form elements.'); 4 \$('#backButton').on('click', function(e){ 5 e.preventDefault(); 6 window.history.back(); 7 }); 8 9 function getOneUser() { 10 const baseUrl = 'http://localhost:5000'; 11 12 let userId = localStorage.getItem('user_id'); 13 ... 14 axios({ 15 ... 16 }) 17 .then(function(response) { 18 ... 19 }) 20 .catch(function(response) { 21 ... 22 }); 23 }; 24 25 ... 26 27 } //End of getOneUser 28 //Call getOneUser function to do a GET HTTP request on an API to retrieve one user record 29 getOneUser(); 30 } //End of checking for \$profileContainer jQuery object</pre>	<pre>public > js > profile.js > ... 1 let \$profileContainer = \$('#profileContainer'); 2 if (\$profileContainer.length != 0) { 3 console.log('Profile page is detected. Binding event handling logic to form elements.'); 4 \$('#backButton').on('click', function (e) { 5 ... 6 }); 7 8 function getOneUser() { 9 const baseUrl = 'https://localhost:5000'; 10 11 let userId = sessionStorage.getItem('user_id'); 12 13 // ----- Start of changes ----- 14 let token = sessionStorage.getItem('token'); 15 ... 16 axios({ 17 ... 18 }) 19 .then(function (response) { 20 ... 21 }) 22 .catch(function (response) { 23 ... 24 }); 25 26 } 27 28 } //End of getOneUser 29 //Call getOneUser function to do a GET HTTP request on an API to retrieve one user record 30 getOneUser(); 31 } //End of checking for \$profileContainer jQuery object</pre>

Original Codes	Fixed Codes
frontEnd > public > js > submit_design.js	
<pre>public > js > submit_design.js > @ on('click') callback > @ then() callback 1 let \$submitDesignFormContainer = \$('#submitDesignFormContainer'); 2 if (\$submitDesignFormContainer.length != 0) { 3 console.log('Submit design form detected. Binding event handling logic to form elements.'); 4 //If the jQuery object which represents the form element exists, 5 //the following code will create a method to submit design details 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'http://localhost:5000'; 10 let userId = localStorage.getItem('user_id'); 11 let designTitle = \$('#designTitleInput').val(); 12 let designDescription = \$('#designDescriptionInput').val(); 13 let webFormData = new FormData(); 14 webFormData.append('designTitle', designTitle); 15 webFormData.append('designDescription', designDescription); 16 // HTML file input, chosen by user 17 webFormData.append('file', document.getElementById('fileInput').files[0]); 18 axios({ 19 ... 20 }) 21 .then(function(response) { ... 22 }) 23 .catch(function(response) { ... 24 }); 25 }); 26}); //End of checking for \$submitDesignFormContainer jQuery object</pre>	<pre>public > js > submit_design.js > @ on('click') callback 1 let \$submitDesignFormContainer = \$('#submitDesignFormContainer'); 2 if (\$submitDesignFormContainer.length != 0) { 3 console.log('Submit design form detected. Binding event handling logic to form elements.'); 4 //If the jQuery object which represents the form element exists, 5 //the following code will create a method to submit design details 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'https://localhost:5000'; 10 let userId = sessionStorage.getItem('user_id'); 11 let designTitle = \$('#designTitleInput').val(); 12 let designDescription = \$('#designDescriptionInput').val(); 13 let webFormData = new FormData(); 14 let token = sessionStorage.getItem('token'); 15 ... 16 // regex check to ensure that the designTitle and designDescription does not contain malicious codes 17 let designTitle_regex = new RegExp('^[a-zA-Z0-9 ,]+\$'); 18 let designDescription_regex = new RegExp('^[a-zA-Z0-9 ,]+\$'); 19 ... 20 if (designTitle_regex.test(designTitle) && designDescription_regex.test(designDescription)) { 21 } else { 22 new Notify({ 23 type: 'error', 24 timeout: '6000', 25 layout: 'topCenter', 26 theme: 'sunset', 27 text: 'Unable to submit design file.' 28 }).show(); 29 } 30 }); 31}); //End of checking for \$submitDesignFormContainer jQuery object</pre>
frontEnd > public > js > update_design.js	
<pre>public > js > update_design.js > @ on('click') callback 1 let \$updateDesignFormContainer = \$('#updateDesignFormContainer'); 2 if (\$updateDesignFormContainer.length != 0) { 3 console.log('Update design form is detected. Binding event handling logic to form elements.'); 4 //If the jQuery object which represents the form element exists, 5 //the following code will create a method to submit design details 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'http://localhost:5000'; 10 //Collect fileId value from the input element, fileIdInput (hidden input element) 11 let fileId = \$('#fileIdInput').val(); 12 //Obtain user id from local storage 13 let userId = localStorage.getItem('user_id'); 14 //Collect design title and description input 15 let designTitle = \$('#designTitleInput').val(); 16 let designDescription = \$('#designDescriptionInput').val(); 17 //Create a FormData object to build key-value pairs of information before 18 //making a PUT HTTP request 19 let webFormData = new FormData(); 20 webFormData.append('designTitle', designTitle); 21 webFormData.append('designDescription', designDescription); 22 webFormData.append('fileId', fileId); 23 axios({ 24 ... 25 }) 26 .then(function(response) { ... 27 }) 28 .catch(function(response) { ... 29 }); 30 }); 31}); //backButton.on("click", function(e) { ... 32}); 33 34 function getOneData() { 35 const baseUrl = 'http://localhost:5000'; 36 //Get the fileId information from the web browser URL textbox 37 let query = window.location.search.substring(1); 38 let arrayData = query.split('='); 39 let fileId = arrayData[1]; 40 console.dir('Obtained file id from URL : ', fileId); 41 let userId = localStorage.getItem('user_id'); 42 axios({ 43 ... 44 }) 45 .then(function(response) { ... 46 }) 47 .catch(function(response) { ... 48 }); 49} 50 51 //End of getOneData 52 //Call getOneData function to do a GET HTTP request on an API to retrieve one user record 53 getOneData(); //Call getOneData to begin populating the form input controls with the existing record information. 54}); //End of checking for \$updateDesignFormContainer jQuery object</pre>	<pre>public > js > update_design.js > @ on('click') callback 1 \$('#submitButton').on('click', function(event) { 2 event.preventDefault(); 3 const baseUrl = 'https://localhost:5000'; 4 //Collect fileId value from the input element, fileIdInput (hidden input element) 5 let fileId = \$('#fileIdInput').val(); 6 //Obtain user id from local storage 7 let userId = sessionStorage.getItem('user_id'); 8 //Collect design title and description input 9 let designTitle = \$('#designTitleInput').val(); 10 let designDescription = \$('#designDescriptionInput').val(); 11 12 // ----- Start of changes ----- 13 let designTitle_regex = new RegExp('^[a-zA-Z0-9 ,]+\$'); 14 let designDescription_regex = new RegExp('^[a-zA-Z0-9 ,]+\$'); 15 16 if (designTitle_regex.test(designTitle) && designDescription_regex.test(designDescription)) { 17 //Create a FormData object to build key-value pairs of information before 18 //making a PUT HTTP request 19 let webFormData = new FormData(); 20 webFormData.append('designTitle', designTitle); 21 webFormData.append('designDescription', designDescription); 22 webFormData.append('fileId', fileId); 23 24 // ----- Start of changes ----- 25 let token = sessionStorage.getItem('token'); 26 axios({ 27 ... 28 }) 29 .then(function(response) { ... 30 }) 31 .catch(function(response) { ... 32 }); 33 } else { 34 ... 35 }; 36 37}); //backButton.on("click", function(e) { 38 e.preventDefault(); 39 window.history.back(); 40}); 41 42 function getOneData() { 43 const baseUrl = 'https://localhost:5000'; 44 //Get the fileId information from the web browser URL textbox 45 let query = window.location.search.substring(1); 46 let arrayData = query.split('='); 47 let fileId = arrayData[1]; 48 console.dir('Obtained file id from URL : ', fileId); 49 let userId = sessionStorage.getItem('user_id'); 50 51 // ----- Start of changes ----- 52 let token = sessionStorage.getItem('token'); 53 axios({ 54 ... 55 }) 56 .then(function(response) { ... 57 }) 58 .catch(function(response) { ... 59 }); 60 61}); //End of getOneData 62 //Call getOneData function to do a GET HTTP request on an API to retrieve one user record</pre>

Original Codes	Fixed Codes
frontEnd > public > js > user_manage_submission.js	
<pre> public > js > user_manage_submission.js > \$(onClick) callback > then() callback 1 let \$searchDesignFormContainer = \$('#searchDesignFormContainer'); 2 if (\$searchDesignFormContainer.length != 0) { 3 console.log('Search design form detected in user manage submission interface. Binding event handling logic to form elements.'); 4 //If the \$query object which represents the form element exists, 5 //the following code will create a method to send key-value pair information to do record searching 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'http://localhost:5000'; 10 let searchInput = \$('#searchInput').val(); 11 let userId = localStorage.getItem('user_id'); 12 axios({ 13 }) 14 .then(function(response) { 15 }) 16 .catch(function(response) { 17 }) 18 }); 19 }); 20 }); 21 }); 22 }); 23 }); 24 }); 25 }); 26 }); 27 }); 28 }); 29 }); 30 }); 31 }); 32 }); 33 }); 34 }); 35 }); 36 }); 37 }); 38 }); 39 }); 40 }); 41 }); 42 }); 43 }); 44 }); 45 }); 46 }); 47 }); 48 }); 49 }); 50 }); 51 }); 52 }); 53 }); 54 }); 55 }); 56 }); 57 }); 58 }); 59 }); 60 }); 61 }); 62 }); 63 }); 64 }); 65 }); 66 }); 67 }); 68 }); 69 }); 70 }); 71 }); 72 }); 73 }); 74 }); 75 }); 76 }); 77 }); 78 }); 79 }); 80 }); 81 }); 82 }); 83 }); 84 }); 85 }); 86 }); 87 }); 88 }); 89 }); 90 }); 91 }); 92 }); 93 }); 94 }); 95 }); 96 }); 97 }); 98 }); 99 }); 100 }); 101 }); 102 }); 103 }); 104 }); 105 }); 106 }); 107 }); 108 }); 109 }); 110 }); 111 }); 112 }); 113 }); 114 }); 115 }); 116 }); 117 }); 118 }); 119 }); 120 }); 121 }); 122 }); 123 }); 124 }); 125 }); 126 }); 127 }); 128 }); 129 }); 130 }); 131 }); 132 }); 133 }); 134 }); 135 }); 136 }); 137 }); 138 }); 139 }); 140 }); 141 }); 142 }); 143 }); 144 }); 145 }); 146 }); 147 }); 148 }); 149 }); 150 }); 151 }); 152 }); 153 }); //End of checking for \$searchDesignFormContainer \$Query object </pre>	<pre> public > js > user_manage_submission.js > ... 1 let \$searchDesignFormContainer = \$('#searchDesignFormContainer'); 2 if (\$searchDesignFormContainer.length != 0) { 3 console.log('Search design form detected in user manage submission interface. Binding event handling logic to form elements.'); 4 //If the \$query object which represents the form element exists, 5 //the following code will create a method to send key-value pair information to do record searching 6 //to server-side api when the #submitButton element fires the click event. 7 \$('#submitButton').on('click', function(event) { 8 event.preventDefault(); 9 const baseUrl = 'http://localhost:5000'; 10 let searchInput = \$('#searchInput').val(); 11 let userId = sessionStorage.getItem('user_id'); 12 let token = sessionStorage.getItem('token'); 13 axios({ 14 }) 15 .then(function(response) { 16 }) 17 .catch(function(response) { 18 }) 19 }); 20 }); 21 }); 22 }); 23 }); 24 }); 25 }); 26 }); 27 }); 28 }); 29 }); 30 }); 31 }); 32 }); 33 }); 34 }); 35 }); 36 }); 37 }); 38 }); 39 }); 40 }); 41 }); 42 }); 43 }); 44 }); 45 }); 46 }); 47 }); 48 }); 49 }); 50 }); 51 }); 52 }); 53 }); 54 }); 55 }); 56 }); 57 }); 58 }); 59 }); 60 }); 61 }); 62 }); 63 }); 64 }); 65 }); 66 }); 67 }); 68 }); 69 }); 70 }); 71 }); 72 }); 73 }); 74 }); 75 }); 76 }); 77 }); 78 }); 79 }); 80 }); 81 }); 82 }); 83 }); 84 }); 85 }); 86 }); 87 }); 88 }); 89 }); 90 }); 91 }); 92 }); 93 }); 94 }); 95 }); 96 }); 97 }); 98 }); 99 }); 100 }); 101 }); 102 }); 103 }); 104 }); 105 }); 106 }); 107 }); 108 }); 109 }); 110 }); 111 }); 112 }); 113 }); 114 }); 115 }); 116 }); 117 }); 118 }); 119 }); 120 }); 121 }); 122 }); 123 }); 124 }); 125 }); 126 }); 127 }); 128 }); 129 }); 130 }); 131 }); 132 }); 133 }); 134 }); 135 }); 136 }); 137 }); 138 }); 139 }); 140 }); 141 }); 142 }); 143 }); 144 }); 145 }); 146 }); 147 }); 148 }); 149 }); 150 }); 151 }); 152 }); 153 }); //End of checking for \$searchDesignFormContainer \$Query object </pre>

Demo Screenshot (After)

The screenshot shows a web browser window with a profile page titled "Profile". The page displays a user's name ("rita") and email ("rita@designer.com"). Below the page content, there is a "BACK" button. To the right of the browser window, the browser's developer tools are open, specifically the "Storage" tab under "Session Storage". It shows two items: "user" (with value "rita") and "token" (with value "eyJhbGciOiUzI1NiJzIwMjSzMjIwIkpVCl99eyIpZC1AMfAeCJ1Um9zZS0iInVzZXhCJpK...").

About this interface: User usually visit this interface to check whether their profile data. For large scale system, the interface will even show their

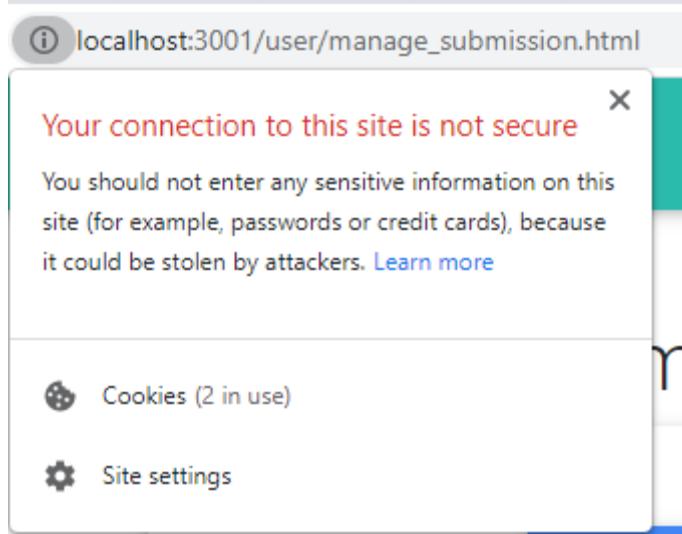
- last login and logout date information

After the code changes, the role_name, token and user_id is now in the sessionStorage instead of localStorage.

Vulnerability 9 – Sensitive Data Exposure (HTTP)

V-03 Sensitive Data Exposure (HTTP)	The current website should use the newer HTTPS encryption as data passed from HTTPS should be encrypted, especially for login which contains sensitive data. This data can potentially be stolen.	Packet sniffers can intercept user's traffic and steal their information. This website uses the normal HTTP protocol which does not encrypt its traffic. However, this requires the hacker to be in the same network as the user to intercept the data and requires some experience.	Recommend changing to HTTPS with an SSL certificate from a Certificate authority (such as Cloudflare) so that user's information will be encrypted during traffic to prevent sensitive information from being stolen.
	Impact - Low	Ease of exploit - Difficult	

Demo Screenshots (Before)



As seen in the screenshot above, Google warns users that the connection is not secure as the site does not have a certified SSL certificate as it still uses HTTP instead of HTTPS

`Content-Disposition: form-data; name="email"`

`robert@admin.com`

`-----WebKitFormBoundaryjgpGgapIrfkJs1ddH`

`Content-Disposition: form-data; name="password"`

`password`

`-----WebKitFormBoundaryjgpGgapIrfkJs1ddH--`

At the Login Page, When the user presses on the “Login” button, if intercepted with Burp, the username and password is shown in the HTTP request as plain text

Original Codes	Fixed Codes
<pre>const baseUrl = 'http://localhost:5000';</pre> <p>Every single endpoint in the frontEnd codes that has a baseUrl pointing to HTTP websites will be switched to HTTPS instead</p>	<pre>const baseUrl = 'https://localhost:5000';</pre>

Demo screenshot (After)

The screenshot shows a browser window with the URL `localhost:3001/login.html`. A security dialog box is overlaid on the page, stating "Connection is secure" with a lock icon, and noting that information is private when sent to the site. It also mentions a "Certificate (Valid)" and "Cookies (1 in use)". The main content area shows a "Login" form with a password field containing "password" and a "SUBMIT" button.

- Public can register as a user
- The database has been **seeded** with user test data.
- `user:rita@designer.com password:password`
normal user role
- `user:robert@admin.com password:password`
admin user role
- Inspect the user table for more information on all the **seeded** user test records
- Use the SQL script `SQLScript_CreateDbAndTables_bee_competition_system` to prepare a new database and necessary tables.

To fix that issue, we needed to switch all the webpages to use HTTPS instead of HTTP where S represents “Secure”. Hence, all the traffic that runs through HTTPS and HSTS (HTTP Strict-Transport Security) enforced such that the requests will not be prone to Man-in-the-middle attacks

Other Code Improvements – Adding Regular Expressions to Check Inputs

Original Codes	Fixed Codes
frontEnd > public > js > user_manage_submission.js (exposts.processLogin)	
<pre>exports.processLogin = (req, res, next) => { let email = req.body.email; let password = req.body.password; try { auth.authenticate(email, function(error, results) { if (error) { //let message = 'Credentials are not valid.'; //return res.status(500).json({ message: message }); //If the following statement replaces the above statement //to return a JSON response to the client, the SQLMap or //any attacker (who relies on the error) will be very happy //because they relies a lot on SQL error for designing how to do //attack and anticipate how much "rewards" after the effort. //such as sabotage (seriously damage the data in database), //data theft (grab and sell), //return res.status(500).json({ message: error }); let data = { message: 'Credentials are not valid' }; return res.status(500).json(data); } else { if (results.length == 1) { if ((password == null) (results[0] == null)) { return res.status(500).json({ message: 'login failed' }); } if (bcrypt.compareSync(password, results[0].user_password) == true) { let data = { user_id: results[0].user_id, role_name: results[0].role_name, token: jwt.sign({ id: results[0].user_id, uRole: results[0].role_name }, config.JWTKey, { expiresIn: 86400 //Expires in 24 hrs }) }; //End of data variable setup return res.status(200).json(data); } else { let data = { message: 'login has failed.' } return res.status(500).json(data); } //End of password comparison with the retrieved decoded password. } //End of checking if there are returned SQL results } } })</pre>	<pre>src > controllers > JS authController.js > processLogin > exports.processLogin 5 const jwt = require('jsonwebtoken'); 6 const validator = require('validator'); 7 8 9 10 exports.processLogin = (req, res, next) => { 11 12 let email = req.body.email; 13 let password = req.body.password; 14 15 > if (validator.isEmail(email)) { ... 16 } else { 17 console.log(`\nregex check unsucessful in backend\n`); 18 let data = { 19 message: 'Unable to complete login' 20 } 21 22 return res.status(500).json(data); 23 } 24 25 } 26 27</pre>
	<p>For all input fields that requires the data to be sent to the database for either querying of results or saving to the database, it must go through a few layers of check before proceeding to enter the database. If no checks are carried out, it makes the system very prone to all sorts of attacks</p>
	<p>To ensure that the above layers of checks are enforced, regular expressions will be introduced for all input fields to test the input before proceeding to enter the database</p>
	<p>If the regular expression for that particular input fails, it will immediately throw an error back to the client for them to re-enter their inputs and not proceed to tamper with the database</p>
	<p>To ensure that this check is done thoroughly, the regular expressions will be checked in both frontEnd codes as well as backEnd codes so that in any case if the frontEnd codes fail to check, there is always a fallback checker which is the backEnd</p>

Other Code Improvements – Wrong Redirection

Original Codes	Fixed Codes
<p>frontEnd > public > user > update_design.html</p> <pre><div class="dropdown-menu dropdown-default" aria-labelledby="navbarDropdownMenuLink-333">SubmitManage submission</div></pre> <p>Before, it redirects to a empty page called submit-design.</p>	<pre><div class="dropdown-menu dropdown-default" aria-labelledby="navbarDropdownMenuLink-333">SubmitManage submission</div></pre> <p>After, it successfully redirects users to submit_design.html now</p>

Other Code Improvements – Code Reusability

backEnd > src > middleware > userController.js

```
function checkUserID(userId, decodedId) {
    console.log("-----");
    console.log("Running checkUserID function inside userController.js");
    console.log("-----");
    console.log("userId: " + userId);
    console.log("decodedId: " + decodedId);

    if (userId != decodedId) {
        console.log("-----");
        console.log("UserID does not match Token ID");
        console.log("Query/Upload can only be done with the same Token ID!");
        console.log("-----");

        userId = decodedId;
        console.log("userID changed to Token ID");

        return userId;
    }

    return userId;
}
```

We have noticed that there was a repeat function that is being placed inside the functions which can take up a little time to re-write. Hence to improve code reusability, we have decided to create a function for this, such that when you pass in the values, it will correctly pass back the values required.

In this example, when you pass in the userId retrieved from the sessionStorage and the decodedId from the token, the function will check to see if the ID matches, if it does not match it will correct it and send the right value back to be used.

As this code does the same for all authorization checks, putting it in a function and just calling it will increase code reusability

[Other Code Improvements – Salted Passwords](#)

backEnd > src > controller > authController.js

```
bcrypt.genSalt(saltRounds, (err, salt) => {
  bcrypt.hash(password, salt, async (err, hash) => {
```

To Improve confidentiality as well as security, we have decided to salt all the passwords for an extra layer of security.

user_password
\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6
\$2b\$10\$QLkBkffqSNaGQrNfRa.yTusM7HJVg6zZc.4c5syGOERxxXXQm3UzS

As shown in the database screenshot above, these passwords are only hashed and are not salted, hence they are all the same.

142	hashtest	hashtest@gmail.com	\$2b\$10\$sf.AHTWCsRJCS1N7AQs2E.fan0BOCK1Y2/UNYICsXSOfD9WxtYMC
144	hashtest2	hashtest2@gmail.com	\$2b\$10\$4F/M6zw9Q93ir5AISC6wUekCYJktRDzDh4Weh7vmjHTbnVEk3R6cC
145	hashtest3	hashtest3@gmail.com	\$2b\$10\$PbZ2dQlkxHUBapVmsjO.xeEDN/p40kzPQ5LSSOc38MPdH6O292Fc6
146	hashtest4	hashtest4@test.com	\$2b\$10\$s0/4qjE4dcGWz.mOBpbhT.PEPWkt4rYw/A6So0bIpDxz6NWxy7kqi
147	hashtest5	hashtest5@test.com	\$2b\$10\$fj1H42rwNJT//ELALOq7Zegp6KqgT/YDYmpbbhSiWYG3QCB1zdsae
148	hashtest6	hashtest6@test.com	\$2b\$10\$pDjMQ4uYTawn40pUtbnqn.jauK/h047.FaDI28/h2GNYjTRg9xRZa
149	hashtest7	hashtest7@test.com	\$2b\$10\$aQeAdkCBh.60s8oD6Xtth.HTy2rcKt1bjkw5TAJid8fxnd3PbSwq6
150	hashtest8	hashtest8@test.com	\$2b\$10\$OhCKYqA8nmMfdmSfOqXAj07dRbJ4/oJkP7O4C3i76qVOcPnAsm1bi
151	hashtest8	hashtest9@test.com	\$2b\$10\$pwAQ.OY3cy6QPLO1pRNGMeaOafJzmZm4D5NoxFQ0X7vm6otWgRSGC
152	hashtest9	hashtest9@gmail.com	\$2b\$10\$WmSIE9B/uG2CxdwXJubUz.LQhDNytXeZVIHvdD4eWhS5BtPz3y/bu
153	hasht...	hashtest10@gmail.com	\$2b\$10\$wugeCnj7hFmS9eLJAQq8luBVJGUbT/yyKR5YmqZqzG4yuWgSIYsZ.

In the new password creation process, the passwords are salted and kept in the database as seen above. In this example, all the hashtest users created have the same password: ‘Passw0rd’ entered, but their hashed and salted values make them appear differently inside the database table.

-- End of Report --