1 Testing

1.1 Introduction

This chapter describes the testing that has been undertaken for the application. This chapter is presented in two sections:

1. Functional Testing
2. User Testing

Functional testing is a type of software testing whereby the system is tested against the functional requirements. The application is tested by inspecting if the actual output for a given input corresponds with the expected output. The tests will be based on the requirements for the app. The results of functional testing would indicate if a piece of software is functional and working, but not if the software is easy to use. Vice versa, the user testing would then look for the indication that a piece of software is easy and intuitive for the user.

1.2 Functional Testing

This section describes the functional tests which were carried out on the app. These functional tests can be categorised as:

- Login/Registration – such that when a user login or registering anew, they would be routed to the correct page and given their respective roles.
- Navigation – the graphical elements that allow user to navigate through the app.
- Calculation – equation to compliment the task shown in the app such as addition, subtraction, division and/or multiplication.
- CRUD – the four basic method of storing data inside a persistent storage.

Functional testing generally uses a Black Box Testing technique which means that the internal logic of the system being tested is not of interest to the tester. The tester is only interested in whether the actual output agrees with the expected output.

1.2.1 Login/Registration

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1.	Login with an incorrect username/password	Incorrect username/password	Error message(s) appear for explanation	Error messages prompted	Function working fine.
2.	Login with a correct username/password	Correct username/password	Logged in as user	User(s) is linked to their respective page	All working well. Admin and User are linked to a separate homepage due to their roles.
3.	Login with a remember me checkbox ticked	Ticked checkbox to remember username and password	Kept user logged in	User is kept logged in	No problem here.
4.	Register with the same e-mail address and/or username	Same Email-address and/or username as another account	Prompt error message(s)	Only error message came up for email-address	Needs to specify the input as unique validation in the controller.
5.	Register with an incorrect confirm password	Input incorrect confirm password	Prompt error message	Error message prompted	Working smoothly. No problem here.

6.	Register correctly	Correct inputs	Link user to homepage, logged in	Redirecting loop issues	Needed to add role to user when registered.

1.2.2 Navigation

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1.	Click on logo to link to home	Mouse is clicked on the logo	The user is directed to the homepage	User has been respectively routed to their homepage	All working as intended for both roles.
2.	Go to requests page	Clicked on the button on homepage	Redirect to art request page	User is redirected to the page	No problem, all good.
3.	Go to profile page	Clicked on the navbar dropdown	Direct to user setting page	Directed to the user setting	All good.
4.	Go to create requests	Clicked on the create button	Direct to create request page	Directed to the create request page	Working as intended
5.	Go to artist profile	Clicked on the user dropdown in the navbar	Direct to artist profile	Directed to the artist profile	
6.	Navigate using the sidebar	Click the icon in the navbar	Side menu should appear	Side menu appeared on the right side	Maybe changing of the background colour would be nice
7.	Navigate to see each request	Click on the explore dropdown	The navigation should take user to the	The navigation has redirected	

		in the navbar	request page	to the request page	

1.2.3 Calculation

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1	Deadline calculation	Carbon class is used to calculate for number of days to the deadline date.	No. of days left till deadline	No. of days left till deadline	Calculation needs to take account for past deadline request(s).
2	Last seen user	Carbon class is used to calculate how long user has been offline	User's last seen in minutes/hour/days	User's last seen in minutes/hour/days	
3	Notification to alert	Show number of notifications received	No. of notifications received by the user		

1.2.4 CRUD

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1.	Create request	Insert into Request WHERE Title, duration, prices = the given input	A request should be created and displayed on the site	A request is created and displayed on the site	
2.	Store request	Insert into Request WHERE Title, duration, prices = the given input	A request should be stored and displayed on the site	A request is stored and displayed on the site	

3.	Edit request	Update Request WHERE inputs = new inputs	The request should be updated	The request is updated	
4.	Delete request	Delete FROM Request WHERE conditions	The request should be deleted	The request has been deleted	
5.	Store image	Insert into Image WHERE image = image location	The image should be stored and display as user image	The image is stored and displayed as user image	
6.	Edit profile image	Insert into User WHERE image = image location	The image should be stored and display as user image	The image is stored and displayed as user image	
7.	Edit profile	Update User WHERE conditions	The profile should be edited and updated	The profile has been successfully updated	
8.	Store favourites	Insert Into User_Favourite WHERE user_id = user, favourite_id = request_id or artist_id	The favourite of the artist profile or request should be added		
9.	Remove favourites	Delete FROM User_Favourite WHERE conditions	The favourite should be removed		
10.	Add roles	Insert Into User_Role Where user_id = user, role_id = role	Role should be added once a user has registered	Role has successfully added to the user	
11.	Edit role(s)	Update User_Role WHERE conditions	Role should be updated	Role has been updated	
12.	Remove roles	Update User_Role	Role should be removed	Role has been removed	

		WHERE conditions			
13.	Edit User status				
14.	SQL injections	SELECT * FROM users	A request posted as normal	A request is posted as intended	The form validation has been implemented to prevent these injections

1.3 Discussion of Functional Testing Results

Through the functional testing of the site, every aspect of the site has been tested with all possible inputs to check for errors, malfunctional anomalies with some function actually has caused an issue, for instance, the registering a new account. The freshly created account would be stuck in an eternal redirecting loop, which in order to resolve the issue, I need to assign them a role whenever a new user has sign up with the site.

1.4 User Testing

1.5 Conclusion

The testing really served a great opportunity for inspection of the system whether they are functioning as they should, or a tweak is required to better improve the site. Which there were some bugs occurring in the system but with the pre-test, it really helped the developers to identify the faults in their system and quickly solve them or search for an alternative to their original ideas.