Code Testing Plans and Results

URXperience - Connecting UR World

ENSE 400/477 Winter 2023

Abraham Murgerwa

Dhruv Modi

Dinesh Dalip

Favor Fasunwon

Table of Contents

Executive Summary

This test report provides a high-level overview of the testing process which includes test plans, cases, and results which are used to ensure the URXperience application is scaled to the specified requirements for meeting users expectations. The test plans outline the approach and method in which specific functionalities were tested. Test cases cover a range of scenarios that users will encounter while navigating the app. Test results are either pass, fail or unstable depending on their outcomes as per the testing steps taken to ensure they work. Overall this documentation demonstrates the thorough testing process of the URXperience application ensuring it meets standard user-friendly qualities.

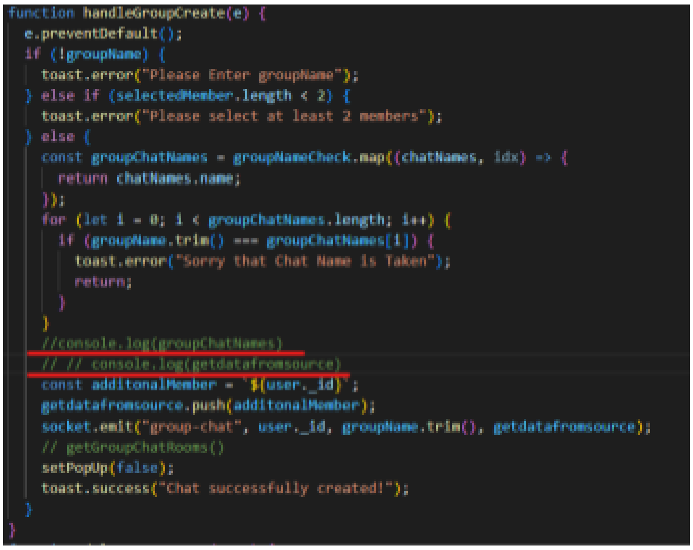
Keywords*: Functionality, Security, Performance, and Integration*

Introduction

Part of the Software Development Life Cycle is testing and integration and this report is a testing documentation that ensures URXperience meets required quality standards. This report will contain comprehensive records of the overall testing process including: test plans,test cases, and results. Through this documentation the testing process will provide a concise outline of the methods used in validating URXperience functionality, performance, and security. This document aids the process of creating URXperience, making it a high-quality application which is free from defects and meets the user's requirements.

Test Plans and Directives

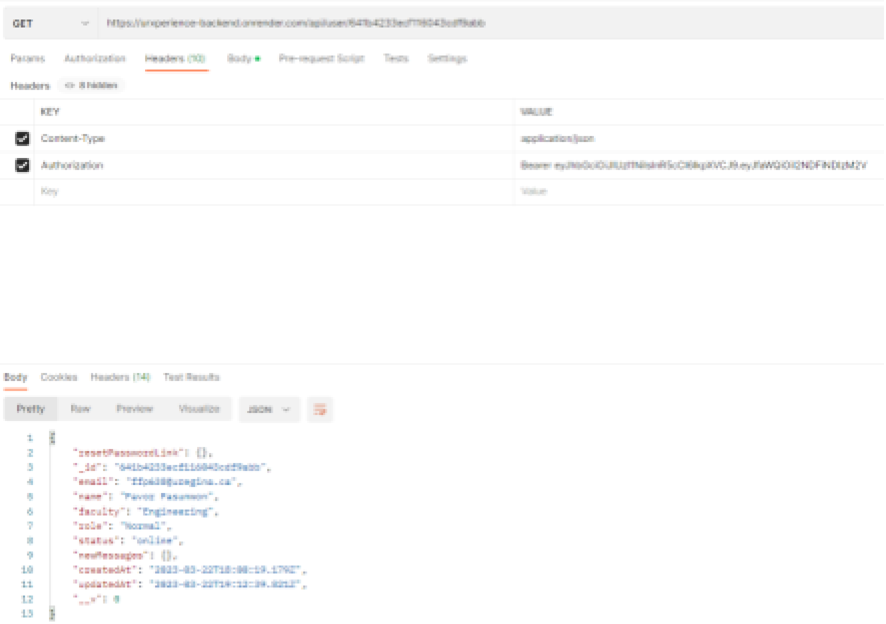
The directives of the test cases, which can be found in the table below, is to provide an analysis of how specific features in the application performs. The plan for testing these features is through the principle of white box testing which goes over the internal structure (code itself) of URXperience verifying its input, output, and overall data flow. The methods used for this testing technique include: functional testing, performance testing and security testing. The concentration for functional testing is integration testing which was achieved through manual efforts. The tools that were used to perform integration testing were “console.log()” - (Front and back-end testing) and “Postman" (API endpoint testing).



```
function handleGroupCreate(e) {
  e.preventDefault();
  if (!groupName) {
    toast.error("Please Enter groupName");
  } else if (selectedMember.length < 2) {
    toast.error("Please select at least 2 members");
  } else {
    const groupChatNames = groupNameCheck.map((chatNames, idx) => {
      return chatNames.name;
    });
    for (let i = 0; i < groupChatNames.length; i++) {
      if (groupName.trim() === groupChatNames[i]) {
        toast.error("Sorry that Chat Name is Taken");
        return;
      }
    }
    //console.log(groupChatNames)
    // // console.log(getdatafromsource)
    const additonalMember = `${user._id}`;
    getdatafromsource.push(additonalMember);
    socket.emit("group-chat", user._id, groupName.trim(), getdatafromsource);
    // getGroupChatRooms()
    setPopUp(false);
    toast.success("Chat successfully created!");
  }
}
```

Figure 1: console.log() used for checking data properties

Figure 1 provides an illustration of utilising “console.log()”. With the visual aid of a browser’s inspection tool, console log was used with every feature’s functional implementation ensuring the right data is being passed before integrating one feature to another. For example, this piece of code in figure 1 shows how console log was used to check if a group chat name was retrieved correctly from the input field provided.



GET https://urxperience-backend.onrender.com/api/user/641b4233ecf116043cdf9abb

	KEY	VALUE
☑	Content-Type	application/json
☑	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NDFiNDIzM2V
	Key	Value

```
{
    "resetPasswordLink": {},
    "_id": "641b4233ecf116043cdf9abb",
    "email": "ffp438@uregina.ca",
    "name": "Favor Pasumon",
    "faculty": "Engineering",
    "role": "Normal",
    "status": "online",
    "newMessages": {},
    "createdAt": "2023-03-22T18:08:19.179Z",
    "updatedAt": "2023-03-22T19:12:39.821Z",
    "__v": 0
}
```

Figure 2: Testing API endpoints with Postman

Figure 2 shows the testing of API endpoints using Postman. Postman is a platform used for functional and integral testing on API endpoints. Postman was used to test every API endpoint in URXperience ensuring its functionality and performance were reliable considering the manipulation of sent and received data. For example, figure 2 illustrates a GET request where the headers (content-type and authorization) are present to imitate a browser’s local storage showing how the stored token is used to retrieve a user’s information.

Performance testing was achieved by categorising specific features' response time and determining if they needed to be faster or slower. Functions in JavaScript such as “setTimeInterval ” were used to organise the performance testing for specific functionalities; ex: the time interval it takes to create a group chat (it was initially set to check the database every 5 secs to determine if a group was created, however it was concluded 5 secs was too long of a response time, and was later cut to a max of 3 secs).

Security testing was used in cases of potential vulnerabilities when it came to authentication mechanisms, and page access. The process of security testing was achieved manually by using valid and invalid login credentials to determine how the application responded.

Test Cases and Results

Below is a table which shows each test case performed on the URXperience application.

Abbreviations to take note of are Fn - Functionality, S- Security , and PF - Performance.

Test Case Type	Description	Test Step	Expected Result	Status
Fn - Login page	Input fields should accommodate user's credentials	Input email and password	User should be logged in and directed to dashboard	Pass
Fn - Register page	Input fields should accommodate user's information	Input email, name, password, and faculty	User is directed to verify their account via the email provided	Pass
Fn - Forgot password	Input field should only accept emails which is present in the DB	Type in a uregina email	User is sent a reset pass link and enters a new password	Pass
Fn - Recipe details	Ensure the show details button of recipe displays the clicked recipe info	Click on the show details button either on the dashboard or food page	Link redirects to a page of the clicked recipe with its details	Pass
Fn - Custom recipe search	Get Recipes by selected ingredients from a list of the ones provided	Type or select available ingredient such as olive oil, flour, eggs, and chicken	4 recipes which are majorly composed of but are not limited to the ingredients typed in.	Pass
Fn - Ping points on map	Ping points on map should reference to the restaurant information on the side panel	Click on a ping point	Scrollable side panel should scroll to the result of the specified restaurant that was clicked.	Unstable
Fn - Filter Events	Calendar with its	Select a date	Display event(s) with	Pass

by Date	dates will be present where specific dates have an event set	where there is an event	its details for the selected date	
Fn - Post Events	An admin can fill the prompted fields to create an event all users on the application can view	Input a title, description, date, time (from and to) and hit the add event button	A notification of a successfully created event is displayed and a log of all created events from that particular admin will be shown on the side as well as on the events landing page	Pass
Fn - Post Complaints	Input fields should accommodate user’s information for a complaint	Input location, image (optional), and detailed description	Submitted complaint should be displayed on the side container as well as the admin’s panel with a resolve button	Pass
Fn - Resolve Complaints	Each complaint submitted by a user should be available on the admin panel to be resolved	Click on the resolve button	Complaint should be removed from the list of present complaints and an email should be sent to the user who created the complaint	Pass
Fn - Alter profile	Input fields should display the user's current information (email and name). Password field should be empty. Changeable fields are name and password	Input a new name and password, then click the update button	Get a notification that profile was successfully changed and user can start using the new password to login	Pass
Fn - Joining Rooms	The faculty the user chose should be a joinable room as well as all the members on the application	Click on the faculty room and any member on the side panel	The faculty room should be joinable and the message field should change to that of the faculty and vice versa with the members online	Pass
Fn - Creating Group chats	A user can create a group chat with any members currently on the application	Hit the add button, type in the group name and select the members to be a part of the group chat	Group chat is created and only those selected will be able to view and interact in the room.	Pass

Fn - Delete Group chats	Only the creator of a group chat can delete the group chat	Hit the trash icon button to delete the group chat	A notification of a successful deleted the group chat is displayed and the user is redirected to their faculty room	Pass
Fn - Logout	User can exit application and ensure session has ended	Hit the logout button on the navbar	User's status is changed to offline. User is then redirected to login page	Pass
Fn - Delete account	Pop up will prompt User's account to be deleted with all available trace except messages	On profile page, click the delete account button and click proceed	Any group chat and complaint made by the user is deleted as well as their account and the screen is redirected to the login page	Pass
PF- Messages	Sending a direct or mass message to a group chat or faculty room.	Join a room and type into the message field and hit enter	Message is instantly displayed in message form container and received by recipient(s)	Unstable
PF- Logging in	Retrieve users credentials and compare with that in the DB	Input login credentials	Process should take no more than 2 secs	Pass
PF - Creating Group chat	A user creates a group chat with any members currently on the application	Fill all required fields and hit the create button	Group chat should not take more than 5 secs to be displayed	Pass
PF - Posting a complaint	Each field input will be sent to the DB while also being displayed in the side panel	User fill all required field to post a complaint, image included	Post should not take more than 3 secs to be rendered on the side panel	Unstable (takes a couple secs to appear)
PF - Resolving a complaint	Hitting the resolve button should instantly clear the complaint from the list	Admin hits the resolve button	The specific complaint should be gone as its resolve button is clicked	Unstable (takes a couple secs to disappear)
S - Confirm Password	Places which allows password change has a confirm password input field present	Type in a password and type it again to confirm	Passwords should match and authentication should redirect to the next page	Pass
S- Password	Verify password rules are working	Enter a password	Password is accepted if it is 8 chars long and	Pass

		matching the indicated requirements	contains 1 digit	
S- Registration	Verify only uregina emails	Fill the required fields on the register page	Receive a link on uregina email provided which redirects to a page to activate account	Pass
S- Pages	Verified accounts can log in and access the other pages on the application	Activate account and log in with credentials	User gets redirected to dashboard and can now access all other pages	Pass
S - Messages	Messages are only seen from recipients whom the messages were directed to	Join a room and type into the message field and hit enter, repeat for another room	Each message should not be displayed in the same room, they should be in the directed rooms independent of each other	Pass
S - Admin route	Admins have separate functionality on events, complaints, and account deletion	Change specific user to admin in the DB, and log in	Only Admins should see buttons to Post events, and resolve complaints. Delete account (only visible for students)	Pass

Table 1: Test cases with the steps and results

Conclusion

In conclusion, the testing performed in this report provided valuable insight towards creating a well versed code base. Through functional, performance, and security testing, areas of success and improvement were highlighted prompting for more robust testing. Although not all cases tested passed, this testing process was a success as it ensures most functionalities meet a standard quality requirement which is to provide a rounded experience that is well-received by users.