Research Mate

Software Test Plan

CSCI-P465/565 (Software Engineering I)

Project Team

<Gulshan Madhwani>

<Jayendra Khandare>

<Shubham Basu>

<Xinquan Wu>

---

1. Overview

1.1 Test Objectives

In sprint one, we develop login module of ResearchMate, and we host it on silo.soic.indiana.edu. This test is to test the functionality of the login module, to check it's working or not.
Unit Test of ResearchMate[Sprint 1]:

1. Create an account.
2. Verify the email address.
3. Login the account.
4. Log off from the account.

In sprint two, we refined login and registration module. We have developed features like user profile and have extended database so that it now includes papers published and their information. We are permanently hosting them on silo.soic.indiana.edu:54545 unless testing and debugging is being performed.

This test is to test the functionalities of the extended database and verification of the user profile page.

Unit Test of ResearchMate[Sprint 2]:

1. Session string management.
2. User profile setup.
3. Extended database verification.

1.2 Test Environment

We are using 3 operating systems to do the testing which are as following:

1. Windows 10 64bit;
2. Ubuntu 64bit;
3. OS-X 64bit;

All use Chrome browser to connect to the web application we have developed. Furthermore, as a requirement this project must be hosted on silo server provided by SICE, hence it is hosted with Jayendra's account.

1.3 Test Personnel

1. Gulshan Madhwani using Windows 10 64bit with Chrome;
2. Jayendra Khandare using Ubuntu 64bit with Chrome;
3. Shubham Basu using Windows 10 64bit with Chrome;
4. Xinquan Wu using OS-X 64bit with Chrome.

At this moment, we have finished with connectivity issues which we had when sending a mail through Gmail. For that, we created an account in gmail with se.researchmate as username.

1.4 Acceptance Criteria

Sprint 1:

Successfully creating an account, receiving an email to verify the email address, and logging in with the account. Only getting through all these steps that will be consider acceptable. All group members carry out the tests separately.

A complete module of the project will be satisfying if the data provided by the user is accepted and stored at in MongoDB database hosted at the server(silo.soic.indiana.edu).

Sprint 2:

Adding data about the user successfully, verifying the account, maintaining and managing session string using cookies, user being able to add himself in a group, user being able to add information about the papers he has published.

A complete module of the project will be satisfying if the user is able to see and change his personal information and he should be also able to upload information about his research papers and join groups dedicated to various topics.

1.5 Noted Omissions

- Group creation criteria and accessibility.
- Load testing for auto increment function that we have implemented separately for userIDs.

2. Test Cases

Sprint 1:
Number: 001
Name: Checking support provided by silo server
Description: This test includes checking whether silo server system supports the tools required for MEAN stack development.
Initial Conditions: No prior test case is needed for this test.
Input Data: Student username and passphrase for accessing any system of IU.
Specifications:

1. **Get access to a silo account.**
2. **Check whether it has MongoDB support.**
3. **Check whether it has Node.js support.**
4. **Check whether it has NPM package manager support.**

Procedure: We used Jayendra Khandare's account to access silo/burrow and do the necessary testing. The testing includes running some basic bash commands like

- **mongod –version**
- **npm –version**
- **node --version**

Number: 002
Name: Hosting the server code on silo
Description: For this test, we had to develop some basic server code using node and run on silo. After that, we had to manage some connectivity issues since 'mongod', which is server for mongodb doesn't give enough rights to students.

Initial Conditions: The access to silo is required for this test. The access is given by default to all students of SICE.
Input Data: server application to be hosted on silo
Specifications:

1. Start an instance of mongod through student account.
2. Run the server code developed by the team.

Procedure: Whenever you run a server on silo, you must reserve a port for your website. This procedure is done through the server code. For this project, we chose 23456 as the port. Hence, the address becomes http://silo.soic.indiana.edu:54545. As the server code is not running all the time, the availability of this address is ambiguous.

Number: 003
Name: Register & Login test
Description: Create an account, receive an email to verify the email address and account, login with the account created, log out with the account.
Initial Conditions: Satisfying test number 002.
Input Data: First name, Last name, User name, Email address.
Specifications:

1. Create an account;
2. Receive an email from se.researchmate@gmail.com with a link to verify the email address;
3. Login with the account;
4. Log out with the account.

Procedure: The link cannot be provided as we are hosting the project at silo server and don't have any idea how to perform the tests on this environment. We have contacted IU knowledge base regarding this. The will provide us with a definite answer in the next week.

Sprint 2:

Number: 004
Name: Session String Management and Verification
Description: Whenever a user tries to login, after verifications and validations, a random string is returned to the user and it is stored in cookies. This string is used hereafter for security checks.
Initial Conditions: Satisfying test number 003.

Input Data: session string.
Specifications:

1. When login() function is called, check username and paossword.
2. After verification, assign a randomly generated session string in user collection(/table) and send the same string to front end.
3. Front end will store this string in cookies.
4. In each sub-sequential transaction with the server, this string will be provided and server will decide based on that string whether to allow access.
5. This string will be required for all transactions for everyone, admins will have a different session string which they can change(if required).

Procedure: For sessionString generation, we are using a module called randomString. This string is generated and stored in the database when the user logs in. This string is stored in cookies file, hence the user doesn't have to worry about keeping the track of this string. It is updated each time the user logs in. It is stored in database which helps the server to find the userID which is required for most of the functionalities.

Number: 005
Name: Data packet verification and validation
Description: The data packets which are sent to and fro while communicating with the server from front-end have to be checked thoroughly if they have the data required for certain function. We are using JSON format for the project, hence it has become easy to keep track of the data flow and apply necessary validation parameters.
Initial Conditions: JSON formatted data packet.
Input Data: JSON data packets.
Specifications:

1. Whenever any kind of data is sent towards server, it is checked at front-end before sending whether it is in JSON format.
2. Whenever any kind of data is received by the server, server checks again whether data is in JSON format.
3. If the data is not in JSON format, the server rejects such data packet, because it possibly could be a query injection attempt.

Procedure: We didn't stick to any module or package for this validation, because in MEAN stack environment, it is a usual practice to change the signatures of functions and their parameter values often. Although it is good for security, it requires you to

change the code every time to accommodate with the shift. In order to avoid that and make it a robust and steady model, we checked the format for each packet before sending[at front-end] and after receiving [at back-end]. Hence, there are no fixed function for that.

Number: 006
Name: User profile maintenance
Description: Every user has a user profile which he can modify as much as he likes. This profile contains his personal information about his research publications, the groups he is following, comments he has made, people he is following and the people that follow him. This information should not be immutable, hence we created functions to access and change this information.
Initial Conditions: user is verified and his session is validated.
Input Data: For accessing, session string. For changing, session string and JSON data.
Specifications:

1. Once the user logs in, he will get the session string that is allocated to him.
2. Using that session string, he can call functions which will give him his personal information.
3. If he wants to change his information, he just needs to be logged in and press submit buttons wherever he feels like changing.
4. Once the user requests to change the data, he provides session string which is needed to locate his ID and JSON data for the elements he wants to change, this data is updated in the respective fields required by user.

Procedure: At this moment, we have used one function each for accessing and modifying user data. These functions accept session string as validation tokens and locate the userID from users collection(/table), this userID is always required if we want to make any changes. At this stage, we have not included data from all the collections(/tables) in user profile, but that will be done as soon as we populate the database.

Revision History

Revision	Date	Change Description
Sprint 1	10/01/2017	Test plan for sprint 1: login and sign up module

Sprint 2	10/15/2017	Added test objectives ,3 test cases, and acceptance criteria specific for Sprint 2

Page Author/Creator: Adeel Bhutta
Last Modified: 8/23/2016