MCI project Testing plan

Team number: 24
Project Title: MindSpace

Revision history

Revised on	Version	Author	Revision Summary
15 May 2021	0.1	Yingyao Lu	Add test execution
15 May 2021	0.2	Eric Ma	Update introduction and test strategy
15 May 2021	0.3	On Ki Ng	Update overview and add appendix II
16 May 2021	0.4	Eric Ma, On Ki Ng	Update test execution and test strategy
18 May 2021	1.0	Yingyao Lu, Jason Ng, Eric Ma	Update authentication test cases; Update http request test; Update backend server test cases

1. Introduction
    1.1. Project Overview

The purpose of this document is to describe a plan to test the project MindSpace, it includes the scope, test strategy, test execution, schedule, and risks, the objective of the test is to test if the application could run as we expected.

Project MindSpace is a mobile application that aims to help users enhance their emotional awareness and provides users resources to cope with their emotional problems. It mainly provides four functions. Firstly, MindSpace reads and stores the input data provided by our clients, the input data are emotions records of users including user id, emotion, posted date, trigger keywords and post content. Secondly, MindSpace presents the user emotion records in various ways in 'Home', 'Explore', and 'Timeline' pages to help user tracking their emotions. Thirdly, MindSpace provides an 'insight' page for user to make reflections, the content could be access in calendar of 'Timeline' page after submission. Lastly, MindSpace provides 'Strategies' page for users to develop their emotional awareness and formulate strategies to deal with their emotional problems.

To support the functioning of the mobile application, our project also involves in developing a Django server running with MySQL database to process data sending to and receiving from the mobile application. We designed database architecture and data flow using sample data provided by our clients. The goal of server development in our project is to create an API interface to: (1) manage user authentication, (2) manage and allow authenticated users to retrieve their past emotions records, (3) save insight records (i.e. reflections about past emotions) submitted by a user to the database, and (4) allow authenticated users to retrieve their submitted insight records.

    1.2. Scope of the test plan document

Our project does not involve development of NLP AI and retrieval of social media data and therefore the scope of this test plan document only covers tests with sample data of NLP output provided by clients.
This test plan only tests if the whole system working on the mobile application simulator and server running on a local machine environment. It does not cover tests running on real mobile devices and production server environment.

2. Test strategy
    2.1. Test scope

    The objective of tests covered in the current testing plan is to ensure sample data provided by clients can be successfully loaded into the backend database, which can then be retrieved by the mobile application for: (1) presenting the occurrence frequency of an authenticated user's past emotions on the 'Home' page, (2) presenting a chosen emotion's occurrence frequency and its trigger keywords frequency within a specified weekly time range on the 'Explore' page,

(3) saving insight records on the 'Insight' page and (4) presenting the occurrence frequency of emotions and the submitted insight records on the 'Timeline' page

2.2. Requirements:

The requirements of tests covered in the current testing plan are: (1) to test if a user can be successfully register a user account with email and password or with a Google user account, (2) to test if a user can be successfully login with the created user account and receive a token generated by the backend server for future authentication, (3) to test if the sample data provided by clients can be successfully imported into database in the backend server and mapped with testing user accounts, (4) to test if data of emotion and triggered keywords of an authenticated user can be successfully retrieved, and can only be accessed with a valid token received in the previous login, (5) to test if the insight records submitted by an authenticated user can be successfully sent to the backend server and saved into the database, (6) to test if the insight records submitted by an authenticated user can be successfully retrieved, and can only be accessed with a valid token received in the previous login, (7) to test if the data retrieved from server can be successfully used to update the 'Home', 'Explore', and 'Timeline' pages Please continue to edit the above paragraph

2.3. Testing report:

Please see appendix I.

2.4. Test assumptions

The application requires login to access functions, we assume all users have a google account or email account so that they could use our application. A demo user has been created for internal debugging.

All tests on the Django backend server are conducted with the assumption that the testing computer has a Docker of version 3.3.1(63152) installed and at least a 10GB storage available for installing the project dependencies required by the server.

A Google Firebase project has been created with two testing user account created. The firebase API key, and the two testing user accounts' UIDs are set as the environment variables FIREBASE_CLIENT_API_KEY, U1_UID and U2_UID defined in the "docker-compose.yml" file in the "/MindSpaceApi" directory.

Backend server is running with the command "Docker-compose up" and the mobile application simulator is running with the command "ionic serve"

3. Test execution

3.1. Defect testing

Defect tests will be taken throughout the whole development process to find out if the application performs correctly. A defect is an error in coding which causes incorrect or unexpected results from the application which does not meet actual requirements.
When a tester executes the test cases, and come across such test results which are contradictory to expected results. This variation in test results should be is recorded to convey information to developers to have a better understand about a defect.
When a defect exposed, the tester shall record defeat information as follow:

- A unique defect id will be given to identify every defect.
- Version of the application in which defect was found.

- Detailed defect description about the module in which defect was found.
- Steps along with screenshots that reproduce the defects in certain circumstance.
- Detected by tester who raised the defect, and name of the developer who fixed it.
- Severity and Priority of the defect in term of a High/Medium/Low to describe the impact of the defect on the application and urgency at which the defect should be fixed respectively.

3.2. Unit testing

Unit testing for different pages in mobile app will be tested as follows:

- Login: Test if user will be redirected. After user entered correct email and password in 'login' page, user should be redirected to 'home' page to access their data.
- Top five emotion circles and all emotion list in 'home' page: test if the size of circles changes by the emotion frequency change. Test if the colour of circles appears as the assigned emotion type. Test if the emotion circles and text show top five emotion correctly. Test if the list shows correctly with its frequency. We will change a test user's data to see if the app shows result as we expected.
- 'Explore' page: test if the page able to show different emotion by clicking next and previous arrows.
    - Weekly charts of emotion: test if the chart shows emotion data correctly as shown date and switch weekly result by clicking next and previous arrows.
    - Trigger keywords list: test if the trigger keywords update when changing to different week and different emotion.
- 'Insight' page: test if the text box allow input. Test if the 'add awareness' clickable.
- 'Strategies' page: test if all the title are correctly loaded. Test if the article title clickable and able to redirect to page.
- 'Timeline' page: test if the calendar correctly shows all months. Test if the specific insight content show when clicking date. Test if the identified emotions show correctly by frequency. Test if the data changed when moving to another month.

Our backend server is developed with Django, which can automatically run all the unit test cases defined in files with name beginning with “test” when running the command “python manage.py test”. The unit test cases related to authorization token generation will be developed in “/MindSpaceApi/app/core/tests.py” and the unit test cases related to retrieval and saving data of emotions and insights will be developed in “/MindSpaceApi/record/tests.py”. These unit tests will test whether making a http GET request to the server endpoints “api/record/insights” and “api/record/emotions” and making a http POST request to the server endpoints “api/user/token” and “api/record/insights” will work as expected when providing with valid authorization token and payload data if required. These unit tests will also test to ensure users cannot retrieve data of emotions and insights by making http GET request to “api/record/insights” and “api/record/emotions” if no valid authorization token is provided in the request. Besides, unit tests will be conducted with Postman to check if the backend server responds with json objects in the expected format (see table in the Appendix for details)

3.3. Integration and release testing

Integration tests will be taken to test the following features:

User Authentication: Signup and Login as a new user with emails, as well as using Google account. Our application will generate two test user account with dummy data. Yingyao Lu will

test if the registered user with Google firebase can be integrated with Django backend server, and causing the Django to generate a new user and token for future authentication.
Http requests: test if frontend can connect with backend database properly. Test if frontend can post data to the database correctly. We will post in “Insights” Page with emotion tags, and check if it is record accordingly at the backend. Jason Ng is responsible for testing data http requests between frontend display and backend database for the pages of “Home”, “Explore”, “Insights” and “Timeline”, which involves retrieving data from the server endpoints “/api/record/emotions” and “api/record/insights”.
Detail of above test cases can be seen in the appendix.

3.4. User acceptance testing

Customers test a system to decide whether it is ready to be accepted from the system developers and deployed in the customer environment.
UAT is the last phase in software testing before the client accepts the application. Defeats and tech issues would be expected to fix at the earlier stage of unit tests and integration tests. UAT focus on the journey of user experiences rather than technique problems.
Our team will present tested application and run functionalities in the client meeting. A demo application will allow the client to test whether the solution works for end users and has required functionalities follows the real-world process. UAT will be undertaken by the clients as a final verification of the emulating real life usage conditions on behave of end users.

4. Testing plan schedule
   Please see appendix II.

5. Risks

Below figure identifies possible risks which will occur during testing process. Possibility and impact indicate how possible will the corresponding risk occur and how will it affect the project respectively. The mitigation plan is our solutions in respond to each possible risk.

Risk	**Possibility**	**Impact**	**Mitigation Plan**
Defects found in late stage	Low	High	An internal deadline which is earlier than the testing due date is set, therefore, all major defects should not be hidden until the late stage of testing. As a result, the project team has more time to fix the defects.
Delay due to unsolved defects	Medium	High	For defects which do not affect major functionalities of the product, defects will be fixed, or backup plan will be executed after testing of main functionalities is passed.
Scope change of testing due to client’s requirement	Low	Medium	For minor change requested, unless the new change requires integration test, testing will be executed according to the original plan to ensure major parts of testing can be passed within schedule

6.References

Appendix I

Unit Testing	Test description	Input	Expected output	Current output	Responsible (Technical side)	Approver (User)
Sign up new user with correct email address	Type correct email and password in the text area for testing	User:(new_user)@test.com Password:123456	Create a new user account with the given email address	Same as expected	Emily Lu	Client
Sign up new user with wrong email address	Type incorrect email address and password in the text area for testing	User: (xxx@test) Password:123456	Error message: the email address is not correct; Do not create new user account	Same as expected	Emily Lu	Client
Login with correct email and password	Type user email and password to login as an existed user	User: test@test.com Password: 123456	Login successfully, show correct user info on dashboard	Same as expected	Emily Lu	Client
Login with wrong password	Type user email and password to login as an existed user	User: test@test.com Password: 1111111 (incorrect password)	Error message: Password is incorrect and ask for another try	Same as expected	Emily Lu	Client
Login with email	Type user email and password to login as an existed user	User: test@test.com Password:123456	Login successfully, show correct user info on dashboard	Same as expected	Emily Lu	Client
Login with google account	Click Google login button to Direct to firebase page; and select with test google account	User:(google account)	Login with the expected account	Same as expected	Emily Lu	Client
Signup with google account	Click Google login button to Direct to firebase page; and select with test google account	Delete existing google account in the database, and then click google auth with user:(a google account)	Create a new user with the expected account	Same as expected	Emily Lu	Client

Add Insight	Click a hashtag and add an insight to the text area for testing if data are posted to the database correctly.	Click a hashtag, type a paragraph in the text area and click submit button	Emotion tile update accordingly as the hashtag changes. Backend database receive the correct insight content and hashtag, date	Same as expected	Emily Lu	Client
Get Insight	Click on a date on calendar of “Timeline” page to return the emotion records and reflection of the selected date	Click on a date on calendar	The posted date of emotion record and reflection is same as the selected date	Same as expected	Jason N	Client
Resize the circles on hompage (“Explore” page)	The size of each circle will change if its corresponding emotion frequency change	Each circle size and its corresponding emotion frequency are pre-set at default values. After that, read updated emotion frequency data.	If update of an emotion frequency applied, its corresponding circle will change its size	Same as expected	Jason N	Client
Display data of 'explore' page	The page should display data of next emotion when clicking arrow to switch emotion.	Click previous or next arrow.	Successfully load and display next and previous emotion data and trigger keywords list.	Same as expected	On Ki Ng	Client
Charts and trigger keyword lists change on 'explore' page	The charts and trigger keyword lists should display data of another week when clicking arrow to switch week.	Click previous or next arrow.	Successfully load and display charts and trigger keyword lists of next and previous week	Same as expected	On Ki Ng	Client
Charts change on 'explore' page when records reach the end.	Click arrows to next or previous week when records reach the end.	Click previous or next arrow.	Alert: there is no more records	Same as expected	On Ki Ng	Client

Article list loads on 'Strategies' page:	Click 'Strategies' page to see if all the articles title, illustration, and part of content are correctly loaded.	Click 'Strategies' page on bottom tab.	Successfully load and display all the articles title, illustration, and part of content.	Same as expected	On Ki Ng	Client
Load strategies'	Click article title to see if it is redirected to correct page	Click article title	Successfully load	Same as expected	On Ki Ng	Client

Retrieve token from backend server using token returned from firebase following successful authentication	Use Postman to make a HTTP POST request to the server endpoint "/api/user/token/"	First, use Postman to make a HTTP POST request to "https://www.googleapis.com/identitytoolkit/v3/relyingparty/verifyPassword?key={Firebase API key}"
		with body of POST request in the following format:
		{
		"email":"{testing account email}",
		"password":"{testing account password}",
		"returnSecureToken": true
		}
		Then, send a POST request to: "http://127.0.0.1:8000/api/user/token/"
		using the idToken returned from the response of the above request to construct the following body of POST request:
		{
		"token":"{retrieved idToken}",
		"uid":"{uid of the testing account}"
		}
			A http response with status code 200 with a returned JSON object:
			{
			"token":"{Django token}"
			}
				Same as expected	Eric Ma	Jason N

Saving Insight to the backend server through http POST request	Use Postman to make a HTTP POST request to the server endpoint “/api/record/insights”	Sending a POST request to:
		
		“http://127.0.0.1:8000/api/record/insights”
		with authorization header:
		“Bearer {Django token returned from the server endpoint /api/user/token/ in the above test}”
		and the body of POST request should be sent as a json object:
		{
		"reflection":"testing data",
		"tag":"sadness"
		}
			A http response with status code 200 with a returned JSON object:
			{
			"id": {id of the created database entry},
			"reflection": "testing data",
			"posted_date": "{today’s data in the format of YYYY-MM-DD}",
			"tag": "sadness"
			}
				Same as expected	Eric Ma	Jason N

Get all the record of emotion from the backend server through http GET request	Use Postman to make a HTTP GET request to the server endpoint “/api/record/emotions”	Sending a POST request to: “http://127.0.0.1:8000/api/record/emotions” with authorization header: “Bearer {Django token returned from the server endpoint /api/user/token/ in the above test}”	A http response with status code 200 with an array of JSON objects, each with the following format: { "id": {id of the database entry}, "emotion": "{emotion name}", "posted_date": "{record posted date}", "trigger_keyword_frequency": [{array of trigger keywords frequency entries}], "post": "{text of social media post associated with the record}"	Same as expected	Eric Ma	Jason N

			}
			and the trigger keywords frequency associated with a record of emotion is in the following format:
			{
			"keyword": "{keyword name}",
			"frequency": {keyword occurrence frequency in the social media post}
			}			

Get all the record of Insight from the backend server through http GET request	Use Postman to make a HTTP GET request to the server endpoint "/api/record/insights"	Sending a POST request to:
		
		"http://127.0.0.1:8000/api/record/insights"
		with authorization header:
		"Bearer {Django token returned from the server endpoint /api/user/token/ in the above test}"
			A http response with status code 200 with an array of JSON objects, each with the following format:
			{
			"id": {id of the database entry},
			"reflection": "{content of reflection}",
			"posted_date": "{saved data in the format of YYYY-MM-DD}",
			"tag": "{name of tagged emotion}"
			}
				Same as expected	Eric Ma	Jason N

Appendix II

Activity	**Responsible person**	**Efforts**	**Progress**
Tests of ‘Login & Signup’ page	Emily Lu	3 days	100%
Tests of ‘Home’ page	Jason Ng	2 days	100%
Tests of ‘Explore’ page	On Ki Ng	2 days	80%
Test of ‘Insight’ page	Emily Lu	2 days	100%
Test of ‘Strategy’ page	On Ki Ng	1 day	100%
Test of ‘Timeline’ page	Eric Ma	2 days	100%
Test of GET request to “/api/user/token/”	Eric Ma	0.5 day	100%
Test of GET and POST requests to “/api/record/insights”	Eric Ma	1 day	100%
Test of GET request to “/api/record/emotions”	Eric ma	0.5 day	100%