Sprint 2 Testing

Back-end

Test ID		1
Test Case Title		User front-end is able to get all active modules
Test Type		Functional
Objective		Verify that user front-end is able to get all active modules
Pro-condition		1. The admin front-end had created modules 2. The admin front-end should make these modules active.
Steps		1. Send the request to the server
Excepted Outcome		User front-end can get the list of all active modules
Acceptance Criteria	**Given**	Make the request
	When	I parse the request to the Eratos server
	Then	I should get the list of all active modules.
Test result		PASS
Notes		[1] Sample: https://eratosuombackend.azurewebsites.net/api/getActiveModules
tester		En Wen Tsai
Date		26 May 2021

Test ID		2
Test Case Title		User front-end is able to create a task
Test Type		Functional
Objective		Verify that the user front-end is able to create a task after finishing the payment.
Pro-condition		1. Login to get their user uri 2. Get the resource uri 3. Get the resource name 4. Get the payment ID & price from front-end 5. Get the geometry info from front-end
Steps		1. Take priority to this task 2. Send the request with these parameters
Excepted Outcome		Output success message with TaskID
Acceptance Criteria	**Given**	Received user's data
	When	I create a task based on provided data
	Then	I should get the success message.
Test result		PASS

Notes	[1] Sample: https://eratosuombackend.azurewebsites.net/api/createTask?userUri=demouser&paymentID=testpayment&price=3.1&moduleType=module3&taskType=GenerateClimateInfo&name=testname&geometry=POINT(140.3142799%20-42.84756651)&priority=low&code=DO8U6Got0JOKu9TYa4qDdE/kykdH8Yd6f7eF8WdCTcJ/WjppQTQOaA==
tester	En Wen Tsai
Date	26 May 2021

Test ID		3.1
Test Case Title		User/admin front-end is able to sync the task information
Test Type		Functional
Objective		Verify that the user front-end syncs the task information in our database with Eratos.
Pro-condition		1. The task status is incomplete (Queued or Processing)
Steps		1. Connect to back-end database 2. Check the status is “Queued” or “Processing” 3. Send the request to Eratos server.
Excepted Outcome		Output the success message “Database is up to date”
Acceptance Criteria	**Given**	Check the task status and connect to the database
	When	I send the request to Eratos server
	Then	I can get the success message.
Test result		PASS
Notes		
tester		En Wen Tsai
Date		26 May 2021

Test ID		3.2
Test Case Title		User/admin front-end is able to get the task information
Test Type		Functional
Objective		Verify that the users can get the latest task status.
Pro-condition		1. Sync the task info 2. Get the user uri
Steps		1. Find the task information by user uri
Excepted Outcome		Users can get the task information including task id, priority, state, type, resource order id and so on.
Acceptance Criteria	**Given**	Received the User uri
	When	I use the user uri as parameter to find task information
	Then	I can get the string of all tasks information.
Test result		PASS

Notes	[1] Sample: https://eratosuombackend.azurewebsites.net/api/getTasksOrdersOfUser?userUri=demouser&code=RJmC0UVgQD09JIaDnf3tozPvEh3slC/jLpIAgqgO8YFlusBsYI94xw==
tester	En Wen Tsai
Date	26 May 2021

Test ID		4.1
Test Case Title		User/admin front-end is able to sync the order information
Test Type		Functional
Objective		Verify that the users sync the order information in our database with Eratos.
Pro-condition		1. The order status is incomplete (Queued or Processing)
Steps		1. Connect to back-end database 2. Check the order status is “Queued” or “Processing” 3. Send the request to Eratos server.
Excepted Outcome		Output the success message “Database is up to date”
Acceptance Criteria	**Given**	Check the order status and connect to the database
	When	I send the request to Eratos server
	Then	I can get the success message.
Test result		PASS
Notes		
tester		En Wen Tsai
Date		26 May 2021

Test ID		4.2
Test Case Title		User/admin front-end is able to get the order history
Test Type		Functional
Objective		Verify that the users can get the latest order history.
Pro-condition		1. Sync the order info 2. Get the user uri
Steps		1. Find the order information by user uri
Excepted Outcome		Users can get the order information including order id, price, status, order time, user id and payment id.
Acceptance Criteria	**Given**	User uri
	When	I use the user uri as parameter to find order information in the database
	Then	I can get the string of all order information.
Test result		PASS

Notes	[1] Sample: https://eratosuombackend.azurewebsites.net/api/getTasksOrdersOfUser?userUri=demouser&code=RJmC0UVgQD09JIaDnf3tozPvEh3slC/jLpIAgqgO8YFIusBsYI94xw==
tester	En Wen Tsai
Date	26 May 2021

Test ID		5
Test Case Title		User/admin front-end is able to register as a new user
Test Type		Functional
Objective		Register new users for our back-end database
Pro-condition		1. Get user data from front-end
Steps		1. Insert these data to the back-end database.
Excepted Outcome		Output the success message with username.
Acceptance Criteria	**Given**	Received user data from front end
	When	I get the user info from the front-end and then insert it to our database.
	Then	I should get the successful message.
Test result		PASS
Notes		[1] Sample payload: {auth0Id: “auth0|60585179790525006901be39”, id:“https://staging.e-pn.io/users/xsrudeulfvtr2eiaiksic2jf”, info: “https://staging.e-pn.io/resources/gu4km2nnimvyqepmaw3mfcwg”, resourcePolicy:“https://staging.e-pn.io/policies/3ylpwlaj6o3wb7p4ui4kkeie”, termLastAcceptedDate:“2021-03-25T03:13:39.912183Z”,termsLastAcceptedVersion: 12}
tester		En Wen Tsai
Date		26 May 2021

Test ID		6.1
Test Case Title		User front-end is able to get the user information
Test Type		Functional
Objective		Verify that user front-end can get their user details from the database
Pro-condition		1. Get the user uri
Steps		1. Query with user uri to their user information.
Excepted Outcome		Output the success message with a string including EratosUserID, Email, Auth0ID, UserID, and name.
Acceptance Criteria	**Given**	Login to get user uri
	When	I query with valid user uri to the database
	Then	I should get the message about the user details.
Test result		PASS

Notes	[1] Sample: https://eratosuombackend.azurewebsites.net/api/getUserInfo?userUri=demouser&code=cT7R1kpoixw6jeEaCxK488gdl3BjDW5C7YgysAnj0S3Ybkay0Gx0xg==
tester	En Wen Tsai
Date	26 May 2021

Test ID		6.2
Test Case Title		Admin front-end is able to get all user information
Test Type		Functional
Objective		Admin front-end can get all user information
Pro-condition		1. Start index of user table 2. End index of user table 3. The range between start and end must be less than 100.
Steps		1. Query with start point and end point of the user ids.
Excepted Outcome		Output the success message with a string including eratosUserId, email, Auth0ID, UserID, and name. Each user is separated by ",".
Acceptance Criteria	**Given**	Put the given start and end point
	When	I send the query to the database
	Then	I should get the message about specific user details.
Test result		PASS
Notes		[1] If an admin wants to get more than a hundred user information, he/she should query more than one time. [2] Sample: https://eratosuombackend.azurewebsites.net/api/getUserInfo?userUri=all&start=1&end=10&code=cT7R1kpoixw6jeEaCxK488gdl3BjDW5C7YgysAnj0S3Ybkay0Gx0xg==
tester		En Wen Tsai
Date		26 May 2021

Test ID		7
Test Case Title		User/admin front-end is able to update their user information
Test Type		Functional
Objective		Users and admins can update user information
Pro-condition		1. Had already registered as an user/admin.
Steps		1. Make some changes in their personal information. ex: name, email, info or isAdmin.
Excepted Outcome		Output the success message "The user info is up to date.".
Acceptance Criteria	**Given**	Make some changes
	When	I update these changes to the database
	Then	I should get the success message.

Test result	PASS
Notes	[1] Only admins can change the isAdmin parameter to true or false. [2] Sample payload: {“Auth0ID”:“demoauth0”,“CreatedAt”: “5\/17\/2021 8:11:33 PM”,“Email”:“demoemail”,“EratosUserID”: “demouser”,“Info”: “ModifiedDemoInfo”, “Name”: “demoname”, “UserID”:1, “isAdmin”:false}
tester	En Wen Tsai
Date	26 May 2021

Test ID		8.1
Test Case Title		Admin front-end is able to create a resource/modules
Test Type		Functional
Objective		Verify that admins are able to create a new resource for users.
Pro-condition		
Steps		1. Create a module name 2. Create a module schema 3. Make this module is active or not
Excepted Outcome		Get the success message
Acceptance Criteria	**Given**	Put needed parameter
	When	I send these parameters to the server
	Then	I should get the success message from the server
Test result		PASS
Notes		[1] Sample: https://eratosuombackend.azurewebsites.net/api/createModifyModule?moduleSchema=https://schemas.eratos.ai/json/person&moduleName=Person&isActive=false&code=T2C73vlWSk2u5gcG2FH2URhZG4Wl15LAFULFiJEGJ2v0ETrMQMUzjA==
tester		En Wen Tsai
Date		26 May 2021

Test ID		8.2
Test Case Title		Admin front-end is able to modify a resource/modules
Test Type		Functional
Objective		Verify that admins are able to modify the existing resource.
Pro-condition		1. Admin had already created a resource
Steps		1. Change the module name 2. Change the module schema 3. Change the module status to inactive.
Excepted Outcome		Get the success message
	Given	Put changed parameters

Acceptance Criteria	**When**	I send these parameters to the server
	Then	I should get the success message from the server
Test result		PASS
Notes		[1] Sample: https://eratosuombackend.azurewebsites.net/api/createModifyModule?moduleSchema=https://schemas.eratos.ai/json/person&moduleName=Person&isActive=false&code=T2C73vlWSk2u5gcG2FH2URhZG4Wl15LAFULFiJEGJ2v0ETrMQMUzjA==
tester		En Wen Tsai
Date		26 May 2021

Test ID		9
Test Case Title		Admin front-end is able to get all available modules
Test Type		Functional
Objective		Verify that admin front-end is able to get the all available modules
Pro-condition		1. Start index of module id 2. End index of module id 3. The range between start and end must be less than 100.
Steps		1. Query with start point and end point of module id
Excepted Outcome		Output the success message with a string including Module id, Module name, Module schema, and IsActive or not. Each module is separated by “,”
Acceptance Criteria	**Given**	Put the given start and end point
	When	I send the query to the database
	Then	I should get the message about available modules’ details.
Test result		PASS
Notes		[1] If an admin wants to get more than a hundred user information, he/she should query more than one time. [1] Sample: https://eratosuombackend.azurewebsites.net/api/createModifyModule?moduleSchema=https://schemas.eratos.ai/json/person&moduleName=Person&isActive=false&code=T2C73vlWSk2u5gcG2FH2URhZG4Wl15LAFULFiJEGJ2v0ETrMQMUzjA==
tester		En Wen Tsai
Date		26 May 2021

Admin-Front

Operation	No	Test case	Precondition	Result	Note
Add a new module	2.1	Click the "Add" button	Admin has logged in Admin is at the Modules page	A new form will pop up to input module detail	

Submit a new module	2.2	Input form information and click the "submit" button	2.1	A new module is created / not created due to some reason	Whether a new module is creatable depends on if the module schema already exists in the database
List all the orders and related information	3.1	Click the Orders button at the sidebar	Admin has logged in	The table will list all the orders and some brief information	
Sort the orders according to different column names	3.2	Click a column's name	3.1	All the orders will be sorted in default/descending/ascending order, according to the content of different columns	Sorting is the extra feature
View a selected order details	3.3	Click a specific row	3.1	Order details will show on a new page	
Check the price of an order	3.4	Check the order's row with the column called "Price"	3.3	Order price will show at the specified row	
Search an order	3.5	Input the search contents and click the search icon	3.1	The filtered orders' information will be returned by rows with highlighted keywords	
Show the logged-in user profile	4.1	Click on the avatar at the top-right of the navigation bar	Admin has logged in	The admin profile will show on a new page	
Change the logged-in user profile	4.2	Revised the form and click the "Submit" button	4.1	The admin profile will be updated	
Show admin statistics (To be done)	5.1	Check home page	Admin has logged in Admin is at the home page (1.1)	The admin statistics will show on the page	The feature may require some extra resources for development, due to the requirement type (nice-to-have) and project constraint (time constraint), we were unable finishing it on time. Therefore, we will leave it as to-be-done.

User-Front

Operation	No	Test case	Precondition	Result	Note
View all active modules	1	Click each module	Admin has created active modules.	The description window will be popped up	
Draw a polygon	2		Web user has logged in	A new module is created / not created due to some reason	
Send "create a task" request to the server	3.1	Click "Payment" button and check the response	User has paid for the order	HTTP state code will be 200, and the response will contain task id.	
View result	3.2	Click "View Result" button	3.1.	The web will jump into a new page from the payment page. the result will show in a table	The result page will be displayed only payment information is successfully sent to the server.
View history orders	3.3	Click "history orders" via "User" component and via the result page.	Web user has logged in	History orders will be displayed in the table on a new page	There are two places that can enter the history order page
User profile	4	Check	User has paid for the order		