Greenwatch Test Case Document Draft

G. Mathers, D. Pollock, D. Portillo, K. Pulliam, E. Mondragon

4/24/2023

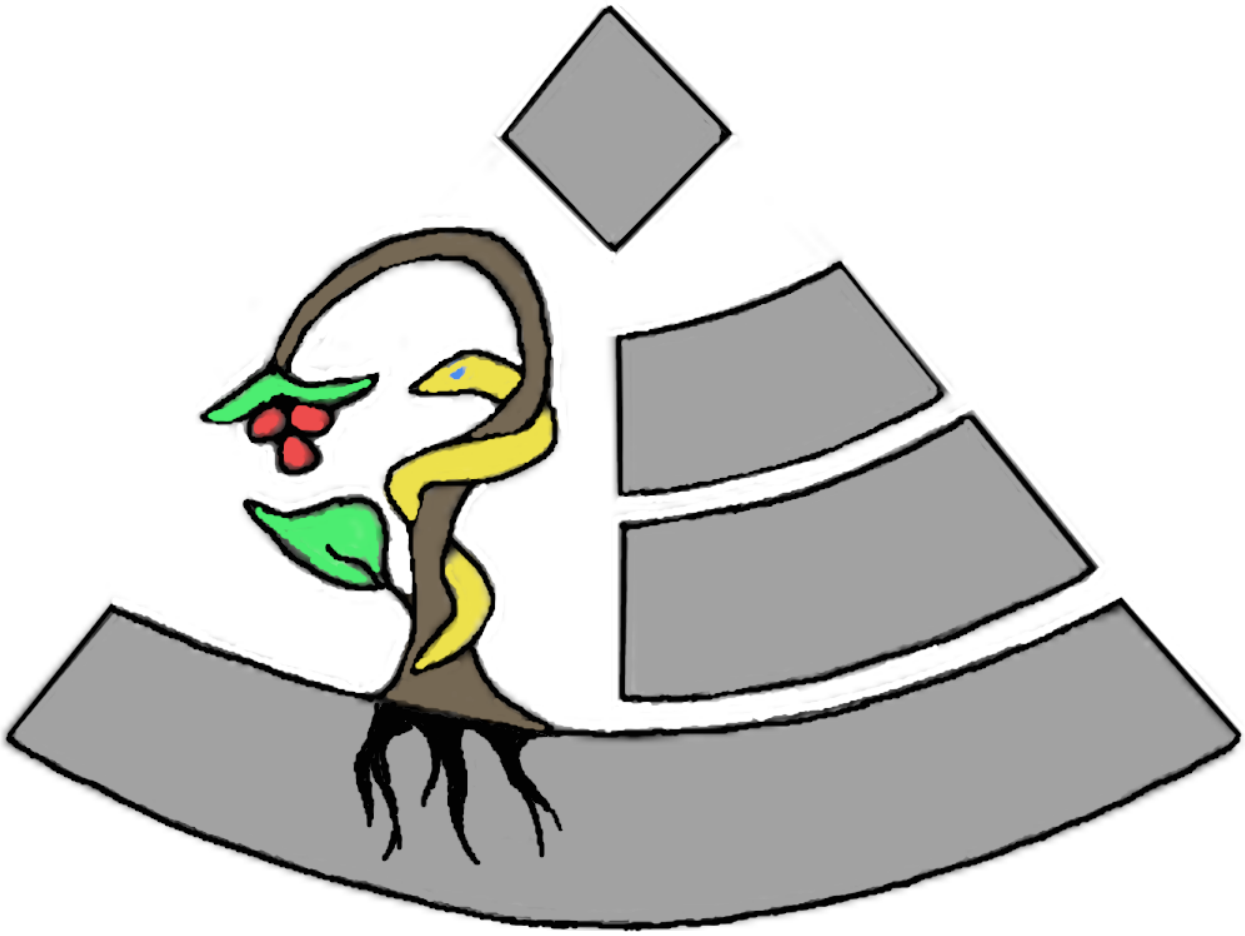


Table of contents

1. Introduction

1.1) Purpose and Overview

This document's purpose is to give an overview and a record of all testing procedures performed for the Greenwatch system as well as additional information related that might impact testing. Testing is split up into two sections: Frontend and Backend testing. Frontend testing revolves around the general UI portions of the system whereas the Backend testing revolves around software related to the hardware and sensors such as the agents and the all around server for greenwatch.

1.2) Key Objective

The objective of these tests is to find any dysfunctional aspects of the system that must be documented in an attempt to rectify them and ensure functional code.

2. Testing Procedures

2.1) Frontend

The frontend tests will consist of black-box style testing to ensure that the behavior of the application is functioning as intended. The tests will be carried out from a user's perspective. There will be varying tests for admin roles and basic roles where applicable, such as allowing the ability to manipulate rooms or other users.

2.2) API (Javascript Fetch Requests)

Testing the javascript api calls will involve using white-box testing techniques to ensure that the data is being returned properly. A single file will make every call possible to the api, and will either return a success or fail status. Each status will then be recorded so that it is easily identifiable which requests have failed and which requests passed.

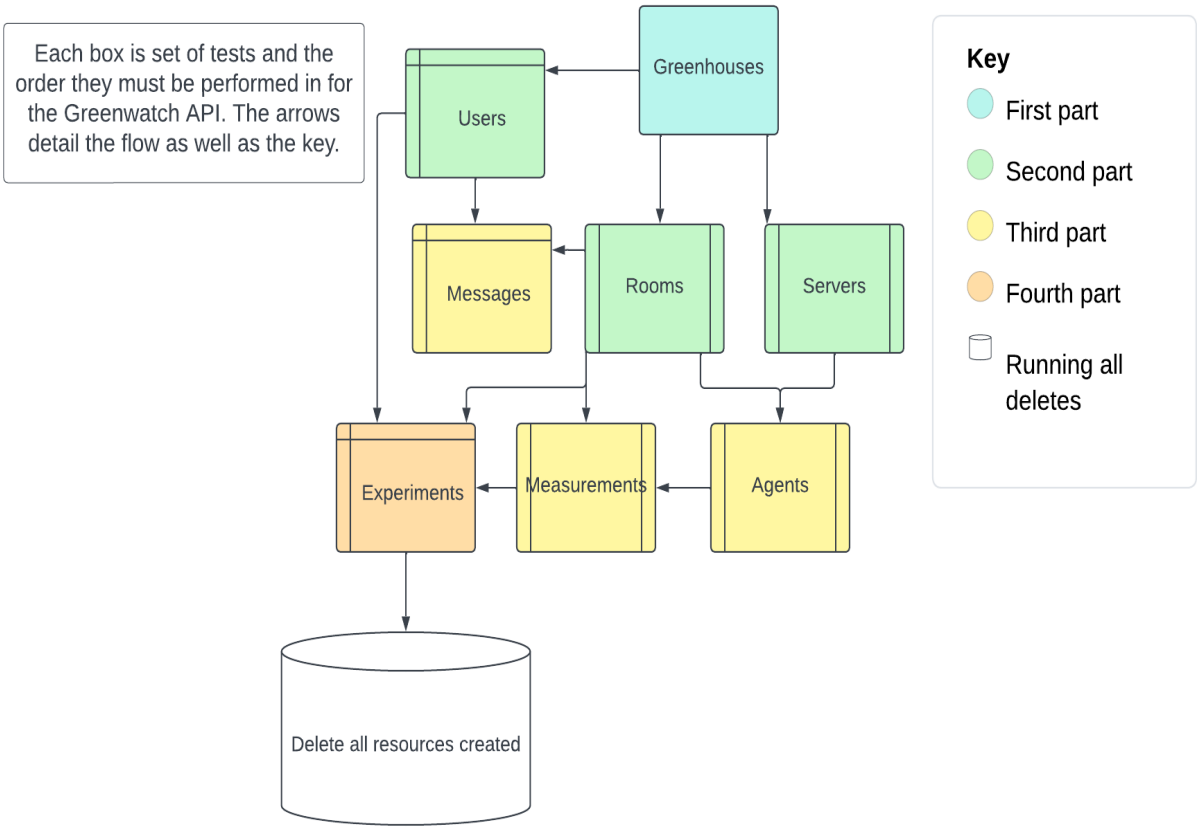
2.3) Backend

The REST API built into Greenwatch has many functionalities. Some routes must be tested before others because of some required foreign keys and relationships in the lower level objects such as Rooms, Servers, etc. A Greenhouse object would sit on top of the test scenario. From the top of the test scenario we would create the other lower level objects tests such as Users and logging them in, then Rooms and Servers with the Greenhouse ID.Then, begin testing Agents, Experiments, Users, Messages, Measurements, all methods will be tracked with the URI method tracker with test_routes in the RouteTester class. The results of the various methods can be dumped to the

screen, a JSON file, and also CSV. Whenever the server is started, the function will run the tests, dump the results, and clear the scene. What does it mean to “clear the scene”? After we get through running all the tests with test_routes the created resources will be deleted and cleared from the database. Finalizing the tests. All resources created will be named using testing aliases so when the logs appear on the server they can be traced to the test at the beginning of start up. The user JWT being passed around is that of a test user as well.

RouteTester, is a custom built testing class for CRUD APIs. Its test_routes method can be given a specifically formatted JSON file detailed in Backend/ Server/ testing/route.json. The formatted JSON file details the URI, methods it implements, and the data if any needed for the operation on every single available route for the API. The specific format will be detailed throughout the testing folder. Test_routes runs the entire test process and the individual routes tests.

Test Process - REST API



The above image details the order in which the tests will be placed. Tests will first create, then read to ensure the creation happened. We will then run updates on the created objects and read again to make sure they were changed. After this process repeats for every single object, we will have been tracking the ID's of all created objects and queuing the delete methods for the very end when looping through the formatted JSON file. With this information, the method will then run a process to delete all created resources in the tests. This will end the testing phase, and move us to reporting on the test data generated. Which will be available via a dump to the terminal and a JSON file detailing the results with much greater detail.

Test Process - Agent

The Agent will have a much smaller testing plan than the Server. Tests will consist of ensuring the Measurements are being taken accurately and confirming the Agent is sending Measurements to the GreenhouseServer. We also have to be able to copy the original Agent file and be able to build an executable that is specific to the Room the Agent was created for. (Very rough problem to solve, but solved).

2.4) Hardware

When it comes to testing the hardware, the process is very simple. First step of the many is to ensure that the layout of the connections to the desired components are all correct. Once the layout is revised and is fully satisfied, it is time to start programming. In the Raspbian OS click on the start symbol on the top left, and a drop down will reveal itself. From there an option for Programming will show itself, clicking on that will show a choice of programming provided by the operating system. The main one that will be used is Thonny IDE. Now depending on the type of component that is being used will ultimately decide how the programming will go. Now prepare the program through intensive research not only from the internet, but through the files of the Raspbian to see what all is included. Before running the program make sure that nothing is disabled from the raspbian interface that is being used by the component. Now run the program that was created and depending on your programming skills, the hardware should work.

3. Testing Schedule

The majority of testing will be conducted at the end of the development cycle of the final Agile sprint. Testing of the frontend will be conducted first while the final sections of the backend will be finished and then tested. Due to there being separate Frontend and Backend teams, testing may coincide or overlap as they can be accomplished independently. System tests will be accomplished after both testing sections are conducted.

4. Limitations and Notes

Due to tight time constraints, features such as the alerts system have been dropped in order to focus on the code being accomplished in a timely manner. There were also additional costs that caused the alerting section to get dropped as well. RouteTester.py can be used to test the API at any point in development to ensure all moving parts on the backend API are coordinating well and working in conjunction with each other.

5. Test Cases

5.1) Frontend

Page	Tested Function	Test Action	Expected Result	Actual Result	Correct Behavior?
N/A (Browser Access)	Accessing Webpage	Search for greenwatch website	Greenwatch Login Page Displays	Greenwatch Login Page Displays	true
Login Page	Proper display of UI elements	Load Login Page	Full UI displays correctly	Full UI displays correctly	true
Login Page	Username & Password	Enter valid credentials	Login in is successful, and directs to homepage	Login in is successful, and directs to homepage	true
Login Page	Username & Password	Enter invalid credentials	Displays an error notification, and denies access	Displays an error notification, and denies access	true
Home Page (Admin)	Proper display of UI Elements	Load Home Page	Full UI displays correctly	Modals don't adjust size properly for varying screen sizes	false
Home Page (Admin)	Create User	Submitting new user information through new user form	Creates a new user, and displays the results in the user list	Creates a new user, and displays the results in the user list	true
Home Page (Admin)	Edit User	Changing attributes of user through edit user form	Edits the user, and displays the results in the user list	Edits the user, and displays the results in the user list	true

Home Page (Admin)	Delete User	Selecting the delete button to confirm deletion of user	Deletes the user, and displays the results in the user list	Deletes the user, and displays the results in the user list	true
Home Page (Admin)	Create Room	Submitting new room information through add room form	Creates a new room, and displays the results in the room list and the home screen	Creates a new room, and displays the results in the room list and the home screen	true
Home Page (Admin)	Edit Room	Changing room information through edit room form	Edits the room, and displays the results in the room list	Edits the room, and displays the results in the room list	true
Home Page (Admin)	Delete Room	Selecting the delete button to confirm deletion of room	Deletes the room, and displays the results in the room list	Deletes the room, and displays the results in the room list	true
Home Page (Basic)	Proper display of UI Elements	Load Home Page	Full UI displays correctly, while hiding admin privileges	Full UI displays correctly, while hiding admin privileges	true
Home Page (Admin/Basic)	Room Card	Click on a room card	Takes you to the corresponding room page	Takes you to the corresponding room page	true
Home Page (Admin/Basic)	Notes Box	Selecting a room in the dropdown menu	Rooms messages are displayed in the notes box	Rooms messages are displayed in the notes box	true
Home Page (Admin/Basic)	Logout	Selecting the logout button in the navigation bar	Logs out the user, and sends them back to the login page	Logs out the user, and sends them back to the login page	true
Room Page (Admin)	Proper display of UI Elements	Load Room Page	Full UI displays correctly	Experiment modals and Chart don’t adjust size properly for	false

				varying screen sizes	
Room Page (Admin)	Create Experiment	Click add experiment button, fill in experiment information, and hit create experiment button	Experiment is added to the database, and is rendered in the experiments list	Experiment is added to the database, and is rendered in the experiments list	true
Room Page (Admin)	Edit Experiment	Click edit experiment button, fill in changes for experiment, and hit save button	Experiment information is updated, and changes are reflected immediately in the experiment list	Experiment information is updated, and changes are reflected immediately in the experiment list	true
Room Page (Admin)	Delete Experiment	Click delete experiment button to confirm the deletion of the experiment	Experiment is removed from database, and experiment is also removed from the experiments list	Experiment is removed from database, and experiment is also removed from the experiments list	true
Room Page (Admin)	Create Agent	Download a new agent executable file by clicking the “Download Agent” button from the agent dropdown	An executable file is downloaded to the users system	Function has yet to be implemented	false
Room Page (Admin)	Edit Agent	Edit agents refresh status	Agent refresh timer is updated	Function has yet to be implemented	false
Room Page (Admin)	Delete Agent	Delete an agent from a room by clicking “delete agent” button	Agent is removed from the room, and room can no longer receive new measurement data	Function has yet to be implemented	false
Room Page (Basic)	Proper display of UI Elements	Load Room Page	Full UI displays correctly, while hiding admin privileges	Experiment modals and Chart don’t adjust size properly for varying screen sizes. Admin privileges are hidden	false

Room Page (Admin/Basic)	Chart	Settings the date range for the chart, and updating it by selecting the update button	Chart displays measurements that fall within the date range given	Chart displays measurements that fall within the date range given	true
Room Page (Admin/Basic)	Chart	Selecting a chart type from the dropdown menu	Chart displays measurements that are only of that type	Chart displays measurements that are only of that type	true
Room Page (Admin/Basic)	Chart	Exporting measurement data by selecting the “export data” button	A file is downloaded to the users system in the CSV format	A file is downloaded to the users system in the CSV format	true
Room Page (Admin/Basic)	Logout	Selecting the logout button in the navigation bar	Logs out the user, and sends them back to the login page	Logs out the user, and sends them back to the login page	true

5.2) API

Request Function	API Call	Action	Expected Result	Actual Result	Correct Behavior ?
Register User	POST /register	Given a user object, create a new user	Adds the user to the database	Adds the user to the database	true
Login	POST /login	Given a user object, validate login information	Adds the user to the database	Adds the user to the database	true
Get Users	GET /users	Get a list of all users	Returns an array of all user objects	Returns an array of all user objects	true
Edit User	PATCH /users/<user_id>	Given a user object and user id, update the information of the user	Updates the information of the user based on the provided user object	Updates the information the user based on provided user object	true
Delete User	DELETE /users/<user_id>	Given a user id, delete the user	Removes the user object from the database	Removes the user object from the database	true
Get Rooms	GET /rooms	Gets a list of all rooms	Returns an array of room objects	Returns an array of room objects	true
Create Room	POST /rooms	Given a room object, create a new room	Creates a new room object within the database	Creates a new room object within the database	true
Get Room By ID	GET /rooms/<room_id>	Given a room id, get the room with the matching id	Returns a room object based on the room id specified	Returns a room object based on the room id specified	true
Edit Room	PATCH /rooms/<room_id>	Given a room object and room id, update the	Updates the information of the	Updates the information of	true

		information of the room	room based on the provided room object	the room based on the provided room object	
Delete Room	DELETE /rooms/<room_id>	Given a room id, delete the room	Removes the room object from the database	Removes the room object from the database	true
Get Measurement By Room ID	PUT /rooms/<room_id> /measurement	Given a date object, and a room id, get all measurements associated with the room between the given dates	Returns an array of measurement objects within the specified dates	Returns an array of measurement objects within the specified dates	true
Create Measurement	POST /rooms/<room_id> /measurement	Given a measurement object, room id, and agent key, create a new measurement	Adds a new measurement object to the database that is associated with the given room id	Adds a new measurement object to the database that is associated with the given room id	true
Get All Messages	GET /rooms/messages	Get a list all messages within every room of the database	Return an array of message objects	Return an array of message objects	true
Get All Messages By Room	GET /rooms/<room_id> /messages	Given a room id, get all messages associated with it	Return an array of message objects associated with the given room	Return an array of message objects associated with the given room	true
Create Room Message	POST /rooms/<room_id> /messages	Given a room id, user id, and a message object, create a new message	Creates a new message object within the database that is associated with the given room and user	Creates a new message object within the database that is associated with the given room and user	true
Get Experiments	GET /experiments	Get a list of all experiments in the database	Returns an array of experiment objects	Returns an array of experiment objects	true
Create Experiment	POST /experiments	Given an experiment object, create a new experiment	Creates a new experiment object within the database that is associated with the room that it was created in	Creates a new experiment object within the database that is associated with the room that it was created in	true

Get Experiment By ID	GET /experiments /<experiment_id>	Given an experiment id, get the experiment object associated with it	Returns an experiments object based on the given experiment id	Returns an experiments object based on the given experiment id	true
Edit Experiment	PATCH /experiments /<experiment_id>	Given an experiment object and experiment id, update the experiment	Updates the experiment object specified by the experiment id with the experiment object that was given	Updates the experiment object specified by the experiment id with the experiment object that was given	true
Delete Experiment	DELETE /experiments /<experiment_id>	Given an experiment id, delete the experiment	Removes the experiment object from the database that was specified with the experiment id	Removes the experiment object from the database that was specified with the experiment id	true

5.3) Agents

Test Name	Test Description	Test Action	Expected Result	Actual Result	Success

Create Agent	API makes a copy of the original Agent file and parses out variables to be the appropriate values for the room it will gather measurements for	POST /servers/agents
		
		{
		duration: “string”
		room_id: int,
		server_id: int,
		ip_address: string
		}
			Newly created file named “agent_{room_id} .py”
			
			Agent ID
				A new python script was created with the details the agent needs in order to take and store measurements for the specific room.	true

Download Agent	API will create an executable file to run on the raspberry pi based off of the python script created when we built the Agent specific to the Room	GET /servers/agents/{ agent_id}	Executable file capable of running on Linux based operating systems.	Not creating executable properly. Problem will be solved in future but not prioritized at moment	false
Create Measurement	Agent script creates a measurement to send to server by accessing various sensors	Pulling data from hardware sensors	JSON object able to be passed to server containing appropriately labeled keys	Grabs accurate environmental information from sensors on raspberry pi and packages into appropriately keyed JSON objects.	true

Send Measurement	Agent script sends a HTTP request to the server that is acting as its intermediary proxy. The proxy will then send to ip of Greenwatch instance	POST /rooms/{room_id} /measurement	201 response from server with JSON object containing the duration to sleep for	201 response from server with JSON object containing the duration to sleep for	true

6. Conclusion

The testing phase was a success. We were able to confirm functionality specified in our requirements document is up and functioning as intended. There are a few mistakes in our implementation and overall design that need to be hammered out such as the ability to download an executable file for the specific agent. We had it functioning in development but when swapped to a containerized version of the application we ran into errors that would not allow the connection that was established to be maintained. When the subprocess would begin, the tcp connection would terminate. I was unable to find an effective workaround that was able to be used in production with the time that was remaining. Luckily, we ran into very few of those issues and were able to actually get a production level instance running with its very own public IP address.