Team Number: Section 101 - Team 5
Team Name: International Justice League of Super Acquaintances (IJLSA)
Team Members: Taylor Ellis, Sam Fitzsimmons, Alyvia Hildebrand, Jason Hong, Steven McDonald, Robert Tennant
Project Title: ECMap

Testing Methods and User Acceptance Cases:
Features

1. **Search Function:** User has a room (or partial information about a room) in mind that they are intending to find. The user should be able to enter the room number, the wing ID, the combination of both, and/or partial information. From the user input, we expect our SQL query to return the necessary information for our Javascript app to run successfully.
    a. **We want to test for:** a user entering complete (or partial) information (entry variables are room number or wing ID) about a room, and our node functions returning the right information (image name for map, 'x' coordinate and 'y' coordinate for the room's pixel location on the map).
    b. **We want to test against:** Users entering non-searchable information, and our system handling their input effectively. This could include invalid characters, SQL queries, or injected code.
    c. **Step-by-Step for Test**
        i. The user visits our website, logs in, and puts their cursor into the search field.
        ii. The user enters a certain room number, wing ID, combination of both, or otherwise incomplete data.
            1. If the search data matches (or partially matches) one or more entries in our database of rooms, all matching rooms will be returned to our Javascript function in an array. This array will store exact room numbers, wing IDs, and the room type, as well as the necessary variables for our Javascript function to return a map and dropped pin (map image name, x-coordinate, and y-coordinate).
            2. If the search data does not match any entry in our database but is still valid input (ie. a 4 digit room number, an invalid wing ID, etc), we will not run our Javascript function, and no maps will be returned.
            3. If the search data is of an invalid form (ie. non ASCII characters, direct SQL queries, etc.) we want to prevent our Javascript application from returning non-expected behaviors.
        iii. Our node query returns a valid array to be used with the rest of the application.
    d. **Acceptance:** We can mark this test as "accepted" when our server.js script returns valid search results for any valid input (as determined by our database),

and it returns either an error or redirects to the base search page when invalid input is entered.

e. **Testers:** Our entire team will be testing for valid and invalid input as we work out any bugs in the Javascript app.

f. **Test Script:** We intend to write multiple scripts that run unit tests on our search function. Sam will be responsible for implementing these tests and examining these results before we make our website live.

2. **Interactive Map:** After entering a valid search and returning valid information as outlined in the step above, we expect our Javascript application to load and display the requested map. If the search returns more than one possible room (ie. someone enters just a wing ID, or enters just a room number that is repeated across multiple wings), then our app should display a drop-down menu with the options before the map is pulled up. This will allow users to choose the correct room before the map is loaded, avoiding any ambiguity in what gets returned to the user.

    a. **We want to test for:** The return variables from a valid search (as outlined above) will return the following, depending on the content of the returned array:

        i. The map image and dropped pin of the room, assuming the returned array is one dimensional. This implies that the search query was sufficient to narrow down the user's request to a single room. The user has no ambiguity for which room they are looking for, and our application should load the image directly.

        ii. A drop down of potential rooms, assuming the returned array is two-dimensional. This implies that the search query was insufficient to narrow down the user's request to a single room. Because the user would likely be unclear what room they want, we would render a drop-down that lists the rooms they might be searching for. When the user selects a complete room, the Javascript function will return the map and dropped pin of that room.

    b. **We want to test against:** The Javascript function only operates on valid searchable material (tied closely to the tests in Step 1). Because the validity of input is determined in the search function, there is less pressure to filter out the input from the user (not a direct interface for the user). We also want to make sure that the application's return information is accurate and scalable with our zooming/panning functionalities.

    c. Step-by-Step for Test

        i. The user enters a search, and its validity is determined prior to starting the Javascript application. An array of information is returned and passed into our Javascript as the function's parameters.

        ii. We run a simple function to determine the dimensionality of the array.

            1. If the array is non-dimensional (ie. no valid input is returned), the map will not load and the user will be redirected to the search page again.

2. If the array is one-dimensional, we can load the respective map and drop the pin.
3. If the array is two-dimensional, we will generate a drop down with the wing ID and room number. The user can select one of the valid rooms, and our application will then load the map/pin in the user's view.

d. **Acceptance:** We can mark this test as "accepted" when we have confirmed that the application can take a non-dimensional array, a one-dimensional array, or a two-dimensional array and return with the appropriate behavior as outlined above.
e. **Testers:** Robbie, Taylor, and Jason will be the primary testers within our group that will be confirming the expected functionalities. This will be a combination of manual testing (making sure the dropped pins are accurate) and automated testing via scripts.
f. **Test Script:** The same group members as above will be writing unit tests to automate and ensure that the application behaves correctly.

3. **Login Function:** Users should be logged in to use the map. We expect that first time users will be prompted to register using their email, and returning users should be prompted with a standard login landing page.
    a. **We want to test for:** The two anticipated behaviors for our login page can be completed for all users.
        i. The sign-up page should have entry fields for all relevant information for a user account (ie. email address, first and last name, passwords, etc.). Upon entering all the required fields, a new user should be added to our user database. Their information should be stored securely, with a priority on encrypting their passwords.
        ii. The login page should have entry fields for retrieving a user's account from our user database. By entering a user's email and password, our database should communicate securely with the website to load the rest of the user's information.
    b. **We want to test against:** Any potential exploits of our database system that might lead to a compromise of user data. We will need to test that our method(s) of encrypting any confidential info is safe and secure. We may also need to implement certain precautionary measures for user creation to avoid conflicts, such as duplicate emails or passwords with invalid characters.
    c. **Step-by-Step for Test**
        i. The user arrives at the website's homepage and is presented with a login popup.
            1. If this is a new user, they will need to create their account before proceeding. A dialogue box with the required fields (email address, first name, last name, and password) will pop up. When the user enters all their information and submits, a new user in the

user database will be added based on the info submitted. All confidential info will be encrypted through a postgres-compatible hashing method that can be securely retrieved later on.

2. If this is a returning user, they should proceed to the standard login screen. This takes their email and password and securely retrieves the associated user account, assuming the credentials match.

ii. After the user logs in, they should be free to roam the rest of the site and use the ECMap at their leisure.

d. **Acceptance:** We can consider this test complete when all incoming traffic to the site can properly create and access their account.

e. **Testers:** Steve will be implementing and testing the setup and security measures for the backend database, while Jason will be testing the front end views and functionality for the log-in and sign up forms.

f. **Test Script:** The same group members as above will be writing unit tests to automate and ensure that the log-in and sign up pages behave appropriately.