1. Users will be prompted to register an account if they do not already have an account when they log in. If the user attempts to log in, and their credentials are not found in our accounts table, they are redirected to the registration page.
    a. Test Cases:
        i. Test 1
            1. Description: User attempts to log in with a valid account
            2. Input Data: Valid credentials (username and password)
            3. Expected Outcome: User successfully logs in.
        ii. Test 2
            1. Description: User attempts to log in without an account
            2. Input Data: Invalid credentials (username not in database)
            3. Expected Outcome: User is redirected to registration page.
        iii. Test 3
            1. Description: User registers an account after redirection
            2. Input Data: New registration data (username, email, password)
            3. Expected Outcome: User account created; user logs in.
        iv. Test 4
            1. Description: User attempts to register with an existing username
            2. Input Data: Username that is already in use
            3. Expected Outcome: Error message: “Email already registered.”
    b. Test Environment:
        i. Platform: Testing will be done on localhost.
        ii. Database: A mock database with a predefined set of accounts will be set up to test cases involving both existing and non-existing users.
    c. Test Results
        i. Pass Criteria: Each test case should complete with the correct expected outcome. For example, if a user logs in with an account that doesn’t exist, they should be redirected to the registration page.
        ii. Fail Criteria: Any deviations from the expected outcomes will be marked as failures and recorded with error logs and screenshots to be reported to the development team.
    d. Acceptance Testing Team
        i. Testers: Internal testers and volunteer beta testers familiar with using registering/logging into accounts.
        ii. Roles and Responsibilities:
            1. Internal Team: Performs initial tests to identify major issues.
            2. Beta Testers: Validate form usability and effectiveness of error messages and redirects, providing feedback on any points of confusion or difficulties encountered

2. The feature allows users to filter climbing routes based on the route's location. This test ensures that the filtering functionality works correctly, providing users with relevant results when searching for climbing routes in specified locations.
    a. Test Cases:
        i. Test 1
            1. Description: User selects a specific location to filter routes
            2. Input Data: Boulder, CO
            3. Expected Outcome: Only routes located in Boulder, CO are displayed.
        ii. Test 2
            1. Description: User selects a location with no available routes
            2. Input Data: Unknown Location
            3. Expected Outcome: Message: "No routes found for this location."
        iii. Test 3
            1. Description: User clears location filter
            2. Input Data: None (clear filter)
            3. Expected Outcome: All climbing routes are displayed.
        iv. Test 4
            1. Description: User selects multiple locations in a single session
            2. Input Data: Boulder, CO and Denver, CO
            3. Expected Outcome: Only routes located in Boulder, CO and Denver, CO are displayed.
        v. Test 5
            1. Description: User enters an invalid location
            2. Input Data: 12345
            3. Expected Outcome: Message: "Please enter a valid location."
    b. Test Environment:
        i. Platform: Testing will be done on localhost.
        ii. Database: A database with a predefined set of climbing routes assigned to specific locations, including test data for locations with no associated routes, will be used to validate the filtering functionality.
    c. Test Results
        i. Pass Criteria: Each test case should yield the expected filtered results based on the chosen location, error messages for invalid inputs, and no unexpected results for non-existent locations.
        ii. Fail Criteria: Any result that does not align with the expected outcomes will be documented, with relevant error logs, screenshots, and steps to reproduce, to be shared with the development team.
    d. Acceptance Testing Team

i. Testers: Internal testers and volunteer beta testers familiar with using filter search.
ii. Roles and Responsibilities:
1. Internal Team: Performs initial tests to identify major issues.
2. Beta Testers: Validate filter search usability and effectiveness of error messages, providing feedback on any points of confusion or difficulties encountered

3. This feature allows users to add new climbing routes by filling out a form with relevant information (e.g., route name, location, difficulty, description). This test ensures that the form submission functionality works correctly, validates inputs, and saves new climbing routes to the routes page.
a. Test Cases:
i. Test 1
1. Description: User successfully adds a new climbing route.
2. Input Data: Valid data for all fields (Route Name: Sunset Crag, Location: Golden, CO, Difficulty: 5.10a, Description: A beautiful route with a scenic view at the top.)
3. Expected Outcome: Route is added to the routes page with the correct information.
ii. Test 2
1. Description: User submits form with missing required fields.
2. Input Data: Missing required field (e.g., Route Name left blank).
3. Expected Outcome: Error message displayed: "Please complete all required fields."
iii. Test 3
1. Description: User adds route with an invalid difficulty level.
2. Input Data: Difficulty level outside expected range (e.g., -1 or 11.5).
3. Expected Outcome: Error message displayed: "Please enter a valid difficulty level."
iv. Test 4
1. Description: User attempts to add a climbing route that already exists.
2. Input Data: Route Name: Sunset Crag, Location: Golden, CO, Difficulty: 5.10a, Description: A beautiful route with a scenic view at the top. (matching an existing route in the database).
3. Expected Outcome: Error message displayed: "This route already exists."
b. Test Environment:
i. Platform: Testing will be conducted on localhost.

ii. Database: A database will store climbing routes for testing. The database will include predefined routes for tests involving duplicate locations and to ensure data persists correctly.

c. Test Results
    i. Pass Criteria: Each test case should lead to the correct response for valid and invalid inputs. Routes added with valid data should display accurately on the routes page, while errors should prompt users to correct issues with invalid data entries.
    ii. Fail Criteria: Any unexpected result, such as improper data saving or missing error messages, will be documented with error logs, screenshots, and reproduction steps to be communicated to the development team.
d. Acceptance Testing Team
    i. Testers: Internal testers and volunteer beta testers familiar with adding data through online forms.
    ii. Roles and Responsibilities:
        1. Internal Team: Performs initial tests to identify major issues.
        2. Beta Testers: Validate form usability and effectiveness of error messages, providing feedback on any points of confusion or difficulties encountered