001. User can sign up on GitHub

Precondition:

- User is on the GitHub sign-up page: Link

Test Steps:

1. Navigate to the sign-up page.
2. Fill in the required fields (e.g., username, email, password, Your Country/Region).
3. Click on the [Continue] button.
4. Verify that the system navigates to the next step (email verification or additional setup).
5. Verify that the system displays an email verification prompt (if applicable).
6. Verify that the user receives an email with a verification link (check the inbox).
7. Click on the verification link and verify the account is activated.

Expected Result:

- The user should be able to sign up successfully.
- The system should prompt for email verification (if required).
- After verification, the user should be able to access their new GitHub account.

Test Script

```
describe('GitHub Sign-Up Process', () => {
 before(async () => {
await browser.url('https://github.com/signup');
 });

 it('should complete sign-up step and prompt for email verification', async () => {
const emailInput = await $('#email');
await emailInput.setValue('testemail' + Date.now() + '@example.com');

const continueButton = await $('button[type="submit"]');
await continueButton.click();
```

```
    await browser.pause(1000);

    const passwordInput = await $('#password');
    await passwordInput.setValue('TestPassword123!');
    await continueButton.click();

    await browser.pause(1000);

    const usernameInput = await $('#login');
    await usernameInput.setValue('TestUser' + Date.now()); // унікальне ім’я
    await continueButton.click();

    await browser.pause(1000);

    const countrySelect = await $('#country');
    if (await countrySelect.isExisting()) {
    await countrySelect.selectByVisibleText('Ukraine');
    await continueButton.click();
    }

    await browser.pause(1000);

    await browser.waitUntil(async () => {
    const currentUrl = await browser.getUrl();
    return currentUrl.includes('verify') || currentUrl.includes('plan') || currentUrl.includes('setup');
    },
    { timeout: 10000, timeoutMsg: 'Did not proceed to the next sign-up step' });

     const verificationTextExists = await $('body').getText();
     expect(
     verificationTextExists.toLowerCase()
    ).toContain('verify') || console.warn('Verification message not explicitly shown.');

    console.log('User proceeded to the next step in sign-up process successfully!');
  });
});
```

002. User can sign in to GitHub

Precondition:

- The user has a GitHub account and valid credentials (username and password).

Test Steps:

1. Go to the GitHub login page: [Link]
2. Enter the correct username in the "Username or email address" field.
3. Enter the correct password in the "Password" field.
4. Click the [Sign in] button to log in.
5. Wait for the page to load after login.
6. Verify that the user is redirected to the GitHub homepage.
7. Verify that the username is displayed in the top right corner of the page, confirming a successful login.

Expected Result:

- The user successfully logs into their GitHub account, and their username is displayed in the top right corner of the page.

Test Script

```
describe('GitHub Login Process', () => {

before(async () => {
await browser.url('https://github.com/login');
});

it('should log in successfully', async () => {
   const usernameInput = await $('#login_field');
   const passwordInput = await $('#password');

   await usernameInput.setValue('TestUsername');
   await passwordInput.setValue('TestPassword123');

   const loginButton = await $('input[type="submit"]');
   await loginButton.click();

   await browser.waitUntil(async () => {
   const url = await browser.getUrl();
   return url.includes('github.com');
   }, { timeout: 5000, timeoutMsg: 'Login failed or user not redirected to home page' });

   console.log('Login successful and user is redirected to the home page!');
});
});
```

003. The user can sign out of GitHub

Precondition:

User is logged into a GitHub account.

Test Steps:

1. Log in with valid credentials.
2. Click on the profile avatar in the top right corner.
3. Click on [Sign out] from the dropdown menu.
4. Verify that the user is redirected to the login page or logged out.

Expected Result:
User is successfully logged out and redirected to the login or homepage without session data.

Test Script

```
describe('GitHub Logout Process', () => {
it('should log out successfully', async () => {
await browser.url('https://github.com/login');
await $('#login_field').setValue('TestUsername');
await $('#password').setValue('TestPassword123');
await $('input[type="submit"]').click();

const avatar = await $('summary[aria-label="View profile and more"]');
await avatar.click();

const signOutButton = await $('form.logout-form button[type="submit"]');
await signOutButton.click();

await browser.waitUntil(async () => {
const currentUrl = await browser.getUrl();
return currentUrl.includes('/login');
}, { timeout: 5000, timeoutMsg: 'User was not redirected after logout' });
```

```
        console.log('User logged out successfully.');
    });
});
```

004. The user can enter a search query from the home page and get results

Precondition:

User is on the GitHub homepage: Link

Test Steps:

1. Go to the GitHub homepage.
2. Locate the search input field.
3. Enter a valid search term (e.g., qa automation).
4. Submit the search query.
5. Wait for the results page to load.
6. Verify that at least one type of result is displayed (repositories, users, code, etc.).

Expected Result:

The system shows relevant search results that match the query, regardless of the type (repository, user, issue, code, etc.).

Test Script

```
describe('GitHub Search Functionality', () => {

it('should display relevant results for search queries', async () => {
await browser.url('https://github.com');

const searchInput = await $('input[name="q"]');
const searchTerm = 'qa automation';

await searchInput.setValue(searchTerm);
await browser.keys('Enter');
await browser.waitUntil(async () => {

const currentUrl = await browser.getUrl();
```

```
 return currentUrl.includes('search?q=');
  },
  { timeout: 10000, timeoutMsg: 'Search page did not load properly' });

  const resultsHeader = await $('h3');
  const resultsText = await resultsHeader.getText();
  expect(resultsText).toContain('results');

  const repositoriesTab = await $('a[href*="type=repositories"]');
  const usersTab = await $('a[href*="type=users"]');
  const codeTab = await $('a[href*="type=code"]');

  expect(await repositoriesTab.isDisplayed()).toBe(true);
  expect(await usersTab.isDisplayed()).toBe(true);
  expect(await codeTab.isDisplayed()).toBe(true);
  console.log('Search returned relevant results successfully!');
    });
 });
```

005. User can create a new repository

Precondition:

- User is logged into their GitHub account.

Test Steps:

1. Click on the [Create repository] button to create a new repository.
2. Enter a unique repository name.
3. Select repository visibility (e.g., Public).
4. Click the [Create repository] button.
5. Verify that the repository is created and the user is redirected to the repository's main page.
6. Verify that the repository name is displayed correctly on the page.

Expected Result:

- The repository is created successfully.
- The user is redirected to the new repository’s main page.
- The repository name is displayed as entered.

Test Script

```
describe('GitHub Create Repository', () => {
    before(async () => {
await browser.url('https://github.com');
 });
```

```
     it('should create a new repository and redirect to its main page', async () => {
    const newRepoButton = await $('a[href="/new"]');
    await newRepoButton.click();

    const repoNameInput = await $('#repository_name');
    const repoName = 'TestRepo' + Date.now();
    await repoNameInput.setValue(repoName);

    const publicRadio = await $('#repository_visibility_public');
    await publicRadio.click();

    const createButton = await $('button[type="submit"]');
    await createButton.click();

    await browser.waitUntil(async () => {
    const currentUrl = await browser.getUrl();
    return currentUrl.includes(`/${repoName}`);
    }, { timeout: 10000, timeoutMsg: 'Did not redirect to new repository page' });

    const repoTitle = await $('h1');
    const repoTitleText = await repoTitle.getText();
    expect(repoTitleText).toContain(repoName);
    console.log('Repository created successfully!');
  });
});
```