Cucumber BDD Framework for Playwright

📌 Project Structure

```
playwright-bdd-framework
|── src
│   ├── test
│   │   ├── features
│   │   │   ├── web.feature
│   │   │   ├── mobile.feature
│   │   │   ├── api.feature
│   │   ├── stepDefinitions
│   │   │   ├── WebSteps.ts
│   │   │   ├── MobileSteps.ts
│   │   │   ├── ApiSteps.ts
│   │   ├── pages
│   │   │   ├── WebPage.ts
│   │   │   ├── MobilePage.ts
│   │   │   ├── ApiHelper.ts
│   ├── support
│   │   ├── hooks.ts
|── playwright.config.ts
|── package.json
|── tsconfig.json
|── cucumber.js
|── README.md
|── .github/workflows/playwright-ci.yml (for GitHub Actions CI/CD)
```

📌 Step 1: Install Dependencies

Run the following command to install the necessary dependencies:

```
npm init -y
npm install @playwright/test @cucumber/cucumber playwright chai
ts-node typescript
```

📌 Step 2: Configure `playwright.config.ts`

This configuration allows us to run tests on Web, Mobile, and API environments.

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  testDir: './src/test',
  timeout: 30000,
  reporter: 'html',
  projects: [
    {
      name: 'web',
      use: { browserName: 'chromium', headless: false },
    },
    {
      name: 'mobile',
      use: { browserName: 'chromium', viewport: { width: 375,
height: 667 } },
    },
    {
      name: 'api',
      use: { baseURL: 'https://jsonplaceholder.typicode.com' },
    },
  ],
});
```

📌 Step 3: Feature Files

🔹 Web Automation - `web.feature`

```
Feature: Web UI Testing with Playwright

Scenario: Verify Login Functionality
  Given I navigate to the login page
  When I enter "testuser" and "password123"
  And I click on the login button
  Then I should see the homepage
```

🔹 Mobile Automation - `mobile.feature`

```
Feature: Mobile Web Testing

Scenario: Verify Mobile Navigation
  Given I open the mobile web application
  When I navigate to the menu
  Then I should see the "Settings" option
```

🔹 API Testing - `api.feature`

```
Feature: API Testing with Playwright

Scenario: Verify API Response for User
  Given I make a GET request to "/users/1"
  Then the response status should be 200
  And the response should contain "Leanne Graham"
```

📌 Step 4: Step Definitions

🔷 Web Step Definition - `WebSteps.ts`

```
import { Given, When, Then } from '@cucumber/cucumber';
import { expect } from '@playwright/test';
import { WebPage } from '../pages/WebPage';

const webPage = new WebPage();

Given('I navigate to the login page', async () => {
  await webPage.openLoginPage();
});

When('I enter {string} and {string}', async (username, password) =>
{
  await webPage.enterCredentials(username, password);
});

When('I click on the login button', async () => {
  await webPage.clickLogin();
});

Then('I should see the homepage', async () => {
  await webPage.verifyHomePage();
});
```

🔹 **Mobile Step Definition - `MobileSteps.ts`**

```
import { Given, When, Then } from '@cucumber/cucumber';
import { expect } from '@playwright/test';
import { MobilePage } from '../pages/MobilePage';

const mobilePage = new MobilePage();

Given('I open the mobile web application', async () => {
  await mobilePage.openApp();
});

When('I navigate to the menu', async () => {
  await mobilePage.openMenu();
});

Then('I should see the {string} option', async (option) => {
  expect(await mobilePage.isOptionVisible(option)).toBeTruthy();
});
```

🔹 **API Step Definition - `ApiSteps.ts`**

```
import { Given, Then } from '@cucumber/cucumber';
import { request, expect } from '@playwright/test';

let response;

Given('I make a GET request to {string}', async (endpoint) => {
  const apiContext = await request.newContext();
  response = await apiContext.get(endpoint);
});

Then('the response status should be {int}', async (statusCode) => {
  expect(response.status()).toBe(statusCode);
});

Then('the response should contain {string}', async (expectedText) =>
{
  const responseBody = await response.json();
  expect(responseBody.name).toContain(expectedText);
});
```

📌 Step 5: Page Object Model (POM)

🔹 **Web Page Object - `WebPage.ts`**

```
import { Page } from '@playwright/test';

export class WebPage {
  constructor(private page: Page) {}

  async openLoginPage() {
    await this.page.goto('https://example.com/login');
  }

  async enterCredentials(username: string, password: string) {
    await this.page.fill('#username', username);
    await this.page.fill('#password', password);
  }

  async clickLogin() {
    await this.page.click('#loginBtn');
  }

  async verifyHomePage() {
    await this.page.waitForSelector('#home');
  }
}
```

📌 Step 6: CI/CD Pipeline (GitHub Actions)

Create `.github/workflows/playwright-ci.yml` for CI/CD execution:

```yaml
name: Playwright Tests

on:
  push:
    branches:
      - main

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repository
        uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: 16

      - name: Install Dependencies
        run: npm install

      - name: Run Web Tests
        run: npx playwright test --project=web

      - name: Run Mobile Tests
        run: npx playwright test --project=mobile

      - name: Run API Tests
        run: npx playwright test --project=api
```

📌 Step 7: Running the Tests

Execute all tests using:

```
npx cucumber-js
```

Run specific tests:

```
npx playwright test --project=web
npx playwright test --project=mobile
npx playwright test --project=api
```

✅ Conclusion

- **Web UI Testing** with **Playwright + Cucumber BDD**
- **Mobile Web Testing** using **Viewport & Playwright**
- **API Testing** using **Playwright APIRequest**
- **CI/CD Integration** with **GitHub Actions**

With this **single Playwright framework**, you can efficiently test **Web, Mobile, and API** within one repository.
