Playwright testing Framework with BDD

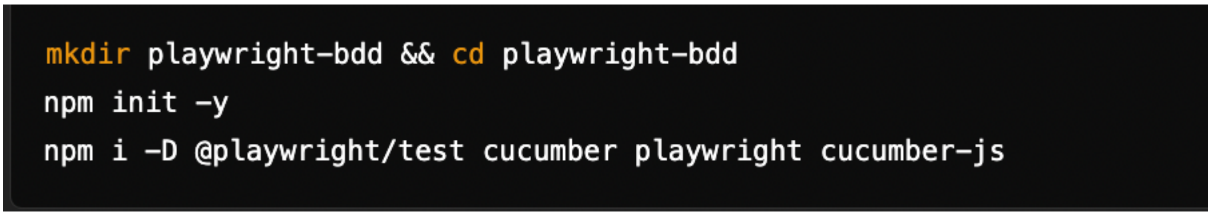
Here's a **detailed tutorial on Playwright testing** where we will:
✅ Set up a **BDD framework** using **Cucumber with Playwright**
✅ Write a **script for login, adding a product to cart, and checkout on Flipkart**
✅ Use **global setup** with **storage state** to bypass login

---

1️⃣ Project Setup

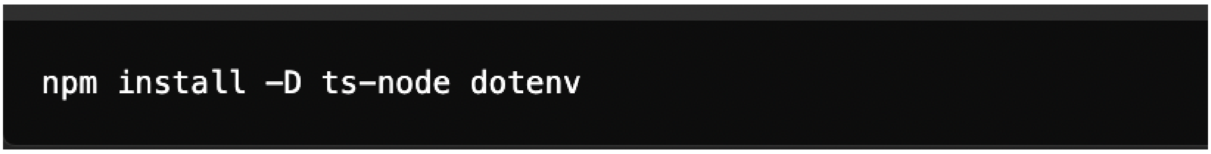
Install Playwright & Cucumber

Run the following commands to initialize a Playwright project and install Cucumber:



```
mkdir playwright-bdd && cd playwright-bdd
npm init -y
npm i -D @playwright/test cucumber playwright cucumber-js
```

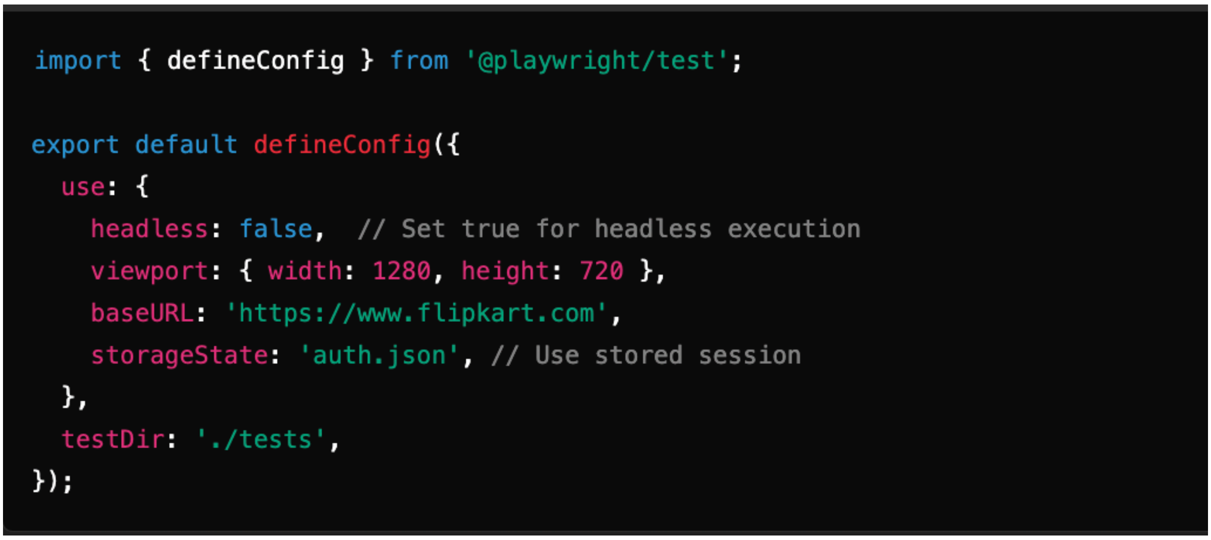
Install Required Dependencies



```
npm install -D ts-node dotenv
```

2️⃣ Configure Playwright in `playwright.config.ts`

Create a Playwright configuration file (`playwright.config.ts`) to define browsers, timeouts, and storage state:

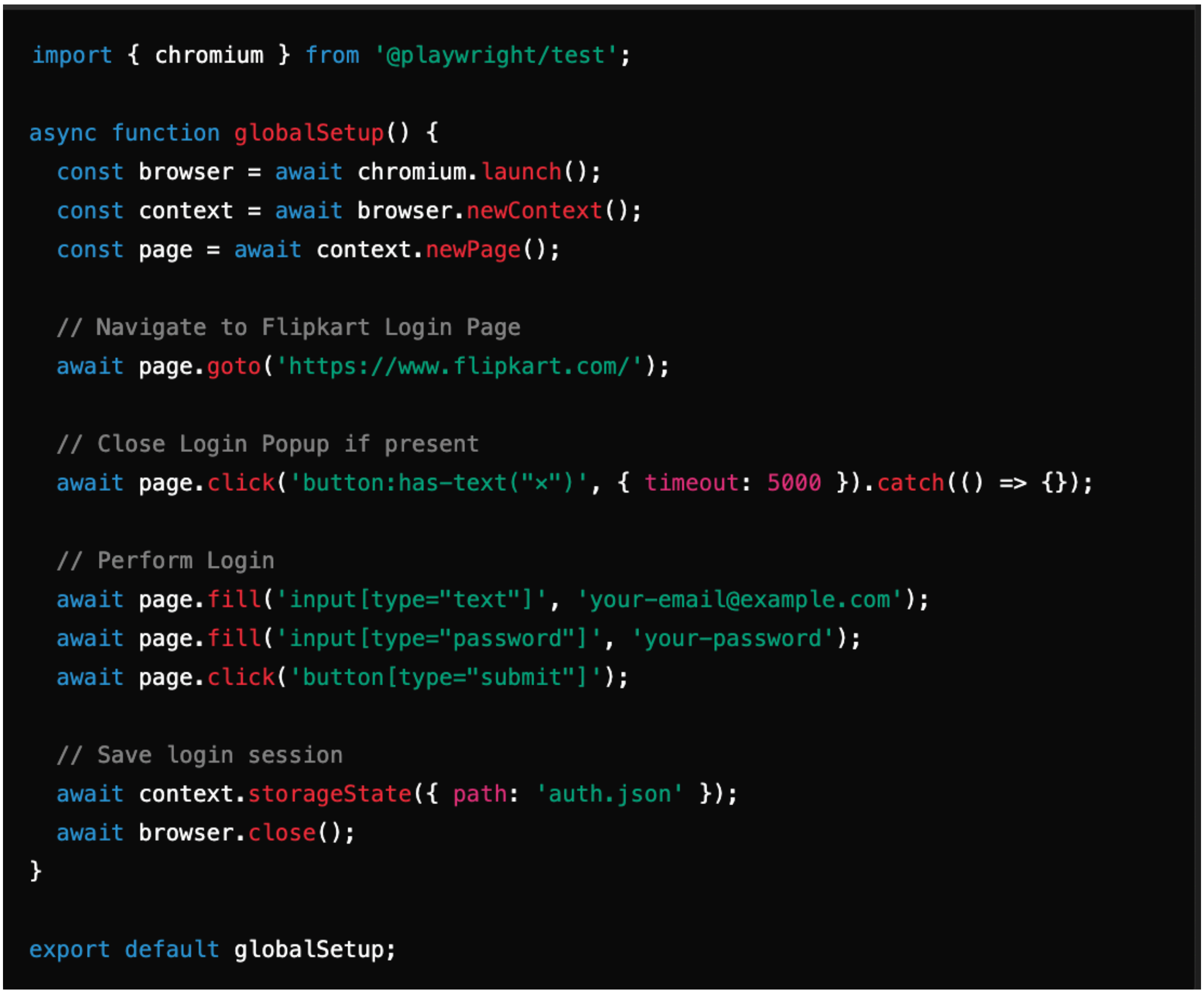


```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  use: {
    headless: false,  // Set true for headless execution
    viewport: { width: 1280, height: 720 },
    baseURL: 'https://www.flipkart.com',
    storageState: 'auth.json', // Use stored session
  },
  testDir: './tests',
});
```

3️⃣ Global Setup to Bypass Login

To avoid repeated login during tests, we create a global setup file.

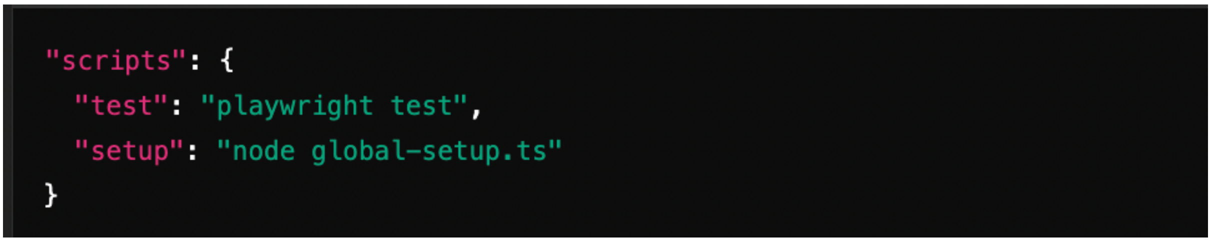
Create `global-setup.ts`



```ts
import { chromium } from '@playwright/test';

async function globalSetup() {
  const browser = await chromium.launch();
  const context = await browser.newContext();
  const page = await context.newPage();

  // Navigate to Flipkart Login Page
  await page.goto('https://www.flipkart.com/');

  // Close Login Popup if present
  await page.click('button:has-text("×")', { timeout: 5000 }).catch(() => {});

  // Perform Login
  await page.fill('input[type="text"]', 'your-email@example.com');
  await page.fill('input[type="password"]', 'your-password');
  await page.click('button[type="submit"]');

  // Save login session
  await context.storageState({ path: 'auth.json' });
  await browser.close();
}

export default globalSetup;
```

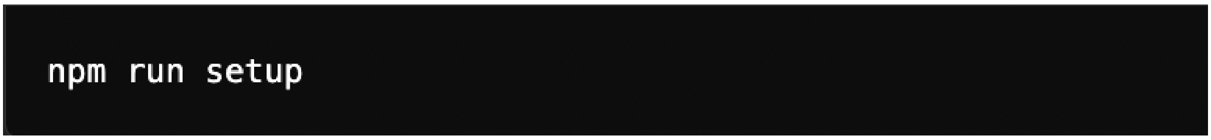
Modify `package.json` to Run Global Setup

Add this script under `scripts` in `package.json`:



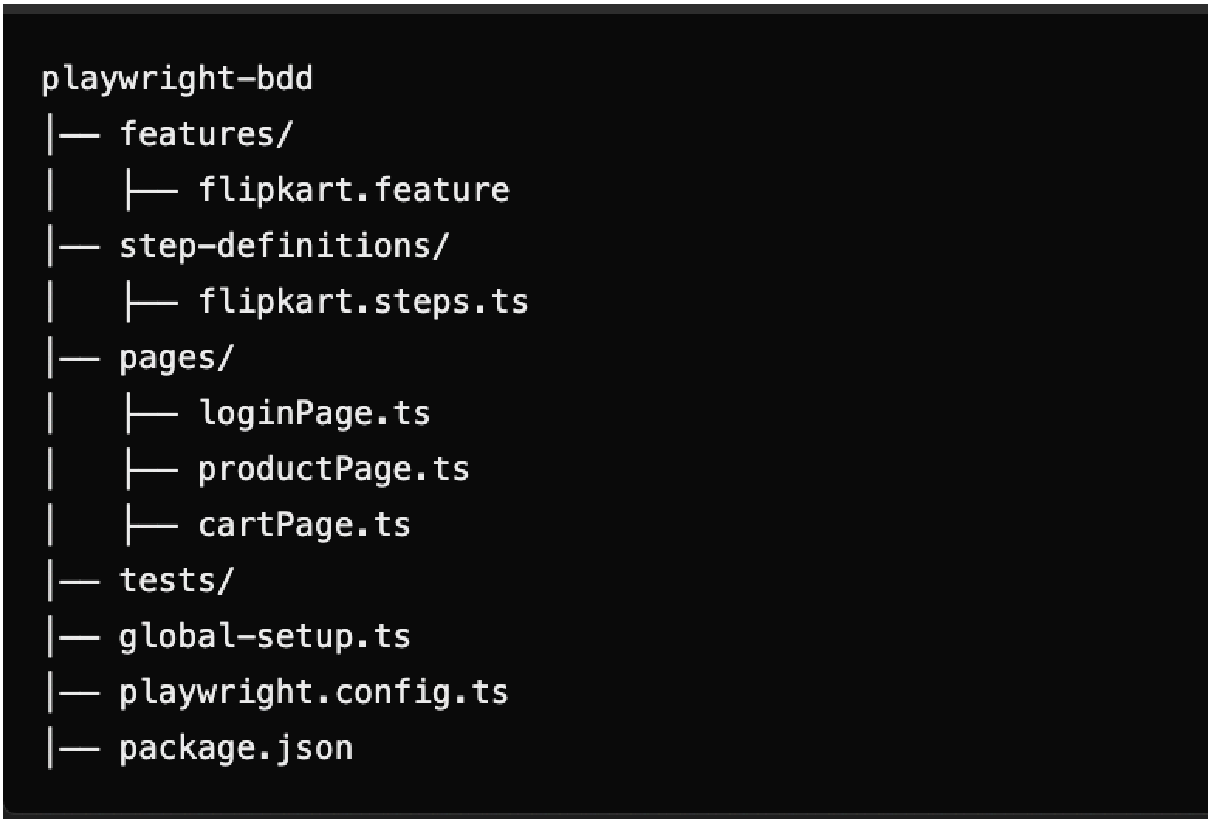
```
"scripts": {
  "test": "playwright test",
  "setup": "node global-setup.ts"
}
```

Run the setup before tests:



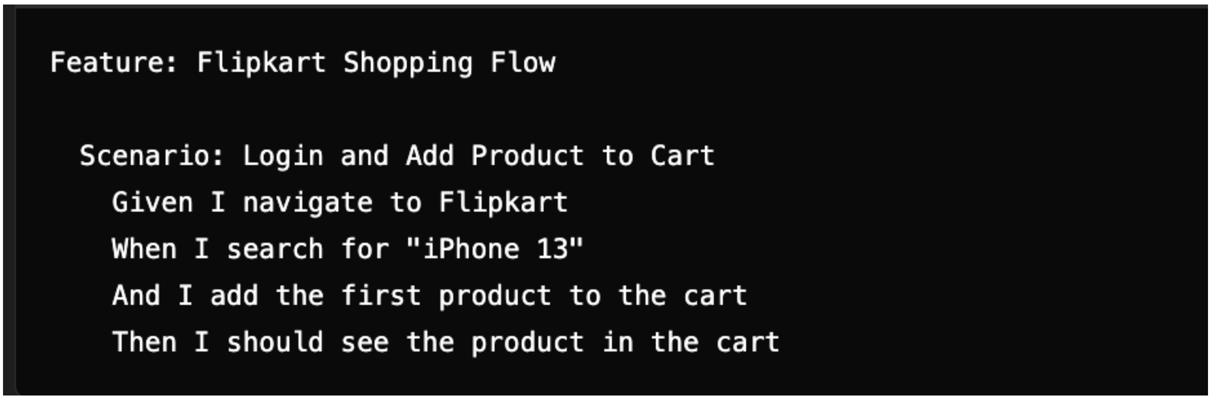
```
npm run setup
```

4️⃣ Create BDD Folder Structure



```
playwright-bdd
|── features/
|    ├── flipkart.feature
|── step-definitions/
|    ├── flipkart.steps.ts
|── pages/
|    ├── loginPage.ts
|    ├── productPage.ts
|    ├── cartPage.ts
|── tests/
|── global-setup.ts
|── playwright.config.ts
|── package.json
```

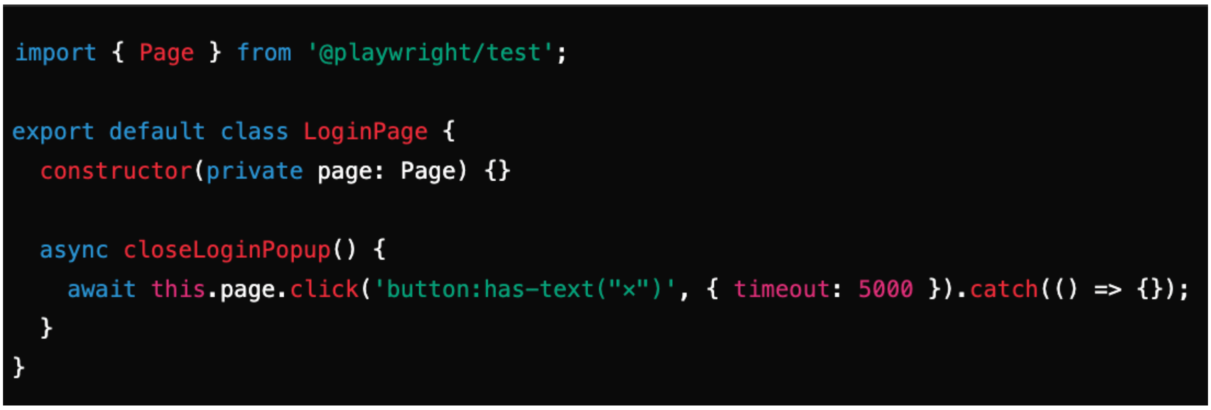
5️⃣ Create Feature File (`features/flipkart.feature`)



```
Feature: Flipkart Shopping Flow

  Scenario: Login and Add Product to Cart
    Given I navigate to Flipkart
    When I search for "iPhone 13"
    And I add the first product to the cart
    Then I should see the product in the cart
```

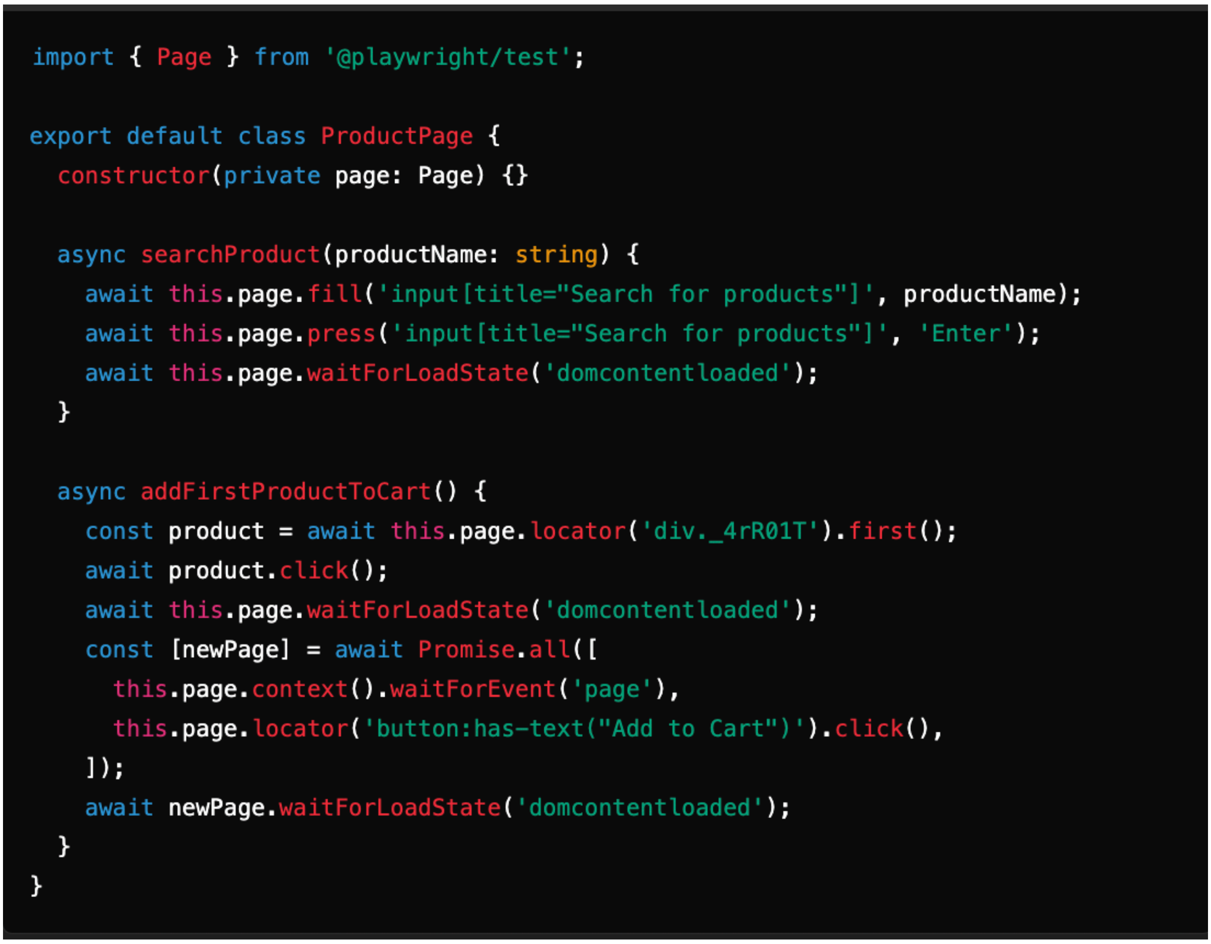
6️⃣ Create Page Object Model (POM)

Login Page (`pages/loginPage.ts`)



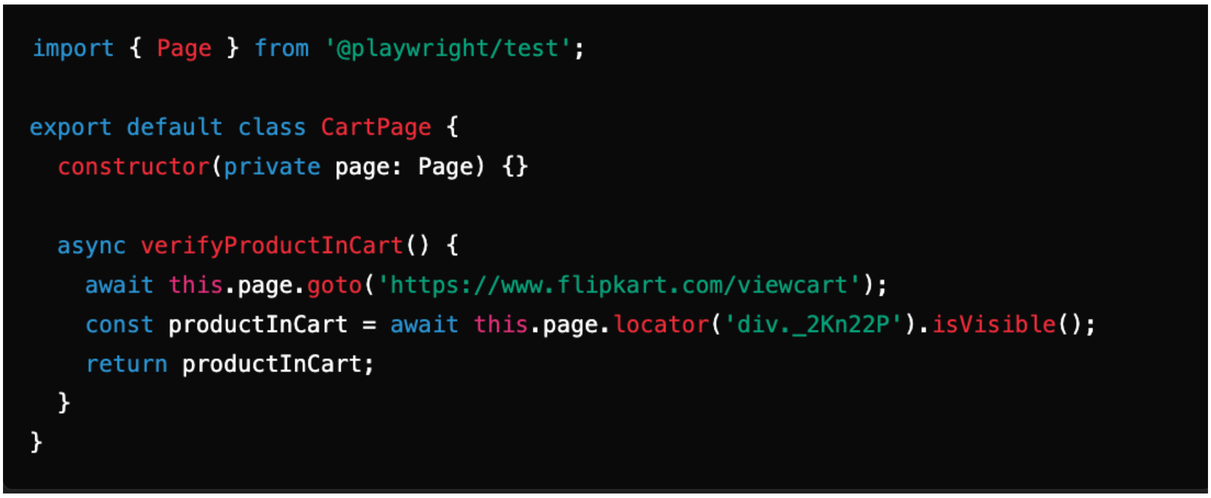
```
import { Page } from '@playwright/test';

export default class LoginPage {
  constructor(private page: Page) {}

  async closeLoginPopup() {
    await this.page.click('button:has-text("×")', { timeout: 5000 }).catch(() => {});
  }
}
```

Product Page (`pages/productPage.ts`)



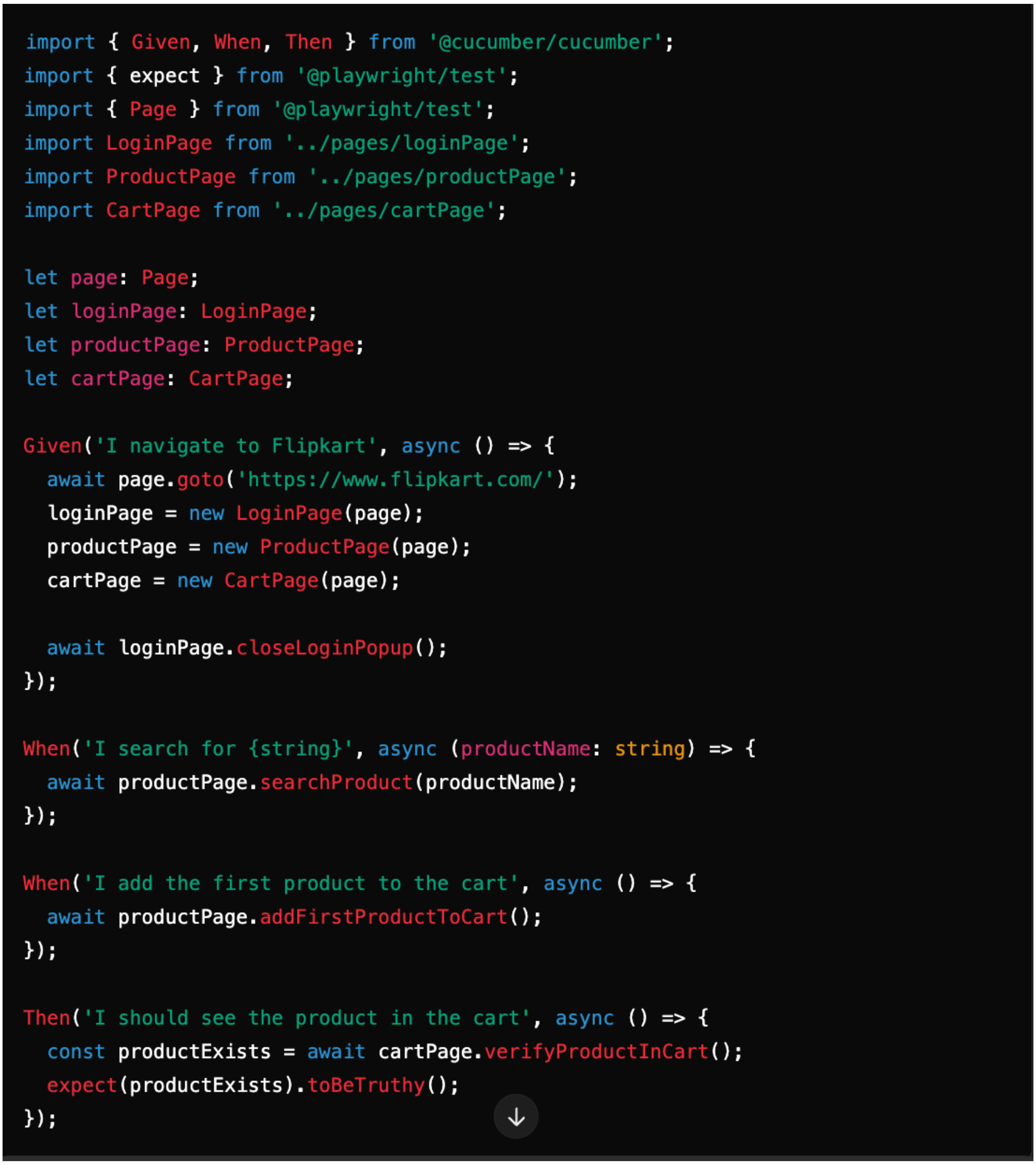
```typescript
import { Page } from '@playwright/test';

export default class ProductPage {
  constructor(private page: Page) {}

  async searchProduct(productName: string) {
    await this.page.fill('input[title="Search for products"]', productName);
    await this.page.press('input[title="Search for products"]', 'Enter');
    await this.page.waitForLoadState('domcontentloaded');
  }

  async addFirstProductToCart() {
    const product = await this.page.locator('div._4rR01T').first();
    await product.click();
    await this.page.waitForLoadState('domcontentloaded');
    const [newPage] = await Promise.all([
      this.page.context().waitForEvent('page'),
      this.page.locator('button:has-text("Add to Cart")').click(),
    ]);
    await newPage.waitForLoadState('domcontentloaded');
  }
}
```

Cart Page (`pages/cartPage.ts`)



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

export default class CartPage {
  constructor(private page: Page) {}

  async verifyProductInCart() {
    await this.page.goto('https://www.flipkart.com/viewcart');
    const productInCart = await this.page.locator('div._2Kn22P').isVisible();
    return productInCart;
  }
}
```

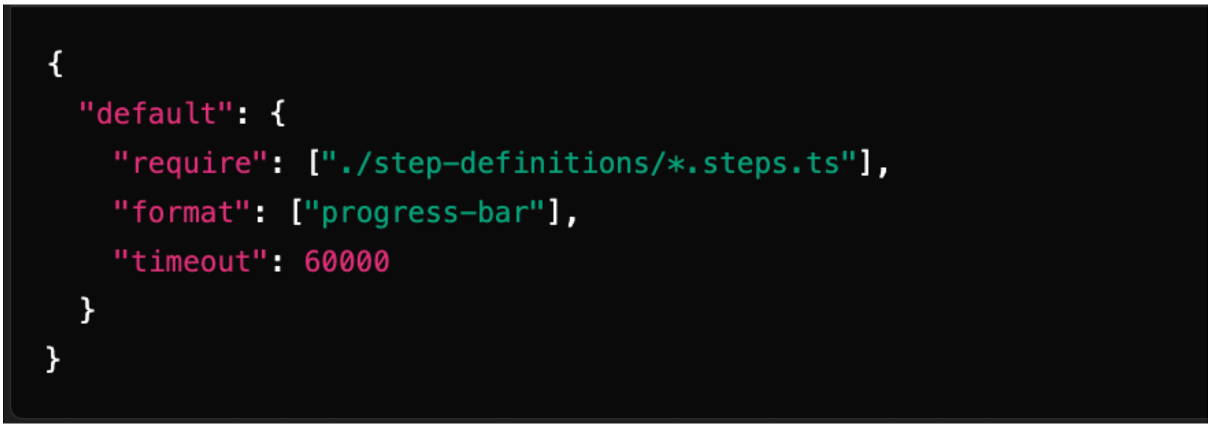
7️⃣ Write Step Definitions (`step-definitions/flipkart.steps.ts`)



```
import { Given, When, Then } from '@cucumber/cucumber';
import { expect } from '@playwright/test';
import { Page } from '@playwright/test';
import LoginPage from '../pages/loginPage';
import ProductPage from '../pages/productPage';
import CartPage from '../pages/cartPage';

let page: Page;
let loginPage: LoginPage;
let productPage: ProductPage;
let cartPage: CartPage;

Given('I navigate to Flipkart', async () => {
  await page.goto('https://www.flipkart.com/');
  loginPage = new LoginPage(page);
  productPage = new ProductPage(page);
  cartPage = new CartPage(page);

  await loginPage.closeLoginPopup();
});

When('I search for {string}', async (productName: string) => {
  await productPage.searchProduct(productName);
});

When('I add the first product to the cart', async () => {
  await productPage.addFirstProductToCart();
});

Then('I should see the product in the cart', async () => {
  const productExists = await cartPage.verifyProductInCart();
  expect(productExists).toBeTruthy();
});
```

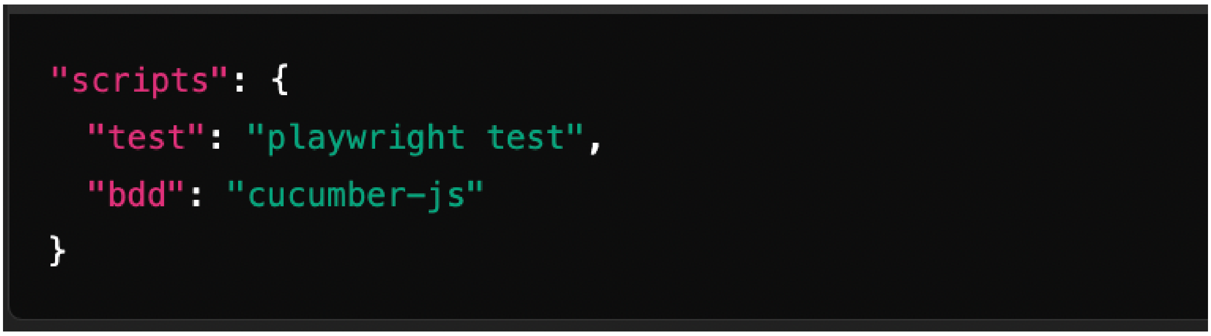
8️⃣ Configure Cucumber to Run Tests

Create `cucumber.js` file in the root folder:



```
{
  "default": {
    "require": ["./step-definitions/*.steps.ts"],
    "format": ["progress-bar"],
    "timeout": 60000
  }
}
```

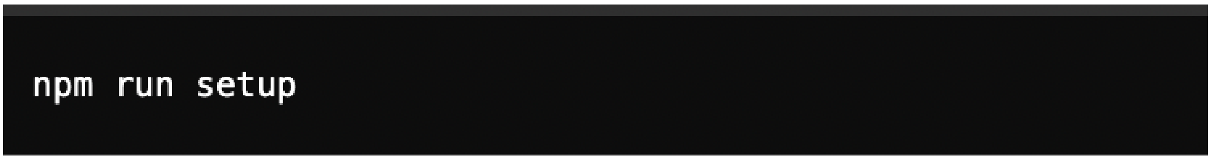
Modify `package.json`:



```
"scripts": {
  "test": "playwright test",
  "bdd": "cucumber-js"
}
```

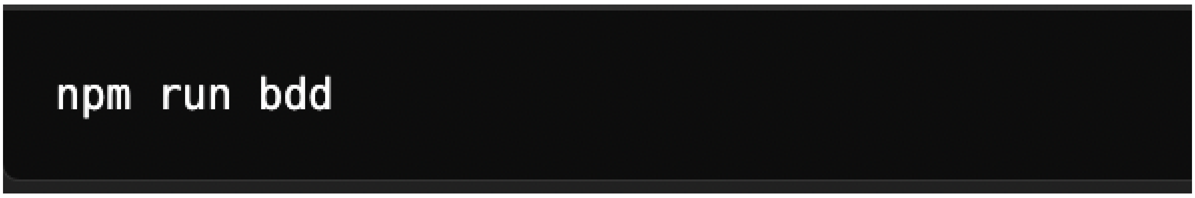
9️⃣ Run the BDD Tests

First, ensure global setup is done:



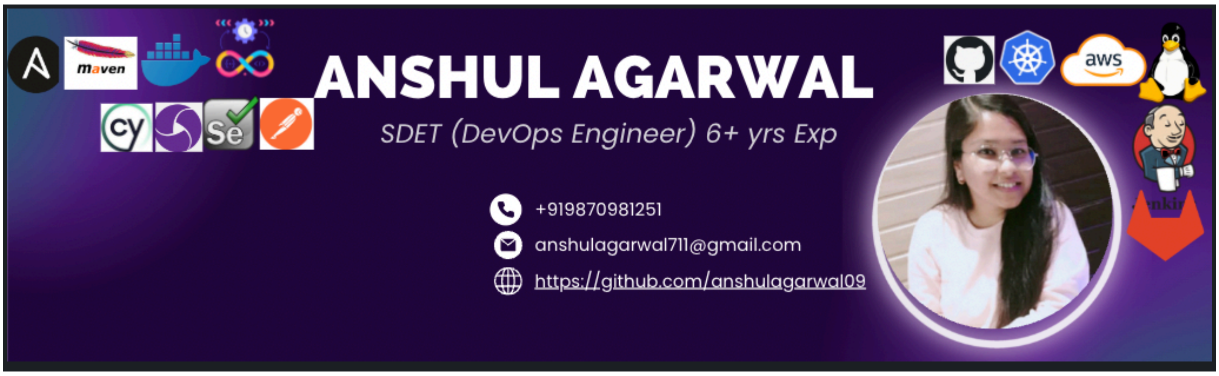
```
npm run setup
```

Run the test using Cucumber:



```
npm run bdd
```

Happy Learning! 🌸


ANSHUL AGARWAL
SDET (DevOps Engineer) 6+ yrs Exp
+919870981251
anshulagarwal711@gmail.com
https://github.com/anshulagarwal09
maven
aws