Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Факультет Инфокоммуникационных Технологий

Лабораторная работа №3

по дисциплине «Тестирование программного обеспечения»

Выполнила:

Соколова Дарья Максимовна

Группа: K3323

Проверил:

Кочубеев Николай Сергеевич

Санкт-Петербург,

2024

Цель работы

Научиться проектировать, писать и применять end-to-end (E2E) тесты для проверки работы всей системы или ключевых пользовательских сценариев. Провести анализ пользовательских путей и подготовить отчёт о проведённом тестировании.

Задачи

1. Выбрать репозиторий с GitHub;
2. Проанализировать пользовательские сценарии;
3. Написать E2E тесты;
4. Составить отчет по выполненной работе.

Ход работы

1. Выбор репозитория с GitHub

Для выполнения лабораторной работы был выбран проект https://github.com/Lada496/to-do-list-with-auth-public. Этот проект написан на React, представляет собой Todo-list с авторизацией.

2. Анализ пользовательских сценариев

Ключевые пользовательские пути:

1. *Регистрация пользователя*

- Пользователь вводит данные (email, пароль, подтверждение пароля) и нажимает кнопку «Sign up»;
- При успешной регистрации пользователь перенаправляется на главную страницу с задачами;
- Должна быть реализована проверка регистрации с валидными и невалидными данными.

2. *Авторизация пользователя*

   – Пользователь вводит email и пароль и нажимает кнопку «Login»;
   – При успешной авторизации пользователь перенаправляется на главную страницу с задачами;
   – Должна быть реализована проверка авторизации с валидными и невалидными данными.

3. *Управление задачами*

   – Пользователь может добавлять и удалять задачи;
   – Должна быть реализована проверка корректного добавления, отображения и удаления задач.

3. Написание E2E тестов

Тесты были написаны с использованием библиотеки Playwright.

Ниже представлены созданные тесты:

1. Тест на регистрацию с невалидными данными email и пароля (рисунок 1). Ожидается, что на экране появятся сообщения об ошибках.

```
test('User cannot register with invalid email and password', async ({ page }) => {
  await page.goto('https://to-do2-f439d9.netlify.app');
  await page.click('button.auth__btn >> text=Sign Up');

  await page.fill('input[name="email"]', 'user');
  await page.fill('input[name="password1"]', '123');
  await page.fill('input[name="password2"]', '123');

  await expect(page.locator('text=Please enter a valid email')).toBeVisible();
  await expect(page.locator('text=Password should be at least 6 letters')).toBeVisible();
});
```

Рисунок 1 - Тест на регистрацию с невалидными данными

2. Тест на регистрацию с валидными данными (рисунок 2). Ожидается переход на главную страницу с задачами.

```
test('User can register with valid data', async ({ page }) => {
  await page.goto('https://to-do2-f439d9.netlify.app');
  await page.click('button.auth__btn >> text=Sign Up');

  await page.fill('input[name="email"]', 'user0@example.com');
  await page.fill('input[name="password1"]', 'Test1234');
  await page.fill('input[name="password2"]', 'Test1234');

  await page.click('button[type="submit"]');

  await expect(page).toHaveURL('https://to-do2-f439d9.netlify.app/home');
});
```

Рисунок 2 - Тест на регистрацию с валидными данными

3. Тест на авторизацию с валидными данными (рисунок 3). Так же ожидается переход на главную страницу с задачами.

```
test('User can login with valid data', async ({ page }) => {
  await page.goto('https://to-do2-f439d9.netlify.app/auth');
  await page.click('button.auth__btn >> text=Login');

  await page.fill('input[name="email"]', 'user0@example.com');
  await page.fill('input[name="password"]', 'Test1234');
  await page.click('button[type="submit"]');

  await expect(page).toHaveURL('https://to-do2-f439d9.netlify.app/home');
});
```

Рисунок 3 - Тест на авторизацию с валидными данными

4. Тест на авторизацию с невалидными данными попыткой войти с некорректным паролем (рисунок 4). Ожидается сообщение об ошибке.

```
test('User cannot login with invalid data', async ({ page }) => {
  await page.goto('https://to-do2-f439d9.netlify.app/auth');
  await page.click('button.auth__btn >> text=Login');

  await page.fill('input[name="email"]', 'user0@example.com');
  await page.fill('input[name="password"]', '123');
  await page.click('button[type="submit"]');

  await expect(page.locator('text=Firebase: Error (auth/wrong-password).')).toBeVisible();
});
```

Рисунок 4 – Тест на авторизацию с невалидными данными

5. Тест на создание задачи и ее корректное отображение после добавления (рисунок 5).

```
test('User can create a task', async ({ page }) => {
  await page.goto('https://to-do2-f439d9.netlify.app');
  await page.click('button.auth__btn >> text=Login');

  await page.fill('input[name="email"]', 'user0@example.com');
  await page.fill('input[name="password"]', 'Test1234');
  await page.click('button[type="submit"]');

  await page.click('button.open');
  await page.fill('input[name="title"]', 'First Task');
  await page.fill('textarea[name="description"]', 'First Task Description');
  await page.click('button.add');

  await expect(page.locator('text=First Task')).toBeVisible();
  await expect(page.locator('text=First Task Description')).toBeVisible();
});
```

Рисунок 5 – Тест на создание задачи

6. Тест на удаление задачи и того, что она больше не отображается (рисунок 6).

```
test('User can delete a task', async ({ page }) => {
  await page.goto('https://to-do2-f439d9.netlify.app');
  await page.click('button.auth__btn >> text=Login');

  await page.fill('input[name="email"]', 'user0@example.com');
  await page.fill('input[name="password"]', 'Test1234');
  await page.click('button[type="submit"]');

  await page.click('button.open');
  await page.fill('input[name="title"]', 'Task to Delete');
  await page.fill('textarea[name="description"]', 'Description to Delete');
  await page.click('button.add');

  await page.click('ul.list li.item:last-child button.item__btn');

  await expect(page.locator('text=Task to Delete')).not.toBeVisible();
  await expect(page.locator('text=Description to Delete')).not.toBeVisible();
});
```

Рисунок 6 – Тест на удаление задачи

7. Тест на выход из системы (рисунок 7). Ожидается, что после выхода пользователь попадает на страницу авторизации.

```
test('User can logout', async ({ page }) => {
  await page.goto('https://to-do2-f439d9.netlify.app');
  await page.click('button.auth__btn >> text=Login');

  await page.fill('input[name="email"]', 'user0@example.com');
  await page.fill('input[name="password"]', 'Test1234');
  await page.click('button[type="submit"]');

  await page.click('button.logout');

  await expect(page).toHaveURL('https://to-do2-f439d9.netlify.app/auth');
});
```

Рисунок 7 – Тест на выход из системы

Перечень протестированных пользовательских сценариев:

1. Регистрация с валидными и невалидными данными;
2. Авторизация с валидными и невалидными данными;
3. Создание задачи;
4. Удаление задачи;
5. Выход из системы.

Эти сценарии охватывают основные действия, которые ожидаются от пользователей.

Видим, что тесты выполнились успешно (рисунок 8). Это показывает, что приложение обеспечивает хороший пользовательский опыт, позволяя пользователям легко регистрироваться, входить в систему, создавать и управлять задачами. Приложение корректно обрабатывает как валидные, так и невалидные данные, предоставляя пользователям четкие сообщения об ошибках.

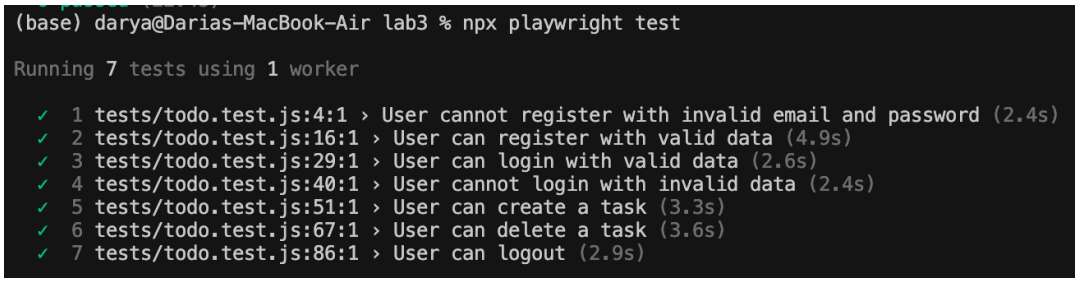

Рисунок 8 – Запуск тестов

Вывод

В данной лабораторной работе был выбран репозиторий с GitHub, были спроектированы и написаны end-to-end (E2E) тесты для проверки работы всей системы или ключевых пользовательских сценариев, а также составлен отчет по выполненной работе.

Ссылка на репозиторий с тестами:
https://github.com/daryasokolova04/testing_PO