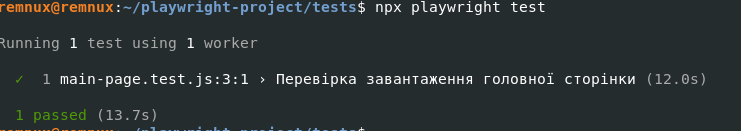
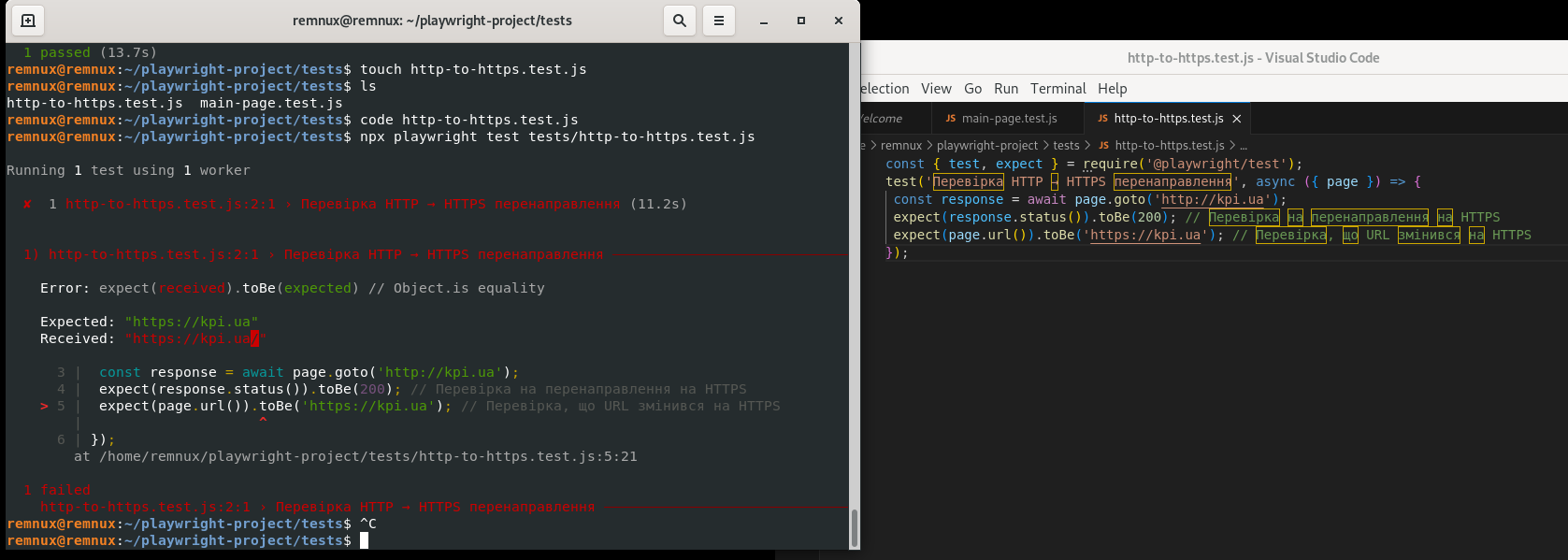
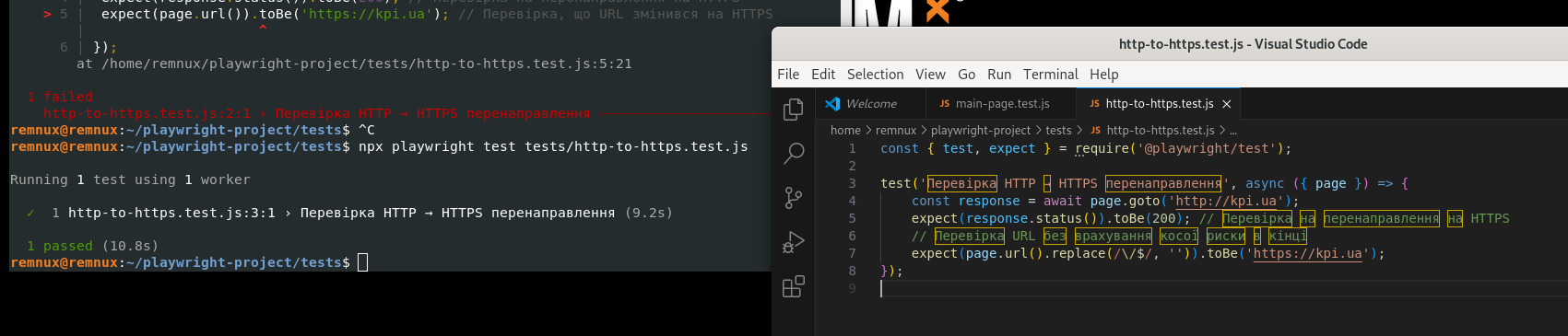
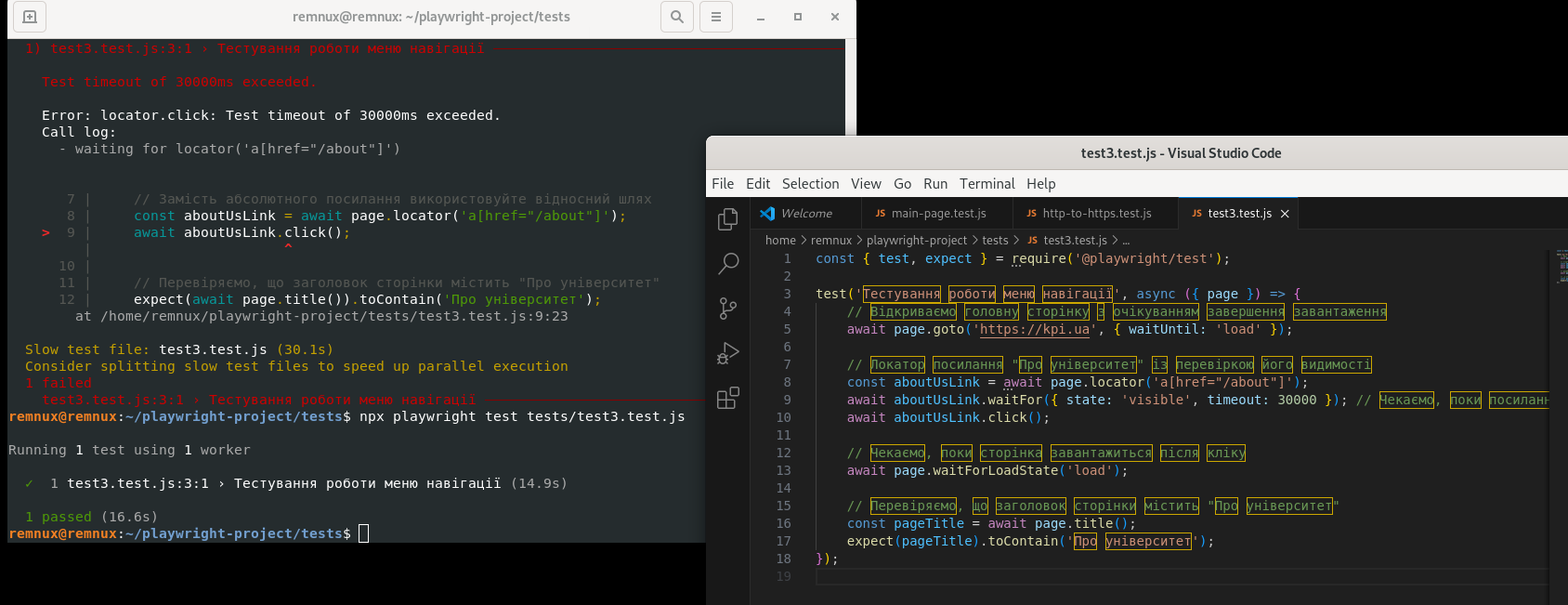
**Завдання 1**

Виконалося відповідно до наведеного коду у файлі, наданому лектором  
  
**Завдання 2**

Виникла помилка, тому дещо зміню тест: прибрала косу риску у кінці посилання.  


**Завдання 3**

Відразу скоригувала і додала певні рядки у тест, бо він не проходився - було занадто мало часу. Виправлений код і результат тесту:  


Це приклад одного пункту меню, аналогічно можна перевірити кожен із близько 100 пунктів тільки основного меню (не кажучи за побічні).

**Завдання 4**  
Чомусь цей теж тест у мене не дуже працював, я дописала його

сonst { test, expect } = require('@playwright/test');

test('Перевірка внутрішніх посилань', async ({ page }) => {

test.setTimeout(120000); // Збільшуємо тайм-аут до 120 секунд

await page.goto('https://kpi.ua');

// Збираємо всі внутрішні посилання

const links = await page.locator('a[href^="/"]');

const totalLinks = await links.count();

console.log(`Знайдено ${totalLinks} внутрішніх посилань.`);

// Масив обіцянок для паралельної перевірки

const checkLinks = [];

for (let i = 0; i < totalLinks; i++) {

const link = links.nth(i);

const href = await link.getAttribute('href');

if (!href) {

console.warn(`Пропущено посилання з індексом ${i}, оскільки href відсутній.`);

continue;

}

// Додаємо перевірку кожного посилання до масиву

checkLinks.push(

page.request.get(`https://kpi.ua${href}`).then(response => {

console.log(`Перевіряємо посилання: ${href} -> Статус: ${response.status()}`);

expect(response.status()).toBe(200); // Очікуємо статус 200

}).catch(error => {

console.error(`Помилка при перевірці посилання ${href}:`, error);

})

);

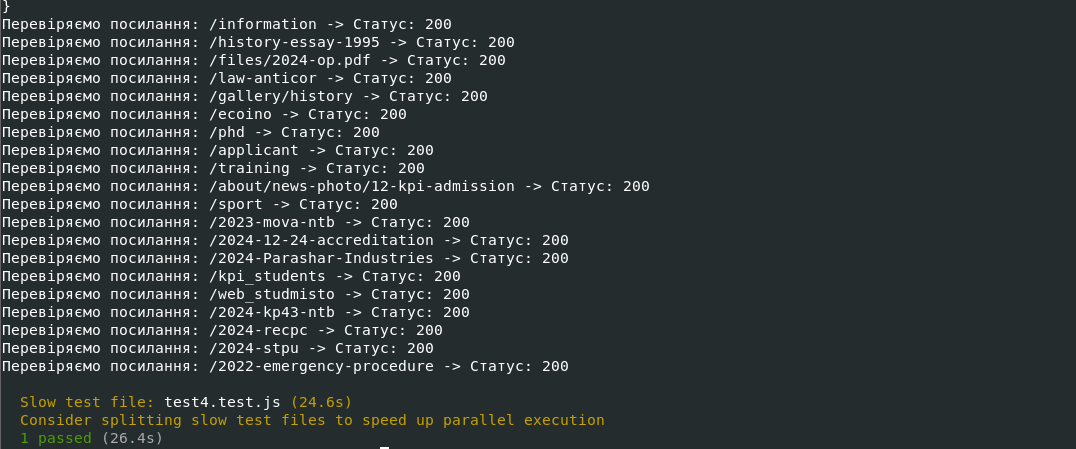
}

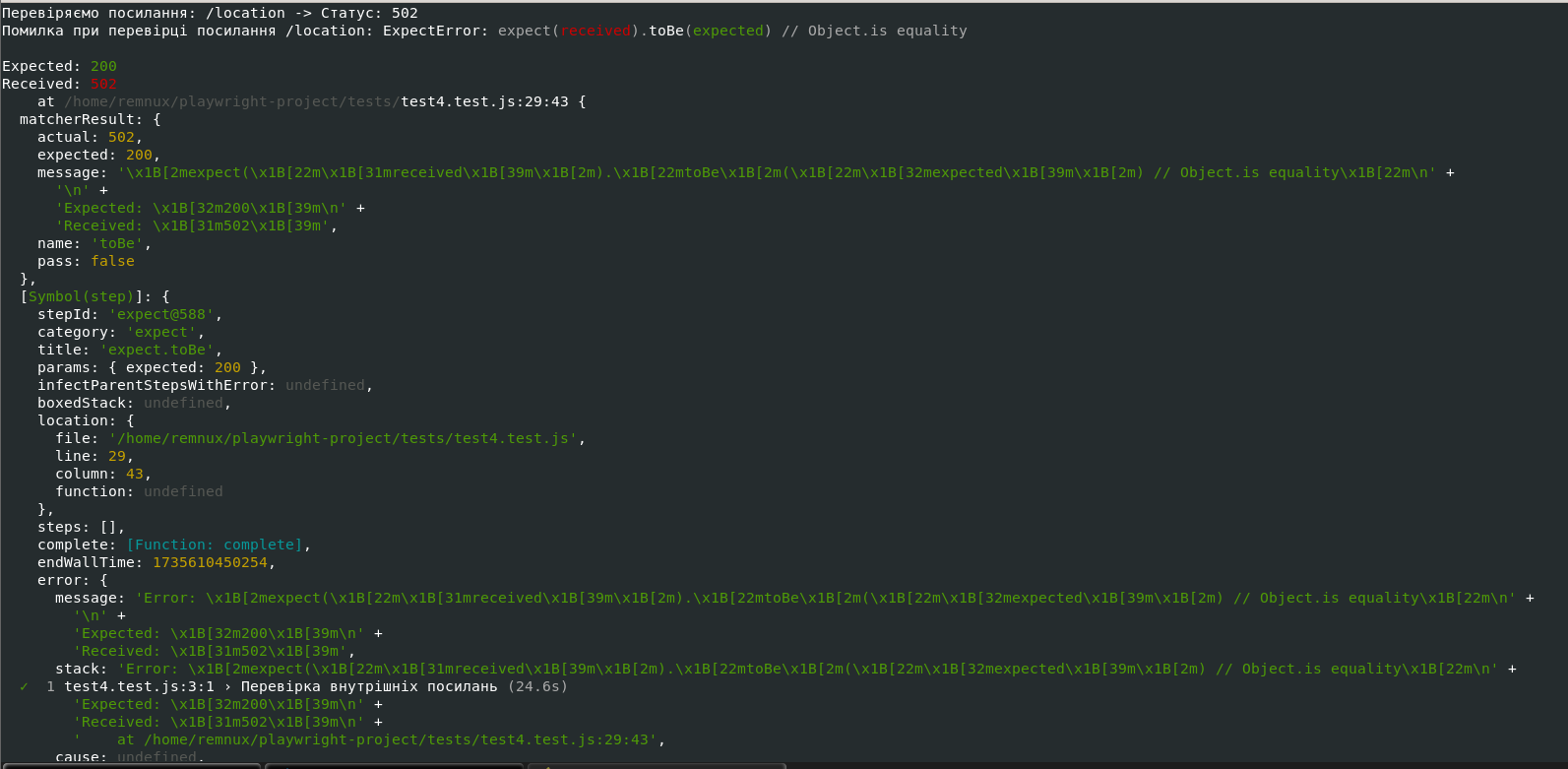
// Виконуємо всі перевірки одночасно

await Promise.all(checkLinks);

});

І ось результат:





Результати дуже об’ємні , тому прикріплюю лише частинку.

**Завдання 5**

Код тесту:

const { test, expect } = require('@playwright/test');

test('Тестування адаптивності сторінки', async ({ page }) => {

// Відкриваємо сайт

await page.goto('https://kpi.ua');

// Масив конфігурацій для пристроїв

const devices = [

{ name: 'Десктоп', width: 1200, height: 800 },

{ name: 'Планшет', width: 768, height: 1024 },

{ name: 'Смартфон', width: 375, height: 667 },

];

// Цикл по пристроях

for (const device of devices) {

console.log(`Тестуємо пристрій: ${device.name}`);

await page.setViewportSize({ width: device.width, height: device.height });

// Перевіряємо наявність основних елементів

const elementsToCheck = ['header', 'nav', 'footer'];

for (const selector of elementsToCheck) {

const isVisible = await page.isVisible(selector);

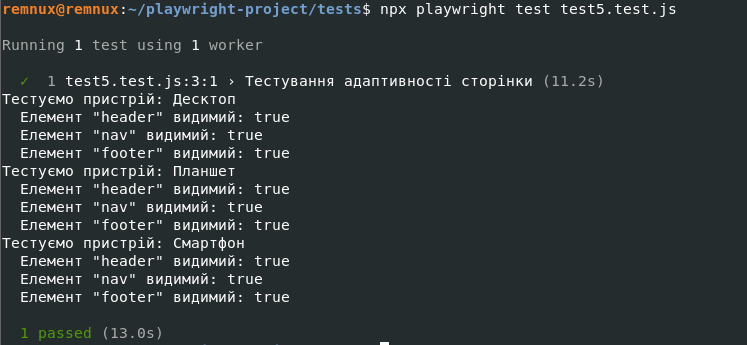
console.log(` Елемент "${selector}" видимий: ${isVisible}`);

expect(isVisible).toBe(true); // Перевіряємо, що елемент присутній

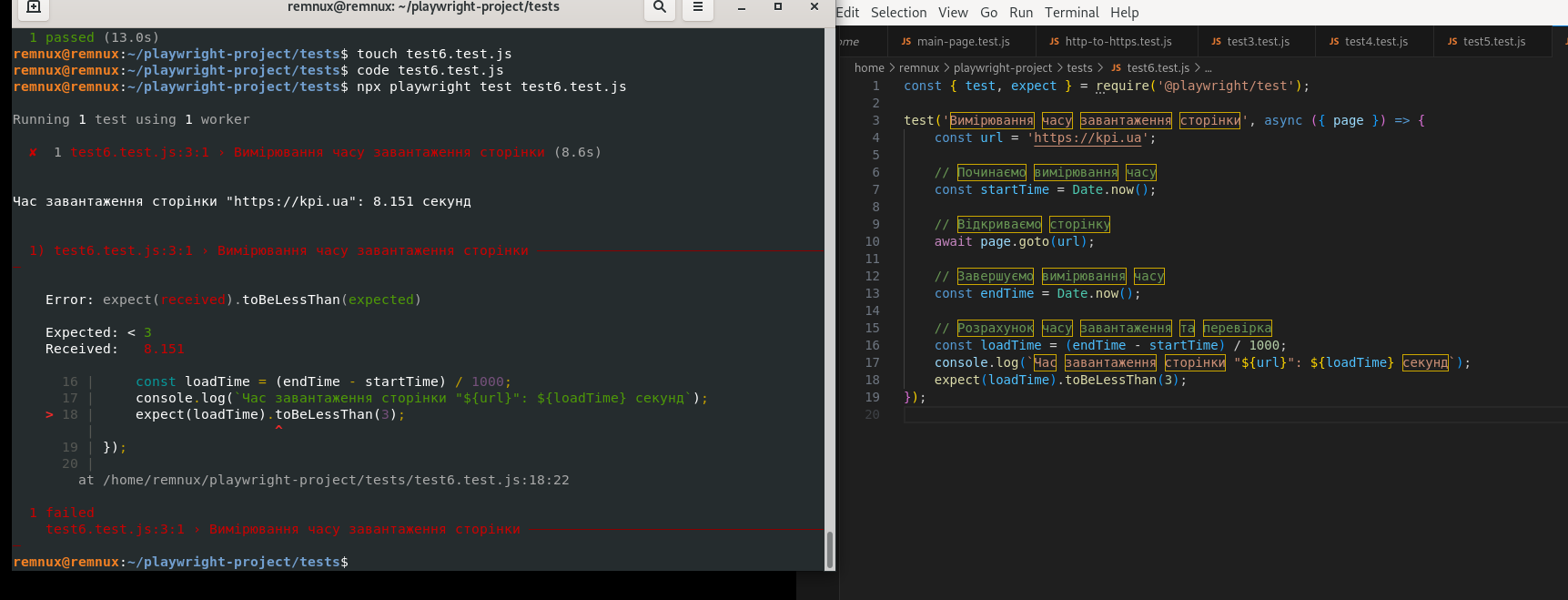
}

}

});

Результат виконання:

**Завдання 6**

Код тесту для перевірки та сама перевірка:  


У мене не вдався тест, скоріше всього через не дуже швидкий інтернет, тому час завантаження трохи більше 8с, замість рекомедованих не більше 3с.

Код:

const { test, expect } = require('@playwright/test');

test('Вимірювання часу завантаження сторінки', async ({ page }) => {

const url = 'https://kpi.ua';

// Починаємо вимірювання часу

const startTime = Date.now();

// Відкриваємо сторінку

await page.goto(url);

// Завершуємо вимірювання часу

const endTime = Date.now();

// Розрахунок часу завантаження та перевірка

const loadTime = (endTime - startTime) / 1000;

console.log(`Час завантаження сторінки "${url}": ${loadTime} секунд`);

expect(loadTime).toBeLessThan(3);

});

**Завдання 7**

Код тесту:  
const { test, expect } = require('@playwright/test');

test('Перевірка основних елементів на сторінці', async ({ page }) => {

const url = 'https://kpi.ua';

// Відкриваємо сторінку

await page.goto(url);

// Перевіряємо наявність логотипу

console.log('Перевіряємо логотип...');

await page.waitForSelector('img[alt="Головна"]', { state: 'visible' });

const logo = page.locator('img[alt="Головна"]');

expect(await logo.isVisible()).toBe(true);

console.log('Логотип присутній і видимий.');

// Перевіряємо головне меню навігації

console.log('Перевіряємо головне меню...');

await page.waitForSelector('#block-thex-mainnavigation', { state: 'visible' });

const mainMenu = page.locator('#block-thex-mainnavigation');

expect(await mainMenu.isVisible()).toBe(true);

console.log('Головне меню присутнє і видиме.');

// Перевіряємо блок новин або оголошень

console.log('Перевіряємо блок новин...');

await page.waitForSelector('.all', { state: 'visible' });

const newsBlock = page.locator('.all');

expect(await newsBlock.isVisible()).toBe(true);

console.log('Блок новин присутній і видимий.');

// Перевіряємо пошуковий рядок

console.log('Перевіряємо пошуковий рядок...');

await page.waitForSelector('.srch > a:nth-child(1)', { state: 'visible' });

const searchBox = page.locator('.srch > a:nth-child(1)');

expect(await searchBox.isVisible()).toBe(true);

console.log('Пошуковий рядок присутній і видимий.');

// Перевіряємо, що елементи залишаються видимими при зміні розміру екрана

const viewports = [

{ width: 1200, height: 800 }, // Десктоп

{ width: 768, height: 1024 }, // Планшет

{ width: 375, height: 667 } // Смартфон

];

for (const viewport of viewports) {

console.log(`Перевіряємо елементи для розміру екрану ${viewport.width}x${viewport.height}...`);

await page.setViewportSize(viewport);

expect(await logo.isVisible()).toBe(true);

console.log(`Логотип видимий для розміру екрану ${viewport.width}x${viewport.height}.`);

expect(await mainMenu.isVisible()).toBe(true);

console.log(`Головне меню видиме для розміру екрану ${viewport.width}x${viewport.height}.`);

expect(await newsBlock.isVisible()).toBe(true);

console.log(`Блок новин видимий для розміру екрану ${viewport.width}x${viewport.height}.`);

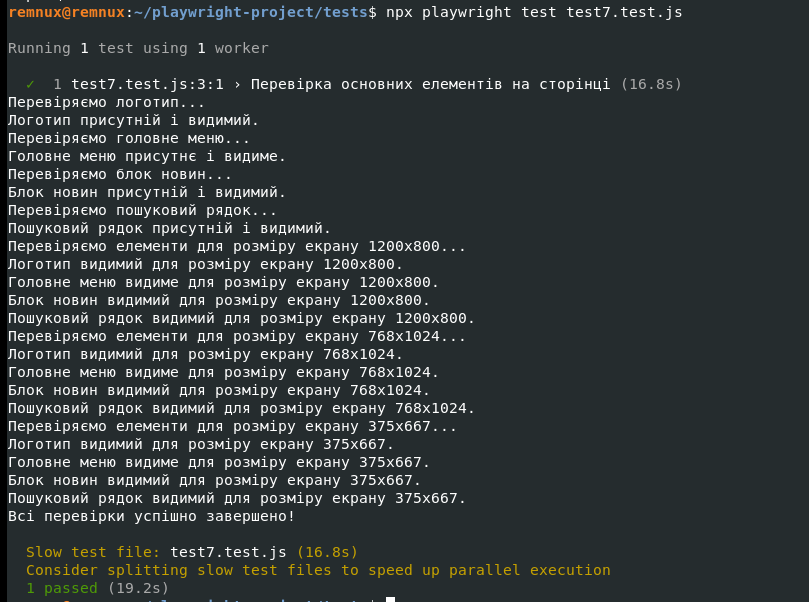
expect(await searchBox.isVisible()).toBe(true);

console.log(`Пошуковий рядок видимий для розміру екрану ${viewport.width}x${viewport.height}.`);

}

console.log('Всі перевірки успішно завершено!');

});

Результат виконання:  


**Завдання 8**

Код тесту:  
const { test, expect } = require('@playwright/test');

test('Перевірка доступності сторінки', async ({ page }) => {

const url = 'https://kpi.ua';

await page.goto(url);

// Отримуємо всі зображення та перевіряємо, чи є у них атрибут alt

const images = await page.locator('img');

const imageCount = await images.count();

for (let i = 0; i < imageCount; i++) {

const altText = await images.nth(i).getAttribute('alt');

expect(altText).not.toBeNull();

console.log(`Зображення ${i + 1} має alt: ${altText}`);

}

// Перевіряємо елементи з aria-label або role

const interactiveElements = await page.locator('[aria-label], [role]');

const elementCount = await interactiveElements.count();

for (let i = 0; i < elementCount; i++) {

const ariaLabel = await interactiveElements.nth(i).getAttribute('aria-label');

const role = await interactiveElements.nth(i).getAttribute('role');

console.log(

`Елемент ${i + 1}: aria-label="${ariaLabel}", role="${role || 'немає'}"`

);

}

// Перевірка заголовків на сторінці (h1, h2, h3)

const headings = await page.locator('h1, h2, h3, h4, h5, h6');

const headingCount = await headings.count();

expect(headingCount).toBeGreaterThan(0); // Переконуємось, що заголовки існують

console.log(`На сторінці знайдено ${headingCount} заголовків.`);

for (let i = 0; i < headingCount; i++) {

const headingTag = await headings.nth(i).evaluate((el) => el.tagName);

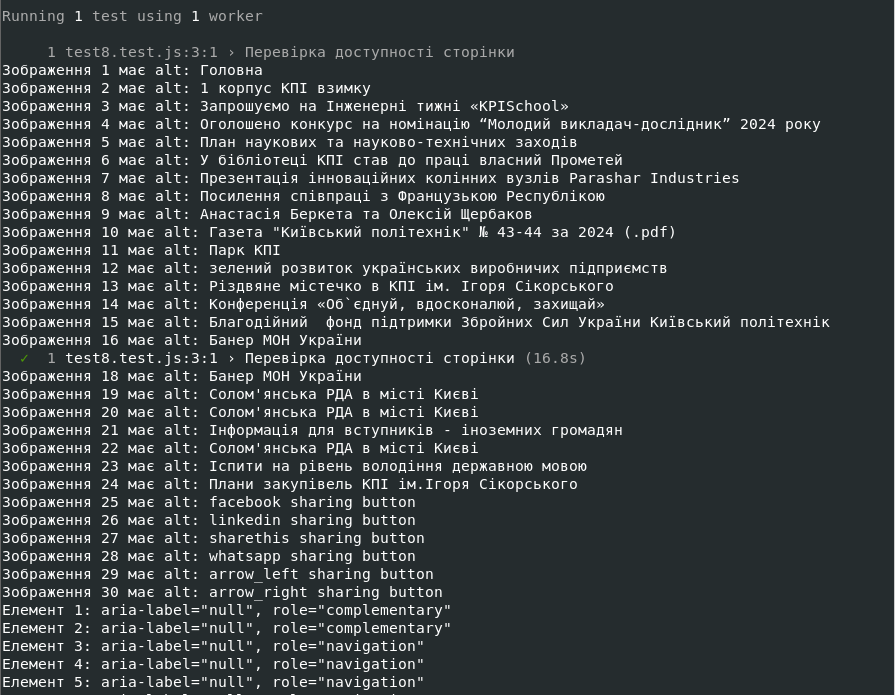
const headingText = await headings.nth(i).innerText();

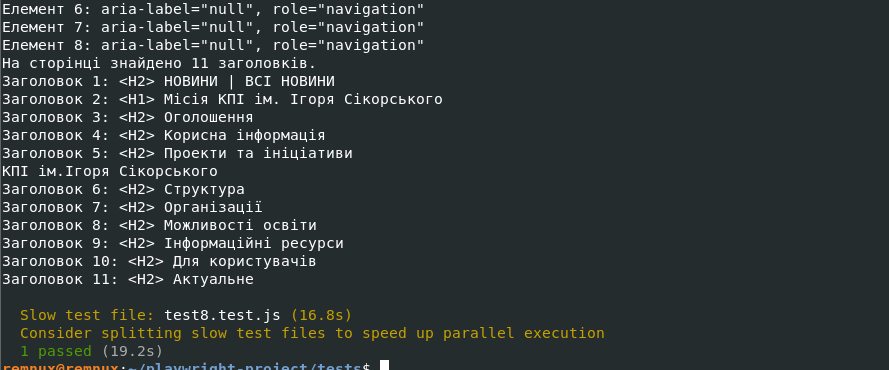
console.log(`Заголовок ${i + 1}: <${headingTag}> ${headingText}`);

}

});

Результат виконання:





**Завдання 9**

Код тесту:  
const { test, expect } = require('@playwright/test');

test('Перевірка зміни мови інтерфейсу сайту', async ({ page }) => {

// URL сайту

const siteURL = 'https://kpi.ua';

// Відкрити сайт

await page.goto(siteURL);

// Знайти прапор для англійської мови та натиснути

const englishFlag = page.locator('.enl > a:nth-child(1)');

await expect(englishFlag).toBeVisible();

await englishFlag.click();

// Зачекати, поки текст у меню зміниться

const menu = page.locator('#block-thex-mainnavigation');

await expect(menu).toContainText('University'); // Очікуйте, що текст зміниться

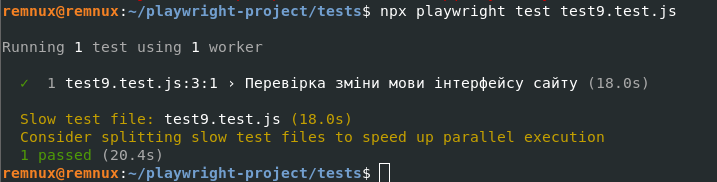
// Перевірити заголовок

const header = page.locator('#block-thex-views-block-announcements-block-1 > h2:nth-child(1)');

await expect(header).toContainText('Announcement');

});

Результат виконання:



**Завдання 10**

Код тесту:  
const { test, expect } = require('@playwright/test');

test('Перевірка завантаження зовнішніх ресурсів на kpi.ua', async ({ page }) => {

// URL сайту

const siteURL = 'https://kpi.ua';

// Список для зберігання результатів

const failedRequests = [];

const slowResources = [];

// Словник для зберігання часу початку запиту

const requestStartTimes = new Map();

// Список URL для ігнорування (включаючи YouTube)

const ignoredUrls = [

'https://www.youtube.com/s/player/03dbdfab/player\_ias.vflset/en\_US/base.js',

'https://www.youtube.com/youtubei/v1/log\_event?alt=json',

'https://www.youtube.com/youtubei/v1/log\_event' // Можна додати інші URL для ігнорування

];

// Прослуховувати події запитів

page.on('request', (request) => {

// Перевіряємо, чи є URL в списку для ігнорування

if (ignoredUrls.some(url => request.url().includes(url))) {

return; // Ігноруємо цей запит

}

// Інакше зберігаємо час початку запиту

requestStartTimes.set(request.url(), Date.now());

});

page.on('requestfinished', (request) => {

// Якщо цей запит був ігнорований, пропускаємо його

if (ignoredUrls.some(url => request.url().includes(url))) {

return;

}

const startTime = requestStartTimes.get(request.url());

if (startTime) {

const loadTime = Date.now() - startTime;

if (loadTime > 3000) {

slowResources.push({ url: request.url(), loadTime: `${loadTime} ms` });

}

}

});

page.on('response', (response) => {

// Якщо цей запит був ігнорований, пропускаємо його

if (ignoredUrls.some(url => response.url().includes(url))) {

return;

}

const status = response.status();

const url = response.url();

// Перевірка статусу відповіді

if (status >= 400) {

failedRequests.push({ url, status });

}

});

// Відкрити сторінку

await page.goto(siteURL);

// Зачекати завершення завантаження сторінки

await page.waitForLoadState('networkidle');

// Вивести результати

console.log('Помилки завантаження (404, 500 тощо):', failedRequests);

console.log('Ресурси із затримкою завантаження > 3 сек:', slowResources);

// Перевірка, що немає помилок 404, 500

expect(failedRequests).toHaveLength(0);

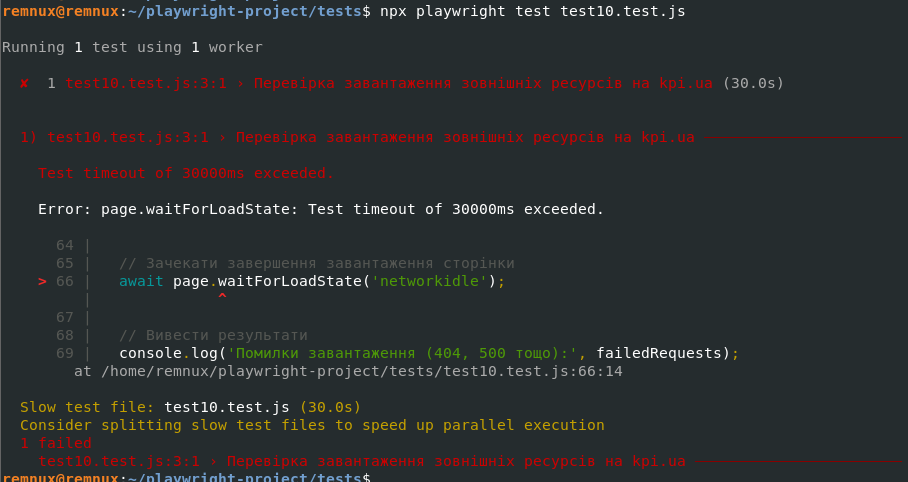
// Перевірка, що немає ресурсів із завантаженням більше 3 сек

// Якщо ви хочете додати більший поріг часу або допустити певну кількість повільних ресурсів:

expect(slowResources.length).toBeLessThanOrEqual(1); // Допустимо 1 повільний ресурс

});

Результат виконання:

  
Оскільки це кпішний сайт, то дуже очікувано, що ми не можемо пройти тест, там постійно будуть якісь непрацюючі ресурси)