Державний вищий навчальний заклад

Ужгородський національний університет

Факультет інформаційних технологій

Кафедра програмного забезпечення систем

**ЛАБОРАТОРНА РОБОТА № 8**

**Тема:** Компонент тести

Виконав студент:

ІІІ курсу спеціальності «Інженерія програмного забезпечення»

Нищий Богдан Вячеславович

**Ужгород-2024**

**Some news project**

Кінцевою метою лабораторної роботи є використовуючи cypress / jest / vitest написати юніт (компонент) тести до всіх компонентів в додатку з покриттям (codecoverage)

Рекомендованим є бібліотека cypress, яку застосуємо в наступній лабораторній роботі, і яка надає інструменти для написання компонент та енд-ту-енд (e2e) тестів

В кінці роботи студент повинен:

1. Створити пул-реквест

2. Оформити відповідного взірця [звіт](https://moodle.uzhnu.edu.ua/pluginfile.php/85717/mod_assign/intro/report_template.docx)

**Хід роботи**

Проєкт складається з декількох компонент, які використовуються для відображення хедера, футера, навігаційної панелі та виведення новин з news.api. Усі вони збережені у відповідній директорії components.

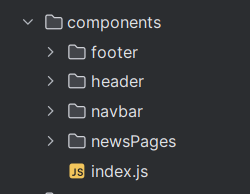


Рис. 1. Вміст директорії components.

Для того, щоб протестувати відповідні компоненти було створено чотири component теста з допомогою cypress.

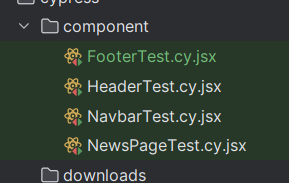


Рис. 2. Файли тестів компонент проєкту.

Кожен з них перевіряє окрему компоненту на наявність деяких елементів та функціонал, який вони надають.

1. Тестові випадки компонента Footer включають: перевірку на повний рендер всіх посилань, перевірка працездатності посилання на головну сторінку та перевірку на правильність електронної пошти, що вказана за посиланням «Report problem».
2. Тестові випадки компонента Header включають: перевірку на коректний рендер хедера, перевірка на наявність логотипа some news, перевірка на наявність навігаційної панелі (окремий компонент Navbar.jsx).
3. Тестові випадки компонента Navbar включають: перевірка на коректний рендер всіх посилань на сторінки та елемента для фото профілю користувача, перевірка на безпроблемну навігацію між сторінками з допомогою посилань на них, перевірка на зникнення фото профілю користувача в панелі при переході на сторінку /profile, перевірка на зміну стилів тексту посилань в залежності він розташування користувача, перевірка працездатності dropdown меню, перевірка можливості вийти з акаунта, натиснувши відповідну кнопку в dropdown меню.
4. Тестовий випадок компонента NewsPages включає перевірку наявності головної новини.

Відповідно, можемо виконати кожен набір тестів з допомогою cypress.



Рис. 3. Тестування компонента Footer.

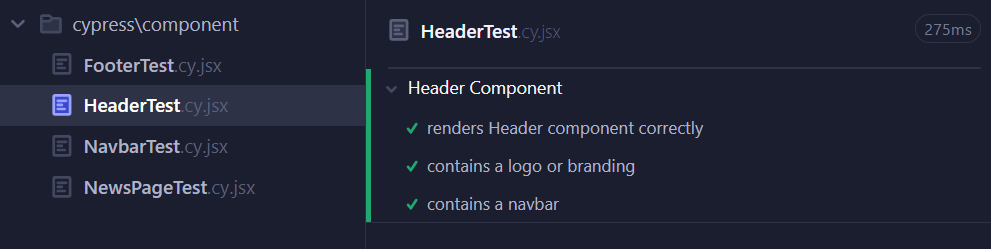


Рис. 4. Тестування компонента Header.

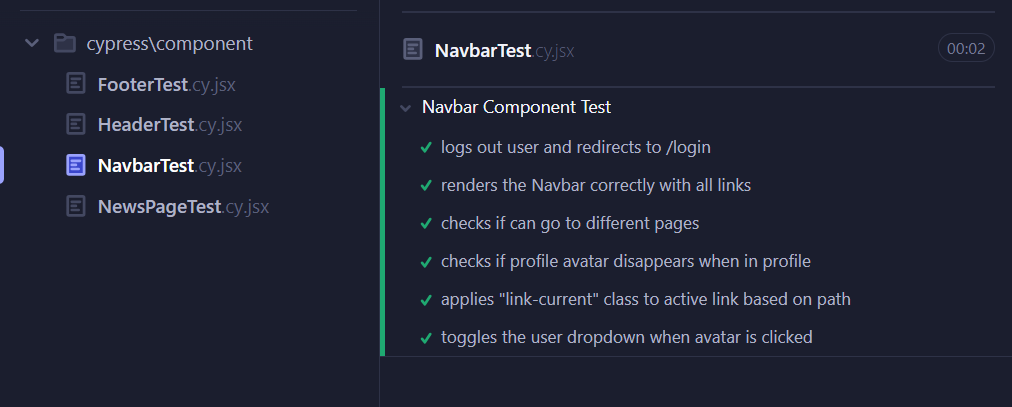


Рис. 5. Тестування компонента Navbar.

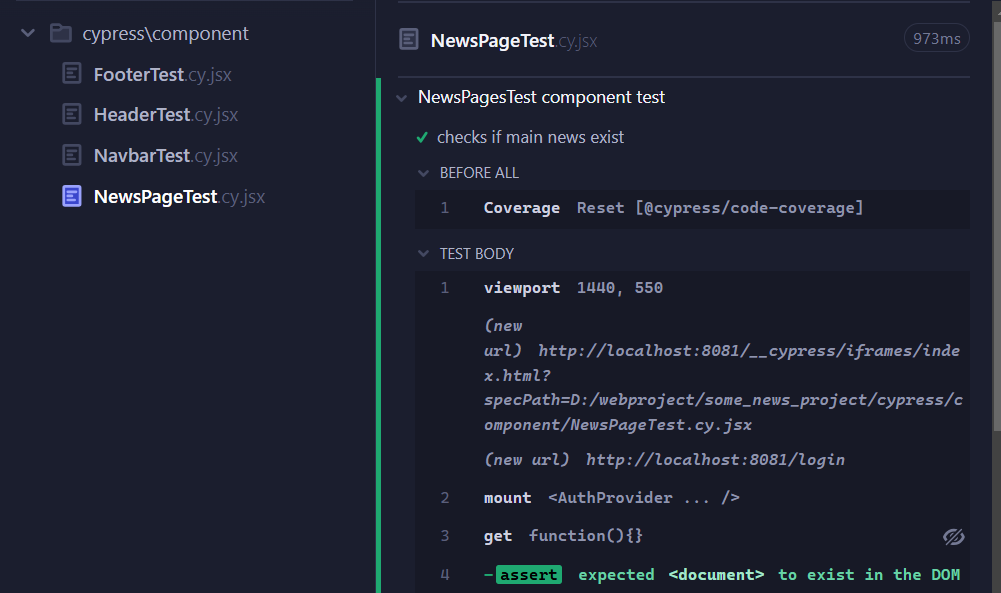


Рис. 6. Тестування компонента NewsPages.

Так як скриптів для тестування компонент багато, розглянемо деякі:

*describe*('Navbar Component Test', () => {  
 *it*('checks if can go to different pages', () => {  
 *cy*.mount(  
 <AuthProvider>  
 <NewsApiProvider>  
 <UserArticlesProvider>  
 <CommunityProvider>  
 <BrowserRouter>  
 <Navbar />  
 </BrowserRouter>  
 </CommunityProvider>  
 </UserArticlesProvider>  
 </NewsApiProvider>  
 </AuthProvider>,  
 );  
  
 *cy*.get('.link-home').should('contain.text', 'Home');  
 *cy*.get('.link-tech').should('contain.text', 'Technology');  
 *cy*.get('.link-sport').should('contain.text', 'Sport');  
 *cy*.get('.link-science').should('contain.text', 'Science');  
 *cy*.get('.link-reposts').should('contain.text', 'Community');  
 *cy*.get('.dropdown-menu-button').should('contain.html', 'img');  
 *cy*.get('.navbar-avatar').should('exist');  
  
 *cy*.get('.link-tech').click();  
 *cy*.url().should('include', '/tech');  
  
 *cy*.get('.link-sport').click();  
 *cy*.url().should('include', '/sport');  
  
 *cy*.get('.link-science').click();  
 *cy*.url().should('include', '/science');  
  
 *cy*.get('.link-reposts').click();  
 *cy*.url().should('include', '/community');  
  
 *cy*.get('.link-home').click();  
 *cy*.url().should('include', '8081/');  
 });  
});

На початку тесту cypress відкриває компоненту (тут Navbar) та перевіряє чи усі посилання відобразились на панелі (*cy*.get('.link-home').should('contain.text', 'Home');).

Далі, беручи кожне посилання та його css-клас, натискає на них з допомогою команди click() та перевіряє чи користувача доправлено на відповідну сторінку. Для цього тест бере маршрут до сторінки, на яку його переправили, та перевіряє чи він включає відповідний маршрут конкретної сторінки або категорії новин (до прикладу /sport).

Врешті, тест повертається на сторінку Home та перевіряє чи вірний маршрут задано.

*it*('checks if logo link works properly', () => {  
 *cy*.mount(  
 <AuthProvider>  
 <NewsApiProvider>  
 <UserArticlesProvider>  
 <CommunityProvider>  
 <BrowserRouter>  
 <Header />  
 <Footer />  
 </BrowserRouter>  
 </CommunityProvider>  
 </UserArticlesProvider>  
 </NewsApiProvider>  
 </AuthProvider>,  
 );  
 *cy*.get('.link-tech').should('contain.text', 'Technology');  
 *cy*.get('.link-tech').click();  
  
 *cy*.url().should('include', '/tech');  
  
 *cy*.get('.footer-logo-link').should('exist');  
 *cy*.get('.footer-logo-link').click();  
 *cy*.url().should('include', '/home');  
});

Наступний тест відноситься до компонента Footer та перевіряє, чи «логотип» з написом some news існує, і після натискання, чи перекинуло користувача на головну сторінку. Проте для того, щоб переконатись, що користувач вже не перебуває на головній сторінці коли відбувається відображення компонента, було підключено Header. З допомогою цього компонента було натиснуто на посилання link-tech, щоб перейти на сторінку категорії новин «технології».

Для того, щоб перевірити наскільки код директорії components протестований, довелось використати інструмент code coverage, що дає змогу отримати дані про те, наскільки добре протестовані окремі частини проєкту. У випадку з component тестами, покриття коду тестами буде перевірятись для директорії components.

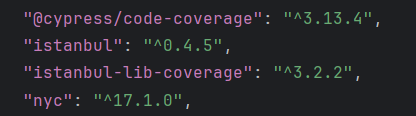


Рис. 7. Підключені залежності для code coverage.

Для того, щоб запустити скрипти для тестування з консолі, а також згенерувати звіт code coverage було додано декілька команд до scripts у package.json.

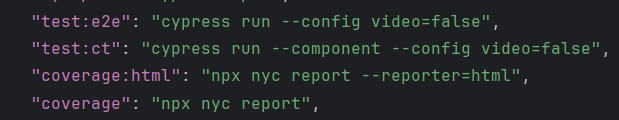


Рис. 8. Скрипти для керування тестами та code coverage.

Для того щоб зручно використовувати code coverage проєкту, використовується команда coverage:html, що дає змогу переглянути звіт у браузері.

З допомогою команди **npm run coverage:html** отримуємо звіт у директорії coverage у файлі index.html.

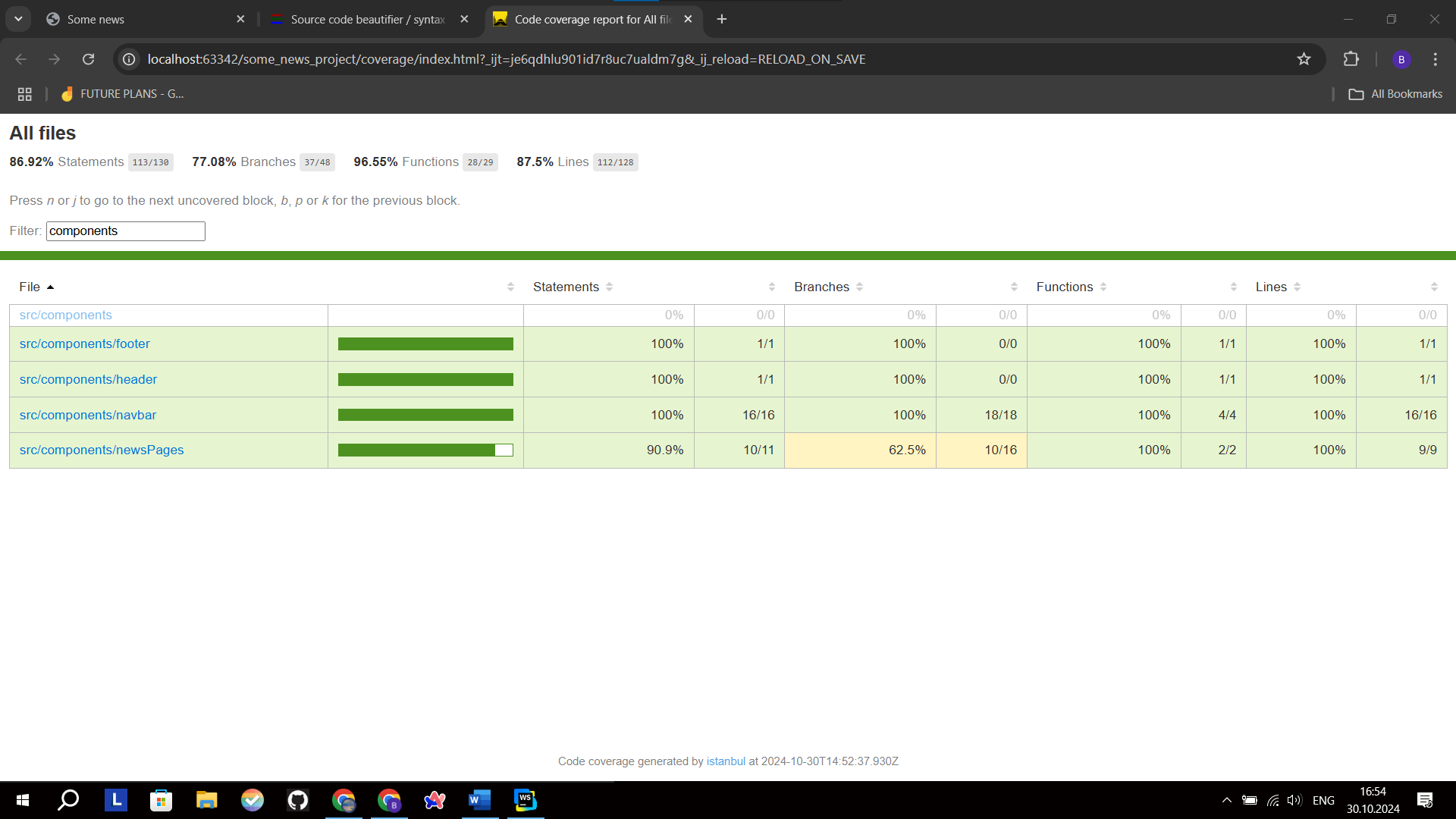


Рис. 9. Звіт з code coverage директорії components.

Як бачимо, загалом, директорія components майже повністю покрита тестами.

Висновок: Було написано декілька тестів для React компонент проєкту. Для написання тестів використовувався cypress. Кожен компонент отримав по одному файлу тестів, з різним набором тестових випадків. Для отримання звіту по тому, наскільки добре протестована директорія components було використано інструмент code coverage, який надає можливість прослідкувати за покриттям проєкту component та end-2-end тесами cypress. Для цього інструмент надає можливість переглядати звіт у браузері.