Державний вищий навчальний заклад

Ужгородський національний університет

Факультет інформаційних технологій

Кафедра програмного забезпечення систем

**ЛАБОРАТОРНА РОБОТА № 9**

**Тема:** E2E (енд-ту-енд) тестування

Виконав студент:

ІІІ курсу спеціальності «Інженерія програмного забезпечення»

Нищий Богдан Вячеславович

**Ужгород-2024**

**Some news project**

Реалізування e2e тестів за допомогою бібліотеки [cypress](https://docs.cypress.io/app/end-to-end-testing/writing-your-first-end-to-end-test). За наступним [посиланням](https://docs.cypress.io/app/end-to-end-testing/writing-your-first-end-to-end-test) можете знайти додаткову інформацію, як це реалізувати <https://docs.cypress.io/app/end-to-end-testing/writing-your-first-end-to-end-test>

В кінці роботи студент повинен:

1. Створити пул-реквест

2. Оформити відповідного взірця [звіт](https://moodle.uzhnu.edu.ua/pluginfile.php/85722/mod_assign/intro/report_template.docx)

**Хід роботи**

Проєкт складається з шістьох енд-ту-енд тестів, які охоплюють всі його існуючі на даний момент сторінки. Тести реалізовано з допомогою cypress.

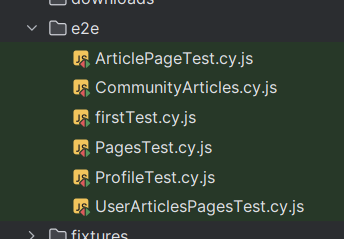


Рис. 1. Вміст директорії e2e для тестів cypress.

Як і для минулої лабораторної, було використано інструмент code coverage для відображення покриття коду проєкта тестами.

Далі наведена інформація про кожен файл e2e тестів:

1. firstTest.cy.js: тест використовується для перевірки відображення сторінок авторизації та реєстрації, а також для перевірки, чи може неавторизований користувач увійти до захищених сторінок проєкту. Тут також було перевірено виведення помилок за їх наявності;
2. PagesTest.cy.js: створений для перевірки на наявність хоча б однієї новини у відповідних категоріях (home, sport, science, technology);
3. ArticlePageTest.cy.js: призначений для перевірки коректного відображення повної статті, що отримується шляхом запиту get до news.api. Тут перевіряється кожна складова сторінки Article.jsx, тобто наявність у статті заголовку, головного зображення, контенту та робочого посилання на повну статтю. Також тут перевіряється ситуація, коли статті з деяким індексом неіснує, отже виводиться відповідне повідомлення.
4. ProfileTest.cy.js: призначений для перевірки нормального відображення та функціоналу сторінки профілю користувача. Тут перевіряється можливість зміни біографії, паролю а також видалення акаунта. Також перевіряються помилкові ситуації під час видалення акаунта чи зміни паролю.
5. UserArticlesPagesTest.cy.js: призначений для перевірки відображення та функціональності створення та збереження статей користувача. Тест охоплює перевірку двох сторінок (UserArticles та NewArticle), які призначені для перевірки відображення статей авторизованого користувача та створення нових відповідно. Також перевіряються помилкові ситуації при створенні статей.
6. CommunityArticles.cy.js: призначений для перевірки відображення списку статей, створених користувачами, а також коректної роботи сторінки UserFullArticle.jsx, тобто перевірка відображення повної статті, а також перевірка функціоналу репостів.

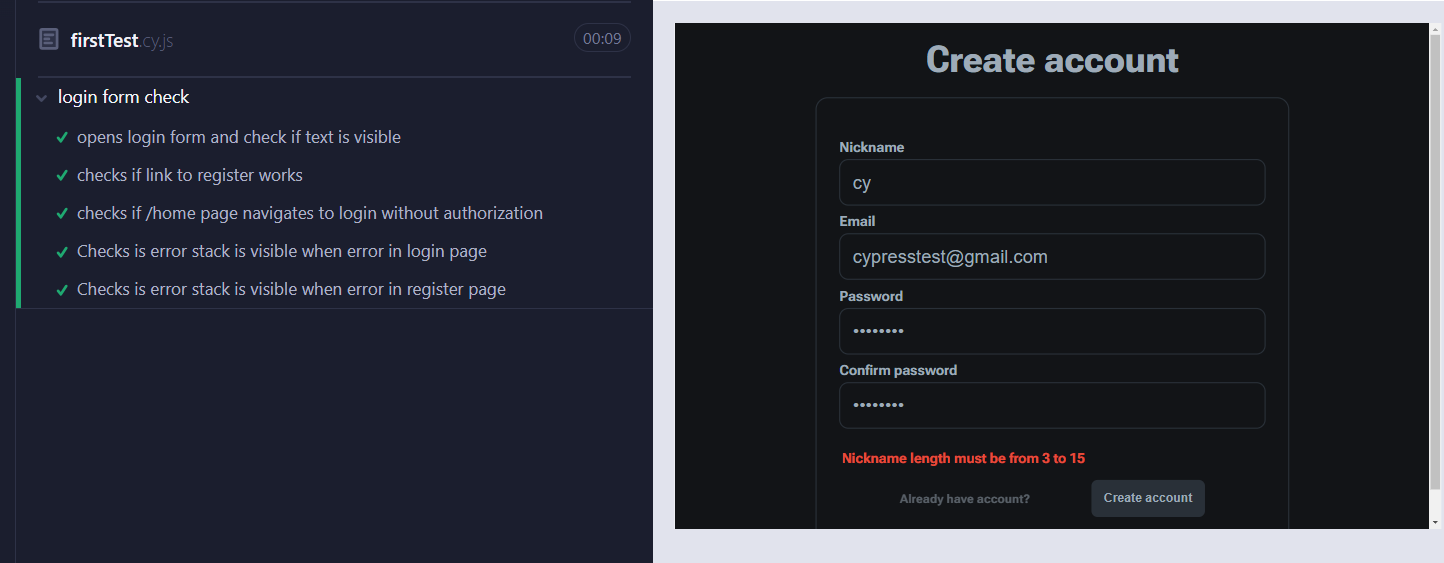


Рис. 2. Виконання тестів з файлу firstTest.cy.js.

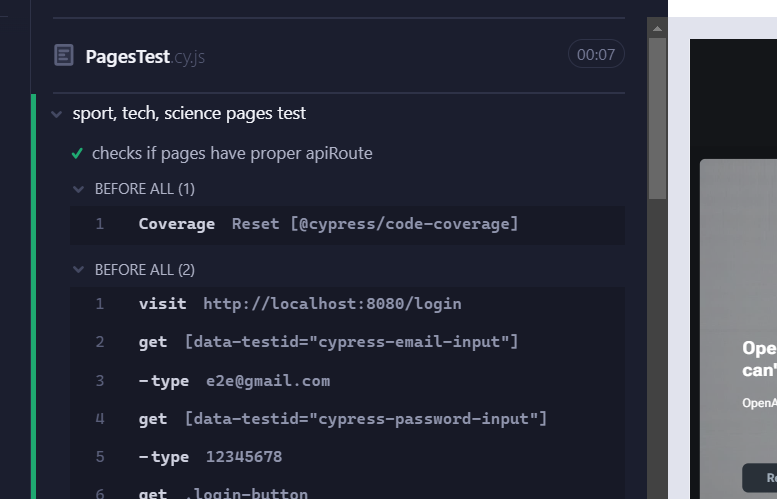


Рис. 3. Результат тесту PagesTest.cy.js.



Рис. 4. Результат тесту ArticlePageTest.cy.js.

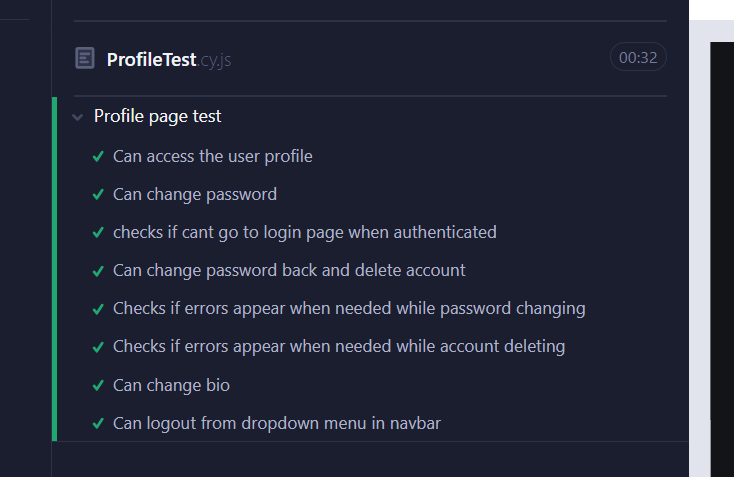


Рис. 5. Результат тесту ProfileTest.cy.js.

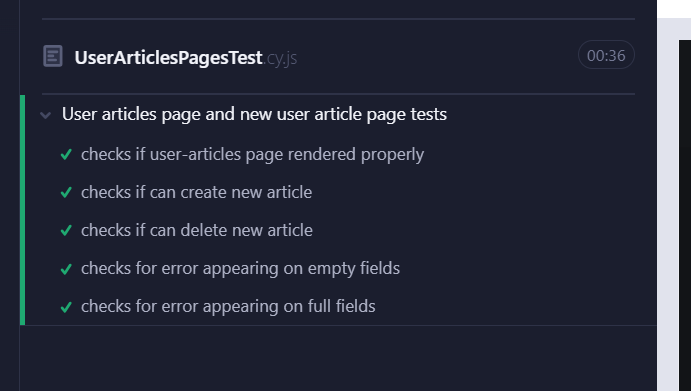


Рис. 6. Результат тесту UserArticlesPagesTest.cy.js.

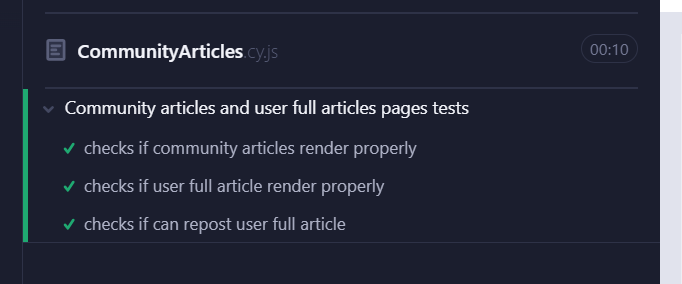


Рис. 7. Результат тесту CommunityArticles.cy.js.

Розглянемо декілька скриптів end-2-end тестів.

it('checks if can create new article', () => {

cy.url().should('include', '/user-articles');

cy.get('.link-to-new-article').should('be.visible');

cy.get('.link-to-new-article').should('have.text', 'New');

cy.get('.link-to-new-article').click();

cy.url().should('include', '/new-article');

cy.get('.new-article-error-stack').should('not.be.visible');

cy.get('.new-article-title-input').should('be.visible');

cy.get('.new-article-title-input').type('Creating new article **in** test');

cy.get('.article-desc-input-field').should('be.visible');

cy.get('.article-desc-input-field').type('Creating new article **in** test description');

cy.get('.new-article-content-field .ql-editor').click().type('This is some sample text typed **in** the editor.');

cy.get('.new-article-content-field .ql-editor').should(

'contain.text',

'This is some sample text typed **in** the editor.',

);

cy.get('.create-article-button').should('be.visible');

cy.get('.create-article-button').click();

cy.url().should('include', '/user-articles');

cy.get('.user-articles-page-title-h1').should('have.text', 'Articles written by yourself');

cy.get('.link-to-new-article').should('be.visible');

cy.get('.link-to-new-article').should('have.text', 'New');

cy.get('.user-articles-list').should('have.length.greaterThan', **0**);

cy.get('.article-item').eq(**0**).find('.article-item-container').should('be.visible');

cy.get('.article-item').eq(**0**).find('.article-title').should('have.text', 'Creating new article **in** test');

cy.get('.article-item')

.eq(**0**)

.find('.article-desc')

.should('have.text', 'Creating new article **in** test description');

cy.get('.article-item').eq(**0**).find('.article-buttons-container').should('be.visible');

cy.get('.article-item').eq(**0**).find('.article-item-img').should('be.visible');

});

Тест взятий з файлу UserArticlesPagesTest та представляє собою

перевірку коректності створення користувацької статті.

Спершу перевіряється поточна сторінка (/user-articles). Далі тест перевіряє чи видно кнопку-посилання «New», що веде на сторінку створення нової статті (/new-article). Після переходу туди, перевіряється відсутність помилок та наявність необхідних полів для створення статті (Заголовок, опис, основний контент). Відповідні дані вводяться у поля з допомогою команди type().

Натискається кнопка «Create article» і очікується, що посилання у вкладці зміниться на сторінку /user-articles.

Так як найвище в списку статей розміщуються ті, які були опубліковані пізніше, то щойно створена стаття повинна мати індекс 0. Далі тест перевіряє чи відповідні елементи мають текст, що вносився в поля при створенні статті.

Наступний тестовий приклад взято з файлу ProfileTest. Він реалізує перевірку на коректність зміни біографії користувача.

it('Can change bio', () => {

cy.visit('http://localhost:**8080**/login');

cy.get('[data-testid="cypress-email-input"]').type('e2e@gmail.com');

cy.get('[data-testid="cypress-password-input"]').type('**12345678**');

cy.get('.login-button').click();

cy.url().should('include', '/home');

cy.get('.dropdown-menu-button').should('be.visible');

cy.get('.dropdown-menu-button').click();

cy.get('.dropdown-menu').should('be.visible');

cy.get('.profile-link').should('exist');

cy.get('.profile-link').click();

cy.url().should('include', '/profile');

cy.get('.profile-bio').should('exist');

cy.get('.bio-input').should('not.exist');

cy.get('.bio-edit-button').should('be.visible');

cy.get('.bio-edit-button').click();

cy.get('.bio-input').should('be.visible');

cy.get('.bio-input-field').type('Testing bio input with cypress');

cy.get('.save-bio-button').should('be.visible');

cy.get('.save-bio-button').click();

cy.visit('http://localhost:**8080**/profile');

cy.url().should('include', '/profile');

cy.get('.bio-input').should('not.exist');

cy.get('.bio-input-text').should('be.visible');

cy.get('.bio-input-text').should('have.text', 'Testing bio input with cypress');

});

Тест розпочинається на сторінці авторизації, де користувач вводить необхідні дані. Після цього на головній сторінці, з допомогою навігаційної панелі та dropdown меню відбувається перехід на сторінку профіля.

На сторінці профілю перевіряється існування блоку з біографією (.profile-bio), яка включає поле для вводу, підпис з поточним текстом біографії (тут порожній) та кнопку збереження біографії. Також перевіряється чи немає на даному етапі поля для вводу нового тексту (.bio-input). Така перевірка робиться з допомогою команди should та атрибутів exist та not.exist. Для перевірки видимості елемента використовується та сама команда та атрибут be.visible або not.be.visible.

Для зміни стану .bio-input натискається кнопка .bio-edit-button, яка фактично присвоює полю для введення стан видимості. Далі тест вводить текст для оновлення біографії та натискає .save-bio-button. Під кінець перевіряється чи дійсно поле для вводу тексту не видно і чи текст біографії дорівнює «Testing bio input with cypress».

Наступний тестовий приклад перевіряє функціонал репостів для користувацьких статей. Суть цієї можливості така: користувач, якому сподобалась стаття, може натиснути кнопку «Repost». Кількість репостів статті збільшиться, а у списку репостів користувача з’явиться відповідна стаття. Протилежним чином користувач може забрати свій репост.

it('checks if can repost user full article', () => {

cy.url().should('include', '/community');

cy.get('.news-list').should('be.visible');

cy.get('.main-news-item').should('be.visible');

cy.get('.news-item').should('have.length.greaterThan', **0**);

cy.get('.news-item').eq(**2**).find('.item-button').should('be.visible');

cy.get('.news-item').eq(**2**).find('.item-button').click();

cy.url().should('include', '/user-full-article');

cy.get('.article-title').should('be.visible');

cy.get('.article-title').should('have.length.greaterThan', **0**);

cy.get('.article-main-text').should('be.visible');

cy.get('.article-main-text').should('have.length.greaterThan', **0**);

cy.get('.article-author').should('have.length.greaterThan', **0**);

cy.get('.article-repost-button')

.should('be.visible')

.and('not.have.class', 'article-button-active')

.and('contain', 'Repost');

cy.get('.article-repost-button').click();

cy.url().should('include', '/user-full-article');

cy.get('.article-repost-button').should('have.class', 'article-button-active').and('contain', 'Reposted');

cy.get('.article-repost-button').should('be.visible');

cy.get('.article-repost-button').click();

cy.get('.article-repost-button')

.should('be.visible')

.and('not.have.class', 'article-button-active')

.and('contain', 'Repost');

});

Спершу тест переходить на сторінку /community, де перевіряє чи виведено список статтей і чи довжина масиву цих статей більша 0 (з допомогою команди should та атрибутів have.length.greaterThan та 0). У цьому списку знаходиться стаття, яку обирає тест і переходить на сторінку повного видання цієї статті. Тут перевіряється чи виводиться заголовок, контент та нікнейм автора.

Після цього натискається кнопка «Repost» і перевіряється чи змінився її клас на активний (змінюється після збереження даних про новий репост статті). Текст кнопки, відповідно, повинен змінитись на «Reposted».

В кінці тесту виконується та сама дія, яка проте, має інше значення: репост забирається та кнопка приходить у свій стан за замовчуванням.

Інші тестові випадки та вміст файлів end-2-end тестів можна переглянути на [github репозиторії проєкту](https://github.com/nyshchyibohdan/SomeNewsProject).

Після створення та успішного запуску тестів необхідно перевірити покриття коду тестами.

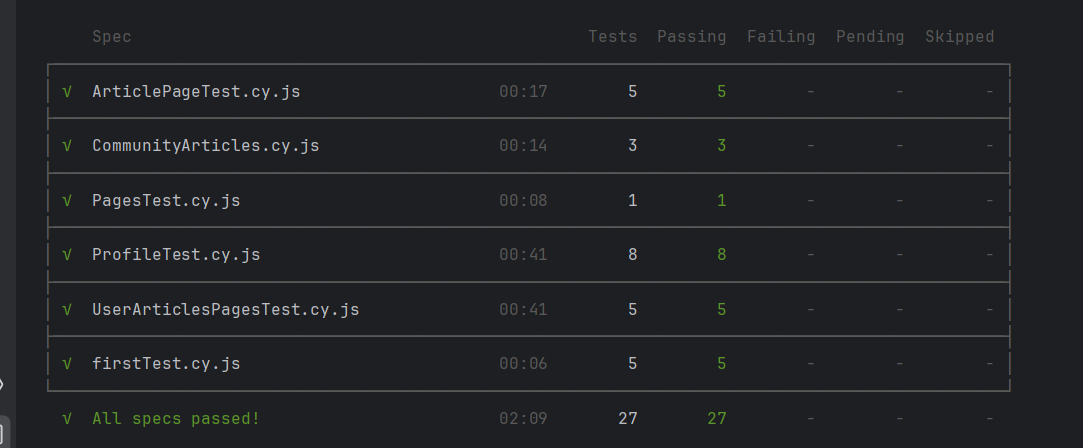


Рис. 8. Результат запуску усіх end-2-end тестів.

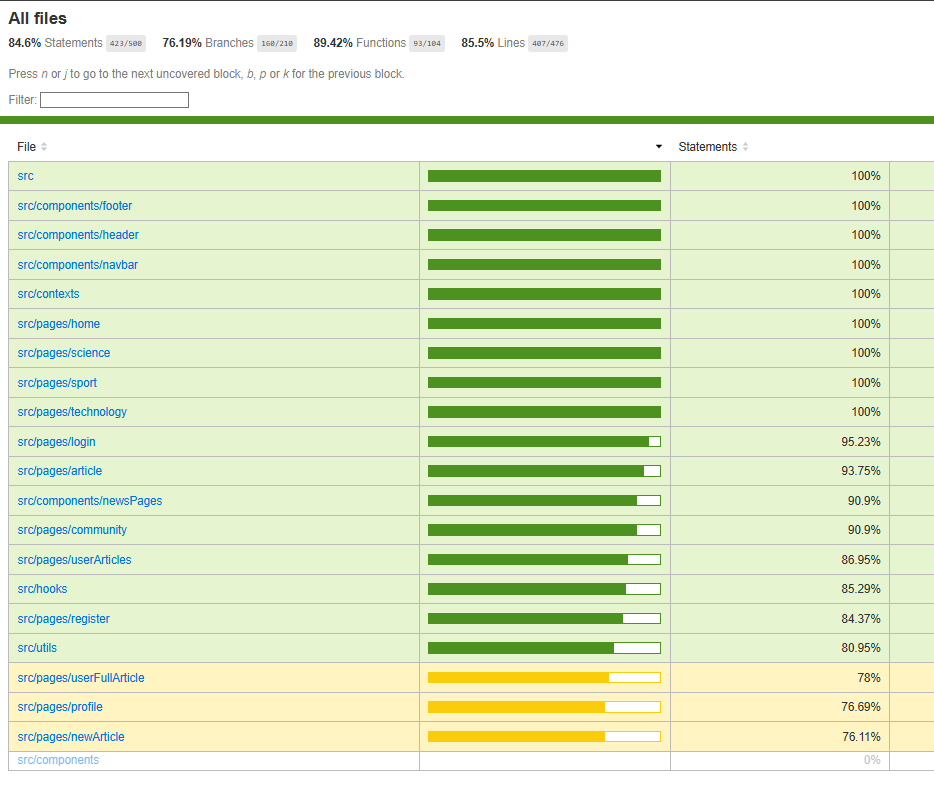


Рис. 9. Звіт code coverage.

Висновок: Були написані end-2-end тести для проєкту some news. Для написання тестів використовувався cypress. Кожен файл тестів призначався для конкретної сторінки або групи сторінок, що є дотичними одна до одної. Було описано деякі тестові ситуації, показано результат виконання тестів у додатку cypress, та додано code coverage після запуску усіх end-2-end тестів, а також переглянуто цей звіт у браузері.