МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«ПРОЄКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ВЕБ ЗАСТОСУНКІВ»**

Виконала: студентка 2 курсу, групи 8.1211-іпз
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
К.В. Каретнікова
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Кривохата А.Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.пед.н. Пшенична О.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2022 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Каретніковій Катерині Вадимівні
(прізвище, ім’я та по-батькові)

1. Тема роботи Проєктування та розробка системи автоматизованого тестування веб застосунків

Керівник роботи Кривохата Анастасія Григорівна, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

Затверджені наказом ЗНУ від « 4 » травня 2022 р. № 500-с

2. Строк подання студентом роботи

3. Вихідні дані до роботи 1. Постановка задачі
2. Перелік літератури

4. Зміст роботи(перелік питань, які потрібно розробити)
1. Постановка задачі
2. Розробка тестових випадків
3. Проєктування системи автоматизованого тестування веб застосунку
4. Розробка системи автоматизованого тестування веб застосунку

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
Презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	15.05.2022	
2.	Обробка методичних та теоретичних	22.05.2022	
	джерел.		
3.	Ознайомлення з веб застосунком.	05.06.2022	
4.	Огляд і вибір засобів реалізації.	01.07.2022	
5.	Розробка першого розділу.	23.07.2022	
6.	Написання тестових випадків.	10.08.2022	
7.	Проєктування системи.	25.08.2022	
8.	Розробка другого розділу.	06.09.2022	
9	Написання коду системи автоматизованого	21.09.2022	
	тестування.		
10.	Розробка третього розділу.	17.10.2022	
11.	Оформлення і нормоконтроль.	25.11.2022	
12.	Захист кваліфікаційної роботи.	15.12.2022	

Студент _____ К.В. Каретнікова
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.Г. Кривохата
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Проєктування та розробка системи автоматизованого тестування веб застосунків»: 63 с., 24 рис., 19 табл., 8 джерел, 1 додаток.

АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ, ВЕБ ЗАСТОСУНОК, ЗВІТ, ІНФОРМАЦІЙНА СИСТЕМА, ПАТЕРН, САЙПРЕС, ТЕСТОВИЙ ВИПАДОК.

Об’єкт дослідження – процес тестування веб застосунків.

Предмет дослідження – система автоматизованого тестування веб застосунків.

Мета роботи: розробка системи автоматизованого тестування для веб застосунку, що призначений вантажовідправників, що шукають перевізників.

Методи дослідження: аналіз, вивчення та узагальнення, порівняння.

У кваліфікаційній роботі магістра наведено огляд фреймворків для автоматизації тестування; вивчено вимоги до веб застосунку для вантажовідправників; обрано оптимальний засіб реалізації; спроєктовано та розроблено систему автоматизованого тестування веб застосунку.

SUMMARY

Master`s Qualifying Paper «Design and Developing the Automated Testing System for Web Applications»: 63 pages, 24 figures, 19 tables, 8 references, 1 supplement.

AUTOMATED TESTING, WEB APPLICATION, REPORT, INFORMATION SYSTEM, PATTERN, CYPRESS, TEST CASE.

The object of the study is the process of testing web applications.

The subject of the study is the automated testing system for web application.

The aim of the study is development of the automated testing system for the web application which is designed for shippers who are looking for carriers.

The methods of research are analysis, study and generalization, comparison.

In the Master`s Qualification Thesis an overview of frameworks for automated testing was provided, specifications for the web application for shippers were explored, the optimal mean of implementation has been chosen, the automated testing system for web application was designed and developed.

ЗМІСТ

ВСТУП

В наші дні інформаційні технології стали невід'ємною частиною життя людей. Комп'ютери та програмне забезпечення покликано зменшити участь людини у деяких процесах повсякдення, а також пришвидшити їх виконання. Вони використовуються майже в усіх сферах, починаючи з банківської сфери і лікарень та закінчуючи сферою розваг.

Дедалі частіше трапляються випадки, коли представники, наприклад, бізнесу, науки або якоїсь державної установи хочуть автоматизувати та зменшити вплив людини на один або декілька процесів, тож кожного дня у всьому світі випускається велика кількість програмного забезпечення.

Часто від того, наскільки якісно написано програмне забезпечення залежить імідж або фінансова сторона замовника та, навіть, життя людей, отже дуже важливо щоб система, яку випускають на ринок, працювала коректно, без помилок, була безпечною та відповідала всім вимогам, які висунув замовник.

Виявлення дефектів розробки, а також оцінка якості програмного продукту є завданнями тестування програмного забезпечення. Тестування дає змогу адекватно оцінити стан продукту на всіх етапах розробки, а також, за потреби, випустити на ринок попередню версію.

Так як на розробку програмного забезпечення витрачається велика кількість коштів, а для того, щоб випустити якісний продукт IT-компанії готові виділити багато ресурсів та приблизно третину, а то і половину часу на тестування, то перед компаніями, які займаються розробкою постає питання як зменшити вартість тестування.

Автоматизоване тестування дозволяє зекономити ресурси та час, які витрачаються на тестування, а також регулярно перевіряти окремі частини системи на наявність помилок. Саме тому багато компаній, які займаються

розробкою програмного забезпечення, приділяють увагу написанню автотестів для своїх продуктів.

Кваліфікаційну роботу присвячено проєктуванню та розробці системи автоматизованого тестування для веб застосунку для вантажовідправників, що шукають перевізників.

Метою дослідження є розробка системи автоматизованого тестування для веб застосунку, що призначений вантажовідправників, що шукають перевізників.

Об’єктом дослідження є процес тестування веб застосунків.

Предметом дослідження є система автоматизованого тестування веб застосунків.

У відповідності з метою дослідження ставляться такі завдання:

– дослідити предметну область та ознайомитись з функціоналом та вимогами до обраного веб застосунку;

– розглянути найбільш популярні засоби автоматизації веб застосунків;

– розробити список тестових випадків для системи автоматизованого тестування веб застосунків;

– виконати проєктування системи автоматизованого тестування згідно з обраним патерном Page Object Model;

– розробити систему автоматизованого тестування веб застосунку.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Актуальність проблеми

Під час розробки програмного забезпечення розробники можуть допускати певні помилки або не врахувати певні вимоги чи неординарні випадки. Особливо це відчувається під час створення масштабного проекту, коли у розробці продукту може приймати участь декілька десятків, а може й навіть сотень людей.

Одним із найважливіших завдань для компанії, яка виконує розробку програмне забезпечення є випустити на ринок якісний продукт, який працював би коректно та відповідав усім вимогам замовника. Саме з цієї причини під час розробки програмного забезпечення особливу увагу приділяють тестуванню.

Головними завданнями процесу тестування є:

– вивчення вимог замовника та перевірка продукту на відповідність цим вимогам;

– перевірка усіх можливих сценаріїв поведінки користувача при роботі з програмним продуктом;

– перевірка зручності користування системою.

Під час розробки програмного забезпечення процес тестування не обмежується тільки перевіркою новостворених модулів. Потрібно ще й враховувати те, як виправлення помилок або нові модулі вплинули на вже існуючі: чи не виникли якісь конфлікти або нові дефекти у вже протестованих ділянках коду. Їх потрібно регулярно перевіряти, наприклад, щоразу коли буде випускатись нова версія програми.

В таких випадках на допомогу приходить автотестування, яке допомагає зекономити ресурси, час, а, отже, і бюджет. Тестувальник може вручну

перевіряти нові ділянки коду та шукати нові дефекти у той час, коли за допомогою автотестів буде перевірятись вже працюючий код.

Звичайно, є і певні складнощі, які виникають під час написання автотестів. Наприклад, не всі фреймворки підтримують тестування мобільних систем, а отже автоматизація тестування мобільних пристроїв стає складнішою. Також при автоматизованому тестуванні неможливо перевірити зручність використання та сприйняття вебдодатку користувачем в цілому.

1.2 Рівні тестування

Рівні тестування, які можна автоматизувати, демонструє піраміда автоматизації тестування (див. рис. 1.1) [1]. Якщо при автоматизації тестів дотримуватись правил цієї піраміди, то надійність тестів значно поліпшиться, а час, який витрачається на визначення того, чи порушує внесена зміна код можна значно скоротити. По суті ця піраміда визначає типи тестів, які повинні бути включені в автоматизоване тестування, а також їх частоту та послідовність [2]. За цією пірамідою існує три рівні тестування, які можливо автоматизувати: юніт тестування, інтеграційне тестування та E2E тестування.

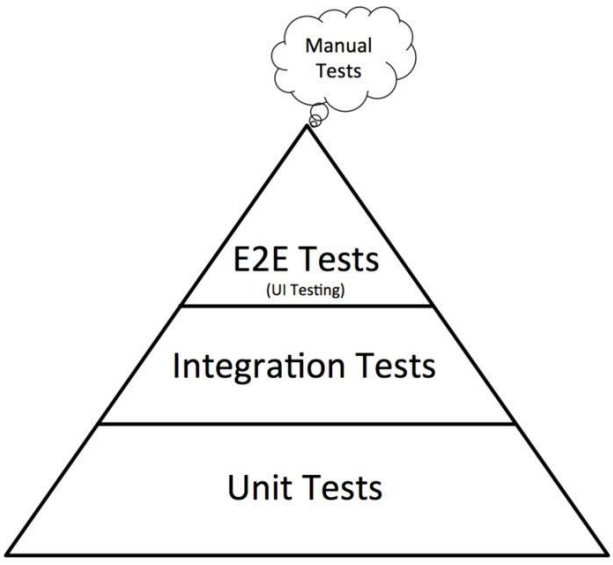

Рисунок 1.1 – Піраміда автоматизації тестування

На першому рівні знаходиться **юніт тестування**, яке зазвичай виконують розробники. Цей тип тестування призначений для перевірки окремих компонентів або методів в ізольованих умовах для того, щоб перевірити, що вони працюють належним чином. Під час написання юніт тестів потрібно враховувати як позитивні, так і негативні випадки, наприклад, як обробка помилок. Такі тести рекомендовано запускати щоразу, коли додається нова функція. Отже, при наявності юніт тестів розробники можуть швидко оцінити ситуацію та отримати відгук про те, чи правильно працюють методи як вони є.

На другому рівні знаходиться **інтеграційне тестування**. Це означає, що ці тести потрібно запускати не так часто як юніт тести. Даний тип тестування перевіряє взаємодію коду із зовнішніми компонентами, тобто фактично взаємодію коду з іншим кодом. Ці компоненти можуть варіюватися від баз даних до зовнішніх служб (API), адже програмне забезпечення має ефективно спілкуватися та отримувати правильну інформацію незалежно від того з яким компонентом взаємодіє.

На третьому рівні знаходиться **E2E тестування**, яке використовується для того, аби зрозуміти чи працює програма коректно від початку до кінця. Під час написання таких тестів важливо уявити точку зору користувача та спробувати відтворити взаємодію користувача з системою. Важливо урахувати всі сценарії, як позитивні, так і негативні, взаємодії користувача з системою та переконатись, що в усіх випадках система буде адекватно реагувати на дії користувача. Також, подібно до інтеграційних тестів, ці тести можуть вимагати взаємодії додатка із зовнішніми залежностями.

У даній кваліфікаційні роботі будуть виконані E2E тести, адже саме цей тип тестування найчастіше використовується при виконанні автоматизованого тестування проектів.

Для деяких тестових випадків також перевірятиметься і правильність відповіді від сервера на деякі запити, отже можна сказати, що частково буде виконано й API тестування.

1.3 Вибір засобу автоматизованого тестування

Існує велика кількість бібліотек та фреймворків, які призначені для автоматизації тестування. Більшість з них імітує натискання на кнопки, ввод даних у поля тощо, тобто взаємодію реального користувача з системою. Розглянемо найбільш відомі та ефективні засоби автоматизації тестування.

Одним з найпоширеніх засобів тестування є **Selenium**. Зручний тим, що користувачі можут писати скрипти на різних мовах програмування. Основною перевагою Selenuim є його гнучкість, а недоліком те, що користувачі повинні мати гарні навички програмування та використовувати багато часу на створення різних бібліотек, потрібних для автоматизації.

Cypress – відносно новий фреймворк, який може тестувати все, що працює у браузері. Тести написані на JavaScript. Має вихідний код, активну спільноту та детальну документацію.

Puppeteer – бібліотека Node.js з відкритим кодом, розроблена компанією Google, що дозволяє автоматизувати дії в Chromium браузері за допомогою API найвищого рівня.

Playwright – це Node.js бібліотека, розроблена компанією Microsoft, яка призначена для автоматизації тестування з єдиним API для різних браузерів (Chromium, Firefox and WebKit).

Для автоматизації тестування веб застосунку було обрано фреймворк Cypress [3]. Він має декілька переваг над іншими фреймворками.

Великою перевагою є те, що Cypress не потребує ніякого попереднього налаштування проекту, на відміну від Selenium, налаштування якого є складним процесом. Налаштування Selenium вимагає завантаження драйверів для кожного типу тестового браузеру, а також налаштування тестового середовища.

На відміну віл Puppeteer, який підтримує тестування тільки у браузерах на базі Chromium, Cypress підтримує декілька видів браузерів.

Також серед переваг можна виділити те, що в тестові сценарії не потрібно додавати явні чи неявні команди очікування, на відміну від Selenium. Наприклад якщо Selenium отримає команду знайти на сторінці певний елемент, то зразу ж буде його шукати. Це може бути проблемою коли, наприклад, сторінка ще не встигла завантажитись, тому під час написання тестів на Selenium у тестах часто потрібно використовувати команди очікування. Під час написання тестів на Cypress цього робити не потрібно, адже випадку, коли він не знаходить певний елемент, то продовжить його шукати певний час, який можна задати самостійно у файлі cypress.json.

Ще однією перевагою Cypress є те, що, на відміну від Selenium, під час виконання тестів у тестовому браузері відображається панель Test Runner, яка показує що відбувається на кожному кроці тесту.

Серед інших переваг можна також виділити те, що програма робить скріншоти під час проведення тесту, а також вичерпну документацію, яку дуже легко вивчати.

Серед недоліків можна виділити певні проблеми з роботою з iFrame та відсутність підтримки декількох вкладок у тестовому браузері.

Отже, Cypress було обрано так як він легкий у використанні, дозволяє подивитись що відбувалось на кожному етапі тестування, підтримує декілька браузерів, а також робить скріншоти під час проведення тесту.

1.4 Опис вебдодатку Shipper TMS

Для реалізації мети кваліфікаційної роботи було обрано вебдодаток, який полегшить облік та керування вантажами, які потрібно доправити до пункту призначення. Система орієнтована на представників бізнесу. Користувач системи може заповнити інформацію про вантаж, який йому потрібно відправити, вказати перевізника, подивитись список уже

сформованих вантажів, а також додати у систему список перевізників, з якими він співпрацює.

Систему можна розділити на декілька основних модулів:

1) авторизація;
2) налаштування аканту;
3) створення нового відправлення;
4) можливість додати інформацію про перевізника.

Розглянемо нижче кожен з модулів детальніше.

Авторизація. Користувач може зареєструвати у системі свою компанію, а також авторизуватись у системі. Приклад сторінки авторизації наведено на рисунку 1.2.

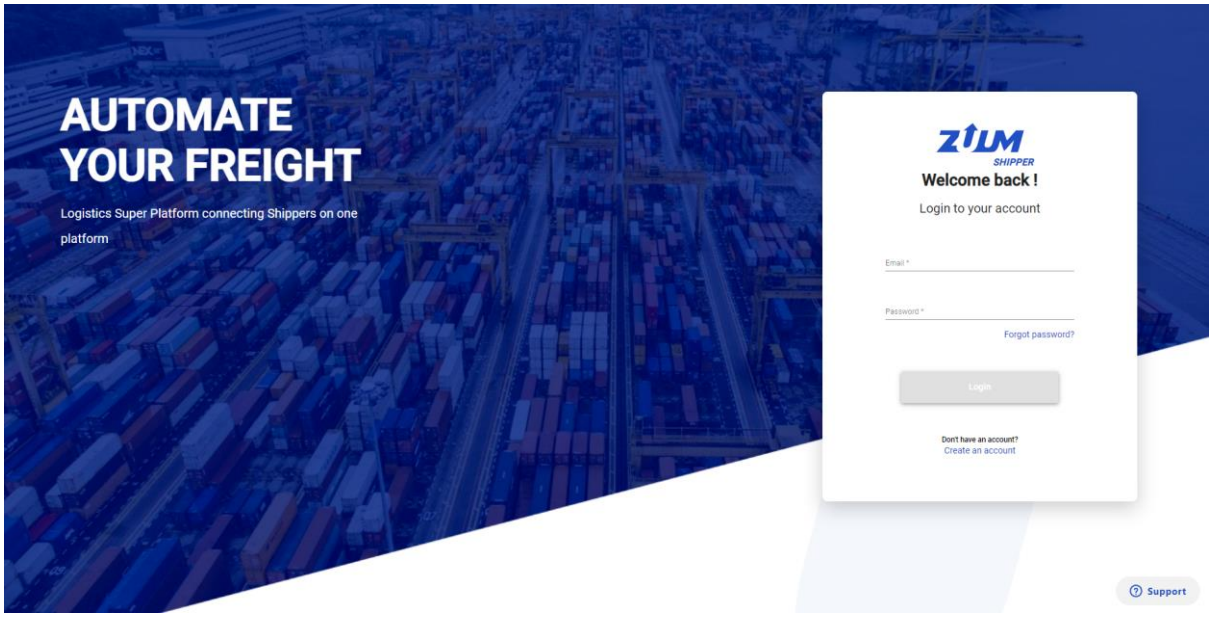

Рисунок 1.2 – Сторінка авторизації

Налаштування акаунту. Користувач має змогу змінити інформацію про свою компанію та свій акаунт, а також додати нових співробітників до системи та виділити їм певну роль. Приклад сторінки зі списком співробітників можна побачити на рисунку 1.3.

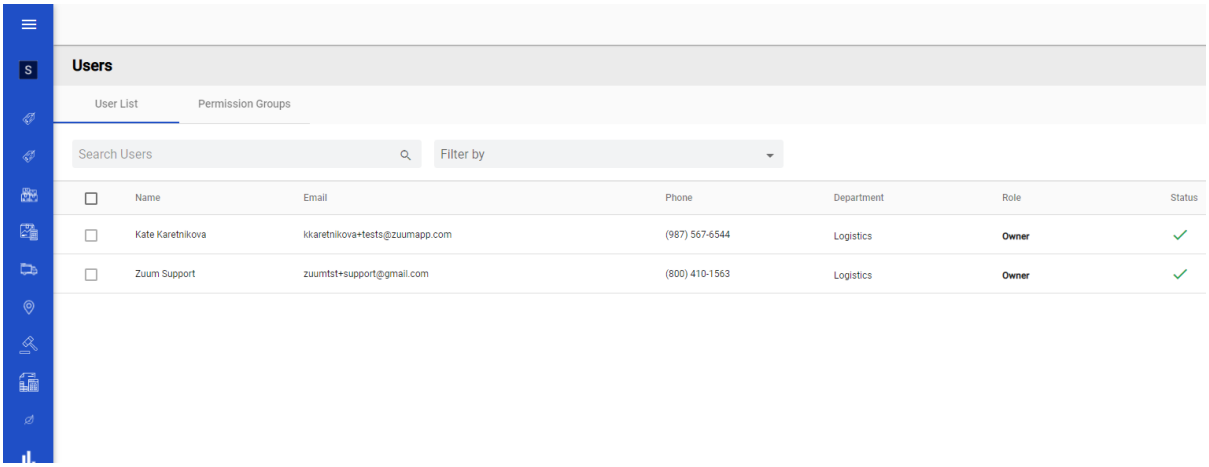

Рисунок 1.3 – Приклад сторінки зі списком користувачів

Створення нового відправлення. Користувач може заповнити усю інформацію про відправлення, включаючи дату та вагу, а також обрати перевізника для цього вантажу. Перевізником може бути брокер або компанія, з якою співпрацює відправник. Приклад форми для заповнення інформації про вантаж можна побачити на рисунку 1.4.

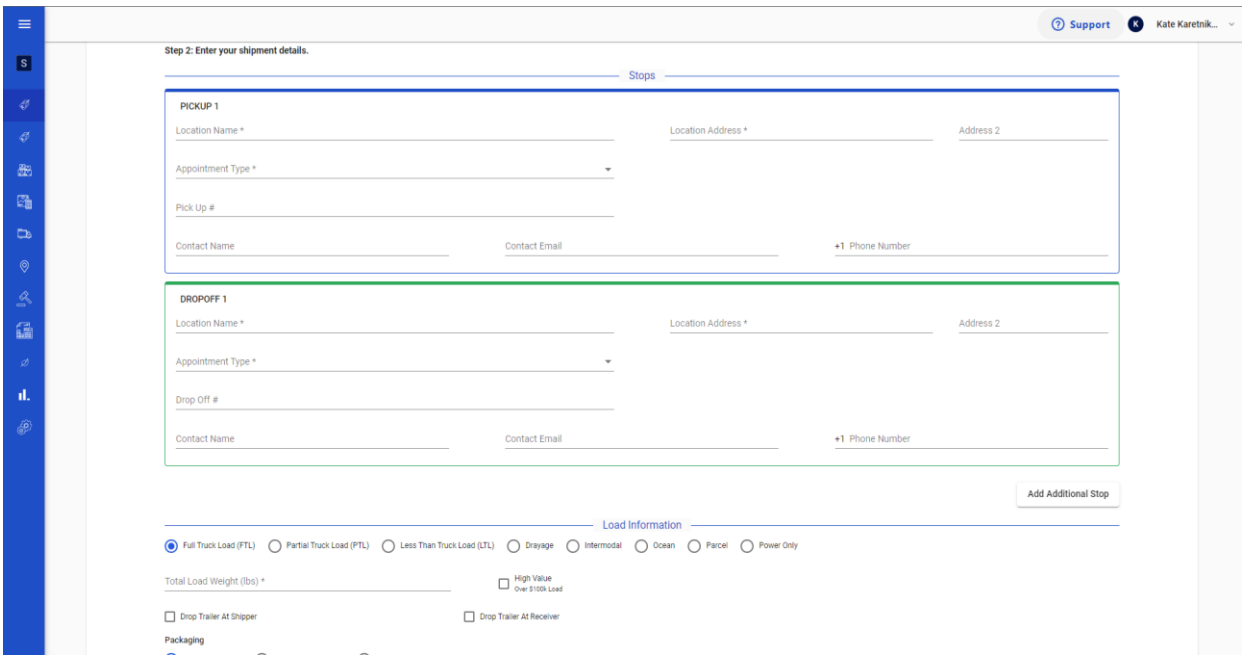

Рисунок 1.4 – Приклад форми для заповнення інформації про вантаж

Можливість додати інформацію про перевізника. Користувач може додати до системи інформацію про перевізника, з яким він співпрацює та

потім робити на нього тендер своїх вантажів. Також повинна бути можливість редагування та видалення раніше доданих до системи компаній перевізників. Приклад форми для додавання перевізника можна побачити на рисунку 1.5.

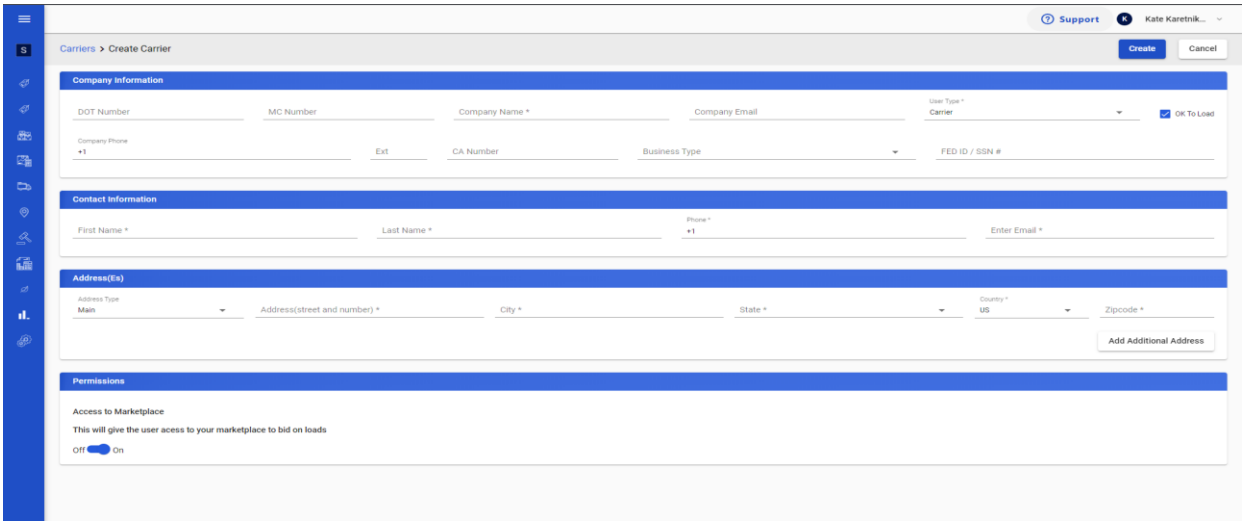

Рисунок 1.5 – Приклад форми для додавання перевізника

1.5 Вимоги до вебдодатку, що тестується

Як вже було описано у попередньому пункті, вебдодаток, який було обрано для тестування призначений для бізнесу, тобто для компаній, яким постійно потрібно кудись відправляти свої вантажі. Отже, першою вимогою для вебдодатку буде наявність функцій реєстрації компанії та авторизації користувача. Також, оскільки у системі буде зареєстровано саме компанію, необхідною буде і функція запрошення до системи інших співробітників компанії. Однією з основних функцій є також і створення нового запиту на доставку вантажу.

Виходячи з того, з якою саме метою було створено обраний для тестування вебдодаток було визначено основні вимоги до системи:

– система повинна мати функцію реєстрації нової компанії та авторизації вже зареєстрованих користувачів;

– користувачу повинно бути заборонено реєструвати більше ніж одну компанію використовуючи одну електрону адресу. Тобто для кожної компанії – унікальний email;

– кожен користувач повинен мати певну роль, а також права доступу до вебдодатку;

– користувачі кожної компанії повинні мати можливість запросити до системи інших співробітників компанії. Для цього потрібно вказати прізвище, ім’я, e-mail, номер телефону, роль, а також права доступу нового користувача;

– на сторінці зі списком співробітників повинна бути функція пошуку користувачів;

– система повинна мати функцію редагування даних як про самих користувачів, так і про компанії;

– користувачі системи повинні мати змогу створити новий запит на доставку вантажу, вказавши при цьому хто саме має займатися доставкою, а також основні відомості про вантаж: адресу відправника, адресу одержувача, бажані дати відправлення та отримання вантажу, тип вантажу, тип вантажівки, вага вантажу;

– запит на доставку вантажу не може бути створений якщо не вказано назву компанії перевіника;

– повинна бути наявна валідація для бажаних дат відправки та доставки вантажу. Дата відправки не може бути пізніше ніж дата доставки;

– система повинна надавати користувачам можливість додавати, редагувати, а також видаляти інформацію про компанії перевізників;

– користувачу повинно бути заборонено створювати дублікати компаній перевізників;

– система повинна виконувати валідацію певних полів, наприклад таких як номер телефону, електрона адреса, тощо.

2 ПРОЄКТУВАННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ

2.1 Створення тестових випадків

Перед тим, як перейти до розробки системи автоматизованого тестування потрібно визначити набір тест-кейсів, адже вони є необхідними для створення авто тестів [4]. Важливо перевірити поведінку системи як у позитивних, так і негативних, наприклад, якщо було введено невалідні дані, сценаріях. Отже, створимо тест-кейси для кожного з модулів.

Перший модуль – авторизація користувача. Для того, щоб почати користуватись системою користувач має зареєструвати нову компанію. При позитивному сценарії у системі повинен бути створений новий акаунт. Тест-кейс зображено на таблиці 2.1.

Таблиця 2.1 – Тест-кейс “Реєстрація компанії, позитивний сценарій”

№	Крок	Очікуваний результат
1	Відкрити сторінку “Login”	Відображається сторінка “Login”
2	Натиснути на кнопку “Create account”	Користувача перенаправлено на сторінку “Shipper sign up”
3	Заповнити поле “Email” валідними даними	В полі “Email” відображено введені дані
4	Заповнити поле “Company Name” валідними даними	В полі “Company Name” відображено введені дані
5	Заповнити поле “First Name” валідними даними	В полі “First Name” відображено введені дані
6	Заповнити поле “Last Name” валідними даними	В полі “Last Name” відображено введені дані

Продовження таблиці 2.1

№	Крок	Очікуваний результат
7	Заповнити поле “Phone Number” валідними даними	В полі “Phone Number” відображено введені дані
8	Заповнити поле “Password” валідними даними	В полі “Password” відображено введені дані
9	Заповнити поле “Confirm Password” валідними даними	В полі “Confirm Password” відображено введені дані
10	Відмітити прапорець “I agree to the Terms and Conditions and Privacy Policy”	Прапорець “I agree to the Terms and Conditions and Privacy Policy” відмічено, кнопка “Next” активна
11	Натиснути на кнопку “Next”	Користувача перенаправлено на сторінку “Verification”
12	Ввести код підтвердження, по одній цифрі у кожну комірку	Код підтвердження відображено на сторінці “Verification”
13	Натиснути на кнопку “Next”	Користувача ере направлено на сторінку “Company Information”
14	Заповнити поле “Address (Street and Number)” валідними даними	В полі “Address (Street and Number)” відображено введені дані
15	Заповнити поле “City” валідними даними	В полі “City” відображено введені дані
16	Заповнити поле “Zipcode” валідними даними	В полі “Zipcode” відображено введені дані
17	Заповнити поле “State” валідними даними	В полі “State” відображено введені дані
18	Заповнити поле “Company Phone” валідними даними	В полі “Company Phone” відображено введені дані

Продовження таблиці 2.1

№	Крок	Очікуваний результат
19	Заповнити поле “Federal Tax ID (EIN)” валідними даними	В полі “Federal Tax ID (EIN)” відображено введені дані
20	Натиснути на кнопку “Complete”	На сторінці відображено “Sign Up Complete!” повідомлення та кнопка “Login”

Також створимо декілька тест-кейсів для перевірки негативних сценаріїв використання. Перший тест-кейс буде стосуватися перевірки валідації поля Email (див. табл. 2.2) та щодо валідації на унікальність користувача у системі (див. табл. 2.3). Вебдодаток повинен адекватно реагувати на невалдіні дані шляхом відображення на екрані повідомлення про помилку.

Таблиця 2.2 – Перевірка валідації поля “Email” на сторінці реєстрації нової компанії

№	Крок	Очікуваний результат
1	Відкрити сторінку “Login”	Відображається сторінка “Login”
2	Натиснути на кнопку “Create account”	Користувача перенаправлено на сторінку “Shipper sign up”
3	Заповнити поле “Email” невалідними даними, наприклад karetnikova,gamil.com або kate@gmail@com	Нижче поля “Email” відображається “Email format not valid” помилка

Таблиця 2.3 – Перевірка валідації на унікальність користувача у системі під час реєстрації нової компанії

№	Крок	Очікуваний результат
1	Відкрити сторінку “Login”	Відображається сторінка “Login”
2	Натиснути на кнопку “Create account”	Користувача перенаправлено на сторінку “Shipper sign up”
3	Ввести в поле “Email” такий email, який вже було зареєстровано раніше	На екрані відображається “The email you entered matches this Zuum Account” модальне вікно
4	Натиснути На кнопку “Cancel” у “The email you entered matches this Zuum Account” модальному вікні	Нижче поля “Email” відображається “Already registered” помилка

Далі створимо декілька тест кейсів для сторінки авторизації. Для виконання позитивного сценарію користувач повинен ввести валідні логін та пароль та натиснути на кнопку “Login” (див. табл. 2.4). Також при виконанні негативного сценарію, коли користувачем введено неправильний пароль, система повинна відобразити на екрані повідомлення про помилку (див. табл. 2.5).

Таблиця 2.4 – Перевірка функції авторизації користувача

№	Крок	Очікуваний результат
1	Відкрити сторінку “Login”	Відображається сторінка “Login”
2	Ввести електронну пошту у поле “Email”	В полі “Email” відображено введені дані
3	Ввести в поле “Password” пароль	В полі “Password” відображено пароль у вигляді послідовності крапок

Продовження таблиці 2.4

№	Крок	Очікуваний результат
4	Натиснути На кнопку “Login”	Користувача перенаправлено на сторінку “Shipments”

Таблиця 2.5 – Перевірка наявності повідомлення про помилку при введенні неправильного паролю

№	Крок	Очікуваний результат
1	Відкрити сторінку “Login”	Відображається сторінка “Login”
2	Ввести електронну пошту у поле “Email”	В полі “Email” відображено введені дані
3	Ввести в поле “Password” неправильний пароль	В полі “Password” відображено пароль у вигляді послідовності крапок
4	Натиснути На кнопку “Login”	Нижче поля “Password” відображено помилку “Invalid password.”

Після реєстрації у системі користувач має можливість редагувати інформацію як про власний акаунт, так і про свою компанію. Отже, напишемо тест кейси для перевірки можливості редагувати інформацію про власний акаунт (див. табл. 2.6), а також неможливості редагувати поле Federal Tax Id (див. табл. 2.7), адже за специфікаціями користувач не має права редагувати ці дані. Також напишемо тест кейс для перевірки валідації номеру телефону на сторінці налаштування аканту (див. табл. 2.8), адже користувач не повинно дозволятись ввести у це поле будь-які знаки крім цифр.

Таблиця 2.6 – Перевірка можливості редагування інформації про свій акаунт

Передумови: 1) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Settings”	Розкрито випадаюче меню “Settings”
2	Натиснути на кнопку “Account”	Користувача перенаправлено на сторінку “Account”
3	Ввести в поле “Phone Number” новий номер телефону	Поле “Phone Number” містить новий номер телефону
4	Натиснути на кнопку “Upload Profile Picture”	На екрані з’явилось діалогове вікно вибору нового зображення
5	Оберіть нове зображення	На екрані відображається превью новообраного зображення
6	Натиснути на кнопку “Save Changes”	На екрані відображено “Profile updated” повідомлення

Таблиця 2.7 – Перевірка неможливості редагування поля “Federal Tax Id”

Передумови: 1) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Settings”	Розкрито випадаюче меню “Settings”
2	Натиснути на кнопку “Company Profile”	Користувача перенаправлено на сторінку “Company Profile”
3	Натиснути на поле “Federal Tax Id”	Поле “Federal Tax Id” не активне.

Таблиця 2.8 – Перевірка валідації поля “Phone Number” на сторінці “Account”

Передумови: 1) Наявність акаунту в системі 2) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Settings”	Розкрито випадаюче меню “Settings”
2	Натиснути на кнопку “Account”	Користувача перенаправлено на сторінку “Account”
3	Ввести в поле “Phone Number” невалідний номер телефону, наприклад: 09htr567h35ju783	Поле “Phone Number” містить тільки цифри, які вводив користувач. Усі інші знаки видалено

Наступний модуль, для якого будуть написані тест кейси – це додавання співробітників до системи та управління їх акаунтами. Будуть написані текст-кейси для додавання нового співробітника (див. табл. 2.9), видалення співробітника (див. табл. 2.10), деактивації співробітника (див. табл. 2.11) та пошуку за ім’ям (див. табл. 2.12).

Таблиця 2.9 – Перевірка функції додавання нового співробітника

Передумови: 1) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Settings”	Розкрито випадаюче меню “Settings”
2	Натиснути на кнопку “Users”	Користувача перенаправлено на сторінку “Users”

Продовження таблиці 2.9

№	Крок	Очікуваний результат
3	Натиснути на кнопку “Add user”	На екрані з’явилось модальне вікно з формою для додавання нового користувача
4	Ввести у поле “First Name” ім’я співробітника	Поле “First Name” містить ім’я співробітника
5	Ввести у поле “Last Name” прізвище співробітника	Поле “Last Name” містить прізвище співробітника
6	Ввести у поле “Phone Number” телефон співробітника	Поле “Phone Number” містить номер телефону співробітника
7	Ввести у поле “Email” електрону адресу співробітника	Поле “Email” містить електрону адресу співробітника
8	Обрати роль для нового користувача	Чекбокс вибрано навпроти обраної ролі
9	Обрати права доступу до системи для нового користувача	На сторінці відмічено усі радіо кнопки, які знаходяться навпроти обраного типу прав доступу
10	Натиснути на кнопку“Save”	На екрані відображається повідомлення “User invited to join as a shipper”

Таблиця 2.10 – Перевірка функції видалення співробітника з системи

Передумови:
1) Наявність акаунту в системі 2) Користувач авторизований в системі

Продовження таблиці 2.10

№	Крок	Очікуваний результат
1	Натиснути на кнопку “Settings”	Розкрито випадаюче меню “Settings”
2	Натиснути на кнопку “Users”	Користувача перенаправлено на сторінку “Users”
3	Натиснути на три крапки навпроти співробітника, якого потрібно видалити з системи	На екрані з’явилось контекстне меню, яке містить список дій, які можна виконати з акаунтом
4	Натиснути на кнопку “Remove”	На екрані з’явилось модальне вікно “Remove user”, що запитує підтвердження дії
5	Натиснути на кнопку “Confirm”	На екрані відображається повідомлення “The user has been removed”. Інформація про співробітника зникла з сторінки

Таблиця 2.11 – Перевірка функції деактивації аканта користувача

Передумови: 1) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Settings”	Розкрито випадаюче меню “Settings”
2	Натиснути на кнопку “Users”	Користувача перенаправлено на сторінку “Users”
3	Натиснути на три крапки навпроти співробітника, акаунт якого потрібно деактивувати	На екрані з’явилось контекстне меню, яке містить список дій, які можна виконати з акаунтом співробітника

Продовження таблиці 2.11

4	Натиснути на кнопку “Deactivate User”	На екрані з’явилось модальне вікно “Deactivate user”, що повідомляє про те, що користувач не буде мати доступу до свого акаутну
5	Натиснути на кнопку “Confirm”	На екрані відображається повідомлення “User deactivated”. Інформація про користувача присутня на сторінці “Users”

Таблиця 2.12 – Перевірка функції пошуку користувача за ім’ям

Передумови: 1) Наявність акаунту в системі 2) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Settings”	Розкрито випадаюче меню “Settings”
2	Натиснути на кнопку “Users”	Користувача перенаправлено на сторінку “Users”
3	Ввести у поле “Search Users” ім’я користувача, якого потрібно знайти у системі	В полі “Search Users” відображаються введені дані. На сторінці “Users” відображається список користувачів, ім’я яких містить текст, введений у поле “Search Users”

Наступний модуль який потрібно перевірити – це додавання до системи та управління інформацією про компанії перевізників. Отже, створимо тест кейс для перевірки функції додавання до системи інформації про компанію перевізника (див. табл. 2.13). Також, якщо користувач має

функцію додавання інформації до системи, то повинен мати можливість і редагування та видалення, тож створимо тестові випадки для перевірки можливості редагування (див. табл. 2.14) та видалення (див. табл. 2.15) інформації про компанію перевізника. Також у якості перевірки негативного випадку створимо тест кейс для перевірки неможливості додавати дублікати компаній перевізників (див. табл. 2.16)

Таблиця 2.13 – Перевірка можливості додавання інформації про компанію перевізника

Передумови: 1) Наявність акаунту в системі 2) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Carriers”	Користувача перенапрвлено на сторінку “ Carriers”
2	Натиснути на кнопку “Create Carrier”	Користувача перенаправлено на сторінку “Create Carrier”
3	Ввести в поле “Company Name” назву компанії перевізника	Поле “Company Name” містить назву компанії перевізника
4	Ввести у поле “First Name” секції “Contact Information” ім’я контактної особи перевізника	Поле “First Name” містить ім’я контактної особи перевізника
5	Ввести у поле “Last Name” секції “Contact Information” прізвище контактної особи перевізника	Поле “Last Name” містить прізвище контактної особи перевізника

Продовження таблиці 2.13

6	Ввести у поле “Phone Number” секції “Contact Information” телефон контактної особи перевізника	Поле “Phone Number” містить номер телефону контактної особи перевізника
7	Ввести у поле “Email” секції “Contact Information” електрону адресу контактної особи перевізника	Поле “Email” містить електрону адресу контактної особи перевізника
8	Натиснути на кнопку “Create”	Користувач бачить повідомлення “Carrier successfully added” Користувача перенаправлено на сторінку “Carriers”

Таблиця 2.14 – Перевірка можливості редагування інформації про компанію перевізника

Передумови: 1) Наявність акаунту в системі 2) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Carriers”	Користувача перенаправлено на сторінку “ Carriers”
2	Натиснути на кнопку “Create Carrier”	Користувача перенаправлено на сторінку “Create Carrier”
3	Ввести в поле “Company Email” нову електрону адресу	Поле “Company Email” містить введені дані
4	Натиснути на кнопку “Save Changes”	На екрані з’явилось повідомлення “Carrier Updated”, поле “Company Email” містить оновлені дані

Таблиця 2.15 – Перевірка можливості видалення інформації про компанію перевізника

Передумови: 1) Наявність акаунту в системі 2) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Carriers”	Користувача перенаправлено на сторінку “ Carriers”
2	Натиснути на три крапки навпроти компанії перевізника, яку потрібно видалити з системи	На екрані з’явилось контекстне меню, яке містить список дій, які можна виконати з компанією перевізника
3	Натиснути на кнопку “Remove”	На екрані з’явилось модальне вікно “Delete Carrier”, що запитує підтвердження дії
4	Натиснути на кнопку “Confirm”	На екрані відображається повідомлення “Carrier has been removed”. Інформація про компанію зникла зі сторінки

Таблиця 2.16 – Перевірка неможливості додавати дублікати компаній перевізників

Передумови: 1) Наявність акаунту в системі 2) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Carriers”	Користувача перенаправлено на сторінку “ Carriers”

Продовження таблиці 2.16

№	Крок	Очікуваний результат
2	Натиснути на кнопку “Create Carrier”	Користувача перенаправлено на сторінку “Create Carrier”
3	Заповнити поле “Email” секції “Contact Information” електроною адресою, яка вже використовується компанією, що була додана раніше	Нижче поля “Email” відображається “Already Registered” помилка

Останній модуль, який буде протестовано – це створення запиту на доставку вантажу. У якості негативних випадків перевіримо валідацію бажаних дат відправки та доставки вантажу. Дата відправки не повинна бути пізніше ніж дата отримання вантажу (див. табл. 2.17). Також користувач обов’язково повинен вибрати бажану компанію перевізника (див. табл. 2.18). У якості позитивного випадку перевіримо можливість створення запиту на перевезення вантажу (див. табл. 2.19).

Таблиця 2.17 – Перевірка валідації дат для відправки та доставки вантажу

Передумови: 1) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Build Shipment”	Розкрито випадаюче меню “Build Shipment”
2	Натиснути на кнопку “Send Direct Tender”	Користувача перенаправлено на сторінку “Build Shipment”

Продовження таблиці 2.17

№	Крок	Очікуваний результат
3	Натиснути на поле “Select Date” у секції “Dropoff”	На сторінці з’являється календар
4	Обрати бажану дату доставки вантажу	Поле “Select Date” заповнене обраною датою
5	Натиснути на поле “Select Date” у секції “Pickup”	На сторінці з’являється календар
6	Обрати дату, що є пізніше ніж дата, яка була обрана для доставки вантажу	На сторінці з’явилось повідомлення про помилку, що містить текст “It is not possible to start the dropoff before the pickup”

Таблиця 2.18 – Перевірка неможливості створення запиту на відправку вантажу без введення назви компанії перевізника

Передумови: 1) Наявність акаунту в системі 2) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Build Shipment”	Розкрито випадаюче меню “Build Shipment”
2	Натиснути на кнопку “Send Direct Tender”	Користувача перенаправлено на сторінку “Build Shipment”
3	Заповнити усі обов’язкові поля, окрім поля “Select Other Carrier”	Усі обов’язкові поля містять введену інформацію
4	Натиснути на кнопку “Confirm”	На екрані з’явилась “Who do you want to send the direct tender to?” помилка

Таблиця 2.19 – Перевірка можливості створити новий запит на доставку вантажу

Передумови: 1) Наявність акаунту в системі 2) Користувач авторизований в системі		
№	**Крок**	**Очікуваний результат**
1	Натиснути на кнопку “Build Shipment”	Розкрито випадаюче меню “Build Shipment”
2	Натиснути на кнопку “Send Direct Tender”	Користувача перенаправлено на сторінку “Build Shipment”
3	Натиснути на кнопку “Select Other Carrier”	На екрані відображається “Carrier Tender Options” модальне вікно
4	Обрати перевізника зі списку	У полі “Search Carrier” відображається назва обраної компанії-перевізника
5	Натиснути на кнопку “Tender”	Модальне вікно “Carrier Tender Options” зникло з екрану
6	Заповнити поле “Location Address” у секції “Pickup 1”	У полі “Location Address” у секції “Pickup 1” відображено введені дані
7	Заповнити поле “Location Address” у секції “Dropoff 1”	У полі “Location Address” у секції “Dropoff 1” відображено введені дані
8	Ввести вагу вантажу в поле “Total Load Weight (lbs)”	У полі “Total Load Weight (lbs)” відображено введені дані
9	Ввести назву товару у поле “Commodity”	У полі “Commodity” відображено введені дані
10	Натиснути на поле “Select Truck Type”	Відкрито модальне вікно “Select Equipment Type”

Продовження табл. 2.19

№	Крок	Очікуваний результат
10	Натиснути на поле “Select Truck Type”	Відкрито модальне вікно “Select Equipment Type”
11	Відмітити прапорці навпроти хоча б одного типу вантажівки	Усі обрані прапорці відмічено
12	Натиснути на кнопку “Confirm”	У полі “Select Truck Type” відображено усі обрані у “Select Equipment Type” модальному вікні типи вантажівок
13	Натиснути на кнопку “Continue”	Користувача перенаправлено на сторінку “Shipment Summary”
14	Відмітити прапорець “I agree to the Terms and Conditions”	Прапорець “I agree to the Terms and Conditions” відмічено, кнопка “Book this Load” активна
15	Натиснути на кнопку “Book this Load”	Користувача пренаправлено на сторінку, що містить деталі новоствореного вантажу

2.2 Створення діаграми класів

Під час написання автотестів буде використовуватись один з найбільш популярних в автотестуванні патернів Page Object Model [5].

Суть патерну полягає в тому що одна сторінка вебдодатку розглядається як окремий клас, який містить набір методів для роботи з певними елементами цієї сторінки. Для виконання певного автотесту буде достатньо побудувати ланцюг викликів методів класу, який відповідає за потрібну сторінку [6].

Основними перевагами цього патерну є:

- поділ коду тестів та опис сторінок;
- об'єднання всіх дій по роботі з вебсторінкою в одному місці.

Доцільність використання цього патерну можна побачити розглянувши простий приклад, коли декілька наборів тест-кейсів, які можливо виконати тільки від ім’я авторизованого користувача. У такій ситуації перед виконанням кожного набору тестів потрібно виконувати вхід користувача у систему.

Якщо перед кожним тестом просто прописувати послідовність дій, які необхідно виконати під час входу у систему, то у ситуації, коли, наприклад, буде оновлено дизайн сторінки доведеться змінювати код входу у систему перед кожним набором тестів. Якщо ж сторінку входу представити у вигляді окремого класу, прописати необхідні методи для роботи з цією сторінкою, а потім ці методи викликати перед кожним набором тест-кейсів, то за необхідністю можна змінити код методів цього класу, не задумуючись про те, перед якими тестами потрібно виконати вхід у систему. Отже, патерн Page Object Model суттєво спрощує розробку та подальшу підтримку автотестів.

Створимо діаграму класів майбутньої системи автоматизованого тестування. У тестування беруть участь шість основних сторінок, отже створимо шість різних класів, а саме: SignUpPage, LogInPage, ShipmentsBuildPage, SettingsAccountPage, SettingsUserPage та NewCarrierPage, які призначено для роботи зі сторінками реєстрації компанії (Sign Up page), входу до системи (Log In page), створення запиту на відправлення вантажу (Build Shipment page), налаштування акаунту (Account page), управління співробітниками компанії (Users page) та додавання до системи інформації про перевізника відповідно (Add new carrier page) (див. рис. 2.1).

Методи кожного з цих класів використовуватимуться при створенні набору з тест-кейсів. Для того, щоб створити набір тест кейсів буде достатньо імпортувати один або декілька класів, що відповідають за потрібні сторінки та викликати методи цих класів у потрібному порядку.

Окремий набір тест-кейсів можна вважати окремим класом, а кожен тест-кейс – як метод цього класу. Враховуючи це, створимо ще п’ять класів, а саме:

1) AuthorizationTests – клас, який містить у собі набір тест-кейсів пов’язаних з тестуванням реєстрації нової компанії та входу користувача у систему;

2) UsersTests – клас, який містить у собі набір тест-кейсів пов’язаних з тестуванням функціоналу, що стосується запрошення співробітників до системи та управління їх акаунтами;

3) CarrierTests – клас, який містить у собі набір тест-кейсів пов’язаних з додаванням до системи інформації про перевізників;

4) SettingsTests – клас, який містить у собі набір тест-кейсів пов’язаних з налаштуванням свого акаунту;

5) BuildShipmentTests – клас, який містить у собі набір тест-кейсів пов’язаних з тестуванням функціоналу створення нового запиту на перевезення вантажу.

Діаграму класів, які було описано вище, можна побачити на рисунку 2.1.

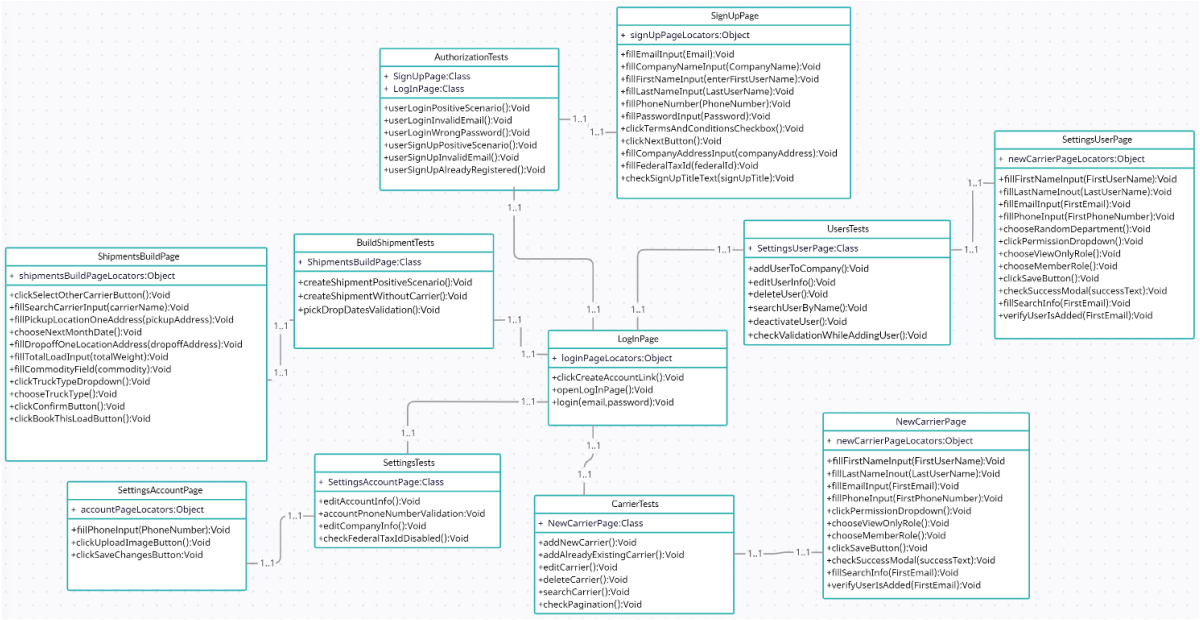

Рисунок 2.1 – Діаграма класів системи автоматизованого тестування

2.3 Створення діаграми послідовності виконання тестів

Для відображення задіяних об’єктів наведемо діаграму послідовності (рисунок 2.2).

Як приклад, наведено тест-кейс, який виконує перевірку можливості входу користувача у систему (див. рис. 2.2). Тестувальник виконує запуск певного тесту, cypress створює нову сесію та запускає браузер, у якому буде проводитись тестування. Після цього відбувається виконання послідовності команд, тобто так званих кроків тест-кейсу, які необхідно виконати для перевірки певного кейсу. У нашому випадку це заповнення полів “Login” та “Password”. Перед тим як натиснути на кнопку “Login” створюємо для cypress задачу на перехоплення відповіді на запит “/login”. Це потрібного для того, щоб дізнатись з яким кодом прийшла відповідь від сервера. Далі виконується наступна команда, тобто натискається кнопка “Login”. Після цього надсилається запит “/login” на сервер. Далі сервер обробляє цей запит та надсилає відповідь, яку перехоплює cypress. Так як під час тестування було введено валідні дані, то сервер має надіслати відповідь з кодом 200. Отже, далі cypress виконує порівняння отриманого у відповіді коду з заданим. Якщо відповідь прийшла з кодом 200, то все працює як потрібно, якщо ні, то є якась проблема.

Як видно, під час тесту можна перевірити як роботу серверної частини вебдодатку, так і взаємодія клієнта і сервера. Також, якщо тест не було пройдено, проаналізувавши код відповіді від сервера можна не лише знайти проблеми у роботі коду серверної частини додатку, а і помітити проблеми в роботі самого сервера.

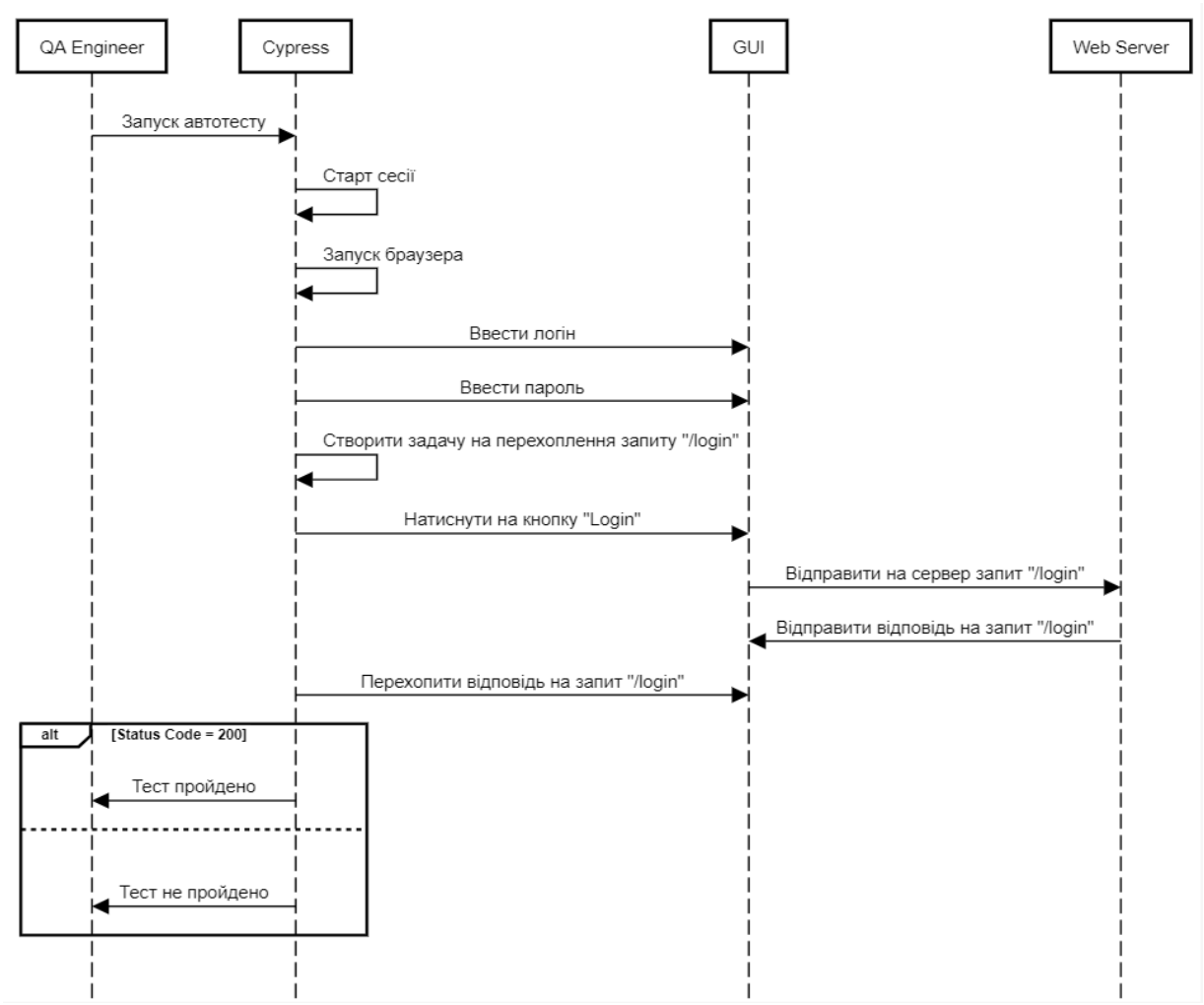

Рисунок 2.2 – Діаграма послідовності виконання тесту

3 РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ

3.1 Створення та ініціалізація проєкту

Перед тим, як приступити до розробки системи автоматизованого тестування потрібно створити проєкт та виконати його налаштування [7]. Встановимо cypress за допомогою команди npm install cypress та виконаємо запуск проєкту за допомогою команди npx cypress open. Після цього буде згенеровано так званий каркас проєкту, який буде використано для розробки. Також встановимо додатково плагін cypress-file-upload, який дозволить завантажувати свої файли на сайт. Це допоможе при тестуванні тих тестових випадків, коли користувачу потрібно завантажити файли зі свого комп'ютера. Також додатково встановимо плагін allure, який знадобиться для генерації звітів.

Виконаємо налаштування файлу "cypress.json". Для цього пропишемо параметри "pageLoadTimeout", "responseTimeout". Ці параметри вказують час очікування завершення завантаження сторінки та повернення відповіді на запит відповідно. Також встановлюємо розмір робочої зони за допомогою параметрів "viewportHeight" та "viewportWidth". У об'єкті "env" пропишемо значення для параметру "allureResultsPath", який буде вказувати шлях до теки, у якій будуть зберігатися результати звітів allure. Також пропишемо значення "true" для параметру "screenshotOnRunFailure" щоб зберігались скрішоти у випадку якщо тест не було пройдено. Код файлу cypress.json можна побачити на рисунку 3.1.

```
{
   "viewportHeight": 768,
   "viewportWidth": 1280,
   "defaultCommandTimeout": 25000,
   "pageLoadTimeout": 50000,
   "requestTimeout": 30000,
   "responseTimeout": 30000,
   "screenshotOnRunFailure": true,
   "integrationFolder": "cypress/integration/tests/",
   "env": {
      "allure": true,
      "allureResultsPath": "allure-results"
    }
   }
```

Рисунок 3.1 – Код файлу cypress.json

Директорія “integration” містить два підкаталоги: “pages” та “tests”. Підкаталог “pages” містить класи, що призначені для роботи з певними сторінками. У підкаталозі “tests” будуть описані самі тести. У цьому каталозі будемо використовувати імпортувати та методи класів з каталогу “pages”.

У директорії “fixtures”, у підкаталозі “src” будуть зберігатися тестові зображення, які будуть використовуватись у тестах, що вимагають завантаження файлів у вебдодаток.

Директорія “plugins” містить у собі файл “index.js”, у якому будемо прописувати ініціалізацію усіх додаткових бібліотек, які будуть використані під час розробки системи автоматизованого тестування.

Директорія “allure-results” призначена для зберігання результатів для звітів allure. У ній зберігаються файли, які містять логи виконання кожного тесту. Потім ці файли будуть використані для генерації звіту у форматі html.

Повну структуру проєкту можна побачити на рисунку 3.2.

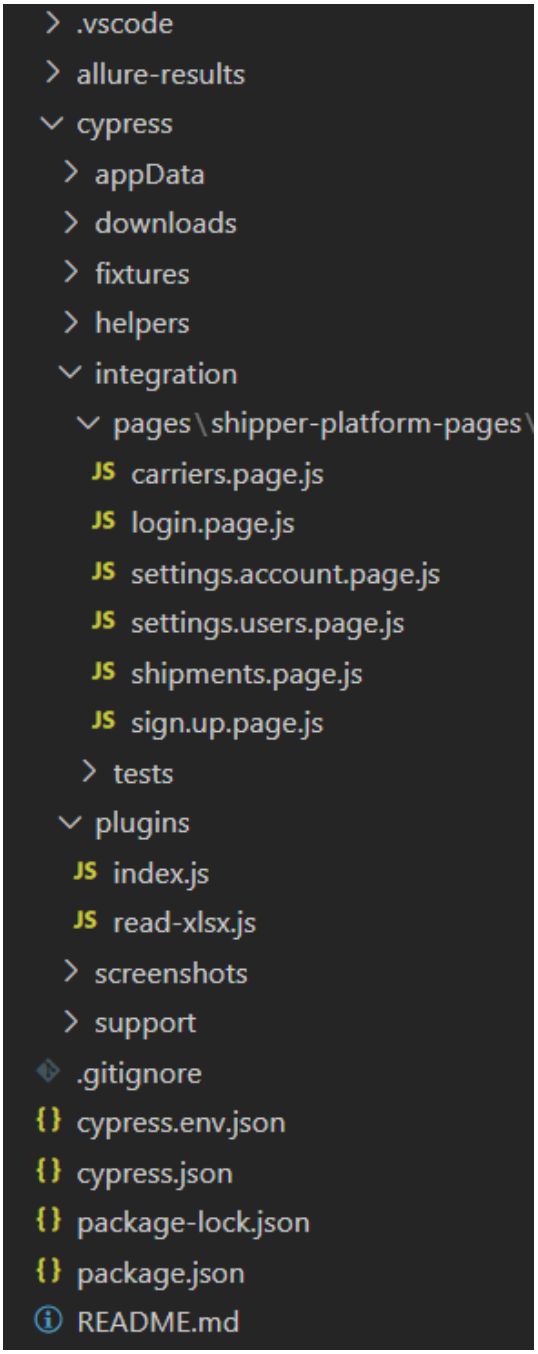

Рисунок 3.2 – Файлова структура проекту

Для відображення звіту зі статистикою та логами тестів Allure потрібно завантажити та виконати налаштування бібліотеки [8]. Спочатку виконуємо завантаження бібліотеки за допомогою команди “npm i -D @shelex/cypress-allure-plugin”. Далі виконуємо налаштування конфігурації бібліотеки. Для цього у файлі index.js, що знаходиться у директорії “plugins”, виконуємо ініціалізацію allure. Код представлено на рисунку 3.3.

```
const allureWriter = require('@shelex/cypress-allure-plugin/writer');
module.exports = (on, config) => {
   allureWriter(on, config);
   return config;};
```

Рисунок 3.3 – Код налаштування конфігурації Allure

3.2 Написання коду системи автоматизованого тестування

Написання коду розпочнемо з класів, що призначені для роботи з певними сторінками, адже згідно з патерном Page Object Model методи цих класів будуть використані при написанні самих тестів.

Так як для роботи у системі користувачеві спочатку потрібно зареєструватися та авторизуватися, то створимо два класи: SignUpPage – для роботи зі сторінкою авторизації та LogInPage – для роботи зі сторінкою входу.

Перед створенням класу SignUpPage пропишемо об’єкт signUpPageLocators, який буде містити усі атрибути, що знадобляться для пошуку всіх потрібних нам елементів. Це потрібно для того, щоб прописати ці атрибути один раз та не використовувати їх щоразу у кожному методі. Це значно полегшить підтримку коду якщо який з атрибутів зміниться. Код об’єкту signUpPageLocators можна знайти на рисунку 3.4.

```
const signUpPageLocators = {
   emailInput: 'input[data-placeholder="Email"]',
   companyNameInput: 'input[data-placeholder="Company Name"]',
   firstNameInput: 'input[data-placeholder="First Name"]',
   lastNameInput: 'input[data-placeholder="Last Name"]',
   phoneNumberInput: 'input[data-placeholder="Phone Number"]',
   passwordField: 'input[data-placeholder="Password"]',
   termsAndConditionsCheckbox: '.mat-checkbox',
   nextStepButton: 'button.mat-raised-button.mat-primary',
   addressInput: 'input[data-placeholder="Address (Street and Number)"]',
```

Рисунок 3.4 – Код об’єкту signUpPageLocators

Далі створимо клас SignUpPage з методами для роботи з різними елементами сторінки. Пропишемо метод fillEmailInput для заповнення поля “Email”, який на вході у якості параметру приймає строку, email, який потрібно ввести в поле. У цьому методі за допомогою методу get() вибирається потрібне поле, в даному випадку поле “Email”, та вводиться email, що був переданий на вході за допомогою методу type() (див. рис. 3.5). Також на сторінці реєстрації є чекбокс “I agree with terms…”. Напишемо метод clickTermsAndConditionsCheckbox() для заповнення цього чекбоксу. Вибираємо HTML-елемент за допомогою методу get() та натискаємо на вибраний чекбокс за допомогою методу click(). Код методу можна побачити на рисунку 3.5. Також створимо метод clickNextButton() для роботи з кнопкою “Next”. Для цього також вибираємо HTML-елемент за допомогою методу get() та натискаємо на кнопку за допомогою методу .click(). Код усього класу подано у додатку А.

```
fillEmailInput(email) {
      cy.get(signUpPageLocators.emailInput)
        .type(email);
      return this;
   }

clickTermsAndConditionsCheckbox() {
      cy.get(signUpPageLocators.termsAndConditionsCheckbox).first()
        .click();
      return this;
   }
clickNextButton() {
      cy.get(signUpPageLocators.nextStepButton).click();
      return this;}
```

Рисунок 3.5 – Код основних методів для роботи зі сторінкою реєстрації

Створимо клас ShipperLogInPage. У цьому класі метод login() виконує заповнення полів Email та Password та натиснення на кнопку “Login”. Також створимо метод checkError(), який на вході приймає текст помилки та

виконує перевірку тексту помилки. Цей метод знадобиться при перевірці реакції вебдодатку на неправильно введені дані. Це може бути, наприклад, неправильний пароль або некоректний Email. У цьому методі спочатку вибираємо блок, у якому повинен відобразитися текст помилки за допомогою методу get(). Далі за допомогою ствердження should('have.text') перевіряємо правильність відображення тексту помилки. Також створимо метод fillEmailField(), який знадобиться під час перевірки валідації email. Даний метод заповнює поле “Email” даними за допомогою методів get() та type(). Також використаємо метод blur(), який прибирає фокус з поля, адже валідація поля “Email” відбувається лише після введення користувачем даних та зняття фокусу або переходу до наступного поля. Код основних методів класу ShipperLogInPage можна побачити на рисунку 3.6, а код усього класу у додатку А.

```
login(login, password) {
      cy.get(loginPageLocators.usernameInput).type(login);
      cy.get(loginPageLocators.passwordInput).type(password);
      cy.get(loginPageLocators.logInButton).click();
      return this; }
   fillEmailField(email) {
      cy.get(loginPageLocators.usernameInput).type(email).blur();
      return this }
   checkError(errorMsg) {
      cy.get(loginPageLocators.errorBlock).should('have.text', errorMsg)
      return this }
```

Рисунок 3.6 – Код основних методів класу ShipperLogInPage

Далі створимо клас ShipmentsBuildPage, методи якого будуть описувати дії на сторінці створення замовлення на перевезення. Створимо методи fillSearchCarrierInput() та chooseAccountFromDropdown() (див. рис. 3.7). Метод fillSearchCarrierInput() призначено для вводу у поле пошуку назви компанії перевізника, яку користувач хоче призначити для перевезення вантажу. Даний метод за допомогою методів .get(), .click(), clear() та .type()

вводить у поле пошуку назву компанії перевізника. Метод clear() виконує попереднє очищення від значення. Метод chooseAccountFromDropdown() обирає з дропдауну потрібну компанію. За допомогою методу .get() отримує список усіх компаній, наявних у дропдауні. Далі за допомогою ствердження should(‘contain’) наводить курсор на потрібну компанію та за допомогою методу click() обирає її зі списку.

Також створимо методи clickPickupStartDateCalendar() та chooseNextMonthDate() (див. рис. 3.8) для роботи з календарем. За допомогою методу clickPickupStartDateCalendar() виділяємо поле, що призначене для вводу бажаної дати відправлення чи отримання вантажу. Даний метод шукає потрібне поле за допомогою методу get() та виділяє це поле за допомогою методу click(). Після виділення поля на екрані повинен з’явитись календар. Метод chooseNextMonthDate(), який призначено для вибору дати у календарі, спочатку натискає на кнопку переходу до наступного місяця, вибирає потрібне число та натискає на нього за допомогою методу click({force: true}). Параметр force: true у методі click() вказує на те, що на елемент потрібно натиснути навіть якщо його перекривають інші елементи.

Далі створимо методи fillPickupLocationOneAddress() та fillDropoffOneLocationAddress() (див. рис. 3.9) для заповнення адрес відправника та адрес отримувача відповідно. У цьому методі спочатку відбувається пошук та вибір потрібного поля, а потім вводиться початок адреси. Далі за допомогою методу type('{downarrow}') вибирається перша адреса з випадаючого списку. Параметр {downarrow} служить для імітації натиснення стрілки “Вниз” на клавіатурі. Потім за допомогою методу click() відбувається імітація натиснення на обрану адресу.

```
fillSearchCarrierInput(enterCarrier) {
      cy.get(carrierTenderModalLocators.searchInput).click().clear()
      .type(enterCarrier);
      return this; }
chooseAccountFromDropdown() {
      cy.get(carrierTenderModalLocators.carrierOption)
         .should('contain', 'QA Carrier').click();
      return this;}
```

Рисунок 3.7 – Код методів для вибору компанії перевізника

```
  clickPickupStartDateCalendar() {
      cy.get(shipmentsBuildPageLocators.calendarInput).click()
      return this;
   }
 chooseNextMonthDate() {
      cy.get(shipmentsBuildPageLocators.nextMonthButton).click();
      cy.get(shipmentsBuildPageLocators.calendarDate).click({force: true});
      return this;
   }
```

Рисунок 3.8 – Код методів для заповнення дати

```
 fillPickupLocationOneAddress(pickupLocationAddress) {
      cy.get(shipmentsBuildPageLocators.locationAddressInput).click().clear()
         .type(pickupLocationAddress)
         .type('{downarrow}')
         .click();
      return this;}
  fillDropoffOneLocationAddress(enterDropoffAddress) {
      cy.get(shipmentsBuildPageLocators.locationAddressInput).last().click()
         .type(enterDropoffAddress).type('{downarrow}').click();
      return this;}
```

Рисунок 3.9 – Код методів для заповнення адреси

Перейдемо до створення тестів для перевірки тестових випадків. За патерном Page Object Model Для цього використаємо методи, які було створені під час написання класів, що призначені для роботи з певними сторінками. Наприклад, для перевірки входу користувача у систему з неправильно введеним паролем будемо використовувати методи описані у класі ShipperLogInPage у потрібному для виконання тесту порядку. Спочатку

викликаємо метод login(), який заповнить поля “Email” та “Password”, а потім виконає натиснення на кнопку “Login”. Потім викликаємо метод checkError(), який знайде блок, у якому повинна відобразитись помилка та перевірить правильність тексту помилки (див. рис. 3.10). Для написання тестів для перевірки створення запиту на перевезення вантажу будемо використовувати методи класу ShipmentsBuildPage.

```
it("Login, wrong password", () => {
      cy.visit(loginPageUrl)
      loginPage.login(Cypress.env("userEmail"),
Cypress.env("invalidPassword"))
      .checkError('Invalid password.')
})
```

Рисунок 3.10 – Перевірка спроби входу користувача з неправильним паролем

Також будемо використовувати у тестах і додаткові методи самої бібліотеки Cypress. Наприклад метод visit() будемо використовувати для переходу на сторінку з конкретною url-адресою, метод url() – для визначення поточної url адреси, метод title() – для визначення заголовку сторінки, на якій зараз виконуються тести, а метод intercept() – для перехоплення відповіді сервера на деякі важливі запити.

Код основних тестів для перевірки реєстрації та авторизації користувача у системі представлено на рисунку 3.11, а код тестів для перевірки створення запиту на перечення вантажу – на рисунку 3.12. Код усіх тестів можна знайти у додатку А.

```
it('Login, positive scenario', () => {
      cy.visit(loginPageUrl)
      cy.intercept('**').as('all')
      loginPage.login(Cypress.env("userEmail"), Cypress.env("userPass"))
      cy.url().should('contain', dashboardPageUrl);
         cy.title().should('eq', 'Zuum Shipper');
         cy.wait('@all').its('response.statusCode').should('be.oneOf', [200,201,300])
            .and('not.be.oneOf', [500,400]);})
it('Positive scenario: Create new account & Fill company info', () => {
      cy.visit(signUpPageUrl)
      cy.intercept('**').as('all')
signUpPage.checkTitle(signUpData.pageTitle).fillEmailInput(signUpData.enterE
mail)
         .fillCompanyNameInput(signUpData.enterCompanyName)
         .fillFirstNameInput(signUpData.enterFirstUserName)
         .fillLastNameInput(signUpData.enterLastUserName)
         .fillPhoneNumber(signUpData.enterPhoneNumber)
         .fillPasswordInput(signUpData.enterPassword)
         .fillConfirmPasswordInput(signUpData.enterPassword)
         .clickTermsAndConditionsCheckbox()
      cy.wait('@all').its('response.statusCode').should('be.oneOf', [200,201,300])
         .and('not.be.oneOf', [500,400]);
      cy.intercept('POST','**send-verification-code').as('getSignUpCode');
      signUpPage.clickNextButton();
      cy.wait('@getSignUpCode').then((resp) => {
         signUpPage.fillDigitInput(resp.request.body.message.toString().split(':
')[1]).clickSubmitButton();
      signUpPage.fillCompanyAddressInput(signUpData.companyAddress)
.fillCompanyCityInput(signUpData.city).fillZipcodeInput(signUpData.zipCode)
         .clickStateDropdown().chooseDropdownOption()
         .fillCompanyPhoneNumber(signUpData.enterPhoneNumber)
         .fillFederalTaxId(signUpData.federalId).clickCompleteButton()
         .checkSignUpTitleText(signUpData.signUpTitle).clickLoginButton();
         cy.url().should('contain', dashboardPageUrl);
         dashboardPage.clickShipmentsSideButton();
      shipmentBuildPage.clickBuildShipmentButton()}); });
```

Рисунок 3.11 – Код основних методів для перевірки реєстрації та авторизації користувача

```
it('Positive scenario: Create shipment', function() {
      cy.visit(Cypress.env('tenderPageUrl'));
      shipmentBuildPage.clickBuildShipmentButton()
         .chooseSendDirectTenderType().clickSelectOtherCarrierButton()

.fillSearchCarrierInput(shipmentData.carrierName).chooseAccountFromDropdown
()
         .fillCarrierPrice(shipmentData.carrierPrice)
.clickCarrierManagerDropdown()
         .chooseCarrierManager().clickSubmitTenderButton()
         .fillPickupOneLocationName(shipmentData.pickupLocation)
         .fillPickupLocationOneAddress(shipmentData.pickupAddress)
         .clickAppointmentTypeDropdown().chooseAppointmentType()
         .clickPickupStartDateCalendar().chooseNextMonthDate()
         .enterContactName(shipmentData.contactName)
         .enterPickupContactPhone(companyUserData.enterFirstPhoneNumber)
         .fillNotesForDriver(shipmentData.noteForDriver)
         .fillPickUpNumber(shipmentData.pickupNumber)
         .fillDropoffOneLocationname(shipmentData.dropoffLocation)
         .fillDropoffOneLocationAddress(shipmentData.dropoffAddress)
         .chooseDropoffAppointmentType().clickDropoffStartDateCalendar()
         .chooseNextMonthDate().fillTotalLoadInput(shipmentData.totalWeight)
.fillCommodityField(shipmentData.commodity).clickTruckTypeDropdown()
.verifyEquipmentTitle(shipmentData.equipmentModalTitle).chooseTruckType()
         .clickConfirmButton().clickContinueButton().clickBookThisLoadButton()
         .checkContactCarrierInfo(shipmentData.carrierName)
         cy.url().should('contain', 'details')});
it('Check dates validation', () => {
      cy.visit(Cypress.env('tenderPageUrl'));
shipmentBuildPage.clickBuildShipmentButton().chooseSendDirectTenderType()
      .chooseDropoffAppointmentType().fillIncorrectDropoffDate()
      .clickAppointmentTypeDropdown().chooseAppointmentType()
      .fillIncorrectPickupDate().checkDateError()})
```

Рисунок 3.12 – Код тестів для перевірки створення запиту на перечення вантажу

3.3 Приклад роботи автотестів

Після завершення написання системи автоматизованого тестування потрібно перевірити правильність її роботи. Запустити тести можна двома

способами: у “headed” режимі, тобто з графічним інтерфейсом, та у “headless” режимі, тобто з виводом логів та результатів виконання тестів у консоль.

Спочатку вводимо команду “npx cypress open” для запуску тестів у режимі з графічним інтерфейсом. Це надасть можливість подивитись у реальному часі як саме виконуються ті чи інші дії у браузері. Після запуску відкриється вікно (див. рис. 3.13), яке дозволить вибрати тести, які потрібно запустити, а також у якому браузері відбуватиметься тестування.

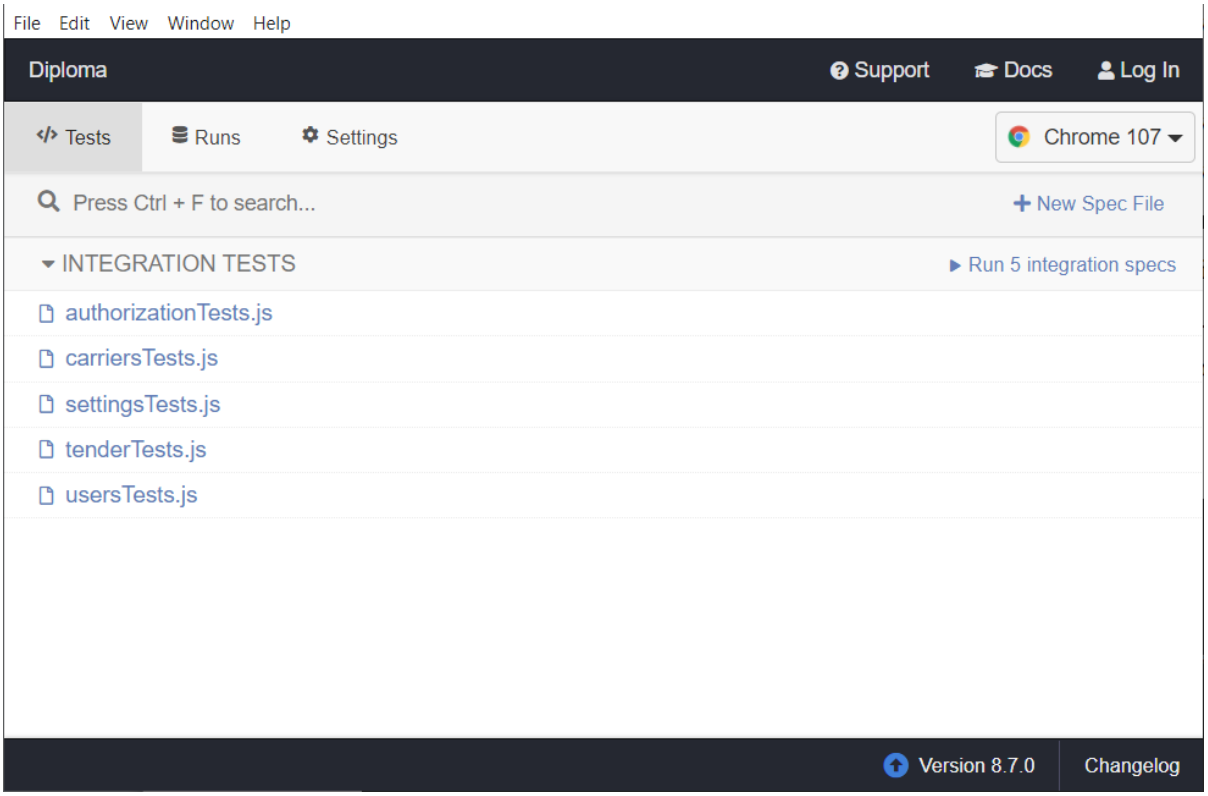

Рисунок 3.13 – Діалогове вікно для вибору браузеру та тестів

Вибираємо тести які потрібно запустити. Нехай це будуть усі тести, що стосуються тестування сторінки зі списком користувачів. Після вибору тестів відкриється тестовий браузер, в якому почнеться тестування обраного функціоналу (див. рис. 3.14). Екран браузера буде розділено на дві частини: в одній будуть показуватись дії, що виконуються в браузері, а в іншій будуть записуватись логи, тобто усі дії, що виконувались під час тестування. Після

закінчення перевірки усіх тестових випадків можна побачити результат, пройдено чи не пройдено, для кожного тестового випадку.

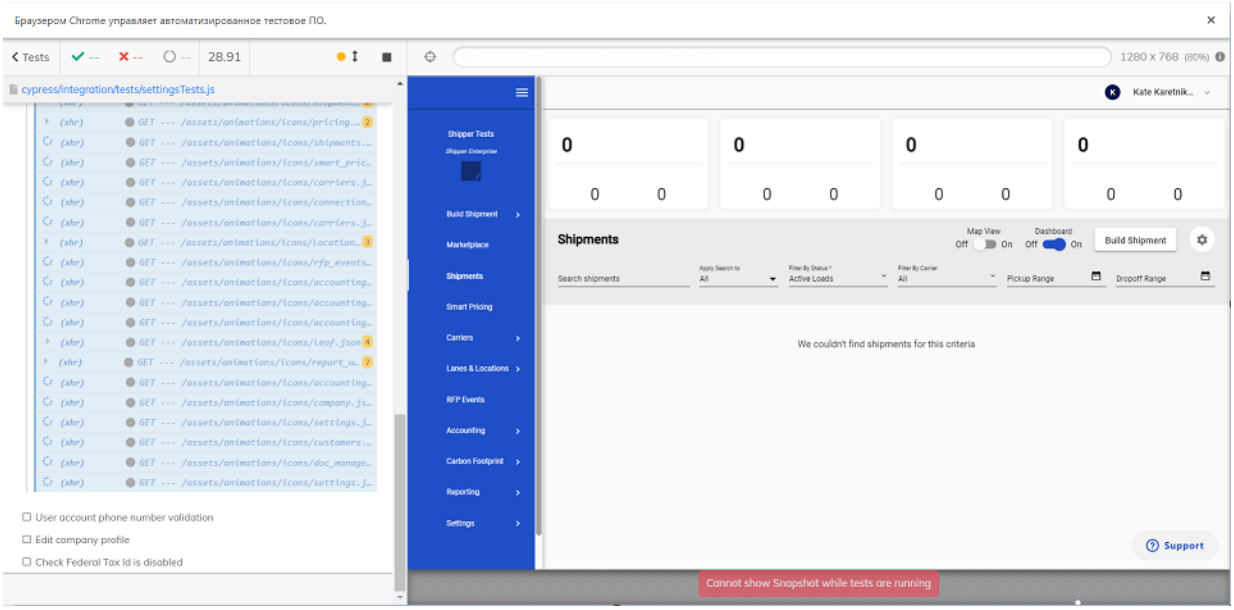

Рисунок 3.14 – Процес виконання тестів у тестовому браузері

Запускати тести у режимі з графічним інтерфейсом зручно коли тестувальнику потрібно спостерігати за процесом виконання тестів. Але якщо, наприклад, потрібно виконати запуск тестів за допомогою Jenkins або ж сгенерувати звіт allure, то такий спосіб не є оптимальним. У таких ситуаціях тести запускаються у “headless” режимі за допомогою команди npx cypress run. Після вводу команди у консолі з’явиться повідомлення, яке містить версію cypress, браузер, у якому буде проводитись тестування та список файлів, які будуть використані. Також під час тестування у консоль будуть виводитись логи тестування, які будуть інформувати про помилки, які відбулись під час запуску тестів, а також статус для кожного тестового випадку. Отже, запустимо тести за допомогою команди “npx cypress run --browser chrome” (див. рис. 3.15). Параметр --browser вказує на назву браузеру, у якому потрібно провести тестування.

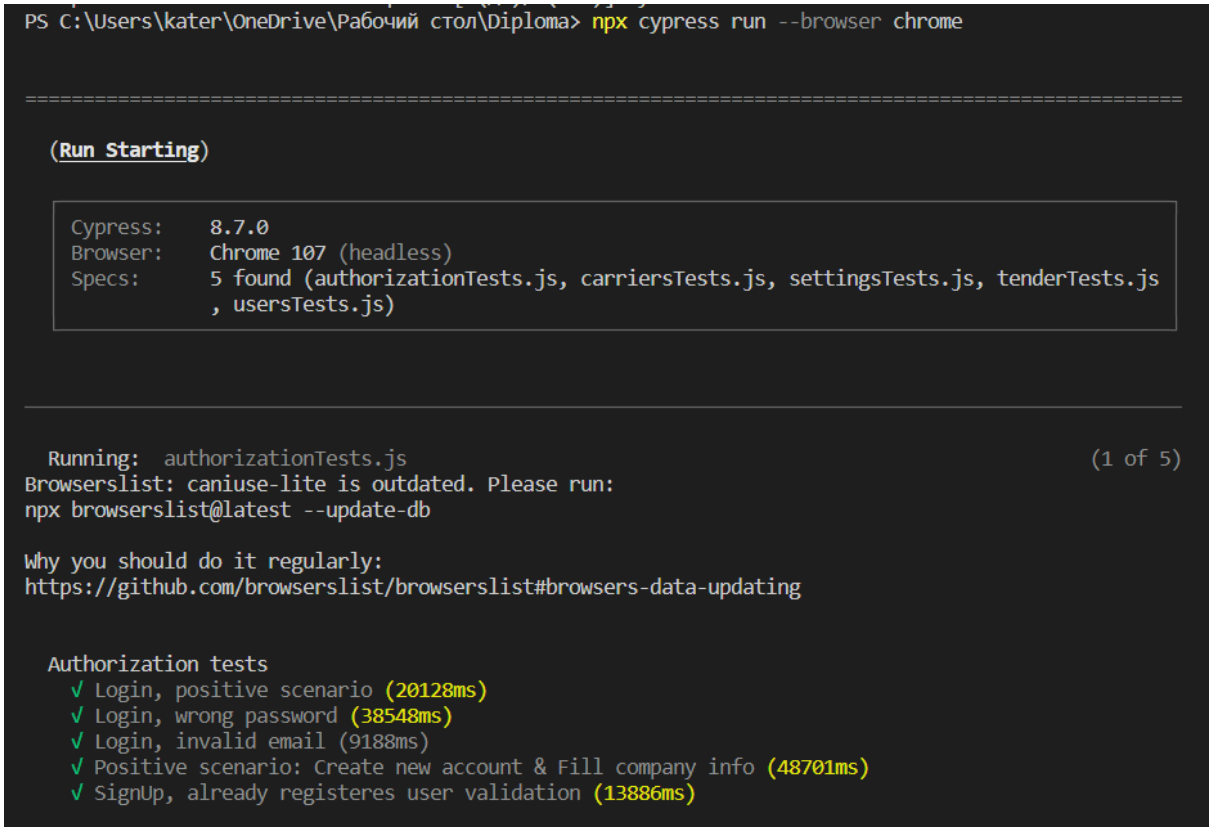

Рисунок 3.15 – Результат запуску тестів у “headless” режимі

Після того, як усі тести було виконано можна сгенерувати звіт allure (див. рис. 3.16) за допомогою команди allure serve. Звіт відображає процентне співвідношення пройдених тестів до не пройдених, кількість пройдених та не пройдених тестів у кожному тестовому наборі та графік, який показує статистику, співвідношення пройдених та не пройдених тестів, за певний період часу (див. рис. 3.16). Також звіт дозволяє передивитись логи для кожного тестового випадку, що є особливо зручним, коли, наприклад тестовий випадок не пройшов тестування. До того ж, якщо у ситуації, коли тестовий випадок не пройшов тестування звіт надає додатково скріншоти з тестового браузера.

Після генерації звіту allure можна побачити, що один тестовий випадок не пройшов тестування (див. рис. 3.16). Цей тестовий випадок мав перевірити правильність відображення помилки користувачу у випадку, якщо не було обрано компанію-перевізника. Передивившись логи тестування цього тестового випадку (див. рис. 3.17) можна зрозуміти, що система

автоматизованого тестування не знайшла елемент з класом .mat-snack-bat-container, тобто повідомлення про помилку, отже, тест не пройдено.

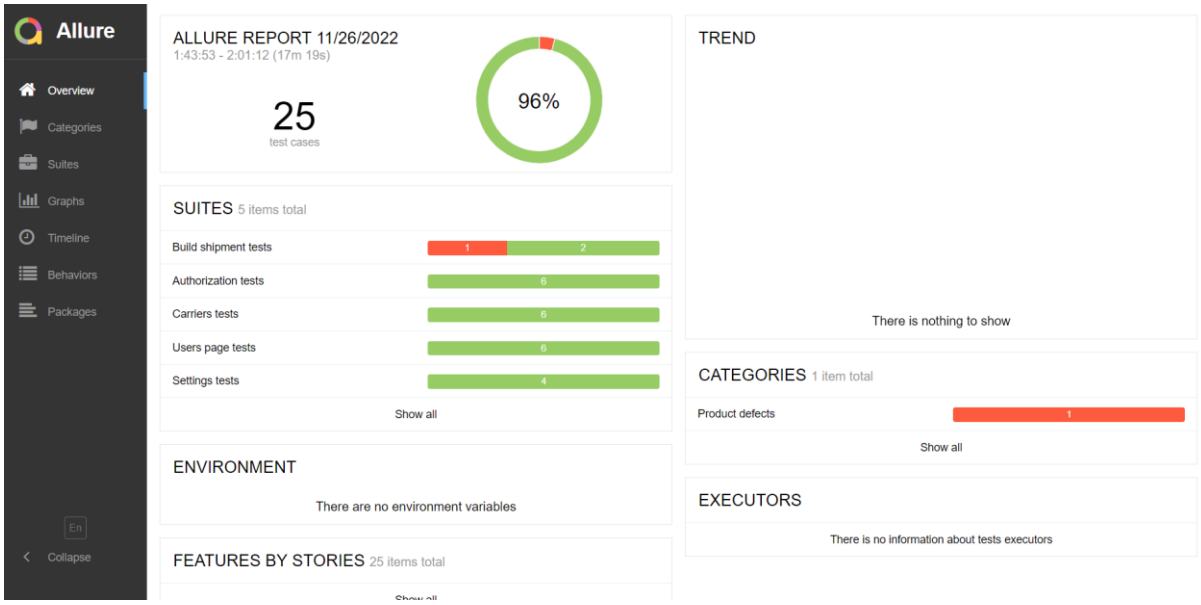

Рисунок 3.16 – Звіт Allure

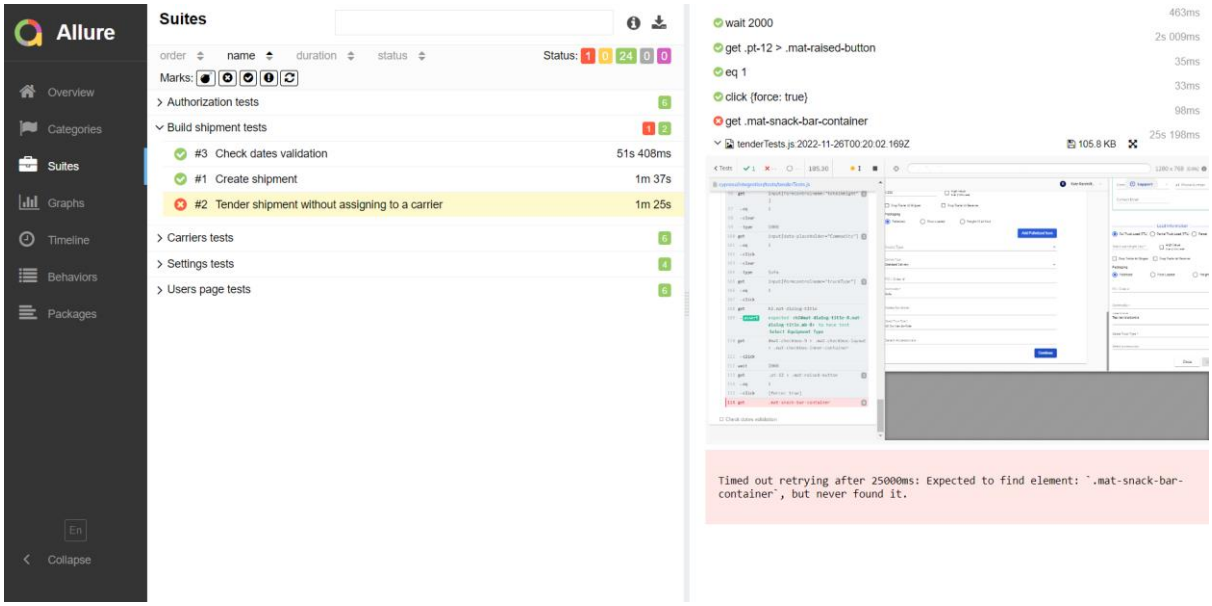

Рисунок 3.17 – Логи проходження тесту

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було створено систему автоматизованого тестування веб застосунку, що призначений для вантажовідправників, що шукають перевізників. Систему автоматизованого тестування було реалізовано засобами фреймворку Cypress, а для зручного відображення звіту про результати тестування було обрано бібліотеку Allure.

Під час виконання кваліфікаційної роботи було досліджено предметну область та вивчено вимоги то веб застосунку, що було обрано для тестування, розглянуто найбільш популярні засоби автоматизації веб застосунків та обрано найкращий, розроблено список тестових випадків, виконано проєктування системи автоматизованого тестування згідно з патерном Page Object Model та розроблено систему автоматизованого тестування.

Дана система може використовуватись для проведення регресійного тестування. Це означає, що спеціалістам не доведеться перевіряти вже раніше протестовані компоненти знову, а отже, це звільнить час для перевірки нових модулів. Згенерований звіт Allure надає статистику про пройдені та не пройдені тести у вигляді графіків, які можна застосувати для демонстрації команді або замовнику. Також завдяки логам та скріншотам, які присутні у звіті тестувальник швидко може з’ясувати у якому саме модулі є проблема.

За потреби дану систему можна інтегрувати у Jenkins та налаштувати розсилку на електрону пошту звіту Allure після кожного виконання тестів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Crispin L. More Agile Testing: Learning Journeys for the Whole Team (Addison-Wesley Signature Series (Cohn)), 2014. 544 p.

2. How to jumpstart a Test Automation Pyramid? URL: https://www.browserstack.com/guide/testing-pyramid-for-test-automation (дата звернення 02.11.2022).

3. Why Cypress? URL: https://docs.cypress.io/guides/overview/why-cypress#What-you-ll-learn (дата звернення 30.06.2022).

4. Copeland L. A Practitioner's Guide to Software Test Design, 2004. 312 p.

5. Angelov A. Design Patterns for High-Quality Automated Tests: High-Quality Test Attributes and Best Practices, 2020. 316 p.

6. Axerold A. Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects, 2018. 558 p.

7. Mwaura W. End-to-End Web Testing with Cypress: Explore techniques for automated frontend web testing with Cypress and JavaScript. Packt Publishing, 2021. 240 p.

8. Cypress-allure-plugin. URL: https://www.npmjs.com/package/@shelex/cypress-allure-plugin (дата звернення 15.08.2022).

ДОДАТОК А

Лістинг коду описаних тестів та класів

А.1 Код файлу login.page.js

```
export class ShipperTmsLogInPage {

    clickCreateAccountLink() {
        cy.get(loginPageLocators.createAnAccountLink).click();
        return this;
    }

    openLogInPage(){
        cy.visit(loginPageUrl);
        return this;
    }

    login(login, password) {
        cy.get(loginPageLocators.usernameInput).type(login);
        cy.get(loginPageLocators.passwordInput).type(password);
        cy.get(loginPageLocators.logInButton).click();
        return this;
    }

    fillEmailField(email) {

cy.get(loginPageLocators.usernameInput).type(email).blur();
        return this
    }

    checkError(errorMsg) {
        cy.get(loginPageLocators.errorBlock).should('have.text',
errorMsg)
        return this
    }
}
```

А.2 Код файлу settings.account.page.js

```
export class SettingsAccountPage {

    checkPhoneNumberValue(phoneNumber) {

cy.get(accountPageLocators.phoneInput).should('have.value',
phoneNumber)
        return this
        return this
    }

    checkCompanyProfileContent() {
        cy.get(accountPageLocators.tenantData)
            .should('contain',
resp.response.body.data.company.tenantId);
        return this;
    }

    checkMyProfileTenantId() {
        cy.wait(`@checkTenant`).then((resp) => {
            cy.get(accountPageLocators.tenantData)
                .should('contain',
resp.response.body.data.company.tenantId);
        });
        return this;
    }
```

```
  clickStateDropdown() {
    cy.get(accountPageLocators.stateDropdown)
      .click();
    return this;
  }

  checkAccountData() {
    cy.scrollTo('bottom', { ensureScrollable: false })
    cy.get(accountPageLocators.accountData)
      .should('exist');
    return this;
  }

  checkAddressData() {
    cy.get(accountPageLocators.addressData)
      .should('contain', 'Country')
      .and('contain', 'State')
      .and('contain', 'Zipcode');
    return this;
  }

  clickCompanyProfileSubButton() {
    cy.get(accountPageLocators.globalSettingsSideMenuTab)
      .contains('Company Profile')
      .click();
    return this;
  }

  checkNaNForTotalAmount() {

cy.get(accountPageLocators.totalAmountCard).should('not.cont
ain', NaN);
    return this;
  }

  checkNaNForPrices() {
    cy.get(accountPageLocators.priceCard).each(() => {

cy.get(accountPageLocators.priceCard).should('not.contain',
NaN);
    })
    return this;
  }

  clickUploadImageButton() {
    cy.get(accountPageLocators.uploadProfileImageButton)
      .attachFile('src/qa-2-min.png');
    return this;
  }

  fillPhoneInput(enterPhone) {
    cy.get(accountPageLocators.phoneInput)
      .click()
      .clear()
      .type(enterPhone)
    return this;
  }

  clickSaveChangesButton() {
    cy.get(accountPageLocators.saveChangesButton).click();
    return this;
  }

}
```

A.3 Код файлу shipments.build.page.js

```
export class ShipmentsBuildPage {
clickBuildShipmentButton() {
    cy.wait(2000)

cy.get(shipmentsBuildPageLocators.buildShipmentButton).click
();
    return this;
  }

chooseSendDirectTenderType() {

cy.get(shipmentsBuildPageLocators.shipmentDropdownOption)
.eq(2).click();
    return this;
  }
```

```
clickSelectOtherCarrierButton() {

cy.get(shipmentsBuildPageLocators.selectOtherCarrierButton).c
lick();
      return this;
   }

fillSearchCarrierInput(enterCarrier) {
cy.get(carrierTenderModalLocators.searchInput)
         .click()
         .clear()
         .type(enterCarrier);
      return this;
   }

chooseAccountFromDropdown() {
      cy.get(carrierTenderModalLocators.carrierOption)
         .wait(1000)
         .should('contain', 'QA Carrier')
         .eq(0)
         .click();
      return this;
   }

fillCarrierPrice(enterCarrierPrice) {
      cy.get(carrierTenderModalLocators.carrierPriceinput)
         .click()
         .clear()
         .type(enterCarrierPrice)
      return this;
   }

clickCarrierManagerDropdown() {

cy.get(carrierTenderModalLocators.carrierManagerDropdown)
         .click({force: true});
      return this;
   }

chooseCarrierManager() {
      cy.wait(5000)
      cy.get(carrierTenderModalLocators.carrierOption)
         .click({force: true})
      return this;
   }

clickSubmitTenderButton() {

cy.get(shipmentsBuildPageLocators.submitTenderButton).click(
);
      return this;
   }

fillPickupOneLocationName(pickupLocationName) {
      cy.wait(2000)
      cy.get(shipmentsBuildPageLocators.locationNameInput)
         .eq(2)
         .click()
         .clear()
         .type(pickupLocationName);
      return this;
   }

fillPickupLocationOneAddress(pickupLocationAddress) {
      cy.get(shipmentsBuildPageLocators.locationAddressInput)
         .eq(2)
         .click()
         .clear()
         .type(pickupLocationAddress)
         .wait(3000)
         .type('{downarrow}')
         .wait(2000)
         .click();
      return this;
   }

clickAppointmentTypeDropdown() {

cy.get(shipmentsBuildPageLocators.appointmentTypeDropdow
n).eq(0).click();
      return this;
   }

chooseAppointmentType() {

cy.get(shipmentsBuildPageLocators.dropdownTypeOption).eq(
0).click();
      return this;
   }

clickPickupStartDateCalendar() {
      cy.get(shipmentsBuildPageLocators.calendarInput)
         .eq(4)
         .click()
      return this;
   }

chooseNextMonthDate() {

cy.get(shipmentsBuildPageLocators.nextMonthButton).click();
```

```
cy.get(shipmentsBuildPageLocators.calendarDate).eq(0).click({
force: true});
        return this;
    }

fillPickUpNumber(enterPickupNumber) {
        cy.get(shipmentsBuildPageLocators.pickUpNumberInput)
            .eq(1)
            .click()
            .clear()
            .type(enterPickupNumber)
        return this;
    }

fillDropoffOneLocationname(enterDropoffName) {
        cy.get(shipmentsBuildPageLocators.locationNameInput)
            .eq(3)
            .click()
            .clear()
            .type(enterDropoffName)
        return this;
    }

fillDropoffOneLocationAddress(enterDropoffAddress) {
        cy.get(shipmentsBuildPageLocators.locationAddressInput)
            .last()
            .click()
            .clear()
            .type(enterDropoffAddress)
            .wait(1000)
            .type('{downarrow}')
            .click();
        return this;
    }

verifyEquipmentTitle(checkEquipmentTitle) {
        cy.get(loadInformationLocators.modalEquipmentTitle)
            .should('text', checkEquipmentTitle);
        return this;
    }

chooseTruckType() {
        cy.get(loadInformationLocators.truckCheckbox).click();
        return this;
    }

clickConfirmButton() {
        cy.wait(2000)
        cy.get(shipmentsBuildPageLocators.confirmButton)
            .eq(1)
            .click({force: true});
        return this;
    }

clickContinueButton() {
        cy.wait(2000)

cy.get(shipmentsBuildPageLocators.continueButton).click();
        return this;
    }


}
```

A.4 Код файлу authorizationTests.js

```
describe('Authorization tests', () => {

    const userHelper = new UserHelper();
    const dashboardPage = new DashboardPage();
    const signUpPage = new SignUpPage();
    const shipmentBuildPage = new ShipmentsBuildPage();
    const loginPage = new ShipperTmsLogInPage();

    it('Login, positive scenario', () => {
        cy.clearCookies();
        cy.clearLocalStorage();
        cy.visit(loginPageUrl)
        cy.intercept('**').as('all')
        loginPage.login(Cypress.env("userEmail"),
Cypress.env("userPass"))
        cy.url().should('contain', dashboardPageUrl);
            cy.title().should('eq', 'Zuum Shipper');
            cy.wait('@all').its('response.statusCode')
                .should('be.oneOf', [200,201,300])
                .and('not.be.oneOf', [500,400]);
    })
```

```
    it("Login, wrong password", () => {
        cy.clearCookies();
        cy.clearLocalStorage();
        cy.visit(loginPageUrl)
        loginPage.login(Cypress.env("userEmail"),
Cypress.env("invalidPassword"))
        .checkError('Invalid password.')
    })

    it('Login, invalid email', () => {
        cy.clearCookies();
        cy.clearLocalStorage();
        cy.visit(loginPageUrl)
        loginPage.fillEmailField('kate@')
        .checkError(' Please enter a valid email address ')
    })

    it('Positive scenario: Create new account & Fill company
info', () => {
        cy.clearCookies();
        cy.clearLocalStorage();

        cy.visit(signUpPageUrl)
        cy.url().should('contain', signUpPageUrl);

        cy.intercept('**').as('all')

        signUpPage.checkTitle(signUpData.pageTitle)
            .fillEmailInput(signUpData.enterEmail)

.fillCompanyNameInput(signUpData.enterCompanyName)
            .fillFirstNameInput(signUpData.enterFirstUserName)
            .fillLastNameInput(signUpData.enterLastUserName)
            .fillPhoneNumber(signUpData.enterPhoneNumber)
            .fillPasswordInput(signUpData.enterPassword)
            .fillConfirmPasswordInput(signUpData.enterPassword)
            .clickTermsAndConditionsCheckbox()
        cy.wait(2000)
        cy.wait('@all').its('response.statusCode')
            .should('be.oneOf', [200,201,300])
            .and('not.be.oneOf', [500,400]);

        signUpPage.clickNextButton();

signUpPage.fillCompanyAddressInput(signUpData.companyAd
dress)
            .fillCompanyCityInput(signUpData.city)
            .fillZipcodeInput(signUpData.zipCode)
            .clickStateDropdown()
            .chooseDropdownOption()

.fillCompanyPhoneNumber(signUpData.enterPhoneNumber)
            .fillFederalTaxId(signUpData.federalId)
            .clickCompleteButton()
            .checkSignUpTitleText(signUpData.signUpTitle)
            .clickLoginButton();

            cy.url().should('contain', dashboardPageUrl);

            dashboardPage.clickShipmentsSideButton();

            cy.wait(3000);

        shipmentBuildPage.clickBuildShipmentButton()
        });
    });

    it('SignUp, already registeres user validation', () => {
        cy.clearCookies();
        cy.clearLocalStorage();
        cy.visit(signUpPageUrl)
        signUpPage.fillEmailInput(Cypress.env("userEmail"))
        .checkAlreadyRegistered()
        .clickAlreadyRegisteredCancelButton()
        .checkError('Already registered ')
    })

    it('SignUp, check email validation', () => {
        cy.clearCookies();
        cy.clearLocalStorage();
        cy.visit(signUpPageUrl)
        signUpPage.fillEmailInput('kate.gmail.com')
        .fillCompanyNameInput(signUpData.enterCompanyName)
        .checkError('Email format not valid')
    })
})
```

А.5 Код файлу tenderTests.js

```
describe('Build shipment', function() {
    const dashboardPage = new DashboardPage();
    const shipmentBuildPage = new ShipmentsBuildPage();
    const userHelper = new UserHelper();

    it('Positive scenario: Create build shipment with tender -
choose carrier from search', function() {
        dashboardPage.openDashboardPage();

        cy.title().should('eq', 'Zuum Shipper');

        userHelper.verifyUserLoggedinShipperTms();

        dashboardPage.clickShipmentsSideButton();

        shipmentBuildPage.clickBuildShipmentButton()
            .chooseSendDirectTenderType()
            .clickSelectOtherCarrierButton()
            .fillSearchCarrierInput(shipmentData.carrierName)
            .chooseAccountFromDropdown()
            .fillCarrierPrice(shipmentData.carrierPrice)
            .clickCarrierManagerDropdown()
            .chooseCarrierManager()
            .clickSubmitTenderButton()

.fillPickupOneLocationName(shipmentData.pickupLocation)

.fillPickupLocationOneAddress(shipmentData.pickupAddress)
            .clickAppointmentTypeDropdown()
            .chooseAppointmentType()
            .clickPickupStartDateCalendar()
            .chooseNextMonthDate()

            .fillPickUpNumber(shipmentData.pickupNumber)

.fillDropoffOneLocationname(shipmentData.dropoffLocation)

.fillDropoffOneLocationAddress(shipmentData.dropoffAddress)
            .chooseDropoffAppointmentType()
            .clickDropoffStartDateCalendar()
            .chooseNextMonthDate()
            .fillTotalLoadInput(shipmentData.totalWeight)
            .fillCommodityField(shipmentData.commodity)
            .clickTruckTypeDropdown()

.verifyEquipmentTitle(shipmentData.equipmentModalTitle)
            .chooseTruckType()
            .clickConfirmButton()
            .clickContinueButton()
            .clickBookThisLoadButton()
            .checkContactCarrierInfo(shipmentData.carrierName)

            cy.url().should('contain', 'details')

    });

    it('Positive scenario: Create build shipment with tender -
choose carrier from search', function() {
        dashboardPage.openDashboardPage();

        cy.title().should('eq', 'Zuum Shipper');

        userHelper.verifyUserLoggedinShipperTms();

        dashboardPage.clickShipmentsSideButton();

        shipmentBuildPage.clickBuildShipmentButton()
            .chooseSendDirectTenderType()

.fillPickupOneLocationName(shipmentData.pickupLocation)

.fillPickupLocationOneAddress(shipmentData.pickupAddress)
            .clickAppointmentTypeDropdown()
            .chooseAppointmentType()
            .clickPickupStartDateCalendar()
            .chooseNextMonthDate()
            .enterContactName(shipmentData.contactName)

.enterPickupContactPhone(companyUserData.enterFirstPhoneN
umber)
            .fillNotesForDriver(shipmentData.noteForDriver)
            .fillPickUpNumber(shipmentData.pickupNumber)

.fillDropoffOneLocationname(shipmentData.dropoffLocation)

.fillDropoffOneLocationAddress(shipmentData.dropoffAddress)
            .chooseDropoffAppointmentType()
            .clickDropoffStartDateCalendar()
            .chooseNextMonthDate()
            .fillTotalLoadInput(shipmentData.totalWeight)
            .fillCommodityField(shipmentData.commodity)
            .clickTruckTypeDropdown()

.verifyEquipmentTitle(shipmentData.equipmentModalTitle)
```

```
        .chooseTruckType()
        .clickConfirmButton()
        .checkSuccessMessage('Who do you want to send tender
to?')
  });

  it('Check dates validation', () => {
    dashboardPage.openDashboardPage();

    cy.title().should('eq', 'Zuum Shipper');

    userHelper.verifyUserLoggedinShipperTms();

    dashboardPage.clickShipmentsSideButton();

    shipmentBuildPage.clickBuildShipmentButton()
    .chooseSendDirectTenderType()
    .chooseDropoffAppointmentType()
    .fillIncorrectDropoffDate()
    .clickAppointmentTypeDropdown()
    .chooseAppointmentType()
    .fillIncorrectPickupDate()
    .checkDateError()
  })
});
```

A.6 Код файлу carriersTests.js

```
describe('Carriers tests', function() {
  const dashboardPage = new DashboardPage();
  const carrierPage = new CarrierPage();
  const newCarrierPage = new NewCarrierPage();
  const userHelper = new UserHelper();
  const existingCarrierPage = new ExistingCarrierPage();
  const deletePreviouslyAddedCarrier = new
DeletePreviouslyAddedCarrier();

  it('Positive scenario: Add a carrier to the system', function() {
    dashboardPage.openDashboardPage();

    cy.title().should('eq', 'Zuum Shipper');

    userHelper.verifyUserLoggedinShipperTms();
    dashboardPage.clickCarriesDropdown()
      .clickCarriersBrokersButton();
    carrierPage.clickAddCarrierButton()
      .chooseAddCarrierMethod();

newCarrierPage.fillCompanyName(carrierData.companyName)
      .fillCompanyPhone(carrierData.companyPhone)
      .fillExt(carrierData.extField)
      .fillCompanyEmail(carrierData.companyEmail)
      .fillDotNumberInput(carrierData.dotNumber)
      .fillFirstNameInput(carrierData.firstName)
      .fillLastNameInput(carrierData.lastName)
      .fillPhoneInput(carrierData.phone)
      .fillEnterEmailInput(carrierData.email)
      .fillAddressInput(carrierData.address)
      .fillCityInput(carrierData.city)
      .clickStateDropdown()
      .chooseState()
      .fillZipCodeInput(carrierData.zipCode)
      .clickCreateButton()
      .clickDontSendEmailButton()

    carrierPage.checkTableRow()
      .openExistingCarrier(carrierData.companyName);
  });
  it('Positive scenario: Trying to add already existing carrier',
function() {
    dashboardPage.openDashboardPage();

    cy.title().should('eq', 'Zuum Shipper');

    userHelper.verifyUserLoggedinShipperTms();

    dashboardPage.clickCarriesDropdown()
      .clickCarriersBrokersButton();

    cy.url().should('contain', 'carriers');

    carrierPage.clickAddCarrierButton()
      .chooseAddCarrierMethod();


newCarrierPage.fillCompanyName(carrierData.companyName)
      .fillCompanyPhone(carrierData.companyPhone)
      .fillExt(carrierData.extField)
```

```
        .fillCompanyEmail(carrierData.companyEmail)
        .fillDotNumberInput(carrierData.dotNumber)
        .fillFirstNameInput(carrierData.firstName)
        .fillLastNameInput(carrierData.lastName)
        .fillPhoneInput(carrierData.phone)
        .fillEnterEmailInput(carrierData.email)
        .fillAddressInput(carrierData.address)
        .fillCityInput(carrierData.city)
        .clickStateDropdown()
        .chooseState()
        .fillZipCodeInput(carrierData.zipCode)
        .checkEmailError()
  });
  it('Positive scenario: Edit existing carrier', function() {
    dashboardPage.openDashboardPage();

    cy.title().should('eq', 'Zuum Shipper');

    userHelper.verifyUserLoggedinShipperTms();

    dashboardPage.clickCarriesDropdown()
      .clickCarriersBrokersButton();

    carrierPage.openExistingCarrier();


newCarrierPage.fillCompanyEmail(carrierData.companyEmail)
        .clickSaveChangesButton()
        .checkSuccessUpdate(carrierData.successUpdated)
  });
  it('Positive scenario: Delete existing carrier', function() {
    dashboardPage.openDashboardPage();

    cy.title().should('eq', 'Zuum Shipper');

    userHelper.verifyUserLoggedinShipperTms();

    dashboardPage.clickCarriesDropdown()
      .clickCarriersBrokersButton();

    carrierPage.openExistingCarrier()

    cy.url().should('contain', 'pro');

    newCarrierPage.clickRemoveButton()
      .checkRemoveCarrierModal(carrierData.removeCarrier)
      .clickConfirmRemoveButton()

.checkSuccessRemovedMessage(carrierData.successRemove)
  });
  it('Positive scenario: Search carrier by parameters', function()
{
    dashboardPage.openDashboardPage();

    cy.title().should('eq', 'Zuum Shipper');

    userHelper.verifyUserLoggedinShipperTms();

    dashboardPage.clickCarriesDropdown()
      .clickCarriersBrokersButton();

    cy.wait(3000);

    cy.url().should('contain', 'carriers');

    existingCarrierPage.checkTableRow()
      .fillSearchCarrierInput(carrierData.existingCarrier)
      .checkTableRow()
      .searchByName()
      .checkCarrierSearchResult()
      .searchByPrimaryContact()
      .checkCarrierSearchResult()
  });
  it('Check pagination in carriers list', function() {
    dashboardPage.openDashboardPage();

    cy.title().should('eq', 'Zuum Shipper');

    userHelper.verifyUserLoggedinShipperTms();

    dashboardPage.clickCarriesDropdown()
      .clickCarriersBrokersButton();

    cy.wait(3000);

    cy.url().should('contain', 'carriers');

    existingCarrierPage.clickPaginationDropdown()
      .choosePaginationSize(dashboardData[50])
      .checkPaginationTableResult(dashboardData.size50)
      .clickPaginationDropdown()
      .choosePaginationSize(dashboardData[25])
      .checkPaginationTableResult(dashboardData.size25)
  });
});
```