МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ВИЗНАЧЕННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАСОБАМИ CYPRESS»**

Виконав: студент 2 курсу, групи 8.1211-іпз
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
Д.В. Чузов
(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 фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(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 СТУДЕНТОВІ

Чузову Дмитру Вячеславовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інформаційної системи визначення якості програмного забезпечення засобами Cypress

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. Основні теоретичні відомості.
3. Проєктування інформаційглї системи

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.	20.05.2022	
2.	Обробка теоретичних джерел.	01.06.2022	
3.	Ознайомлення з засобами.	08.07.2022	
	автоматизованого тестування.		
4.	Розробка першого розділу.	26.07.2022	
5.	Ознайомлення з обраним вебдодатком.	12.08.2022	
6.	Написання набору тестових випадків.	26.08.2022	
7.	Проєктування інформаційної системи.	17.09.2022	
8.	Розробка другого розділу.	26.09.2022	
9.	Написання коду автотестів.	07.10.2022	
10.	Розробка третього розділу.	15.11.2022	
11.	Оформлення і нормоконтроль.	24.11.2022	
12.	Захист кваліфікаційної роботи.	16.12.2022	

Студент _____ Д.В. Чузов
(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 «Розробка інформаційної системи визначення якості програмного забезпечення засобами Cypress»: 64 с., 43 рис., 7 джерел, 1 додаток.

АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ, ВЕБДОДАТОК, ІНФОРМАЦІЙНІ СИСТЕМИ, КОНТРОЛЬ ЯКОСТІ, САЙПРЕС.

Об’єкт дослідження – процес тестування веб застосунків.

Предмет дослідження – процес визначення якості програмного забезпечення.

Мета роботи: розробка інформаційної системи визначення якості програмного забезпечення.

Метод дослідження: аналіз, моделювання, порівняння, проєктування.

У кваліфікаційній роботі магістра наведено огляд сучасних систем автоматизації тестування вебдодатків, проведено аналіз вимог до сайту, на основі тестових випадків для контролю якості веб-додатку розроблено набір автотестів для перевірки функціоналу сайту, створено інформаційну систему визначення якості програмного забезпечення з автоматичним запуском тестів та генерацією звітів.

SUMMARY

Master`s Qualifying Paper «Development of the Software Quality Evaluation Information System using Cypress»: 64 pages, 43 figures, 7 references, 1 supplement.

AUTOMATED TESTING, WEB APPLICATION, INFORMATION SYSTEMS, QUALITY ASSURANCE, CYPRESS.

The object of the study – the process of testing web application.

The subject of the study – the information system for the software quality evaluation.

The aim of the study – development of the software quality evaluation information system.

The methods of research are analysis, modeling, comparison, design.

In the Master`s Qualification Thesis an overview of modern web application testing automation systems was provided, an analysis of site requirements was carried out, a set of autotests was developed based on test cases for web application quality control to check the functionality of the site, an information system for evaluating software quality was created with automatic test launch and report generation.

ЗМІСТ

ВСТУП

Розробка сучасного програмного забезпечення це складний процес, в якому приймає участь велика кількість фахівців, різних команд чи навіть різних компаній. А чим більше людей приймає участь у розробці, тим сильніше відчувається вплив людського фактору на кінцевий результат. Будь то клієнт, який постійно змінює свої рішення та бажання, змушуючи команду переробляти ключову логіку проекту буквально на льоту, або різні бачення реалізації кінцевого результату між командами, що працюють над різними частинами системи. Все це може призвести до різноманіття помилок в роботі системи, або невідповідності кінцевого результату з бажаннями замовника. А тому не дивно, що в таких умовах компанії готові виділяти на тестування та забезпечення якості аж до половини ресурсів та часу що відводяться на сам проект.

Реалізація нового функціоналу супроводжується спочатку димовим тестуванням, функціональним тестуванням, а потім і регресивним тестуванням, щоб переконатись що новий функціонал не вплинув на роботу все існуючих частин системи. Ймовірно цей цикл доведеться пройти декілька раз, адже майже неможливо реалізувати складний функціонал без непередбачуваних помилок. А крім цього за необхідністю може бути треба провести й інші типи тестування, наприклад перевірити такі, здавалося б, банальні речі, як юзабіліті чи локалізацію, або більш складні, наприклад продуктивність системи чи безпеку.

В таких умовах гостро стає питання як оптимізувати роботу тестувальників, яким часто доводиться перевіряти одні й ті самі компоненти системи, щоб не вийти за рамки часу відведенного на тестування, але в той же час не допустити наявність помилок в системі. Тому рішенням цієї проблеми все частіше стає використання засобів автоматизованого контролю якості програмного забезпечення.

Системи автоматизованого тестування покликані мінімізувати витрати часу тестувальників на перевірку вже існуючих компонентів системи, адже дії, виконувані людиною, можна імітувати, використовуючи засоби автоматизації.

В цій роботі будуть розглянуті інструменти та засоби автоматизації тестування вебдодатків та їх використання при написання ряду автотестів для створення інформаційної системи автоматизованого контролю якості програмного забезпечення, а саме вебсайту зі ставками на спорт.

Об’єктом дослідження є процес визначення якості програмного забезпечення.

Предметом дослідження є інформаційна система визначення якості програмного забезпечення.

Для досягнення поставленої мети визначено такі завдання: дослідити предметну область обраного вебдодатку, розробити набір тест кейсів для тестування функціоналу системи, розглянути засоби та інструменти автоматизації тестування вебдодатків, реалізувати набір автотестів засобами обраної бібліотеки та реалізувати систему автоматичного запуску автотестів при змінах в код вебдодатку.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Рівні автоматизованого тестування

Рівні тестування, на яких використовується автоматизація процесу, можна розділити на рівні за допомогою піраміди, запропонованої Майком Коном [1]. Згідно цієї піраміди основну частину всіх автотестів складають юніт тести, які найчастіше створюються програмістами в ході створення нового коду. Вони перевіряють коректність роботи окремих функції та методів. А наступні два рівня, інтеграційне тестування, або API тестування, та E2E тестування – це саме те з чим найчастіше доводиться працювати спеціалістам з автоматизації тестування ПО.

Юніт тестування – процес у програмуванні, що дозволяє перевірити на коректність окремі модулі вихідного коду програми. Ідея полягає в тому, щоб писати тести для кожної нетривіальної функції чи методу. Це дозволяє досить швидко перевірити, чи не призвела чергова зміна коду до регресії, тобто до появи помилок у вже відтестованих місцях програми, а також полегшує виявлення та усунення таких помилок.

Інтеграційне тестування – це фаза тестування програмного забезпечення, під час якої окремі модулі програми комбінуються та тестуються разом, у взаємодії. Інтеграційне тестування виконується після модульного тестування та перед верифікацією та валідацією ПЗ. При тестуванні вебдодатків в цю фазу відносять тестування API, або серверною частини додатку. Перевіряють всі можливі маршрути та запити, звіряють отримані результати, та результат їх виконання на інші частини системи, наприклад записи в базу даних.

E2E тестування – це техніка, яка перевіряє весь програмний продукт від початку до кінця, щоб переконатися, що бізнес-логіка програми працює належним чином. Основною метою наскрізного тестування (E2E) є

тестування на досвіді кінцевого користувача шляхом моделювання сценарію реального користувача використовуючи засоби графічного інтерфейсу готового продукту.

З точки зору QA-інженера створення юніт-тестів не є ключовою задачею, адже це потребує знання проекту з точки зору коду, розуміння роботи функцій та методів, а тому і створення таких тестів є відповідальністю розробника.

З іншого боку, API-тестування, а тим паче E2E тестування, – це саме задача спеціаліста з тестування ПО. Отже, в даній роботі будемо спиратися саме на створення такого ряду тестів.

1.2 Дослідження функціоналу сайту dublzplay.com

Для тестування інформаційної системи було обрано сайт dublzplay.com. Це типовий представник сайту зі ставками на спорт. Він надає користувачу функціонал для отримання ігрової валюти – дублонів, які можна використати для ставок на ігрові події: футбольні, баскетбольні, тенісні матчі тощо. Якщо описати всі компоненти системи, то можна виділити наступний перелік функціональних блоків, тестування яких можна автоматизувати:

– авторизація;

– налаштування облікового запису;

– отримання ігрової валюти;

– можливість зробити ставку.

Розглянемо нижче кожен з цих блоків детальніше.

Авторизація. У користувача є можливість створити новий акаунт, вказавши електронну адресу, дату народження та пароль, а також вигадав собі псевдонім, а потім авторизуватись в системі використовуючи ці дані. Приклад сторінки для реєстрації на сайті можна побачити на рисунку 1.1.

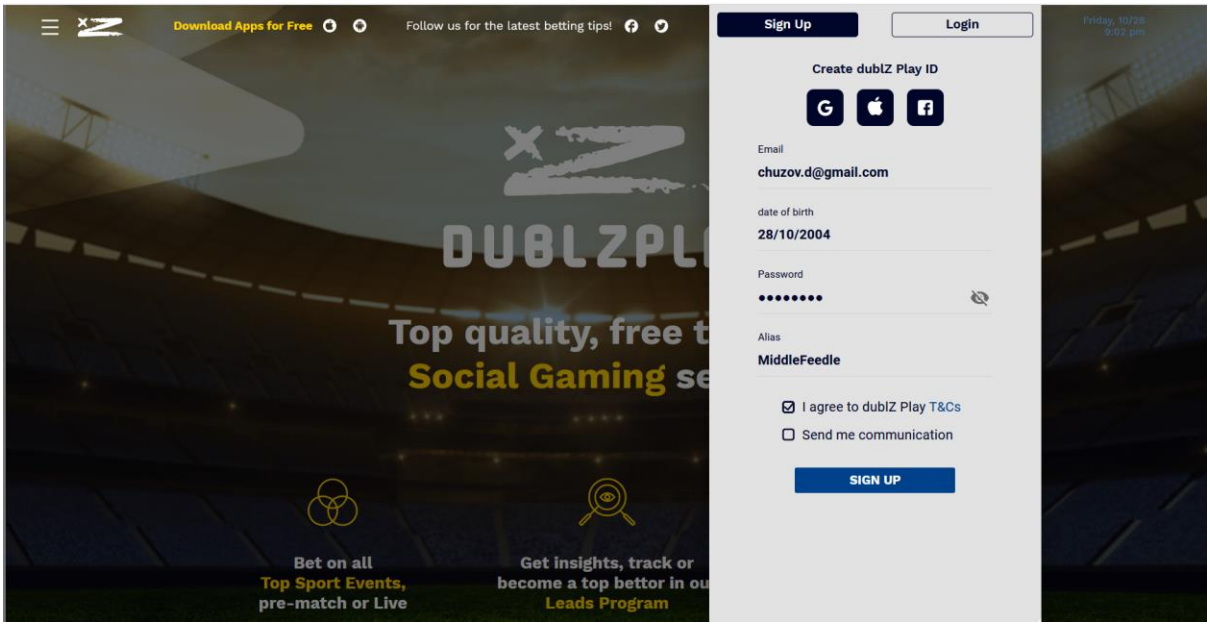

Рисунок 1.1 – Сторінка для реєстрації на сайті

Налаштування облікового запису. Після реєстрації користувач може редагувати свій обліковий запис, наприклад змінювати псевдонім чи пароль, або навіть зовсім видалити свій профіль (рис. 1.2).

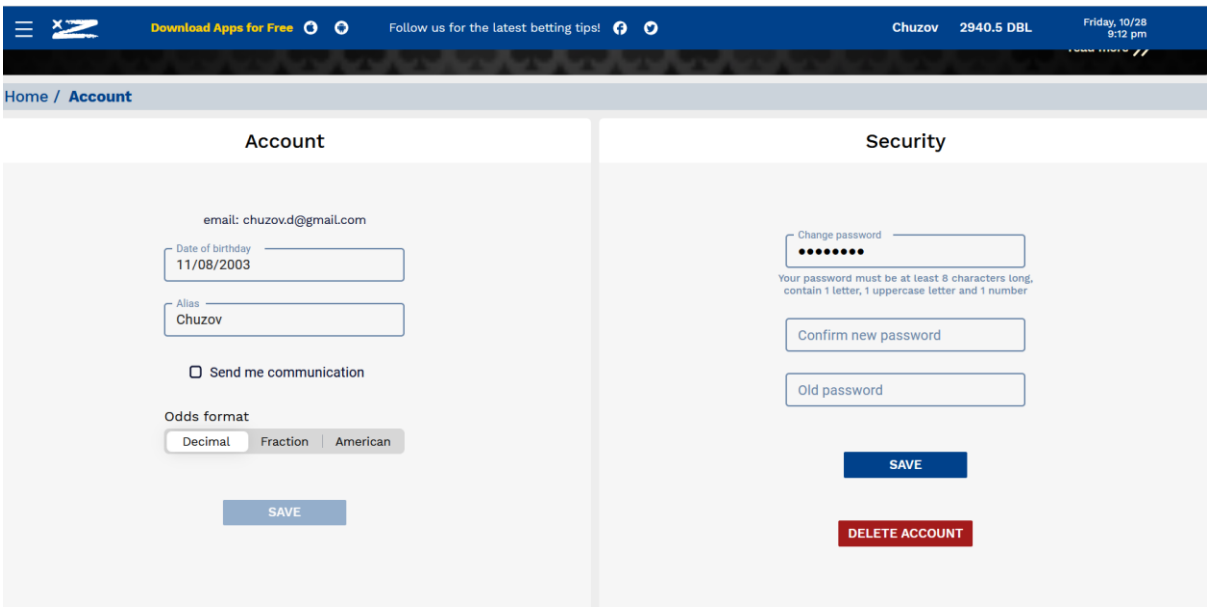

Рисунок 1.2 – Сторінка для редагування облікового запису користувача

Отримання ігрової валюти. Для дублонів користувачу надається можливість раз в день прийняти участь у міні-грі “Lucky Roll”. В залежності від результату користувач отримує певну кількість валюти, яку зможе використати для ставок (рис 1.3).

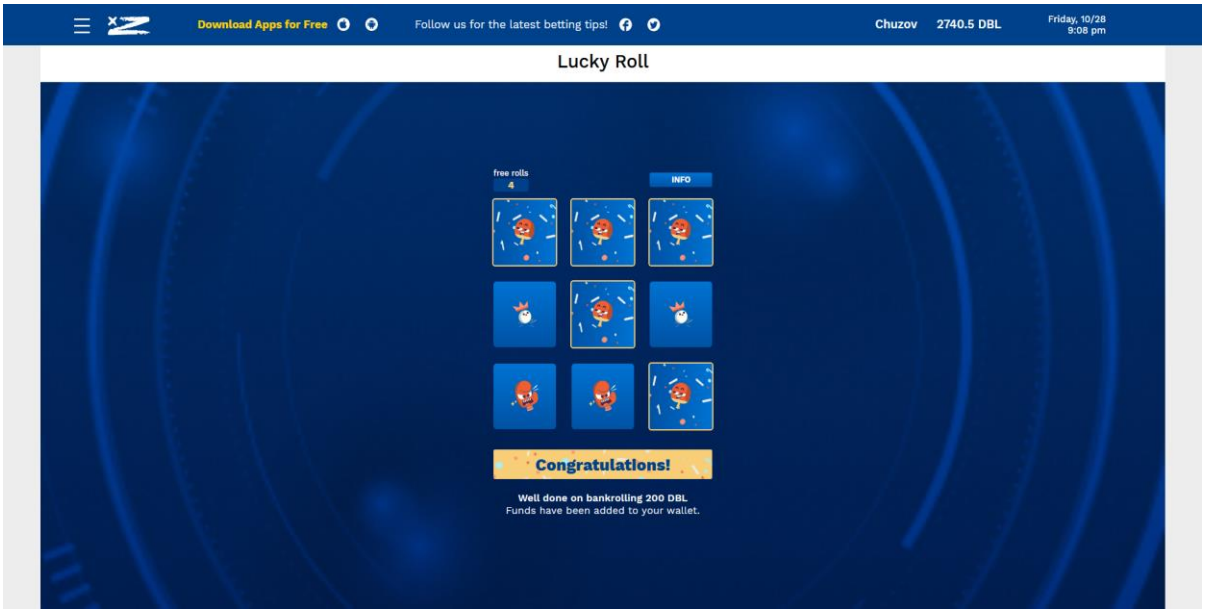

Рисунок 1.3 – Повідомлення про виграш і отримання ігрової валюти

Можливість зробити ставку. Основний функціонал сайту це можливість робити ставки на певні спортивні події. На відповідній сторінці (рис. 1.4) користувачу надається широкий вибір з матчів та список певних подій до кожного з них, на які саме можна поставити. Наприклад якщо мова йдеться про футбольний матч, то поставити можна на перемогу певної команди, кількість голів у матчі тощо. Кожна ігрова подія, в залежності від вірогідності збутися, має свій коефіцієнт, від котрого залежить можливий виграш. Для однієї ставки можна обрати до десяти ігрових подій одночасно, тоді на можливий виграш впливає можен з коефіцієнтів обраних подій, але якщо хоча б одна подія не трапиться, то і вся ставка програє.

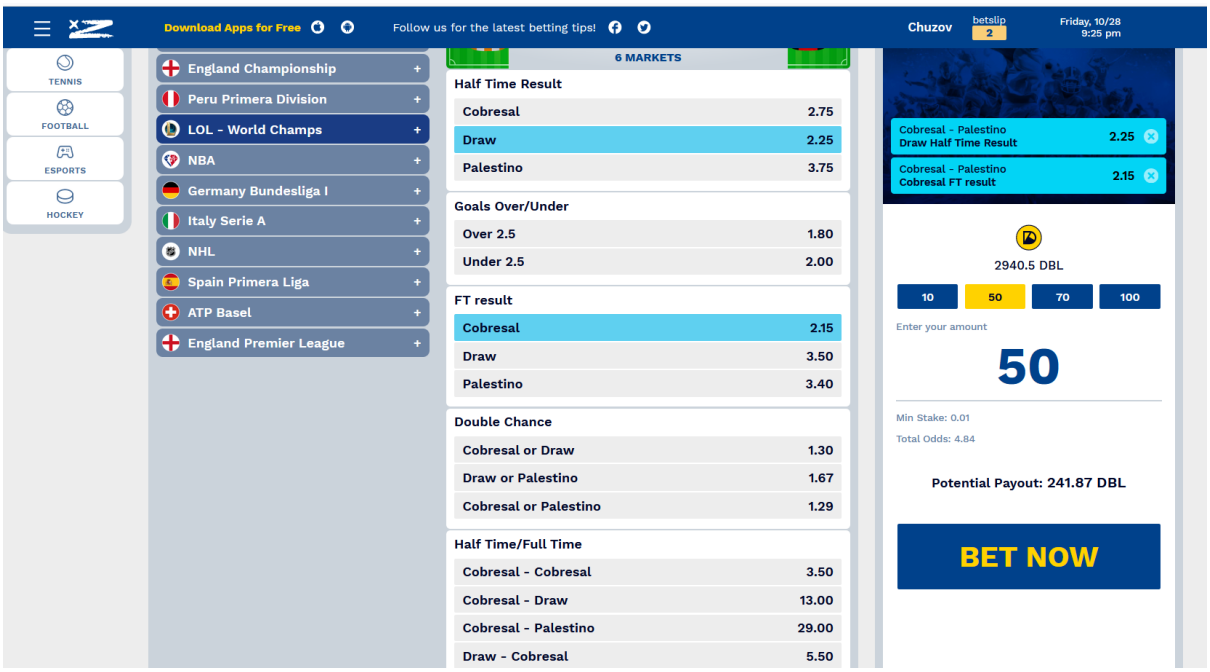

Рисунок 1.4 – Сторінка з переліком матчів та подій, на які можна зробити ставку

Для кожного описаного компоненту системи можна написати набір функціональних E2E тестів, які перевірятимуть роботу систему при певних сценаріях використання. При написанні автотесту для перевірки певного функціоналу будемо зіставляти зміни, що трапляються у графічному інтерфейсі з очікуваним результатом, щоб переконатися чи правильно працює система. Наприклад при авторизації з використанням неправильного паролю, користувачу має відобразитися відповідна помилка, а при використанні коректних облікових даних, навпаки, користувача має авторизувати в систему. Окрім відстежування змін в графічному інтерфейсі будемо слідкувати за станом та статусами API-запитів, що відправляються під час роботи вебдодатку щоб перевірити коректність роботи системи на більш низькому рівні.

1.3 Аналіз вимог до сайту

Після ознайомлення з сайтом можна визначити список функціональних вимог до кожного компоненту системи які мають виконуватись для коректної роботи системи. Надалі спираючись на цей список можна буде побудувати список тестових випадків які необхідно перевірити під час тестування. Отже спочатку визачимося з основними функціональними вимогами до системи.

Система має надавати можливість створити новий обліковий запис вказавши унікальний пароль довший за 8 символів, електрону адресу, псевдонім та поставивши галочку «Згоден з конфіденційністю і умовами».

Можливість авторизуватися в існуючий обліковий запис вказавши відповідну електронну адресу та пароль.

При спробі авторизуватися чи зареєструватися використовуючи некоректні дані, наприклад закороткий пароль або електронну адресу чи псевдонім що все використовуються іншим користувачем, повинна з’явитися відповідна помилка.

Після авторизації користувачу має відобразитися запит на дозвіл відправлення нотифікацій з веб-сторінки. Поки доступ не надано сторінка має не реагувати на будь-які дії, тобто бути заблокованою.

На головній сторінці сайту має відображатися список матчів та список ігрових подій (хто переможе, скільки буде голів, хто заб’є більше всього голів тощо) на які саме можна зробити ставку.

На головній сторінці має бути поле для пошуку ігор. При вводі тексту у це поле список ігор має містити лише ті матчі, назва команд, спорту чи ліги яких має в назві текст введений користувачем.

Для одного матчу можна обрати до десяти ігрових подій.

Після того як були обрані бажані ігрові події у користувача має бути можливість ввести бажану кількість дублонів за які він може зробити ставку. Після вводу бажаної кількості дублонів має відображатися можливий виграш

у разі якщо результат усіх ігрових подій співпав з результатом матчу, враховуючи коефіцієнти на ці події.

Після вводу бажаної кількості дублонів має бути можливість зафіксувати ставку натиснувши відповідну кнопку. При цьому дублони мають бути списані з балансу користувача.

Коли будуть відомі результати матчу у разі якщо користувач вгадав результати усіх обраних ігрових подій йому на баланс має бути доданий відповідний виграш.

У разі вводу некоректних даних (спроби обрати більш ніж 10 ігрових подій для одного матчу, спроби поставити більше дублонів ніж зараз є на балансі користувача) має бути відображено відповідне повідомлення, а сама дія має бути проігнорована.

Можливість змінити псевдонім облікового запису на інший, якщо він не зайнятий іншими користувачами.

Можливість змінити формат відображення коефіцієнтів на головній сторінці до американського формату, чи формату правильних та неправильних дробів.

Можливість змінити пароль вказавши свій поточний пароль та вписавши новий.

Можливість зіграти в гру “Lucky Rolls” на окремій сторінці під назвою “Perks”. У разі вигравши користувачу має бути записано на баланс відповідну кількість дублонів.

1.4 Засоби автоматизації тестування вебдодатків

Для автоматизації тестування вебдодатків використовуються спеціальні фреймворки автоматизації, що надають можливість автоматизувати роботу в браузері для симуляції дій, що можуть виконуватися реальним користувачем, наприклад відкриття сторінки та натискання на кнопку.

Серед актуальних інструментів для автоматизації тестування вебсайтів найчастіше зустрічаються фреймворки наведені далі.

Selenium – це портативний фреймворк, який широко використовується для тестування вебдодатків. Він надає інструментарій для написання функціональних E2E тестів використовуючи різні мови програмування.

Cypress – відносно новий фреймворк для E2E-тестування з відкритим кодом. Його автори пропонують дуже обширну документацію надаваємого функціоналу та інтегроване середовище для написання тестів, що робить фреймворк зручним у використанні та вивченні.

Playwright – це популярний інструмент автоматизації тестування, підтримуваний Google. Він підходить для автоматизації браузерів Chrome і Firefox. Фундаментально, Playwright є інструментом автоматизації, тому найчастіше він використовується для вебскрапінгу або наприклад генерації pdf-документів, тому використання його як інструменту для тестування потребує додаткових налаштувань, наприклад інструментарію для генерації зрозумілих звітів.

Для реалізації інформаційної системи було вирішено обрати фреймворк Cypress, тому визначимо його переваги та недоліки щоб переконатися що він підходить для конкретного завдання.

Істотною перевагою Cypress є зручність та прозорість в написанні коду, а також легкість в розгортанні проєкту та налаштуванні середовища для запуску тестів. Оскільки при створені інформаційної системи доведеться налаштовувати автоматичний запуск тестів на сервері, невибагливість цього фреймоврку до середовища виконання є вагомою перевагою, адже з цим би обов’язково довелось зустрітися при роботі з Selenium або Playwright.

До того ж готове інтегроване середовище надає широкий інструментарій для відлагодження написаних тестів. Кожен етапу тесту можна подивитися у браузері, щоб дізнатися шо саме пішло не так, та отримати готовий звіт про кожну помилку [2].

Окрім того Cypress надає широкий інструментарій для роботи з API запитами, наприклад засоби для відправлення власних запитів та отримання відповіді від сервера, а також засоби для перехоплення та слідкування за запитами, відправленими браузером на вебсторінці. Цей функціонал буде широко використовуватись при тестуванні вебдодатку, адже зазвичай недостатньо відстежити зміни у графічному інтерфейсі щоб переконатися у коректній роботі системи. Треба також переконатись що при виконанні дій що потребують взаємодії з сервером, наприклад авторизація чи зміна балансу ігрової валюти, відсилаються та приходять коректні дані.

Серед недоліків фреймворку можна відмітити відносно низьку швидкість роботи тестів, а також неможливість одночасно працювати з кількома сторінками одночасно, тобто кожен сценарій має виконуватися в одній вкладці браузера [3]. В певних ситуаціях це може стати критичним недоліком, проте в даному випадку на сайті, що тестуватиметься, не буде сценаріїв що потребують одночасну роботу в двух вкладках браузера одночасно. Ще одним недоліком є відсутність інструменту для генерації єдиного звіту після запуску всіх тестів одночасно, проте цей недолік відстежується при роботі з будь-яким фреймворком, тому компенсуємо його за допомогою окремої бібліотеки для генерації звітв Allure.

Отже фреймворк Cypress було обрано завдяки зручність у використанні, широкому інструментарію для написання як E2E-, так і API-тестів, обширну документація функціоналу надаваємого фреймворком, та стабільність в роботі тестів.

1.5 Автоматичний запуск тестів

Важливим кроком до автоматизації контролю якості вебдодатків є своєчасний запуск створених тестів. Зазвичай найкращим рішенням є запуск тестів після кожного внесення змін у код сайту, щоб переконатися що нові

зміни не затронули работу все існуючих компонентів системи. Такий підхід можна реалізувати багатьма способами, але найкращим рішенням буде інтегрування тестів у вже існуючу систему. В даному випадку це буде використання системи Jenkins, яка широко використовуються при розробці сучасних вебдодатків.

Система Jenkins являє собою умовний веб-сервер, що надає можливість створення конвеєрів безперервної інтеграції та безперервного розгортання (CI/CD), тобто серію кроків, які необхідно виконати, щоб розгорнути нову версію програмного забезпечення. Завдяки автоматизації CI/CD на етапах розробки, тестування, виробництва та моніторингу життєвого циклу розробки програмного забезпечення організації можуть швидше розробляти якісніший код [4].

Одним з таких конвеєрів є набір команд і етапів необхідний для розгортання нашого сайту у тестовому середовищі. У цей перелік етап може входити збірка проекту, тобто вихідного коду сайту, встановлення необхідних пакетів, запуск докер-контейнеру, будь-які серверні налаштування тощо. Слідкуючи за цим конвеєром ми можемо дізнатися коли саме сайт зазнає нових змін, що ми зможемо використовувати як тригер для запуску автотестів на сервері.

2 ПРОЄКТУВАННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО КОНТРОЛЮ ЯКОСТІ

2.1 Написання тестових випадків

Першим етапом практичної реалізації системи автоматизованого контролю якості програмного забезпечення є складання набору тестових випадків за якими будуть написані самі автотести. Тест кейси мають покривати всі основні сценарії використання вебдодатку, щоб перевірити відповідність роботи системи визначеним вимогам, включаючи як позитивні, так і негативні випадки. У попередньому пункті роботи було описано основні функціональні компоненти системи та вимоги до них, тому створимо набір кейсів для кожного з цих компонентів.

Почнемо з першого компоненту системи – авторизації. Для тестування цього компонента, в якості позитивних випадків, необхідно перевірити можливість просто авторизуватися та зареєструватися в системі, використовуючи коректні дані, тобто унікальну електрону адресу, псевдонім, та пароль, що відповідає вимогам системи. А якості негативних випадків має бути перевірена коректність роботи системи при неправильному вводі даних, наприклад при використанні неправильного паролю при авторизації користувач має побачити відповідне повідомлення.

Список позитивних та негативних тестових випадків пов’язаних з авторизацією та реєстрацією зображено на рисунках 2.1 – 2.5.

1) Відкрити сторінку dublzplay.com/.
2) Натиснути кнопку “Sign Up” у верхній частині сторінки.
3) У з'явившийся панелі увести свою електронну адресу у поле “Email”.
4) Ввести пароль шо містить якнайменше 8 символів, одну букву та одну цифру у поле “Password”.
5) Ввести унікальний псевдонім у поле “Alias”.
6) Поставити прапорець біля поля “I agree to dublZ Play T&Cs”.
7) Натиснути кнопку “Sign Up” знизу форми реєстрації .
8) Переконатися що користувача авторизовано в систему під новим акаунтом.

Рисунок 2.1 – Тест кейс “Реєстрація нового облікового запису”

Передумова: Наявність облікового запису на сайті dublzplay.com/

1) Відкрити сторінку dublzplay.com/.
2) Натиснути кнопку “Login” у верхній частині екрану.
3) У з'явившийся панелі увести електронну адресу від свого облікового запису у поле “Email”.
4) Ввести пароль від облікового запису у поле “Password”.
5) Натиснути кнопку “Login” знизу форми авторизації.
6) Переконатися що користувача авторизовано в систему під потрібним акаунтом.

Рисунок 2.2 – Тест кейс “Авторизація користувача використовуючи існуючий обліковий запис”

Передумова: Наявність облікового запису на сайті dublzplay.com

1) Відкрити сторінку dublzplay.com/ .
2) Натиснути кнопку “Login” у верхній частині екрану.
3) У з'явившийся панелі увести електронну адресу від свого облікового запису у поле “Email” .
4) Ввести пароль від облікового запису у поле “Password” Натиснути кнопку “Login” знизу форми авторизації .
5) Переконатися що користувача авторизовано в систему під потрібним акаунтом.
6) Відкрити навігаційне меню сайту натиснувши на “кнопку-бургер” (кнопку з трьома горизонтальними лініями “☰”) у верхній частини екрану.
7) Натиснути кнопку “Log Out”.
8) Переконатися що користувач вийшов зі свого облікового запису і більш не авторизован у системі.

Рисунок 2.3 – Тест кейс “Вихід з облікового запису користувача”

1) Відкрити сторінку dublzplay.com/
2) Натиснути кнопку “Sign Up” у верхній частині сторінки .
3) У з'явившийся панелі увести некоректний формат електронної адреси у поле “Email” (наприклад chuzov.gmail.com або chuzov@gmail тощо).
4) Ввести пароль шо містить якнайменше 8 символів, одну букву та одну цифру у поле “Password”.
5) Ввести унікальний псевдонім у поле “Alias”.
6) Поставити прапорець біля поля “I agree to dublZ Play T&Cs”
7) Натиснути кнопку “Sign Up”.
8) Переконатися що новий обліковий запис не було створено, а під полем “Email” з’явилося повідомлення про помилку (“Enter valid email”).

Рисунок 2.4 – Тест кейс “Спроба зареєструвати обліковий запис використовуючи некоректний формат електронної адреси”

1) Відкрити сторінку dublzplay.com/ .
2) Натиснути кнопку “Sign Up” у верхній частині сторінки .
3) У з'явившийся панелі увести свою електронну адресу у поле “Email” .
4) Ввести пароль шо містить менше ніж 8 символів у поле “Password”.
5) Ввести унікальний псевдонім у поле “Alias”.
6) Поставити прапорець біля поля “I agree to dublZ Play T&Cs”.
7) Натиснути кнопку “Sign Up”.
8) Переконатися що новий обліковий запис не було створено, а під полем “Password” з’явилося повідомлення про помилку (“Your password must be at least 8 characters long, must contain 1 letter, 1 uppercase letter and 1 number”).
9) Очистити поле “Password” та ввести туди пароль що містить більш ніж 8 символів, але не містить цифр.
10) Переконатися що обліковий запис знов не було створено, а помилка залишилася.

Рисунок 2.5 – Тест кейс “Спроба зареєструвати обліковий запис використовуючи некоректний формат паролю”

Наступним компонентом системи є редагування облікового запису. В якості позитивних сценаріїв має бути перевірена можливість змінювати пароль, псевдонім та можливість змінити формат відображення коефіцієнтів на ставках, а для негативних перевіримо коректність роботи системи при спробі змінити псевдонім на все існуючий або змінити пароль на такий що не відповідає вимогам системи. Тестові випадки для цього функціоналу зображено на рисунках 2.6 – 2.11.

В свою чергу тестові випадки для перевірки функціоналу блоку зі ставками, а саме можливість обрати ігрові події на які можна зробити ставку, відображення можливого виграшу та обробка некоректного вводу суми ставки зображено на рисунках 2.12 – 2.15.

Передумова: Наявність облікового запису та авторизований стан у системі.

1) Відкрити навігаційне меню сайту натиснувши на кнопку з трьома горизонтальними лініями “☰” у верхній частини екрану.
2) Натиснути на кнопку “Account”.
3) На сторінці редагування облікового запису користувача ввести новий унікальний псевдонім довший за 3 символи у поле “Alias”.
4) Натиснути кнопку “Save”.
5) Оновити сторінку та переконатися що псевдонім було змінено.

Рисунок 2.6 – Тест кейс “Редагування псевдоніму користувача”

Передумова: Наявність облікового запису та авторизований стан у системі.

1) Відкрити навігаційне меню сайту натиснувши на кнопку з трьома горизонтальними лініями “☰” у верхній частини екрану.
2) Натиснути на кнопку “Account”.
3) На сторінці редагування облікового запису користувача натиснути на кнопку “Fraction” у слайдері “Odds Format”
4) Натиснути кнопку “Save”.
5) Перейти на головну сторінку на переконатися що всі коефіцієнти на виграш біля спортивних подій відображаються як правильні дроби (7/10, 7/4 тощо)
6) Повернутся на сторінку редагування акаунту та обрати варіант відображення “Decimal”
7) Перейти на головну сторінку на переконатися що всі коефіцієнти на виграш біля спортивних подій відображаються як десятичні дроби (2.75, 1.40 тощо)
8) Повернутся на сторінку редагування акаунту та обрати варіант відображення “American”
9) Перейти на головну сторінку на переконатися що всі коефіцієнти на виграш біля спортивних подій відображаються у американському форматі (140, -275 тощо)

Рисунок 2.7 – Тест кейс “Зміна формату відображення коефіцієнтів на виграш”

Передумова: Наявність облікового запису та авторизований стан у системі.

1) Відкрити навігаційне меню сайту натиснувши на кнопку з трьома горизонтальними лініями “☰” у верхній частини екрану.
2) Натиснути на кнопку “Account”.
3) Ввести новий бажаний пароль у поле “Change password”
4) Ввести такий самий пароль у поле “Confirm new password”
5) Ввести поточний пароль у поле “Old Password”
6) Натиснути кнопку “Save”
7) Вийти з системи
8) Натиснути кнопку “Login”
9) Спробувати авторизуватися в систему використовуючи старий пароль від облікового запису
10) Переконатися що вхід в систему неможливий використовуючи старий пароль
11) Переконатися що використовуючи новий пароль користувач може увійти в свій обліковий запис

Рисунок 2.8 – Тест кейс “Редагування паролю користувача”

Передумова: Наявність двох облікових запису та авторизований стан у системі.

1) Відкрити навігаційне меню сайту натиснувши на кнопку з трьома горизонтальними лініями “☰” у верхній частини екрану.
2) Натиснути на кнопку “Account”.
3) У поле “Alias” ввести такий самий псведонім як і від іншого вашого облікового запису.
4) Переконатися в неможливості зміни такого псевдоніму і в тому що користувачу відображено повідомлення про те що такий псевдонім все зайнятий.

Рисунок 2.9 – Тест кейс “Спроба змінити псевдонім на вже існуючий”

Передумова: Наявність облікового запису та авторизований стан у системі.

1) Відкрити навігаційне меню сайту натиснувши на кнопку з трьома горизонтальними лініями “☰” у верхній частини екрану.
2) Натиснути на кнопку “Account”.
3) У поле “New Password” ввести пароль що відповідає вимогам системи (містить вісім або більше символів, одну букву та цифру).
4) У поле “Confirm New Password” ввести такий самий пароль як і на попередньому етапі.
5) У поле “Old Password” ввести неправильний поточний пароль від облікового запису.
6) Натиснути кнопку “Save Changes”
7) Переконатися що пароль не було змінено, а користувачу відобразилася відповідна помилка.

Рисунок 2.10 – Тест кейс “Спроба змінити пароль вказавши неправильний поточний пароль”

Передумова: Наявність облікового запису та авторизований стан у системі.

1) Відкрити навігаційне меню сайту натиснувши на кнопку з трьома горизонтальними лініями “☰” у верхній частини екрану.
2) Натиснути на кнопку “Account”.
3) У поле “New Password” ввести пароль що не відповідає вимогам системи (містить менш ніж 8 символів, або не містить цифр чи букв).
4) У поле “Confirm New Password” ввести такий самий пароль як і на попередньому етапі.
5) У поле “Old Password” ввести правильний поточний пароль від облікового запису.
6) Натиснути кнопку “Save Changes”
7) Переконатися що пароль не було змінено, а користувачу відобразилася відповідна помилка.

Рисунок 2.11 – Тест кейс “Спроба змінити пароль на такий, що не відповідає вимогам системи”

Передумова: Наявність облікового запису та авторизований стан у системі.

1) Натиснути на кнопку з логотипом у навігаційному меню сайту щоб перейти на головну сторінку.
2) Пролистати сторінку до блоку з видами спорту та матчами.
3) Натиснути на один з запропонованих матчів.
4) Спробувати обрати одну з ігрових подій на яку можна зробити ставку (загальна кількість голів, хто переможе тощо).
5) Переконатися що обрана подія стала виділена синім кольором.
6) Переконатися що справа над полем для вводу суми ставки відображається обрана подія.
7) Спробувати знов обрати цю ж саму подію.
8) Переконатися що з нею зникло виділення.

Рисунок 2.12 – Тест кейс “Спроба обрати ігрову подію для ставки”

Передумова: Наявність облікового запису та авторизований стан у системі.

1) Натиснути на кнопку з логотипом у навігаційному меню сайту щоб перейти на головну сторінку.
2) Пролистати сторінку до блоку з видами спорту та матчами.
3) Натиснути на один з запропонованих матчів.
4) Обрати кілька ігрових подій на які можна зробити ставку (загальна кількість голів, хто переможе тощо).
5) У поле для вводи суми ставки ввести будь-яке число (краще за все не кратне десяти).
8) Переконатися що у текстовому полі “Potential Payout” відображається правильний можливий виграш за ставку з врахуванням коефіцієнтів на обрані події.

Рисунок 2.13 – Тест кейс “Відображення можливого виграшу”

Передумова: Наявність облікового запису та авторизований стан у системі.

1) Натиснути на кнопку з логотипом у навігаційному меню сайту щоб перейти на головну сторінку.
2) Пролистати сторінку до блоку з видами спорту та матчами.
3) Натиснути на один з запропонованих матчів.
4) Обрати одну з ігрових подій на яку можна зробити ставку (загальна кількість голів, хто переможе тощо).
5) У поле для вводи суми ставки ввести невелике число дублонів, яке є на балансі облікового запису.
6) Переконатися що ставку було зроблено та перевірити чи всі дані на з'явившомуся чеці правильні (назва гри, матчу, команд).

Рисунок 2.14 – Тест кейс “Спроба зробити ставку”

Передумова: Наявність облікового запису та авторизований стан у системі.

1) Натиснути на кнопку з логотипом у навігаційному меню сайту щоб перейти на головну сторінку.
2) Пролистати сторінку до блоку з видами спорту та матчами.
3) Натиснути на один з запропонованих матчів.
4) Обрати одну з ігрових подій на яку можна зробити ставку (загальна кількість голів, хто переможе тощо).
5) У поле для вводи суми ставки ввести велике число дублонів, якого нема на балансі облікового запису.
6) Переконатися що ставка не була зроблена, а користувачу відобразилась відповідна помилка.

Рисунок 2.15 – Тест кейс “Спроба зробити ставку з при недостатньому балансі”

Передумова: Наявність облікового запису та авторизований стан у системі.

1) Відкрити навігаційне меню сайту натиснувши на кнопку з трьома горизонтальними лініями “☰” у верхній частини екрану.
2) Натиснути на кнопку “Perks”.
3) Натиснути кнопку “Lucky Rolls”.
4) Натиснути кнопку “Play”.
5) Переконатися що дев’ять карток, що відображенні на екрані, перегорнулися картинкою догори, а на баланс облікового запису була додана відповідна кількість дублонів в залежності від результату.

Рисунок 2.16 – Тест кейс “Спроба гри у Lucky Rolls”

2.2 Патерн Page Object Model або об'єктна модель сторінки

При написанні тестових випадків було помітно що при виконанні тестів часто зустрічаються багаторазово повторювані дії: натискання на кнопки в навігаційному меню, заповнення текстових полів, відкриття необхідних сторінок тощо. З одного боку це саме та причина чому використання автоматизованого тестування є необхідним у сучасних реаліях, але з іншого боку це змушує подбати про оптимізацію написання та виконання самих тестів.

Робота фреймоврку Cypress для тестування вебдодатків, як і більшості звичних фреймовркв, базується на пошуку необхідних HTML-елементів у DOM та виконанні певних дій з ними. Наприклад шукається HTML-елемент с ідентифікатором “password” щоб ввести туди свій пароль. Проте якщо в автотесті за кожною необхідністю окремо прописувати пошук потрібного елемента, то по-перше код такого тесту вийде занадто багатослівним і заплутаним, а по-друге підтримка такого коде стане непосильною задачею, адже при змінах в дизайн веб-сторінки необхідно буде відредагувати кожен

раз де викликається змінений елемент. Тому щоб вирішити ці проблеми при написанні тестів будемо використовувати патерн Page Object Model.

Суть цього патерну полягає у інкапсуляції роботи к кожною окремою сторінкою вебдодатку в окремий клас. Тобто за кожну дію на певній сторінці буде відповідати один метод з цього класу, а код самого автотесту буде складатися з почергового виклику цих методів. Якщо дія повторюється декілька раз до достатньо буде повторно викликати той самий метод замість того щоб знов писати повторюємий код. Підтримувати такий код буде в рази легше, адже при змінах в дизайн сторінки сайти буде достатньо один раз внести зміни у відповідний клас [5].

Основними сторінками на яких буде перевірятися функціонал додатку є головна сторінка сайту, на якій знаходиться форма для реєстрації і авторизації, а також список ігрових подій з можливістю зробити ставку, сторінка з налаштуваннями профілю користувача і сторінка під назвою “Perks”, де користувач може зіграти в гру “Lucky Rolls” щоб заробити ігрову валюту. Отже за патерном page object model у тесті будемо використовувати три основні класи: MainPage, AccountPage та PerksPage.

У класі MainPage будуть описані методи необхідні для реєстрації та авторизації користувача, а також методами для роботи зі ставками. Наприклад клас “openLogin” буде натискати кнопку щоб відкрити форму авторизації. Клас “typeEmail” буде вводити переданий текст у поле для електронної адреси користувача, клас “typePassword“ буде вводити пароль, а кнопка “clickLogin” в свою чергу відправлятиме заповнену форму. У класі PerksPage будуть методи для гри в “Lucky Rolls” та дій пов’язаних з цим, а у класі AccountPage будуть методи для редагування облікового запису користувача.

Далі описані класи будуть імпортуватися в створюємі фреймоврком Cypress набори з тестів (test suites), де для виконання тесту будуть використовуватися методи для роботи зі сторінками. Ці набори тестів можна також розглядати як класи, а методи в них як окремий тест кейс, який в свою

чергу складається з почергово виклику методів описаних в класах-сторінках. Наприклад для перевірки авторизації користувача достатньо буде викликати описані раніше методи “openLogin”, “typeEmail”, “typePassword” та “clickLogin” з класу “MainPage”.

Всього створимо п’ять наборів з тестовими випадками: набір “Login Tests” відповідатиме за авторизацію, “Signup Tests” – за реєстрацію, “Bets Tests” – за роботу зі ставками, “Account Page Tests” – за редагування акаунуту, а “Perks Page Tests” – за дії на сторінці “Perks”. Діаграму всіх описаних класів можна побачити на рисунку 2.17.

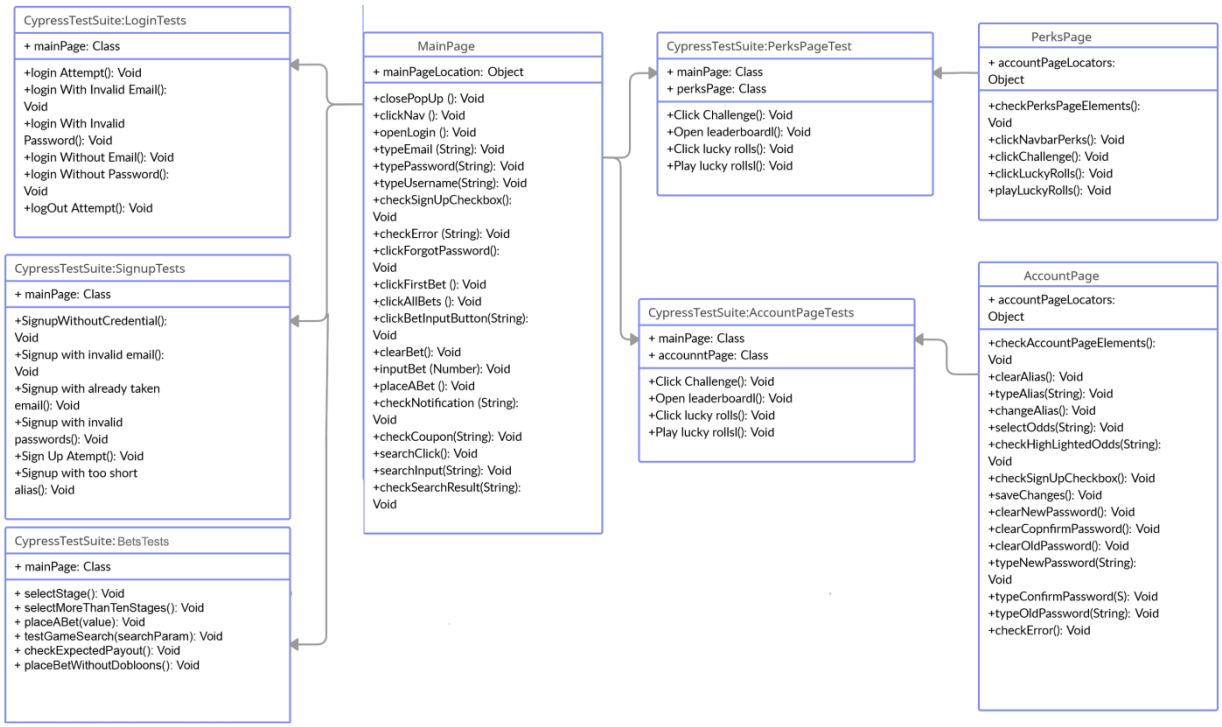

Рисунок 2.17 – Діаграма класів автотестів

2.3 Діаграма послідовності інформаційної системи

Перед тим як приступати до написання автотестів треба спроектувати як саме вони будуть інтегруватися в систему Jenkins і коли будуть запускатися. У попередньому етапі було вирішено, що найкращим варіантом

буде запускати тести після кожного оновлення вихідного коду вебдодатку, залишилося питання про інтеграцію етапу з тестуванням у існуючий конвеєр.

Наразі цей конвеєр запускається девопсом або програмістом коли є необхідність в оновлені вебдодатку, після чого система Jenkins розпочинає відповідний процес: спочатку з системи керування версіями, в даному випадку це git, клонується остання версія коду зі вказаної гілки, потім відбувається компіляція коду та створення докер-контейнера, в якому працює серверний код, а далі цей докер контейнер завантажуються на сервіс “Amazon ECR private registry”, котрий і є веб-сервером додатку.

Щоб не переповнювати цей конвеєр діями, краще буде створити окрему задачу, котра буде запускати тести через деякий час після завантаження докер контейнера з новим кодом на сервер. Після того як всі тести відпрацюють, за бажанням можна налаштування отримання сповіщень на електронну адресу або у мессенджер. Далі можна відкрити результат роботи задачі у Jenkins, переглянути звіт з результатами роботи тестів і за необхідністю завести баг-репорти зі знайденими помилками. Створену діаграму послідовності системи можна побачити на рисунку 2.18.

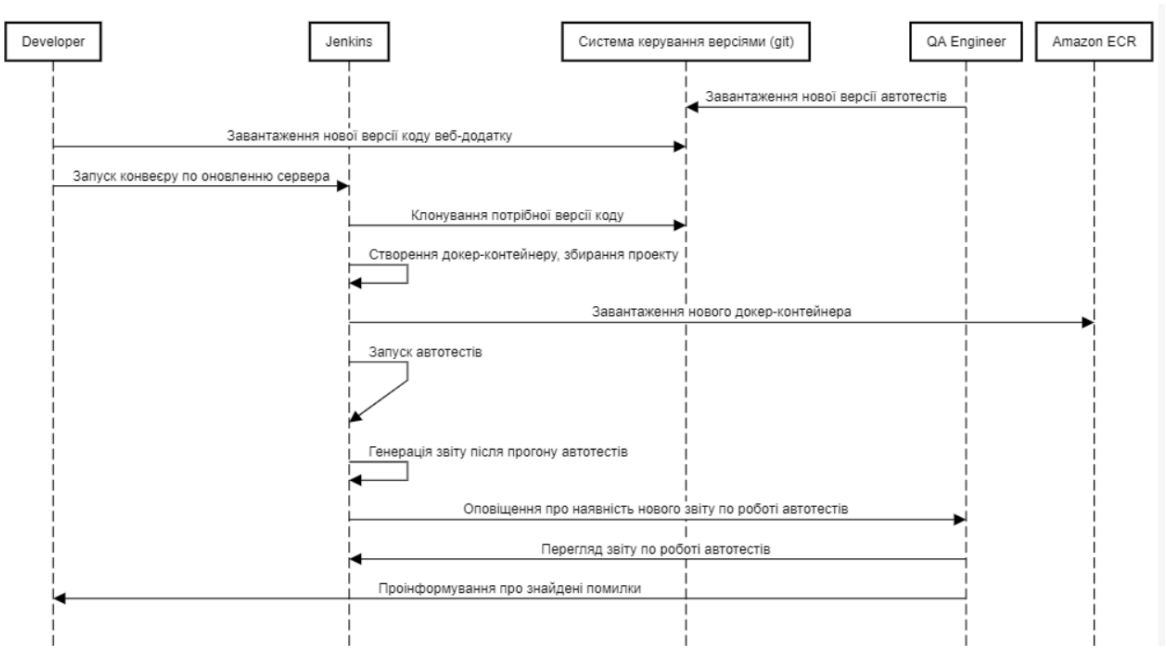

Рисунок 2.18 – Діаграма послідовності роботи інформаційної системи

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Ініціалізація проєкту та файлової структури

Першим етапом практичної реалізації інформаційної системи є створення нового проєкту та завантаження всіх необхідних бібліотек. Використовуючи фреймворк Cypress зробити це досить легко – достатньо завантажити і запустити відповідний npm-пакет. При першому запуску цього пакета автоматично згенеруються необхідні файли та файлова структура майбутнього проєкту, яку після деяких модифікацій будемо притримуватись в подальшій розробці. Окрім самого фреймворку для автоматизованого тестування нам також знадобляться бібліотека Allure для генерації візуально зрозумілих звітів з роботи тестів та плагін “cypress-browser-permissions”, який надає функціонал для надання веб-сторінці додаткових прав, наприклад доступ до геолокації чи можливості відправляти користувачу нотифікації. Згідно вимогам до системи, дозвіл на відправлення нотифікацій є необхідним для доступу до функціоналу сайту, тому плагін налаштуємо відповідним чином. Отриману файлову структуру проєкту можна побачити на рисунку 3.2.

У директорії “integration” буде знаходитись дві директорії – “pages”, у якій будемо описувати класи з методами для виконання дій на кожній окремій сторінці та експортувати їх, і саме “tests”, де будемо описувати набори тестів використовуючи методи з імпортуємих класів та синтаксис фреймворку Cypress.

У директорії “plugin” у файлі index.js підключимо та налаштуємо роботу плагінів Allure та cypress-browser-permissions. Код файлу можна побачити на рисунку 3.1.

Також необхідні налаштування необхідно прописати у файл cypress.json що знаходиться у кореневому каталозі проєкту. А саме вкажемо

розмір робочої зони вікна браузера у якому будуть виконуватись тести. Це налаштування буває особливо корисне коли треба протестувати роботу сайту на дисплеях з різною роздільною здатністю, але поки що просто пропишемо розмір вікна достаній для коректного відображення контенту. Окрім того пропишемо змінні оточення для виконання тестів, насамперед це всі посилання котрі будуть використовуватися у тестах, як адреси самих сторінок, так і маршрути API-викликів що відправляються на цих сторінках. Дуже зручно мати всі маршрути в одному місці, адже це дозволяє легко вносити в них необхідні зміни і не засмічувати код самих тестів.

Додатково пропишемо параметр “retries” котрий відповідає за те скільки раз тест кейс буде повторюватись у разі помилки. Це дає можливість відстежувати помилки, які складно відтворити та підвищує стабільність тестів. Код усього файлу cypress.json можна побачити на рисуснку 3.3.

```
const { cypressBrowserPermissionsPlugin } =       require('cypress-browser-permissions')
const allureWriter = require('@shelex/cypress-allure-plugin/writer');

module.exports = (on, config) => {
  // grant browser all permissions
  config = cypressBrowserPermissionsPlugin(on, config)

  // use allure to generate reports
  allureWriter(on, config);

  return config
}
```

Рисунок 3.1 – Код файлу з налаштування плагінів Allure та cypress-browser-permissions

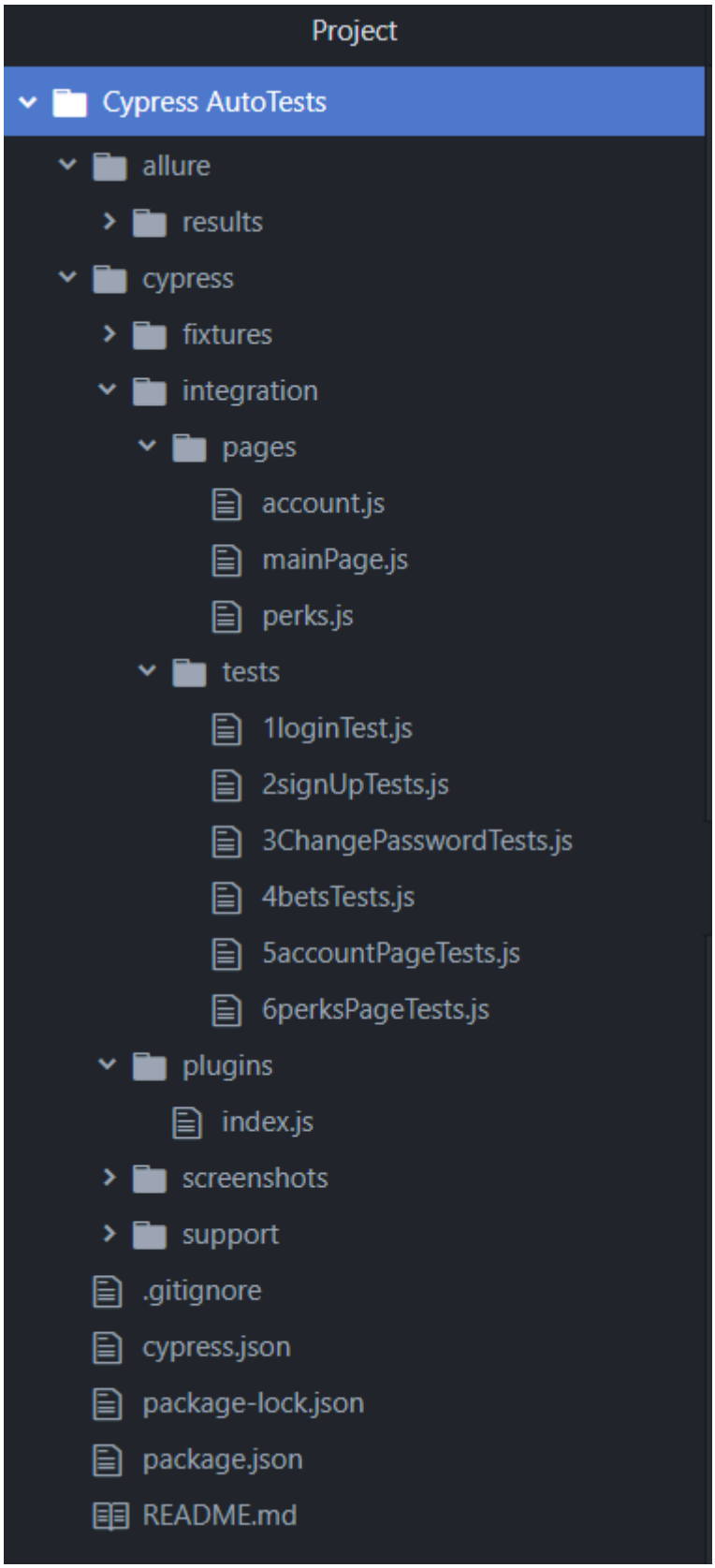

Рисунок 3.2 – Файлова структура проєкту

```
 "ignoreTestFiles": "**/pages/*.js",
 "viewportWidth": 1600,
 "viewportHeight": 700,
 "video": false,
  "baseUrl": https://dublzplay.com,
 "env":
  {
        "allureResultsPath": "./allure/results",
        "homePage": "https://dublzplay.com",
        "accountPage": "https://dublzplay.com/account",
        "myBetsPage": "https://dublzplay.com/my_bets",
        "perksPage": "https://dublzplay.com/perks",
        "luckyRollsPage": "https://dublzplay.com/perks/luckyrolls",
        "challengePage": "https://dublzplay.com/perks/challenges",
        "leaderboardPage": "https://dublzplay.com/perks/leaderboard",
        "luckyRollsPage": "https://dublzplay.com/perks/luckyrolls",

        "luckyRollsPlayAPI": "https://api.dublzplay.com/lucky/play",
        "profileAPI": "https://api.dublzplay.com/users/profile",
        "placeBetAPI": "https://api.dublzplay.com/bets/bet/bet365",
        "contactUsAPI": "https://api.dublzplay.com/users/contact-us",
        "loginAPI": "https://api.dublzplay.com/auth/login",
        "balanceAPI": "https://api.dublzplay.com/users/balance",
        "passwordChangeAPI": "https://api.dublzplay.com/users/password/change",

        "browserPermissions":
        {
        "notifications": "allow",
        "images": "allow",
        "popups": "allow",
        "plugins": "allow",
        "cookies": "allow"
        }
  },
  "defaultCommandTimeout": 10000,
  "pageLoadTimeout": 20000,

  "retries": {
        "runMode": 2,
        "openMode": 0
  }
}
```

Рисунок 3.3 – Код файлу cypress.json

3.2 Написання коду автотестів

Тепер можна приступити до написання коду самих автотестів. За патерном page object model треба почати з написання класів з методами для виконання необхідних дій на кожній сторінці. Наприклад однією з перших дій яку доведеться виконати при тестуванні сайту це авторизація користувача, тому розділимо цей процес на декілька етапів та напишемо відповідні методи. Перший метод буде використовуватись для відкриття форми авторизації, для цього використовуючи функції фреймворку Cypress .get() та .click() знайдемо та натиснемо на кнопку, що відкриває потрібну форму. Кнопку в свою чергу будемо знаходити за допомогою селекторів спираючись на атрибути її HTML-елементу [6]. Використовуючи інспектор в консолі браузера знаходимо атрибути цієї кнопки, та обираємо за яким з них будемо шукати її в тесті (рис. 3.5). В даному випадку кнопка має лише один клас – “authBar_login__2AA8y”, тому його і запишемо як поле об’єкту mainPageLocators під назвою “loginButton”, та будемо використовувати це поле при звертанні до цього елемента. Окрім цього селектора так само запишемо у цей клас селектори всіх інших елементів необхідних для проведення тестів на головній сторінці, наприклад всіх полів для вводу даних користувача: псевдоніму, паролю, електроної адреси тощо, блоків у яких буде з’являтися помилка, адже наявність і зміст цього блоку будемо перевіряти у негативних тестових випадках, кнопок для відправлення форм. а також елементів, що знадобляться для перевірки можливості зробити ставку, адже вони також знаходяться на головній сторінці. Код отриманого об’єкту mainPageLocators з усіма селекторами можна знайти на рисунку 3.4.

```
const mainPageLocators = {
  login: '.authBar_login__2AA8y',
  mainMenu: '.headerBar_content__2rTO9',
  navMenu: '.burgerMenu_options__3bnpy',
  menuItem: '.burgerMenu_item__2Stn5',
  navigation: '.MuiButtonBase-root',
  logout: '.burgerMenu_item__2Stn5',
  signup: '.authBar_signUp__23DKW',
  emailInput: 'input[name=username]',
  passwordInput: 'input[name=password]',
  dateOfBirth: 'input[name=dateOfBirth]',
  resetEmailInput: 'input[name=email]',
  aliasInput: 'input[name=alias]',
  checkBox: 'input[name=terms]',
  notification: '.notification_contentWrapper__3BE2x',
  errorMessage: '.MuiFormHelperText-root',
  loginButton: '.loginForm_button__3uEJX',
  signupBtn: '.registerForm_button__2BUFx',}
```

Рисунок 3.4 – Код класу з усіма атрибутами за якими шукатимуться необхідні елементи

Далі створимо методи для введення електронної адреси, паролю та відправки форми авторизації. Для вводу паролю та адреси так само знайдемо необхідні елемент використовуючи функцію .get() і передамо туди текст завдяки функції .type, після чого перевіримо наявність у полях введенного тексту використовуючи ствердження .should('have.value'). Залишилося лише створити метод для відправки цієї форми, для чого достатньо натиснути відповідну кнопку, але можна додатково зробити перевірку API-виклику що відповідає за авторизацію. Використовуючи засоби фреймворку що дозволяють слідкувати за відправленими запитами [7], а саме функцію intercept, додамо перевірку що після натискання на кнопку “Login” сторінка має відправити відповідний запит на сервер і отримати відповідь з кодом “200” що означає вдалу спробу авторизації. Код всіх методів що беруть участь у тесті для перевірки авторизації користувача можна знайти на рисунку 3.6, а код самого тесту з використанням цих методів – на рисунку 3.7.

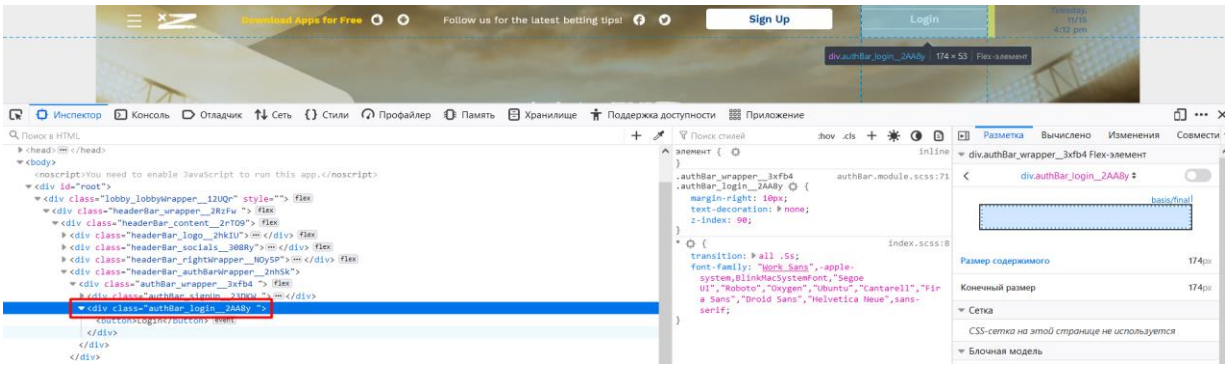

Рисунок 3.5 – Пошук атрибутів HTML-елемента

```
        openLogin() {
            cy.get(mainPageLocators.login).click()
            this.isLoginVisible()
            return this
    }
        typeEmail(text) {
            cy.get(mainPageLocators.emailInput).type(text).should('have.value', text)
            return this
    }
        typePassword(text) {
            cy.get(mainPageLocators.passwordInput).type(text).should('have.value', text)
            return this
}
        clickLogin() {
            cy.intercept('POST', Cypress.env('loginAPI')).as('login')
            cy.intercept('GET', Cypress.env('balanceAPI')).as('balance')
            cy.get(mainPageLocators.loginButton).click()
            cy.wait('@login', { timeout: 10000 }).its('response.statusCode').should('eq', 200)
            cy.wait('@balance').its('response.statusCode').should('eq', 200)
            cy.contains("Logged in successfully")
            cy.get('.loader_wrapper__3lfjj').should('not.not.exist')
}
```

Рисунок 3.6 – Код методів необхідних для перевірки авторизації на сайті

```
it('Login Attempt', () => {
        mainPage.openlogin()
        .typeEmail('chuzov.d@gmail.com')
        .typePassword('notarealpassword')
        .clickLogin()
})
```

Рисунок 3.7 – Код тест кейсу для перевірки авторизації на сайті

Використовуючи вже створені методи для роботи з формою авторизації можна створити тести для перевірки негативних тестових випадків, наприклад перевірити поведінку системи при вводі неправильного паролю чи відправки взагалі пустої форми, та випадків для перевірки реєстрації. До того ж дописавши кілька додаткових методів можна відразу написати тести для перевірки редагування облікового запису користувача, адже там використовуються майже такі ж самі поля для вводу паролю або псевдоніму як і на сторінці авторизації.

Код з перевіркою негативних кейсів авторизації можна побачити на рисунку 3.8, приклад коду тестів для перевірки функціоналу реєстрації на рисунку 3.9, а прикладу коду тестів редагування облікового запису на рисунку 3.10. Код усіх класів та тестів можна знайти у додатку А.

```
it('Login Using Wrong Password', () => {
        mainPage.openlogin()
        .typeEmail('chuzov.d@gmail.com')
        .typePassword('wrongpassword')
        .clickLogin()
        .checkError('Login failed')  })
it('Login Without Credentials', () => {
        mainPage.openlogin()
        .clickLogin()
        .checkError('Enter valid email')
        .checkError('Your password must be at least 8 characters long, must contain 1
letter, 1 uppercase letter and 1 number')
 })
```

Рисунок 3.8 – Код тест кейсу для перевірки негативних сценаріїв при авторизації

```
it('Sign Up attempt', () => {
        mainPage.clickSignUp()
        .typeEmail('chuzov.d' + Date.now() + '@gmail.com')
        .typePassword('Testpass123')
        .typeAlias("Autotest" + Date.now()
        .checkboxCheck()
        .clickSignUpBtn()
  })

  it('Signup Without Credential', () => {
        mainPage.clickSignUp()
        .clearPassword().clearEmail().clearAlias()
        .clickSignUpBtn()
        .checkError("Enter valid email")
        .checkError('Alias field is required')
        .checkError('Your password must be at least 8 characters long')
  })

  it('Signup with invalid email', () => {
        mainPage.clickSignUp()
.typePassword("Testpass1").typeEmail("email@test").typeAlias("chuzov"+Date.now())
        .clickSignUpBtn()
        .checkError('Enter valid email')
  })

  it('Signup with already taken email', () => {
        mainPage.clickSignUp()
.typePassword("Testpass1").typeEmail("chuzov.d@gmail.com").typeAlias("chuzov"+Da
te.now())
        .checkboxCheck()
        .clickSignUpBtn()
        .checkNotification("Account with this email")

  })


  it('Signup with invalid password', () => {
        mainPage.clickSignUp()
.typePassword("123")
.typeEmail("chuzov.d@gmail.com"+Date.now())
.typeAlias("chuzov"+Date.now())
        .clickSignUpBtn()
        .checkPasswordError('Your password must be at least 8 characters long')
        .checkboxError()
  })
```

Рисунок 3.9 – Код тестів для перевірки позитивних та негативних сценаріїв при реєстрації

```
  it('Change alias', () => {
    mainPage.login()
    cy.visit(Cypress.env('accountPage'))
    accountPage.clearAlias()
    .typeAlias('ChuzovTest2')
    .changeAlias()
    .verifyAlias()})
  it('Change alias to a too short one', () => {
    mainPage.login()
    cy.visit(Cypress.env('accountPage'))
    accountPage.clearAlias()
    .typeAlias('c1')
    .changeAlias()
    accountPage.checkError('Your alias must be at least 3 characters long')
  })
  it('Change alias to already existing one', () => {
    mainPage.login()
    cy.visit(Cypress.env('accountPage'))
    accountPage.clearAlias()
    .typeAlias('Middle')
    .changeAlias()
    mainPage.checkNotification('User already exists')
  })
  it('Change password', () => {
    accountPage.typeNewPassword("NewPassword123!")
    .typeConfirmPassword("NewPassword123")
    .typeOldPassword(Cypress.env('randomPassword'))
    .sendChangePassForm()
    mainPage.logout()
    .clickLogin()
    .typeEmail("chuzov.d@gmail.com")
    .typePassword("NewPassword123!")
    cy.visit(Cypress.env('accountPage'))
    accountPage.typeNewPassword("randomPassword")
    .typeConfirmPassword("randomPassword")
    .typeOldPassword(Cypress.env('NewPassword123'))
    .sendChangePassForm()
  })
  it('Change password with invalid new password', () => {
    accountPage.typeNewPassword("123456789")
    .typeConfirmPassword("123456789")
    .typeOldPassword(Cypress.env('randomPassword'))
    .sendChangePassForm(null)
    .checkError('Your password must be at least 8 characters long, must contain 1 letter, 1
uppercase letter and 1 number')
  })
```

Рисунок 3.10 – Код тестів для перевірки редагування псевдоніму чи паролю облікового запису

3.3 Результат роботи автотестів

Після написання коду автотестів необхідно перевірити їх працездатність та коректність обробки тестових сценаріїв. Для цього фреймоврк Cypress надає два способи запуску тестів – у “headed” та “headless” режимах браузеру використовуючи команду “npx cypress open” та “npx cypress run” відповідно. Їх різниця полягає у тому, що при запуску браузера у “headed” режимі буде відкрите повноцінне вікно браузера, у якому розробник може подивитися кожен етап виконання тесту в режимі реального часу та потім повернутися до кожного з них щоб побачити на якому етапі та саме через що саме трапилася помилка. Саме такий спосіб будемо використовувати зараз для перевірки та можливих виправлень написаних тестів. В той же час у “headless” режимі веб-браузер буде відкрито без графічного інтерфейсу користувача, тобто неможливо побачити що саме в ньому відбувається окрім виводів в командну строку про результат роботи кожного тесту. Такий спосіб будемо використовувати при запуску тестів у системі Jenkins для економії ресурсів серверу, адже побачити відкрите вікно браузера в будь-якому разі не можна.

При запуску тестів спочатку відкривається елемент графічного інтерфейсу який дозволяє обрати який саме набір тестових сценаріїв та у якому браузері треба виконати (рис. 3.11).

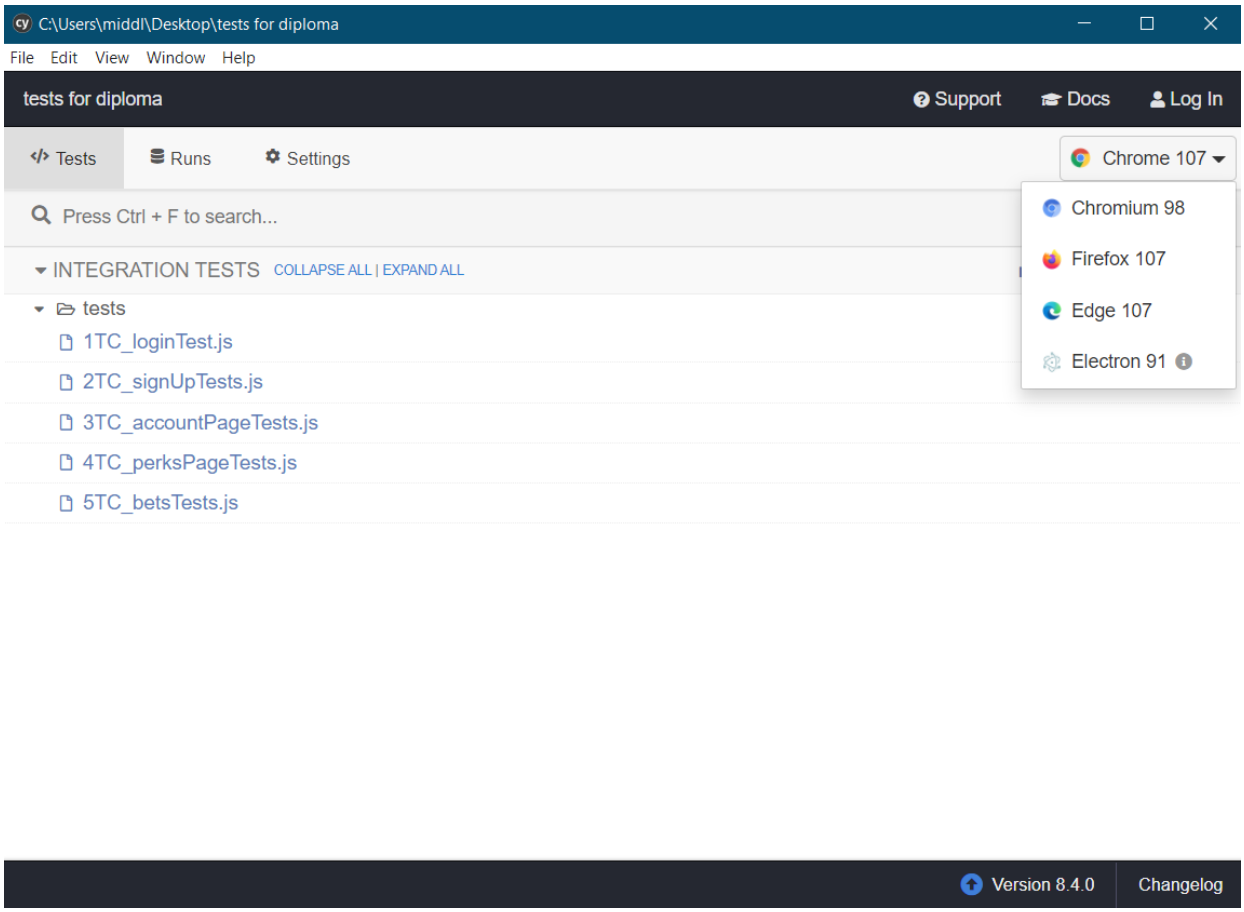

Рисунок 3.11 – Вікно для вибору набору тестових сценаріїв

Після вибору набору тестових сценаріїв відкривається браузер, в якому починають автоматично виконуватись дії описані у коді, наче їх виконує реальний користувач. Окрім того у браузері також знаходиться елемент графічного інтерфейсу додаваємий фреймворком Cypress котрий відображає список всіх тестів що мають виконуватись та який тест і яка саме дія зараз виконується. Стан браузера в якому виконуються тести можна побачити на рисунку 3.12.

Після виконання тестів можна побачити увесь список тестів та чи пройшли вони перевірку. Наприклад після запуску тестів на сторінці авторизації можна переконатися що всі з них відпрацювали коректно. Стан браузера в якому відображено результати роботи тестів можна знайти на рисунку 3.13.

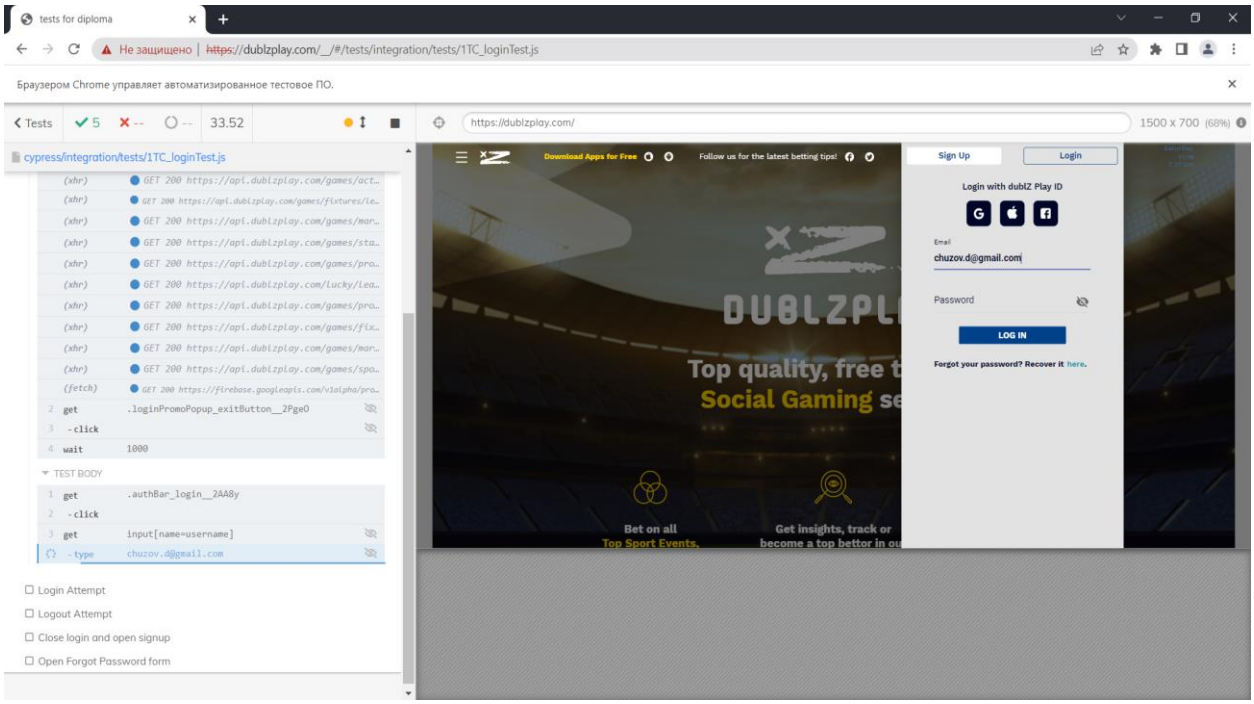

Рисунок 3.12 – Стан браузера під час виконання тестів

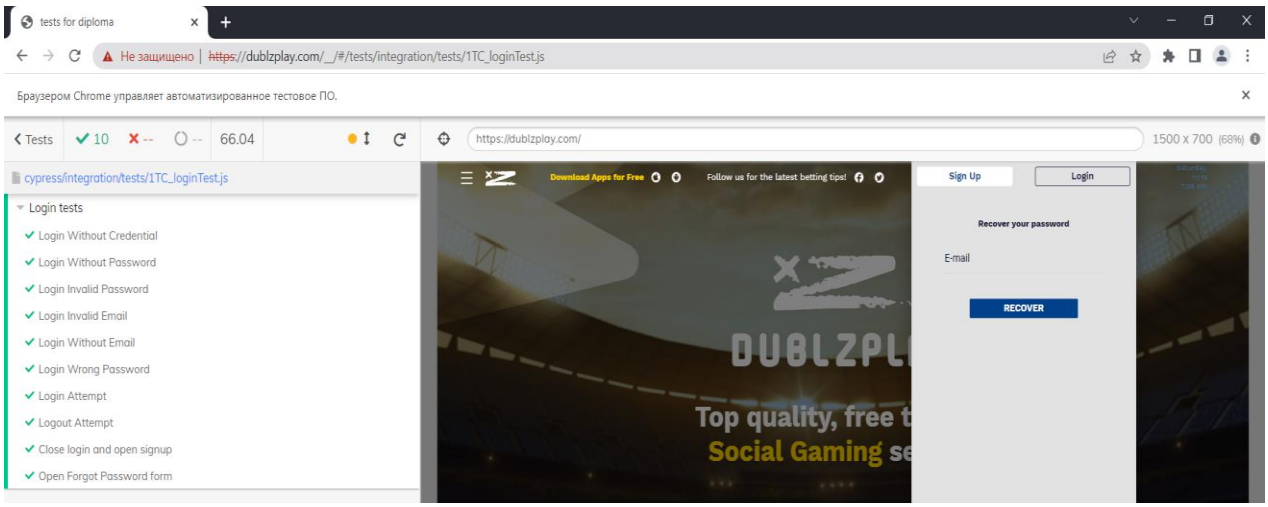

Рисунок 3.13 – Стан браузера після виконання тестів зі списком тестів що пройшли перевірку

А під час перевірки можливості реєстрації користувача виникла помилка. Зі списку тестів видно, що проблема трапилася під час виконання негативного тестового випадку, що перевіряє неможливість зареєструвати новий обліковий запис, використовуючи все зайнятий псевдонім. Якщо відкрити журнал виконання можна знайти всі кроки, що виконувались під час виконання тесту, в якому можна знайти саме етап винний за помилку у тесті.

Виявилось, що при реєстрації користувача використовуючи зайнятий псевдонім новий обліковий запис не буде створений, але користувачу не відобразиться відповідна помилка (рис. 3.14).

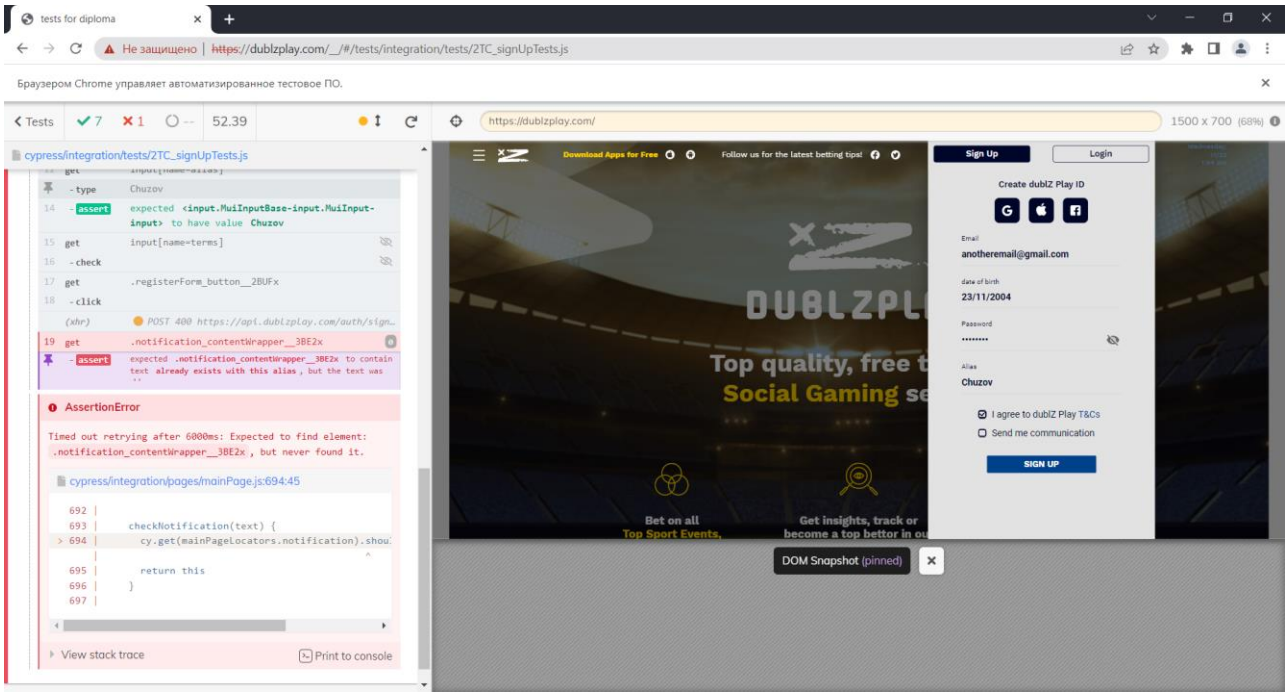

Рисунок 3.14 – Журнал виконання проваленого тесту з етапом на якому трапилася помилка

3.4 Інтеграція тестів у систему Jenkins

Завершальним етапом створення інформаційної системи є інтеграція створених тестів у систему Jenkins та налаштування їх автоматичного запуску. Для цього авторизувавшись у Jenkins та ознайомившись з інтерфейсом необхідно створити та налаштувати нову задачу яка буде відповідальна за запуск автотестів (рис 3.15).

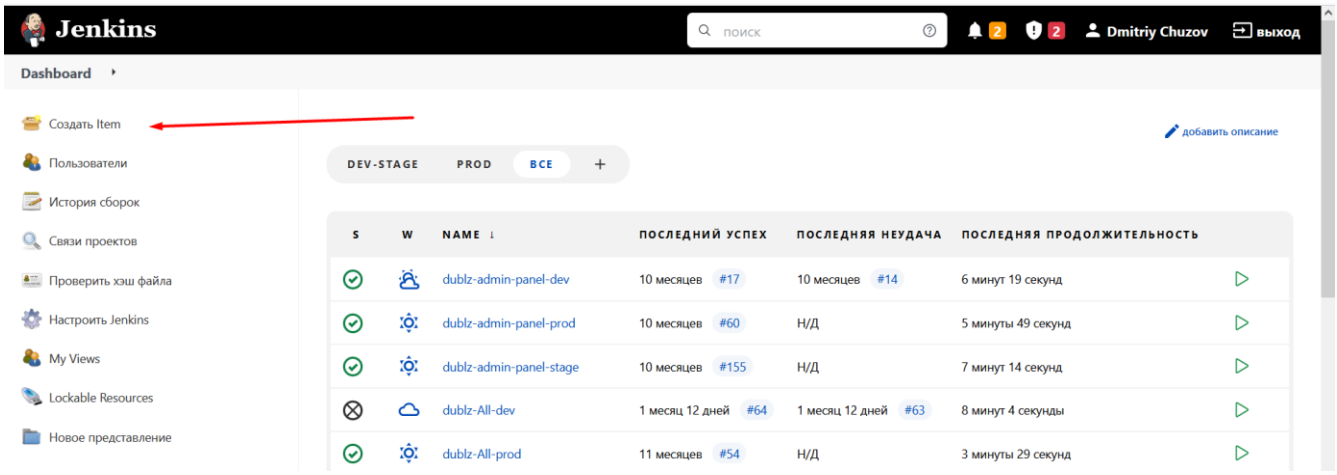

Рисунок 3.15 – Графічний інтерфейс системи Jenkins

При створенні задачі необхідно вказати необхідні параметри та налаштування, а саме:

– вказати шлях до репозиторію та гілки де зберігається код тестів (рис. 3.16);

– налаштувати умови при яких будуть запускатися тести. Оскільки було вирішено запускати тести при кожному внесенні змін у код сайту то можна прив’язатися до статусу конвеєрів відповідаючих за розгортання нової версії сайту на сервері, тобто автотести будуть запускатися кожен раз коли відпрацює конвеєр по оновленню сайту (рис. 3.17). А якщо буде потреба перевірити стабільність роботи сайту навіть коли в код не додаються нові зміні також налаштуємо запуск тестів кожного тижня;

– вказати середовище та команду для запуску тестів. Оскільки Cypress є JavaScript фреймворком то в середовищі виконання треба вказати node.js, а в командах для запуску пропишемо команду встановлення потребуємих бібліотек та для виконання самих тестів (рис. 3.18);

– вказати чи будуть та де будуть зберігатися звіти по роботі тестів.

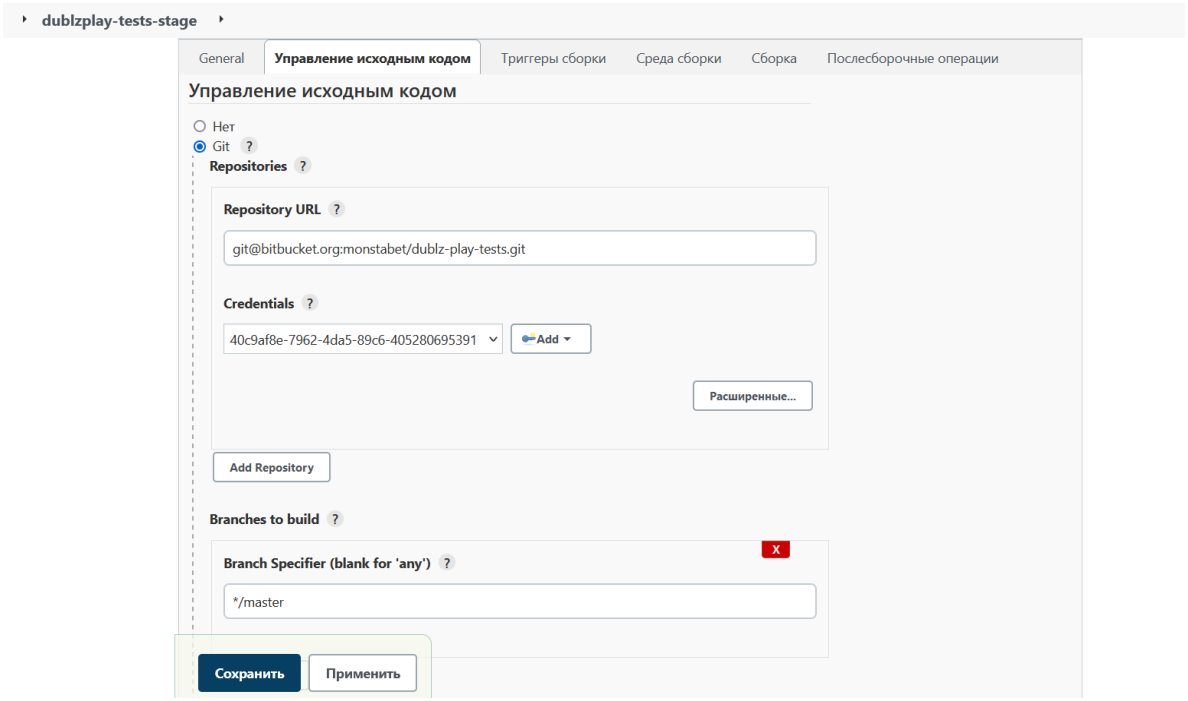

Рисунок 3.16 – Вказаний шлях до репозиторію з тестами при створенні нової задачі по їх запуску

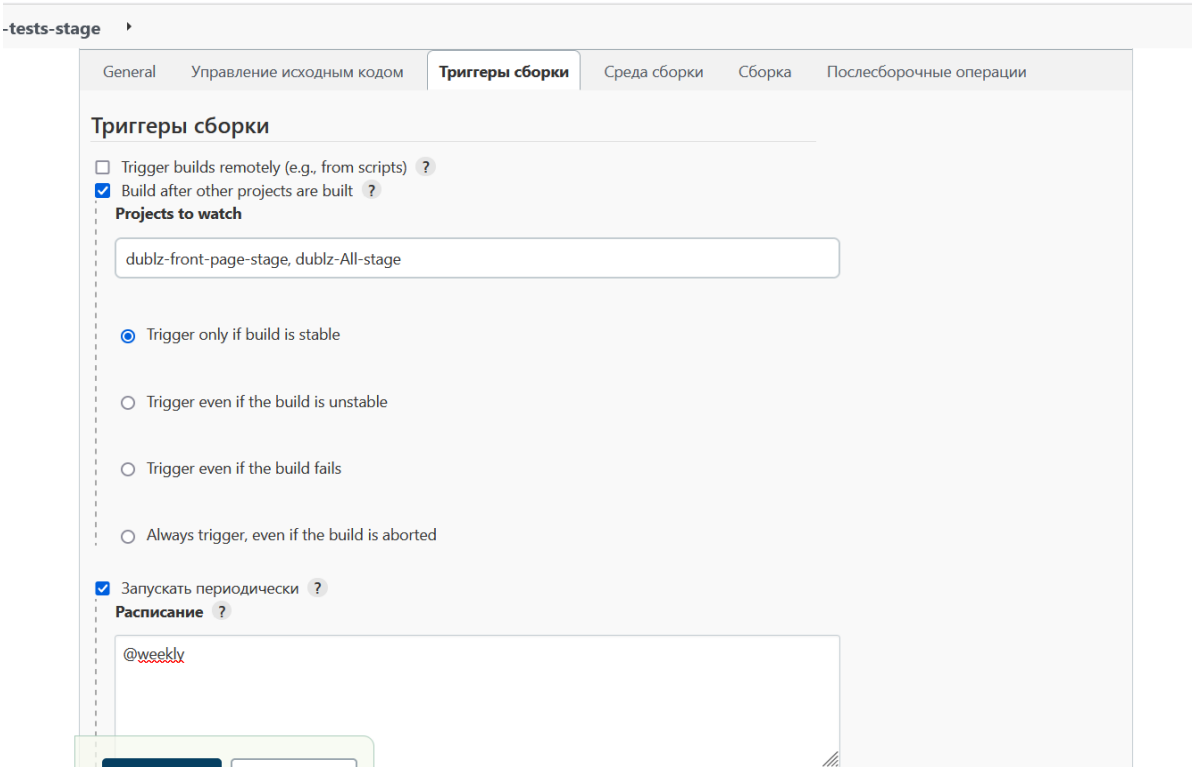

Рисунок 3.17 – Налаштування умов для запуску тестів

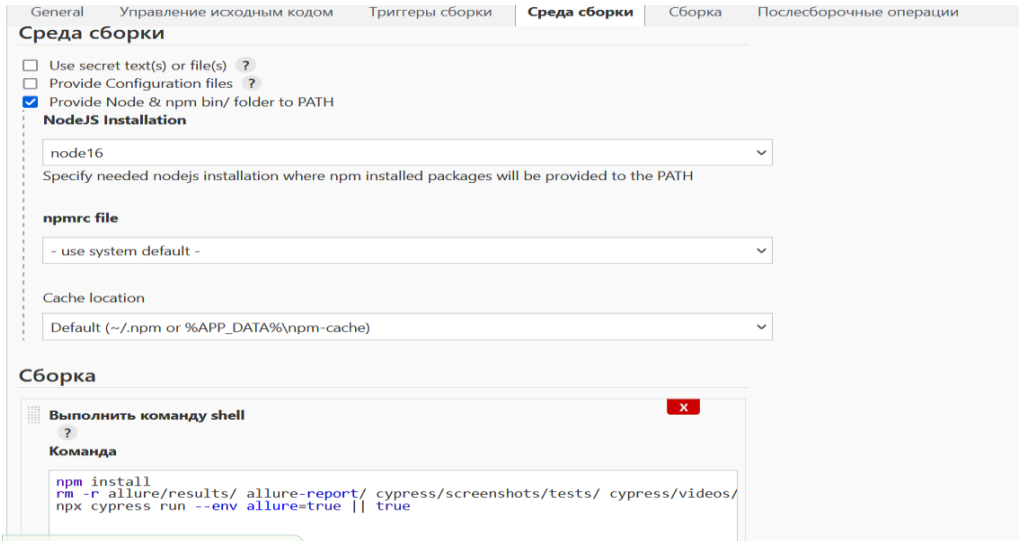

Рисунок 3.18 – Вибір середи виконання та команд для запуску тестів

Результатом виконаних налаштувань є нова задача в системі Jenkins, що відповідає за запуск тестів при виконанні вказаних умов, а саме при оновлені версії вебдодатку що тестується або раз у тиждень. Відкривши сторінку створеної задачі можна побачити коли останній раз запускались автотести, історію запусків та графік з результатами останніх запусків (рис. 3.19).

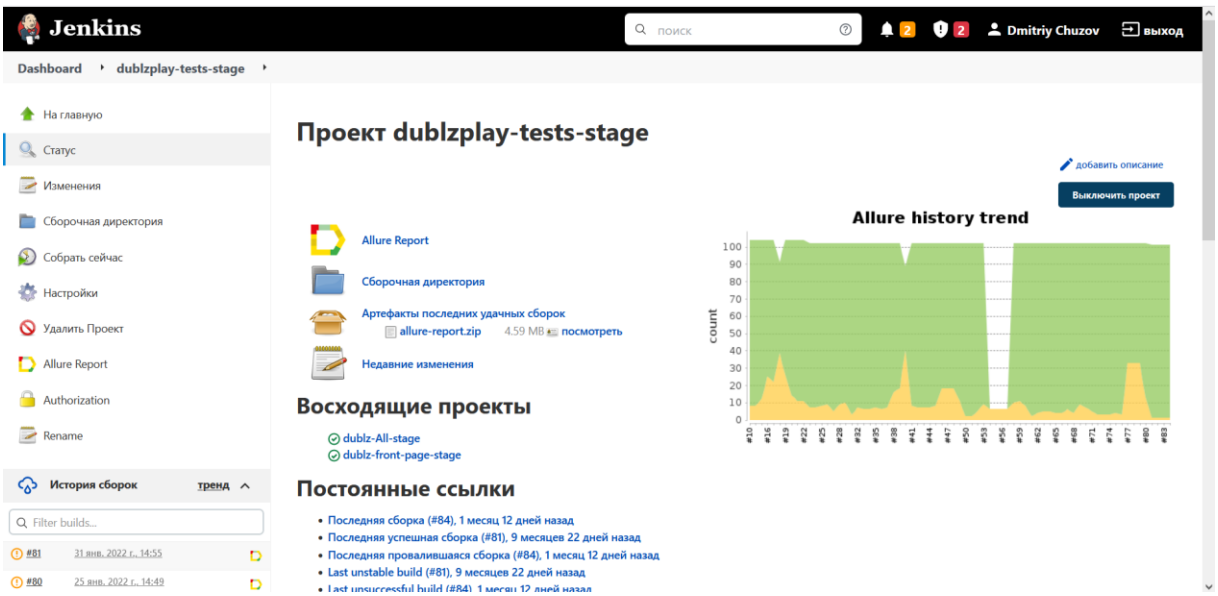

Рисунок 3.19 – Сторінка створеної задачі у системі Jenkins після прогону кількох тестів

Окрім того натиснувши на кнопку “Allure Report” можна подивитись детальний звіт з результатами роботи тестів (рис. 3.20). На звіті видно що при виконанні трапилася одна помилка, та сама що була виявлена на попередньому етапі. Натиснувши на неї у звіті можна подивитися більш детальний звіт саме про неї, перелік кроків які було виконано під час проходження тесту, етап на якому саме трапилася помилка та знімок екрану, зроблений на цьому етапі (рис 3.21).

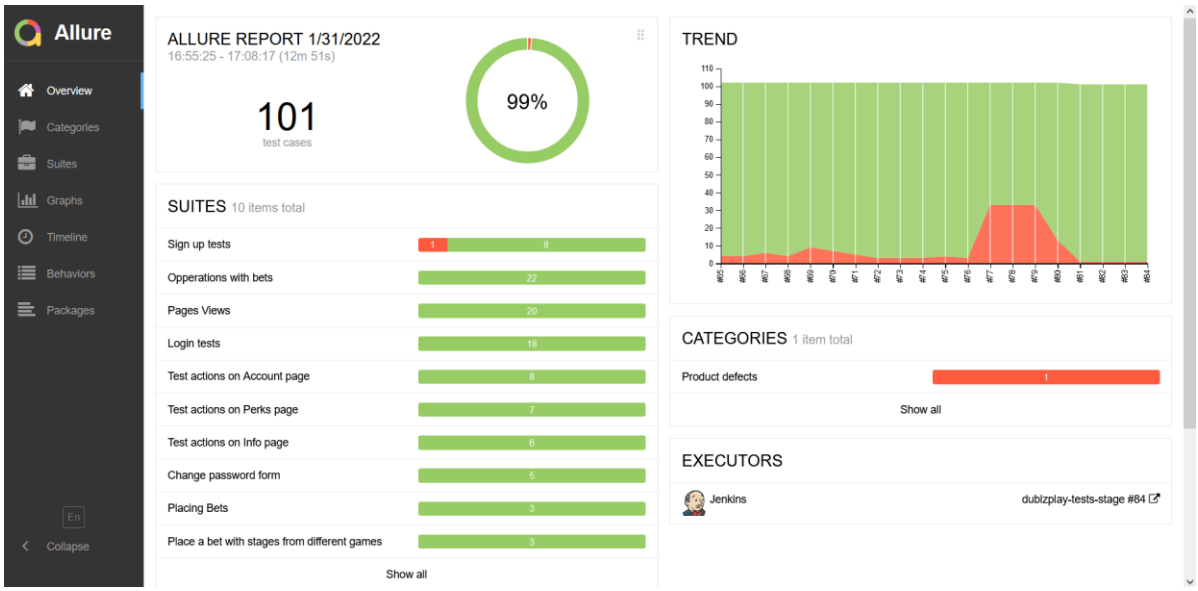

Рисунок 3.20 – Звіт Allure з результатами останнього запуску автотестів

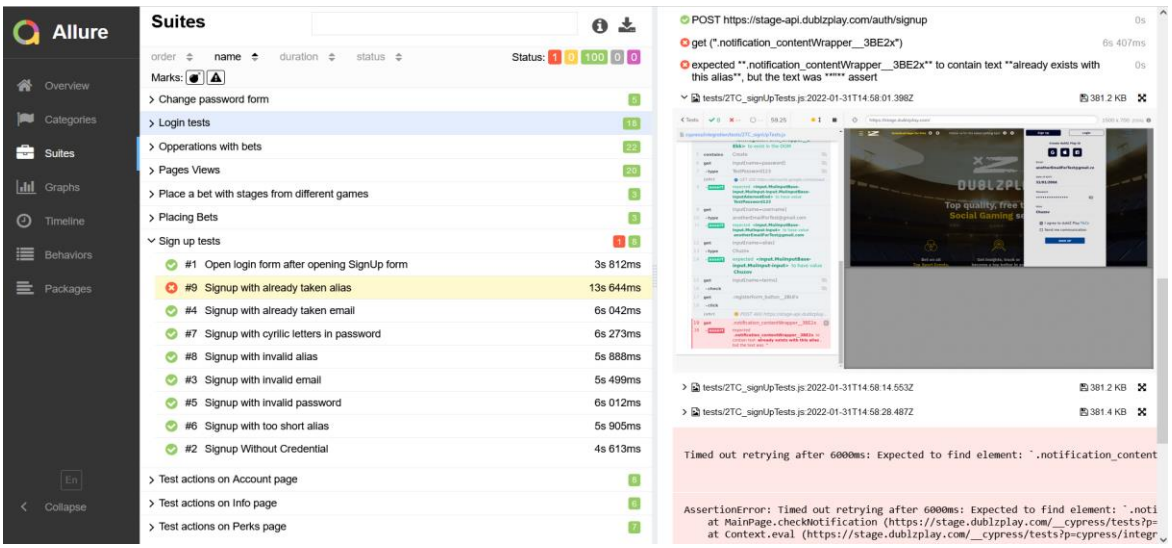

Рисунок 3.21 – Звіт про конкретну помилку у виконанні тесту

ВИСНОВКИ

Результатом роботи над кваліфікаційною роботою є створена система автоматизованого контролю якості програмного забезпечення, реалізована використовуючи засоби фреймворку Cypress та інтегрована в конвеєр системи безперервної інтеграції Jenkins. Дана система гарантує швидке та якісне тестування обраного вебдодатку після кожного оновлення його вихідного коду, що значно мінімізує необхідність участі у цьому процесі людських ресурсів, які в свою чергу зможуть бути виділені інші завдання.

При виконанні роботи було досліджено предметну область та створено список вимог до тестуємого вебсайту, розглянуто різні засоби та інструменти автоматизації тестування вебдодатків, створено список тестових випадків, які мають бути пройдені для перевірки відповідності функціоналу системи до визначених вимог, написано набір тест-кейсів використовуючи засоби фреймворка Cypress для E2E-тестування вебдодатку та інтегровано створені автотести в систему безперервної інтеграції Jenkins для своєчасного запуску тестів після кожного оновлення вихідного коду сайту та генерації звітів після кожного запуску тестів.

Особливістю створеної системи є повна автоматизація процесу контролю якості, включаючи запуск тестів та генерацію звітів, а також легкість у підтримці отриманого коду завдяки використанню сучасних патернів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Cohn M. Succeeding with Agile. Addison-Wesley Professional, Boston, 2009. 512 p.

2. Cypress vs Selenium vs Playwright vs Puppeteer speed comparison. URL: https://blog.checklyhq.com/cypress-vs-selenium-vs-playwright-vs-puppeteer-speed-comparison (дата звернення 24.09.2022).

3. Why Cypress? URL: https://docs.cypress.io/guides/overview/why-cypress (дата звернення 30.09.2022).

4. Ferguson D. Jenkins: The Definitive Guide. O'Reilly Media, Inc., 2011. 404 с.

5. Siminiuc A. Improve Selenium Code with Automation Patterns: Page Object Model Page Factory Page Elements Base Page Loadable Component. Kindle Edition, 2017. 112 p.

6. Cypress Documentation. URL: https://docs.cypress.io/api/table-of-contents (дата звернення 15.10.2022).

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

ДОДАТОК А

Код файлів з описаними класами та тестами

A.1 Account.js

```
const accountPageLocators = {
  fractions: '.oddsFormatSelector_wrapper__2R5Aa',
  save: '.accountForm_button__3nZS7',
  oddSelector: '.oddsFormatSelector_pointer__2detp',
  accountForm: '.accountForm_wrapper__LDCHo',
  securityForm: '.securityForm_wrapper__3o0Du',
  sectionTitle: '.tabItem_wrapper__3qAcT',

  newPassword: 'input[name=new_password]',
  confirmPassword: 'input[name=confirm_new_password]',
  oldPassword: 'input[name=old_password]',
  birthday: 'input[name=dateOfBirth]',
  alias: 'input[name=alias]',



  dateArrows: '.MuiButtonBase-root.MuiIconButton-
root.MuiPickersCalendarHeader-iconButton',
  dateNumber: '.MuiTypography-root.MuiTypography-
body2.MuiTypography-colorInherit',
  okButton: '.MuiButton-label',


  deleteBtn: '.securityForm_deleteBtn__3AvSg',
  secondDelete: '.accountDelete_delete__3Pc-T',
  deletionCode: 'input[name=code]',
  finalDeleteBtn: '.deleteForm_button__nGTLi',

  errorField: '.MuiFormHelperText-root',
  changeBtn: '.securityForm_button__2mvxf',
  notification: '.notification_contentWrapper__3BE2x',
}

export class AccountPage {
  checkAccountPageElements() {

cy.get(accountPageLocators.accountForm).scrollIntoView().should('exist'
).should('be.visible')

    cy.get(accountPageLocators.sectionTitle).last().click()

cy.get(accountPageLocators.securityForm).scrollIntoView().should('exist'
).should('be.visible')

    cy.get(accountPageLocators.sectionTitle).first().click()

cy.get(accountPageLocators.accountForm).scrollIntoView().should('exist'
).should('be.visible')
    return this
  }

    clickSecurity() {
      cy.get(accountPageLocators.sectionTitle).last().click()
      return this;
    }

    checkNotification(text) {

cy.get(accountPageLocators.notification).should('exist').should('contain',
text)
    }

    clearAlias() {
      cy.get(accountPageLocators.alias, { timeout: 10000
}).should('not.be.disabled',{ timeout: 10000 } ).clear()
      return this
    }

    typeAlias(text) {
      cy.get(accountPageLocators.alias, { timeout: 10000
}).should('not.be.disabled',  { timeout: 10000
}).type(text).should('have.value', text)
      return this
    }

    changeAlias(code) {
      cy.intercept('PUT', Cypress.env('profileAPI')).as('changeAlias')
      cy.get(accountPageLocators.save).click()
      cy.wait('@changeAlias').its('response.statusCode').should('eq', code)
      return this
    }

    changeAlias() {
      cy.get(accountPageLocators.save).click()
      return this
    }

    clickBirthday() {
      cy.get(accountPageLocators.birthday, { timeout: 10000 }).click()
      return this;
    }

    changeBirthday() {
       cy.get(accountPageLocators.dateArrows).first().click()
       cy.wait(2000)
       cy.get(accountPageLocators.dateNumber).contains('11').click()
       cy.get(accountPageLocators.okButton).contains('OK').click()
       cy.intercept('PUT', Cypress.env('profileAPI')).as('changeBirthday')
       cy.get(accountPageLocators.save).click({force: true})
       cy.wait('@changeBirthday').its('response.statusCode').should('eq',
200)
       return this;
      }

    selectOdds (odd) {

cy.get(accountPageLocators.fractions).contains(odd).should('be.visible').c
lick()
       cy.wait(500)

cy.get(accountPageLocators.fractions).contains(odd).should('be.visible').c
lick()
       cy.intercept('PUT', Cypress.env('profileAPI')).as('sendOddsType')

       cy.get(accountPageLocators.save).click()

       cy.wait('@sendOddsType').its('response.statusCode').should('eq',
200)
       cy.wait(2000)
       cy.contains('attributes changed successfully')
       return this;
    }

    highLightedOdds (odd) {
      cy.wait(2000)
      if(odd === "Decimal") {
        cy.get(accountPageLocators.oddSelector, { timeout: 10000
}).invoke('css', 'left')
        .then(str => parseInt(str)).should('be.lessThan',30)
      }

      else if(odd === "Fraction") {
```

```
        cy.get(accountPageLocators.oddSelector, { timeout: 10000
}).invoke('css', 'left')
        .then(str => parseInt(str)).should('be.greaterThan',
90).and('be.lessThan',95)
      }

      else if(odd === "American") {
        cy.get(accountPageLocators.oddSelector, { timeout: 10000
}).invoke('css', 'left')
        .then(str => parseInt(str)).should('be.greaterThan', 100)
      }
      return this;
    }

    typeNewPassword(text) {
      cy.get(accountPageLocators.newPassword, { timeout: 10000
}).type(text).should('have.value', text)
      return this
    }

    typeConfirmPassword(text) {
      cy.get(accountPageLocators.confirmPassword, { timeout: 10000
}).type(text).should('have.value', text)
      return this
    }

    typeOldPassword(text) {

cy.get(accountPageLocators.oldPassword).type(text).should('have.value',
text)
      return this
    }

    clearNewPassword(text) {
      cy.get(accountPageLocators.newPassword, { timeout: 10000
}).clear()
      return this
    }

    clearOldPassword(text) {
      cy.get(accountPageLocators.oldPassword, { timeout: 10000 }).clear()
      return this }
    clearConfirmPassword(text) {
      cy.get(accountPageLocators.confirmPassword, { timeout: 10000
}).clear()
      return this
    }

    sendChangePassForm(code) {

      if(code != null) {
      cy.intercept('POST',
Cypress.env('passwordChangeAPI')).as('changePass')

      cy.get(accountPageLocators.changeBtn).click()

      cy.wait('@changePass').its('response.statusCode').should('eq', code)
    }
    else {
      cy.get(accountPageLocators.changeBtn).click()
    }
      return this
    }

    clickDelete() {
      cy.get(accountPageLocators.deleteBtn).click({force: true})
      return this
    }

    clickSecondDelete() {
      cy.get(accountPageLocators.secondDelete).click({force: true})
      return this
    }

    typeCode(code) {
      cy.get(accountPageLocators.deletionCode).type(code)
      return this
    }

    finishDeletion() {
       cy.get(accountPageLocators.finalDeleteBtn).click()
       return this
    }

    checkError(message) {
      cy.get(accountPageLocators.errorField).contains(message)
      return this;
    }


}
```

A.2 MainPage.js

```
import moment from 'moment';

const mainPageLocators = {

  login: '.authBar_login__2AA8y',

  mainMenu: '.headerBar_content__2rTO9',
  navMenu: '.burgerMenu_options__3bnpy',
  menuItem: '.burgerMenu_item__2Stn5',
  navigation: '.MuiButtonBase-root',
  logout: '.burgerMenu_item__2Stn5',
  dateBar: '.headerBar_clock__VAzsN',
  loginForm: '.loginForm_wrapper__1HZhC',
  signupForm: '.registerForm_wrapper__2-Ekk',
  signup: '.authBar_signUp__23DKW',
  forgotPassword: '.loginForm_loginLink__BTVd8',
  emailInput: 'input[name=username]',
  passwordInput: 'input[name=password]',
  dateOfBirth: 'input[name=dateOfBirth]',
  resetEmailInput: 'input[name=email]',
  aliasInput: 'input[name=alias]',
  checkBox: 'input[name=terms]',
  notification: '.notification_contentWrapper__3BE2x',
  dateOfBirthWindow: '.MuiPickersBasePicker-container',
  errorMessage: '.MuiFormHelperText-root',
  checkboxError: '.registerForm_termsError__2ISRL',
  loginButton: '.loginForm_button__3uEJX',
  signupBtn: '.registerForm_button__2BUFx',
  recoverBtn: '.recoverForm_button__-rR7O',

  header: '.header_header__23cM1 ',
  promoBaner: '.promoBanner_content__2z54N',

  normalEventItem: '.leagueItem_item__2gKqC',
  eventItem: '.marketItem_item__3WSKc',

  eventBanner: '.nextUpFixture_wrapper__3gRJR',
  betsBlock: '.fixtureLabel_wrapper__RqgEf',
  jpotsBetsBlock: '.marketLabel_wrapper__UQyQd',
  bets: '.fixtureLabel_item__QePNN',
  jpotBets: '.marketLabel_item__3XckN',
  betInput: 'input[name=amountValue]',
  betInputButton: '.couponItem_wrapper__2dcDB',
  makeABetBlock: '.coupons_wrapper__H2rBW',
  betslip: '.betslipCounter_wrapper__XSED1',
  sportTypeBtn: '.sportbutton_wrapper__lQtgg',
  jpotType: '.marketHeader_typeInfo__1D0xj',
  jpotPoolSize: '.marketHeader_price__22mtx',

  betNowBtn: '.coupons_button__jVvUD',
  betCoupon: '#betCoupon',
  closeCoupon: '.betCoupon_closeIcon__25YZL',

  sideBetBar: '.coupons_labels__2W4kB',
  ideBarBet: '.selectedBetLabel_wrapper__ihM-p',
  sideBarBetCross: '.selectedBetLabel_closeBtn__1Pw1_',
```

```
  loadMore: '.loadMoreBtn_loadMore__17iSB',


  gameTypeColorFilled: 'rgb(255, 210, 0)',
  gameTypeColorEmpty: 'rgb(255, 255, 255)',
  colorFilled: 'rgb(95, 208, 240)',
  colorEmpty: 'rgb(237, 237, 237)',
  leagueBar:
'.marketItem_wrapper__aV9ha.marketItem_selected__xNhA8',

    gameTypesList: '.sportsbar_wrapper__3lmO4',
    leagueGame: ".leagueItem_item__2gKqC",
    gameDate: '.leagueItem_date__Mm6-I',


    filterAndSearchBar: '.headerBar_content__2rTO9',
    searchClick: '.searchInput_input__2P29O',
    searchInput: '.searchInput_opened__2rx8x',
    searchResult: '.searchItem_wrapper__w9Lwz',
    searchFilter: '.filterchip_labelWrapper__lXj2h',
    closeSearch: '.MuiSvgIcon-root',
    filterWraper: ".MuiButtonBase-root",
    filterButton: ".MuiButton-label",


    closePopup: '.loginPromoPopup_exitButton__2PgeO',

}

export class MainPage {

    closePopUp () {
       cy.get(mainPageLocators.clsoePopup, { timeout: 10000 }).click()
       return this;
    }

    isLoginVisible() {
      cy.get(mainPageLocators.loginForm).should('exist')
      return this
    }

    isSignupVisible() {
      cy.get(mainPageLocators.signupForm).should('exist')
      return this
    }

    isMainMenuVisible() {
      cy.get(mainPageLocators.mainMenu).should('be.visible')
      return this
    }

    isNavMenuVisible() {
      cy.get(mainPageLocators.navMenu).should('be.visible')
      return this
    }

    isNavMenuNotVisible() {
      cy.get(mainPageLocators.navMenu).should('not.be.visible')
      return this
    }

    clickNavMenuComponents() {

cy.get(mainPageLocators.menuItem).contains('Account').click().url().sho
uld('contain', Cypress.env('homePage'))
      cy.go('back')
      cy.get('.loader_wrapper__3lfjj').should('not.not.exist')


cy.get(mainPageLocators.menuItem).contains('Wallet').click().url().shoul
d('contain', Cypress.env('walletPage'))
      cy.go('back')
      cy.get('.loader_wrapper__3lfjj').should('not.not.exist')

      cy.get(mainPageLocators.menuItem).contains('My
Bets').click().url().should('contain', Cypress.env('myBetsPage'))
      cy.go('back')
      cy.get('.loader_wrapper__3lfjj').should('not.not.exist')


cy.get(mainPageLocators.menuItem).contains('Perks').click().url().should
('contain', Cypress.env('perksPage'))
      cy.go('back')
      cy.get('.loader_wrapper__3lfjj').should('not.not.exist')

      cy.get(mainPageLocators.menuItem).contains('Info
Centre').click().url().should('contain', Cypress.env('infoPage'))
      cy.go('back')
      cy.get('.loader_wrapper__3lfjj').should('not.not.exist')

      cy.get(mainPageLocators.menuItem).contains('Privacy &
Terms').click().url().should('contain', Cypress.env('privacyPage'))
      return this;
    }

    clickNavMenuComponentsUnlogged() {

cy.get(mainPageLocators.menuItem).contains('Account').should('not.exist
')

cy.get(mainPageLocators.menuItem).contains('Wallet').should('not.exist')
      cy.get(mainPageLocators.menuItem).contains('My
Bets').should('not.exist')

cy.get(mainPageLocators.menuItem).contains('Perks').click().url().should
('contain', Cypress.env('perksPage'))
      cy.go('back')
      cy.get('.loader_wrapper__3lfjj').should('not.not.exist')

      this.clickNav()
      cy.get(mainPageLocators.menuItem).contains('Info
Centre').click().url().should('contain', Cypress.env('infoPage'))
      cy.go('back')
      cy.get('.loader_wrapper__3lfjj').should('not.not.exist')
      this.clickNav()
      cy.get(mainPageLocators.menuItem).contains('Privacy &
Terms').click().url().should('contain', Cypress.env('privacyPage'))
      return this;
    }
    checkMainPageElements(text) {
      if(text == "unlogged") {
        cy.get('.unloggedHeader_header__9mnSv').should('exist')
        cy.get(mainPageLocators.login).should('exist')
        cy.get(mainPageLocators.signup).should('exist')
      }
      else if(text == 'logged') {
        cy.get(mainPageLocators.header).should('exist')
      }

cy.get(mainPageLocators.searchClick).scrollIntoView().should('exist')
      cy.get(mainPageLocators.betsBlock).first().should('be.visible')
      //cy.get(mainPageLocators.leagueGame).should('be.visible')
      cy.get(mainPageLocators.gameTypesList).should('be.visible')
      cy.get(mainPageLocators.filterAndSearchBar).should('be.visible')
      cy.get(mainPageLocators.makeABetBlock).should('be.visible')

      return this
    }


    clickNav() {
      cy.get(mainPageLocators.navigation).first().click()
      return this
    }


    clickAllSportTypes() {
      cy.get(mainPageLocators.sportTypeBtn).each((item, index) => {

        if(index !== 0) {
          cy.wrap(item).should('have.css', 'background-color',
mainPageLocators.gameTypeColorEmpty)
        }
        cy.wrap(item).click().should('have.css', 'background-color',
mainPageLocators.gameTypeColorFilled)

      })
      return this;
    }

    clickJpots() {
      cy.get(mainPageLocators.sportTypeBtn).contains('jpots').click()
      cy.get(mainPageLocators.leagueBar).should('contain', 'JPOTS')
      cy.wait(1000)
      return this;
    }

    testAllJpots(inputNumber) {
      cy.get(mainPageLocators.eventItem).filter(':visible').each((item,
index) => {
        cy.wrap(item).click()
```

```
        this.clickAllJpotBets()
        cy.get(mainPageLocators.jpotType).invoke('text').then((text) => {
        if(text.includes("GUARANTEED")) {
        //ADD ONCE GDT IS ADDED
        }

        else if (text.includes("ADDED")) {

          cy.get(mainPageLocators.jpotPoolSize).invoke('text').then((text)
=> {

            const num = text.match(/[+-]?\d+(\.\d+)?/g)[0]
            const pool = parseFloat(num);
            this.clearBet()

cy.get(mainPageLocators.betInput).type(inputNumber).should('have.valu
e', inputNumber)


cy.get('.coupons_potentialPay__10NSl').invoke('text').then((winningText)
=> {
              const winningNum = winningText.match(/[+-]?\d+(\.\d+)?/g)[0]
              const winnings = parseFloat(winningNum);


expect(winningNum).contains(((inputNumber+pool).toString()).slice(0, -
1))
            })
          })
        }
      })
    })
    }

    clickBanner() {
        cy.get('.fixtureHeader_player__1Ahm1')

cy.get('.nextUpFixture_player__2nqvM').eq(0).invoke('text').then((teamN
ame) => {
          cy.get('.nextUpFixture_player__2nqvM').eq(0).click({force: true})
          cy.get('.fixtureHeader_player__1Ahm1').first().should('contain',
teamName)
        })
        return this
    }

    clickEvent(number) {
      cy.get(mainPageLocators.normalEventItem).eq(number).click()
      return this
    }

    clickBetslip() {
        cy.scrollTo('top')

cy.get(mainPageLocators.betslip).should('exist').should('be.visible').click(
)
        cy.get(mainPageLocators.betInputButton).should('be.visible')

    }

    ClickAllBets() {
      cy.get(mainPageLocators.loadMore, {timeout: 10000}).click()
      cy.get(mainPageLocators.betsBlock).each((item, index) => {
        if(index < 10) {

cy.wrap(item).find(mainPageLocators.bets).first().click().should('have.css
', 'background-color', mainPageLocators.colorFilled)
        }
        else {

cy.wrap(item).find(mainPageLocators.bets).first().click().should('have.css
', 'background-color', mainPageLocators.colorEmpty)
        }
      })
      return this;
    }

    clickAllJpotBets() {
      // cy.get(mainPageLocators.loadMore).click()
      cy.get(mainPageLocators.jpotsBetsBlock).filter(':visible').each((item,
index) => {
        if(index < 10) {

cy.wrap(item).find(mainPageLocators.jpotBets).first().click().should('hav
e.css', 'background-color', mainPageLocators.colorFilled)
        }
        else {

cy.wrap(item).find(mainPageLocators.jpotBets).first().click().should('hav
e.css', 'background-color', mainPageLocators.colorEmpty)
        }
      })
      return this;
    }

    deselectAllBets() {
      cy.get(mainPageLocators.betsBlock).each((item, index) => {
        if(index < 10) {

cy.wrap(item).find(mainPageLocators.bets).first().click().should('have.css
', 'background-color', mainPageLocators.colorEmpty)
        }
      })
      cy.get(mainPageLocators.sideBetBar).should('not.exist')
      return this;
    }
    checkAllDeselected() {
      cy.get(mainPageLocators.betsBlock).each((item, index) => {
        cy.wrap(item).find(mainPageLocators.bets).should('have.css',
'background-color', mainPageLocators.colorEmpty)
        cy.get(mainPageLocators.sideBetBar).should('not.exist')
      })
      return this;
    }
    checkAllJpotsDeselected() {
      cy.get(mainPageLocators.sideBetBar).should('not.exist')
      return this;
    }

    deselectAllBetsSideBar() {

cy.get(mainPageLocators.sideBetBar).find(mainPageLocators.sideBarBet
).each((item, index) => {

cy.wrap(item).find(mainPageLocators.sideBarBetCross).first().click()
      })
      return this;
    }

    clickFirstBetWhite() {
      cy.get(mainPageLocators.bets, { timeout: 10000
}).first().click().should('have.css', 'background-color',
mainPageLocators.colorFilled)
      return this;
    }

    clickFirstBetBlue() {
      cy.get(mainPageLocators.bets).first().click().should('have.css',
'background-color', mainPageLocators.colorEmpty)
      return this;
    }

    clearBet() {
      cy.get(mainPageLocators.betInput).clear()
      return this;
    }

    inputBet(number) {
      cy.get(mainPageLocators.betInput).type(number).should('have.value',
number)
      return this;
    }

    clickBetInputButton(value) {
      cy.get(mainPageLocators.betInputButton).contains(value).click()
      return this;
    }

    isBetInputButtonDisabled() {
      cy.get(mainPageLocators.betNowBtn).should('be.disabled')
      return this;
    }

    inputBetWithoutCheck(number) {
      cy.get(mainPageLocators.betInput).type(number)
      return this;
    }

    checkBetInput(value) {
      cy.get(mainPageLocators.betInput).should('have.value', value)
      return this;
    }
```

```
    /*
    makeABet(code) {
      cy.intercept('POST', Cypress.env('placeBetAPI')).as('makeABet')
      cy.get(mainPageLocators.betNowBtn).click()
      cy.wait('@makeABet').its('response.statusCode').should('contain',
code)
      return this
    }
    */

    makeABet() {
      cy.get(mainPageLocators.betNowBtn).click()
      return this
    }

    checkCoupon(text) {
      cy.get(mainPageLocators.betCoupon).contains(text)
      return this
    }

    closeCoupon() {
      cy.get(mainPageLocators.closeCoupon).click()
      return this
    }


    clickFilter(text) {

cy.get(mainPageLocators.filterWraper).get(mainPageLocators.filterButto
n).contains(text).click({force: true})
      return this;
    }

    checkTodayFilter() {
      const today = new Date();
      var date = today.getFullYear()+'-'+(today.getMonth()+1)+'-
'+today.getDate();
      var day = today.getDate();
      var month = today.getMonth()+1;

      var currentDate = "";
      if(month>9) {
        var currentDate = day + '/' + month;
      }
      else {
        var currentDate = day + '/0' + month;
      }

      cy.get(mainPageLocators.leagueGame).each((item, index) => {
         cy.wrap(item).should('contain.text', currentDate)
      })
      return this;
    }

    checkDate() {
      const today = new Date();
      var day = today.getDate();
      var month = today.getMonth()+1;

      var currentDate = "";
    if(day < 10) { day = '0'+day}
      if(month>9) {
        var currentDate =  month + '/' + day;
      }
      else {
        var currentDate = '0'+month+'/'+day;
      }
      cy.get(mainPageLocators.dateBar).should('contain.text', currentDate)
    }


    checkThisWeekFilter() {
      const today = moment()

      cy.get(mainPageLocators.leagueGame).each((item, index) => {
         cy.wrap(item).find('.leagueItem_date__Mm6-
I').invoke('text').then((text) => {
           var gamedate = text.split(" ")[0]
           const today = moment()
           assert.isTrue(moment(gamedate).isSame(new Date(), 'week'))

         })
    })
      return this;
    }

    checkThisMonthFilter() {
      const today = moment()

      cy.get(mainPageLocators.leagueGame).each((item, index) => {
         cy.wrap(item).find('.leagueItem_date__Mm6-
I').invoke('text').then((text) => {
           var gamedate = text.split(" ")[0]
           const today = moment()
           assert.isTrue(moment(gamedate).isSame(new Date(), 'month'))

         })
    })
      return this;
    }

    searchClick() {
      cy.get(mainPageLocators.searchClick).click({force: true})
      return this;
    }

    searchInput(text) {

cy.get(mainPageLocators.searchInput).type(text).wait(500).should('have.
value', text)
      cy.get(mainPageLocators.searchInput).type(' ')
      return this;
    }

    checkSearchResult(text) {
      cy.get(mainPageLocators.searchResult).each(($el) => {
         cy.wrap($el).contains(text, { matchCase: false })
        })
        return this;
    }

    isSearchFilterVisibe() {
      cy.get(mainPageLocators.searchFilter).should('exist')
      return this
    }

    clickSearchFilter() {
      cy.get(mainPageLocators.searchFilter).first().invoke('text').then((text)
=> {
         cy.get(mainPageLocators.searchFilter).click({force: true})
         this.checkSearchResult(text)
        })
    return this
    }


    checkNoSearchResult(text, date) {
      cy.get(mainPageLocators.searchResult).should('not.exist')
      cy.get('.searchResults_wrapper__1-nRI').should('contain.text', 'No
results for this at present')
        return this;
    }

    closeSearch() {
      cy.get(mainPageLocators.closeSearch).click({force: true})
      return this;
    }

    loginNoCredentials() {
      cy.get(mainPageLocators.login).click()
      cy.get(mainPageLocators.emailInput).clear()
      cy.get(mainPageLocators.passwordInput).clear()
      cy.get(mainPageLocators.loginButton).click()
      cy.get(mainPageLocators.errorMessage).contains('Enter valid email')
      cy.get(mainPageLocators.errorMessage).contains('Your password
must be at least 8')

        return this;
    }


    loginNoPassword() {
      cy.get(mainPageLocators.login).click()

cy.get(mainPageLocators.emailInput).type(Cypress.env('email')).should('
have.value', Cypress.env('email'))
      cy.get(mainPageLocators.passwordInput).clear()
      cy.get(mainPageLocators.loginButton).click()
      cy.get(mainPageLocators.errorMessage).should('contain.text', 'Your
password must be at least 8')
```

```
    return this;
  }

  loginInvalidPassword() {
    cy.get(mainPageLocators.login).click()

cy.get(mainPageLocators.emailInput).type(Cypress.env('email')).should('
have.value', Cypress.env('email'))

cy.get(mainPageLocators.passwordInput).type('123').should('have.value',
'123')
    cy.get(mainPageLocators.loginButton).click()
    cy.get(mainPageLocators.errorMessage).should('contain.text', 'Your
password must be at least 8')
    return this;
  }

  loginWrongPassword() {
    cy.get(mainPageLocators.login).click()

cy.get(mainPageLocators.emailInput).type(Cypress.env('email')).should('
have.value', Cypress.env('email'))

cy.get(mainPageLocators.passwordInput).type('Gaben12345678').should(
'have.value', 'Gaben12345678')
    cy.get(mainPageLocators.loginButton).click()
    cy.get(mainPageLocators.notification).should('contain.text', 'Login
failed')
    return this;
  }

  loginNoEmail() {
     cy.get(mainPageLocators.login).click()

cy.get(mainPageLocators.passwordInput).type(Cypress.env('password')).s
hould('have.value', Cypress.env('password'))
     cy.get(mainPageLocators.emailInput).clear()
     cy.get(mainPageLocators.loginButton).click()
     cy.get(mainPageLocators.errorMessage).should('contain.text', 'Enter
valid email')
     return this;
  }

  loginInvalidEmail() {
    cy.get(mainPageLocators.login).click()
    cy.get(mainPageLocators.emailInput).type('123').should('have.value',
'123')

cy.get(mainPageLocators.passwordInput).type(Cypress.env('password')).s
hould('have.value', Cypress.env('password'))
    cy.get(mainPageLocators.loginButton).click()
    cy.get(mainPageLocators.errorMessage).should('contain.text', 'Enter
valid email')
    return this;
  }

  login() {
    cy.get(mainPageLocators.login).click()

cy.get(mainPageLocators.emailInput).type(Cypress.env('email')).wait(50
0).should('have.value', Cypress.env('email'))
    cy.wait(200)

cy.get(mainPageLocators.passwordInput).type(Cypress.env('password')).
wait(500).should('have.value', Cypress.env('password'))
    cy.intercept('POST', Cypress.env('loginAPI')).as('login')
    cy.intercept('GET', Cypress.env('balanceAPI')).as('balance')
    cy.get(mainPageLocators.loginButton).click()

    cy.wait('@login', { timeout: 10000
}).its('response.statusCode').should('eq', 200)
    cy.wait('@balance').its('response.statusCode').should('eq', 200)
    cy.contains("Logged in successfully")
    cy.get('.loader_wrapper__3lfjj').should('not.not.exist')
    return this;
  }

  logout() {
    cy.get(mainPageLocators.logout).contains('Log out').click()
    cy.contains('Login')
    return this;
  }

  clickSignUp() {
    cy.get(mainPageLocators.signup).click()
    this.isSignupVisible()
    cy.contains('Create')
    return this
  }


  clickForgotPassword() {
    cy.get(mainPageLocators.forgotPassword).click()
    cy.contains('Recover your')
    return this
  }

  clickDateOfBirth() {
    cy.get(mainPageLocators.dateOfBirth).click()
    cy.get(mainPageLocators.dateOfBirthWindow).should('be.visable')
    return this
  }


  dateOfBirthChange() {
    cy.get('.MuiTouchRipple-root').first().click()
    cy.get('.MuiPickersCalendar-week').contains('1').last().click()
    return this
  }

  clearUsername() {
    cy.get(mainPageLocators.emailInput).focus().clear()
    return this
  }

  clearResetEmail() {
    cy.get(mainPageLocators.resetEmailInput).focus().clear()
    return this
  }

  typeResetEmail(text) {

cy.get(mainPageLocators.resetEmailInput).type(text).should('have.value',
text)
    return this
  }

  clearPassword() {
    cy.get(mainPageLocators.passwordInput).focus().clear()
    return this
  }

  clearAlias() {
    cy.get(mainPageLocators.aliasInput).clear()
    return this
  }

  openLogin() {
     cy.get(mainPageLocators.login).click()
     this.isLoginVisible()
     return this
    }
    typeEmail(text) {
      cy.get(mainPageLocators.emailInput).type(text).should('have.value',
text)
      return this
    }

  typePassword(text) {

cy.get(mainPageLocators.passwordInput).type(text).should('have.value',
text)
    return this
  }

 clickLogin(text) {
   cy.intercept('POST', Cypress.env('loginAPI')).as('login')
   cy.intercept('GET', Cypress.env('balanceAPI')).as('balance')
   cy.get(mainPageLocators.loginButton).click()

   cy.wait('@login', { timeout: 10000
}).its('response.statusCode').should('eq', 200)
   cy.wait('@balance').its('response.statusCode').should('eq', 200)
   cy.contains("Logged in successfully")
   cy.get('.loader_wrapper__3lfjj').should('not.not.exist')

   typeAlias (text) {
     cy.get(mainPageLocators.aliasInput).type(text).should('have.value',
text)
    return this
   }
```

```
    checkEmailError(text) {

cy.get(mainPageLocators.errorMessage).contains('Email').should('contain
.text', text)
      return this
    }

    checkPasswordError(text) {
      cy.get(mainPageLocators.errorMessage).contains('password
').should('contain.text', text)
      return this
    }

    checkAliasError(text) {

cy.get(mainPageLocators.errorMessage).contains('alias').should('contain.t
ext', text)
      return this
    }


    checkError(text)  {
      cy.get(mainPageLocators.errorMessage).should('contain.text', text)
      return this
   }

    checkboxCheck() {
      cy.get(mainPageLocators.checkBox).check()
      return this
    }

    checkboxError() {
      cy.get(mainPageLocators.checkboxError).should('contain.text',
'Terms and Privacy Policy checkbox must be checked')
      return this
    }
    checkNotification(text) {
      cy.get(mainPageLocators.notification).should('contain.text', text)
      return this }
```

```
    clickSignUpBtn() {
      cy.get(mainPageLocators.signupBtn).click()
      return this
    }

    clickRecoverBtn() {
      cy.get(mainPageLocators.recoverBtn).click()
      return this
    }


    checkOddsType(type) {
      var regex = /[1-9]*/g

      if(type == "Fraction") {
        regex =  /[1-9][1-9]*(\/[1-9][1-9]*)?/
      }
      else if (type == "Decimal") {
        regex =  /^\d+\.\d{0,2}$/
      }
      else if (type == "American") {
        regex = /^-?\d*\.{0,1}\d+$/
      }
      cy.get(mainPageLocators.bets).each(($el) => {
          cy.wrap($el).find('p').last().invoke('text').then((text) => {
              text = text.replace(`^`, '')
              console.log(text)
              expect(text).to.match(regex)
          })
      })
      return this;
}

  elementsAreVisible() {
    cy.get(mainPageLocators.normalEventItem , { timeout: 10000 })
    cy.get(mainPageLocators.bets , { timeout: 10000 })
    return this
  }
}
```

A.3 Perks.js

```
const perksPageLocators = {
    header: '.header_header__23cM1',

    navBar: '.breadcrumbs_wrapper__2FWrj',

    contentColumn: '.perks_column__23AmI',
    perksTitle: '.tabItem_wrapper__3qAcT',

    perksContent: '.perks_content__2XdG9',
    challengeBtn: '.rewardsChallengeItem_wrapper__3zcdg',
    challengeBtnLogout: '.rewardsItem_content__2PALi',
    challengeItemLogout: '.challenges_challenges__3rIvK',

    luckyRolls: '.rewardsItem_content__2PALi',
    leaderboard: '.rewardsLeaderboardsList_content__1Np5N',


    columnTitle: '.perks_title__1bVhj',
    banners: '.perks_promobanners__hG0LZ',
    rewardList: '.perks_rewardsContentWrapper__su-Ik',
    bannerItem: '.bannerItem_wrapper__3V-MD',

    promoBanner: '.promoBanner_content__2z54N',
    readMore: '.perkItem_text__3unbi',
    promoPopUp: '.MuiDialogContent-root',
    closePromo: '.promoBanner_exit__NunIb',



    rollPlay: '.rollsBtn_wrapper__tThbG',
    openedImage: '.roll_open__1X3zh',
}


export class PerksPage {
```

```
  checkPerksPageElements() {

cy.get(perksPageLocators.header).scrollIntoView().should('exist').should(
'be.visible')

    cy.get(perksPageLocators.perksContent).should('exist')

    cy.get(perksPageLocators.perksTitle).contains('What`s on?').click()

cy.get(perksPageLocators.bannerItem).first().scrollIntoView().should('exi
st').should('be.visible')

    cy.get(perksPageLocators.perksTitle).contains('Rewards').click()

cy.get(perksPageLocators.rewardList).scrollIntoView().should('exist').sho
uld('be.visible')

    return this
  }

  clickChallange() {

cy.get(perksPageLocators.challengeBtn).first().click().url().should('contai
n', Cypress.env('challengePage')+'/1')
    return this
  }

  clickChallangesLoggedOut() {

cy.get(perksPageLocators.challengeBtnLogout).contains('Challenge').scr
ollIntoView().click({force: true})

cy.get('.loginForm_wrapper__1HZhC').should('exist').should('be.visible')
    return this
  }
```

```
  acceptChallengeLoggedOut() {

cy.get(perksPageLocators.challengeItemLogout).first().scrollIntoView().c
lick().url().should('contain', Cypress.env('challengePage')+'/1')
    cy.wait(2000)
    cy.contains('Accept challenge').click()
    return this;
  }

  clickLeaderboard() {
    cy.wait(2000)
    cy.get(perksPageLocators.leaderboard).click().url().should('contain',
Cypress.env('leaderboardPage'))
    return this
  }
  clickNavbarPerks() {

cy.get(perksPageLocators.navBar).contains("Perks").click().url().should('
contain', Cypress.env('perksPage'))
    return this
  }
  clickNavbarHome() {

cy.get(perksPageLocators.navBar).contains("Home").click().url().should('
contain', Cypress.env('homePage'))
    return this
  }
  clickLuckyRolls() {
    cy.get(perksPageLocators.luckyRolls).click().url().should('contain',
Cypress.env('luckyRollsPage'))
    return this
  }
  playLuckyRolls() {
    cy.wait(5000)
    cy.get(perksPageLocators.rollPlay).scrollIntoView({ offset: { top: -
150, left: 0 } }).click()
//    cy.intercept('GET', Cypress.env('balanceAPI')).as('PlayRolls')
//    cy.wait('@PlayRolls').its('response.statusCode').should('eq', 200)
    cy.wait(3000)
    cy.get(perksPageLocators.openedImage).each((item, index) => {
      cy.wrap(item).should('exist').should('be.visible')
    })
     return this
  }
  clickBanner() {
    cy.get(perksPageLocators.perksTitle).contains('What`s on?').click()

cy.get(perksPageLocators.bannerItem).first().should('exist').should('be.vis
ible').scrollIntoView().click({force: true})

cy.get(perksPageLocators.promoPopUp).should('exist').should('be.visible'
)
     cy.get('.promoBannerModal_button__3970W').contains('Play
Now').click()
     cy.url().should('not.include', 'dublzplay.com/perks')
    return this;
  }}
```

A.4 LoginTests.js

```
import { MainPage } from '../pages/mainPage';


describe('Login tests', () => {

  const mainPage = new MainPage();

  beforeEach(() => {
    cy.visit(Cypress.env('homePage'))
    mainPage.closePopUp()
    cy.wait(1000)
  })

  it('Login Without Credential', () => {
    mainPage.loginNoCredentials()
  })

  it('Login Without Password', () => {
    mainPage.loginNoPassword()
  })

  it('Login Invalid Password', () => {
    mainPage.loginInvalidPassword()
  })

  it('Login Invalid Email', () => {
    mainPage.loginInvalidEmail()
  })

  it('Login Without Email', () => {
    mainPage.loginNoEmail()
  })

  it('Login Wrong Password', () => {
    mainPage.loginWrongPassword()
  })

  it('Login Attempt', () => {
    mainPage.login()
  })
  it('Login Attempt', () => {
    mainPage.openlogin()
.typeEmail('chuzov.d@gmail.com')
    .typePassword('notarealpassword')
    .clickLogin()
  })

  it('Login Using Wrong Password', () => {
    mainPage.openlogin()
    .typeEmail('chuzov.d@gmail.com')
    .typePassword('wrongpassword')
    .clickLogin()
    .checkError('Login failed')
  })

  it('Login Without Credentials', () => {
    mainPage.openlogin()
    .clickLogin()
    .checkError('Enter valid email')
    .checkError('Your password must be at least 8 characters long, must
contain 1 letter, 1 uppercase letter and 1 number')
  })

  it('Logout Attempt', () => {
    mainPage.login().clickNav().logout()
  })

  it('Close login and open signup', () => {
      cy.wait(2000)
      mainPage.clickLogin()
      cy.get('.MuiBackdrop-root').first().click({force: true})
      mainPage.clickSignUp()
  })

  it('Open Forgot Password form', () => {
    cy.wait(2000)
    mainPage.clickLogin().clickForgotPassword()
  })

  it('Close Password Recovery form and open login form', () => {
    cy.wait(2000)
    mainPage.clickLogin().clickForgotPassword()
    cy.get('.MuiBackdrop-root').first().click({force: true})
    mainPage.clickLogin()
```

```
  })

  it('Close Password Recovery form and open SignUp form', () => {
    cy.wait(2000)
    mainPage.clickLogin().clickForgotPassword()
    cy.get('.MuiBackdrop-root').first().click({force: true})
    mainPage.clickSignUp()
  })

  it('Send empty Forgot password form', () => {
    cy.wait(1000)
    mainPage.clickLogin()
    .clickForgotPassword()
    .clearResetEmail()
    .clickRecoverBtn()
    .checkError('is required')
  })

  it('Password recovery with invalid email', () => {
    cy.wait(1000)
    mainPage.clickLogin()
    .clickForgotPassword()
    .typeResetEmail('124343kj44')
    .clickRecoverBtn()
    .checkError('Enter valid email')
  })

  it('Password recovery with non existant email', () => {
    cy.wait(1000)
    mainPage.clickLogin()
    .clickForgotPassword()
    .typeResetEmail('absolutelyRandomEmaillllll@gmail.com')
    .clickRecoverBtn()
    .checkNotification('email')
  })

  it('Send Password Recovery form with existing email', () => {
    cy.wait(1000)
    mainPage.clickLogin()
    .clickForgotPassword()
    .typeResetEmail('chuzov.d@gmail.com')
    .clickRecoverBtn()
    .checkNotification('Confirmation code')
  })

  it('Enter invalid code in password recovery form', () => {
    cy.wait(1000)
    mainPage.clickLogin()
    .clickForgotPassword()
    .typeResetEmail('chuzov.d@gmail.com')
    .clickRecoverBtn()
    .checkNotification('Confirmation code')

    cy.get('input[name=password]').type('Gaben54321')
    cy.get('input[name=code]').type('123456')
    mainPage.clickRecoverBtn()
    mainPage.checkNotification('Recover password fail: Incorrect
verification')

  })

  it('Enter invalid password in password recovery form', () => {
    cy.wait(1000)
    mainPage.clickLogin()
    .clickForgotPassword()
    .typeResetEmail('chuzov.d@gmail.com')
    .clickRecoverBtn()

    cy.get('input[name=password]').type('123')
    cy.get('input[name=code]').type('123456')

    cy.contains('Your password must be at least 8 characters long')
  })
})
```

A.5 SignUpTests.js

```
import { MainPage } from '../pages/mainPage';

const randomUser = {
  email: "anotherEmailForTest@gmail.com",
  password: "TestPassword123",
  alias: "newRandomAcc",
}

describe('Sign up tests', () => {

  const mainPage = new MainPage();

  beforeEach(() => {
    cy.visit(Cypress.env('homePage'))
    cy.wait(2000)
    mainPage.closePopUp()
  })

  it('Open login form after opening SignUp form', () => {
      mainPage.clickSignUp()
      cy.get('.MuiBackdrop-root').first().click({force: true})
      mainPage.clickLogin()
    })

  it('Signup Without Credential', () => {
    mainPage.clickSignUp()
    .clearPassword().clearUsername().clearAlias()
    .clickSignUpBtn()
    .checkError("Enter valid email")
    .checkError('Alias field is required')
    .checkError('Your password must be at least 8 characters long')
    .checkboxError()
  })

  it('Signup with invalid email', () => {
    mainPage.clickSignUp()

.typePassword(randomUser.password).typeUsername("123").typeAlias(ra
ndomUser.alias)
    .clickSignUpBtn()
    .checkError('Enter valid email')
    .checkboxError()
  })

  it('Signup with already taken email', () => {
    mainPage.clickSignUp()

.typePassword(randomUser.password).typeUsername("chuzov.d@gmail.
com").typeAlias(randomUser.alias)
    .checkboxCheck()
    .clickSignUpBtn()
    .checkNotification("Account with this email")

  })


  it('Signup with invalid password', () => {
    mainPage.clickSignUp()

.typePassword("123").typeUsername(randomUser.email).typeAlias(rando
mUser.alias)
    .clickSignUpBtn()
    .checkPasswordError('Your password must be at least 8 characters
long')
    .checkboxError()
  })

  it('Signup with too short alias', () => {
    mainPage.clickSignUp()
```

```
.typePassword(randomUser.password).typeUsername(randomUser.email)
.typeAlias('MR')
    .checkboxCheck()
    .clickSignUpBtn()
    .checkError('Your alias must be at least')
  })


  it('Signup with cyrilic letters in password', () => {
    mainPage.clickSignUp()

.typePassword("!@#$A6BDDF233DF").typeUsername(randomUser.ema
il).typeAlias(randomUser.alias)
    .checkboxCheck()
    .clickSignUpBtn()
  })


  it('Signup with invalid alias', () => {
    mainPage.clickSignUp()

.typePassword("Gaben12345").typeUsername(randomUser.email).typeAl
ias('!1   22~  3')
    .checkboxCheck()
    .clickSignUpBtn()
    .checkError('Your alias must not includes whitespaces')
  })

  it('Signup with already taken alias', () => {
    mainPage.clickSignUp()

.typePassword(randomUser.password).typeUsername(randomUser.email)
.typeAlias("Chuzov")
    .checkboxCheck()
    .clickSignUpBtn()
    .checkNotification("already exists with this alias")
  })
  })
```

A.6 ChangePassTests.js

```
import { MainPage } from '../pages/mainPage';
import { AccountPage } from '../pages/account';
describe('Change password form', () => {
  const mainPage = new MainPage();
  const accountPage = new AccountPage();

  beforeEach(() => {
    cy.visit(Cypress.env('homePage'))
    mainPage.closePopUp().login()
    cy.visit(Cypress.env('accountPage'))
    accountPage.clickSecurity()
  })

it('Change password without data', () => {
  accountPage.clearNewPassword()
  .clearConfirmPassword()
  .clearOldPassword()
  .sendChangePassForm(null)
  .checkError('Your password must be at least 8 characters long')
  .checkError('Old password field is required')
})

it('Change password with invalid new password', () => {

  accountPage.typeNewPassword("123456789")
  .typeConfirmPassword("123456789")
  .typeOldPassword(Cypress.env('password'))
  .sendChangePassForm(null)
  .checkError('Your password must be at least 8 characters long, must
contain 1 letter, 1 uppercase letter and 1 number')
})
it('Change password with wrong confirm password field', () => {
  accountPage.clearNewPassword()
  .clearConfirmPassword()
  .clearOldPassword()
  .typeNewPassword("Gaben321")
  .typeConfirmPassword("Gaben1234567")
  .typeOldPassword(Cypress.env('password'))
  .sendChangePassForm(null)
  .checkError('Please make sure your passwords match!')
})

it('Change password with wrong current password', () => {
  accountPage.clearNewPassword()
  .clearConfirmPassword()
  .clearOldPassword()
  .typeNewPassword("Gaben321")
  .typeConfirmPassword("Gaben321")
  .typeOldPassword('Gaben123321123dfjd')
  .sendChangePassForm(400)
  .checkNotification('Incorrect current password')
})

it('Change password with correct data', () => {
  accountPage.clearNewPassword()
  .clearConfirmPassword()
  .clearOldPassword()
  .typeNewPassword(Cypress.env('password'))
  .typeConfirmPassword(Cypress.env('password'))
  .typeOldPassword(Cypress.env('password'))
  .sendChangePassForm(200)
  .checkNotification('Password changed')
})

})
```

A.7 BetsTests.js

```
import { MainPage } from '../pages/mainPage';
describe('Opperations with bets', () => {
  const mainPage = new MainPage();
  beforeEach(() => {
    cy.visit(Cypress.env('homePage'))
    mainPage.closePopUp()
  })

  it('Stage selection', () => {
    mainPage.clickFirstBetWhite()
    .clickFirstBetBlue()
    .clickFirstBetWhite()
  })

  it('Banner game selection', () => {
```

```
  mainPage.clickBanner()
 })

 it('Select and deselect all (10+) stages on a game', () => {
  mainPage.ClickAllBets()
  .deselectAllBets()
 })

 it('Test betslip button', () => {
  mainPage.login()
  mainPage.clickFirstBetWhite()
  cy.scrollTo('top')
  mainPage.clickBetslip()
 })

 it('Click all sport types', () => {
  mainPage.clickAllSportTypes()
  .clickAllSportTypes()
 })

 it('Test side bar bet deselection', () => {
  mainPage.ClickAllBets()
  .deselectAllBetsSideBar()
  .checkAllDeselected()
 })


 it('Test stages deselection after opening jpots', () => {
  mainPage.ClickAllBets()
  .clickJpots()
  .checkAllJpotsDeselected()
 })

 it('Test jpot stages deselection after opening normal game', () => {
  mainPage.clickJpots()
  .clickAllJpotBets()
  .clickAllSportTypes()
  .clickJpots()
  .checkAllJpotsDeselected()
 })

 it('Very small "Search" test', () => {
    cy.get('.leagueItem_competitors__28d-
e').first().invoke('text').then((text) => {

      mainPage.searchClick()
      let changedText = text.split(" ")[0]
      if(changedText.length < 3) {
        changedText = text.split(" ")[1]
      }

       mainPage.searchClick()

mainPage.searchInput(changedText).checkSearchResult(changedText).cl
oseSearch()
      })
  })


  it('Is search history saved after closing search results', () => {
      cy.get('.leagueItem_competitors__28d-
e').first().invoke('text').then((text) => {
         mainPage.searchClick()
         let changedText = text.split(" ")[0]
         if(changedText.length < 3) {
           changedText = text.split(" ")[1]
         }

mainPage.searchInput(changedText).checkSearchResult(changedText).cl
oseSearch().isSearchFilterVisibe()
       })
    })

    it('Test no search results', () => {
         mainPage.searchClick()
         mainPage.searchInput('fihfg').checkNoSearchResult()
    })

    it('Click filter from search history', () => {
      cy.get('.leagueItem_competitors__28d-
e').first().invoke('text').then((text) => {
         let changedtext = text.split(" ")[0]
         if(changedtext.length < 3) {
           changedtext = text.split(" ")[1]
         }
         mainPage.searchClick()
         .searchInput(changedtext)
         .closeSearch()
         .clickSearchFilter()
      })
    })


 it('Test 10, 25, 100, 200 buttons', () => {
     mainPage.clickBetInputButton("10").checkBetInput("50")
     mainPage.clickBetInputButton("50").checkBetInput("50")
     mainPage.clickBetInputButton("70").checkBetInput("70")
     mainPage.clickBetInputButton("100").checkBetInput("100")
 })


 it('Input 0 DBL as a bet value', () => {
  const inputNumber = 0

mainPage.clearBet().inputBet(inputNumber).isBetInputButtonDisabled()
  })

  it('Clear bet value', () => {
    mainPage.clearBet().isBetInputButtonDisabled()
   })

  it('Input not allowed symbols in bet input', () => {
    const inputNumber = '123--212'

mainPage.clearBet().inputBetWithoutCheck(inputNumber).checkBetInpu
t('123212')
  })


  it('Input bet value and check expected payout', () => {

    const inputNumber = 100

    mainPage.clearBet().inputBet(inputNumber)
    mainPage.ClickAllBets()
    const text = cy.contains('Total Odds').invoke('text').then((text) => {
      const num = text.match(/[+-]?\d+(\.\d+)?/g)[0]
      const odds = parseFloat(num);
      expect(odds).to.be.gt(0);

      const winningText =
cy.get('.coupons_potentialPay__10NSl').invoke('text').then((winningText)
=> {
        const winningNum = winningText.match(/[+-]?\d+(\.\d+)?/g)[0]

        const winnings = parseInt(winningNum);

expect(winningNum).contains(((Math.round(inputNumber*odds)).toStrin
g()).slice(0, -1))
     });
    });
    })


  it('Test low bet value in jpots', () => {
    const inputNumber = 10
    mainPage.clickJpots()
    .clickAllJpotBets()
    .clearBet()
    .inputBet(inputNumber)
    .isBetInputButtonDisabled()
  })


  it('Test expected payout in jpots (only for added atm)', () => {
    const inputNumber = 100
    mainPage.clickJpots()
    .testAllJpots(inputNumber)
  })

  it('Make a bet with insufficient funds', () => {
     mainPage.login()
     .clickFirstBetWhite()
     .clearBet()
     .inputBet(1000000)
     .makeABet()
     .checkNotification('User has not enough money to place a bet')
  })

  it('Make a bet without selecting stage', () => {
     mainPage.login()
     .clearBet()
```

```
      .inputBet(5)
      .makeABet()
      .checkNotification('You must select at least one stage')
   })
   it('Make a bet logged out', () => {
      mainPage.clearBet()
      .inputBet(200)
      .makeABet()
      .isLoginVisible()
   })
})
describe('Placing Bets', function(){
  const mainPage = new MainPage();
  var dbl = 0;

  it('Make a bet', () => {

    cy.visit(Cypress.env('homePage'))
    mainPage.closePopUp()
    mainPage.login()
    cy.wait(10000)

    const text =
cy.get('.authBar_userName__DTssj').last().invoke('text').then((text) => {
      cy.wait(4000)
      var num = text.match(/[+-]?\d+(\.\d+)?/g)[0]
      dbl = parseFloat(num);
      cy.wait(4000)
    })
    cy.intercept('POST', Cypress.env('placeBetAPI')).as('placeABet')
      mainPage.clickFirstBetWhite()
      .clearBet()
      .inputBet(Cypress.env('betValue'))
      .makeABet()
      .checkCoupon("Your bet has been placed!")
      .checkCoupon(Cypress.env('betValue')+'.00 DBL')
      cy.wait('@placeABet').its('response.statusCode').should('eq', 200)
  })
  it('Check bet coupon', () => {
    var date, time, gameName;
    cy.get('.leagueItem_date__Mm6-I').first().invoke('text').then((text) =>
{
      date = text.split(" ")[0]
      time = text.split(" ")[1]
      cy.get('.leagueItem_competitors__28d-
e').first().invoke('text').then((gameText) => {
          const team1 = gameText.split(" at ")[0]
          const team2 = gameText.split(" at ")[1]

          mainPage.checkCoupon(date).checkCoupon(time)
        })
    })
  })
  it('Check DBL after the bet', () => {
    mainPage.closeCoupon()
    cy.get('.authBar_userName__DTssj').last().invoke('text').then((text) =>
{
      const num = text.match(/[+-]?\d+(\.\d+)?/g)[0]
      const dblAfterBet = parseFloat(num);
      const betAmmount = parseFloat(Cypress.env('betValue'))
      expect(dbl-betAmmount).to.equal(dblAfterBet)
    })
  })
})
```

```
describe('Place a bet with stages from different games', function(){

  const mainPage = new MainPage();
  var dbl = 0;

  it('Make a bet', () => {
    cy.visit(Cypress.env('homePage'))
    cy.wait(2000);
    mainPage.closePopUp()
    mainPage.login()
    cy.wait(2000);

    const text =
cy.get('.authBar_userName__DTssj').last().invoke('text').then((text) => {
      cy.wait(4000);
      var num = text.match(/[+-]?\d+(\.\d+)?/g)[0]
      dbl = parseFloat(num);
      cy.wait(4000)
    })
    cy.intercept('POST', Cypress.env('placeBetAPI')).as('placeABet')

      mainPage.clickFirstBetWhite()
    //  .clickEvent(1)
    //  .clickFirstBetWhite()
      .clearBet()
      .inputBet(Cypress.env('betValue'))
      .makeABet()
      .checkCoupon("Your bet has been placed!")
      .checkCoupon(Cypress.env('betValue')+'.00 DBL')
      cy.wait('@placeABet').its('response.statusCode').should('eq', 200)
  })
  it('Check bet coupon', () => {
    var date, time, gameName;
    cy.get('.leagueItem_date__Mm6-I').first().invoke('text').then((text) =>
{
      date = text.split(" ")[0]
      time = text.split(" ")[1]

      cy.get('.leagueItem_competitors__28d-
e').first().invoke('text').then((gameText) => {
          const team1 = gameText.split(" at ")[0]
          const team2 = gameText.split(" at ")[1]


          mainPage.checkCoupon(date)
          .checkCoupon(time)
        })
    })
  })
  it('Check DBL after the bet', () => {
    mainPage.closeCoupon()
    cy.get('.authBar_userName__DTssj').last().invoke('text').then((text) =>
{
      const num = text.match(/[+-]?\d+(\.\d+)?/g)[0]
      const dblAfterBet = parseFloat(num);
      const betAmmount = parseFloat(Cypress.env('betValue'))
      expect(dbl-betAmmount).to.equal(dblAfterBet)

    })
  })
})
```

A.8 AccountPageTests.js

```
import { MainPage } from '../pages/mainPage';
import { AccountPage } from '../pages/account';


describe('Test actions on Account page', () => {



  const mainPage = new MainPage();
  const accountPage = new AccountPage();

  beforeEach(() => {
    cy.visit(Cypress.env('homePage'))
    mainPage.closePopUp()
  })

  it('Change alias', () => {
    mainPage.login()
    cy.visit(Cypress.env('accountPage'))
    accountPage.clearAlias()
    .typeAlias('ChuzovTest2')
    .changeAlias()
    .verifyAlias()
```

```
  })

  it('Change alias to a too short one', () => {
    mainPage.login()
    cy.visit(Cypress.env('accountPage'))
    accountPage.clearAlias()
    .typeAlias('c1')
    .changeAlias()
    accountPage.checkError('Your alias must be at least 3 characters long')
  })

  it('Change alias to already existing one', () => {
    mainPage.login()
    cy.visit(Cypress.env('accountPage'))
    accountPage.clearAlias()
    .typeAlias('Middle')
    .changeAlias()
    mainPage.checkNotification('User already exists')
  })

  it('Change password', () => {
    accountPage.typeNewPassword("NewPassword123!")
    .typeConfirmPassword("NewPassword123")
    .typeOldPassword(Cypress.env('randomPassword'))
    .sendChangePassForm()
    mainPage.logout()
    .clickLogin()
    .typeEmail("chuzov.d@gmail.com")
    .typePassword("NewPassword123!")
    cy.visit(Cypress.env('accountPage'))
    accountPage.typeNewPassword("randomPassword")
    .typeConfirmPassword("randomPassword")
    .typeOldPassword(Cypress.env('NewPassword123'))
    .sendChangePassForm()
  })
  it('Change password with invalid new password', () => {
    accountPage.typeNewPassword("123456789")
    .typeConfirmPassword("123456789")
    .typeOldPassword(Cypress.env('randomPassword'))
    .sendChangePassForm(null)
    .checkError('Your password must be at least 8 characters long, must
contain 1 letter, 1 uppercase letter and 1 number')
  })



  it('Check Fractions', () => {
    mainPage.login()
    cy.visit(Cypress.env('accountPage'))
    accountPage.selectOdds("Fraction")
    cy.reload()
    accountPage.highLightedOdds("Fraction")
    cy.visit(Cypress.env('homePage'))
    mainPage.checkOddsType("Fraction")
  })

  it('Check Decimals', () => {
    mainPage.login()
    cy.visit(Cypress.env('accountPage'))
    accountPage.selectOdds("Decimal")
    cy.reload()
    accountPage.highLightedOdds("Decimal")
    cy.visit(Cypress.env('homePage'))
    mainPage.checkOddsType("Decimal")
  })


  it('Check American', () => {
    mainPage.login()
    cy.visit(Cypress.env('accountPage'))
    accountPage.selectOdds("American")
    cy.reload()
    accountPage.highLightedOdds("American")
    cy.visit(Cypress.env('homePage'))
    mainPage.checkOddsType("American")
    cy.visit(Cypress.env('accountPage'))
    accountPage.selectOdds("Decimal")
  })

})
```

A.9 PerkPageTests.js

```
import { MainPage } from '../pages/mainPage';
import { PerksPage } from '../pages/perks';
describe('Test actions on Perks page', () => {

  const mainPage = new MainPage();
  const perksPage = new PerksPage();
  beforeEach(() => {
    cy.visit(Cypress.env('homePage'))
    cy.wait(2000)
    mainPage.closePopUp().login()
    cy.visit(Cypress.env('perksPage'))
    cy.wait(2000)
  })
  it('Click challenge', () => {
    perksPage.clickChallange()
  })

  it('Click challenge logged out', () => {
    mainPage.clickNav().logout()
    perksPage.clickChallangesLoggedOut()
 })
  it('Click leaderboard', () => {
    perksPage.clickLeaderboard()
  })
  it('Click banners', () => {
    perksPage.clickBanner()
  })
  it('Test navbar clicks', () => {
    perksPage.clickLeaderboard()
    .clickNavbarPerks()
    .clickNavbarHome()
  })
```