IT 303 – Software Verification, Validation and Testing

TESTING DOCUMENTATION

Test automation for Liverpool FC website

Prepared by:
Faris Čolaković

Proposed to:
Samed Jukić, Assist. Prof. Dr.
Adnan Miljković, Teaching Assistant
Muhamed Mulic, Lecture Assistant

DATE OF SUBMISSION

Contents

1. Introduction

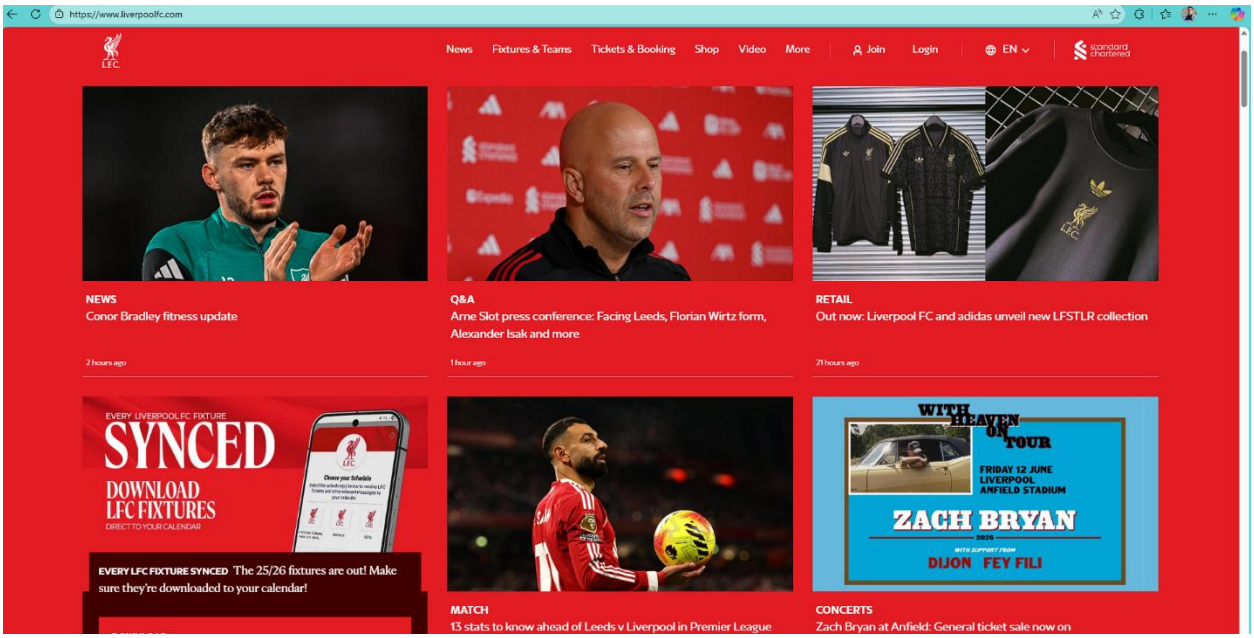
1.1. About the Project

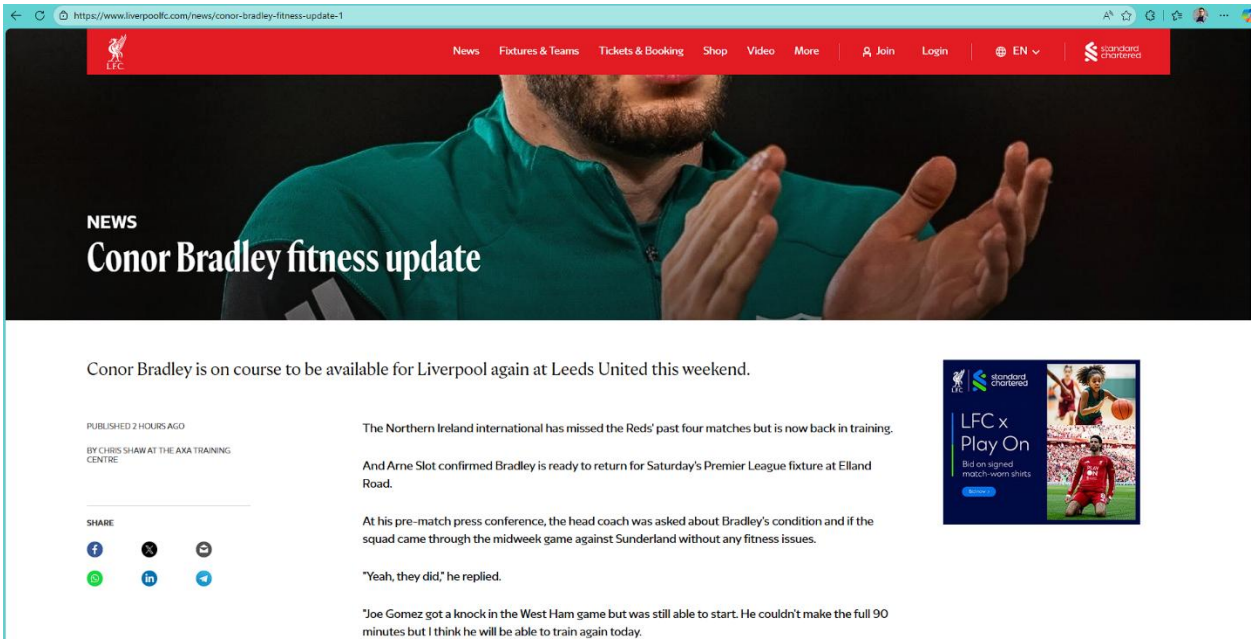
The web application under test is a football team platform that provides users with access to news articles, videos, player information, and other content related to everyday activities of a football club. It includes user authentication features such as login, logout, registration, as well as profile management. Users can access personal information such as name, date of birth or phone number used for verification. Homepage presents dynamic content such as the latest news, articles, videos, and multilingual content. Users can search for videos, filter news, navigate through sections, and update certain personal information in their profile.

Link to the homepage is: Liverpool FC — Homepage.

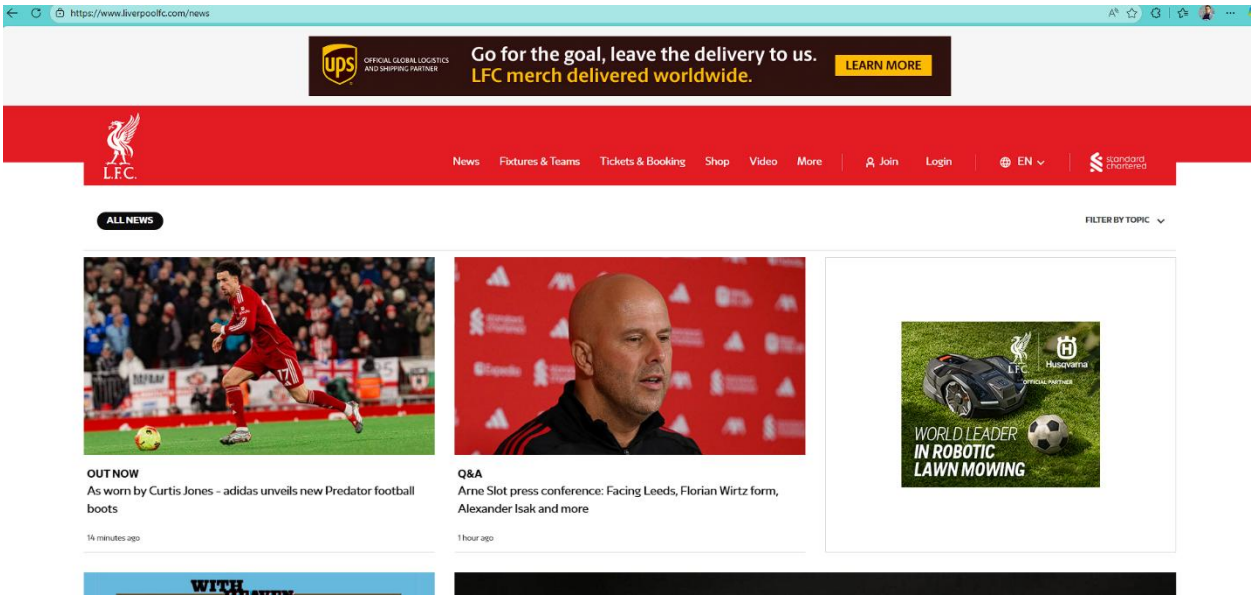
1.2. Project Functionalities and Screenshots

The project includes a full articles section on the home page where users can see all published articles along with their images, titles, and posting dates, and they can click into any article to read it in full.

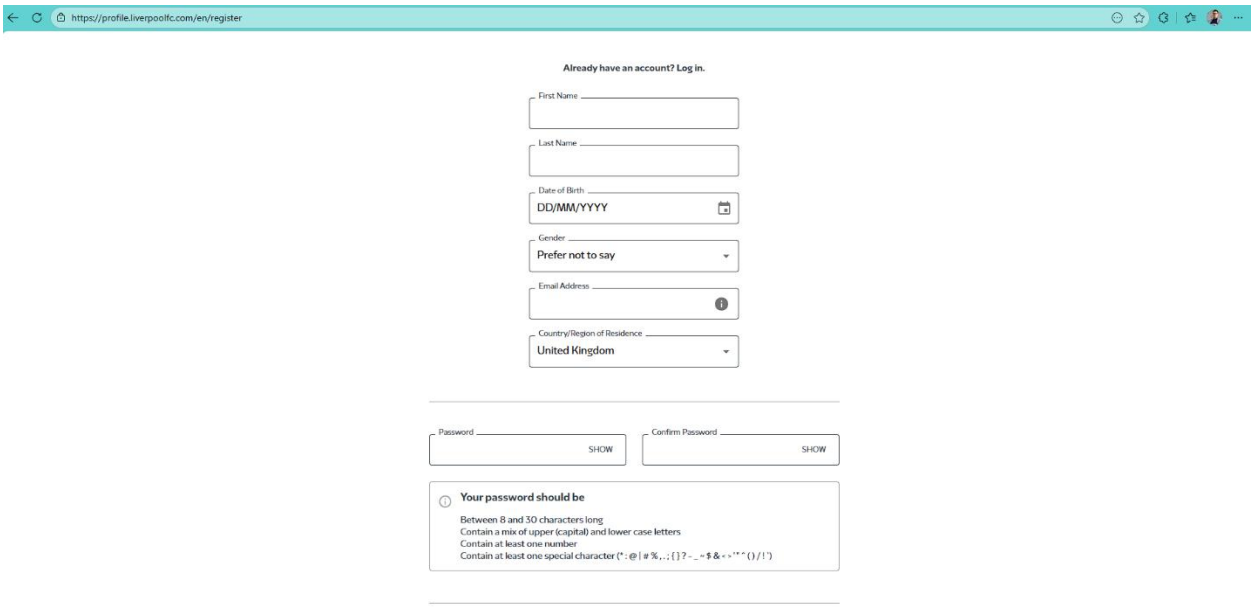


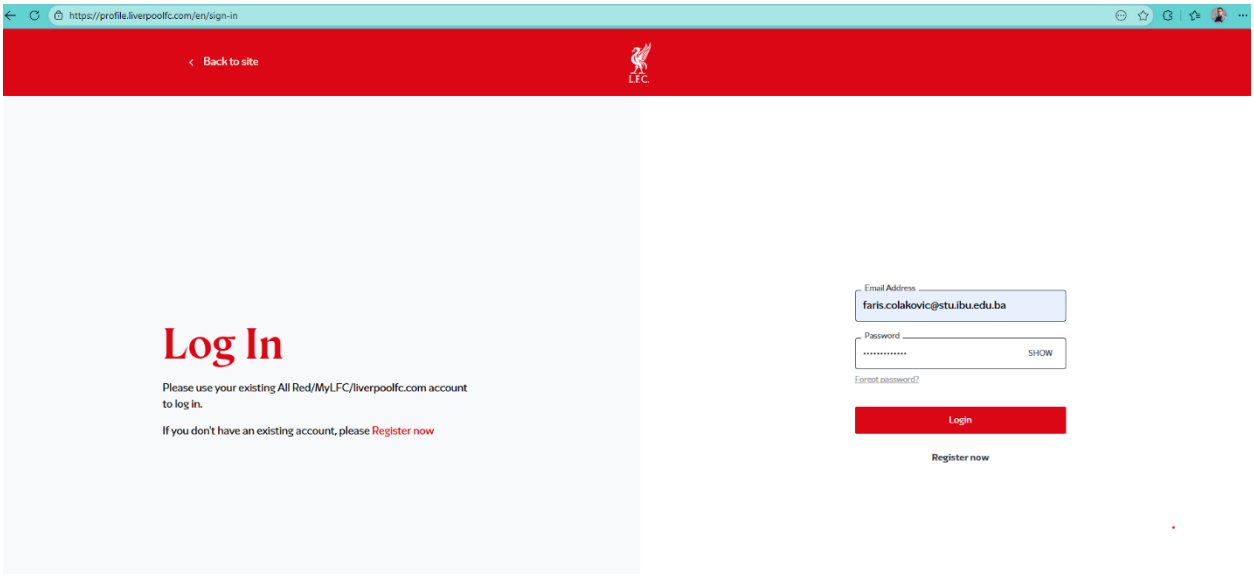


There's also a dedicated news tab that shows news titles and dates, and users can open each news item to read the complete details.

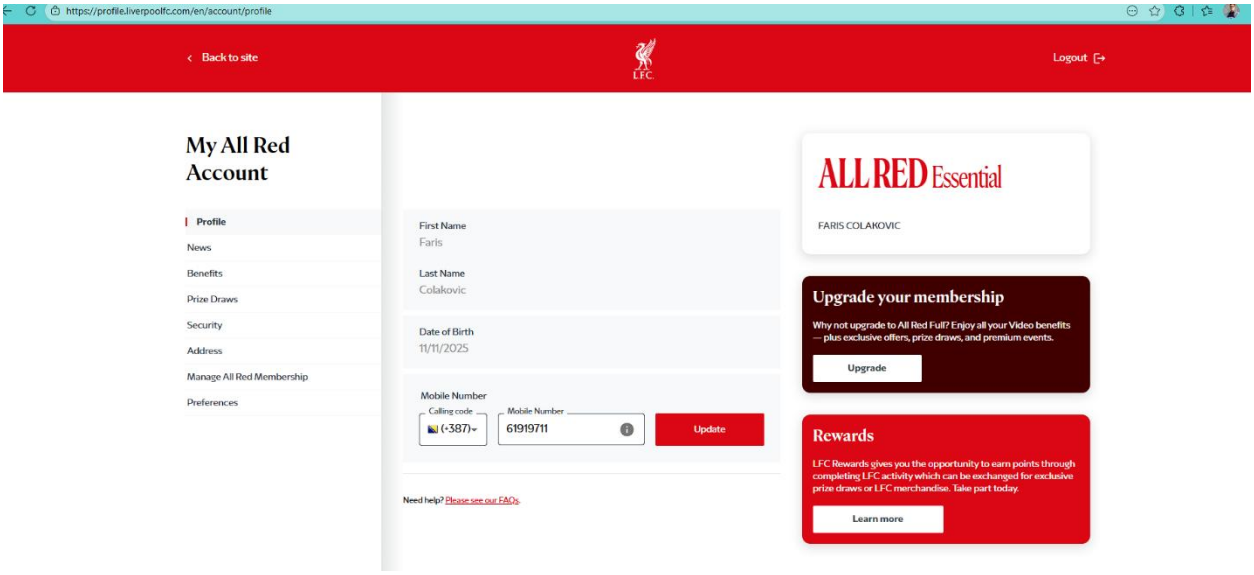


The platform supports user authentication with the ability to log in using an email and password, after registering a new account using first name, last name, date of birth, gender (optional) and country. When registering, password has strict rules which need to be followed such as specific length (between 8 and 30 characters), it has to contain upper and lower case letters, as well as at least one number and a special character.

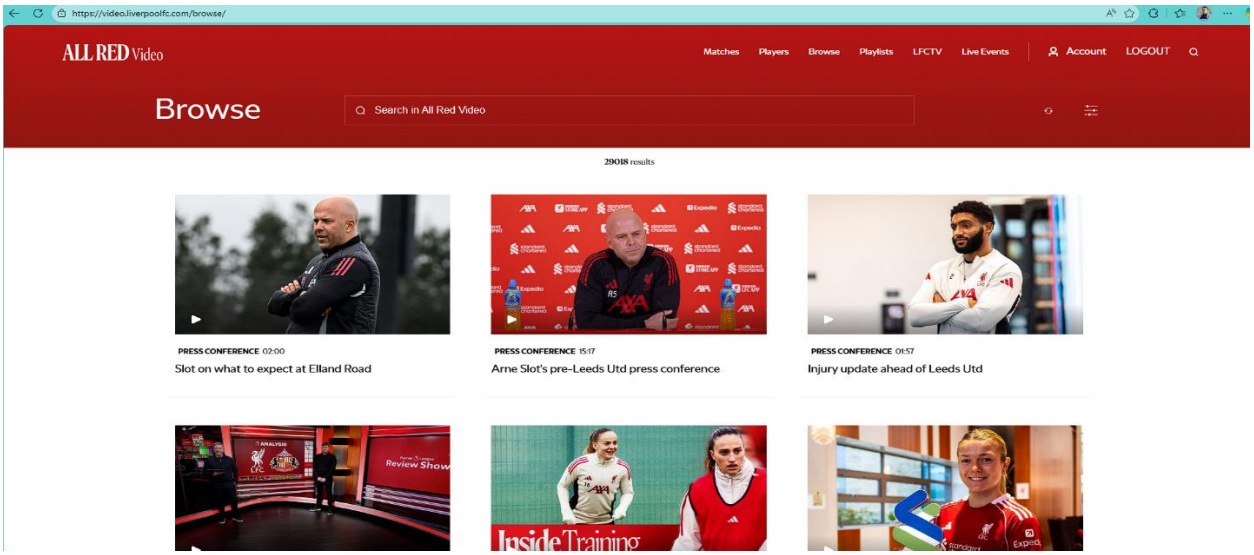


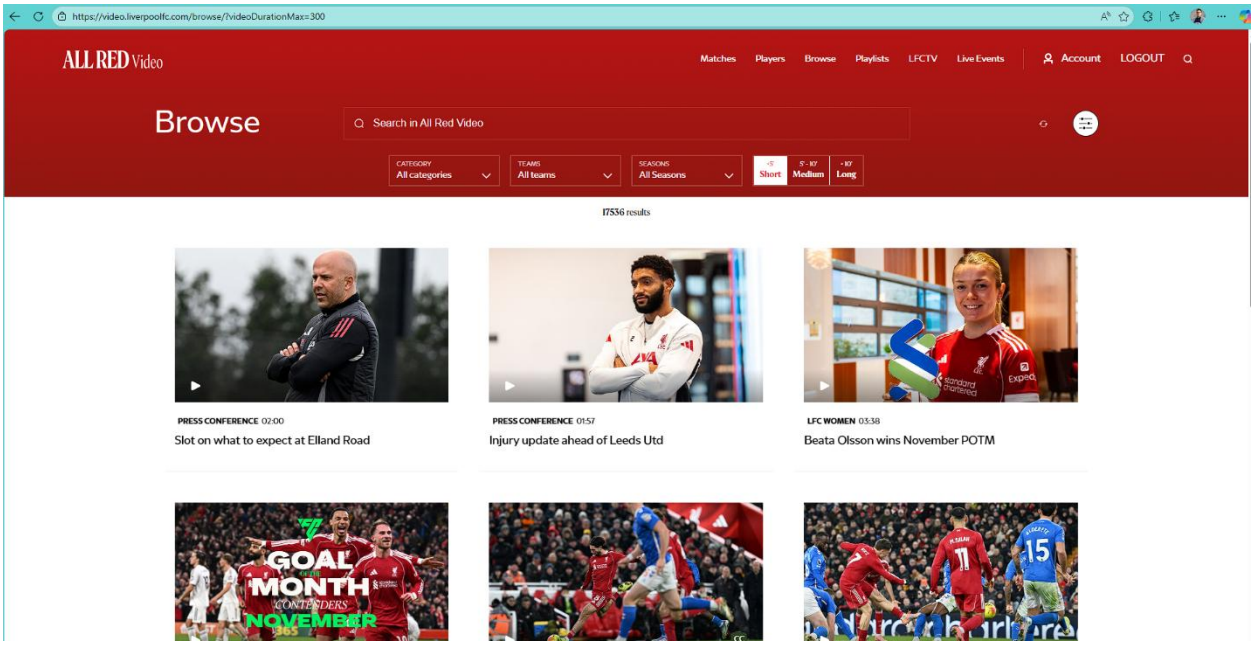


Once logged in, users can access their personal profile page, where they can view their account information and add a phone number for verification.

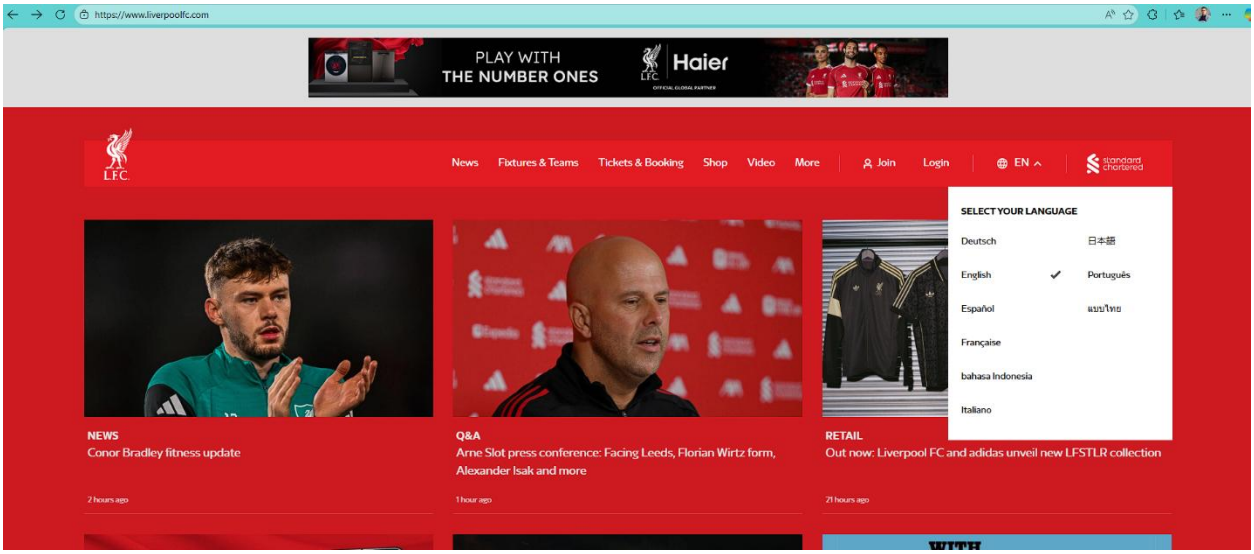


A major part of the project is the video section, where users can open and watch different videos. Free videos can be viewed by anyone with a free account, while paid videos require an active subscription and are not watchable otherwise. The video system also includes search functionality that allows users to look up videos from past seasons or filter them based on their content. Additional filtering is available based on video length, making it easier for users to find exactly what they want.





The entire platform supports multi-language functionality as well, allowing users to change the language of the full interface to a wide range of internationally recognized languages.



There are a lot of other features, however these are the main ones. This project is intended to be used as a platform for fans, giving them a place to follow the latest news, watch match-related videos, explore past seasons, and stay connected with everything happening around their team. That's why these features are the core foundation of the system.

2. Test Plan

2.1. Scope

For this project, I am planning to test all of the main features described earlier, including browsing articles, opening news items, user login and registration, accessing the personal profile, watching free videos, ensuring paid videos are restricted without a subscription, searching for videos from previous seasons, filtering by video length, and verifying that the multi-language functionality works correctly across the application. These represent the core functionality of the platform, so they will be the primary focus of the testing effort.

In addition to functional testing, I will also focus on basic security aspects, such as ensuring HTTPS enforcement throughout the application, proper handling of cookies, and secure session management to prevent unauthorized access or session hijacking.

There are certain parts of the application that I am not planning to test, mainly because they are not practical to automate or fully verify in the current environment. For example, I am not testing the phone-number verification process, since it requires reading a real verification code from a

physical mobile device. Similarly, I am not testing the feature that allows users to add the football calendar to their local computer calendar, because this interaction depends on the user's local operating system and cannot be reliably automated. Such features can be covered through manual testing instead.

2.2. Testing Environment and Tools

For the testing of this project, I will be using Selenium as the main tool for automating tests. Selenium allows me to simulate real user actions such as clicking buttons, filling out forms, navigating between pages, and validating UI behavior across different browsers (in this project I am using Chrome WebDriver). Also, in the latest updates Selenium handles browsers automatically through Selenium Manager, so there is no need to manually download and setup different browsers. If a certain browser is not installed on the local system, it will automatically be downloaded and set up by Selenium when the test engineer runs the tests. It's also very suitable for end-to-end testing of web applications and is a good fit for verifying the main features of this platform, including login, registration, searching, language changes, and video accessibility.

The programming language I use in this project is Java. Java integrates well with Selenium, has strong community support, and offers reliable performance and structure for larger automation projects. It also works seamlessly with the additional frameworks and tools I'm using. So far, I had never used Java with Selenium before, since most of my previous test automation work was done in C# (previously C# + Selenium or Playwright). However, I've been pleasantly surprised by how well everything works together in Java. The setup, structure, and workflow have been clean and intuitive, and overall I'm very satisfied with the experience. I especially enjoyed how easy the actual setup was with maven after specifiyng needed dependencies in the pom.xml file.

For assertions and test validation, I am using JUnit. JUnit provides a clean and simple way to write test cases, compare expected results with actual results, organize tests, and generate reports. It helps ensure all key functionalities of the application behave exactly as expected.
As my development environment, I am using IntelliJ IDEA, which is a powerful and user-friendly IDE for Java. IntelliJ provides features like intelligent code completion, built-in support for

Maven/Gradle, integrated debugging tools, and excellent handling of Selenium and JUnit project structures. This allows me to efficiently write, run, debug and maintain this automated test suite.

3. Test Execution

The screenshots included in this report show the test code used to automate the described test cases. The tests are written using a structured, readable automation approach (Page Object Model), where page elements and actions are organized into descriptive methods such as homePage.clickSign(), registration.fillRegistrationForm(), or profile.openSettings().

This ensures that the tests remain clean, maintainable, and easy to understand, rather than containing long, low-level or repetitive code directly inside the test files. In a way, I have built my own small test automation framework for the tested website.

Including full screenshots of all underlying page code would make this report excessively long (well over 100 pages). Therefore, only the actual test code is shown in this document as expected, while the complete detailed implementation can be reviewed directly within the project's IDE.

3.1. Successful User Registration

Users should be able to create a new account by providing all required information. This scenario verifies that a new user can register successfully and that the system behaves correctly after registration, by logging in the user automatically and by prompting them to verify their email.

<table>
<tr><td colspan="5">Test Name: Test valid user registration</td></tr>
<tr><td colspan="5">Description: Check if user account exists after valid registration and user is logged in</td></tr>
<tr><td colspan="5">Pre-condition(s):</td></tr>
<tr><td>Test Steps:
1. Go to the landing page
2. Click on join button
3. Click on register now button
4. Enter all valid data in all the form fields
5. Click on register
6. Click on back to site
7. Verify user is on landing page and is logged in</td><td>Test Data:
Valid data generated randomly. Valid email (correct email form) and random valid password (number, special char, lower & upper case, 8-30 chars)</td><td>Expected Result:

The user is logged in and his account exists after registering with valid data.</td><td>Actual Result:

The user is logged in and his account exists after registering with valid data.</td><td>Status:

PASS</td></tr>
<tr><td colspan="5">Notes:</td></tr>
</table>

```java
@Test  Faris Čolaković *
public void userIsLoggedInAfterValidRegistration() {
    home.clickSignIn();
    login.choseRegistrationPage();
    String email = RandomDataGenerator.generateEmail( domain: "gmail");
    String password = RandomDataGenerator.generatePassword( length: 10);
    String gender = RandomDataGenerator.generateGender();
    String birthDate = RandomDataGenerator.generateBirthdate(18, 30);
    registration.fillRegistrationForm( fName: "Faris",  lName: "Colakovic", birthDate,
            gender, email,  countryName: "United Kingdom", password, password,  subscribe: true);
    registration.finalizeRegistration( proceedBackToSite: true);
    home.validateUserIsCorrectlyLoggedInAfterRegistering();
}
```

Test Name: Test verify email flow				
Description: Check if user is navigated to verify email page after valid registration				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on join button 3. Click on register now button 4. Enter all valid data in all the form fields 5. Click on register 6. Verify user is taken to verify email page	**Test Data:** Valid data generated randomly. Valid email (correct email form) and random valid password (number, special char, lower & upper case, 8-30 chars)	**Expected Result:** The user is logged in and is taken to the verify email page automatically after valid registration.	**Actual Result:** The user is logged in and is taken to the verify email page automatically after valid registration.	**Status:** PASS
Notes:				

```java
@Test  Faris Čolaković
public void userIsAskedToVerifyMailAfterValidRegistration() {
    home.clickSignIn();
    login.choseRegistrationPage();
    String email = RandomDataGenerator.generateEmail( domain: "gmail");
    String password = RandomDataGenerator.generatePassword( length: 10);
    String gender = RandomDataGenerator.generateGender();
    String birthDate = RandomDataGenerator.generateBirthdate(18, 30);
    registration.fillRegistrationForm( fName: "Faris", lName: "Colakovic", birthDate,
            gender, email, countryName: "United Kingdom", password, password, subscribe: true);
    registration.finalizeRegistration( proceedBackToSite: false);
    registration.validateUserIsRedirectedToMailVerification();
}
```

3.2. Unsuccessful User Registration

Users must enter all required information in the correct format when creating a new account. This scenario verifies that registration fails when mandatory fields are missing or invalid, including empty inputs (first name, last name, email, password, confirm password, date of birth), incorrect email formats, mismatched password and confirmation fields, or the use of an already registered email address. The system should prevent account creation and display clear validation messages for each error condition.

Test Name: Test registration with existing email				
Description: Check if user is not allowed to register with email which already exists in the system				
Pre-condition(s): Have an existing email ready (previously register with email from Step 4)				
Test Steps: 1. Go to the landing page 2. Click on join button 3. Click on register now button 4. Enter all valid data in all the form fields except for the email which has already been used 5. Click on register 6. Validate correct error about the email is displayed	**Test Data:** Valid data generated randomly. Invalid email (already used for registration) and random valid password (number, special char, lower & upper case, 8-30 chars)	**Expected Result:** Registration fails and user can see an error message which says that an account with existing email already exists.	**Actual Result:** Registration fails and user can see an error message which says that an account with existing email already exists.	**Status:** PASS
Notes:				

```java
@Test  Faris Čolaković
public void registrationWithExistingEmailShowsError() {
    home.clickSignIn();
    login.choseRegistrationPage();
    String email = "fariscolakovic00@gmail.com";
    String password = RandomDataGenerator.generatePassword( length: 10);
    String gender = RandomDataGenerator.generateGender();
    String birthDate = RandomDataGenerator.generateBirthdate(18, 30);
    registration.fillRegistrationForm( fName: "Faris",  lName: "Colakovic", birthDate,
            gender, email,  countryName: "United Kingdom", password, password,  subscribe: true);
    registration.finalizeRegistration( proceedBackToSite: false);
    registration.validateDuplicateEmailError();
}
```

Test Name: Test registration with incorrect email				
Description: Check if user is not allowed to register with email which has incorrect mail format				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on join button 3. Click on register now button 4. Enter all valid data in all the form fields except for the email which has already been used 5. Click on register 6. Validate correct error about the email is displayed	**Test Data:** Valid data generated randomly. Invalid email (wrong email format in this case not containing '@') and random valid password (number, special char, lower & upper case, 8-30 chars)	**Expected Result:** Registration fails and user can see an error message which says that a valid email is required.	**Actual Result:** Registration fails and user can see an error message which says that a valid email is required.	**Status:** PASS
Notes:				

```java
@Test  Faris Čolaković
public void registrationWithIncorrectEmailShowsError() {
    home.clickSignIn();
    login.choseRegistrationPage();
    String email = "fariscolakovic00gmail.com";
    String password = RandomDataGenerator.generatePassword( length: 10);
    String gender = RandomDataGenerator.generateGender();
    String birthDate = RandomDataGenerator.generateBirthdate(18, 30);
    registration.fillRegistrationForm( fName: "Faris",  lName: "Colakovic", birthDate,
            gender, email,  countryName: "United Kingdom", password, password,  subscribe: true);
    registration.finalizeRegistration( proceedBackToSite: false);
    registration.validateWrongEmailFormatError();
}
```

Test Name: Test registration with without first name				
Description: Check if user is not allowed to register without entering first name				
Pre-condition(s): Have an existing email ready (previously register with email from Step 4)				
Test Steps: 1. Go to the landing page 2. Click on join button 3. Click on register now button 4. Enter all valid data in all the form fields except for the first name 5. Click on register 6. Validate correct error about the first name is displayed	**Test Data:** Valid data generated randomly except for first name. Valid email and random valid password (number, special char, lower & upper case, 8-30 chars).	**Expected Result:** Registration fails and user can see an error message which says that entering first name is required.	**Actual Result:** Registration fails and user can see an error message which says that entering first name is required.	**Status:** PASS
Notes:				

```java
@Test  Faris Čolaković
public void registrationWithoutFirstNameShowsError() {
    home.clickSignIn();
    login.choseRegistrationPage();
    String email = RandomDataGenerator.generateEmail( domain: "gmail");
    String password = RandomDataGenerator.generatePassword( length: 10);
    String gender = RandomDataGenerator.generateGender();
    String birthDate = RandomDataGenerator.generateBirthdate(18, 30);
    registration.fillRegistrationForm( fName: "",  lName: "Colakovic", birthDate,
            gender, email,  countryName: "United Kingdom", password, password,  subscribe: true);
    registration.finalizeRegistration( proceedBackToSite: false);
    registration.validateFirstNameEmptyError();
}
```

Test Name: Test registration with without date of birth				
Description: Check if user is not allowed to register without entering date of birth				
Pre-condition(s): Have an existing email ready (previously register with email from Step 4)				
Test Steps: 1. Go to the landing page 2. Click on join button 3. Click on register now button 4. Enter all valid data in all the form fields except for the date of birth 5. Click on register 6. Validate correct error about date of birth is displayed	**Test Data:** Valid data generated randomly except date of birth. Valid email and random valid password (number, special char, lower & upper case, 8-30 chars).	**Expected Result:** Registration fails and user can see an error message which says that entering date of birth is required.	**Actual Result:** Registration fails and user can see an error message which says that entering date of birth is required.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void registrationWithoutDateShowsError() {
    home.clickSignIn();
    login.choseRegistrationPage();
    String email = RandomDataGenerator.generateEmail( domain: "gmail");
    String password = RandomDataGenerator.generatePassword( length: 10);
    String gender = RandomDataGenerator.generateGender();
    String birthDate = "";
    registration.fillRegistrationForm( fName: "Faris",  lName: "Colakovic", birthDate,
            gender, email,  countryName: "United Kingdom", password, password,  subscribe: true);
    registration.finalizeRegistration( proceedBackToSite: false);
    registration.validateDateOfBirthEmptyError();
}
```

Test Name: Test registration with without last name				
Description: Check if user is not allowed to register without entering last name				
Pre-condition(s): Have an existing email ready (previously register with email from Step 4)				
Test Steps: 1. Go to the landing page 2. Click on join button 3. Click on register now button 4. Enter all valid data in all the form fields except for the last name 5. Click on register	**Test Data:** Valid data generated randomly except last name. Valid email and random valid password (number, special char, lower &	**Expected Result:** Registration fails and user can see an error message which says that entering last name is required.	**Actual Result:** Registration fails and user can see an error message which says that entering last name is required.	**Status:** PASS

6. Validate correct error about last name is displayed	upper case, 8-30 chars).			
Notes:				

```
@Test  Faris Čolaković
public void registrationWithoutLastNameShowsError() {
    home.clickSignIn();
    login.choseRegistrationPage();
    String email = RandomDataGenerator.generateEmail( domain: "gmail");
    String password = RandomDataGenerator.generatePassword( length: 10);
    String gender = RandomDataGenerator.generateGender();
    String birthDate = RandomDataGenerator.generateBirthdate(18, 30);
    registration.fillRegistrationForm( fName: "Faris",  lName: "", birthDate,
            gender, email,  countryName: "United Kingdom", password, password,  subscribe: true);
    registration.finalizeRegistration( proceedBackToSite: false);
    registration.validateLastNameEmptyError();
}
```

Test Name: Test registration with without email				
Description: Check if user is not allowed to register without entering email				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on join button 3. Click on register now button 4. Enter all valid data in all the form fields except for the email 5. Click on register 6. Validate correct error about email is displayed	**Test Data:** Valid data generated randomly except for email. Valid random password (number, special char, lower & upper case, 8-30 chars).	**Expected Result:** Registration fails and user can see an error message which says that entering email is required.	**Actual Result:** Registration fails and user can see an error message which says that entering email is required.	**Status:** PASS
Notes:				

```java
@Test  Faris Čolaković
public void registrationWithoutEmailShowsError() {
    home.clickSignIn();
    login.choseRegistrationPage();
    String email = "";
    String password = RandomDataGenerator.generatePassword( length: 10);
    String gender = RandomDataGenerator.generateGender();
    String birthDate = RandomDataGenerator.generateBirthdate(18, 30);
    registration.fillRegistrationForm( fName: "Faris",  lName: "Colakovic", birthDate,
            gender, email,  countryName: "United Kingdom", password, password,  subscribe: true);
    registration.finalizeRegistration( proceedBackToSite: false);
    registration.validateEmailEmptyError();
}
```

Test Name: Test registration with without password				
Description: Check if user is not allowed to register without entering password				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on join button 3. Click on register now button 4. Enter all valid data in all the form fields except for the password 5. Click on register 6. Validate correct error about password is displayed	**Test Data:** Valid data generated randomly except for password.	**Expected Result:** Registration fails and user can see an error message which says that entering password is required.	**Actual Result:** Registration fails and user can see an error message which says that entering password is required.	**Status:** PASS
Notes:				

```java
@Test  Faris Čolaković
public void registrationWithoutPasswordShowsError() {
    home.clickSignIn();
    login.choseRegistrationPage();
    String email = RandomDataGenerator.generateEmail( domain: "gmail");
    String password = "";
    String gender = RandomDataGenerator.generateGender();
    String birthDate = RandomDataGenerator.generateBirthdate(18, 30);
    registration.fillRegistrationForm( fName: "Faris",  lName: "Colakovic", birthDate,
            gender, email,  countryName: "United Kingdom", password, password,  subscribe: true);
    registration.finalizeRegistration( proceedBackToSite: false);
    registration.validatePasswordEmptyError();
}
```

Test Name: Test registration with without confirming password				
Description: Check if user is not allowed to register without confirming password				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on join button 3. Click on register now button 4. Enter all valid data in all the form fields except for the confirm password 5. Click on register 6. Validate correct error about confirming password is displayed	**Test Data:** Valid data generated randomly.	**Expected Result:** Registration fails and user can see an error message which says that entering confirmed password is required.	**Actual Result:** Registration fails and user can see an error message which says that entering confirmed password is required.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void registrationWithoutConfirmingPasswordShowsError() {
    home.clickSignIn();
    login.choseRegistrationPage();
    String email = RandomDataGenerator.generateEmail( domain: "gmail");
    String password = RandomDataGenerator.generatePassword( length: 10);
    String gender = RandomDataGenerator.generateGender();
    String birthDate = RandomDataGenerator.generateBirthdate(18, 30);
    registration.fillRegistrationForm( fName: "Faris",  lName: "Colakovic", birthDate,
            gender, email,  countryName: "United Kingdom", password,  confirmPass: "",  subscribe: true);
    registration.finalizeRegistration( proceedBackToSite: false);
    registration.validateConfirmPasswordEmptyError();
}
```

Test Name: Test registration with with non-matching confirmed password				
Description: Check if user is not allowed to register without confirming password correctly				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on join button 3. Click on register now button 4. Enter all valid data in all the form fields except for confirm password 5. In the confirm password field, enter a non matching password 6. Click on register 7. Validate correct error about confirming password is displayed	**Test Data:** Valid data generated randomly.	**Expected Result:** Registration fails and user can see an error message which says that inputs from password and confirm password fields need to be matching.	**Actual Result:** Registration fails and user can see an error message which says that inputs from password and confirm password fields need to be matching.	**Status:** PASS
Notes:				

3.3. Successful User Login

Users should be able to access their accounts by entering valid credentials. This scenario verifies that the login process works correctly under normal conditions.

Test Name: Test valid login				
Description: Check if user can log in using existing email and password				
Pre-condition(s): Have an email and password of a previously created account				
Test Steps: 1. Go to the landing page 2. Click on login button in the top right section of the landing page 3.Enter existing email and password 4. Click on Login button 5. Verify user is taken to the landing page but now he is logged in	**Test Data:** Email and password from a previously created/registered account.	**Expected Result:** Login is successful and user is taken to the landing page where he is now logged in (there are account and logout buttons displayed).	**Actual Result:** Login is successful and user is taken to the landing page where he is now logged in (there are account and logout buttons displayed).	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void validLogin() {
    home.clickSignIn();
    login.login( user: "fariscolakovic00@gmail.com",  pass: "hBSBR!LKDsna1");
    home.validateUserIsCorrectlyLoggedInAfterRegistering();
}
```

Test Name: Test valid login with case-insensitive email				
Description: Check if user can log in using existing case-insensitive email and password				
Pre-condition(s): Have an email and password of a previously created account				
Test Steps: 1. Go to the landing page 2. Click on login button in the top right section of the landing page 3.Enter existing case-insensitive email and password 4. Click on Login button 5. Verify user is taken to the landing page but now he is logged in	**Test Data:** Case-insensitive mail and password from a previously created/registered account.	**Expected Result:** Login is successful and user is taken to the landing page where he is now logged in (there are account and logout buttons displayed).	**Actual Result:** Login is successful and user is taken to the landing page where he is now logged in (there are account and logout buttons displayed).	**Status:** PASS
Notes:				

```
@Test  new *
public void validLoginWithCaseInsensitiveMail() {
    home.clickSignIn();
    login.login( user: "farisCOLAKovic00@gmail.com",  pass: "hBSBR!LKDsna1");
    home.validateUserIsCorrectlyLoggedInAfterRegistering();
}
```

3.4. Unsuccessful User Login

Users must provide valid account information to access the platform. This scenario verifies that the login process prevents access when incorrect or invalid data is used, including wrong email, wrong password, invalid email format, empty fields, or attempts to log in with a non-existent account. The system should block the login attempt without creating a session.

Test Name: Test login with invalid password				
Description: Check if user is not allowed to login with invalid password				
Pre-condition(s): Have an email and password of a previously created account				
Test Steps: 1. Go to the landing page 2. Click on login button in the top right section of the landing page 3.Enter existing email but different password from the pre-condition 4. Click on Login button 5. Verify user is not taken to the landing page and not logged in.	**Test Data:** Same email and a different password from a previously created/registered account.	**Expected Result:** Login is usuccessful and user is not taken to the landing page and not loged in (there are no account and logout buttons displayed). Login fails.	**Actual Result:** Login is usuccessful and user is not taken to the landing page and not loged in (there are no account and logout buttons displayed). Login fails.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković *
public void invalidPasswordFailsLogin() {
    home.clickSignIn();
    login.login( user: "fariscolakovic00@gmail.com",  pass: "wrongpass");
    home.validateUserIsNotLoggedIn();
}
```

Test Name: Test login with empty password				
Description: Check if user is not allowed to login with empty password				
Pre-condition(s): Have an email and password of a previously created account				
Test Steps: 1. Go to the landing page 2. Click on login button in the top right section of the landing page 3.Enter existing email but empty password 4. Click on Login button 5. Verify user is not taken to the landing page and not logged in.	**Test Data:** Same email from the pre-condition and an empty password	**Expected Result:** Login is usuccessful and user is not taken to the landing page and not loged in (there are no account and logout buttons displayed). Login fails.	**Actual Result:** Login is usuccessful and user is not taken to the landing page and not loged in (there are no account and logout buttons displayed). Login fails.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković *
public void emptySpacesPasswordFailsLogin() {
    home.clickSignIn();
    login.login( user: "fariscolakovic00@gmail.com",  pass: "   ");
    home.validateUserIsNotLoggedIn();
}
```

Test Name: Test login with invalid email				
Description: Check if user is not allowed to login with invalid email				
Pre-condition(s): Have an email and password of a previously created account				
Test Steps: 1. Go to the landing page 2. Click on login button in the top right section of the landing page 3.Enter different email from the pre-condition but same password from the pre-condition 4. Click on Login button 5. Verify user is not taken to the landing page and not logged in.	**Test Data:** Different email and the same password from a previously created/registered account.	**Expected Result:** Login is usuccessful and user is not taken to the landing page and not loged in (there are no account and logout buttons displayed). Login fails.	**Actual Result:** Login is usuccessful and user is not taken to the landing page and not loged in (there are no account and logout buttons displayed). Login fails.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković *
public void invalidEmailFailsLogin() {
    home.clickSignIn();
    login.login( user: "fariscolakovic0@gmail.com",  pass: "hBSBR!LKDsna1");
    home.validateUserIsNotLoggedIn();
}
```

Test Name: Test login with empty email				
Description: Check if user is not allowed to login with empty email				
Pre-condition(s): Have an email and password of a previously created account				
Test Steps: 1. Go to the landing page 2. Click on login button in the top right section of the landing page 3.Enter email email but existing password from the pre-condition 4. Click on Login button 5. Verify user is not taken to the landing page and not logged in.	**Test Data:** Empty email and the same password from the pre-condition	**Expected Result:** Login is usuccessful and user is not taken to the landing page and not loged in (there are no account and logout buttons displayed). Login fails.	**Actual Result:** Login is usuccessful and user is not taken to the landing page and not loged in (there are no account and logout buttons displayed). Login fails.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković *
public void emptySpacesEmailFailsLogin() {
    home.clickSignIn();
    login.login( user: "    ",  pass: "hBSBR!LKDsna1");
    home.validateUserIsNotLoggedIn();
}
```

3.5. Updating user profile data

Users should be able to change their phone number through the profile page by entering a valid new number. This field is the only editable field in the profile section, since date of birth, first and last name are not editable. This scenario verifies that the system correctly updates the phone number when valid data is submitted and prevents the update when invalid or incomplete information is provided, displaying the appropriate validation messages.

This test scenario also has certain test fixtures and pre-conditions such as beforeEach and afterEach:

```
@BeforeEach  Faris Čolaković
public  void loginBeforeAll() {
    home.clickSignIn();
    login.correctLoginForRefactoring();
    home.openAccountSettings();
}

@AfterEach  Faris Čolaković
public void logoutAfterAll() {
    home.logOutFromProfilePage();
}
```

Since user profile tests require user to be logged in, login will be achieved before and logout will be done after each test. Also, before all the actual tests of updating the phone number start, account settings will be opened through the home.openAccountSettings() method:

```
public void openAccountSettings() {  6 usages  Faris Čolaković
    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
    wait.until(ExpectedConditions.elementToBeClickable(accountButton)).click();
}
```

Test Name: Test update of phone number with valid data				
Description: Check if user can successfully edit phone number and save the new number				
Pre-condition(s): User should be logged in before the test with a valid user account				
Test Steps: 1. Go to the landing page 2. Click on the account button in top right corner 3. Edit the phone number country and the phone number input field with valid data for the relevant country (e.g. +387 followed by 9 numbers for BiH) 4. Click on update	**Test Data:** Random valid phone number format for Bosnia and Herzegovina	**Expected Result:** Phone number edit is successful, there is a notification that notifies the user about the successful update.	**Actual Result:** Phone number edit is successful, there is a notification that notifies the user about the successful update.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void updateWithValidPhoneNumber() {
    profile.selectCountryByDataValue("BA|+387");
    profile.enterPhoneNumber("61919711");
    profile.clickUpdateButton();
    Assertions.assertTrue(profile.validateSuccessfulUpdateNotificationDisplayed());
}
```

Test Name: Test update of phone number with empty input				
Description: Check if user is not allowed to edit phone number with new empty input				
Pre-condition(s): User should be logged in before the test with a valid user account				
Test Steps: 1. Go to the landing page 2. Click on the account button in top right corner 3. Enter empty string in the phone number input field 4. Click on update	**Test Data:** Just empty string for the phone number input	**Expected Result:** Phone number edit is unsuccessful, there is an error message that the phone number field is required.	**Actual Result:** Phone number edit is unsuccessful, there is an error message that the phone number field is required.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void updateWithEmptyPhoneNumber() {
    profile.selectCountryByDataValue("BA|+387");
    profile.enterPhoneNumber("");
    profile.clickUpdateButton();
    profile.validateRequiredFieldError();
}
```

Test Name: Test update of phone number with short number				
Description: Check if user is not allowed to edit phone number with too short number format				
Pre-condition(s): User should be logged in before the test with a valid user account				
Test Steps: 1. Go to the landing page 2. Click on the account button in top right corner 3. Enter too short phone number (e.g. just 3 numbers for the Bosnia and Herzegovina format) 4. Click on update	**Test Data:** Random phone number which only contains e.g. 3 characters	**Expected Result:** Phone number edit is unsuccessful, there is an error message that the phone number is invalid.	**Actual Result:** Phone number edit is unsuccessful, there is an error message that the phone number is invalid.	**Status:** PASS
Notes:				

```
36      @Test  Faris Čolaković
37      public void updateWithTooShortPhoneNumber() {
38          profile.selectCountryByDataValue("BA|+387");
39          profile.enterPhoneNumber("629");
40          profile.clickUpdateButton();
41          profile.validateInvalidPhoneNumberFieldError();
42      }
```

Test Name: Test update of phone number with too long number				
Description: Check if user is not allowed to edit phone number with too long number format				
Pre-condition(s): User should be logged in before the test with a valid user account				
Test Steps: 1. Go to the landing page 2. Click on the account button in top right corner 3. Enter too long phone number (e.g. more than 8 characters for the Bosnia and Herzegovina format) 4. Click on update	**Test Data:** Random phone number which contains too many characters e.g. 10 characters for BiH format	**Expected Result:** Phone number edit is unsuccessful, there is an error message that the phone number is invalid.	**Actual Result:** Phone number edit is unsuccessful, there is an error message that the phone number is invalid.	**Status:** PASS
Notes:				

```
44      @Test  Faris Čolaković
45      public void updateWithTooLongPhoneNumber() {
46          profile.selectCountryByDataValue("BA|+387");
47          profile.enterPhoneNumber("6241059482");
48          profile.clickUpdateButton();
49          profile.validateInvalidPhoneNumberFieldError();
50      }
```

Test Name: Test update of phone number with special characters				
Description: Check if user is not allowed to edit phone number with value that contains special characters				
Pre-condition(s): User should be logged in before the test with a valid user account				
Test Steps: 1. Go to the landing page 2. Click on the account button in top right corner 3. Enter phone number with special characters (such as '!' or '*') 4. Click on update	**Test Data:** Random phone number which contains any of the special characters such as !, *, -, dot etc.	**Expected Result:** Phone number edit is unsuccessful, there is an error message that the phone number is invalid.	**Actual Result:** Phone number edit is unsuccessful, there is an error message that the phone number is invalid.	**Status:** PASS
Notes:				

```
52      @Test  Faris Čolaković
53      public void updateWithSpecialCharsPhoneNumber() {
54          profile.selectCountryByDataValue("BA|+387");
55          profile.enterPhoneNumber("62410!*.-");
56          profile.clickUpdateButton();
57          profile.validateInvalidPhoneNumberFieldError();
58      }
```

3.6. Validating browse video page functionalities

Users should be able to browse the Videos page and see the available video content without issues. This scenario verifies that the initial list of videos loads correctly, displays the expected number of items and that additional videos can be retrieved through the “Load More” functionality.

Test Name: Test relevant video categories are shown				
Description: Check if user is shown all the possible relevant video categories				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Verify popup appears and it contains all the relevant video categories	**Test Data:**	**Expected Result:** User can browse and see All Red Video, Matches, Players, Browse, Playlists, Live categories.	**Actual Result:** User can browse and see All Red Video, Matches, Players, Browse, Playlists, Live categories.	**Status:** PASS
Notes:				

```
11      @Test  Faris Čolaković
12      public void openVideosPageAndValidateVideoCategories() {
13          home.clickOnSection( sectionName: "Video");
14          videos.validateAllCategories();
15      }
```

```
186     public void validateAllCategories() {  1 usage  Faris Čolaković
187         List<WebElement> categoryLinks = wait.until(ExpectedConditions.visibilityOfAllElementsLocatedBy(categoriesTags));
188
189         List<String> visibleTexts = new ArrayList<>();
190         for (WebElement link : categoryLinks) {
191             if (link.isDisplayed()) {
192                 visibleTexts.add(link.getText());
193             }
194         }
195
196         for (String category : categories) {
197             assertTrue(visibleTexts.contains(category),
198                     message: "Expected category not visible: " + category);
199         }
200     }
```

Test Name: Test browse category is correctly opened				
Description: Check if user clicking on the browse video category opens up the correct page				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Verify user is taken to the Browse Videos page	**Test Data:**	**Expected Result:** User is taken to the Browse Video page correctly.	**Actual Result:** User is taken to the Browse Video page correctly.	**Status:** PASS
Notes:				

```
17      @Test  Faris Čolaković
18      public void openVideosPageAndValidateBrowseCategory() {
19          home.clickOnSection( sectionName: "Video");
20          videos.openVideosSubSection("Browse");
21          videos.validateCurrentUrl();
22          videos.validateCorrectCategoryIsOpened( nameOfCategory: "Browse");
23      }
```

Test Name: Test correct number of videos is shown				
Description: Check if user can see number of videos in the browse videos category and that this number is correct				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Verify user is shown the actual number of all existing videos and that this number is correct	**Test Data:**	**Expected Result:** User can see the correct number of existing videos up until that point.	**Actual Result:** User can see the correct number of existing videos up until that point.	**Status:** PASS
Notes:				

```
25      @Test  Faris Čolaković
26      public void openBrowseVideosPageAndValidateNumberOfResults() {
27          home.clickOnSection( sectionName: "Video");
28          videos.openVideosSubSection("Browse");
29          videos.validateCurrentUrl();
30          videos.validateNumberOfAllVideos( numOfVideos: 28000);
31      }
```

Test Name: Test 12 videos are displayed per 1 showing				
Description: Check if user can see 12 videos by default without loading more of them				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Verify user can see 12 different videos on this page	**Test Data:**	**Expected Result:** User can see 12 videos by default.	**Actual Result:** User can see 12 videos by default.	**Status:** PASS
Notes:				

```
33      @Test  Faris Čolaković
34      public void openBrowseVideosPageAndValidateThereAre12Videos() {
35          home.clickOnSection( sectionName: "Video");
36          videos.openVideosSubSection("Browse");
37          videos.validateCurrentUrl();
38          videos.validateNumberOfVideosInTheList( sizeOfVideoList: 12);
39      }
```

Test Name: Test load more functionality				
Description: Check if user can see additional 12 videos each time he clicks on load more button				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Click on the load more button 5. Verify user can now see 24 videos (12 videos after each load more) after clicking on load more button once	**Test Data:**	**Expected Result:** User can see 24 videos after clicking on load more button once.	**Actual Result:** User can see 24 videos after clicking on load more button once.	**Status:** PASS
Notes:				

```
41      @Test  Faris Čolaković
42      public void openBrowseVideosPageAndValidateLoadMoreVideos() {
43          home.clickOnSection( sectionName: "Video");
44          videos.openVideosSubSection("Browse");
45          videos.validateCurrentUrl();
46          videos.closeMarketingPopUp();
47          videos.loadMoreVideos();
48          videos.validateNumberOfVideosInTheList( sizeOfVideoList: 24);
49      }
50  }
```

3.7. Validate search videos functionality

Users should be able to quickly locate specific video content by entering a title or partial title into the search bar. This scenario verifies that the search functionality returns accurate and relevant results based on the user's input, correctly filters out unrelated content, and handles cases where no matching videos are found.

Test Name: Test valid videos search that will return results				
Description: Check if user can search for any video by entering				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Enter name of any player in the search input field 5. Click Enter 6. Verify search results	**Test Data:** Use name of any player with first capitalized letter, as user would for any other name (e.g. “Wirtz”).	**Expected Result:** User can only see videos which contain that relevant player.	**Actual Result:** User can only see videos which contain that relevant player.	**Status:** PASS
Notes:				

```
8       @Test  Faris Čolaković
9       public void searchVideosByValidInputWithResults() {
10          home.clickOnSection( sectionName: "Video");
11          videos.openVideosSubSection("Browse");
12          videos.closeMarketingPopUp();
13          videos.enterSearchVideosValue( searchValue: "Wirtz");
14          videos.validateAllVideoTitlesContainInputString( expectedTitle: "Wirtz");
15      }
```

Test Name: Test video searching that will return no results				
Description: Check if user will not get any results if he searches for something totally irrelevant that does not exist in the videos section				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Enter anything that is irrelevant to the project in the videos search field e.g. testers name (“fariscolakovic”) 5. Click Enter 6. Verify search results	**Test Data:** Use any input for the search field that is totally irrelevant to this project and does not exist in the videos database such as testers name	**Expected Result:** There are no results since user searched for something that does not exist in the videos section and in the projects database and scope.	**Actual Result:** There are no results since user searched for something that does not exist in the videos section and in the projects database and scope.	**Status:** PASS
Notes:				

```
17      @Test  Faris Čolaković
18      public void searchVideosByValidInputWithNoResults() throws InterruptedException {
19          home.clickOnSection( sectionName: "Video");
20          videos.openVideosSubSection("Browse");
21          videos.closeMarketingPopUp();
22          videos.enterSearchVideosValue( searchValue: "fariscolakovic");
23          videos.validateExactNumberOfAllVideos( numOfVideos: 0);
24      }
```

Test Name: Test valid videos search that will return results with spaces				
Description: Check if user can search for any video by entering string with leading and trailing spaces				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Enter name of any player in the search input field but with leading and trailing spaces. 5. Click Enter 6. Verify search results	**Test Data:** Use name of any player with first capitalized letter, as user would for any other name (e.g. “ Isak “).	**Expected Result:** User can only see videos which contain that relevant player and spaces are trimmed.	**Actual Result:** User can only see videos which contain that relevant player and spaces are trimmed.	**Status:** PASS
Notes:				

```
26      @Test  Faris Čolaković
27      public void searchVideosByInputWithLeadingAndTrailingSpaces() {
28          home.clickOnSection( sectionName: "Video");
29          videos.openVideosSubSection("Browse");
30          videos.closeMarketingPopUp();
31          videos.enterSearchVideosValue( searchValue: " Isak ");
32          videos.validateAllVideoTitlesContainInputString( expectedTitle: "Isak");
33      }
```

Test Name: Test videos search by input with case sensitivity				
Description: Check if user can search for any video by entering string that will have both lower and upper case letters				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Enter name of any player in the search input field but make it case sensitive. 5. Click Enter 6. Verify search results	**Test Data:** Use name of any player with case sensitivity (e.g. "WiRtZ").	**Expected Result:** User can only see videos which contain that relevant player and search is case-insensitive.	**Actual Result:** User can only see videos which contain that relevant player and search is case-insensitive.	**Status:** PASS
Notes:				

```
35      @Test  Faris Colakovic
36      public void searchVideosByInputWithCaseSensitivity() {
37          home.clickOnSection( sectionName: "Video");
38          videos.openVideosSubSection("Browse");
39          videos.closeMarketingPopUp();
40          videos.enterSearchVideosValue( searchValue: "WiRtZ");
41          videos.validateAllVideoTitlesContainInputString( expectedTitle: "Wirtz");
42      }
```

Test Name: Test videos search by input with special characters				
Description: Check if user can search for any video by entering string with extra special characters				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Enter name of any player in the search input field but add some special characters 5. Click Enter 6. Verify search results	**Test Data:** Use name of any player with some extra special characters (e.g. “Wirtz!!”).	**Expected Result:** User can only see videos which contain that relevant player and special characters are ignored.	**Actual Result:** User can only see videos which contain that relevant player and special characters are ignored.	**Status:** PASS
Notes:				

```
44      @Test  Faris Čolaković
45      public void searchVideosByInputWithSpecialCharacters() {
46          home.clickOnSection( sectionName: "Video");
47          videos.openVideosSubSection("Browse");
48          videos.closeMarketingPopUp();
49          videos.enterSearchVideosValue( searchValue: "Wirtz!!");
50          videos.validateAllVideoTitlesContainInputString( expectedTitle: "Wirtz");
51      }
52  }
```

3.8. Validate videos filter

Users should be able to filter videos based on their duration to quickly find content that matches their preferred viewing time. This scenario verifies that the video length filter correctly categorizes videos as short, medium, or long, displays only the videos that meet the selected criteria, and updates the results dynamically when a different filter is applied.

Test Name: Test short videos filter				
Description: Check if user can filter only short videos and only short videos are displayed				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Click on the search filter icon 5. Select Short 6. Verify only short videos are shown	**Test Data:**	**Expected Result:** User can only see videos which are short form (less than 5 minutes).	**Actual Result:** User can only see videos which are short form (less than 5 minutes).	**Status:** PASS
Notes:				

```
7       @Test  Faris Čolaković
8       public void filterShortVideos() throws InterruptedException {
9           home.clickOnSection( sectionName: "Video");
10          videos.openVideosSubSection("Browse");
11          videos.closeMarketingPopUp();
12          videos.openVideoLengthFilter();
13          videos.selectVideoLengthFilter("Short");
14          videos.validateAllVideoTitlesHaveSetLength("Short");
15      }
```

Test Name: Test medium length videos filter				
Description: Check if user can filter only medium length videos and only medium length videos are displayed				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Click on the search filter icon 5. Select Medium 6. Verify only medium length videos are shown	**Test Data:**	**Expected Result:** User can only see videos which are of medium length (5-10 minutes with 05:00 and 10:00 included).	**Actual Result:** User can only see videos which are of medium length (5-10 minutes with 05:00 and 10:00 included).	**Status:** PASS
Notes:				

```
17      @Test  Faris Čolaković
18      public void filterMediumLengthVideos() throws InterruptedException {
19          home.clickOnSection( sectionName: "Video");
20          videos.openVideosSubSection("Browse");
21          videos.closeMarketingPopUp();
22          videos.openVideoLengthFilter();
23          videos.selectVideoLengthFilter("Medium");
24          videos.validateAllVideoTitlesHaveSetLength("Medium");
25      }
```

Test Name: Test long videos filter				
Description: Check if user can filter only long videos and only long videos are displayed				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Click on the search filter icon 5. Select Long 6. Verify only long videos are shown	**Test Data:**	**Expected Result:** User can only see videos which are long form (more than 10 minutes).	**Actual Result:** User can only see videos which are long form (more than 10 minutes).	**Status:** PASS
Notes:				

```
27      @Test  Faris Čolaković
28      public void filterLongVideos() throws InterruptedException {
29          home.clickOnSection( sectionName: "Video");
30          videos.openVideosSubSection("Browse");
31          videos.closeMarketingPopUp();
32          videos.openVideoLengthFilter();
33          videos.selectVideoLengthFilter("Long");
34          videos.validateAllVideoTitlesHaveSetLength("Long");
35      }
36  }
```

3.9. Validate video playback page

Users should be able to select a video from the list and view it on the playback page with the correct associated information. This scenario verifies that when a user clicks on a video, the playback page displays the correct video along with its corresponding title, description, thumbnail, and other relevant metadata, ensuring that the selected content matches the user's choice.

It is possible that Liverpool FC may introduce marketing offers, seasonal promotions (especially during Christmas and New Year period), or other pop-ups at their discretion. Currently, such pop-ups are handled in test automation using method closeMarketingPopUp(). Since these promotional elements are determined by the club, their timing, frequency or content cannot be fully predicted and may affect automated test execution in the future. In case of actual test automation work in the workplace, this would be handled by test maintanance user stories and these tests would be updated according to these future changes.

Test Name: Test free video title is correct				
Description: Check if opening a free video opens a correct video title				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Choose a random free video, note the title and open it 5. Verify video title is correct and correct video has been opened	**Test Data:**	**Expected Result:** User can see correct video opened. Opened video title is the same as the one noted in step 4.	**Actual Result:** User can see correct video opened. Opened video title is the same as the one noted in step 4.	**Status:** PASS
Notes:				

```
10      @Test  Faris Čolaković
11      public void openFreeVideoAndValidateTitle() {
12          home.clickOnSection( sectionName: "Video");
13          videos.openVideosSubSection("Browse");
14          videos.closeMarketingPopUp();
15          String videoTitle = videos.getHomeFreeVideoTitle( index: 1);
16          videos.openFreeAccessVideo( index: 1);
17          videos.validateVideoTitle(videoTitle);
18      }
```

Test Name: Test paid video will not play				
Description: Check if opening a paid video results in that video not playing. This protects the projects subscription plan and business model.				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Choose a random paid video and open it (paid videos have the FULL/VIDEO tag) 5. Verify video will not start playing	**Test Data:**	**Expected Result:** User can not watch the video because it is paid. Video will also not start playing automatically, or manually.	**Actual Result:** User can not watch the video because it is paid. Video will also not start playing automatically, or manually.	**Status:** PASS

Notes: It is worth mentioning that during automated testing, the "Browse" navigation behaves differently compared to manual testing. Specifically, when clicking "Browse" for the first time in automation, the user is redirected directly to the "All Red Videos" page. As a result, subsequent test steps require selecting "Browse" from the navigation bar again to continue testing other sections.

This behavior may occur because some websites implement mechanisms to detect automated browsing. Similar to platforms like Netflix, these mechanisms can restrict automated interactions or prevent scraping of video player data in order to protect content. While this does not affect normal manual usage, it can influence the flow and behavior of tests executed through automation.

```
30      @Test  Faris Čolaković
31      public void openPaidVideoAndValidateItDoesntPlay() {
32          home.clickOnSection( sectionName: "Video");
33          videos.openVideosSubSection("Browse");
34          videos.closeMarketingPopUp();
35          videos.openVideosSubSection("Browse"); //need to select Browse
36          videos.openFullAccessVideo( index: 1);
37          videos.validateVideoDoesntPlay();
38      }
```

Test Name: Test free video category is correct				
Description: Check if opening a free video opens a correct video category				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Choose a random free video, note the category and open it 5. Verify video category is correct and correct video has been opened	**Test Data:**	**Expected Result:** User can see correct video opened. Opened video category is the same as the one noted in step 4.	**Actual Result:** User can see correct video opened. Opened video category is the same as the one noted in step 4.	**Status:** PASS
Notes: There is a need to close the marketing pop ups that will occur every now and then, especially during the winter break and holiday season where different subscription models which offer extensive videos can happen. This is pretty much impossible to predict in test automation and at this point of creating tests, as the project (Liverpool FC) can create these marketing offers anytime and this can in return break tests.				

```
40      @Test  Faris Čolaković
41      public void openFreeVideoAndValidateCategoryName() {
42          home.clickOnSection( sectionName: "Video");
43          videos.openVideosSubSection("Browse");
44          videos.closeMarketingPopUp();
45          String videoCategory = videos.getFreeVideoCategoryFromBrowsePage( index: 1);
46          videos.openFreeAccessVideo( index: 1);
47          videos.validateVideoCategory(videoCategory);
48      }
```

Test Name: Test paid video category is correct				
Description: Check if opening a paid video opens a correct video category				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the Video button 3. Click on the Browse category 4. Choose a random paid video, note the category and open it 5. Verify video category is correct and correct video has been opened	**Test Data:**	**Expected Result:** User can see correct video opened. Opened video category is the same as the one noted in step 4.	**Actual Result:** User can see correct video opened. Opened video category is the same as the one noted in step 4.	**Status:** PASS
Notes: There is a need to close the marketing pop ups that will occur every now and then, especially during the winter break and holiday season where different subscription models which offer extensive videos can happen. This is pretty much impossible to predict in test automation and at this point of creating tests, as the project (Liverpool FC) can create these marketing offers anytime and this can in return break tests.				

```
50      @Test  Faris Čolaković
51      public void openPaidVideoAndValidateCategoryName() {
52          home.clickOnSection( sectionName: "Video");
53          videos.openVideosSubSection("Browse");
54          videos.closeMarketingPopUp();
55          String videoCategory = videos.getPaidVideoCategoryFromBrowsePage( index: 1);
56          videos.openFullAccessVideo( index: 1);
57          videos.validateVideoCategory(videoCategory);
58      }
59  }
```

3.10. Validate profile data

Users should be able to view their profile information accurately after logging in. This scenario verifies all user details displayed on the profile page (such as first name, last name, date of birth and subscription).

This test scenario also has certain test fixtures and pre-conditions such as beforeEach and afterEach:

```
@BeforeEach  Faris Čolaković
public void beforeEachTest(){
    home.clickSignIn();
    login.correctLoginForRefactoring();
}

@AfterEach  Faris Čolaković
public void afterEachTest(){
    home.logOutFromProfilePage();
}
```

Since user profile tests require user to be logged in, login will be achieved before and logout will be done after each test.

Test Name: Test correct first name in profile section				
Description: Check if correct first name is displayed for the user in the profile section				
Pre-condition(s): Register an account and note its first name. Using that same account log in				
Test Steps: 1. Go to the landing page 2. Click on the Account 3. Verify first name	**Test Data:**	**Expected Result:** User can see correct first name noted in the registration process.	**Actual Result:** User can see correct first name noted in the registration process.	**Status:** PASS
Notes:				

```
22      @Test  Faris Čolaković
23      public void openProfilePageAndValidateCorrectUserName() {
24          home.openAccountSettings();
25          profile.validateUserFirstName("Faris");
26      }
```

Test Name: Test correct last name in profile section				
Description: Check if correct last name is displayed for the user in the profile section				
Pre-condition(s): Register an account and note its last name. Using that same account log in				
Test Steps: 1. Go to the landing page 2. Click on the Account 3. Verify last name	**Test Data:**	**Expected Result:** User can see correct last name noted in the registration process.	**Actual Result:** User can see correct last name noted in the registration process.	**Status:** PASS
Notes:				

```
28      @Test  Faris Čolaković
29      public void openProfilePageAndValidateCorrectLastName() {
30          home.openAccountSettings();
31          profile.validateUserLastName("Colakovic");
32      }
```

Test Name: Test correct subscription name in profile section				
Description: Check if correct subscription name is displayed for the user in the profile section				
Pre-condition(s): Register an account and note its subscription name. Using that same account log in				
Test Steps: 1. Go to the landing page 2. Click on the Account 3. Verify subscription name	**Test Data:**	**Expected Result:** User can see correct subscription name noted in the registration process.	**Actual Result:** User can see correct subscription name noted in the registration process.	**Status:** PASS
Notes: By default, all users (without paying additional money) have the same subscription.				

```
34      @Test  Faris Čolaković
35      public void openProfilePageAndValidateCorrectSubscription() {
36          home.openAccountSettings();
37          profile.validateCorrectSubscription();
38      }
```

Test Name: Test correct date of birth in profile section				
Description: Check if correct date of birth is displayed for the user in the profile section				
Pre-condition(s): Register an account and note its date of birth. Using that same account log in				
Test Steps: 1. Go to the landing page 2. Click on the Account 3. Verify date of birth	**Test Data:**	**Expected Result:** User can see correct date of birth noted in the registration process.	**Actual Result:** User can see correct date of birth noted in the registration process.	**Status:** PASS
Notes:				

```
40      @Test  Faris Čolaković
41      public void openProfilePageAndValidateCorrectDateOfBirthFormat() {
42          home.openAccountSettings();
43          String dateOfBirth = profile.getDateOfBirth();
44          profile.validateUserDateOfBirthFormat(dateOfBirth);
45      }
```

3.11. Validate articles data

Users should be able to select an article from the list on the home page and view it on the special article page with the correct associated information. This scenario verifies that when a user clicks on an article, the article page displays the correct article along with its corresponding metadata.

Test Name: Test articles exist				
Description: Check if home page contains articles				
Pre-condition(s):				
Test Steps: 1. Go to the landing page	**Test Data:**	**Expected Result:** User can see at least 1 article.	**Actual Result:** User can see at least 1 article.	**Status:** PASS
Notes:				

```
9       @Test  Faris Čolaković
10      public void validateThereIsAtLeast1Article() { home.validateThereAreHomePageArticles(); }
```

Test Name: Test article page title				
Description: Check if opening an article opens correct article with correct title				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Choose a random article and note the title 3. Open that article 4. Verify the article title	**Test Data:**	**Expected Result:** Correct article has been opened. Article title is the same as the title noted in Step 2.	**Actual Result:** Correct article has been opened. Article title is the same as the title noted in Step 2.	**Status:** PASS
Notes:				

```
22      @Test  Faris Čolaković
23      public void validateArticleType() {
24          home.validateThereAreHomePageArticles();
25          String expectedType = home.getHomeArticleType( index: 1);
26          home.openArticleByIndex( index: 1,  allArticles: true);
27          articles.validateCorrectArticleType(expectedType);
28      }
```

Test Name: Test article page type				
Description: Check if opening an article opens correct article with correct type				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Choose a random article and note the type 3. Open that article 4. Verify the article type	**Test Data:**	**Expected Result:** Correct article has been opened. Article type is the same as the type noted in Step 2.	**Actual Result:** Correct article has been opened. Article type is the same as the type noted in Step 2.	**Status:** PASS
Notes:				

```
22      @Test  Faris Čolaković
23      public void validateArticleType() {
24          home.validateThereAreHomePageArticles();
25          String expectedType = home.getHomeArticleType( index: 1);
26          home.openArticleByIndex( index: 1,  allArticles: true);
27          articles.validateCorrectArticleType(expectedType);
28      }
```

Test Name: Test article page published time				
Description: Check if opening an article opens correct article with correct published time				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Choose a random article and note the published time 3. Open that article 4. Verify the article published time	**Test Data:**	**Expected Result:** Correct article has been opened. Article published time is the same as the published time noted in Step 2.	**Actual Result:** Correct article has been opened. Article published time is the same as the published time noted in Step 2.	**Status:** PASS
Notes:				

```
30      @Test  Faris Čolaković
31      public void validateArticlePublishedTime() {
32          home.validateThereAreHomePageArticles();
33          String expectedTime = home.getHomeArticleTime( index: 1).toLowerCase();
34          home.openArticleByIndex( index: 1,  allArticles: true);
35          articles.validateCorrectArticlePublishedTime(expectedTime);
36      }
```

Test Name: Test article image				
Description: Check if opening an article opens correct article with correct image				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Choose a random article and note the image 3. Open that article 4. Verify the article image	**Test Data:**	**Expected Result:** Correct article has been opened. Article image is the same as the image noted in Step 2.	**Actual Result:** Correct article has been opened. Article image is the same as the image noted in Step 2.	**Status:** PASS
Notes:				

```
38      @Test  Faris Čolaković
39      public void validateArticleImage() {
40          home.validateThereAreHomePageArticles();
41          String expectedImageAlt = home.getHomeImageAlt( index: 1);
42          home.openArticleByIndex( index: 1,  allArticles: false);
43          articles.validateCorrectImage(expectedImageAlt);
44      }
```

3.12. Validate landing page in different languages

Users should be able to change the application language directly from the landing page. This scenario verifies that selecting a different language updates the page accordingly.

Test Name: Test switch to German language				
Description: Check if changing language dropdown selector to German changes the language				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the language selector (by default EN) 3. Select Deutsch 4. Verify language change to German	**Test Data:**	**Expected Result:** UI has been switched to German language.	**Actual Result:** UI has been switched to German language.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void changeLanguageToGermanAndValidateChanges() {
    home.openLanguageDropdown();
    home.selectLanguage( languageText: "Deutsch");
    home.validateUrlEndsWith( languageCode: "de");
    home.validateAccountAndLoginButtons( accountText: "Verbinden",  loginText: "Anmeldung");
}
```

Test Name: Test switch to Spanish language				
Description: Check if changing language dropdown selector to Spanish changes the language				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the language selector (by default EN) 3. Select Spanish 4. Verify language change to Spanish	**Test Data:**	**Expected Result:** UI has been switched to Spanish language.	**Actual Result:** UI has been switched to Spanish language.	**Status:** PASS
Notes:				

```
16      @Test  Faris Čolaković
17      public void changeLanguageToSpanishAndValidateChanges() {
18          home.openLanguageDropdown();
19          home.selectLanguage( languageText: "Español");
20          home.validateUrlEndsWith( languageCode: "es");
21          home.validateAccountAndLoginButtons( accountText: "Unir",  loginText: "Acceso");
22      }
```

Test Name: Test switch to French language				
Description: Check if changing language dropdown selector to French changes the language				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the language selector (by default EN) 3. Select French 4. Verify language change to French	**Test Data:**	**Expected Result:** UI has been switched to French language.	**Actual Result:** UI has been switched to French language.	**Status:** PASS
Notes:				

```
24      @Test  Faris Čolaković
25      public void changeLanguageToFrenchAndValidateChanges() {
26          home.openLanguageDropdown();
27          home.selectLanguage( languageText: "Française");
28          home.validateUrlEndsWith( languageCode: "fr");
29          home.validateAccountAndLoginButtons( accountText: "Rejoindre",  loginText: "Se connecter");
30      }
```

Test Name: Test switch to Bahasa Indonesia language				
Description: Check if changing language dropdown selector to Bahasa Indonesia changes the language				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the language selector (by default EN) 3. Select Bahasa Indonesia 4. Verify language change to Bahasa Indonesia	**Test Data:**	**Expected Result:** UI has been switched to Bahasa Indonesia language.	**Actual Result:** UI has been switched to Bahasa Indonesia language.	**Status:** PASS
Notes:				

```
32      @Test  Faris Čolaković
33      public void changeLanguageToBahasaIndonesiaAndValidateChanges() {
34          home.openLanguageDropdown();
35          home.selectLanguage( languageText: "bahasa Indonesia");
36          home.validateUrlEndsWith( languageCode: "id");
37          home.validateAccountAndLoginButtons( accountText: "Bergabung",  loginText: "Gabung");
38      }
```

Test Name: Test switch to Japanese language				
Description: Check if changing language dropdown selector to Japanese changes the language				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the language selector (by default EN) 3. Select Japanese 4. Verify language change to Japanese	**Test Data:**	**Expected Result:** UI has been switched to Japanese language.	**Actual Result:** UI has been switched to Japanese language.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void changeLanguageToJapaneseAndValidateChanges() {
    home.openLanguageDropdown();
    home.selectLanguage( languageText: "日本語");
    home.validateUrlEndsWith( languageCode: "ja");
    home.validateAccountAndLoginButtons( accountText: "参加する", loginText: "ログイン");
}
```

Test Name: Test switch to Portuguese language				
Description: Check if changing language dropdown selector to Portuguese changes the language				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the language selector (by default EN) 3. Select Portuguese 4. Verify language change to Portuguese	**Test Data:**	**Expected Result:** UI has been switched to Portuguese language.	**Actual Result:** UI has been switched to Portuguese language.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void changeLanguageToPortugueseAndValidateChanges() {
    home.openLanguageDropdown();
    home.selectLanguage( languageText: "Português");
    home.validateUrlEndsWith( languageCode: "pt");
    home.validateAccountAndLoginButtons( accountText: "Juntar", loginText: "Acesso");
}
```

Test Name: Test switch to Thai language				
Description: Check if changing language dropdown selector to Thai changes the language				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the language selector (by default EN) 3. Select Thai 4. Verify language change to Thai	**Test Data:**	**Expected Result:** UI has been switched to Thai language.	**Actual Result:** UI has been switched to Thai language.	**Status:** PASS
Notes:				

```
64      @Test  Faris Čolaković
65      public void changeLanguageToThaiAndValidateChanges() {
66          home.openLanguageDropdown();
67          home.selectLanguage( languageText: "แบบไทย");
68          home.validateUrlEndsWith( languageCode: "th");
69          home.validateAccountAndLoginButtons( accountText: "เข้าร่วม",  loginText: "เข้าสู่ระบบ");
70      }
```

3.13. Validate news filters

Users should be able to filter news articles using the available filter options. This scenario verifies that applying a specific filter displays only the news items that match the selected category or criteria, ensuring that the filtering logic works correctly and returns accurate, relevant results.

Test Name: Test All News filter				
Description: Check if filtering news by All News filter displays correct news category				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the News 3. Click on All News 4. Validate All News section is opened	**Test Data:**	**Expected Result:** User is taken to the All News section of the News tab.	**Actual Result:** User is taken to the All News section of the News tab.	**Status:** PASS
Notes:				

```
9       @Test  Faris Čolaković
10      public void openNewsPageAndValidateFilterForAllNews() {
11          home.clickOnSection( sectionName: "News");
12          news.openNewsSubSection("All News");
13          news.validateCurrentUrl();
14          news.validateCorrectTabIsOpened( categoryName: "All news");
15      }
```

Test Name: Test Men filter				
Description: Check if filtering news by men filter displays correct news category (men)				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the News 3. Click on All News 4. Click on the Filter dropdown in top right corner 5. Select Men filter	**Test Data:**	**Expected Result:** Only news with the Men tag are being shown.	**Actual Result:** Only news with the Men tag are being shown.	**Status:** PASS
Notes:				

```
17      @Test  Faris Čolaković
18      public void openNewsPageAndValidateFilterForMen() {
19          home.clickOnSection( sectionName: "News");
20          news.openNewsSubSection("All News");
21          news.validateCurrentUrl();
22          news.filterByCategory( categoryName: "Men");
23          news.validateCorrectTabIsOpened( categoryName: "Men");
24      }
```

Test Name: Test women filter				
Description: Check if filtering news by women filter displays correct news category (women)				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the News 3. Click on All News 4. Click on the Filter dropdown in top right corner 5. Select women filter	**Test Data:**	**Expected Result:** Only news with the women tag are being shown.	**Actual Result:** Only news with the women tag are being shown.	**Status:** PASS
Notes:				

```
26      @Test  Faris Čolaković *
27      public void openNewsPageAndValidateFilterForWomen() {
28          home.clickOnSection( sectionName: "News");
29          news.openNewsSubSection("All News");
30          news.validateCurrentUrl();
31          news.filterByCategory( categoryName: "Women");
32          news.validateCorrectTabIsOpened( categoryName: "Women");
33      }
```

Test Name: Test academy filter				
Description: Check if filtering news by academy filter displays correct news category (academy)				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the News 3. Click on All News 4. Click on the Filter dropdown in top right corner 5. Select academy filter	**Test Data:**	**Expected Result:** Only news with the academy tag are being shown.	**Actual Result:** Only news with the academy tag are being shown.	**Status:** PASS
Notes:				

```
34      @Test  Faris Čolaković
35      public void openNewsPageAndValidateFilterForAcademy() {
36          home.clickOnSection( sectionName: "News");
37          news.openNewsSubSection("All News");
38          news.validateCurrentUrl();
39          news.filterByCategory( categoryName: "Academy");
40          news.validateCorrectTabIsOpened( categoryName: "Academy");
41      }
```

Test Name: Test media watch filter				
Description: Check if filtering news by academy filter displays correct news category (media watch)				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the News 3. Click on All News 4. Click on the Filter dropdown in top right corner 5. Select media watch filter	**Test Data:**	**Expected Result:** Only news with the media watch tag are being shown.	**Actual Result:** Only news with the media watch tag are being shown.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void openNewsPageAndValidateFilterForMediaWatch() {
    home.clickOnSection( sectionName: "News");
    news.openNewsSubSection("All News");
    news.validateCurrentUrl();
    news.filterByCategory( categoryName: "Media Watch");
    news.validateCorrectTabIsOpened( categoryName: "Media Watch");
}
```

Test Name: Test club filter				
Description: Check if filtering news by academy filter displays correct news category (club)				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the News 3. Click on All News 4. Click on the Filter dropdown in top right corner 5. Select club filter	**Test Data:**	**Expected Result:** Only news with the club tag are being shown.	**Actual Result:** Only news with the club tag are being shown.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void openNewsPageAndValidateFilterForClub() {
    home.clickOnSection( sectionName: "News");
    news.openNewsSubSection("All News");
    news.validateCurrentUrl();
    news.filterByCategory( categoryName: "Club");
    news.validateCorrectTabIsOpened( categoryName: "Club");
}
```

Test Name: Test tickets filter				
Description: Check if filtering news by academy filter displays correct news category (tickets)				
Pre-condition(s):				
Test Steps: 1. Go to the landing page 2. Click on the News 3. Click on All News 4. Click on the Filter dropdown in top right corner 5. Select tickets filter	**Test Data:**	**Expected Result:** Only news with the tickets tag are being shown.	**Actual Result:** Only news with the tickets tag are being shown.	**Status:** PASS
Notes:				

```
61      @Test  Faris Čolaković *
62      public void openNewsPageAndValidateFilterForTickets() {
63          home.clickOnSection( sectionName: "News");
64          news.openNewsSubSection("All News");
65          news.validateCurrentUrl();
66          news.filterByCategory( categoryName: "Tickets");
67          news.validateCorrectTabIsOpened( categoryName: "Tickets");
68      }
```

3.14. Validate HTTPS enforcement

The application must enforce secure communication over HTTPS for all pages. This scenario verifies that any attempt to access the site using an insecure HTTP connection is automatically redirected to HTTPS, ensuring that user data, authentication details, and all transmitted information remain encrypted and protected from interception.

```
15      @BeforeEach  Faris Čolaković
16      public  void loginBeforeAll() {
17          home.clickSignIn();
18          login.correctLoginForRefactoring();
19          home.validateUserIsCorrectlyLoggedInAfterRegisteringWithoutLogout();
20      }
21
22      @AfterEach  Faris Čolaković
23      public void logoutAfterAll() {
24          home.logOut();
25      }
```

Test Name: Test homepage HTTPS enforcement				
Description: Check if opening homepage url with HTTP enforces HTTPS				
Pre-condition(s):				
Test Steps: 1. Open http://www.liverpoolfc.com	**Test Data:**	**Expected Result:** HTTPS is enforced and user is redirected to https://www.liverpoolfc.com	**Actual Result:** HTTPS is enforced and user is redirected to https://www.liverpoolfc.com	**Status:** PASS
Notes:				

```
9       @Test  Faris Čolaković
10      public void testHttpHomepageRedirectsToHttps() throws Exception {
11          driver.get("http://www.liverpoolfc.com");
12          String current = driver.getCurrentUrl();
13          Assertions.assertTrue(current.startsWith("https://"),
14                  message: "Expected redirect to https but was: " + current);
15      }
```

Test Name: Test news HTTPS enforcement				
Description: Check if opening news url with HTTP enforces HTTPS				
Pre-condition(s):				
Test Steps: 1. Open http://www.liverpoolfc.com/news	**Test Data:**	**Expected Result:** HTTPS is enforced and user is redirected to https://www.liverpoolfc.com/news	**Actual Result:** HTTPS is enforced and user is redirected to https://www.liverpoolfc.com/news	**Status:** PASS
Notes:				

```
17      @Test  Faris Čolaković
18      public void testHttpNewsRedirectsToHttps() throws Exception {
19          driver.get("http://www.liverpoolfc.com/news");
20          String current = driver.getCurrentUrl();
21          Assertions.assertTrue(current.startsWith("https://"),
22                  message: "Expected video browse to redirect to https but was: " + current);
23      }
```

Test Name: Test fixtures HTTPS enforcement				
Description: Check if opening fixtures url with HTTP enforces HTTPS				
Pre-condition(s):				
Test Steps: 1. Open http://www.liverpoolfc.com/fixtures	**Test Data:**	**Expected Result:** HTTPS is enforced and user is redirected to https://www.liverpoolfc.com/fixtures	**Actual Result:** HTTPS is enforced and user is redirected to https://www.liverpoolfc.com/fixtures	**Status:** PASS
Notes:				

```
25      @Test  Faris Čolaković
26      public void testHttpFixturesRedirectsToHttps() throws Exception {
27          driver.get("http://video.liverpoolfc.com/fixtures");
28          String current = driver.getCurrentUrl();
29          Assertions.assertTrue(current.startsWith("https://"),
30                  message: "Expected video browse to redirect to https but was: " + current);
31      }
```

Test Name: Test shop HTTPS enforcement				
Description: Check if opening shop url with HTTP enforces HTTPS				
Pre-condition(s):				
Test Steps: 1. Open http://www.liverpoolfc.com/shop	**Test Data:**	**Expected Result:** HTTPS is enforced and user is redirected to https://www.liverpoolfc.com/shop	**Actual Result:** HTTPS is enforced and user is redirected to https://www.liverpoolfc.com/shop	**Status:** PASS
Notes:				

```
33      @Test  Faris Čolaković
34      public void testHttpShopRedirectsToHttps() throws Exception {
35          driver.get("http://video.liverpoolfc.com/shop");
36          String current = driver.getCurrentUrl();
37          Assertions.assertTrue(current.startsWith("https://"),
38                  message: "Expected video browse to redirect to https but was: " + current);
39      }
```

3.15. Validate Cookies flags

The project must store authentication cookies securely to protect user sessions. This scenario verifies that all relevant cookies (such as access token) include proper security attributes such as Secure, HttpOnly and that access token has a valid expiration time set. These measures help ensure that cookies cannot be accessed by client side scripts.

Since user profile tests require user to be logged in, login will be achieved before and logout will be done after each test.

```
@BeforeEach  Faris Čolaković
public  void loginBeforeAll() {
    home.clickSignIn();
    login.correctLoginForRefactoring();
    home.validateUserIsCorrectlyLoggedInAfterRegisteringWithoutLogout();
}

@AfterEach  Faris Čolaković
public void logoutAfterAll() {
    home.logOut();
}
```

Test Name: Test access token httponly flag is set				
Description: Check that access token contains httponly flag				
Pre-condition(s): User is logged in with a relevant account previously registered				
Test Steps: 1. Open the landing page 2. Right click on the page and select inspect 3. Click on Application 4. Find lfc_sso_access_token 5.Verify httponly flag of cookie from Step 4	**Test Data:**	**Expected Result:** There is a lfc_sso_access_token which has HttpOnly flag set.	**Actual Result:** There is a lfc_sso_access_token which has HttpOnly flag set.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void testCookieHttpOnlyFlag() {
    Set<Cookie> cookies = driver.manage().getCookies();

    for (Cookie cookie : cookies) {
        if (cookie.getName().equals("lfc_sso_access_token")) {
            Assertions.assertTrue(cookie.isHttpOnly(),
                    message: cookie.getName() + " should have HttpOnly flag set.");
        }
    }
}
```

Test Name: Test access token Secure flag is set				
Description: Check that access token contains Secure flag				
Pre-condition(s): User is logged in with a relevant account previously registered				
Test Steps: 1. Open the landing page 2. Right click on the page and select inspect 3. Click on Application 4.Find lfc_sso_access_token 5. Verify Secure flag of cookie from Step 4	**Test Data:**	**Expected Result:** There is a lfc_sso_access_token which has Secure flag set.	**Actual Result:** There is a lfc_sso_access_token which has Secure flag set.	**Status:** PASS
Notes:				

```
39      @Test  Faris Čolaković
40      public void testCookieSecureFlag() {
41          Set<Cookie> cookies = driver.manage().getCookies();
42
43          for (Cookie cookie : cookies) {
44              if (cookie.getName().equals("lfc_sso_access_token")) {
45                  Assertions.assertTrue(cookie.isSecure(),
46                          message: cookie.getName() + " should have Secure flag set.");
47              }
48          }
49      }
```

Test Name: Test access token has expiry set				
Description: Check that access token has expiry set				
Pre-condition(s): User is logged in with a relevant account previously registered				
Test Steps: 1. Open the landing page 2. Right click on the page and select inspect 3. Click on Application 4.Find lfc_sso_access_token 5. Verify cookie from Step 4 has expiry set	**Test Data:**	**Expected Result:** There is a lfc_sso_access_token which has expiry set.	**Actual Result:** There is a lfc_sso_access_token which has expiry set.	**Status:** PASS

Notes:

```
@Test  Faris Čolaković
public void testCookieExpiry() {
    Set<Cookie> cookies = driver.manage().getCookies();

    for (Cookie cookie : cookies) {
        if (cookie.getName().equals("lfc_sso_access_token")) {
            Assertions.assertNotNull(cookie.getExpiry(),  message: cookie.getName() + " should have an expiry set.");
        }
    }
}
```

3.16. Validate Session handling

The application must correctly manage authentication cookies throughout the user session. This scenario verifies that authentication cookies are properly created and persist while navigating through different pages after login, ensuring an uninterrupted authenticated experience. It also ensures that sso access cookies is removed or invalidated immediately after logging out.

Since user profile tests require user to be logged in, login will be achieved before and logout will be done after each test.

```
    private boolean loggedOut = false;  3 usages

    @BeforeEach  Faris Čolaković
    public  void loginBeforeAll() {
        home.clickSignIn();
        login.correctLoginForRefactoring();
    }

    @AfterEach  Faris Čolaković
    public void logoutAfterAll() {
        if (!loggedOut) home.logOut();
    }
```

Test Name: Test sso access token after login				
Description: Check that sso access token is created and exists after login				
Pre-condition(s): User is logged in with a relevant account previously registered				
Test Steps: 1. Open the landing page 2. Right click on the page and select inspect 3. Click on Application 4.Find lfc_sso_access_token	**Test Data:**	**Expected Result:** There is a lfc_sso_access_token after login.	**Actual Result:** There is a lfc_sso_access_token after login.	**Status:** PASS
Notes:				

```
31      @Test  Faris Čolaković
32      public void testSessionCookieExistsAfterLogin() {
33          home.validateUserIsCorrectlyLoggedInAfterRegisteringWithoutLogout();
34          Cookie sessionCookie = driver.manage().getCookieNamed("lfc_sso_access_token");
35          Assertions.assertNotNull(sessionCookie,  message: "Session cookie should exist after login.");
36      }
```

Test Name: Test sso access token after login				
Description: Check that sso access token is created and exists after login				
Pre-condition(s): User is logged in with a relevant account previously registered				
Test Steps: 1. Open the news page 2. Right click on the page and select inspect 3. Click on Application 4.Find lfc_sso_access_token 5. Navigate to the landing page 6. Right click on the page and select inspect 7. Click on Application 8. Find token from Step 4	**Test Data:**	**Expected Result:** There is a lfc_sso_access_token after login and it persists after navigating to news and landing page.	**Actual Result:** There is a lfc_sso_access_token after login and it persists after navigating to news and landing page.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void testSessionPersistsAcrossNavigation() {
    home.validateUserIsCorrectlyLoggedInAfterRegisteringWithoutLogout();
    // Navigate elsewhere
    driver.get("https://www.liverpoolfc.com/news");
    Cookie sessionCookie = driver.manage().getCookieNamed("lfc_sso_access_token");
    Assertions.assertNotNull(sessionCookie, message: "Session cookie should exist after login.");
    driver.get("https://www.liverpoolfc.com");
    Cookie sessionCookieSecondTry = driver.manage().getCookieNamed("lfc_sso_access_token");
    Assertions.assertNotNull(sessionCookieSecondTry, message: "Session cookie should still exist after login.");
}
```

Test Name: Test sso access token behavior on logout				
Description: Check that sso access token is removed on logout				
Pre-condition(s): User is logged in with a relevant account previously registered				
Test Steps: 1. Click on logout in the top right corner of the landing page 2. Right click on page and click on inspect 3. Verify there is no lfc_sso_access token anymore	**Test Data:**	**Expected Result:** There is no lfc_sso_access_token after logout. It can not be found in the cookies list.	**Actual Result:** There is no lfc_sso_access_token after logout. It can not be found in the cookies list.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void testSessionCookieRemovedAfterLogout() {
    home.validateUserIsCorrectlyLoggedInAfterRegistering();
    Cookie sessionCookie = driver.manage().getCookieNamed("lfc_sso_access_token");
    Assertions.assertNull(sessionCookie, message: "Session cookie should not exist after logout.");
    loggedOut=true;
}
```

3.17. Validate navigation

Users should be able to navigate from the homepage to primary sections of the application using the top navigation bar. This scenario verifies that each navigation item is visible, clickable, and correctly routes the user to the corresponding page.

Test Name: Test news section route				
Description: Check that navigation to news section works correctly				
Pre-condition(s):				
Test Steps: 1. Open the landing page 2. Click on the news section in the top bar	**Test Data:**	**Expected Result:** News page is opened.	**Actual Result:** News page is opened.	**Status:** PASS
Notes:				

```
@Test  Faris Čolaković
public void openNewsPageAndValidateNavigation() {
    home.clickOnSection( sectionName: "News");
    news.openNewsSubSection("All News");
    news.validateCurrentUrl();
}
```

Test Name: Test video browse section route				
Description: Check that navigation to video browse section works correctly				
Pre-condition(s):				
Test Steps: 1. Open the landing page 2. Click on the video section in the top bar 3. Click on the video browse section	**Test Data:**	**Expected Result:** Video browse page is opened.	**Actual Result:** Video browse page is opened.	**Status:** PASS
Notes:				

```
17      @Test  Faris Čolaković
18      public void openVideoPageAndValidateNavigation() {
19          home.clickOnSection( sectionName: "Video");
20          news.openNewsSubSection("Browse");
21          videos.validateCurrentUrl();
22      }
```

Test Name: Test profile route				
Description: Check that navigation to account page works correctly				
Pre-condition(s): User is logged in with previously registered account.				
Test Steps: 1. Open the landing page 2.Click on the Account button in the top bar.	**Test Data:**	**Expected Result:** Account page page is opened.	**Actual Result:** Account page is opened.	**Status:** PASS
Notes:				

```
24      @Test  Faris Čolaković
25      public void openProfilePageAndValidateNavigation() {
26          home.clickSignIn();
27          login.login( user: "fariscolakovic00@gmail.com",  pass: "hBSBR!LKDsna1");
28          home.openAccountSettings();
29          profile.validateCurrentUrl();
30      }
```

Test Name: Test reset password route				
Description: Check that navigation to reset password page works correctly				
Pre-condition(s):				
Test Steps: 1. Open the landing page 2.Click on the Login buton 3. Click on Forgot password button	**Test Data:**	**Expected Result:** Reset password page is opened.	**Actual Result:** Reset password page is opened.	**Status:** PASS
Notes:				

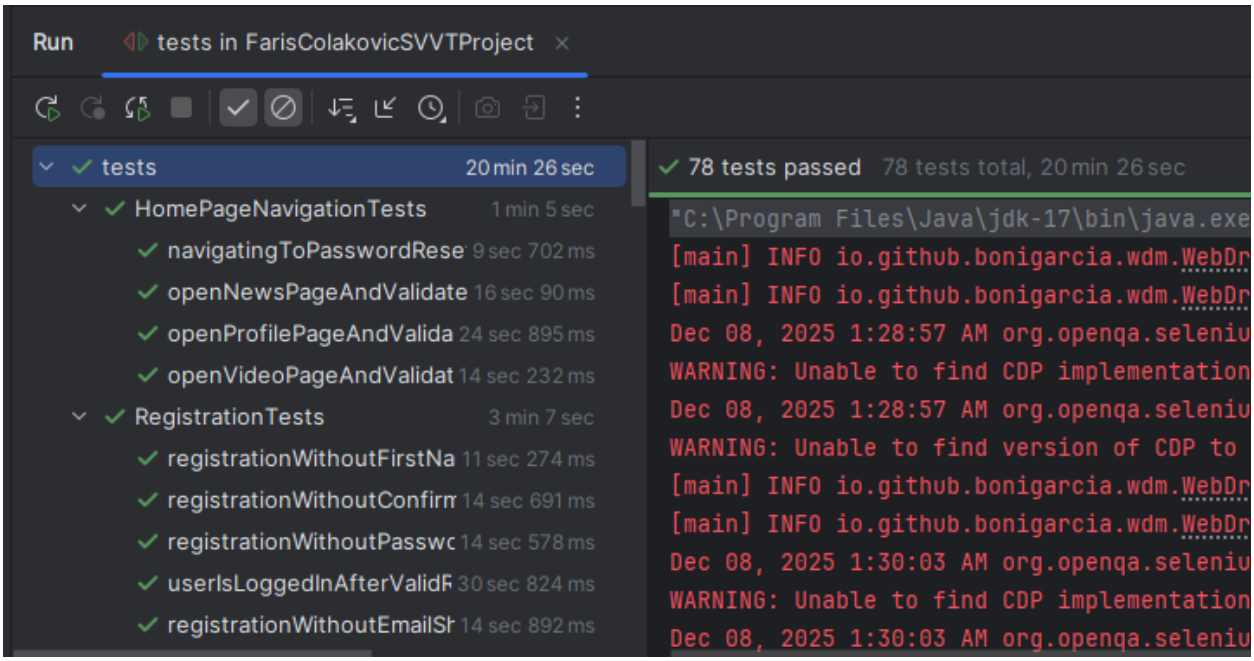
```
@Test  new *
public void navigatingToPasswordResetThroughLoginPage () {
    home.clickSignIn();
    login.clickForgetPassword();
    login.validateUserIsOnResetPasswordPage();
}
```

4. Conclusion

4.1. Testing Summary

Testing Tool	Total Tests	Passed Tests	Failed Tests
Test were written in Selenium framework paired with Java programming language. JUnit package was used for assertions. IDE used was Intellij IDEA.	78	78	0

All the tests ran simultaneously:



9.2. Final Thoughts

Overall, the platform appears to be well implemented with structured functionality and a generally smooth user experience. Most features behave consistently and the core workflows such as registration, login, browsing content, and viewing videos are implemented in a clear and reliable manner.

During testing, I did identify a few areas where improvements could enhance stability and maintainability but I also have a few interesting observations and plus points for the project. For example, the website contains mechanisms that seem to detect automated test execution, particularly on the video playback pages, which prevents automated retrieval or scraping of video playback data. I have noticed this same mechanicsm on other video playback sites such as Netflix.

This is a positive security measure from the project perspective. It even blocks automation when using special ChromeDriver options with an idea to hide automation, but it does introduce limitations for automated testing.

Dynamic marketing pop-ups also appear frequently, and I expect additional seasonal pop-ups (e.g., during New Year or Christmas promotions), which may interfere with automated test flows. While the tests currently handle these using a closeMarketingPopUp() method, future changes or new popups could still require updates on the automation side. This is a decision on clubs' side which I have nothing to do with. In a real workplace, I would handle this by using a maintanance user story and updating tests whenever new popups or marketing offers are added.

In some parts of the application, asynchronous loading is noticeable. For instance, when displaying the total number of available videos in the browse videos section, the page initially shows "0 results" for a moment before updating to the correct value (over 28.000 at the time of writing this). This is likely to happen due to an AJAX call, and while not functionally incorrect, it might be worth improving to avoid misleading intermediate states.

One more thing that I noticed is that a lot of web elements contain "data-testid" attribute which is a great example of developers taking test automation in consideration and helping test automation engineers to find the needed elements for their tests in an easier way.

Overall, the project demonstrates solid implementation quality, with only a few minor usability or automation related considerations that could be refined to improve both user experience and testability in future iterations and help test engineers do their work even more efficiently.