SOFTWARE VERIFICATION, VALIDATION AND TESTING

TESTING DOCUMENTATION

PROJECT NAME

Prepared by:
Rania Weiss

Proposed to:
Samed Jukić, Assist. Prof. Dr.
Aldin Kovačević, Teaching Assistant

Date of submission

TABLE OF CONTENTS

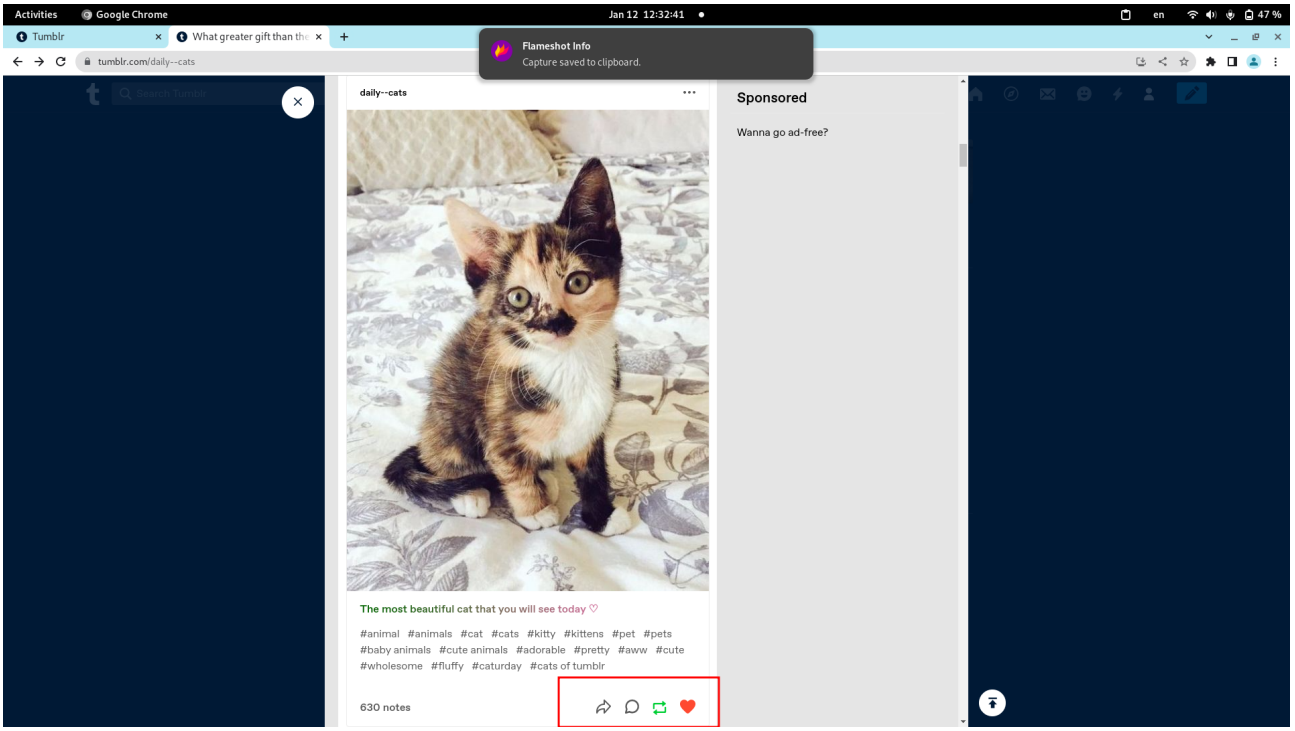
1. Introduction

1.1. About the Project

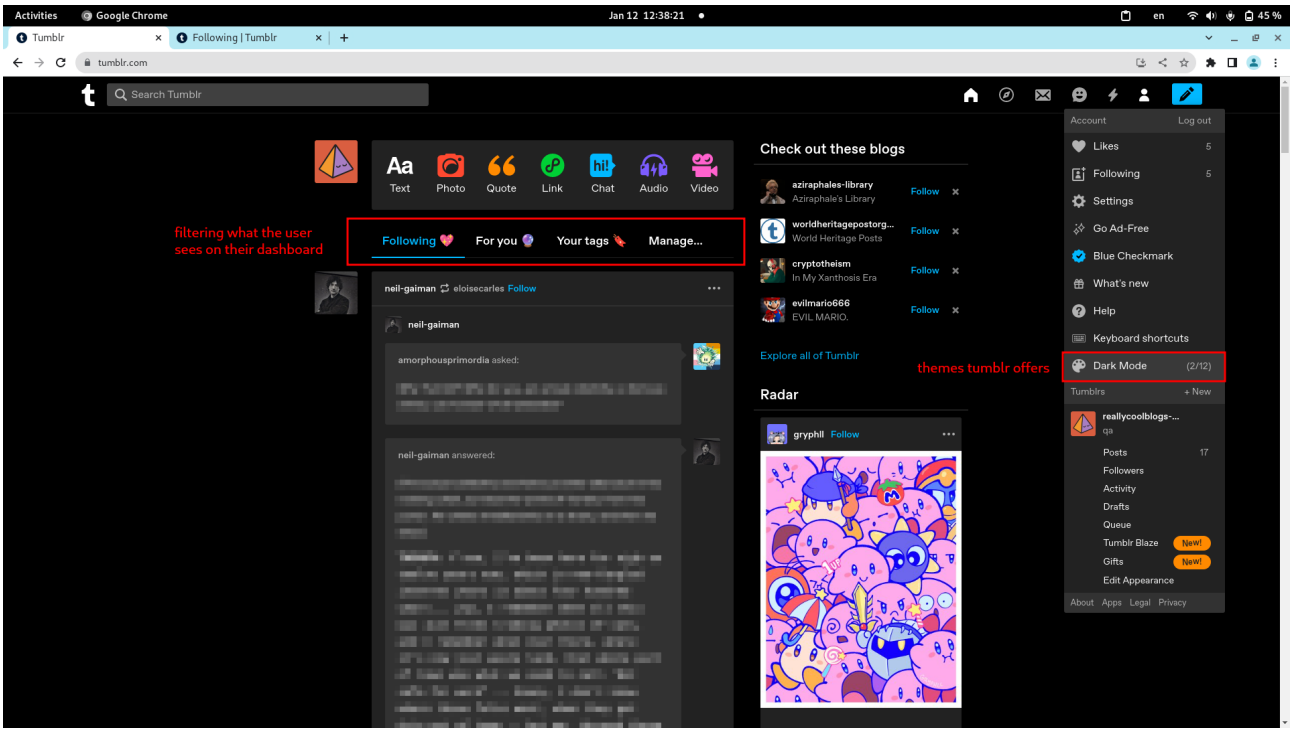
Tumblr is a blogging and social media platform. It is considered to be more complex to use than other social media platforms (Instagram, Twitter), which is why the user base is not large. https://www.tumblr.com/

1.2. Project Functionalities and Screenshots

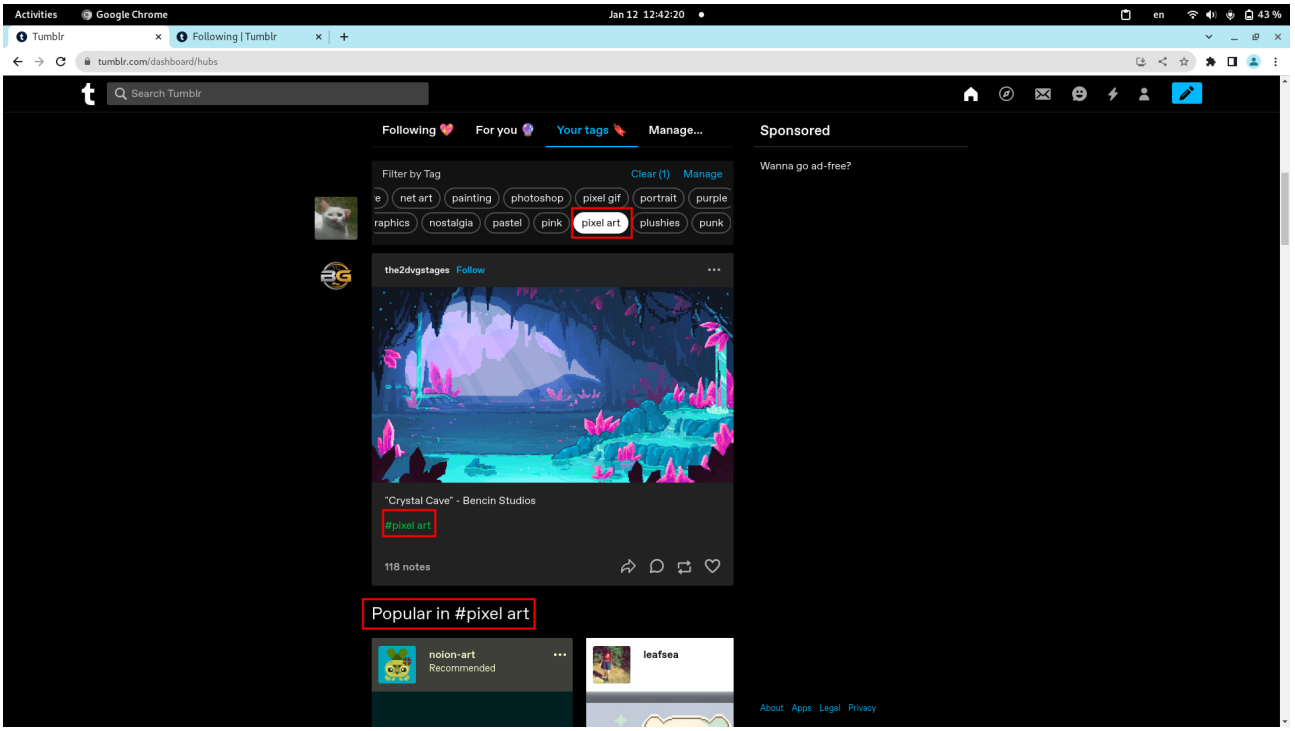
- Notes - interacting with posts; include:
    - Replies
    - Likes
    - Reblogs



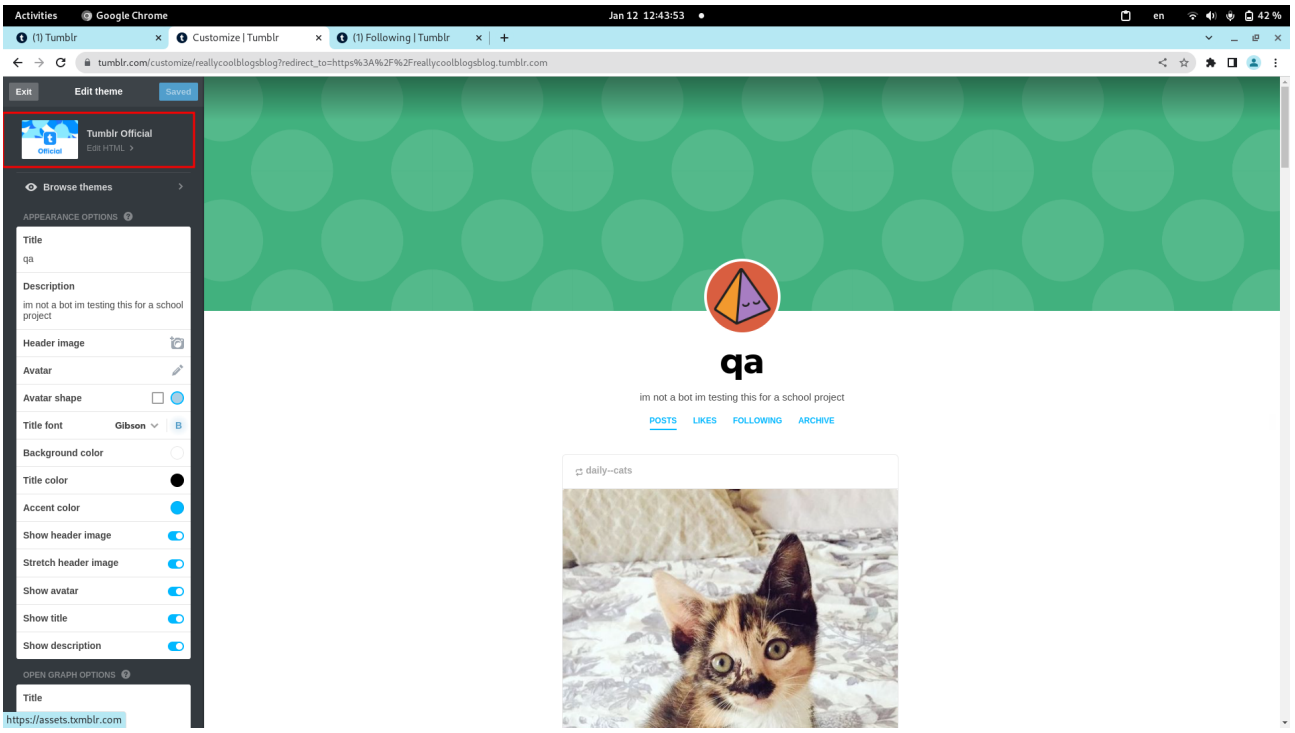
- Blog management
    - Dashboard - the user can filter which posts they would like to see on their dashboard based on:
        - following
        - posts the user might be interested in
        - tags

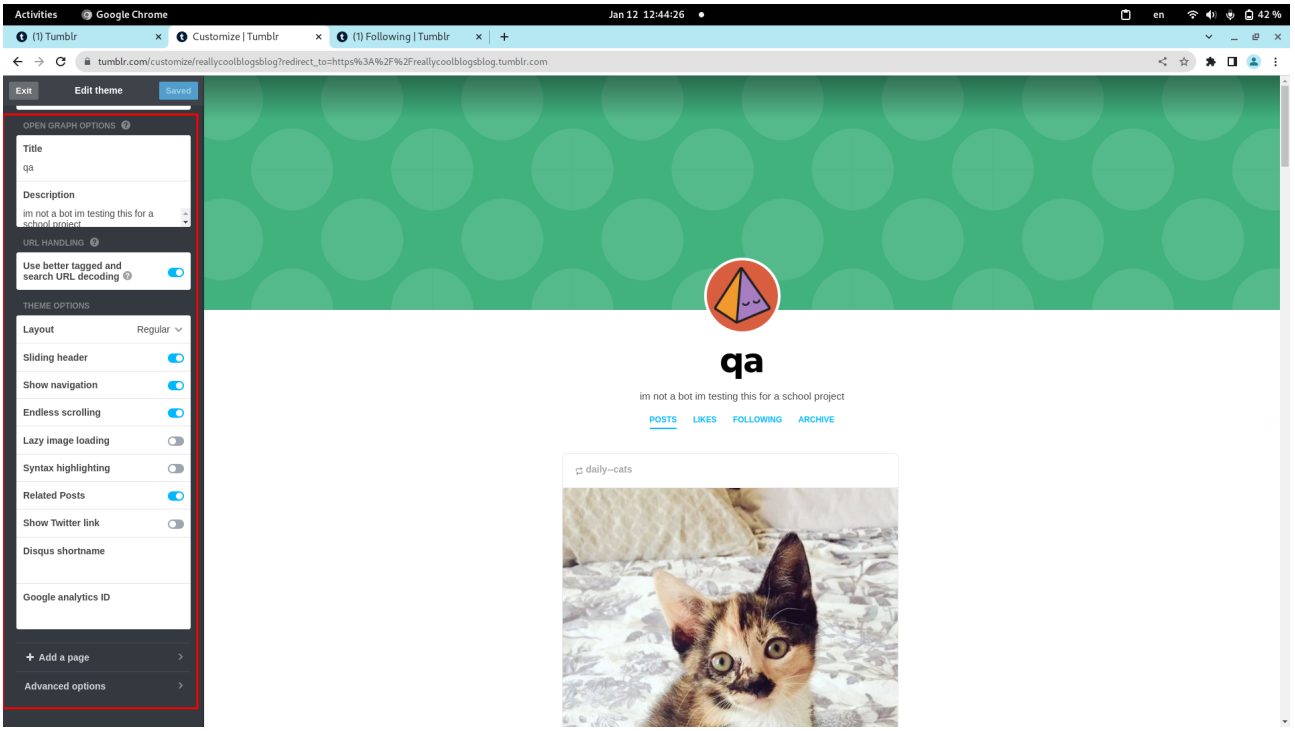


- Tags - tags allow the user to find specific content from blogs all over tumblr

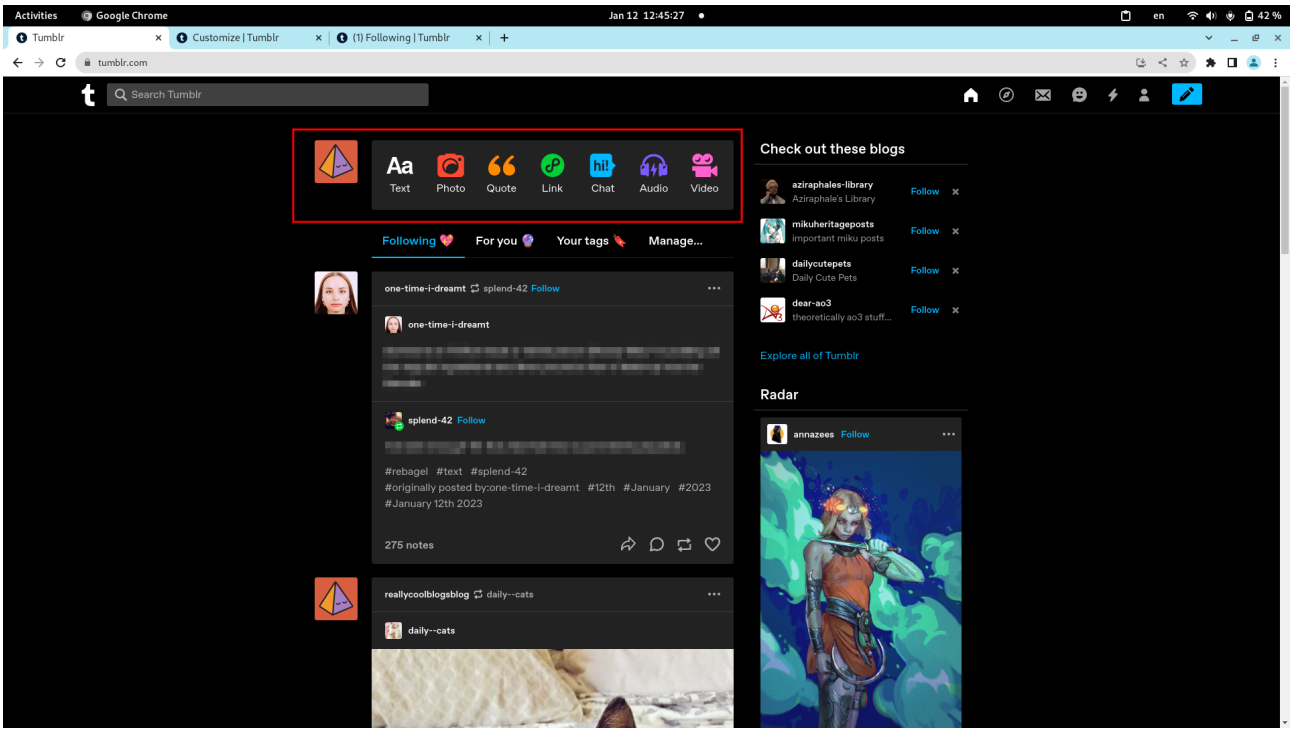


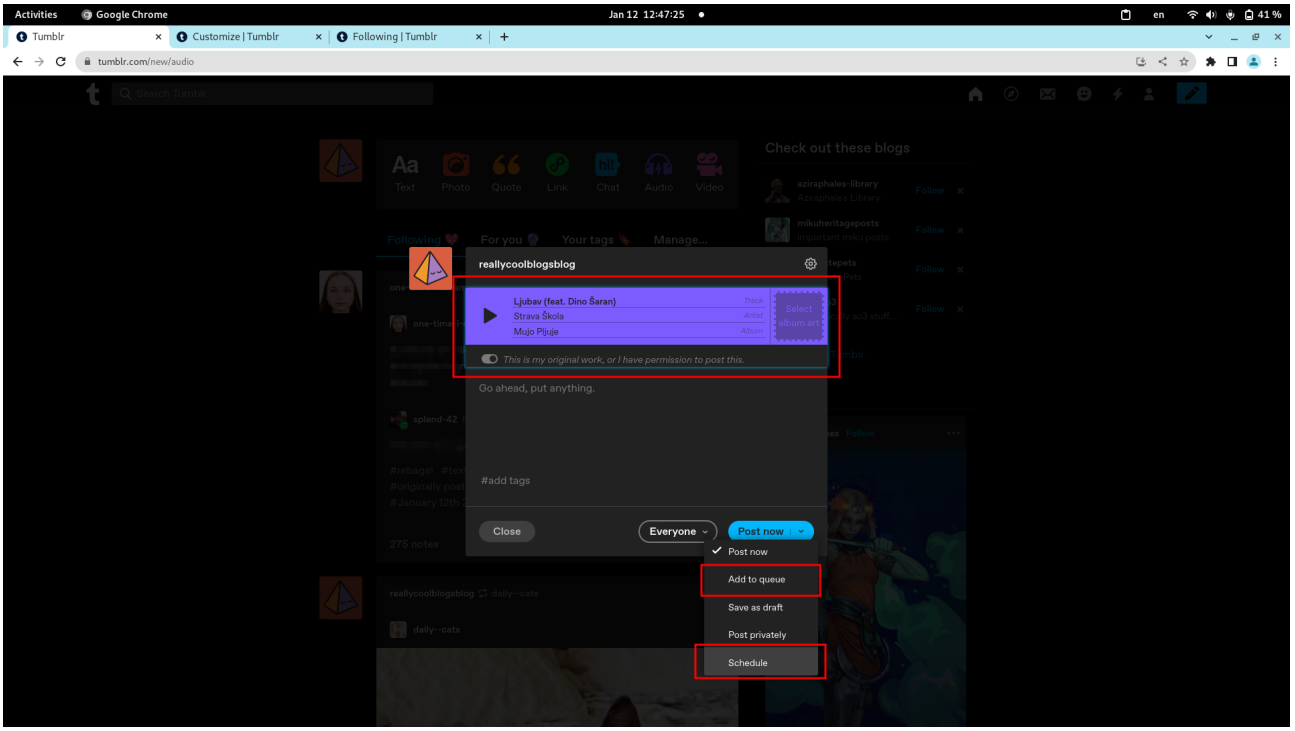
- blog editing



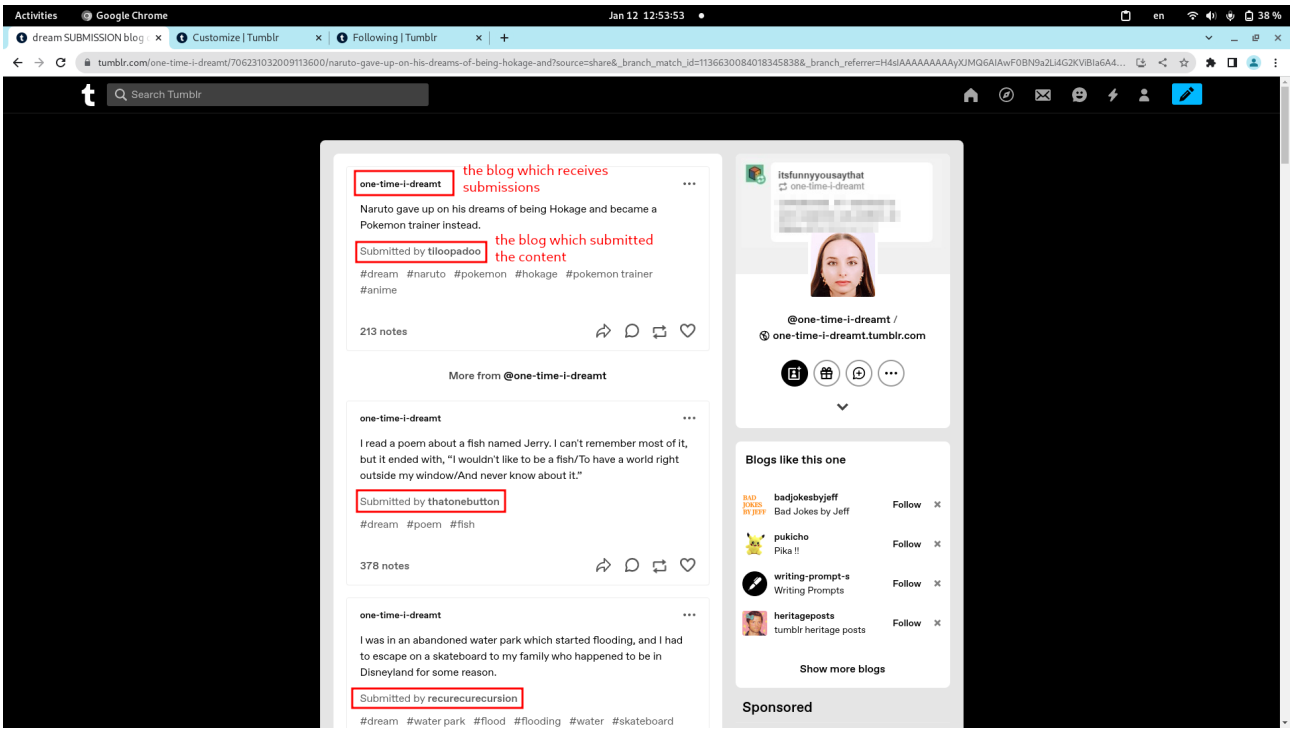


- Posts
  - post scheduling and queuing - the user can decide not to post content in that moment, rather set a specific time to post it
  - supports uploading images, text, links, audio, video, etc.

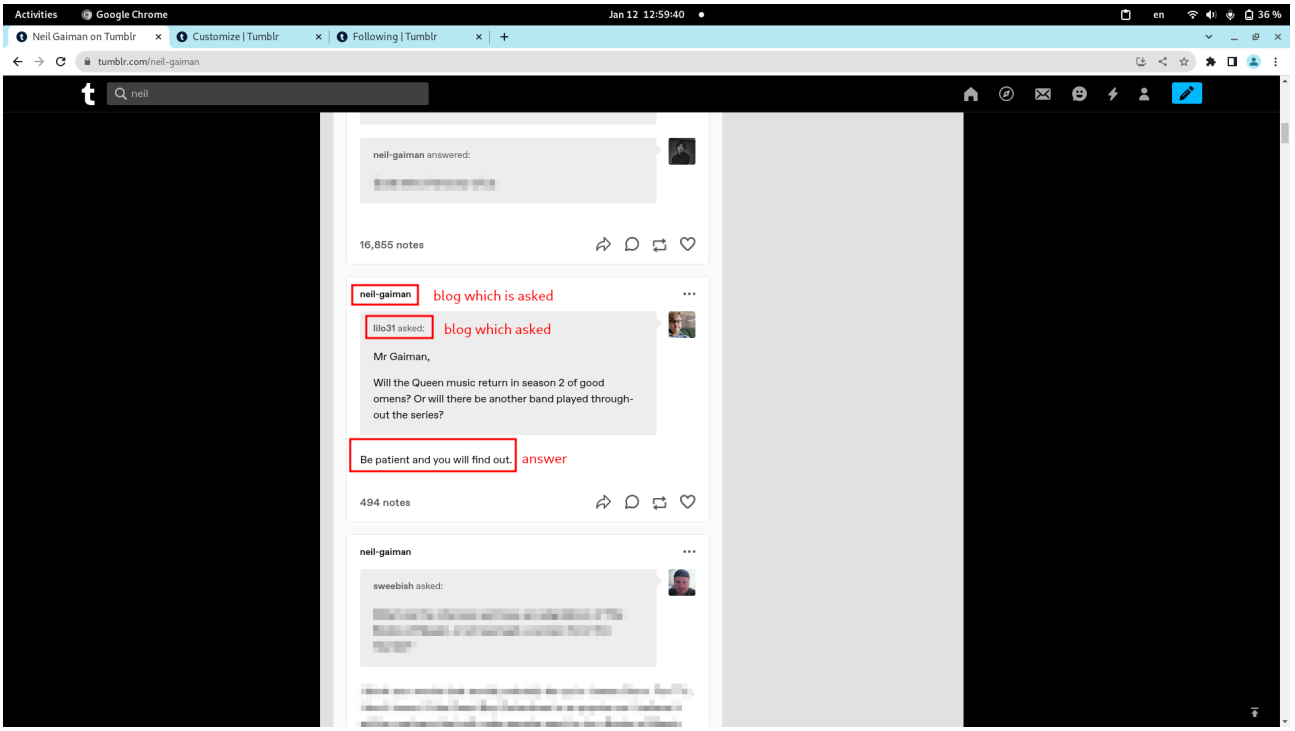




- Messaging - other than traditional messaging, it is possible to send other blog submissions and asks, if they have selected in their settings to allow them. It is possible to send asks and submissions with the user's username, or anonymously
    - submissions - users send to other blogs something they would like the other blog to post



- asks - users ask blog owner questions



- chats

2. Test Plan

2.1. Scope

I intend to test the features of posting and reblogging content, the features relating to following other blogs. Login will be tested as well. Basic customization of blog and dashboard will be tested. Features relating to messaging (chatting, asks, submissions) will not be tested. Also, elaborate customization of a blog will not be tested.

2.2. Testing Environment and Tools

For this project I am using Selenium, Java, and JUnit 5.

3. Test Execution

The structure of the project is divided into six components:

1. Login
    a. Testing if a user can properly log in, and what happens when a user inputs invalid or unregistered data
2. Unregistered user
    a. it is not possible to do much when unregistered, so the majority of tests end with the user being prompted to log in or register
3. Posts
    a. Testing creating a simple post, and reblogging a post with the user's content
4. Notes
    a. Testing reblogging and liking other blogs' posts on their regular and customized blogs
5. Customization
    a. Simple customization of blog name, description, and changing the color scheme of the app
6. Follow
    a. Testing the follow/unfollow mechanisms

3.1 Login

There are three login tests, the first one being logging in with a registered email, the second attempting to login with an email which is not registered, and the third and final test will attempt to log in with data that is not an email at all. Registration is a lengthy and complex process, consisting of a lot of steps, which is why it was skipped.

```
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
class Login {
    public static WebDriver webDriver;
    public static String baseUrl;
    public static WebDriverWait wait;

    @BeforeAll
    static void setUpBeforeClass() throws Exception {
        System.setProperty("webdriver.chrome.driver", "/home/rani/V Semester/SVVT/chromedriver_linux64/chromedriver");
        ChromeOptions options = new ChromeOptions();
        options.addArguments("--no-sandbox");
        options.addArguments("--start-maximized");
        options.addArguments("--incognito");
        webDriver = new ChromeDriver(options);
        baseUrl = "https://www.tumblr.com/";
        wait = new WebDriverWait(webDriver, Duration.ofSeconds(5));
    }

    @AfterAll
    static void tearDownAfterClass() throws Exception {
        webDriver.quit();
    }

    @BeforeEach
    void setUp() throws Exception {
        webDriver.get(baseUrl);
        webDriver.findElement(By.linkText("Log in")).click();
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.className("EaAON")));
        webDriver.findElement(By.xpath("/html/body/div[1]/div/div[4]/div/div/div/div[2]/div/form/div[1]/div/button"))
                .click();
    }

    @AfterEach
    void tearDown() throws Exception {
    }
```

3.1.1 Login

Test Name: Test logging in				
Description: Log in using valid credentials				
Pre-condition(s): Have existing account				
Test Steps: 1. Go to home page 2. Click log in button 3. Select continue with email 4. Enter email address and click next 5. Enter password and log in	**Test Data:** email: rpbjfgnumsfqwglmkk@tmmwj.net password: ThisIsAPassword!!12	**Expected Result:** The user is logged in and has landed on the dashboard	**Actual Result:** The user is logged in and has landed on the dashboard	**Status:** PASS
Notes: the email was created using a 10-minute-email website				

```
@Test
@Order(3)
void login() throws InterruptedException {
    WebElement toPass = webDriver
            .findElement(By.xpath("/html/body/div[1]/div/div[4]/div/div/div/div[2]/div/form/div[1]/div[1]/button"));
    // valid email
    webDriver.findElement(By.name("email")).sendKeys("rpbjfgnumsfqwglmkk@tmmwj.net");
    // waiting until button becomes enabled
    wait.until(ExpectedConditions.elementToBeClickable(toPass));
    toPass.click();
    // waiting until password field can be modified
    wait.until(ExpectedConditions.elementToBeClickable(By.name("password")));
    webDriver.findElement(By.name("password")).sendKeys("ThisIsAPassword!!12");
    // waiting until log in button is enabled
    WebElement logInButton = webDriver.findElement(
            By.xpath("/html/body/div[1]/div/div[4]/div/div/div/div[2]/div/form/div[1]/div[1]/div/button"));
    wait.until(ExpectedConditions.elementToBeClickable(logInButton));

    logInButton.click();
    Thread.sleep(2500);
    // after log in should go to dashboard
    String currUrl = webDriver.getCurrentUrl();
    assertEquals("https://www.tumblr.com/dashboard", currUrl);
}
```

3.1.2 Login with no account

Test Name: Test logging in with unregistered email				
Description: Log in using valid credentials				
Pre-condition(s): Have existing account				
Test Steps: 1. Go to home page 2. Click log in button 3. Select continue with email 4. Enter email address and click next	**Test Data:** email: rania.weiss@stu.ibu.edu.ba	**Expected Result:** The user is prompted to register	**Actual Result:** The user is prompted to register	**Status:** PASS
Notes:				

```
@Test
@Order(1)
void loginNoAcc() {
    WebElement toPass = webDriver
            .findElement(By.xpath("/html/body/div[1]/div/div[4]/div/div/div/div[2]/div/form/div[1]/div[1]/button"));
    webDriver.findElement(By.name("email")).sendKeys("rania.weiss@stu.ibu.edu.ba");
    toPass.click();
    assertTrue(wait.until(ExpectedConditions.not(ExpectedConditions.visibilityOfElementLocated(
            By.xpath("/html/body/div[1]/div/div[4]/div/div/div/div[2]/div/form/div[1]/div[1]/p[2]/button")))));
}
```

3.1.3 Login with invalid data

Test Name: Test logging in with invalid data				
Description: Log in using valid credentials				
Pre-condition(s): Have existing account				
Test Steps: 1. Go to home page 2. Click log in button 3. Select continue with email 4. Enter email address and click next	**Test Data:** email: this is not an email	**Expected Result:** An error saying that the entered data is invalid	**Actual Result:** An error saying that the entered data is invalid	**Status:** PASS
Notes:				

```java
@Test
@Order(2)
void loginBadMail() {
    WebElement toPass = webDriver
            .findElement(By.xpath("/html/body/div[1]/div/div[4]/div/div/div/div[2]/div/form/div[1]/div[1]/button"));
    webDriver.findElement(By.name("email")).sendKeys("this is not an email");
    toPass.click();
    // error claiming the entered value is not a valid email
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[3]")));
}
```

3.2 Unregistered user

An unregistered user cannot do anything other than register, log in, and scroll 2000 pixels before being prompted to log in or register

3.2.1 Scrolling

Test Name: Test register/login popup				
Description: Check if user can scroll without registering/logging in				
Pre-condition(s):				
Test Steps: 1. Go to the home page 2. Scroll random number of pixels	**Test Data:** 2525, 1254, 6989, 100	**Expected Result:** When scrolling more than 2000 pixels a popup screen for registration/login shows; if less than 2000 pixels popup does not show	**Actual Result:** When scrolling more than 2000 pixels a popup screen for registration/login shows; if less than 2000 pixels popup does not show	**Status:** PASS
Notes:				

```java
class PopUpLogin {
    public static WebDriver webDriver;
    public static String baseUrl;
    public static JavascriptExecutor js;
    public static String frogXpath = "/html/body/div[1]/div/div[4]/div/div[2]/div";
    public static WebDriverWait wait;

    @BeforeAll
    static void setUpBeforeClass() throws Exception {
        System.setProperty("webdriver.chrome.driver", "/home/rani/V Semester/SVVT/chromedriver_linux64/chromedriver");
        ChromeOptions options = new ChromeOptions();
        options.addArguments("--no-sandbox");
        options.addArguments("--incognito");
        options.addArguments("--start-maximized");
        webDriver = new ChromeDriver(options);
        baseUrl = "https://www.tumblr.com/";
        js = (JavascriptExecutor) webDriver;
        wait = new WebDriverWait(webDriver, Duration.ofSeconds(5));
    }

    @AfterAll
    static void tearDownAfterClass() throws Exception {
        webDriver.quit();
    }

    @BeforeEach
    void setUp() throws Exception {
    }

    @AfterEach
    void tearDown() throws Exception {
    }

    @ParameterizedTest
    @ValueSource(ints = { 2525, 1254, 6989, 100 })
    void testFrog(int number) throws InterruptedException {
        webDriver.get(baseUrl);
        js.executeScript("window.scrollBy(0, " + number + " )", "");
        if (number > 2000) {
            wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(frogXpath)));
        } else if (number <= 2000) {
            wait.until(ExpectedConditions.not(ExpectedConditions.visibilityOfElementLocated(By.xpath(frogXpath))));

        }
    }

```

3.3 Posts

A user can create all kinds of original posts, but in this suite creating text posts and editing their text was only tested. The most popular method of adding content to a user's blog is by reblogging (reposting with information on where it was reposted from, and who the original poster was). When reblogging a user can add their own content

3.3.1 Creating text post

Test Name: Test creating text post				
Description: Create a text post and use several text editing options provided, as well as adding tags				
Pre-condition(s): User is logged in				
Test Steps: 1. Go to the home page 2. Click button to create post 3. Select text post 4. Enter text 5. Enter tags	**Test Data:** title: A title bold: This text is bold italic: This text should be italic tags: absolutely anything can go	**Expected Result:** The created post is found on the user's blog	**Actual Result:** The created post is found on the user's blog	**Status:** PASS

6. Click button to post 7. Go to blog of user 8. Check whether post is on blog 9. Check if text is edited	in a tag			

Notes: To check if text is not plain text, but rather title, bold, and italic, elements are found by tag names

```
@Test
void postText() throws InterruptedException {
    webDriver.get(baseUrl);
    // button for creating posts
    webDriver.findElement(By.xpath("/html/body/div/div/div[2]/div[1]/header/div[2]/a")).click();
    String title = "A title";
    String bold = "This text is bold";
    String italic = "This text should be italic";
    String tags = "absolutely anything can go in a tag";
    Actions builder = new Actions(webDriver);

    // waiting for popup menu
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.className("HT4SV")));
    // creating text post
    webDriver.findElement(By.xpath("/html/body/div[1]/div/div[4]/div/div/div[1]/div/a")).click();
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.className("RuIGO")));
    Thread.sleep(2000);

    // typing the title and text post
    Action type = builder.sendKeys(Keys.ARROW_UP).sendKeys(title).sendKeys(Keys.ARROW_DOWN).keyDown(Keys.CONTROL)
            .sendKeys("b").keyUp(Keys.CONTROL).sendKeys(bold).keyDown(Keys.CONTROL).sendKeys("b")
            .keyUp(Keys.CONTROL).sendKeys(Keys.SPACE).keyDown(Keys.CONTROL).sendKeys("i").keyUp(Keys.CONTROL)
            .sendKeys(italic).build();
    type.perform();

    // add tags
    webDriver.findElement(By.className("mbROR")).sendKeys(tags);
    // post the text post
    webDriver.findElement(By
            .xpath("/html/body/div[1]/div/div[4]/div/div/div/div/div[2]/div[2]/div/div[3]/div/div/div/button/span"))
            .click();

    // checking whether the post can be found on the blog, and if all the text is
    // edited how it should be
    webDriver.get("https://www.tumblr.com/reallycoolblogsblog");
    assertEquals(title, webDriver.findElement(By.xpath(
            "/html/body/div/div/div[2]/div[2]/div/div/div[1]/main/div/div/div/div[2]/div[1]/div/div/article/div[1]/div/span/div/div[1]/h1"))
            .getText());
    assertEquals(bold, webDriver.findElement(By.tagName("strong")).getText());
    assertEquals(italic, webDriver.findElement(By.tagName("em")).getText());
}
```

3.3.2 Reblog with content

Test Name: Test reblogging posts from other blogs				
Description: Checking if reblogged posts will show up on user's blog				
Pre-condition(s): User is logged in				
Test Steps: 1. Go to post page 2. Add custom text to reblog 3. Add a tag 4. Click reblog button 5. Go to user's blog 6. Check if the original post, the added text, and tags appear in user's post	**Test Data:** firstBlog: https://www.tumblr.com/one-time-i-dreamt secondBlog: https://www.tumblr.com/reallycoolblogsblog	**Expected Result:** The post reblogged is found on user's blog, with text and tags they added	**Actual Result:** The post reblogged is found on user's blog, with text and tags they added	**Status:** PASS
Notes:				

```
51    @Test
52    void reblogWithContent() throws InterruptedException {
53        webDriver.get(firstBlog);
54        String reblogText = "I'm adding a reblog";
55        String tagText = "Gotta add a tag";
56        // find reblog button and click on it
57        webDriver.findElement(By.xpath(
58                "/html/body/div/div/div[2]/div[2]/div/div/div[1]/main/div/div/div/div[2]/div[1]/div/div/article/div[3]/footer/div[1]/div[2]/div[3]/span/span/span/span/a"))
59                .click();
60        wait.until(ExpectedConditions.visibilityOfElementLocated(By.className("RuIGO")));
61
62        // get original text
63        String originalText = webDriver.findElement(By.xpath(
64                "/html/body/div[1]/div/div[4]/div/div/div/div/div[2]/div[2]/div/div[1]/div[2]/div/div/div[2]/div/div/p"))
65                .getText();
66        // add text and tags
67        webDriver.findElement(By.xpath(
68                "/html/body/div[1]/div/div[4]/div/div/div/div/div[2]/div[2]/div/div[1]/div[3]/div/div[5]/div/div/div[1]/p"))
69                .sendKeys(reblogText);
70        webDriver
71                .findElement(By.xpath(
72                        "/html/body/div[1]/div/div[4]/div/div/div/div/div[2]/div[2]/div/div[2]/div/span/span/textarea"))
73                .sendKeys(tagText);
74
75        // confirm reblog
76        webDriver
77                .findElement(By.xpath(
78                        "/html/body/div[1]/div/div[4]/div/div/div/div/div[2]/div[2]/div/div[3]/div/div/div/button"))
79                .click();
80
81        // check on user's blog if the post was reblogged properly
82        webDriver.get(secondBlog);
83        Thread.sleep(3000);
84        assertEquals(originalText, webDriver.findElement(By.xpath(
85                "/html/body/div/div/div[2]/div[2]/div/div/div[1]/main/div/div/div/div[2]/div[1]/div/div/article/div[1]/div/span/div[1]/div[2]/div/div/p"))
86                .getText());
87        assertEquals(reblogText, webDriver.findElement(By.xpath(
88                "/html/body/div/div/div[2]/div[2]/div/div/div[1]/main/div/div/div/div[2]/div[1]/div/div/article/div[1]/div/span/div[2]/div[2]/div/div/p"))
89                .getText());
90        assertEquals("#".concat(tagText), webDriver.findElement(By.xpath(
91                "/html/body/div/div/div[2]/div[2]/div/div/div[1]/main/div/div/div/div[2]/div[1]/div/div/article/div[2]/div/div/a"))
92                .getText());
93    }
```

3.4 Notes

Most commonly used tumblr features are liking and reblogging other posts, which all fall under the category of notes. Notes include: likes, reblogs, and replies. Since reblogs and likes are much more commonly used, those features will be tested on one post in two scenarios – a post on a custom URL (where the owner of the blog has edited the HTML to their liking), and a post on a regular blog. Posts should be the same on both custom URLs and regular blogs, and so should their features.

3.4.1 Reblog from regular blog

Test Name: Test reblogging posts from other blogs				
Description: Checking if number of posts will change on account menu				
Pre-condition(s): User is logged in				
Test Steps: 1. Go to page of post on blog 2. Click reblog and wait for reblog window to open 3. Confirm reblog 4. Check whether the number of posts changed	**Test Data:** baseUrl: https://www.tumblr.com/doggosource/704577297320165376/doggosource-a-n-g-e-l	**Expected Result:** The number of posts is changed	**Actual Result:** The number of posts is changed	**Status:** PASS
Notes:				

```
@BeforeEach
void setUp() throws Exception {
    webDriver.get(baseUrl);
```

```
51    @Test
52    void reblogBlog() {
53        String postNumberBefore;
54        String postNumberAfter;
55        // click on menu that shows number of posts and likes
56        WebElement menuButton = webDriver
57                .findElement(By.xpath("/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/span/button"));
58        menuButton.click();
59        // the number of posts
60        WebElement postNumber = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(
61                "/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/div/div/div/ul[2]/div/li/div/ul/li[1]/a/span[2]")));
62        postNumberBefore = postNumber.getText();
63
64        // click reblog button
65        webDriver.findElement(By.xpath(
66                "/html/body/div/div/div[2]/div[2]/div/div/div[1]/main/div/div/div/div[2]/div/div/div/article/div[2]/footer/div[1]/div[2]/div[3]/span/span/span/span/a"))
67                .click();
68        // waiting until the reblog modal shows
69        wait.until(ExpectedConditions.visibilityOfElementLocated(By.className("RuIGO")));
70
71        // confirm simple reblog
72        webDriver
73                .findElement(By.xpath(
74                        "/html/body/div[1]/div/div[4]/div/div/div/div/div[2]/div[2]/div/div[3]/div/div/div/button"))
75                .click();
76        //check the menu for number of posts after reblog
77        wait.until(ExpectedConditions.visibilityOf(menuButton));
78        menuButton.click();
79        WebElement postNumberA = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(
80                "/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/div/div/div/ul[2]/div/li/div/ul/li[1]/a/span[2]")));
81        postNumberAfter = postNumberA.getText();
82        assertNotSame(postNumberBefore, postNumberAfter);
83    }
```

3.4.2 Like on regular blog

Test Name: Test liking posts from other blogs				
Description: Checking if number of liked posts changes on account menu				
Pre-condition(s): User is logged in				
Test Steps: 1. Go to page of post on blog 2. Click like 3. Check whether the number of liked posts changed	**Test Data:** baseUrl: https://www.tumblr.com/doggosource/704577297320165376/doggosource-a-n-g-e-l	**Expected Result:** The number of liked posts is changed	**Actual Result:** The number of liked posts is changed	**Status:** PASS
Notes:				

```
@BeforeEach
void setUp() throws Exception {
    webDriver.get(baseUrl);
```

```
@Test
void likeBlog() throws InterruptedException {
    String likeNumberBefore;
    String likeNumberAfter;
    // click on menu that shows number of posts and likes
    WebElement menuButton = webDriver
            .findElement(By.xpath("/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/span/button"));
    menuButton.click();
    // getting the number of likes
    WebElement likeNumber = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(
            "/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/div/div/div/ul[2]/div/li/div/ul/li[1]/a/span[2]")));
    likeNumberBefore = likeNumber.getText();

    // click like button
    webDriver.findElement(By.xpath(
            "/html/body/div/div/div[2]/div[2]/div/div/div[1]/main/div/div/div/div[2]/div/div/div/article/div[2]/footer/div[1]/div[2]/div[4]/span/span/span/span/button"))
            .click();
    //check the number of likes after liking post
    menuButton.click();
    likeNumberAfter = likeNumber.getText();
    assertNotSame(likeNumberBefore, likeNumberAfter);
}
```

3.4.3 Reblog from custom blog

Test Name: Test reblogging posts from customized blogs				
Description: Checking if number of posts changes on account menu				
Pre-condition(s): User is logged in				
Test Steps: 1. Go to page of post on blog 2. Click reblog and wait for reblog window to open 3. Confirm reblog 4. Check whether the number of posts changed	**Test Data:** baseUrl: https://doggosource.tumblr.com/post/704577297320165376/doggosource-a-n-g-e-l	**Expected Result:** The number of posts is changed	**Actual Result:** The user never sees the reblog window and the post is never reblogged – the number of posts stays the same	**Status:** FAIL
Notes: When checking this feature manually it works properly, but when using Selenium it fails				

```
@Test
void reblogCustomURL() throws InterruptedException {
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/div[3]/div/a[1]")));
    webDriver.findElement(By.xpath("/html/body/div[2]/div[3]/div/a[1]")).click();
}
```

3.4.4 Like on custom blog

Test Name: Test liking posts from customized blogs				
Description: Checking if number of liked posts changes on account menu				
Pre-condition(s): User is logged in				
Test Steps: 1. Go to page of post on blog 2. Click like 3. Check whether the number of liked posts changed	**Test Data:** baseUrl: https://doggosource.tumblr.com/post/704577297320165376/doggosource-a-n-g-e-l	**Expected Result:** The number of liked posts is changed	**Actual Result:** The post is not liked and the number of liked posts never changes	**Status:** FAIL
Notes: When checking this feature manually it works properly, but when using Selenium it fails				

```
@Test
void likeCustomUrl() throws InterruptedException {
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/div[3]/div/button[1]")));
    webDriver.findElement(By.xpath("/html/body/div[2]/div[3]/div/button[1]")).click();
}
```

3.5 Customization

While it is possible to alter the HTML of a blog, tumblr provides some simple customization options such as editing the title, description, and adding an avatar and header photos for a blog. Simply altering the title and description will be tested. Another feature that goes into customization is the different palettes (themes) the user will see their app in.

```
@BeforeEach
void setUp() throws Exception {
    webDriver.get(baseUrl);

}
```

3.5.1 Title

Test Name: Customize title of blog				
Description: Checking changing the name of the user's blog and whether the new changes can be found on the blog				
Pre-condition(s): User is logged in				
Test Steps: 1. Go to home page 2. Click account dropdown menu 3. Click edit appearance button 4. Click edit appearance button on blog 5. Click on the title 6. Change the title 7. Click save button 8. Go to user's blog 9. Check whether title on blog is the same as the new one	**Test Data:** title: qa	**Expected Result:** The user has changed their blog name successfully	**Actual Result:** The user has changed their blog name successfully	**Status:** PASS
Notes:				

```
@Test
void testTitle() {
    webDriver.findElement(By.xpath("/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/span/button")).click();
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/div/div/div/ul[2]/div/li/div/ul/li[8]/a")));
    webDriver.findElement(By.xpath("/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/div/div/div/ul[2]/div/li/div/ul/li[8]/a")).click();
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div/div/div[2]/div[2]/div[1]/main/div/div[1]/div[2]/button")));
    webDriver.findElement(By.xpath("/html/body/div/div/div[2]/div[2]/div[1]/main/div/div[1]/div[2]/button")).click();
    webDriver.findElement(By.className("c1JQY")).clear();
    String newName = "qa";
    webDriver.findElement(By.className("c1JQY")).sendKeys(newName);

    //save
    webDriver.findElement(By.xpath("/html/body/div/div/div[2]/div[2]/div[1]/main/div/div[1]/div[2]/div[1]/button[2]")).click();
    webDriver.get("https://tumblr.com/reallycoolblogsblog");
    String blogTitle = webDriver.findElement(By.xpath("/html/body/div/div/div[2]/div[2]/div/div/div[1]/header/div/div/h1")).getText();
    assertEquals(blogTitle, newName);
}
```

3.5.2 Description

Test Name: Customize blog description				
Description: Checking changing the description of the user's blog and whether the new changes can be found on the blog				
Pre-condition(s): User is logged in				
Test Steps: 1. Go to home page 2. Click account dropdown menu 3. Click edit appearance button 4. Click edit appearance button on blog 5. Click on the description 6. Change the description 7. Click save button 8. Go to user's blog 9. Check whether description of blog is the same as the new one	**Test Data:** description: im not a bot im testing this for a school project	**Expected Result:** The user has changed their blog description successfully	**Actual Result:** The user has changed their blog description successfully	**Status:** PASS
Notes:				

```
@Test
void testDescription() {
    webDriver.findElement(By.xpath("/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/span/button")).click();
    webDriver.findElement(By.xpath("/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/div/div/div/ul[2]/div/li/div/ul/li[8]/a")).click();
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div/div/div[2]/div[2]/div[1]/main/div/div[1]/div[2]/button")));
    webDriver.findElement(By.xpath("/html/body/div/div/div[2]/div[2]/div[1]/main/div/div[1]/div[2]/button")).click();
    webDriver.findElement(By.className("BL18W")).clear();
    String newDesc = "im not a bot im testing this for a school project";
    webDriver.findElement(By.className("BL18W")).sendKeys(newDesc);

    //save
    webDriver.findElement(By.xpath("/html/body/div/div/div[2]/div[2]/div[1]/main/div/div[1]/div[2]/div[1]/button[2]")).click();
    webDriver.get("https://tumblr.com/reallycoolblogsblog");
    String blogDesc = webDriver.findElement(By.xpath("/html/body/div/div/div[2]/div[2]/div/div/div[1]/header/div/div/div[1]/div/div")).getText();
    assertEquals(blogDesc, newDesc);
```

3.5.3 Palettes

Test Name: Customize palette of dashboard				
Description: Going through all 12 palettes tumblr offers and checking if selected changes on reflected on the dashboard				
Pre-condition(s): User is logged in				
Test Steps: 1. Go to home page 2. Click account dropdown menu 3. Click change palette button 12 times (to cover all 12 palettes) 4. With each click, check whether the body of page changes through the HTML	**Test Data:** **CSV:**{"True Blue,palette--trueBlue", "Dark Mode,palette--darkMode", "Low-Contrast Classic,palette--lowContrastClassic", "Cement,palette--cement", "Cybernetic,palette--cybernetic", "Canary,palette--canary", "Ghost,palette--ghost", "Vampire,palette--vampire", "Pumpkin,palette--pumpkin", "Snow Bright,palette--snowBright", "Goth Rave,palette--gothRave", "Pride,palette--pride"}	**Expected Result:** The body of the page changes palettes a dozen times with respect to the palette the user chose	**Actual Result:** The body of the page changes a dozen times with respect to the palette the user chose	**Status:** PASS
Notes:				

```java
@ParameterizedTest
@CsvSource({ "True Blue,palette--trueBlue", "Dark Mode,palette--darkMode",
        "Low-Contrast Classic,palette--lowContrastClassic", "Cement,palette--cement",
        "Cybernetic,palette--cybernetic", "Canary,palette--canary", "Ghost,palette--ghost",
        "Vampire,palette--vampire", "Pumpkin,palette--pumpkin", "Snow Bright,palette--snowBright",
        "Goth Rave,palette--gothRave", "Pride,palette--pride" })
void palette(String given, String expected) throws InterruptedException {
    webDriver.findElement(By.xpath("/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/span/button"))
            .click();
    WebElement paletteButton = webDriver.findElement(
            By.xpath("/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/div/div/div/ul[1]/li[9]/button"));
    // has to be clicked at least once in order to see current palette
    paletteButton.click();
    WebElement body = webDriver.findElement(By.xpath("/html/body"));
    String s = webDriver.findElement(By.xpath(
            "/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/div/div/div/ul[1]/li[9]/button/span/div[2]/span[1]/span"))
            .getText();
    int i = 0;
    while (!s.equals(given) && i < 12) {
        paletteButton.click();
        s = webDriver.findElement(By.xpath(
                "/html/body/div/div/div[2]/div[1]/header/div[2]/div[7]/span/div/div/div/ul[1]/li[9]/button/span/div[2]/span[1]/span"))
                .getText();
        i++;
        Thread.sleep(2500);
    }
    assertEquals(expected, body.getAttribute("class"));

}
```

3.6 Follow

It is possible to follow blogs on tumblr so that they can show up on the user's dashboard. The simplest way to follow a blog is to search for their username (partial username or complete username) and find the searched blog in the result set. The usernames are used to find, tag blogs; usernames are also used for the custom URLs
I.e. username.tumblr.com

3.6.1 Follow

Test Name: Test following blogs				
Description: Checking how to find and follow blogs using search				
Pre-condition(s): User is logged in				
Test Steps: 1. Go to home page 2. Click search 3. Enter username 4. Click on searched blog 5. Find and click on Follow button 6. Check whether the newly followed blog is found on the Following page	**Test Data:** CSV file where the first value is the search parameter, and the second is the actual username: {"neil,neil-gaiman", "blockchain-official,blockchain-official" }	**Expected Result:** The searched blog is found on the Following page	**Actual Result:** The searched blog is found on the Following page	**Status:** PASS
Notes:				

```
@ParameterizedTest
@CsvSource({ "neil,neil-gaiman", "blockchain-official,blockchain-official" })
void follow(String search, String blogName) {
    webDriver.get(baseUrl);
    // start search
    webDriver.findElement(By.name("q")).sendKeys(search);
    // wait until the searched blog appears
    wait.until(ExpectedConditions.visibilityOfElementLocated(By
            .xpath("/html/body/div/div/div[2]/div[1]/header/div[1]/div[2]/span/div/div/div/div[2]/div[1]/div/a")));
    // click on the searched blog
    webDriver
            .findElement(By.xpath(
                    "/html/body/div/div/div[2]/div[1]/header/div[1]/div[2]/span/div/div/div/div[2]/div[1]/div/a"))
            .click();
    wait.until(ExpectedConditions.visibilityOfAllElementsLocatedBy(By.xpath(
            "/html/body/div/div/div[4]/div/div[2]/div[2]/div/div/div[1]/header/div/div/div[2]/button[2]/span")));
    // find follow button
    WebElement followButton = webDriver.findElement(By.xpath(
            "/html/body/div/div/div[4]/div/div[2]/div[2]/div/div/div[1]/header/div/div/div[2]/button[2]/span"));
    if (followButton.getText().equals("Follow")) {
        followButton.click();
    }
    // check whether newly followed blog appears on followed page
    webDriver.get(baseUrl + "/following");
    List<WebElement> followingList = webDriver.findElements(By.className("UulOO"));
    ArrayList<String> namesOfBlogs = new ArrayList<String>();
    for (int i = 0; i < followingList.size(); i++) {
        namesOfBlogs.add(followingList.get(i).getText());
    }
    assertTrue(namesOfBlogs.contains(blogName));
}
```

3.6.2 Unfollow

Test Name: Test unfollowing blogs				
Description: Checking how to unfollow blogs from the following page				
Pre-condition(s): User is logged in				
Test Steps: 1. Go to following page 2. Click unfollow next to any blog 3. Refresh and check whether that blog is found	**Test Data:** blogToUnfollow: blockchain-official	**Expected Result:** A single blog is unfollowed	**Actual Result:** A single blog is unfollowed	**Status:** PASS
Notes:				

```
@Test
void unfollow() throws InterruptedException {
    webDriver.get(baseUrl + "/following");
    String blogToUnfollow = "blockchain-official";
    WebElement followButton;
    List<WebElement> followingList = webDriver.findElements(By.className("UulOO"));
    for (WebElement i : followingList) {
        if (i.getText().equals(blogToUnfollow)) {
            i.click();
            wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(
                    "/html/body/div/div/div[4]/div/div[2]/div[2]/div/div/div[1]/header/div/div/div[2]/button[2]/span")));
            followButton = webDriver.findElement(By.xpath(
                    "/html/body/div/div/div[4]/div/div[2]/div[2]/div/div/div[1]/header/div/div/div[2]/button[2]/span"));
            if (followButton.getText().equals("Following")) {
                followButton.click();
            }
        }
    }
    webDriver.get(baseUrl + "/following");
    ArrayList<String> namesOfBlogs = new ArrayList<String>();
    for (int i = 0; i < followingList.size(); i++) {
        namesOfBlogs.add(followingList.get(i).getText());
    }
    assertFalse(namesOfBlogs.contains(blogToUnfollow));
}
```

4. Conclusion

4.1. Testing Summary

Testing Tool	Total Tests	Passed Tests	Failed Tests
Java, Selenium, JUnit	15	13	2

The tests which failed are regarding notes on blogs who have customized their blogs by changing the HTML of the page. Regardless of custom HTML, the features of reblogging and liking should still be properly available.

4.2. Final Thoughts

Tumblr has a lot of interesting and useful features, and it was difficult to pick which ones seem most important to test. Generally, users of the website complain that it does not function properly often, and I have run into a couple situations as well.
For example, changing the description of the user's blog was not possible since the element was not clickable. This happened during the beginning of the project and the issue seems to be resolved since the test has passed now, but it was still an important feature.
Customizing the blog is one of the key features of tumblr, but it becomes difficult, or impossible to interact with the content on these blogs.
Aside from these issues (where one was even resolved during the project), I was fairly surprised at how well the website works, considering the large complaints. This project could definitely be larger to cover the features tumblr provides.