Testavimo veiklų planavimas

Įvadas

Šis dokumentas aprašo testavimo planą mūsų kuriamai programėlei „Nom nom“. „Nom nom“ – programėlė *Android* operacinei sistemai, skirta dalintis aplankytų restoranų atsiliepimais su savo draugais. Programėlės funkcinius reikalavimus aprašo šie *user stories*:

1. As a new user, I want to be able to register for an account.
2. As a registered user, I want to log in with my username and password.
3. As a user I want to be able to see a map where I can zoom in, zoom out and scroll in all directions.
4. As a user, I want to place pins in the map that correspond with the restaurants/caffes/bars I have been to.
5. As a user I want to be able to see the pins my friends have created.
6. As a user I want to be able to delete the pins I have created.
7. As a user, I want to be able to create posts associated with my pins.
8. As a user, I want to be able to edit my previously posted posts.
9. As a user, I want to be able to delete my previously posted posts.
10. As a user, I want to be able to comment on posts.
11. As a user, I want to be able to edit my comment.
12. As a user, I want to be able to delete comments.
13. As a user, I want to receive a notification if someone comments on my post.
14. As a user, I want to add new friends by sending them a friend request.
15. As a user, I want to cancel a sent friend request.
16. As a user, I want to manage my incoming friend requests.
17. As a user, I want to unfriend other users.
18. As a user, I want to receive notifications when I get a new friend request.
19. As a user, I want to be able to browse all the posts posted by me and my friends.
20. As a user, I want to be able to filter the posts posted by me and my friends by criteria such as post author, date, location.
21. As a user, I want to be able to sort all the posts posted by me and my friends by criteria such as date and name of the restaurant.
22. As a user, I want to be able to turn on and off app notifications.

Nefunkcinius reikalavimus aprašantys *user stories*:

23. As a product owner, I want the NomNom app to be compatible with all Android operating systems that are officially supported.
24. As a user, I want the NomNom app to be light-weight on my phone resources.
25. As a user, I want the crash rate of the NomNom client to be guaranteed to be low.
26. As a large user group, we want to concurrently be able to use the NomNom app without any performance issues.
27. As a user, I want the crash rate of the NomNom server to be guaranteed to be low.
28. As a user, I want the NomNom server to automatically reboot after a crash.
29. As a user, I want my data to be encrypted by industry-standard methods.
30. As a user, I want to be able to connect to NomNom server 24/7.

31. As a product owner, I want the requests to app server be validated.
32. As a developer, I want the code to be written in a way that makes it easy to update existing and add new features.
33. As a developer, I want the code to be well documented and understandable.

Testavimo apimtis

Testavimas apims programėlės „Nom Nom“ funkcinius bei nefunkcinius reikalavimus, kuriuos aprašėme su *user stories*. Testavimo tikslas yra patikrinti, ar apibrėžti funkciniai bei nefunkciniai reikalavimai yra įgyvendinti tinkamai bei kokybiškai. Pastebėjus netikslumus testuojant, bus galimybė pataisyti ir taip gerinti programėlės kokybę.

Testavimo strategija

Programėlę testuosime trimis etapais.

Pirmasis etapas: *unit* testavimas. *Unit* testavimo metu yra testuojami atskiri programos elementai. Tai pasirinkome kaip pirmą etapą, kadangi tai padės daug ankščiau mums pastebėti pradėtos kurti programėlės netikslumus funkcionalume.

Antrasis etapas – *static* testavimas. Tai pasirinkome kaip antrą etapą, kadangi jis padės įvertinti mūsų parašyto kodo kokybę. Kadangi funkcionalumą jau būsime įvertinę ir žinosime, kad jis veikia teisingai, tai galėsime susikoncentruoti į kodo kokybės gerinimą.

Trečiasis etapas – *performance* testavimas. Tai pasirinkome kaip paskutinį etapą, kadangi pasiekus šį etapą programėlė bus pabaigta, taigi norėsime sužinoti, kaip jį veikia esant daugiau užklausų ar kitomis programėlės darbą apsunkinančiomis sąlygomis.

Pradinės sąlygos

Šios sąlygos turi būti įvykdytos prieš pradedant vykdyti testavimo veiklas:

1. Funkcijos, kurios bus testuojamos, turi būti įgyvendintos ir veikiančios.
2. Turi būti pasirinkta testavimo aplinka;
3. Yra aprašyti priėmimo kriterijai (*acceptance criteria*);

Testavimo prioritetai

Žemiau esančiame sąraše aprašytos testavimo veiklos yra surašytos pagal mažėjančią jų svarbą, t.y. pati pirma aprašyta veikla yra pati svarbiausia.

1. Funkcijų testavimas – visos įgyvendintos funkcijos turi veikti taip, kaip suplanuota;
2. Vartotojui yra lengva ir patogu naudotis programėle;
3. Saugomi duomenys yra tinkamai apsaugoti;
4. Programėlė veikia sklandžiai ir greitai;

Testavimo tikslai

1. Visi programėlės komponentai veikia tinkamai;
2. Kodas yra tvarkingas, aiškus, nėra nereikalingų kintamųjų ar funkcijų;
3. Kodas nėra lengvai pažeidžiamas;
4. Programėlė dirba stabiliai net ir esant didesniam vartotojų kiekiui;
5. Programėlė sparčiai atsako į užklausas;
6. Programėlė atnaujina darbą, jei jis nutraukiamas;

Testavimo technikos

1. Testų *scripts* – testai, su apibrėžtais įvedamais duomenimis bei kokio rezultato tikimasi;
2. Testų *scripts* su ribinėmis savybėmis – testai, kurių metu naudojamos ribinės savybės patikrinti, ar teisingai elgiamasi su labai dideliais ar labai mažais duomenimis;
3. Kodo peržiūra padės užtikrinti, kad kodas būtų kokybiškas;
4. *Static* kodo analizės įrankiai;
5. *Load* testavimas;
6. *Stress* testavimas;
7. *Response* laiko testavimas;

Rolės ir atsakomybės

Testavimo procese dalyvauja komanda, kurią sudaro:

1. Lukas Andrijauskas – kokybės užtikrinimo vadovas. Atsakingas už testavimo procesų planavimą ir įvykdymą.
2. Rugilė Petraitytė – testuotojas. Atlieka testavimo veiklas aprašytas testavimo plane.
3. Evita Šriupšaitė – testuotojas. Atlieka testavimo veiklas aprašytas testavimo plane.
4. Ieva Žukauskaitė – produkto vadovas. Užtikrina, kad testai atliekami teisingai iš vartotojo pusės.

Rezultatai

Po testavimo, šie dokumentai turi būti prieinami:

1. Testavimo planas – šis dokumentas su visais pakeitimais, kurie padaryti testavimo metu.
2. *Unit* testavimo ataskaita;
3. *Static* testavimo ataskaita;
4. *Performance* testavimo ataskaita.

Testavimo aplinka

Mūsų kuriama programinė įranga bus testuojama šiais „Android“ OS turinčiais telefonais:

- Samsung S20+, 128GB, 8GB RAM, Exynos 990 (7 nm+), Android 13, One UI 5.1;
- Google Pixel 4, 64GB, 6GB RAM, Qualcomm SM8150 Snapdragon 855 (7 nm), Android 13, Android 13 baked in UI;

Taip pat testuosime šiais kompiuteriais:

- Intel(R) Core(TM) i7-7700K 4.2GHz, 16GB RAM, 1TB HDD, 512GB SSD, Microsoft Windows 10 Professional;
- AMD Ryzen 7 4700U 2.00 GHz, 8 GB RAM, 512GB SSD, Microsoft Windows 11 Home;
- Intel(R) Core(TM) i5-1135G7 2.40GHz , 8GB RAM, 256GB SSD, 1T SSD, Microsoft Windows 11 Home;
- AMD Ryzen 5 7600 6-Core 3.8 GHz, 32GB RAM, WD_BLACK SN770 1T, Samsung SSD 980 PRO 1T, Microsofft Windows 11 Home.

Testų scenarijai

Įvadas

Toliau aprašyti testavimo scenarijai yra skirti testuoti kiekvieną programos funkciją.

Kiekvienas testo scenarijus yra sudarytas iš:

1. Aprašymas;
2. Pradinė sąlyga;
3. Galutinė sąlyga;
4. Parametrai (jei tokie yra naudojami);
5. Testavimo žingsniai;

Test Script 01.1 Login form validation testing

- **Description:**

  This test case tests the functionality of being able to log in.

- **Pre-conditions:**

  DB contains user: test/test@test.com/pwd1.

- **Post-conditions:**

  User logs in successfully.

- **Parameters:**

  Valid username: test

  Invalid username: testInvalid

  Valid password: pwd1

  Invalid password: pwd2

- **Steps to reproduce:**

	Action	Data	Expected result
1	Open app		App opens.
2	User enters data and clicks Log In	testInvalid/pwd2	General error "Invalid credentials" is thrown.
3	User enters data and clicks Log In	test/pwd2	General error "Invalid credentials" is thrown.
4	User enters data and clicks Log In	test/pwd1	User logged in successfully.

Test Script 01.2 Sign up form validation testing

- **Description:**

  This test case tests the functionality of being able to sign up for the app.

- **Pre-conditions:**

  DB does not contain user: test/test@test.com/pwd1;

  DB contains user: test2/test2@test.com/pwd2.

- **Post-conditions:**

  User signed up sucessfully.

- **Parameters:**

  Invalid username: test2

  Invalid email: test2@test.com

  Valid username: test

  Valid email: test@test.com

  Password: dummyPwd

- **Steps to reproduce:**

	Action	Data	Expected result
1	Open app		App opens.
2	User clicks Sign Up button		Sign Up form is opened.
3	User enters data and clicks submit	test2/ test@test.com/ dummyPwd	Error "This username is already taken" is thrown.
4	User enters data and clicks submit	test/ test2@test.com/ dummyPwd	Error "This email is already taken" is thrown.
5	User enters data and clicks submit	test2/ test2@test.com/ dummyPwd	Error "This username and email is already taken. Please try logging in." is thrown.
6	User enters data and clicks submit	test/ test@test.com/ pwd1	User has signed up successfully.

Test script: 02.1. Map Zoom Boundaries

- **Description:**

  This test case verifies the behavior of the map when attempting to zoom out beyond the defined boundaries.

- **Pre-conditions:**

The application is launched;

User is logged in.

- **Post-conditions:**

  The map should not exhibit unexpected behavior or errors when attempting to zoom out beyond the defined boundaries.

- **Steps to reproduce:**

	Action	Data	Expected result
1	Navigate to the page with the map component		The map component should be visible without any rendering issues.
2	Attempt to zoom out within the defined boundaries of the map		The map should respond to the zoom action and zoom-in or zoom-out of the map.
3	Attempt to zoom out beyond the defined boundaries of the map		The map should respond to the zoom-out action, but the level of zoom should be limited to reasonable boundaries.

Test script 02.2. Map Scroll Boundaries

- **Description:**

  This test case verifies the behavior of the map when attempting to scroll beyond the defined boundaries.

- **Pre-conditions:**

  The application is launched;

  User is logged in.

- **Post-conditions:**

  The map should not exhibit unexpected behavior or errors when attempting to scroll beyond the defined boundaries.

- **Steps to reproduce:**

	Action	Data	Expected result
1	Navigate to the page with the map component		The map component should be visible without any rendering issues.
2	Attempt to scroll within the defined boundaries of the map		The map should respond to the scroll action and move in the direction coresponding to the scroll action.
3	Attempt to scroll beyond the defined boundaries of the map		The map should respond to the scroll action, but the movement should be limited to reasonable boundaries.

Test script 02.3. View Pins Created by Friends

- **Description:**

  This test case verifies the functionality of seeing the pins created by friends on the map.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  The user has at least one friend added in the application;

  At least one pin has been created by a friend.

- **Post-conditions:**

  The pins created by friends should be visible on the map.

- **Steps to reproduce:**

	Action	Data	Expected result
1	Navigate to the page with the map component		Pins created by user's friends should be visible.

Test script 02.4. Deleting a pin

- **Description:**

This test case verifies the functionality of a user being able to delete a pin on the map.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  The user has created at least one pin on the map.

- **Post-conditions:**

  The selected pin should be successfully deleted from the database and should no longer be visible on the map.

- **Steps to reproduce:**

	Action	Data	Expected result
1	Navigate to the page with the map component		The map component should be visible without any rendering issues.
2	Click on a pin to open its details		The details related to the selected pin should be displayed.
3	Click on the "Delete Pin" option/button		The system should prompt the user for confirmation or proceed with the deletion.
4	The system should prompt the user for confirmation or proceed with the deletion		The selected pin should be deleted from the map, and the map should reflect the change.

Test script 02.5. Creating a pin

- **Description:**

  This test case checks whether the post to be created is valid.

- **Pre-conditions:**

  The application is launched;

  User is logged in.

- **Post-conditions:**

  The post information should be successfully saved in the database, the newly created pin should be visible on the map.

- **Parameters:**

  RestaurantName:

[0 characters] *empty*,
Normal Name 123,
`-=[]\;',./+<>?:"{}|~!@#$%^&*()_+,
Name with tabs \t \t,
Name with new line symbol \n,
[>60 characters] Very long name Very long name Very long name Very long name Very long name Very long name Very long name Very long name Very long name

TextReview:

[0 characters] *empty*,
Just a normal review about a normal restaurant,
`-=[]\;',./+<>?:"{}|~!@#$%^&*()_+,
Review with tabs \t \t,
Review with new line symbol \n,
[>1000 characters] Very long review Very long review Very long review Very long review Very long review Very long review Very long review Very long review Very long review Very long review Very long review Very long review Very long review Very long review

- **Steps to reproduce:**

	Action	Data	Expected result
1	Navigate to the page with the map component		The map component should be visible without any rendering issues.
2	Press on specific place of the map		Create a post pop-up must appear.
3	Input RestaurantName1		Message "Please provide restaurant name" appears, the "Create" button is not active.
4	Input RestaurantName4		Message "Restaurant name not valid" appears, the "Create" button is not active.
5	Input RestaurantName5		Message "Restaurant name not valid" appears, the "Create" button is not active.
6	Input RestaurantName6		Message "Character limit exceeded" appears, the "Create" button is not active.
7	Input RestaurantName3		"Create" button active.
8	Input RestaurantName2		"Create" button active.
9	Input TextReview6		Message "Character limit exceeded" appears, the "Create" button is not active.
10	Input TextReview1		"Create" button active.
11	Input TextReview2		"Create" button active.
12	Input TextReview3		"Create" button active.
13	Input TextReview4		"Create" button active.
14	Input TextReview5		"Create" button active.
15	Add photos that are collectively under 30MB		"Create" button active.
16	Add additional photos, so collectively they exceed 30MB		Message "Photos exceed 30MB limit" appears, the "Create" button is not active.
17	Remove excessive photos		"Create" button active.
18	Press "Create" button		Pin created, post info saved to DB.

Test script 02.6. Aborting a pin while creating

- **Description:**

This test case verifies the ability to abort pin and/or post creation.

- **Pre-conditions:**

  The application is launched;

  User is logged in.

- **Post-conditions:**

  The post information should be successfully saved in database.

- **Steps to reproduce:**

	Action	Data	Expected result
1	Navigate to the page with the map component		The map component should be visible without any rendering issues.
2	Press on a specific place of the map		"Create a post" pop-up must appear.
3	Press "Cancel"		Post information is not saved to DB, pin is not created on the map.

Test script 03.1. Receiving a notification about a new comment

- **Description:**

  This test case verifies the functionality of getting notificationsafter recieving a new comment.

- **Pre-conditions:**

  User is logged in;

  User has "friend3" in their friends list;

  User has at least one post.

- **Post-conditions:**

  An in-app and/or push notification was recieved depending on the user's preferances.

- **Steps to reproduce:**

	Action	Data	Expected result
1	User, who has push notifications enabled, gets a new comment on their post		User gets in-app and push notification "Friend3 has commented on your post".
2	User, who has push notifications disabled, gets a new comment on their post		User gets in-app but not push notification "Friend3 has commented on your post".

Test script 03.2. Photo uploading validation

- **Description:**

  This test case ensures the accurate and secure uploading of photos on the platform, validating the system's handling of various scenarios.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  Map interface is opened;

  TextReview and RestaurantNames input is correct;

  An empty place has been selected on the map;

  The post creating interface is open.

- **Post-conditions:**

  While creating the post it is possible to remove and add more photos, the platform allows multiple photo uploads, but only allowing the clusters to be up to a designated size, post info is saved to DB.

- **Steps to reproduce:**

	Action	Data	Expected result
1	Press "Upload photos" button		Pop-up with choice of photo repository appears.
2	Choose main gallery as photo repository		Opened phone gallery.
3	Select photos >30MB and upload		Message "Photos exceed 30MB limit" appears, the "Create" button is not active.
4	1-2 steps repeat		
5	Select photos <30MB and upload		"Create" button active.
6	Press "Delete selection" button		"Create" button active.
7	1-2 steps repeat		
8	Select photos <30MB and upload		"Create" button active.
9	Press "Create" button		Pin created, post info saved to DB.

Test script 03.3. Created post photo field editing

- **Description:**

  This test case ensures that users can successfully edit the photo field of a post they have created on the platform.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  Map interface is opened;

  User has at least one post created that has images.

- **Post-conditions:**

  User is able to successfully edit the photo field of a created post, the post now appears as modified, the post information has been updated in the DB.

- **Parameters:**

  Photos1: {photo1, photo2}[<30MB]

  Photos2: {photo1}[<30MB]

  Photos3: { }[<30MB]

  Photos4: {photo1, photo2, photo3, photo4}[<30MB]

Post1: {restaurantName = "Name", reviewText = "text", uploadedPhotos = Photos1}

- **Steps to reproduce:**

	Action	Data	Expected result
1	Open post browser		Post browser opens successfully.
2	Click on Post1 post		Post editor opens.
3	Press "Delete Selection" button		"Save" button is active uploadedPhotos = Photos3
4	Press "Upload Photos" button		Gallery opens.
5	Upload photo set Photos2		"Save" button active uploadedPhotos = Photos2
6	Press "Upload Photos" button		Gallery opens.
7	Upload photo set Photos4		"Save" button active uploadedPhotos = Photos4
8	Press "Save" button		DB: Post1.uploadedPhotos = Photos4.

Test script 03.4. Created post text fields editing

- **Description:**

  This test case ensures that users can successfully edit the text fields of a post they have created on the platform.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  User has created at least one post.

- **Post-conditions:**

  User has edited their post, the post now appears as modified, the post's informating has been updated in the DB.

- **Parameters:**

  Post1: {reviewText = "review", "restaurantName = "name", photosUploaded={photo1, photo2}[<30MB]}

  Post2: {reviewText = "review2", "restaurantName = "name2", photosUploaded={photo1, photo2}[<30MB]}

- **Steps to reproduce:**

	Action	Data	Expected result
1	Open post browser		Post browser opens successfully.
2	Click on Post1 post		Post editor opens.
3	Change Post1 to Post2		"Save" button active.
4	Press "Save" button		Post info on DB updated to Post2.

Test script 03.5. User cannot edit/delete someone else's post

- **Description:**

  This test case validates that users are unable to edit or delete posts created by other users on the platform.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  The user has at least one friend added in the application;

  At least one post has been created by a friend.

- **Post-conditions:**

  The system denies the user the ability to edit/delete posts made by their friends, and only allows them to view them in browse mode.

- **Parameters:**

  Post1: {restaurantName = "Name", reviewText = "text", creator = friend1}

- **Steps to reproduce:**

	Action	Data	Expected result
1	Open Post Browser		Post browser opens successfully.
2	Select Post1		Post opens in browse mode.

Test script 03.6. Deleting one's own post

- **Description:**

This test case verifies that users are able to successfully delete their own posts from the platform.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  User must have created at least 1 post.

- **Post-conditions:**

  The user is able to delete their own post, the post is permanently removed from the DB.

- **Parameters:**

  Post1: {creator=currentUser, restaurantName = "Name", reviewText = "text", uploadedPhotos = Photos1}

- **Steps to reproduce:**

	Action	Data	Expected result	Attachments
1	Open post browser		Post browser opens successfully.	
2	Select Post1		Opens Post1 in editing mode.	
3	Press "Delete" button		DB entry for Post1 deleted, associated pin removed from map, post deleted from post browser.	

Test script 03.7. Commenting on friend's post

- **Description:**

  This test case ensures that users can successfully add comments to posts made by their friends on the platform.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  The user has at least one friend added in the application;

  At least one post has been made by a friend.

- **Post-conditions:**

  User has succefully made a comment on their friend's post, the comment appears below post information in edit/browse mode, a new comment entry has appeared in the DB.

- **Parameters:**

Post1: {creator=friend1, restaurantName = "Name", reviewText = "text", uploadedPhotos = Photos1}

Comment: *empty*, normal_comment, [>1000 characters] very long comment

- **Steps to reproduce:**

	Action	Data	Expected result
1	Open Post Browser		Post browser opens successfully.
2	Select Post1		Post1 opens in view mode.
3	Press "Add Comment" button		New text field "New Comment" appears.
4	User types comment	Comment1	"Save comment" button is not active, the message "Comment should be non-empty" appears.
5	User types comment	Comment3	"Save comment" button is not active , the message "Comment exceeds the limit" appears.
6	User types comment	Comment2	"Save comment" button active.
7	Press "Save comment" button		DB: new comment entry: postId = Post1.id User: comment appears below post information in edit/browse mode.

Test script 03.8. Commenting on one's own post

- **Description:**

  This test case confirms that users can successfully add comments to their own posts on the platform.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  User has created at least one post.

- **Post-conditions:**

  User has commented on their own post, and the comment appears below post information in edit/browse mode, a new comment entry has appeared in the DB.

- **Parameters:**

  Post1: {creator=currentUser, restaurantName = "Name", reviewText = "text", uploadedPhotos = Photos1}

  Comment: *empty*, normal_comment, [>1000 characters] very long comment

- **Steps to reproduce:**

	Action	Data	Expected result
1	Open Post Browser		Post browser opens successfully.
2	Select Post1		Post1 opens in edit mode.
3	Press "Add Comment" button		New text field "New Comment" appears.
4	User types comment	Comment1	"Save comment" button is not active, a message "Comment should be non-empty" appears.
5	User types comment	Comment3	"Save comment" button is not active, a message "Comment exceeds the limit" appears.
6	User types comment	Comment2	"Save comment" button active.
7	Press "Save comment" button		DB: new comment entry: postId = Post1.id comment appears below post information in edit/browse mode.

Test script 03.9. User can delete all comments on one's post

- **Description:**

  This test case verifies that users can successfully delete all comments on a post they have created on the platform.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  The user has at least one friend added in the application;

  At least one comment has been left by a friend on one's post.

- **Post-conditions:**

User has deleted all comments on their own post, the comments have been deleted for the DB, the deleted comments are no longer visible.

- **Parameters:**

  Post: {creator=currentUser, restaurantName = "Name", reviewText = "text", uploadedPhotos = Photos1}

  Comment: {creator=friend1, postId = Post1, content = "text"}, {creator=friend2, postId = Post2, content ="text"}

- **Steps to reproduce:**

	Action	Data	Expected result
1	Open Post browser		Post browser opens successfully.
2	Select post	Post1	Post1 in edit mode.
3	Select comment	Comment1	Comment1 in browse mode.
4	Press "Delete Comment"		DB: entry deleted post in edit mode: comment not visible anymore.
5	Select comment	Comment2	Comment2 in browse mode.
6	Press "Delete Comment"		DB: entry deleted post in edit mode: comment not visible anymore.

Test script 03.10. Deleting one's own comment

- **Description:**

  This test case confirms that users can successfully delete their own comments on posts within the platform.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  The user has written at least one comment on a friend's post;

  The user has written at least one comment on their own post.

- **Post-conditions:**

User has deleted their own comment, the comment not visible anymore, the comment has been deleted from the DB.

- **Parameters:**

  Post: {creator=currentUser, restaurantName = "Name", reviewText = "text", uploadedPhotos = Photos1},

  {creator=friend1, restaurantName = "Name", reviewText = "text", uploadedPhotos = Photos1}

  Comment: {creator=currentUser, postId = Post1, content = "text"}, {creator=currentUser, postId = Post2, content = "text"}

- **Steps to reproduce:**

	Action	Data	Expected result
1	Open Post browser		Post browser opens successfully.
2	Select post	Post1	Post1 in edit mode.
3	Select comment	Comment1	Comment1 in edit mode.
4	Press "Delete Comment"		DB: entry deleted post in edit mode: comment not visible anymore.
5	Select post	Post2	Post2 in view mode.
6	Select comment	Comment2	Comment2 in edit mode.
7	Press "Delete Comment"		DB: entry deleted post in edit mode: comment not visible anymore.

Test script 03.11. Editing one's comment

- **Description:**

  This test case ensures that users can successfully edit their own comments on posts within the platform.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  The user has written at least one comment on a friend's post;

  The user has written at least one comment on their own post.

- **Post-conditions:**

  User has edited their own comment, the comment now appears as modified, the comment information has been updated in the DB.

- **Parameters:**

  Post: {creator=currentUser, restaurantName = "Name", reviewText = "text", uploadedPhotos = Photos1},

  {creator=friend1, restaurantName = "Name", reviewText = "text", uploadedPhotos = Photos1}

  Comment: {creator=currentUser, postId = Post1, content = "text"}, {creator=currentUser, postId = Post2,

  content = "text"}, {creator=currentUser, postId = Post1, content = "text2"}, {creator=currentUser, postId = Post2, content = "text2"}

- **Steps to reproduce:**

	Action	Data	Expected result
1	Open Post browser		Post browser opens successfully.
2	Select post	Post1	Post1 in edit mode.
3	Select comment	Comment1	Comment1 in edit mode.
4	Edit comment	Comment3	Comment1 → Comment3.
5	Press "Save Comment"		DB: Comment3 post in edit mode: Comment3.
6	Select post	Post2	Post2 in view mode.
7	Select comment	Comment2	Comment2 in edit mode.
8	Edit comment	Comment4	Comment2 → Comment4.
9	Press "Save Comment"		DB: Comment4 post in view mode: Comment4.

Test script 03.12. Cannot edit friends' comments on not one's own posts

- **Description:**

  This test case verifies that users are unable to edit comments made by their friends on other user's posts.

- **Pre-conditions:**

The application is launched;

User is logged in;

The user has at least one friend added in the application;

At least one comment has been left by a friend on another friend's post.

- **Post-conditions:**

  The system denies the user the ability to edit comments made by their friends, and only allows them to view them in browse mode.

- **Parameters:**

  Post: {creator=friend1, restaurantName = "Name", reviewText = "text", uploadedPhotos = Photos1}

  Comment: {creator=friend2, postId = Post1, content = "text"}, {creator=friend1, postId = Post2, content ="text"}

- **Steps to reproduce:**

	Action	Data	Expected result
1	Open Post browser		Post browser opens successfully.
2	Select post	Post1	Post1 in view mode.
3	Select comment	Comment1	Comment1 in browse mode.
4	Select comment	Comment2	Comment2 in browse mode.

Test script 03.13. Cannot edit friends' comments

- **Description:**

  This test case verifies that users are unable to edit comments made by their friends on posts.

- **Pre-conditions:**

  The application is launched;

  User is logged in;

  The user has at least one friend added in the application;

  At least one comment has been left by a friend.

- **Post-conditions:**

  The system denies the user the ability to edit comments made by their friends, and only allows them to view them in browse mode.

- **Parameters:**

  Post: {creator=currentUser, restaurantName = "Name", reviewText = "text", uploadedPhotos = Photos1},

  {creator=friend1, restaurantName = "Name", reviewText = "text", uploadedPhotos = Photos1}

  Comment: {creator=friend1, postId = Post1, content = "text"}, {creator=friend1, postId = Post2, content ="text"}

- **Steps to reproduce:**

	Action	Data	Expected result
1	Open Post browser		Post browser opens successfully.
2	Select post	Post1	Post1 in edit mode.
3	Select comment	Comment1	Comment1 in browse mode.
4	Select post	Post2	Post2 in view mode.
5	Select comment	Comment2	Comment2 in browse mode.

Test script 04.01. Adding user as a new friend

- **Description:**
  This test case tests the functionality of adding a friend by using their username.
- **Pre-conditions:**
  DB contains user "friend1";
  DB doesn't contain "friend2";
  The application is launched;
  User is logged in;
  User doesn't have "friend1" in their friends list.
- **Post-conditions:**
  Friend request is sent to "friend1".
- **Steps to reproduce**

	Action	Data	Expected result
1	User navigates to the "Add a friend" page		"Add a friend" page opens successfully.
2	User types in validUser and clicks "add"	friend1	User gets a prompt about successfully inviting other user to be friends.
3	User types in invalidUser and clicks "add"	friend2	Error: this user does not exist.

Test script 04.02. Cancelling outgoing friend request

- **Description:**
  This test case tests the functionality of users being able to cancel an outgoing friend request.
- **Pre-conditions:**
  The application is launched;
  User is logged in;
  Friend request has been sent to user "friend1".
- **Post-conditions:**
  The outgoing friend request is canceled.
- **Steps to reproduce**

	Action	Data	Expected result
1	User navigates to the "Sent friend requests" page		"Sent friend requests" page opens successfully.
2	User locates the friend request that they want to cancel and clicks on "cancel"		User gets a confirmation prompt about cancelling a friend request.
3	User clicks "yes" on cancellation prompt		User has successfully cancelled the outgoing friend request. When going to friend1's profile it no longer shows that request was sent.
4	User clicks "no" on cancellation prompt		User hasn't cancelled outgoing friend request. When going to friend1 profile it shows that request was sent.

Test script 04.03. Accepting a friend request

- **Description:**
  This test case tests the functionality of a user's ability to accept a friend request.
- **Pre-conditions:**
  The application is launched;

User is logged in;
A friend request from "friend3" has been received.

- **Post-conditions:**
  User has a new friend - "friend3", they appear in user's friend list.
- **Steps to reproduce**

	Action	Data	Expected result
1	User navigates to the "Received friend requests" page		User opens up "Received friend requests" page successfully.
2	User locates the friend request they want to accept and clicks on "accept"		User sees an indicator about successfully accepting a friend request, the new friend is now in their friends list.

Test script 04.04. Denying a friend request

- **Description:**
  This test case tests the functionality of a user's ability to deny a friend request.
- **Pre-conditions:**
  The application is launched;
  User is logged in;
  A friend request from "friend3" has been received.
- **Post-conditions:**
  Friend request from "friend3" has been declined, there is no longer a friend request from "friend3".
- **Steps to reproduce**

Action	Data	Expected result
User navigates to the "Received friend requests" page		"Received friend requests" page opens successfully.
User locates the friend request that they want to decline and clicks on "decline"	Decline "friend3" friend request.	User sees an indicator about declining a friend request.

Test script 04.05. Blocking and denying a friend request

- **Description:**
  This test case tests the functionality of a user's ability to block other users and cancel their friend request at the same time.
- **Pre-conditions:**
  The application is launched;

User is logged in;
A friend request from "friend4" has been received.

- **Post-conditions:**
  User has blocked "friend4", the friend request from "friend4" is no longer there and there is no way to send a friend request from/to "friend4".
- **Steps to reproduce:**

	Action	Data	Expected result
1	User navigates to the "Received friend requests" page		"Received friend requests" page opens successfully.
2	User locates the user that they want to block and clicks "block" button		User gets a confirmation prompt about blocking "friend4".
3	User clicks "yes" on confirmation		User gets a prompt about successfully blocking "friend4", their friend request was automatically rejected and is no longer shown in "Received friend requests".

Test script 04.06. Successfully removing a friend

- **Description:**
  This test case tests the functionality of friend removal.
- **Pre-conditions:**
  The application is launched;
  User is logged in;
  User has user "friend5" in their friend list.
- **Post-conditions:**
  User has removed "friend5" from their friends list, "friend5" no longer appears on their friends list.
- **Steps to reproduce:**

	Action	Data	Expected result
1	User navigates to an "Friends list" page		"Friends list" page opens successfully.
2	User locates the user that they want to remove from their friends list and clicks the "x" button		User gets a confirmation prompt about removing "friend5".
3	User clicks "yes" on confirmation		User gets a prompt about successfully removing "friend5", when going to friend5's profile, it no longer shows that they are a friend and there is a button for sending a friend request.
4	User clicks "no" on confirmation		Prompt about confirmation is closed and user "friend5" is still in their friends list.

-

Test script 04.07. Receiving a notification about a friend request

- **Description:**
  This test case tests the functionality of getting notifications when receiving a friend request.
- **Pre-conditions:**
  User is logged in.
- **Post-conditions:**
  An in-app and/or push notification was recieved depending on the user's preferances.
- **Steps to reproduce:**

	Action	Data	Expected result
1	User, who has enabled push notifications, receives a new friend request		User gets in-app and push notification "Friend3 has sent a new friend request".
2	User, who has disabled push notifications, receives a new friend request		User gets in-app but not push notification "Friend3 has sent a new friend request".

Test script 05.1. Toggle notifications on own posts

- **Description:**
  This test case verifies the functionality of a user being able to turn on or off notifications on their own posts.
- **Pre-conditions:**
  The application is launched;
  User is logged in.

- **Post-conditions:**
  The user's notification settings about their posts should be updated according to their selection.
- **Steps to reproduce:**

	Action	Data	Expected result
1	Navigate to the user's account settings and click the option related to notifications		The notification settings page should be opened.
2	Search for a specific setting related to post notifications and toggle the setting to turn on/off notifications for own posts		The notification setting should be switched to the "on" or "off" position depending on the user's preferance.

Test script 05.2. Toggle notifications for new posts by friends

- **Description:**
  This test case verifies the functionality of a user being able to turn on or off notifications for new posts created by their friends.
- **Prie-conditions:**
  The application is launched;
  User is logged in.
- **Post-conditions:**
  The user's notification settings for new posts by friends should be updated according to their selection.
- **Steps to reproduce:**

	Action	Data	Expected result
1	Navigate to the user's account settings and click the option related to notifications		The notification settings page should be opened.
2	Search for a specific setting related to new posts by friends and toggle the setting to turn on/off notifications for new posts by friends		The notification setting should be switched to the "on" or "off" position depending on the user's preferance.

Test script 05.3. Toggle friend requests notifications

This test case verifies the functionality of a user being able to turn on or off notifications for friend requests.

- **Pre-conditions**
  The application is launched;
  User is logged in.
- **Post-conditions**
  The user's notification settings for friend requests should be updated according to their selection.
- **Steps to reproduce:**

	Action	Data	Expected result
1	Navigate to the user's account settings and click the option related to notifications		The notification settings page should be opened.
2	Search for a specific setting related to friend request notifications and toggle the setting to turn on/off notifications for friend requests		The notification setting should be switched to the "on" or "off" position depending on the user's preferance.

Test script 06.01. Sort posts alphabetically

- **Description:**
  This test case tests the functionality of sorting available posts by restaurant names in alphabetical order.
- **Pre-conditions**
  The application is launched;
  User is logged in;
  User must have some available posts (either posted by themselves or friends).
- **Post-conditions**
  Alphabetically sorted posts should be visible within the interface.
- **Steps to reproduce:**

	Action	Data	Expected result
1	User navigates to "Posts" page		Posts page opens successfully.
2	User clicks the "Sort" button		Sort menu opens successfully.
3	User selects "Sort alphabetically"		An indicator showing that this specific mode is active appears.
4	User selects the sorting order (ascending/ descending)		An indicator showing that this specific order has been selected appears.
5	User clicks "Accept" button		The sort interface is closed, posts are now displayed in alphabetical order in ascending or descending order.

Test script 06.02. Sort posts by date

- **Description:**
  This test case tests the functionality of sorting available posts by their post date.
- **Pre-conditions**
  The application is launched;
  User is logged in;
  User must have some available posts (either posted by themselves or friends).
- **Post-conditions**
  Posts sorted by date should be visible within the interface.
- **Steps to reproduce:**

	Action	Data	Expected result
1	User navigates to "Posts" page		Posts page opens successfully.
2	User clicks the "Sort" button		Sort menu opens successfully.
3	User selects "Sort by date"		An indicator showing that this specific mode is active appears.
4	User selects the sorting order (ascending/ descending)		An indicator showing that this specific order has been selected appears.
5	User clicks "Accept" button		The sort interface is closed, posts are now displayed by date in ascending or descending order.

Test script 06.03. Filter posts by specific user

- **Description:**
  This test case tests the functionality of filtering posts that were created by a specific user.
- **Pre-conditions:**
  The application is launched;
  User is logged in;
  User must have some available posts (either posted by themselves or friends).
- **Post-conditions:**
  Filtered posts should be visible within the interface.
- **Steps to reproduce:**

	Action	Data	Expected result
1	User navigates to "Posts" page		Posts page opens successfully.
2	User clicks the "Filter" button		Filter menu opens successfully.
3	User selects "Filter posts by user"		An indicator showing that this specific mode is active appears and a textbox with a username request shows up.
4	User types in a username		Textbox now contains a username.
5	User clicks "Filter"		The filter interface is closed, the only posts that are now displayed were created by the specified user.

Test script 06.04. Filter posts by specific restaurant

- **Description:**
  This test case tests the functionality of filtering posts by a specific restaurant.
- **Pre-conditions**
  The application is launched;
  User is logged in;
  User must have some available posts (either posted by themselves or friends).
- **Post-conditions**
  Filtered posts should be visible within the interface.
- **Steps to reproduce:**

	Action	Data	Expected result
1	User navigates to "Posts" page		Posts page opens successfully.
2	User clicks the "Filter" button		Filter menu opens successfully.
3	User selects "Filter posts by restaurant"		An indicator showing that this specific mode is active appears and a textbox with a restaurant name request shows up.
4	User types in a restaurant name		Textbox now contains a restaurant name.
5	User clicks "Filter"		The filter interface is closed, the only posts that are now displayed are of the specific restaurant.

Test script 06.05. Filter posts by creation time frame

- **Description:**
  This test case tests the functionality of filtering posts that were created in a specific time frame (e.g. day/week/month ago).
- **Pre-conditions**
  The application is launched; User is logged in;
  User must have some available posts (either posted by themselves or friends).
- **Post-conditions**
  Filtered posts should be visible within the interface.
- **Steps to reproduce:**

	Action	Data	Expected result
1	User navigates to "Posts" page		Posts page opens successfully.
2	User clicks the "Filter" button		Filter menu opens successfully.
3	User selects "Filter posts by date"		An indicator showing that this specific mode is active appears and a dropdown menu with specific time frames to choose from is displayed.
4	User types in/ selects a timeframe		Dropdown menu contains selected time frame.
5	User clicks "Filter"		The filter interface is closed, the only posts that are now displayed were created in the specified time frame.

Testų valdymas

Testai yra aprašyti Qase sistemoje. Ši sistema bus naudojama stebėti testavimo procesą bei registruoti defektus.

Testavimo tvarkaraštis

Testavimo užduotis	Pradžia	Terminas
Unit testing	2024-03-07	2024-04-04
Static testing	2024-04-04	2024-05-02
Performance testing	2024-05-02	2024-05-22

Testavimo rizikos

Galimos rizikos, kurios gali įtakoti testavimo procesą:

Rizika	Aprašymas	Sprendimas
Netinkamai atliekamas testavimas	Dėl žinių stokos, yra galimybė testus atlikti ne pagal standartus	Daugiau domėtis atliekamu testavimu bei konsultuotis su dėstytojais laboratorinių darbų metu ar kitais komandos nariais
Nepakankamai ištestuotas produktas	Produktas gali būti nepakankamai ištestuotas, kadangi skiriama nepakankamai tam laiko	Komandos viduje stengtis tinkamai planuoti laiką bei testavimą atlikti po truputį per visą suplanuotą laikotarpį.