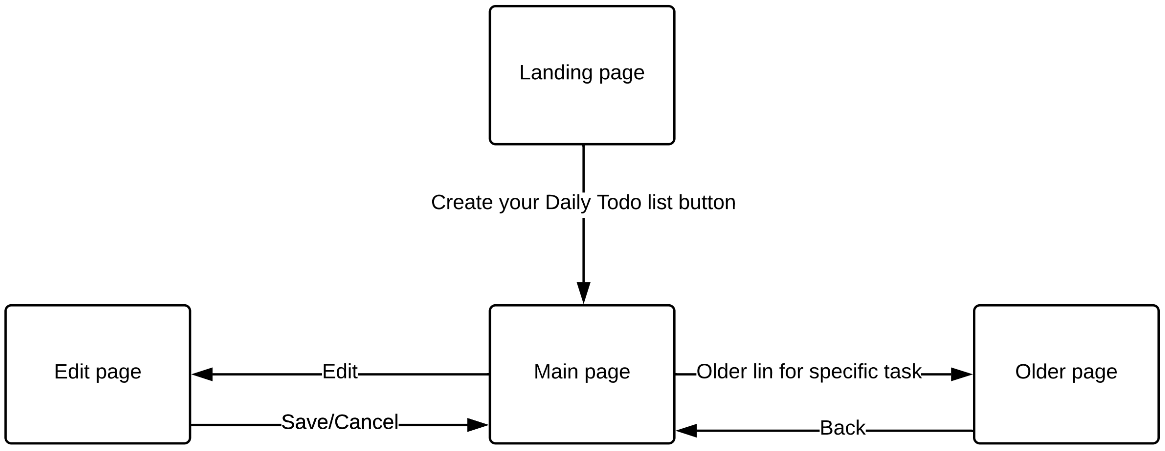
Daily To Do

This document contains basic information about the web application in question, along with the documentation for the two assigned tasks.

General information

The Daily To Do web app is a simple application with the basic functionality of tracking tasks over a time period (1 year total). The application is located on the following URL: https://dailytodo.org/ . The application has four simple pages and basic functionality. The pages are:

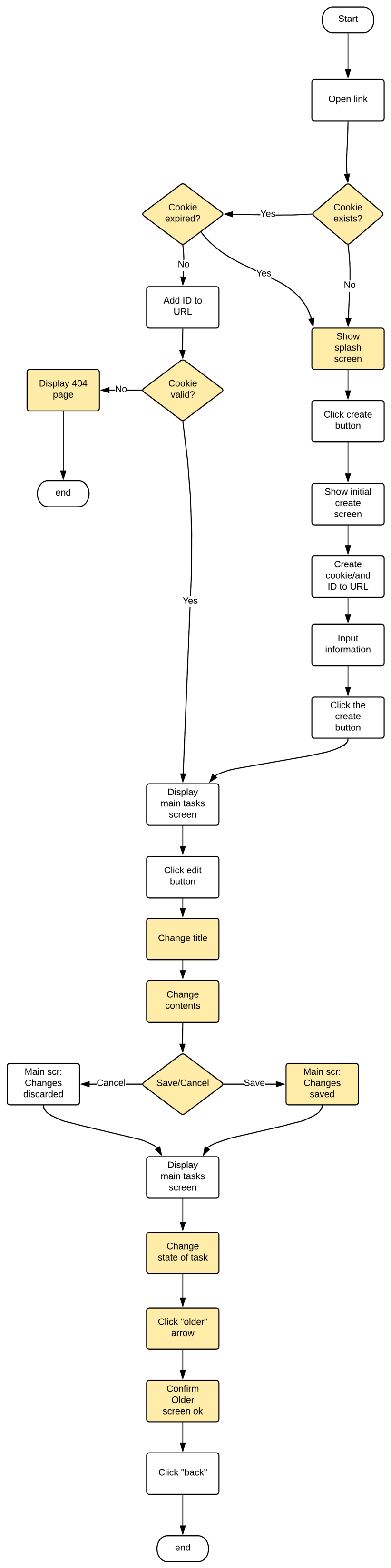
- Landing page - displaying basic information and call to action. For new users without the cookie or returning users with expired cookie
- Main page - manipulation of existing tasks and links to other pages
- Edit page - manipulation of the tasks and the title displayed on the main page
- Older page - provides a graphical representation of task completion for the last year (for the period thats tracked, following states of the elements:
    - Gray - was not tracked, no info
    - Green - task completed on the day
    - Red - task not completed on the day


Landing page
Create your Daily Todo list button
Edit page
Edit
Save/Cancel
Main page
Older lin for specific task
Back
Older page

There is no login as such and user tracking is done via cookie. The cookie contains basic information (the todo list id that is also set as a URL parameter, 2 year expiration time). The main page contains the last week of statuses for the tasks. The statuses can have the following states:

- Empty checkbox - today only
- Gray (no information) - tasks created after that date
- Green (with outline) - task completed for the specific date
- Green (without outline) - task status changed from red (refresh will add the outline)
- Red - task not completed for the specific date

The basic flow through the can be seen in the following image. The highlighted areas of the diagram contain areas that would need to be covered in detailed tests.


Start
Open link
Cookie exists?
Yes
Cookie expired?
No
Yes
No
Add ID to URL
Show splash screen
Cookie valid?
No
Display 404 page
end
Click create button
Show initial create screen
Create cookie/and ID to URL
Input information
Click the create button
Yes
Display main tasks screen
Click edit button
Change title
Change contents
Save/Cancel
Cancel
Save
Main scr: Changes discarded
Main scr: Changes saved
Display main tasks screen
Change state of task
Click "older" arrow
Confirm Older screen ok
Click "back"
end

Full resolution PDF file can be found here:
https://github.com/mmestric/toDoTestCases/blob/main/testFlow.pdf

Potential actions in the application

Main page:

- trigger a state change for a task
- access the edit page
- access the older page for a task

Edit page

- change the title
- change the content
- remove the title
- remove the content
- max length and max number of tasks (151 limit on task length and max 100 tasks)
- trigger a 500 error on the title (1501 chars, 500 server error, refreshing the page throws 405)
- confirm save functionality
- confirm cancel functionality

Older page

- confirm state changes history
- confirm state changes real time (change on main page and confirm)
- back working correctly

Cookie manipulation

- change cookie - trigger 401
- change URL parameter
- expired cookie - trigger splash screen

Tracking pixel manipulation (would require additional information and/or DB access to develop further)

Test cases (assignment 1)

The test cases present sample cases that would be used in manual testing with the basic information. This includes positive, negative (current example is not a true negative case as the issue is not handled) and edge cases. The test data is provided in the scenarios (in addition to the CSV file for the DDT principle example).

1. Create a task

(positive scenario)

Description:
A returning user should be able to successfully add a new task to the list

Precondition:
The user has already visited the site and has a valid cookie

Assumption:
No edge case content will be used

Steps:
1. Open https://dailytodo.org/
2. Click the edit button
3. Add a new row with the following information in the tasks text area: New test task
4. Click the "Save tasks" button

Expected result:
On clicking the save button the user is taken back to the main screen and a new task titled New test task is displayed on the bottom of the task list

2. Tasks title too large

(negative scenario)

Description:
A returning user should be able to set a long title and validation should trim the title and display it correctly

Precondition:
The user has already visited the site and has a valid cookie

Assumption:
Correct validation implemented, user should be able to input title value

Steps:

1. Open https://dailytodo.org/
2. Click the edit button
3. Enter the string from the test data in the title text input (1501 characters)
4. Click the "Save tasks" button

Expected result:
On clicking the save button the user is taken back to the main screen and a trimmed version of the title is displayed with the trailing 3 dots (...)

Actual result:
500 Internal server error page is displayed

Test data:
testCase2.csv
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum sed congue lorem. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla ac magna ut velit imperdiet bibendum. Fusce vel mauris sit amet dolor ullamcorper condimentum eu eget tortor. Phasellus auctor libero tincidunt, consequat ligula viverra, fringilla mauris. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Duis condimentum, neque eu venenatis varius, enim metus porta sapien, eu euismod lacus metus at felis. Aliquam porttitor vestibulum lectus, sit amet consectetur lectus vestibulum in. Mauris at ante id lacus sagittis iaculis. Donec facilisis venenatis risus. Curabitur eget elementum sapien, a mollis magna. Cras tempus nisi sit amet vehicula hendrerit. Vestibulum vel auctor dui. Cras lacinia orci porta sem fermentum convallis. Integer vitae rutrum urna, at tincidunt erat. Suspendisse laoreet ut nisi et ullamcorper. Mauris posuere scelerisque nisi quis luctus. Phasellus sed urna hendrerit, aliquet massa vel, semper tortor. Vivamus scelerisque placerat metus, eu dapibus massa. Maecenas cursus, odio id mollis condimentum, ipsum mi vehicula urna, et tincidunt metus nisl a justo. Donec fermentum sit amet dui at accumsan. Aliquam sollicitudin vitae orci at vehicula. Suspendisse lobortis faucibus lacus id lacinia. Aliquam dapibus ornare dignissim. Duis scelerisque pretium eros, in ultrices ligula consectetur sit amet.Proin1234567890

3. Task content size validation

(edge case)

Description:
A returning user should be able to set a long task text and validation should trim the task and display it correctly

Precondition:
The user has already visited the site and has a valid cookie

Assumption:
Correct validation implemented, user should be able to input task value in the text area

Steps:
1. Open https://dailytodo.org/
2. Click the edit button
3. Enter the string from the test data in the tasks text area input (151 characters)
4. Click the "Save tasks" button

Expected result:
On clicking the save button the user is taken back to the main screen and a trimmed version of the task is displayed with the trailing 3 dots (...), input text ends in 123, the 3 should be cut off and the task should end in "12..."

Test data:
testCase1_3_5.csv
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum sed congue lorem. Orci varius natoque penatibus et magnis dis parturient montes,123

4. Expired cookie

(edge case)

Description:
Returning to the page after cookie has expired should take the user to the splash screen

Precondition:
The user has already visited the site and has a valid cookie

Assumption:
Ability to access and modify cookies

Steps:
1. Open https://dailytodo.org/
2. Open the dev tools in the browser

3. Confirm that the cookie is still valid
4. Set the cookie expiration date in the past (e.g. 2020-02-06T13:47:09.166Z)
5. In a new tab access https://dailytodo.org/

Expected result:
When accessing the URL directly the expired cookie should disappear and the user should be taken to the splash screen

5. Max number of tasks

(edge case)

Description:
A returning user should be able to create a large number of tasks and validation should be able prevent the creation of additional ones

Precondition:
The user has already visited the site and has a valid cookie

Assumption:
Correct validation implemented on the task creation, user should be able to input task value in the text area

Steps:
1. Open https://dailytodo.org/
2. Click the edit button
3. Enter (Copy) the list of tasks from the test data in the tasks text area input (101 task rows)
4. Click the "Save tasks" button

Expected result:
On clicking the save button the user is taken back to the main screen and they are able to see 100 task rows. The test data contains 101 rows so the last row should not be displayed

Actual result (Note):
100 task rows are displayed in alternating between white and #eee but at row 41 they all start having the white background

Test data:

testCase1_3_5.csv

task	1
task	2
task	3
task	4
task	5
task	6
task	7
task	8
task	9
task	10
task	11
task	12
task	13
task	14
task	15
task	16
task	17
task	18
task	19
task	20
task	21
task	22
task	23
task	24
task	25
task	26
task	27
task	28
task	29
task	30
task	31
task	32
task	33
task	34
task	35
task	36
task	37
task	38
task	39
task	40
task	41
task	42
task	43

task	44
task	45
task	46
task	47
task	48
task	49
task	50
task	51
task	52
task	53
task	54
task	55
task	56
task	57
task	58
task	59
task	60
task	61
task	62
task	63
task	64
task	65
task	66
task	67
task	68
task	69
task	70
task	71
task	72
task	73
task	74
task	75
task	76
task	77
task	78
task	79
task	80
task	81
task	82
task	83
task	84
task	85
task	86
task	87

task 88
task 89
task 90
task 91
task 92
task 93
task 94
task 95
task 96
task 97
task 98
task 99
task 100
task 101

Automated testing (assignment 2)

The automated tests were created using IntelliJ with Selenium. They run on a chrome driver and are used as a proof of concept for tests and demonstrates basic principles that would be created for the web application.

This includes elements from

- simple DOM traversal,
- Data Driven Testing principles (DDT)
- slightly more complicated targeting test to illustrate the issues with poorly written source software

All of the tests implemented have additional comments better describing their specific functionality and elements.

The standard practice is decoupling the data from the tests themselves as demonstrated in the titleChange test.

- Test class is located in **test/java/com/example/toDoSelTest**
- Targeting classes have been created in **main/java/com/example/toDoSelTest** and contain classes for each of the page tested
    - usually those classes would contain all the potential targets on the pages in question

The test data is in the project root (**test1Title.csv**) and it follows the basic principle of DDT. The data is stored in the format of {input},{expected result} pairs, with every row representing an additional test run.

Once completed the test will generate a basic allure result (allure-results) and basic test report with a screenshot in (build/reports/tests)

The following tests have been implemented:

titleChange

Basic test demonstrating the DDT principle, reads test data from a csv file, prepares the data and runs the test for each row. As currently set up the last run will fail since it will hit the unhandled error.
The test handles the issue in the second testing scenario (and fails due to the unhandled 500 server error).

addingTask

Basic traversal on the front end and setting a value set in a defined in constant, once finished the test runs basic cleanup to ensure repeatability.
Using the principles in the first test, this test could be easily updated to handle the edge cases regarding tasks (max 100 tasks, max length of a single task is 150 etc.)

checkboxTest

This test is just a short demo of potential issues with poorly written code. Due to lack of simple unique identifiers, the only way to target and confirm that the status was changed is to use dates to create a string to use in matching to target a specific element. Once located soft assert is used not to break the entire test execution.