Table of Contents

Introduction

This time the team worked together and followed our waterfall model and the reuse oriented development process. After each phase, we should have an outcome of that phase which is all in this document.

We went back to the Component Analysis phase once and made some changes to our initial plan. When we went into Development and Unit Testing, we had two meetings, the **first one** was our original plan to have a **manual test** for this feature, and the **second one** was planning unit testing with **Espresso**.

1. Requirement Specification

Meeting Running:

Members: All team members

Date: Mar 16th, 2020

Duration: 30 mins

Purpose: Understand the user requirements and the outcome of this meeting should be about a detailed specification of the user requirement.

Outcome: Detailed specifications of the user requirements.

Meeting Outcome:

Make a expand button to replace the erase button, which will open a sub button menu when it is clicked. The menu will contain some predefined buttons such as erase, refresh and go back. We are not going to implement the trigger events as the same as the ChromePie as the issue mentions, since it is outdated. We decided to use an expand button and let it be moveable to allow users to have a control that is fast and intuitive.

2. Component Analysis

Meeting Running:

Members: All team members

Date: Mar 17th, 2020

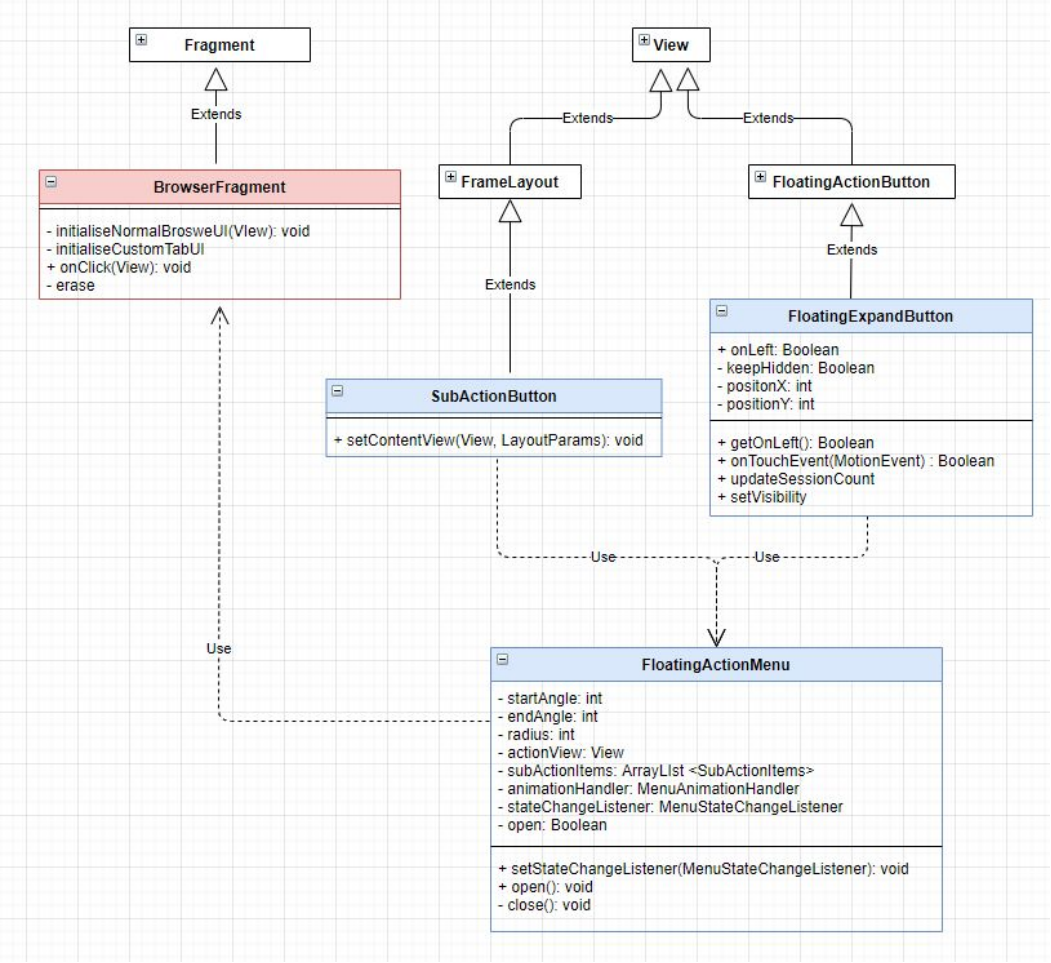
Duration: 1.5 hours

Purpose: Analyse the existing components in the program and the outcome of this meeting is the details about the component we need to modify.

Outcome: Details about the components and functions we need to modify.

Meeting Outcome:

This feature requires us to modify the existing **FloatingEraseButton**, since we need to change it to an expand button. Since we do not want to break the existing relationships between it and other components, we need to keep the functions inside the **FloatingEraseButton**. How to make the button movable was examined in deliverable 2's process documentation, so we do not need to worry about that here. Inside the **BrowserFragment**, we need to create a **FloatingActionMenu** activity listener and attach it to the expand button so that the button can open a menu of sub buttons and play animations to open and close the menu. We also need to create the sub buttons every time the app initializes the browser UI and attach their functionalities.



The UML for components we plan to make changes

Classes in white are the classes in android. We are not going to show the attributes and methods since they are too many.

Classes in blue are those we need to add into the project, the **FloatingExpandButton** is the one to replace the original FolatingEraseButton.

Class in red is the existing component in the project, we are not going to show all the existing methods of it since there are too many, so we just show the method we need to modify.

Existing components to be modified

widget/FloatingEraseButton (rename to **FloatingExpandButton**)[1]

We plan to use the original erase button as an “Expand” button. We need to change the icon’s UI and the function from erasing history to expanding the submenu

[1] before “/” is the package name, after “/” is the file name and “->” means the function needs to be modified inside the file

buttons. We will keep the changes we made in the previous deliverable so that the button still can be moveable. The new “Expand” button should be able to get its current location on the screen since it can either be on the left side of the screen or the right side of the screen, the sub buttons need to be expanded in the correct direction.

Fragment/BrowserFragment.kt -> **initialiseNormalBrowserUi**[1]

Create widgets such as the “Expand” button, sub buttons, and the expanded menu in BrowserFragment.

Fragment/BrowserFragment.kt -> **onClick**[1]

We need to add **setOnClickListener** to each sub button and override the **onClick** method so that each button can listen to users’ actions and do the corresponding functions such as delete history and refresh the page in the onClick method.

Fragment/BrowserFragment.kt -> **initialiseCustomTabUi**[1]

The widgets view should be hidden when the user is scrolling the page. This can be done by hiding its view.

Components need to be added

***SubActionButton.java**[2]

We need to add the items which expanded from the “Expand” button such as erase, back, and refresh buttons.

***FloatingActionMenu.java**[2]

The FloatingActionMenu should be attached to the “Expand” button and has multiple sub buttons when building the menu should also be able to set animation radius and angle for the expansion.

[2] “*” means the files need to be added into the project

*__FloatingActionMenu.java__ -> __MenuStateChangeListener__[2]

Set a listener for FloatingActionMenu when it is open or closed, we can add animation to the menu during the expansion and closure.

Overall, in order to make this feature, we need to modify some existing functions in the program, especially the UI initialization of the button. Also, we have to add some new functions to realize the menu of buttons and listeners to user actions.

3. System Design With Reuse

Meeting Running:

Members: All team members

Date: Mar 18th, 2020

Duration: 30 mins

Due Date For Tasks: Mar 27th, 2020

Purpose: This meeting is for designing tasks and assigning tasks to members to finish. The outcome of this meeting is tasks designed and we use Trello to keep track of each task.

Outcome: A solution based on component analysis

Meeting Outcome:

After analyzing the component, we decided to divide tasks as following:

Task 4.1: Replace the erase button to an expand button and add a FloatingActionMenu with the listener.

Task 4.2: Add the erase button to the sub button menu.

Task 4.3: Add the refresh button to the sub button menu.

Task 4.4: Add the go back button to the sub button menu.

Task 4.5: Make the menu expand to the right direction and angle based on the expand button current location on the screen. For example, if the button is on the bottom right of the screen, the menu should expand to the direction top left.

Task 4.6: Make the button only draggable when the menu is closed.

4. Development and Unit Testing

Meeting Running:

Members: All team members

Date: Mar 23th, 2020

Duration: 1 hour

Purpose: This meeting is for testing the feature we have added to the program, the outcome of this meeting should be a tested component.

Outcome: A tested and work component.

Test Steps: Since the feature requires a lot of frontend actions, we hardly can write unit tests for this whole feature, so we only write unit test cases for testing the expanded angle of the menu and manually test on Android simulators.

The following the is the steps we conduct the test:

Step 1: Launch the app in IntelliJ or Android Studio and keep a simulator running.

Step 2: In the URL toggle bar, type google.ca and enter.

Step 3: Move the button up and down on the right side of the screen, the button should still stick to the right side of the screen but the height of the button should change.

Step 4: Click the button and it should open the sub button menu.

Step 5: Click on refresh, the page should reload, but the menu should not be closed.

Step 6: Click on go back, it should direct to the default homepage and the menu closes.

Step 7: In the URL toggle bar, type youtube.com and go.

Step 8: Move the button to the left side of the screen.

Step 9: Move the button up and down on the left side of the screen, the button should still stick to the left side of the screen but the height of the button should change.

Step 10: Click on the button, it should open the menu.

Step 11: Click on the expand button again, it should close the menu.

Step 12: Open the menu again by clicking the button, and none of the buttons can be draggable.

Step 13: Click on the erase button, it should leave youtube.com to the homepage of Firefox-Focus (delete the browsing history as wanted).

Test No.	Test Cases	Operation	Expected Behavior	Actual Behavior	Pass/Failed
1	Test if the button will stick to the right side of the screen	User presses the button the drag it only in the right half area of the screen (change height)	The button should stick to the right side of the screen automatically	The button sticks to the right side of the screen automatically	Pass
2	Test if the button opens the menu in the direction of top left when it is in the right bottom area	User presses the button	The button should open the menu on the top left direction of the button	The button opens the menu on the top left direction of the button	Pass
3	Test if the button opens the menu in the direction of bottom left when it is in the right bottom area	User presses the button	The button should open the menu on the bottom left direction of the button	The button opens the menu on the bottom left direction of the button	Pass
4	Test if the button can be dragged to the left side of the screen from the right side of the screen	User presses the button and drag it to the left half area of the screen and release	The button should stick to the left side of the screen automatically	The button sticks to the left side of the screen automatically	Pass

5	Test if the button will stick to the left side of the screen	User presses the button the drag it only in the left half area of the screen (change height)	The button should stick to the left side of the screen automatically	The button sticks to the left side of the screen automatically	Pass
6	Test if the button opens the menu in the direction of top right when it is in the right bottom area	User presses the button	The button should open the menu on the top right direction of the button	The button opens the menu on the top right direction of the button	Pass
7	Test if the button opens the menu in the direction of bottom right when it is in the right bottom area	User presses the button	The button should open the menu on the bottom right direction of the button	The button opens the menu on the bottom right direction of the button	Pass
8	Test if the button can be dragged to the right side of the screen from the left side of the screen	User presses the button and drag it to the right half area of the screen and release	The button should stick to the right side of the screen automatically	The button sticks to the right side of the screen automatically	Pass

9	Test refresh button	User clicks the refresh button when the sub button menu is open	Current web page should reload and menu should not close	Current web page reloads and menu closes	Pass
10	Test go back button	User clicks the go back button when the sub button menu is open	It should direct to previous page that users visit and menu should not close	It directs to previous page that users visit and menu closes	Pass
11	Test erase button	User clicks the erase button when the sub button menu is open	Current browsing history should be cleaned and showing the default homepage of Firefox-Focus,the menu and the expand button should disappear	Current browsing history cleaned and the default homepage of Firefox-Focus is displayed, the menu and the expand button disappears	Pass
12	Test the display of the menu and the expand button when scrolling down	User opens the menu and scroll down the screen	All sub buttons and the expand button should disappear	All sub buttons and the expand button disappear	Pass

13	Test the display of the menu and the expand button when scrolling up	User scrolls up the screen	Only the expand button should be displayed	Only the expand button is displayed	Pass

Meeting Outcome:

The expand button works as specified and all the sub buttons operate their own functionalities as mentioned.

5. Component Analysis

Meeting Running:

Members: All team members

Date: Mar 24th, 2020

Duration: 30 mins

Purpose: Re-analyse the existing components in the program since we found that we cannot set the expand angle inside the action menu open() method (Task 4.5 cannot be achieved).

Outcome: Modified details about the components and functions we need to modify.

Meeting Outcome:

We directly set the expanded angle after the user finishes dragging it by calling **setStartAngle** and **setEndAngle** of the action menu (we need to add these two methods in the **FloatingActionMenu** first). And we do not need the **onLeft()** method to determine the current location of the expand button anymore.

Existing components modified

widget/FloatingEraseButton (rename to **FloatingExpandButton**)[3]

We plan to use the original erase button as an “Expand” button. We need to change the icon’s UI and the function from erasing history to expanding the sub buttons. We will keep the changes we made in the previous deliverable so that the button still can be moveable. Everytime when the “Expand” button location changes we update the start angle and end angle in the “Expand” menu so that the sub buttons will expand in the correct direction. The visibility of sub buttons should follow the visibility of the

[3] before “/” is the package name, after “/” is the file name and “->” means the functions modified inside the file

“Expand” button, so when we show or hide the “Expand” button, we also need to set visibility to each sub button.

Fragment/BrowserFragment.kt -> **initialiseNormalBrowserUi**[1]

Initialise widgets the “Expand” button, sub buttons, and the expanded menu in BrowserFragment. Replace UI of “Expand” button to “cross”, add sub buttons (erase, refresh, and back) to the “Expand” menu, and also attach “Expand” button to it as well. Implement the setStateChangeListener interface so that we can add animation during the menu opening and closing.

Fragment/BrowserFragment.kt -> **onClick**[1]

Override the **onClick** method so that each button can listen to users’ actions and do the corresponding functions based on the id of each sub button such as delete history, refresh the page, and go back to the last page in the onClick method.

Components be added

widget/*SubActionButton.java[4]

We need to add the items which expanded from the “Expand” button such as erase, back, and refresh buttons. It extends FrameLayout. We are implementing it using a builder design pattern so that it's easier for us to initialise.

widget/*FloatingActionMenu.java[2]

The “Expand” button should be attached to a FloatingActionMenu and it has multiple sub buttons. We also implement it using a builder design pattern so that it’s easier for us to initialise, add sub buttons, or attach the “Expand” button to it...

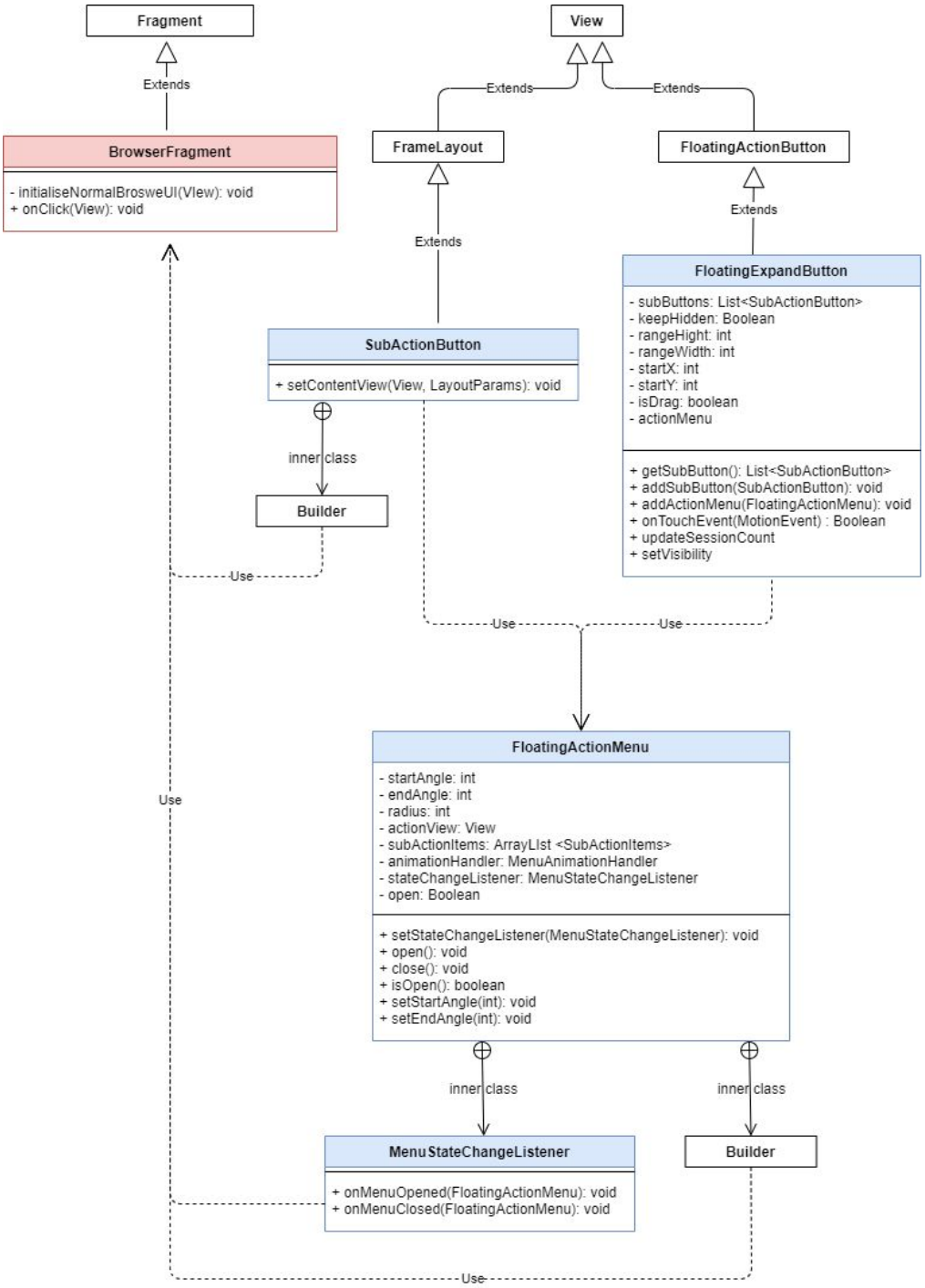
[4] “*” means the files added into the project

widget/*FloatingActionMenu.java -> **MenuStateChangeListener**[2]

We need an inner class to Set a listener for FloatingActionMenu when it is open or closed, we can add animation to the menu during the expansion and closure.

Fragment/BrowserFragment.kt -> **initialiseSubButton**[2]

Initialise sub Buttons UI and functionality. Add **setOnClickListener** and function id to each sub button. Add the sub buttons to the “Expand button”.



Classes in white are the classes in android or the builder class for construct objects. We are not going to show the attributes and methods since they are too many. Classes in blue are those we need to add into the project, the FloatingExpandButton is the one to replace the original FolatingEraseButton. Class in red is the existing component in the project, we are not going to show all the existing methods of it since there are too many, so we just show the method we need to add or modify.

6. System Design With Reuse

Meeting Running:

Members: All team members

Date: Mar 25th, 2020

Duration: 30 mins

Due Date For Tasks: Mar 27th, 2020

Purpose: This meeting is for adding new tasks based on the changes we made and bugs we found during implementation. Again, new tasks will be reflected on Trello.

Outcome: A solution based on component re-analysis.

Meeting Outcome:

After re-analyzing the component, we decided to divide tasks as following:

Task 4.7: Make unit test cases for testing the expanded angle of the menu.

Task 4.8: Add setStartingAngle and setEndAngle method in FloatingActionMenu.

Task 4.9: Fix the go back functionality. Currently, if the user goes to a new page from the default page, the go back button cannot render to the default homepage.

Task 4.10: Cleaning up the code, since we add some duplicate code when creating sub buttons.

7. Development and Unit Testing

Meeting Running:

Members: All team members

Date: Mar 27th, 2020

Duration: 30 mins

Purpose: This meeting is for designing the unit test cases with Espresso.

Outcome: We decided to test the expand angle for the menu of sub buttons and drag locations with Espresso. This test file will be uploaded to the group repository under the same folder with this process document.

8. Integration and System Validation

Meeting Running:

Members: All team members

Date: Apr 6th, 2020

Meeting: 1 hour

Purpose: Integration and validation of our solution

Outcome: Code should be ready to be merged.

Meeting Outcome:

The expand button works as specified and the functionality of cleaning browsing histories remains as well as refresh page and go back. Code is validated and ready to create a pull request.