Send A:
Project Deliverable 3

Team Members:
Anthony Le,
Birathan Somasundaram,
Pratana Atikhomkamalasai,
Suxin Hong,
Yuewen Ma

Table of Contents

Table of Contents **1**

Issues Considered **2**

Descriptions of Selected Issues 2

Issue #3873: Pie Contorls **3**

Description 3

Implementation Design 3

Existing components to be modified 3

Components need to be added 4

UML Diagram 5

Issue #3808: **6**

Description 6

Implementation Design 6

Existing components to be modified 6

Components need to be added 7

UML Diagram 8

Selected Issue **9**

Selected Issue to Implement 9

Testing **10**

Acceptance Testing 10

Architecture Revision **12**

Issues Considered

Descriptions of Selected Issues

During the team meeting we have, and we decided to look into the following issues of Firefox-Focus:

Issue Link	Description
Issue #3873	A pie control that allows the user to quickly open new pages, navigate between the opened tabs, move back & forward in history, etc, so that users can have a fast and intuitive control of this app.
Issue #3808	A search shortcuts in the URL search bar to allow users to search contents with their prefered search engine, such as Yahoo or Bing, by declaring “@” plus the search engine name before the content users want to search.

Issue #3873: Pie Contorls

Description

In Firefox Focus, users clean the browsing history by clicking the “Delete” button at the bottom right of the screen. Afterward, there is no option to click the “Refresh” button in the app. These two are quite important functionalities for a privacy browser. In order to present these two features together for a fast and intuitive control, the Issue #3873 suggests developers implement a pie control for users. Starting with a menu button, the button will spread out a few submenu buttons such as Delete and Refresh when clicked.

Implementation Design

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 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.

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

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.

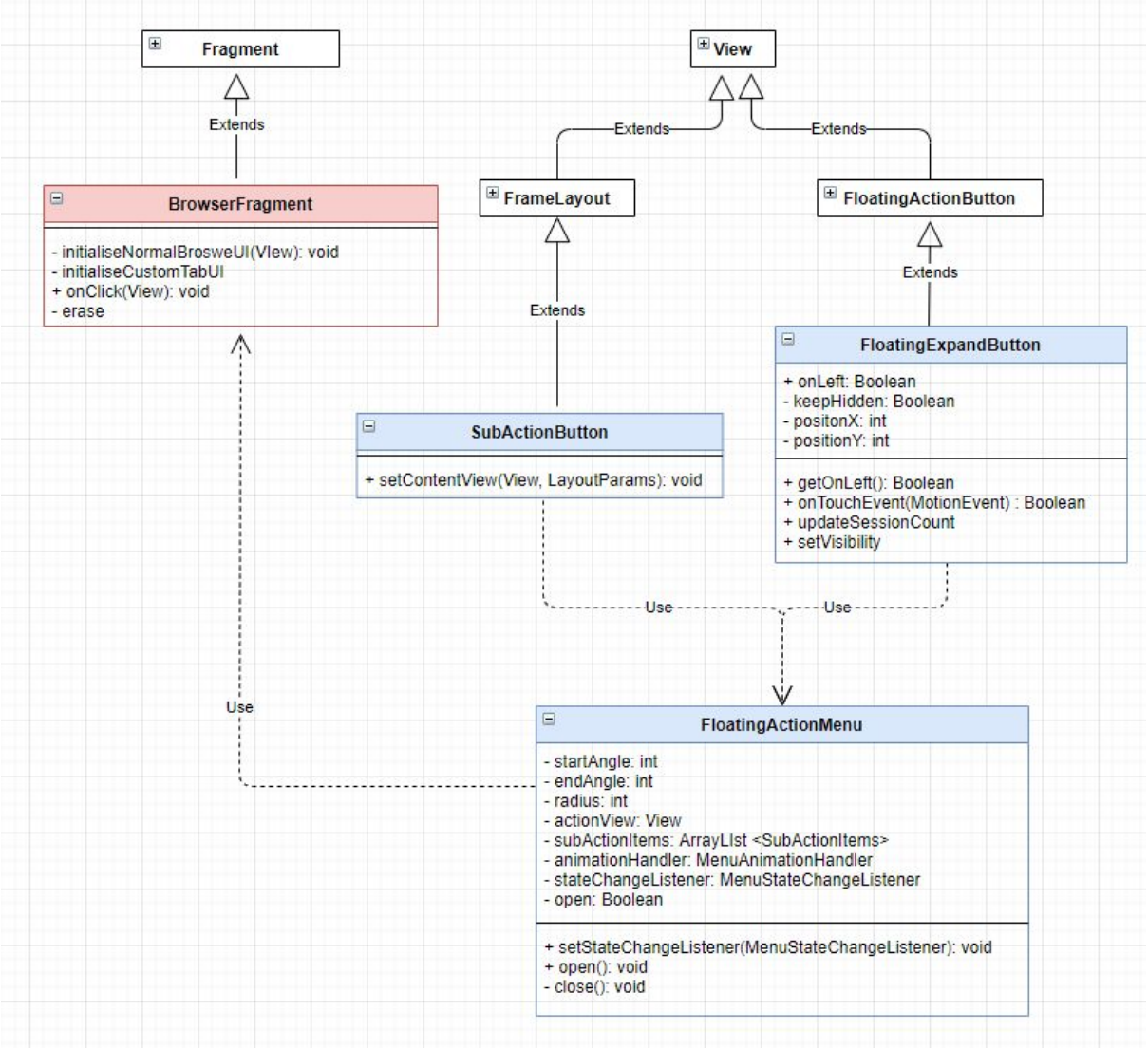
***FloatingActionMenu.java** -> **MenuStateChangeListener**[2]

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.

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

UML Diagram



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.

Issue #3808:

Description

In Firefox Focus, there is no option to use other popular search functions to narrow down specific types of searches and reach wanted pages quickly. Currently, users will have to go to the search engine site, type the search query and finally press search. A better and intuitive way would be to add an optional query parameter to select a search engine. An example of this is the use of Bangs from DuckDuckGo. Users would type to search “!amazon keyboard”, to use the amazon search engine and go straight to an amazon listing of keyboard products instead of going to the amazon website and typing “keyboard” in the search bar.

Implementation Design

Existing components to be modified

Everytime when a search query is entered, the onCommit() function of UrlInputFragment is called and the query is converted to a search url which is ten executed to display a page of results. This onCommit function uses another function called normalizeUrlAndSearchTerms to provide it with this search URL. We will need to modify this function to return the search url from the selected search engine

fragment/UrlInputFragment.kt/normalizeUrlAndSearchTerms

This function normalizes the search query before creating a search url based on the default search engine. It then returns the search url.

We plan to modify this function to include an option to use the new UrlAddressSearchEngineParser function (created below) to return the correct search url based on the input query. If a specific search engine is given in the query (using

the correct Bangs syntax for DuckDuckGo), then we will search using that engine and use the default engine otherwise.

Components need to be added

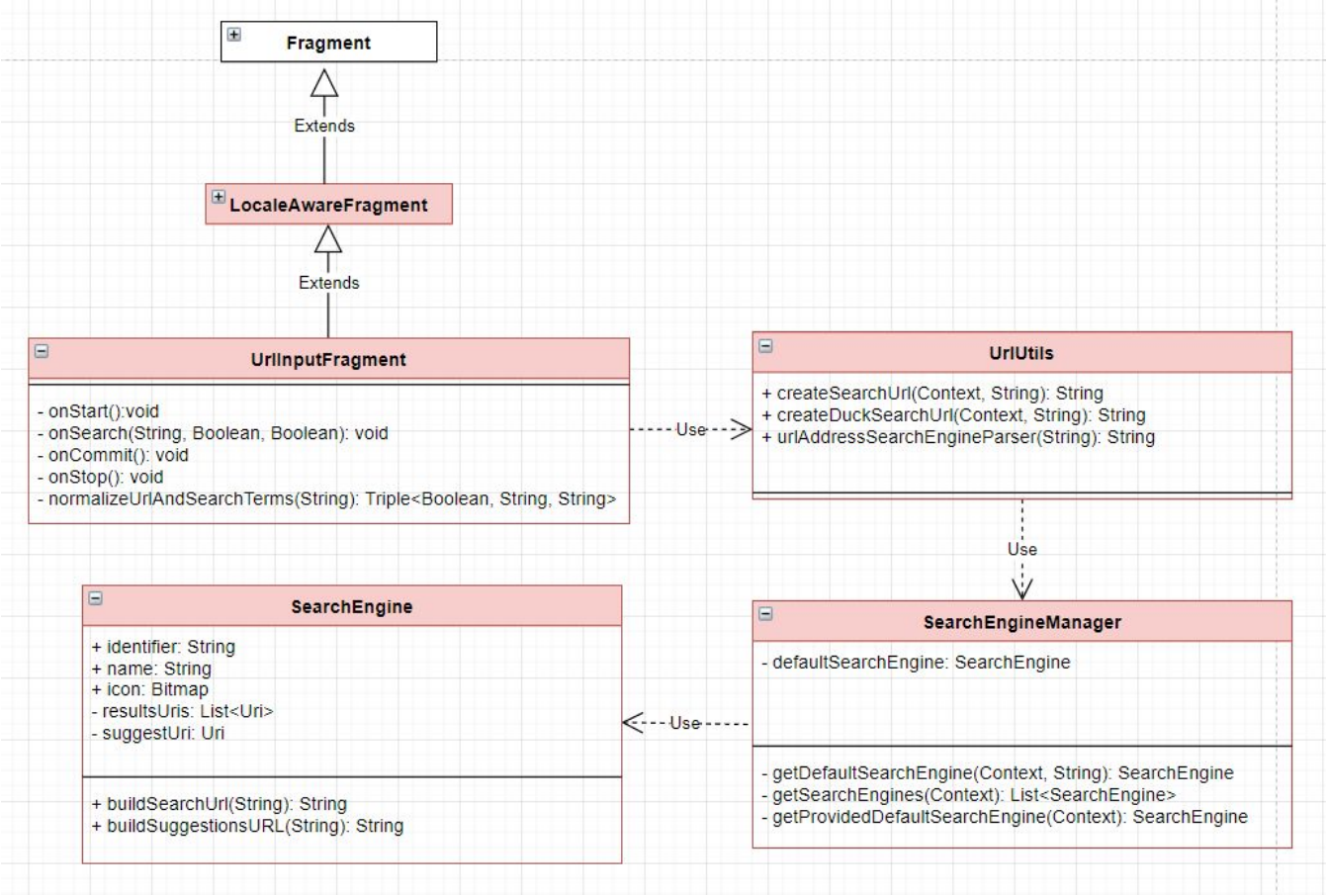
utils/UrlUtils.java -> UrlAddressSearchEngineParser

Parse the search string query to separate the search engine and the actual search query. This will need to be implemented with regex to cover all the cases, including the case where no search engine is given. We also need to think of the format for the optional query parameter. It is going to be hard to implement due to edge cases e.g. “@amazon Cats@ cute” , the parser should be able to identify amazon as a search engine and leave “Cats@ cute” as the search term.

utils/UrlUtils.java -> CreateDuckSearchUrl

This function will convert our query into a DuckDuckGo search url, instead of a search url based on the default searchEngine. This is to use the Bangs Feature from DuckDuckGo.

UML Diagram



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

Selected Issue

Selected Issue to Implement

As a team, we all agreed to work on the Issue **#3873**. The issue itself requires us to add an additional controlling panel to the app. This does not require us to know much about other components in the project. We only need to know how to use and modify the existing functions in the project such as **erase** (deleting the history), **refresh**(reload) and **goBack**(going backward).

Furthermore, the issue itself is very similar to Issue **#4477**, Make the Button movable. We were able to fix this bug successfully in deliverable 2. We can expand on this solution. In addition, The majority of UI component analysis has already been done to implement this fix. This would have saved us a lot of time in both analysing and designing a solution.

As for testing, we will be mainly working on the frontend. This would have made it difficult for unit testing. We would only be able to write unit testing for a change in functionality which the majority is already implemented in the project. As a result, we can only expand on these test suites and manual test the UI component. While this may be a disadvantage, it would have saved us time delivering the solution.

Testing

Acceptance Testing

Acceptance Citeria

- Clicking the expand button should expand the sub menu-button such as delete, refresh
- Each sub menu-button should work according to its corresponding function

Pre-requisites:

- Github
- Android Studio (java and sdk environment installed correctly)

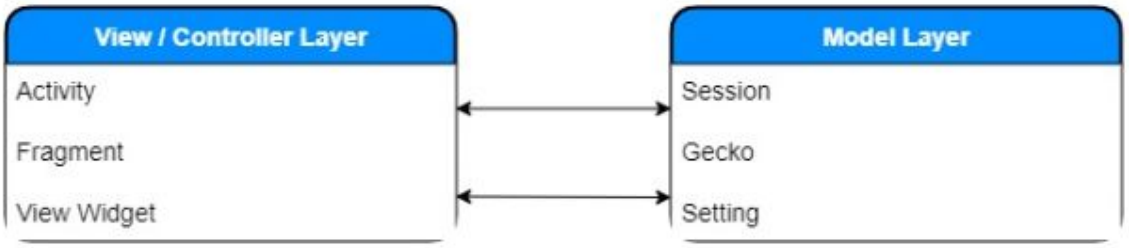
Set up:

- Git clone from our repository (branch name unknown for now)
- In the Android Studio, select the build variant as focusX86Debug
- Run the project with an emulator

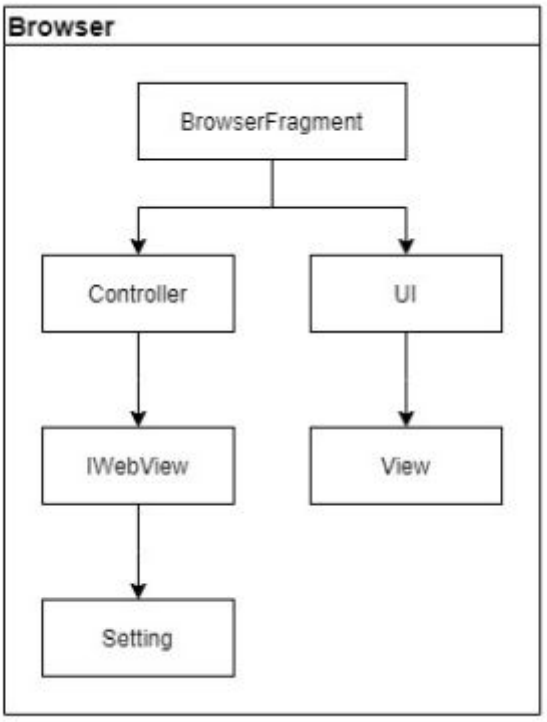
Step No.	Steps	Expected Results
1	Type any website's address in the URL toggle bar and go to that page	The button only will show up when the user goes to a webpage.
2	Drag and drop the button to see if the button can be moveable and stick to the left or right side of the screen	The button should stick to the right when move to the right , and to the left when move to the left
3	Click the button, there should be a menu expanded as a pie surrounds the button	The button should expand and show the correct submenu in the right direction. 1. button right, the submenu expands to left 2. button left, the submenu expands to right
4	Click each sub buttons, the functionality of each sub button should be present, for example, if the user clicks the "Refresh" button, the page should be refreshed, if the user clicks the "Delete" button, current browsing history should be cleaned and redirect to the default main page of Firefox-Focus	Each sub button should work correctly 1. Refresh should refresh the page 2. Delete should delete the browsing history and redirect to the default main page
5	Click the main button again, this time the menu should be closed and the main button should stay where it gets expanded	The sub menu should be closed. The main expandable button should not be moved.
6	When the menu is closed, the main button still can be dragged by users and functionalities will act the same as above saying	The main button should be draggable and stick to the page based on its position

Verify all the above functionalities work as stated when the user is following the above steps.

Architecture Revision

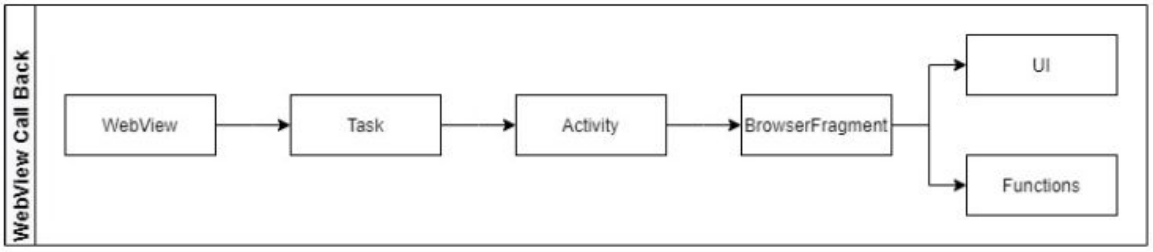


The project uses Android's traditional MVC structure. Activity, Fragment and View Widget are acting as both the View Layer and the Controller Layer. Some important time-consuming logic or operations related to user data are wrapped in the Model Layer, such as Gecko, Session, and Setting.

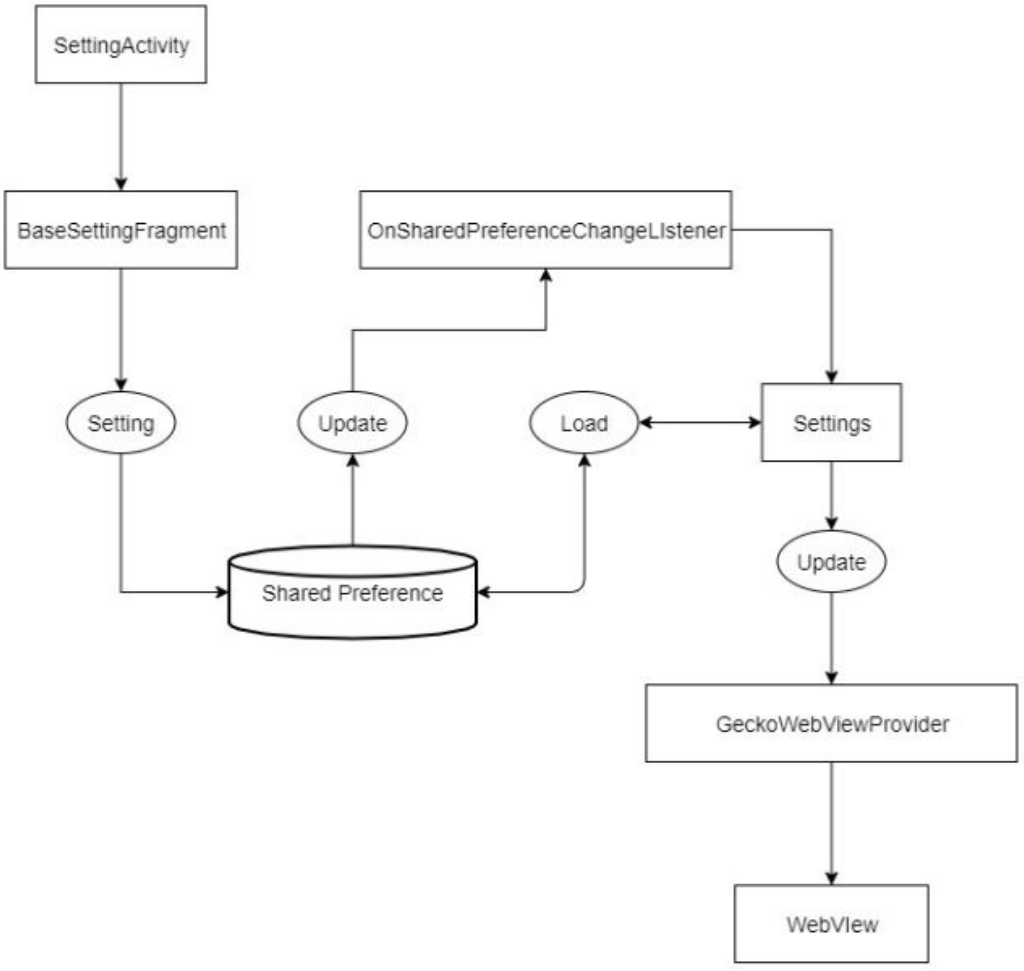


BrowserFragment inherits from Fragment, and provides control over the life cycle of the Browser application and an entry point for events. The events here include Key events, menu events, etc. BrowserFragment contains not only event logic code but also UI display. The complicated operations on the webpage are completed through

the WebView control. Some operations also need to use the user's default settings. For front-end UI display, it comes from the powerful Android framework View packages.



Since WebView itself is a controller that inherits from View. It also has the context information of Activity/Fragment. Therefore, WebVIew can create events and respond by BrowserFragment. A typical example is a long-click on an image and a pop-up ContentMenu. The process see above.



Each WebView has a Settings corresponding to it, but the browser as a whole, all of its WebView settings are the same, so a **singleton** class of Settings is defined in package utils. Therefore the entire system is well managed synchronously. The settings also need to be saved locally so that they can be synchronized when the next start. According to the way android data is saved, this configuration is saved locally on the phone by SharedPreferences.