Team 25: BJDJ

Brennan Law
Johnson Zhu
Jonathan Jarvi
Xu Dong Li

UML Commentary

Generally, each feature in Firefox Focus corresponds with a package containing a Fragment class which controls the UI for the content pertaining to the feature and potentially some classes to help manage the structure of the content. The 3 main fragments pertaining to the main browser features are BrowserFragment (UI for the browser), URLInputFragment (UI for the URL/search bar), and WebFragment (UI for a webpage's content). Interactions with elements within these fragments may introduce new fragments on the page or redirect the page to a different fragment.

In UML-Settings.pdf, we can see that the settings menu is composed of a separate fragment for each of the different types of settings and each of these classes implement the abstract class BaseSettingsFragment. The ActionBarUpdater is an interface inside BaseSettingsFragment which allows each subsetting to update the title and icon of the header of the current menu.

In UML-Biometrics.pdf, the BiometricAuthenticationDialogFragment simply holds the UI and most of the work is done by FingerprintManagerCompat.AuthenticationCallback which is a built in class for Android which deals with the authentication of a user using the fingerprint scanner. BiometricAuthenticationHandler is used to send requests and interpret the authentication results from the external class and a listener class BiometricAuthenticationListener is used to update the UI afterwards.

In UML-Autocomplete.pdf, the main Fragment is the AutocompleteListFragment which displays the list of custom domains added by the user via the AutocompleteAddFragment UI. Each domain in the list is formatted using a regex pattern matcher through

AutocompleteDomainFormatter. In the settings for autocomplete, (AutocompleteSettingsFragment), the user can choose to opt into or out of the autocompletion of default domains or custom domains and open the AutocompleteRemoveFragment UI to remove any domains they do not want to be autocompleted anymore.

In UML-SearchSuggestions.pdf, the class SearchSuggestionsPreferences holds the users settings for search suggestions. The SearchSuggestionsFragment UI has an adapter which updates the SuggestionViewHolder once results have been fetched from the web in the SearchSuggestionsFetcher class.

In UML-Menu.pdf, each of the items in the overflow menus are represented by ViewHolder implementations. This includes WhatsNewViewHolder for the "*What's New*" menu item and MenuItemViewHolder for regular menu items found in the overflow home menu, itself represented by HomeMenu. The browser menu section is structured differently from the home menu section, as it is associated and interacts with BrowserFragment, when home menu does not. As seen, a BrowserMenuViewHolder abstract class is used and extended from (multiple extensions but only 1 is shown) for different types of menu items in the overflow browser menu.

Software Development Process

We are using Kanban as our software development process for this project on Mozilla's Firefox Focus. There are four main principles for Kanban which embody the mentality promoted by Kanban. The first principal is "Start With What You Do Now", which recommends that Kanban is introduced as slowly as necessary such that it does not disturb any currently functioning processes. The second principal is "Agree to Pursue Incremental, Evolutionary Change", where incremental changes are suggested as they are likely to meet less resistance than large sudden changes. The third principal is "Respect the Current Process, Roles & Responsibilities", which states that it is not necessary to make changes to the existings roles. The fourth principal "Encourage Acts of Leadership at All Levels", strives to empower each member of the team to continuously improve, improving the overall strength of the entire team.

There are also six practices which describe the overall format of Kanban as a software development process. The first practice is "Visualize the Workflow", which requires the use of a Kanban board to display all the tasks currently known for the project, and displays the tasks current status. The second practice is "Limit Work in Progress", where a maximum is set on the number of tasks which can be in progress. This helps ensure that the developers are not multitasking too much, and helps reduce inefficiency. The third practice is "Manage Flow", this is to help identify any bottlenecks that can occur, and helps solve issues faster. The fourth practice is "Make Process Policies Explicit", which is to ensure that all the members of the team are able to remain informed about what other members are doing. Having everything properly documented and well labeled makes it far easier for other members to assist you if necessary. The fifth practice, "Feedback Loops", consists of meetings where developers are informed of

what all the other members of the group have done. The sixth practice is “Improve Collaboratively (using models & the scientific method)”, where developers use their knowledge of what others have implemented to provide suggestions and other feedback on methods that the implementation can be improved.

Kanban is a good fit for us since it is a process that works well with smaller teams of developers. Kanban is also simple to implement, even for those without any prior experience working with it and provides an easy visual overview of the status of the project with the Kanban board. We will be using github’s project board as our Kanban board because it allows the code and the board to be one place, as well as github’s built in automation that can help organize the cards. This board should help identify issues quickly and help maintain everyone’s knowledge of the entire project. Since we have a relatively short amount of time to work on this project, along with all the other work from other courses, Kanban’s limitation on the number of works in progress should help ensure that we stay focused on completing the tasks already in progress to ensure that there will definitely be completed requirements, and not a bunch of incomplete ones. We will be using a modified version of Kanban where the team meetups can be held over a voice chat and also changing the meeting frequency from daily to every other day, since we also have a variety of other courses this semester which would make scheduling daily in person meetings rather difficult.

Unlike in the Rational Unified Process where there are several prescribed roles for members of the team, Kanban prescribes that all members are aware of the entirety of the project, which allows us to work in whichever role we need currently and would make assisting other team members far easier. Where as in Extreme Programming it would also be impractical for us to

perform Pair Programming as we do not have a large amount of time for project, and Pair Programming significantly increases the time development of a program requires. It is also suggested to not have pairs between two novices, which poses another problem for us. Extreme Programming also possess strict schedules which may cause conflicts with the work required by other courses, and other commitments. One of the benefits for Extreme programming is the flexibility allowed for the requirements during an iteration, Scrum on the other hand is extremely strict is this regard. Kanban does not possess this issue, and allows requirements to be freely modified as needed. Incremental Delivery requires a large amount of user interaction for receiving feedback, and other suggestions to modify the requirements for the next iteration, it would not be easy for us to gather all the necessary data.