Integrating the KernelWorkflow toolkit with lore.kernel.org

Enhancing the Linux kernel developer interaction with mailing lists

David de Barros Tadokoro
University of São Paulo
email: davidbtadokoro@gmail.com
GitHub: https://github.com/davidbtadokoro
Last modified: 04/05/23

The KernelWorkflow project?

Nowadays, the Linux kernel is a ubiquitous and critical piece of software for the modern world, as it has been for years. Kernel development has a giant impact on the technology industry as a whole. The KernelWorkflow (also known as kworkflow or simply kw) project aims to provide tools for everyday tasks and be a unified environment for kernel developers.

Since I became involved in the kw project, I learned a lot about working with a community (which I find the most invaluable), the Linux kernel, bash, git, and the kw project.

You can learn more about the project at kworkflow.org.

Final Paper Proposal

The Linux kernel is collaboratively developed using mailing lists. Currently, kw already has a rich feature to aid in formatting and sending patches through email, but only a prototype for dealing with the recipient side of the patches. Actions like consulting the mailing lists, reviewing/replying patches, applying them, and building and deploying the kernel patch version are recurrent tasks for maintainers/contributors and anyone involved in the Linux kernel community.

With this in mind, my proposal aims to refine and expand `kw upstream-patches-ui`, which is the feature responsible for providing an interface with the lore archives of the mailing lists related to the Linux kernel (lore.kernel.org) and tools for dealing with the tasks listed above. In particular, I aim to allow users to use the feature in their patch review routine, letting them do inline reviews, reply with Reviewed-by, and much more. Also, the current user interface used for the feature is dialog, and I would like to at least experiment with using other interfaces like a web interface.

`kw upstream-patches-ui` starting point:

Below are screenshots of the starting point of the feature (before my contributions) and some interesting changes that can be done.

Figure 2 - Dashboard (main screen) of the feature

As we can see, it lacks some important menus, like one to manage the registered mailing lists (only when first launching the feature can the user register/unregister lists) and another to configure the feature settings (preferred dialog theme, default local kernel source tree, etc.). Also, we can see that the message box (upper section of the screenshot) is not related to this screen.

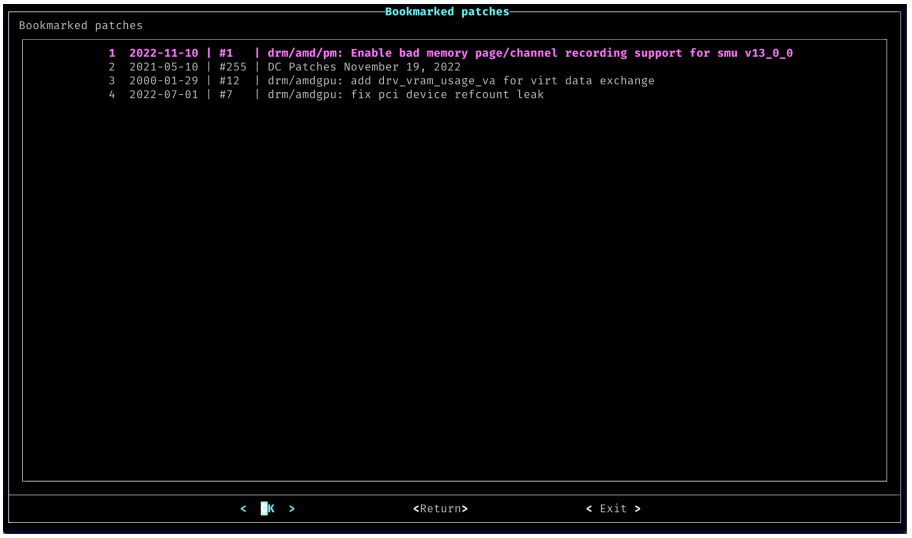

Figure 3 - Bookmarked Patches screen

The screenshot represents a desirable look and feel of the Bookmarked Patches screen. However, the patches displayed are hardcoded because the bookmarking of patches is not implemented.

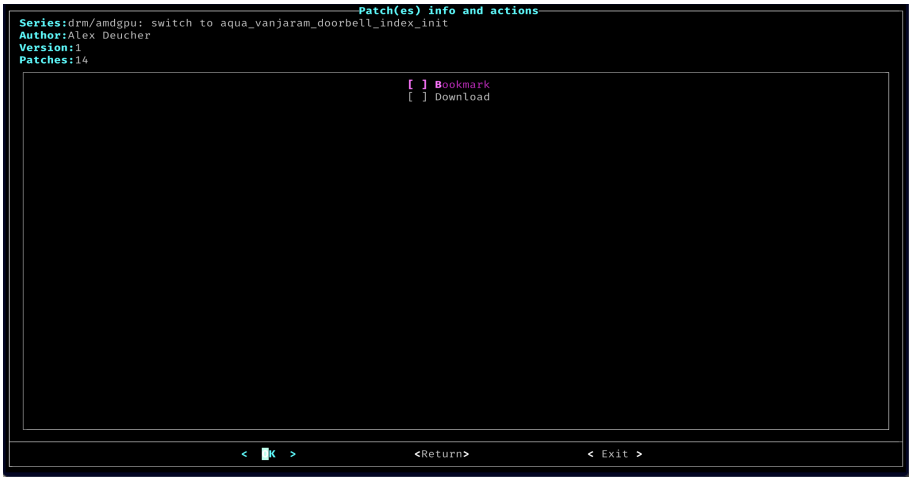

Figure 4 - Patch (series) details and actions screen

There are many actions that we can add here. Aside from implementing the bookmark action, we could add 'Apply', 'Build', 'Deploy', 'Reply with Reviewed-by', 'Reply with Tested-by', 'View', and more. We could also refine the 'Download' action to let the user choose where to save the patch (series) and implement a way to schedule actions.

`kw upstream-patches-ui` interface mockup:

Below are some mockups of the main screens related to the feature.

- **Main screen of the feature (Dashboard):**

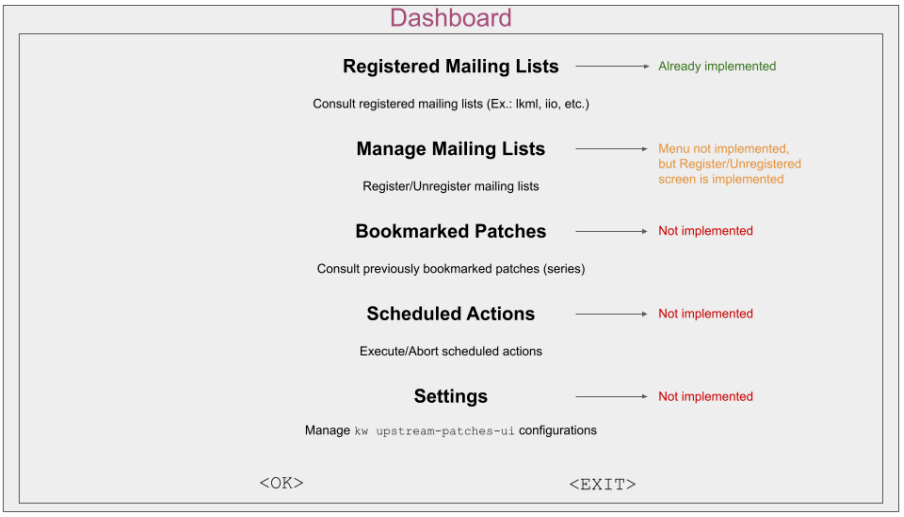

Figure 5 - Dashboard screen mockup

The Dashboard (**Figure 5)** is the first screen shown, besides when first launching the feature. Here the user will have access to the main menus:

1. Registered Mailing Lists: this screen is basically implemented and represents the menu to access the public mailing lists registered by the user.
2. Manage Mailing Lists: screen to register/unregister mailing lists. The screen itself is already implemented, however, there is no menu to access it from the Dashboard at the moment.
3. Bookmarked Patches: screen to view patches (series) marked previously by the user. For the same patch (series), the user will have almost the same view as if it was accessed through the 'Registered Mailing Lists' menu.
4. Scheduled Actions: screen with a list of patches (series) with actions scheduled to be executed in batch. Schedulable actions can be those that the user may want to run afterward that (potentially) can take some time, like applying a series of patches, building the series version of the kernel, deploying the series version of the kernel, etc.
5. Settings: screen to change configurations through the feature itself.

- **Registered Mailing Lists Sequence:**

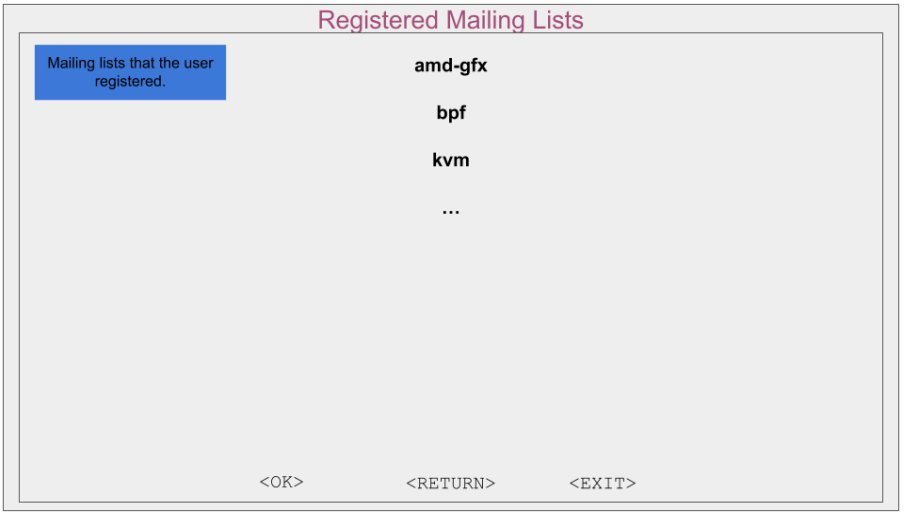

Figure 6 - Registered Mailing Lists screen mockup

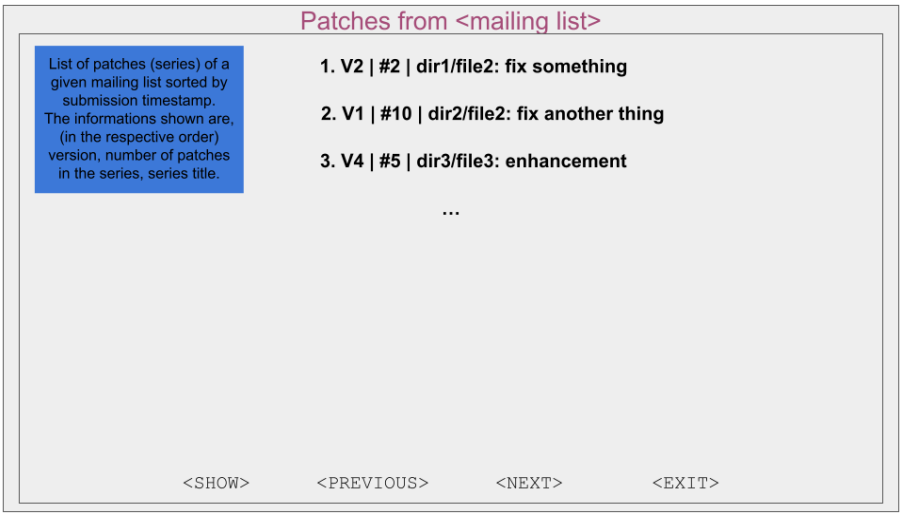

Figure 7 - Patches from given mailing list screen mockup

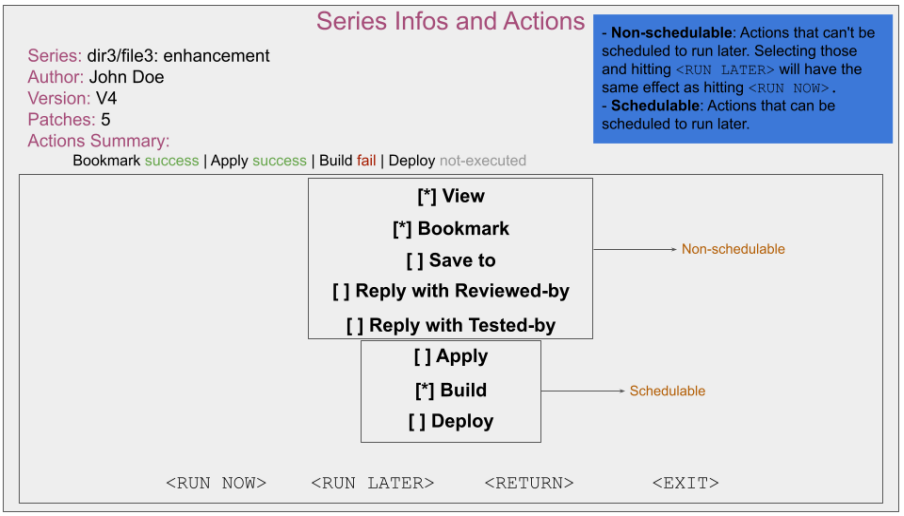

Figure 8 - Series informations and actions screen mockup

A screen with the registered mailing lists will be displayed when selecting the Dashboard menu 'Registered Mailing Lists' (**Figure 6**). After the user chooses a mailing list, the series of the list will be displayed from the latest to the oldest (**Figure 7**). As the volume of series from a list can be enormous, the series will be displayed on pages, and the buttons <PREVIOUS> and <NEXT> can be used to navigate through them. By choosing a given series with the <SHOW> button, the user can consult general information on the top of the screen (series title, author, etc.), check the status of some actions (if they ran and if they were successful or not) and select actions to run. Note that there are two types of actions: ones that can't be run later (Non-schedulable) and those that can (Schedulable) (**Figure 8**). If the user chooses to run some actions later by hitting <RUN LATER>, those will be displayed in the 'Scheduled Actions' menu. Hitting <RUN LATER> with actions checked that are Non-schedulable will have the same effect as hitting <RUN NOW> for those.

- **Manage Mailing Lists Sequence**

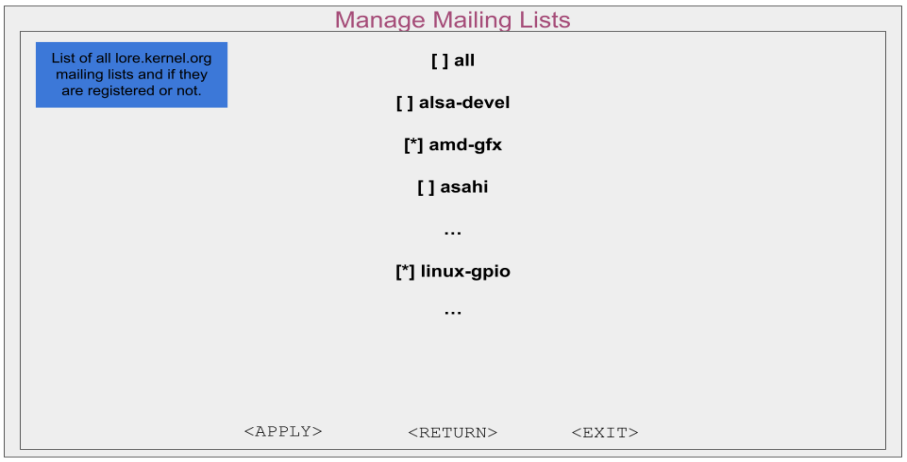

Figure 9 - Manage Mailing Lists screen mockup

By selecting the Dashboard menu 'Manage Mailing Lists' the screen to register/unregister mailing lists will be displayed (**Figure 9**). Important to note that the screen itself is already implemented.

- **Bookmarked Patches Sequence**

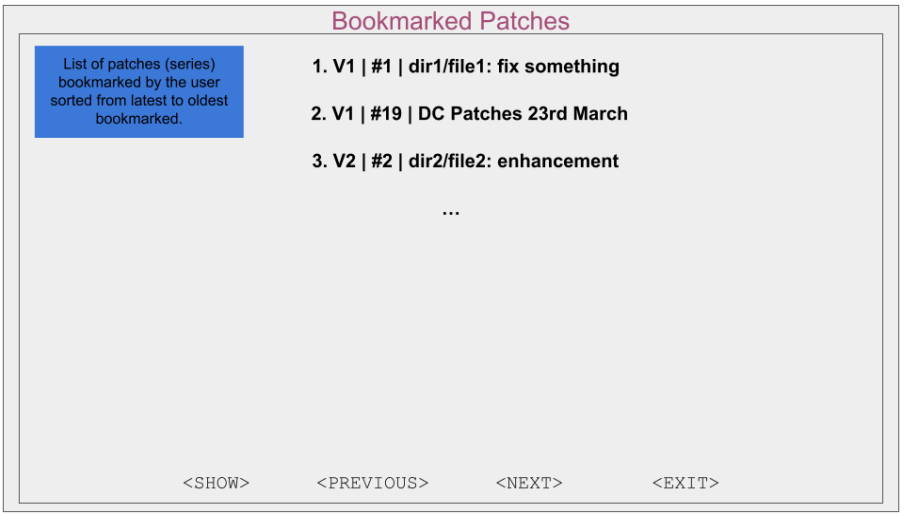

Figure 10 - Bookmarked Patches screen mockup

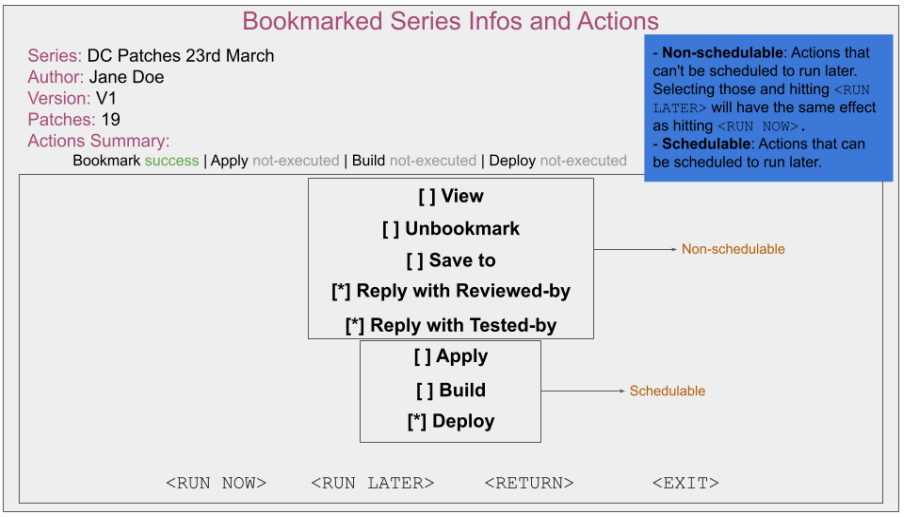

Figure 11 - Bookmarked Series information and actions screen mockup

When accessing the Dashboard menu 'Bookmarked Patches', a screen with the list of the bookmarked series, from latest to oldest bookmark, will be displayed (**Figure 10**). By selecting a given series, a screen with its information and actions will be displayed (**Figure 11**). This screen is almost identical to the one displayed when selecting a series directly from a given mailing list (**Figure 8**), with the difference that it has an option 'Unbookmark' that removes the series from the bookmarks after it is run.

- **Scheduled Actions Sequence**

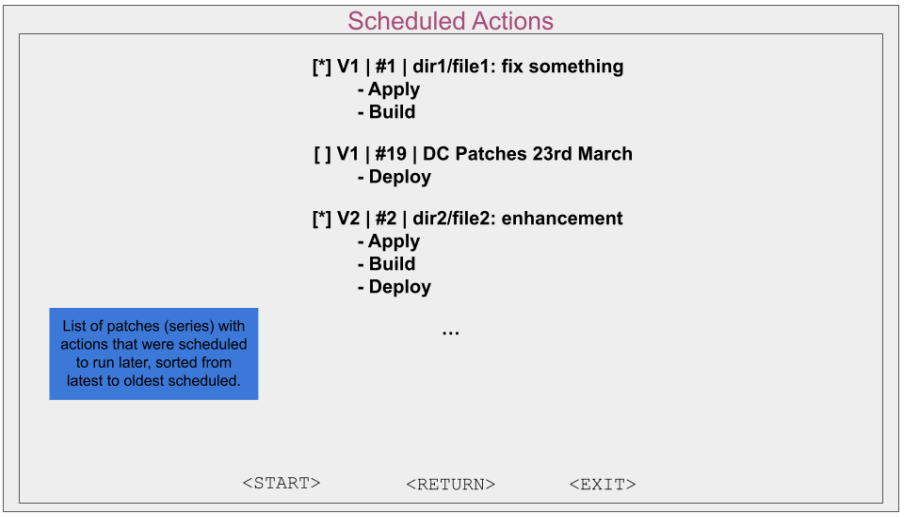

Figure 12 - Scheduled Actions screen mockup

By choosing the Dashboard menu 'Scheduled Actions', a screen with the list of series with actions that were scheduled from the 'Series Infos and Actions' (**Figure 8**) and 'Bookmarked Series Infos and Actions' (**Figure 11**), from the latest to oldest scheduled, will be displayed (**Figure 12**). By hitting `<START>`, the actions from the selected series will begin to run.

- **Settings Sequence**

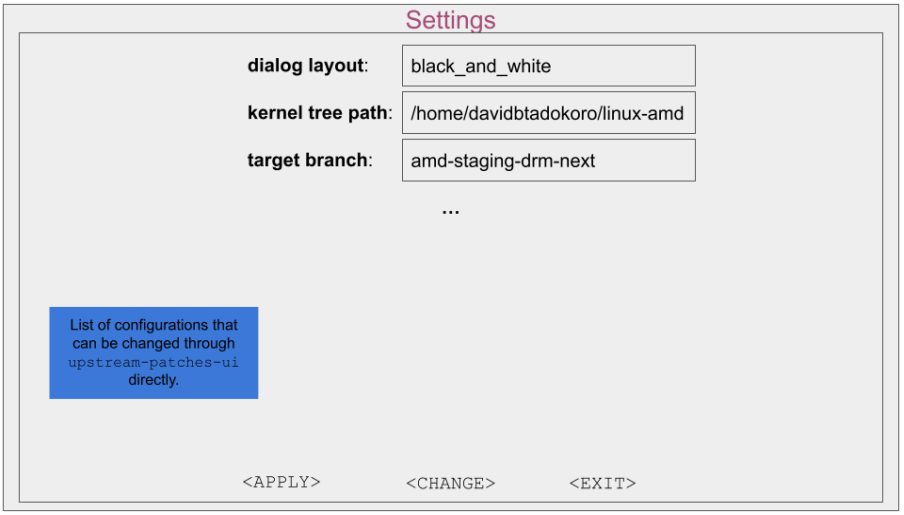

Figure 13 - Settings screen mockup

When selecting the Dashboard menu 'Settings', a screen with all the configurations that can be set through `upstream-patches-ui` will be displayed (**Figure 13**). By hitting `<CHANGE>` with a given configuration selected, the user can edit the given configuration; and by hitting `<APPLY>`, the changes will be applied.

- **Notes**

Figure 14 - Loading Screen Notification mockup

Selecting some menus (like selecting a given mailing list) and actions (like saving or building a series) will trigger a loading screen notification just to inform the user that there may be a delay between the request and the completion (**Figure 14**).

Wherever there is an `<EXIT>` button, hitting it will exit `upstream-patches-ui` and return to the shell. Also, wherever there is a `<RETURN>` button, hitting it will return to the previous screen. In the case of the `<PREVIOUS>` and `<NEXT>` buttons, hitting `<PREVIOUS>` on the first page will have the same effect as the `<RETURN>` button.

Deliverables

- `kw upstream-patches-ui`:
    1. Implement an interface that is user-friendly, simple, and without major bugs.
    2. Make the feature capable of downloading, building, and deploying patches and displaying the status of those actions to the user.
    3. Allow users to reply to the patches in the public mailing lists with Reviewed-by, Tested-by, and inline reviews.
- `kw mail`:
    1. Update feature codestyle.
    2. Fix known bugs.
    3. Improve feature where possible.

Proposal Timeline

Week 1: May 4 - May 7
- Integrate `kw build` with `upstream-patches-ui`
- Add basic documentation for `upstream-patches-ui`
- Add Bash and Zsh completions for `upstream-patches-ui`

Week 2: May 8 - May 14
- Integrate `kw deploy` with `upstream-patches-ui`
- Get used to `kw mail` codebase

Week 3: May 15 - May 21
- Add 'Settings' menu to Dashboard screen
- Add `upstream-patches-ui` short option
- Map codestyling improvements to `kw mail`

Week 4: May 22 - May 28
- Add 'Manage Mailing Lists' menu to Dashboard screen
- `upstream-patches-ui` can't download patches with apostrophe in title
- Fix/update `kw mail` codestyle

Weeks 5, 6, 7, 8 and 9: May 29 - July 2
- Refine the 'Download' action on `kw upstream-patches-ui`
- Make `upstream-patches-ui` query X patches
- Enable query for specific string from lore
- Add the possibility of reply a patch with Reviewed/Tested-by
- Add possibility of viewing patch changes through `upstream-patches-ui`
- Add inline review of patches
- Fix some bugs and make some enhancements to `kw mail`

Week 10: July 3 - July 9
- Refine a prototype of the feature that can be used for a full patch review routine

Midterm Evaluation week: July 10 - July 14
- Finalize and submit midterm evaluation

Weeks 11, 12, 13 and 14: July 17 - August 6
- Refine `upstream-patches-ui` local database
- Revise `kw mail` documentation

- Experiment with other views for `kw upstream-patches-ui` besides `dialog`

Weeks 15, 16 and 17: August 7 - August 27

- Validate features and User Experience
- Make refinements and features improvements

Remaining weeks: TBD