

User Interface Design Final Report

Jordan Dehmel, 2025

Creating an Open-Source Text Editor for Static Documents with Embedded Running Code

Abstract: When preparing documents in math or computer science courses, PDFs with embedded running code are often desirable. R-markdown (RMD) is one solution to this, but is embedded in the large rstudio application. Previous work has created the JKnit literate programming tool, which is a lightweight and generic command-line alternative to RMD: The current project details iteration on a graphical text editor for JKnit as the final project of CSCI 337 (User Interface Design).

Purpose

TODO

Requirements

TODO

Audience

TODO

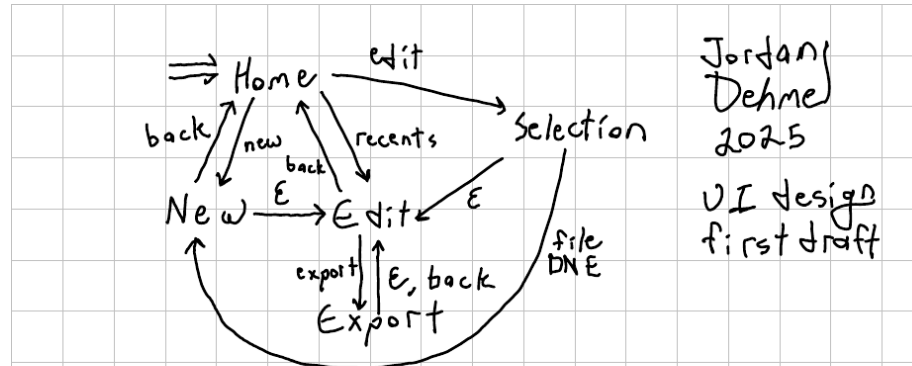
User Stories (See also UI Test)

TODO

First Iteration

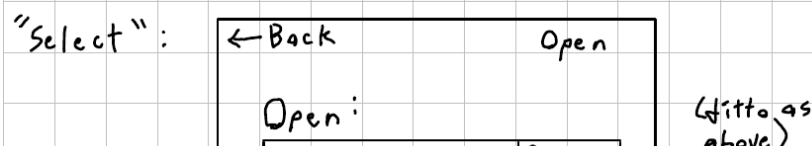
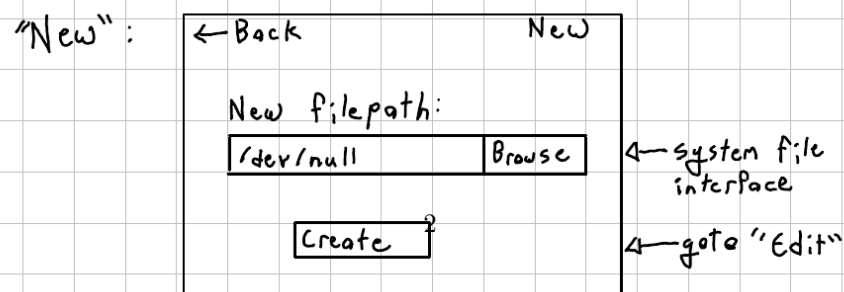
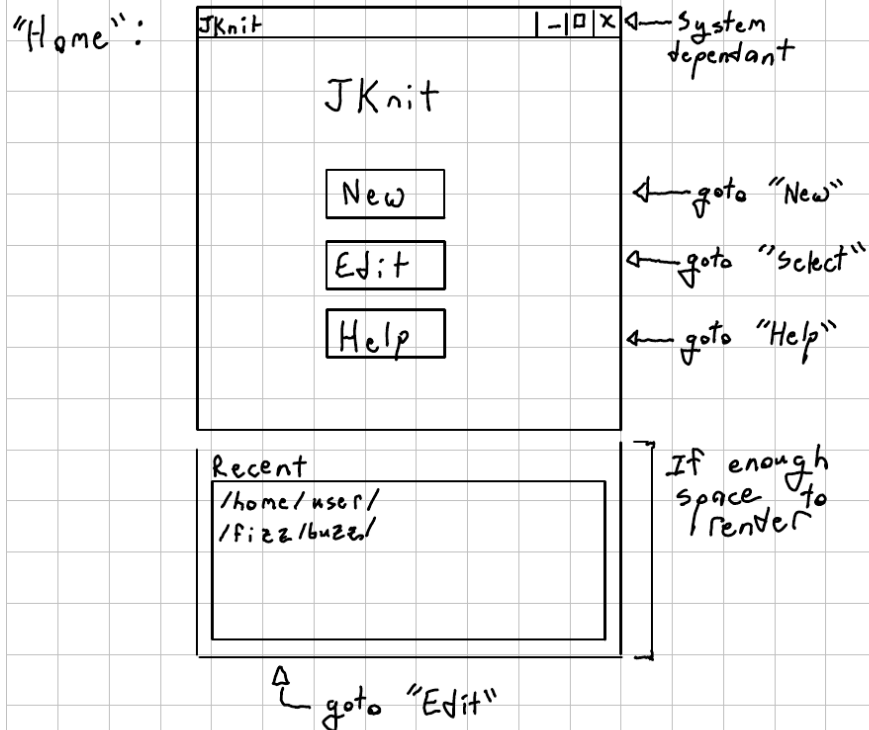
The first major iterations were as follows: 1. Paper version 2. Android version 3. First Flutter iteration

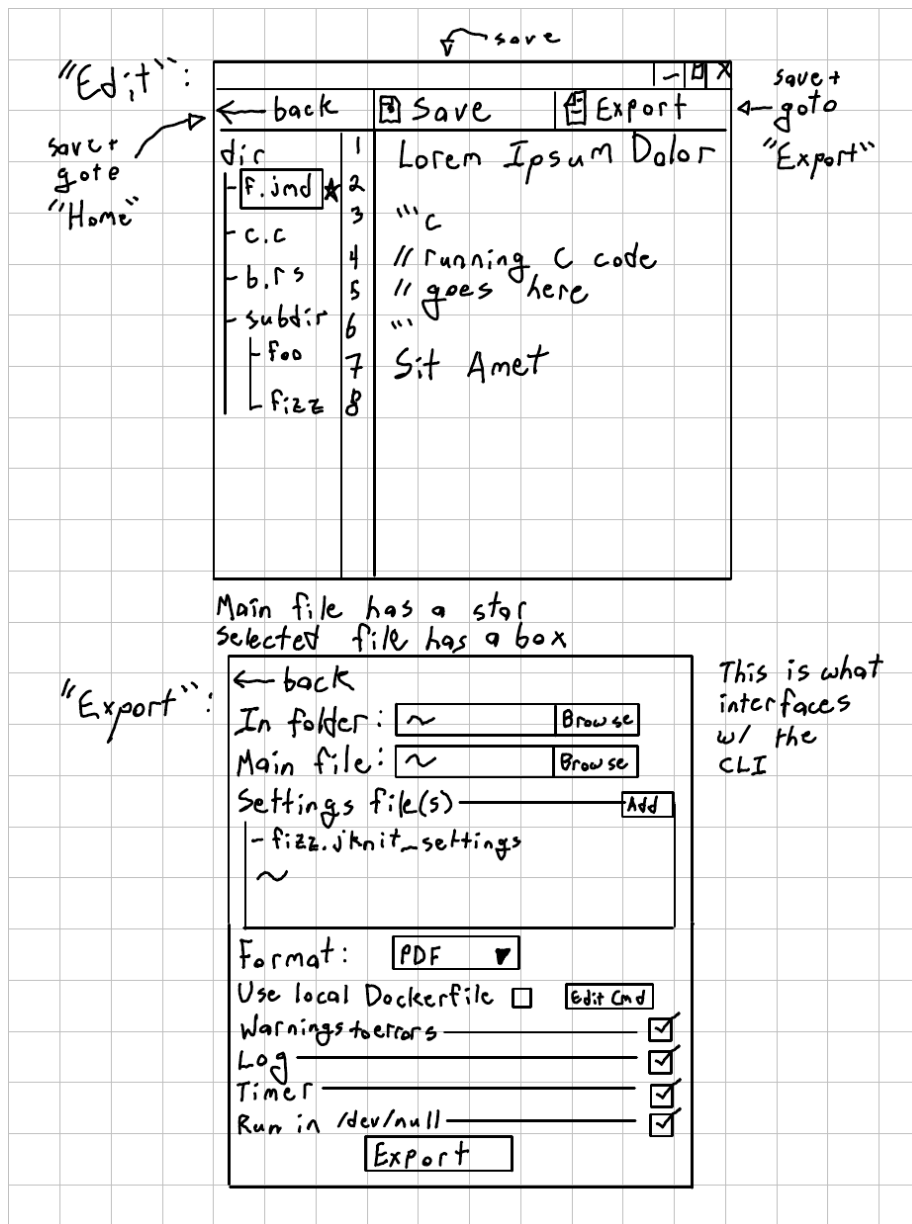
Paper UI



Jordan
Dehme
2025

UI design
first draft





UI Test

Stakeholder Stories / Test Cases

1. "I am an intro to applied math student and I want to create a new document to edit, then a new file in the current document"
2. "I am an intro to applied math student and I want to find out what

the ‘Dockerfile’ option means in the export menu, then open an existing document named ‘hello.jmd’ ”

3. “I am a teacher and I want to demonstrate how to create and export a document to `markdown` format in my class”

Measuring Success

1. Test case 1 will be considered a success if they are able to create a new document. It will be a partial success if they cannot create a new file. It will be a failure if they are not able to create a new document.
2. Test case 2 will be considered a success if they are able to navigate to the help menu, then to the ‘Docker’ menu. Then they should be able to navigate back to home and open an existing document. It will be a failure if they are unable to find help or to escape the help menu and open an existing document.
3. Test case 3 will be considered a success if they are able to create a new document, then successfully reach the **export** screen. It will be a partial success if they stop at this point, and a full success if they are able to change the default document type and export. It will be a failure if they are unable to create or export a document.

Experimental Design

- Only the vaguest follow-up questions will be asked after the fact so as not to retroactively influence their opinions

The following are both experimental design points and things I will say to the subject before beginning:

- I will ask them about their experience in CS and math, especially in writing reports
- I will have my paper design and simulate the computer
- I will instruct them to narrate their thoughts and actions, and I will simulate those actions
- I will give them a task that they will try to complete
- For text blocks and dropdowns that I have not detailed in my paper model, I will narrate what will be displayed.

Choosing A Subject

Test subject criteria:

- College student
- In a math / CS class (preferably applied math)
- **Not** an experienced CS person

Post-Test Reflection

1. Task 1 was a partial success: She was able to create a new document, but there was no clear way to make a new file in that document’s folder, and

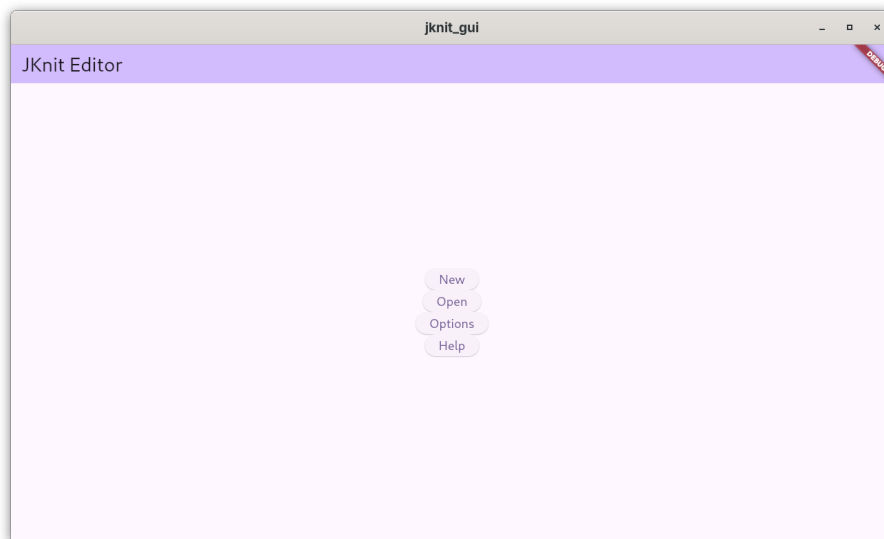
she made a second document instead.

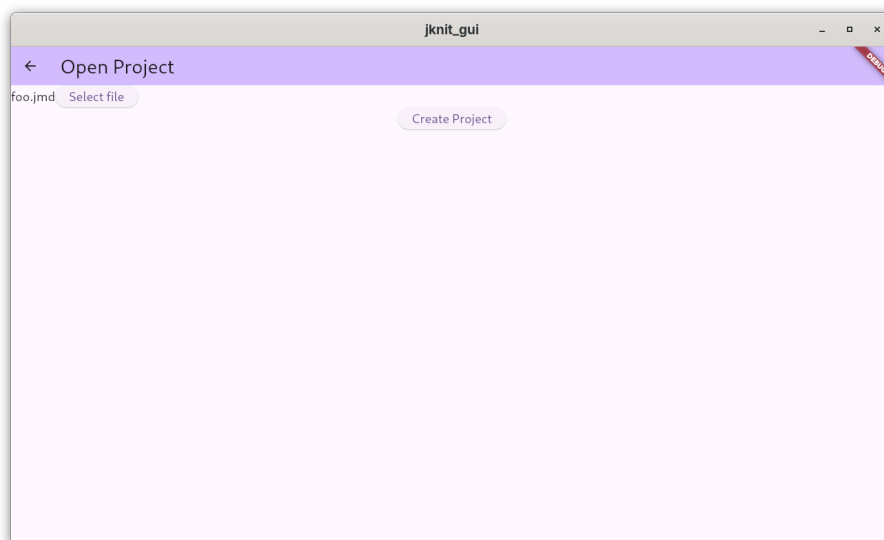
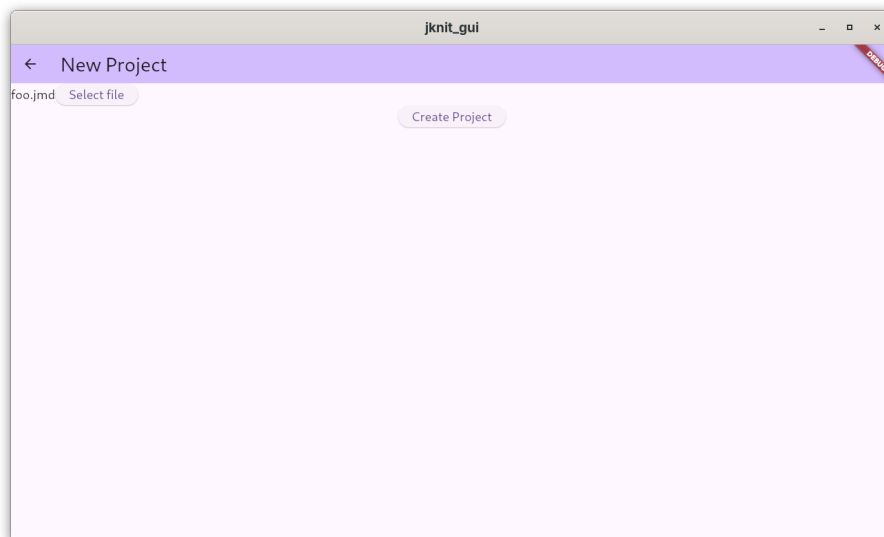
2. Task 2 was a success with notes. She found it unclear how to find help for a specific keyword, and suggested a search function. She was able to navigate back to the home menu and open an existing document, although she suggested different verbiage for the button doing so.
3. Task 3 was a partial success. She was able to create a new document and navigate to the export menu, but was unable to proceed without further direction. She found this menu overly technical.

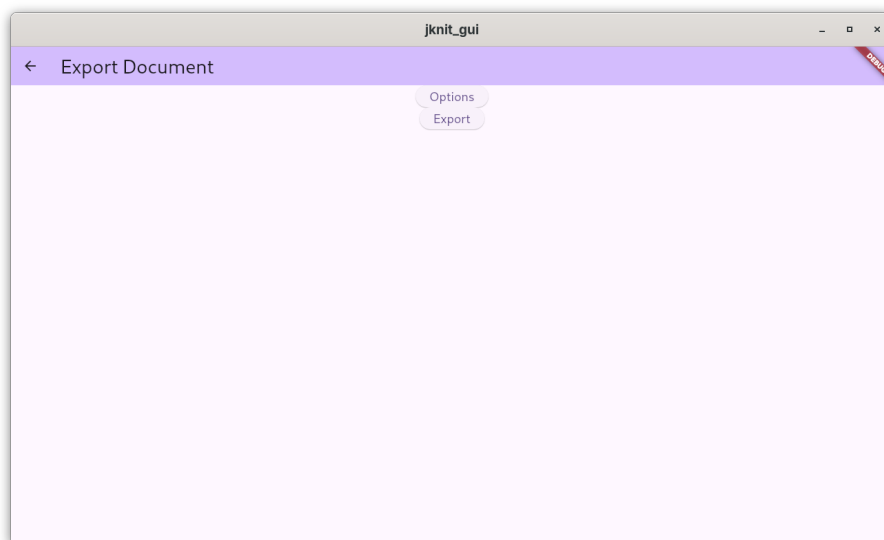
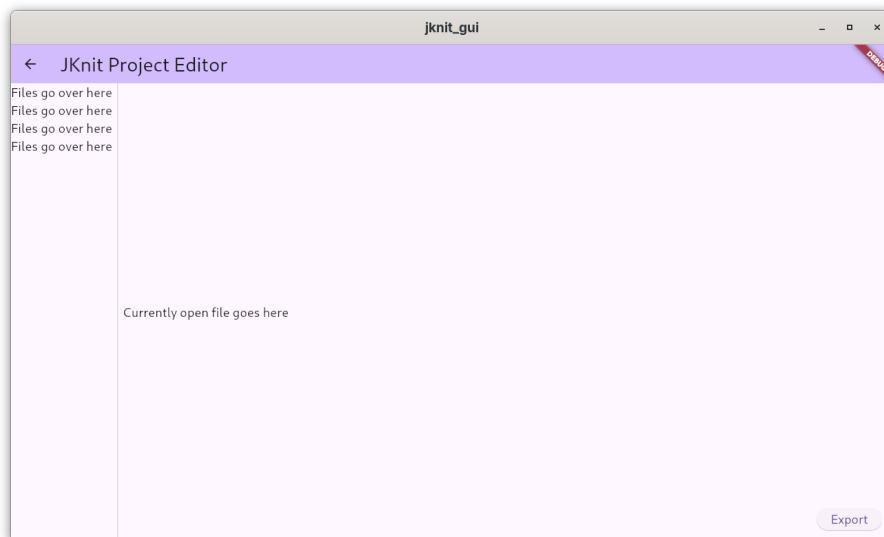
The overall impression was that the app was overly technical and lacking in help. A search bar for keyword help and an “advanced settings” submenu would likely solve these problems. The menu’s “edit” button should also say something more like “open”, and the help menu should be more intuitive.

First Flutter Iteration

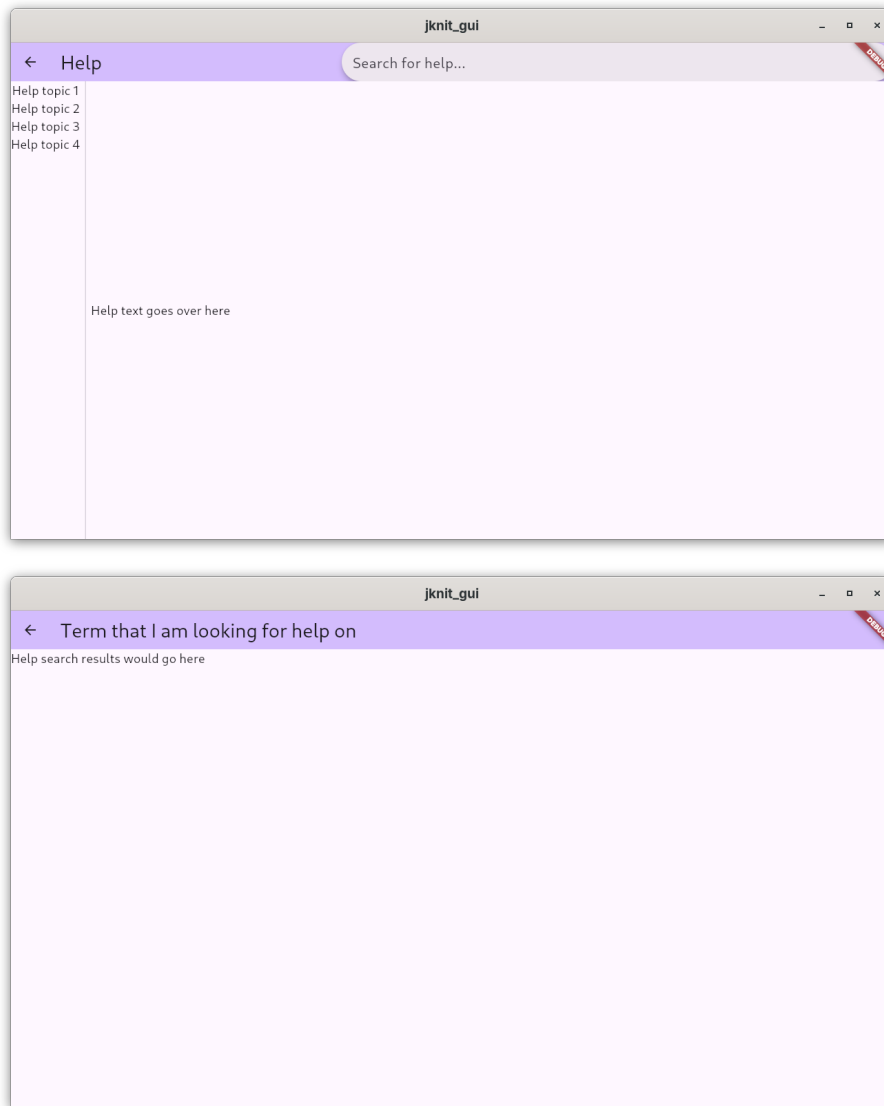
The first Flutter iteration was a proof-of-concept. The goal was not so much to make a *pretty* app, but to make a *functional* app. Accordingly, it is sparse and bland but functional.











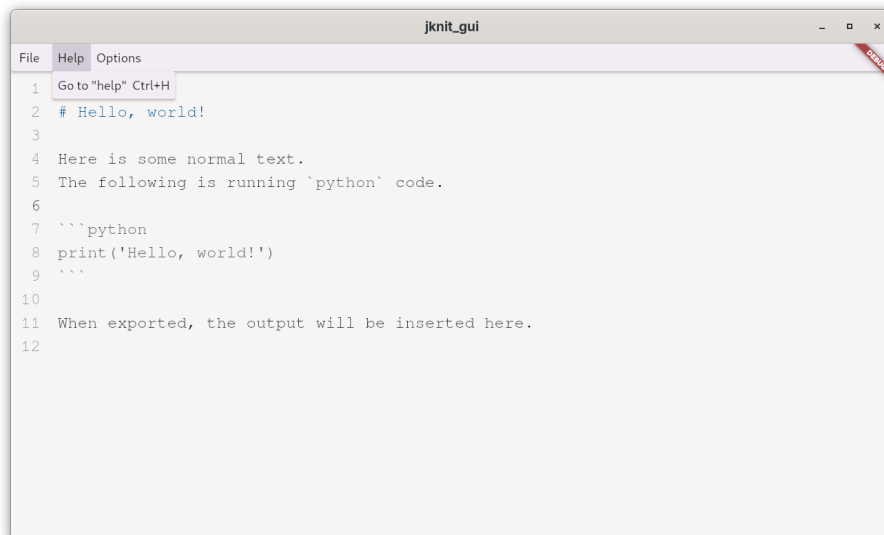
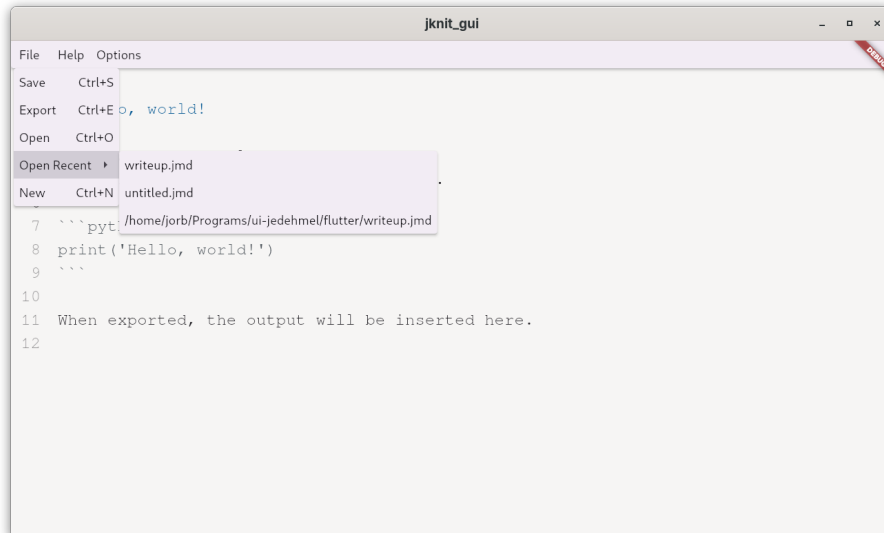
This is a good place to start, and more-or-less reflects the original paper design.

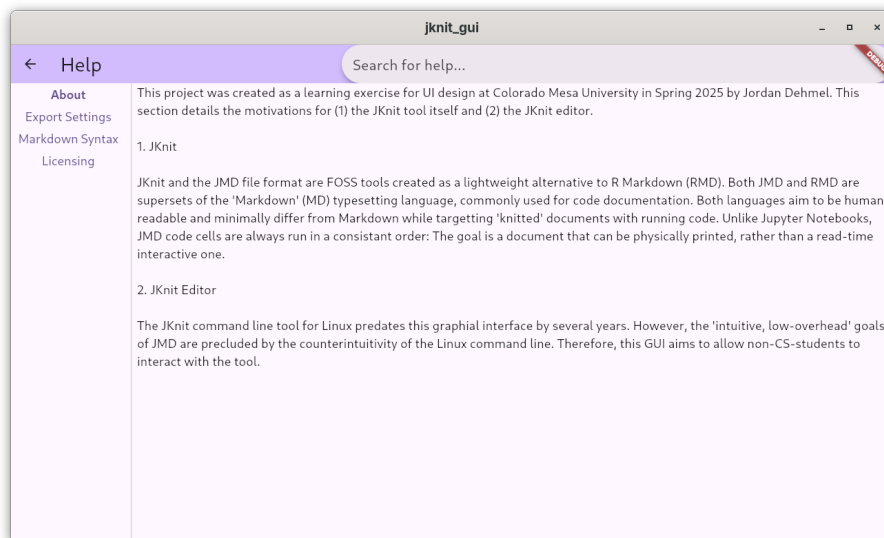
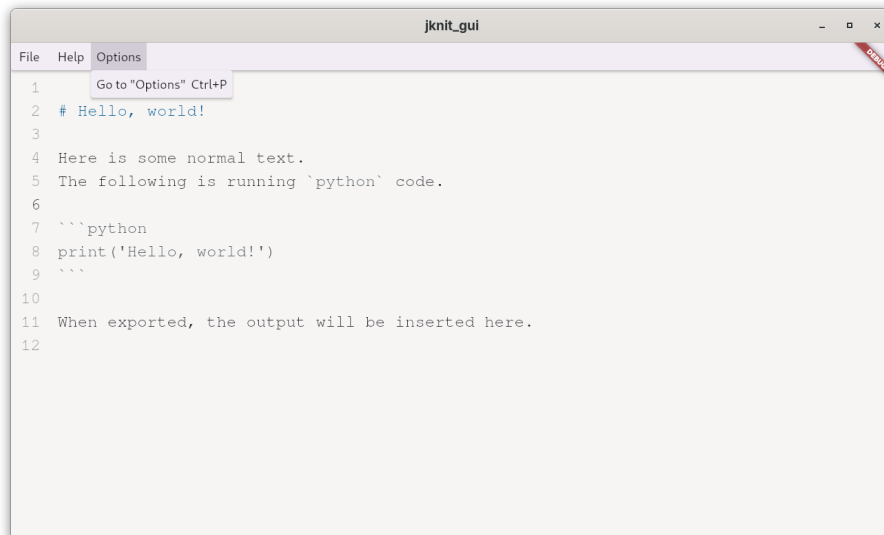
Design Choices Made on UI/UX Principles

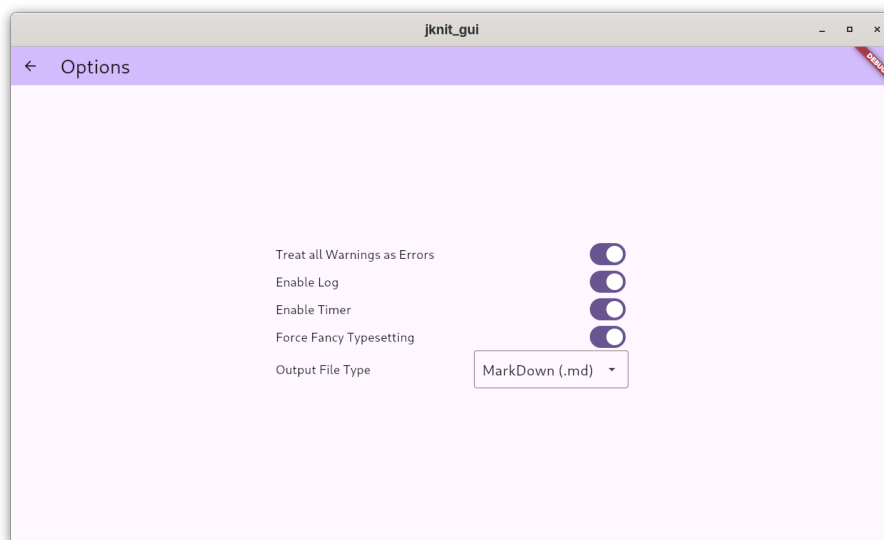
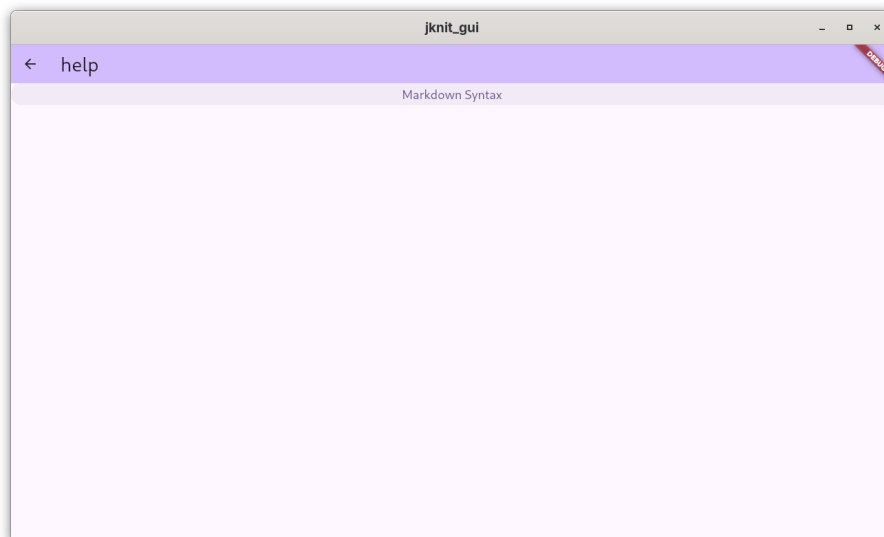
1. Removed home screen, replaced with most recent file editor
 - Users will not usually need this intro: It would only be helpful for first-time users
2. Replaced **New** and **Open** with system dialogues
 - They had previously only contained a button that opened the system dialogues, and thus were redundant

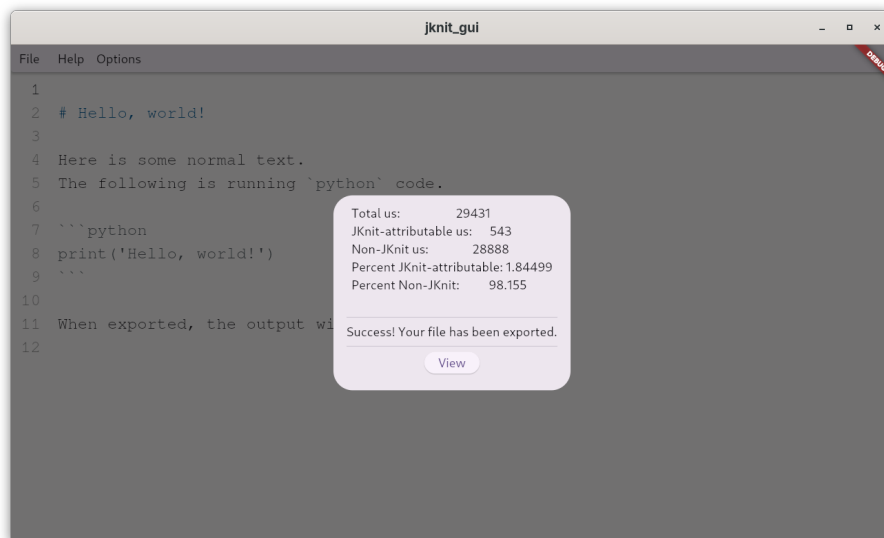
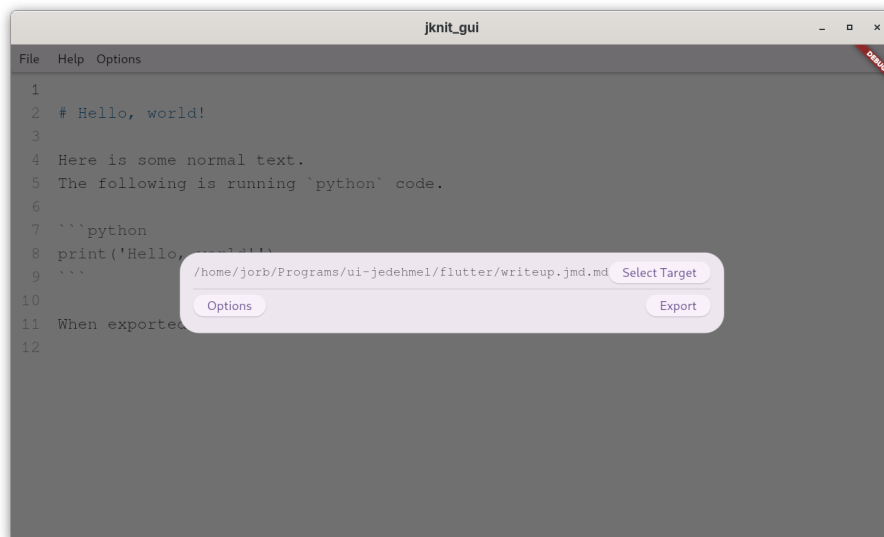
- This choice avoids an additional unnecessary step which might have confused or annoyed users
3. Removed initially proposed file bar on left-hand-side of editor
 - A `jmd` document is a single file, so I realized that it was not necessary to have a vscode-style file selector. A single file being open is fine
 - This eliminates wasted space on the screen and removes an opportunity for the user to be confused
 4. Made the **Export** screen a pop-up dialogue
 5. Centered options in the middle of the screen
 6. Left-aligned option labels and right-aligned option toggles
 7. Made export error/success a pop-up
 8. Added system-style file bar at top of editor
 9. Added traditional key bindings in editor
 10. Added **recents** tray in editor
 11. Added **View** option for exported Markdown documents

Final Product











Final Unit Testing

```
● jorb@archlinux ~/Programs/ui-jedehmel/flutter (dev)> flutter test --coverage
00:04 +1: All tests passed!
○ jorb@archlinux ~/Programs/ui-jedehmel/flutter (dev)>
```

Conclusion

TODO