Open-Source Sorcerers

Professor Nick Murphy

SWE 4783 – User Interaction Engineering

Kennesaw State University – Spring 2025

# Sprint 1 - Introduction

In this sprint, we have created multitude of numerous software artifacts to better understand and explain our proposed feature design of the “Undo closed tab” to the Kiwix software. We have provided this document as a hub for all of our artifacts in one place for easier readability. The first artifact this document covers is the wireframe of the new design layouts for our new feature in Kiwix. We used the wireframing tool wireframe.cc, and we found that this tool was extremely intuitive and easy to pick up as first-time users.

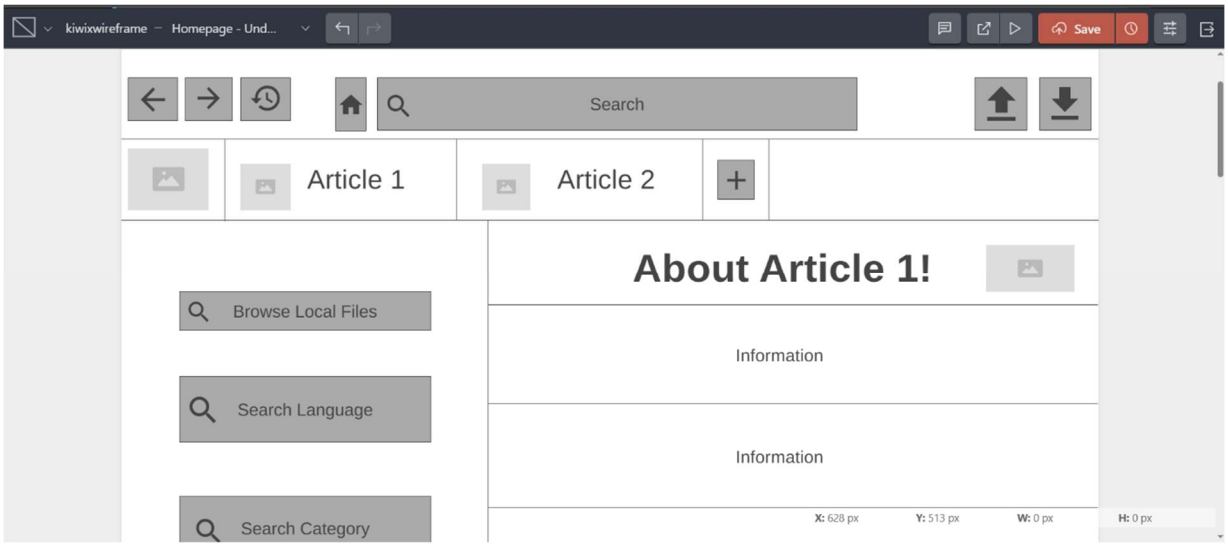
The second software artifact in this document is the storyboards that further explain the interactions users will partake in when utilizing the feature we will be implementing. We have also targeted the files we will be changing in the code base to make our vision a reality. We have isolated those files into this document and explained how we will be changing them to push back into the code base. On top of that, we have also included a short video showcasing a small change to the UI. The next page will start explaining the wireframe for our feature.

## Wireframe – Matt Crowley

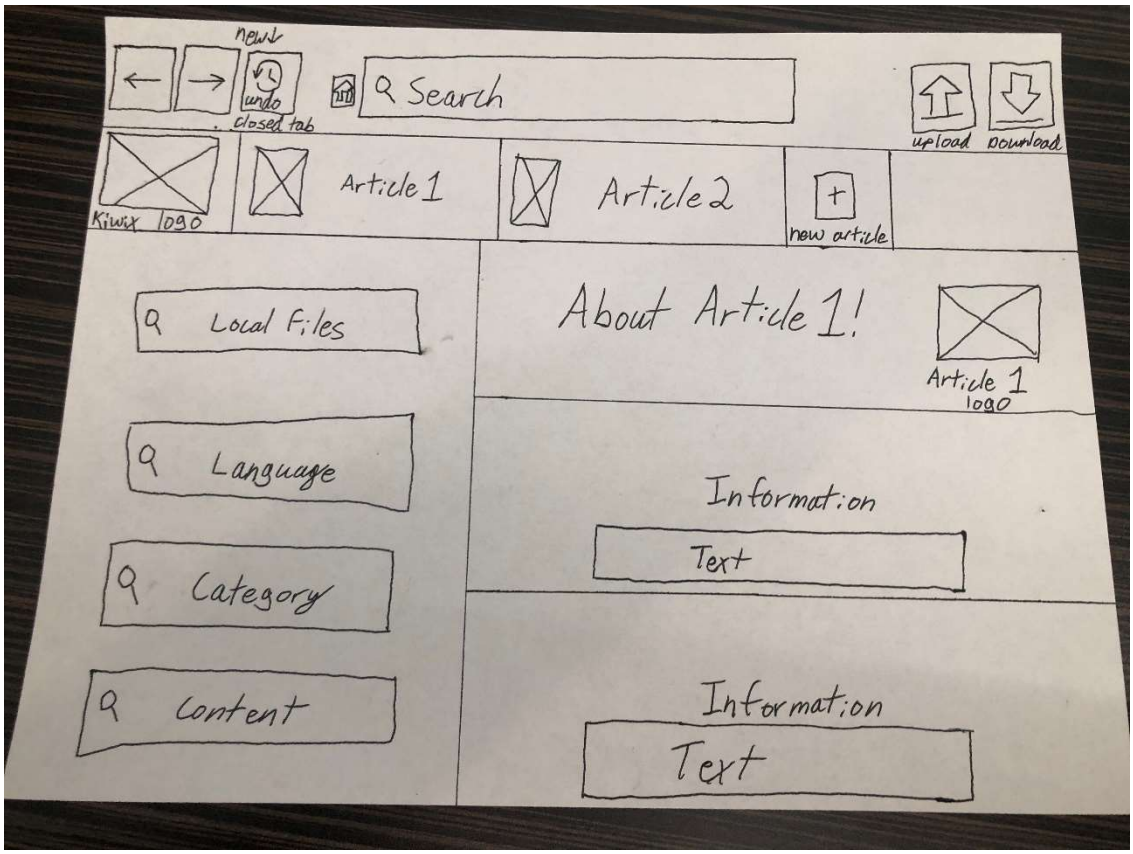
Wireframe link: https://wireframe.cc/pro/pp/e369b35ca882091

Wireframing is important for us to simply illustrate the changes we will be making in the Kiwix application. Since our project is focused around implementing a new feature to further enhance user interaction and usability, it was important for us to not completely make Kiwix unrecognizable in our final redesign. We spent a long time deciding how best to implement our "Undo closed tab" feature in Kiwix, and we finally decided to add a button near the search bar that looks like an undo button that many internet users are familiar with. Our wireframe is an extremely bare-bones and has the simple design of the system with the new change. In our digital wireframe, the button is located near the search bar at the top, and right next to the left and right cursor buttons in the top left corner.

We decided to place this button here because the users will spend a lot of their time traversing tabs and articles at the top of the screen, and we do not want them to go to a random corner of the screen to undo the tab that they may have closed for any reason that they wish to retrieve.

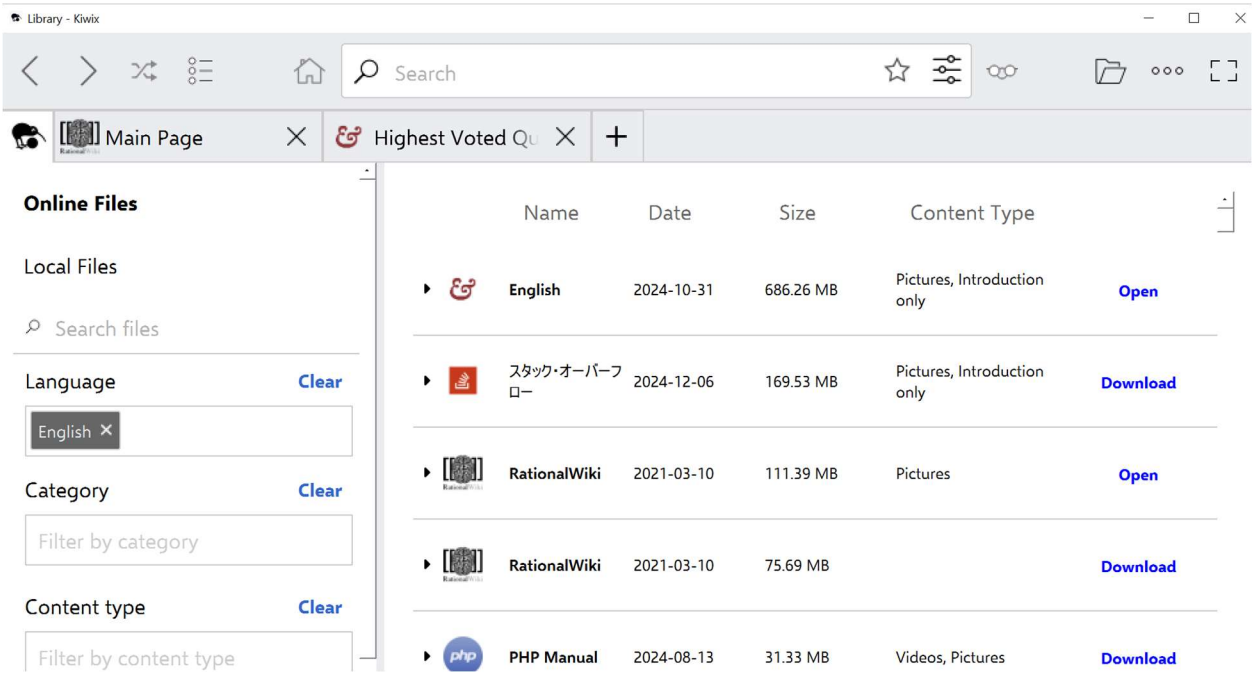


We have also hand drawn the wireframe for our proposed implementation. This allowed us to better understand what we were trying to accomplish and how it would all fit into the Kiwix UI without having to make too many corrections in wireframe.cc. As you can see below, our initial hand drawn wireframe on paper was not deviated from much at all in the final revision in the digital wireframe that we created.



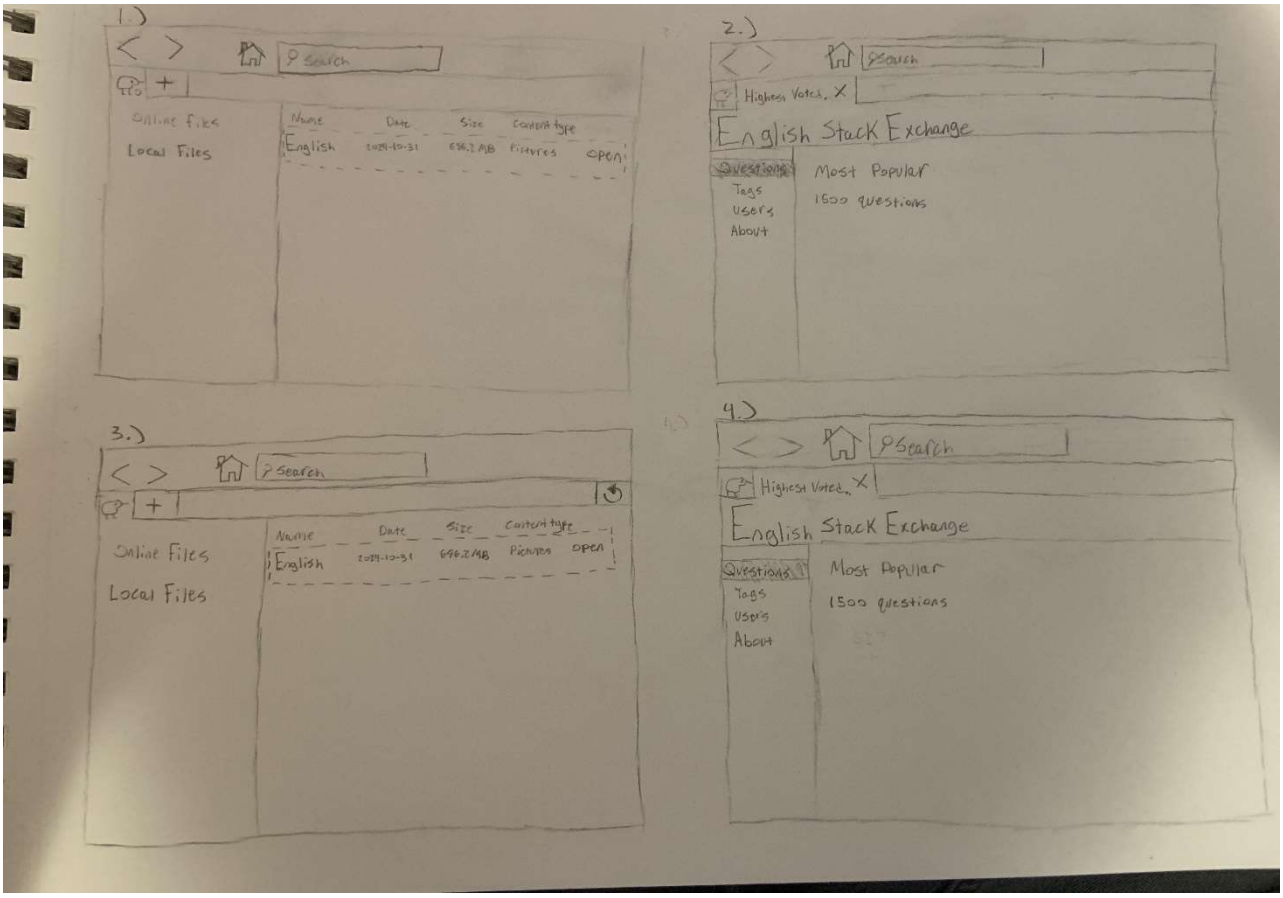

With our new feature, we have strategically planned to Kiwix UI as true to its former self as possible. This mean designing the new button exactly like all of the other buttons in the region of the user's screen near the search bar. We will utilize proper spacing and color matching of this button to make it consistent with all of the other buttons in the region. This new button will further enhance user experience, interaction, and flow during the use of Kiwix; all while keeping the majority of the UI true to its former self.

Below, we have included an image of the true Kiwix layout as seen on their most recent version. Our “Undo closed tab” feature will go in the whitespace between the four buttons on the top left side of the application and the home button right next to the search bar of the application. We will resize each button slightly to make sure the screen will appropriately display the new “Undo closed tab” button without spearing out of place.




During our testing phase of deciding whether or not to work on Kiwix for our project, we were frustrated when we accidentally closed an entire tab of articles and content. We discovered that downloading content for education took a long time due to some sources like Wikipedia sizing over 100 gigabytes of data. This means that users should feel safe when viewing the content, and not have to worry about having their time wasted if they accidentally closed a tab. We are eager to implement this new feature to Kiwix, and we believe that with the proper planning with storyboards, wireframes, and other artifacts, our feature will be useful to many users.

**Storyboard – RJ Straiton**




This storyboard showcases how users will utilize our “Undo closed tab” button and how that interaction would go. The first screen shows the home page of the Kiwix desktop application. From here, users can search for articles or open articles which they may have already downloaded. The second screen shows a new tab after opening an article that I already had downloaded during our initial research. The third screen takes us back to the Kiwix home screen after closing out of a tab, with a new undo button appearing on the right side of the tab portion of the screen. Clicking this undo button will take the user to the fourth screen where their previously closed tab has been restored, and they are able to continue from where they had left off.

## Updated Code– Ian Chorne

When attempting to build the project in the QTCreator IDE, we were getting -Winconsistent-missing-override errors in the searchbar.h and tabbar,h scripts due to certain methods not being explicitly overridden.

This error was resolved by changing the following lines in searchbar.h:

```
// BEFORE:
virtual void focusInEvent(QFocusEvent *);
virtual void focusOutEvent(QFocusEvent *);

// AFTER:
void focusInEvent(QFocusEvent *) override;
void focusOutEvent(QFocusEvent *) override;
```

And these lines in tabbar.h:

```
// BEFORE:
virtual QSize tabSizeHint(int index) const;
void mousePressEvent(QMouseEvent *event);
void paintEvent(QPaintEvent *);
```

```
// AFTER:

QSize tabSizeHint(int index) const override;

void mousePressEvent(QMouseEvent *event) override;

void paintEvent(QPaintEvent *) override;
```

The custom code we wrote can be found in the header file for the tabbar, tabbar.h, and the logic file for the tabbar, tabbar.cpp. The snippets can be viewed here:

tabbar.h:

```
//Just a line to initialize the button as a pointer, it is null by default:

  QToolButton*     m_undoButton = nullptr;
```

tabbar.cpp:

```
//statement in the constructor to include button in the tabbar:

//UNDO BUTTON m_undoButton = nullptr; createUndoButton();

// Show the button if a tab is closed
connect(this, &TabBar::tabRemovedSignal, this, [=]() {
  m_undoButton->setVisible(true);

  //Hide after five seconds
  QTimer::singleShot(5000, this, [=]() {
    m_undoButton->setVisible(false);
  });
});
//method for undo button logic:

void TabBar::createUndoButton() { qDebug() << "Creating undo button"; m_undoButton =
new QToolButton(this); m_undoButton->setObjectName("undoCloseTabButton");
m_undoButton->setIcon(QIcon(":icons/undo.svg")); m_undoButton->setToolTip(gt("undo-
close-tab"));
```

```
// Button styling
m_undoButton->setStyleSheet("background-color: yellow; min-width: 30px; min-height:
30px;");
m_undoButton->move(50, 5);
m_undoButton->setVisible(true);

// Position it in a very obvious place for testing purposes
m_undoButton->move(500, 5);

// Connect button click to undo action
connect(m_undoButton, &QToolButton::clicked, this, [=]() {
   qDebug() << "Undo button clicked!";
});


}
```