Project: Comments in Sidebar

Name: Mohit Marathe
Country: India
Email: mohitmarathe@proton.me, mohitmarathe23@gmail.com
LinkedIn:
https://www.linkedin.com/in/mohit-marathe-9a3605232/
IRC Nick: macroni23

Project Overview

The goal of this project is to enhance the **LibreOffice Writer** by implementing a **comments sidebar** feature. Currently, comments are displayed at the document margin, which can be cumbersome and disrupt the user's workflow. By adding the comment section in the sidebar, the aim is to improve usability, streamline the commenting process, and enhance collaboration within the application.

About Me

Hi there! My name is Mohit Marathe. I'm currently an undergraduate studying at the prestigious Indian Institute of Technology (BHU), Varanasi.

I realized early on that my passion for coding goes beyond just writing code. I wanted to actively collaborate and learn from others while adding value to the community. That's when I began contributing to open-source projects.

In 2023, I participated in Google Summer of Code as a mentee for **VideoLAN.** I contributed to VLC Media Player 4 (for Windows & Linux), by adding various features ranging from enhancing track synchronization, adding more options in context menu,

introducing preset configurations in the Preferences. I also added support to toggle playback with mouse, improved the control interface for interactive videos and so on. You can find my project here: https://summerofcode.withgoogle.com/archive/2023/projects/z26bcE5j

Being part of an open-source community has given me a sense of purpose and fulfillment like nothing else. It's an incredible feeling to work on something that has the potential to impact people all around the world. And it's not just about the end result - it's about the journey of learning, collaborating, and growing as a developer.

My Contributions to LibreOffice:

All of my contributions to LibreOffice can be found here: https://gerrit.libreoffice.org/q/owner:mohitmarathe%2540proton.me

Note: *I'll continue to add my contributions here until GSoC selection. So that I can become familiar with the codebase.*

Deliverables

The following deliverables are essential for achieving this goal:

1. **Comments Sidebar Implementation**:
    - Develop a functional sidebar panel that displays comments.
    - Ensure seamless integration with the existing codebase.

- Enable users to view, edit, and manage comments directly from the sidebar.

2. **Reviewer Indication**:
    - Color-code comments based on authors (reviewers).
    - Allow customization of colors to accommodate user preferences.

3. **Threaded Comment Display**:
    - Display comments in their natural relation, especially notes with replies.
    - Limit the depth of conversation threads to maintain usability.

4. **Comment Deletion**:
    - Implement the ability to delete individual comments or entire threads.
    - Consider whether deleting all comments should be a separate action accessible from the main menu.

5. **Resolved Status Setting**:
    - Allow marking individual comments or entire threads as resolved.
    - Provide a clear visual indicator for resolved comments.

6. **Filter and Search Functionality**:
    - Design a filter system (or simply a visibility option) within the sidebar to filter by status (resolved/unresolved) or even my authors

7. **Sorting Options**:
    - Implement sorting options for comments:
        - Sort by time (chronologically).
        - Sort by position.

8. **Rich Text Formatting (Optional)**:
    - Although not explicitly part of this project, consider adding basic rich text formatting capabilities to comments (e.g., bold, italic, bullet points). This enhancement would improve user interaction and readability.

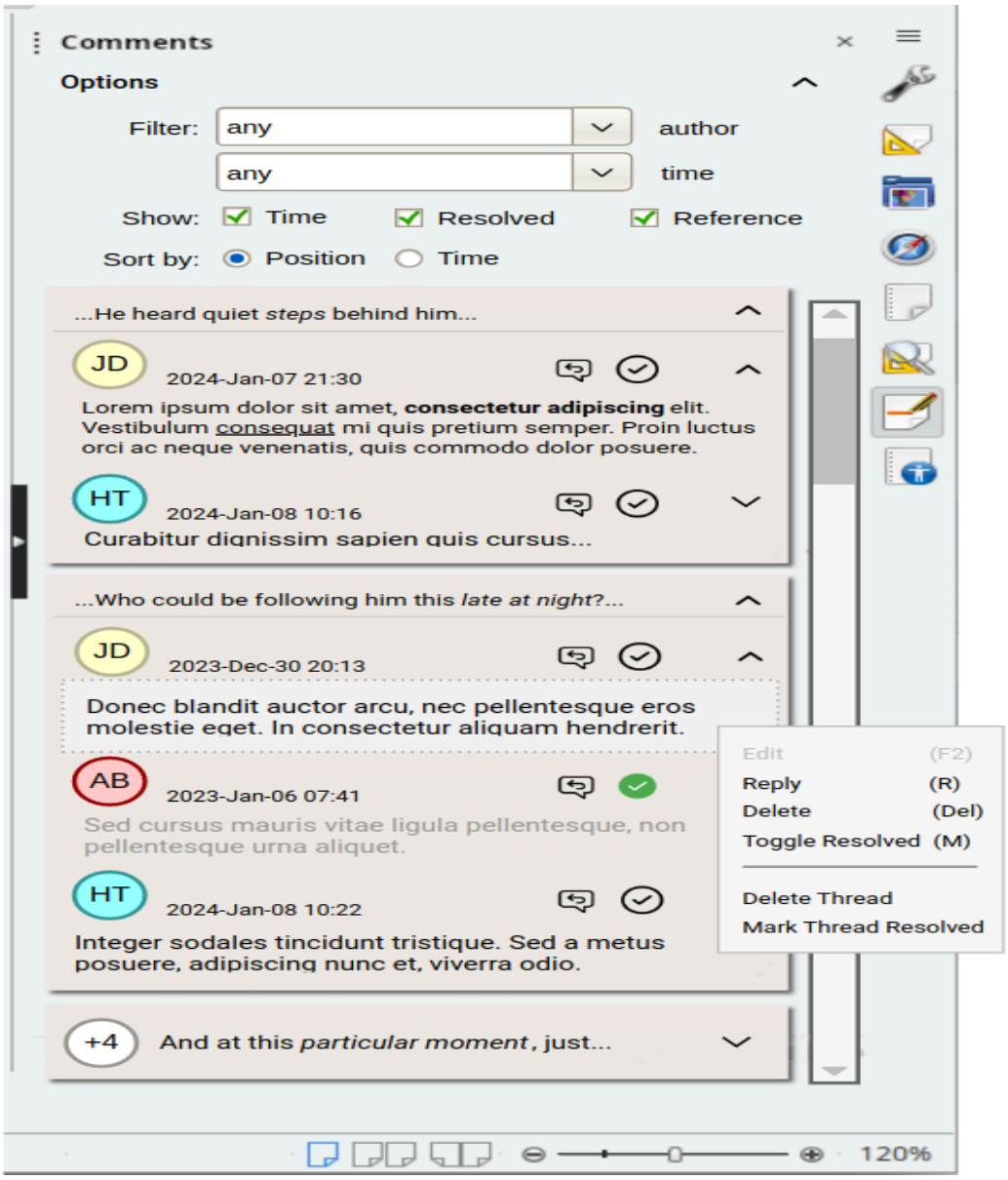
Reference:
https://design.blog.documentfoundation.org/2024/01/12/comments-in-sidebar/

Draft Plan

Note: *All the code snippets provided below are just a glimpse of how an actual implementation may look. It is subject to change.*

1. Create .ui file

Based on the mockup provided, create a .ui file using **Glade**.
I have previously worked with .ui using *Qt Designer* during the last GSoC. [Link]


Comments
Options
Filter:
any
author
any
time
Show:
Time
Resolved
Reference
Sort by:
Position
Time
...He heard quiet steps behind him...
JD
2024-Jan-07 21:30
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat mi quis pretium semper. Proin luctus orci ac neque venenatis, quis commodo dolor posuere.
HT
2024-Jan-08 10:16
Curabitur dignissim sapien quis cursus...
...Who could be following him this late at night?...
JD
2023-Dec-30 20:13
Donec blandit auctor arcu, nec pellentesque eros molestie eget. In consectetur aliquam hendrerit.
AB
2023-Jan-06 07:41
Sed cursus mauris vitae ligula pellentesque, non pellentesque urna aliquet.
HT
2024-Jan-08 10:22
Integer sodales tincidunt tristique. Sed a metus posuere, adipiscing nunc et, viverra odio.
+4
And at this particular moment, just...
Edit (F2)
Reply (R)
Delete (Del)
Toggle Resolved (M)
Delete Thread
Mark Thread Resolved
120%

2. Create a `CommentsPanel` class

```
class CommentsPanel:
    public PanelLayout,
    public ::sfx2::sidebar::ControllerItem::ItemUpdateReceiverInterface
{
public:
    static std::unique_ptr<PanelLayout> Create(
        weld::Widget* pParent,
        SfxBindings* pBindings);

    virtual void NotifyItemUpdate(
        const sal_uInt16 nSId,
        const SfxItemState eState,
        const SfxPoolItem* pState) override;

    virtual void GetControlState(
        const sal_uInt16 /*nSId*/,
        boost::property_tree::ptree& /*rState*/) override {};

    SfxBindings* GetBindings() const { return mpBindings; }
    CommentsPanel(
        weld::Widget* pParent,
        SfxBindings* pBindings);
    virtual ~CommentsPanel() override;
```

3. Create a Comment Widget

- Create a new C++ class that inherits from `VclContainer`
- Implement the constructor and destructor. In the constructor, load the UI from a .ui file (which will be created in the next step)
- If time permits, add Rich Text formatting; details of how this will be implemented will be researched during Community Bonding period

4. Design the Comment Widget UI

Create a new .ui file. This UI should include areas for the comment text, author tag, reply button, toggle resolved button, etc.

5. Populate the Comments Panel

- Add a method to populate the panel with comments. This method should create a new `comment widget` for each comment, set the comment details, and add the widget to the panel.
- To handle changes made to the comments via the main document view:
    - **Listen for changes is the main document view:** Wherever a change is made to a comment in the main document view, an event will be triggered
    - **Update the Comments Panel:** When the event is triggered, the Comments Panel should capture this event and update the corresponding comment.
    - **Reflect changes in the Comments Panel**

6. Add basic functionalities from the current implementation

- **Write/Edit Comment:**

```
void CommentsPanel::SetPostItText()
{
    //If the cursor was visible, then make it visible again after
    //changing text, e.g. fdo#33599
    vcl::Cursor *pCursor = GetOutlinerView()->GetEditView().GetCursor();
    bool bCursorVisible = pCursor && pCursor->IsVisible();

    //If the new text is the same as the old text, keep the same insertion
    //point .e.g. fdo#33599
    mpField = static_cast<SwPostItField*>(mpFormatField->GetField());
    OUString sNewText = mpField->GetPar2();
    bool bTextUnchanged = sNewText == mpOutliner->GetEditEngine().GetText();
    ESelection aOrigSelection(GetOutlinerView()->GetEditView().GetSelection());

    // get text from SwPostItField and insert into our textview
    mpOutliner->SetModifyHdl( Link<LinkParamNone*,void>() );
    mpOutliner->EnableUndo( false );
    if( mpField->GetTextObject() )
        mpOutliner->SetText( *mpField->GetTextObject() );
    else
    {
        mpOutliner->Clear();
        GetOutlinerView()->SetStyleSheet(SwResId(STR_POOLCOLL_COMMENT));
        GetOutlinerView()->InsertText(sNewText);
    }

    mpOutliner->ClearModifyFlag();
    mpOutliner->GetUndoManager().Clear();
    mpOutliner->EnableUndo( true );
    mpOutliner->SetModifyHdl( LINK( this, SwAnnotationWin, ModifyHdl ) );
    if (bTextUnchanged)
        GetOutlinerView()->GetEditView().SetSelection(aOrigSelection);
    if (bCursorVisible)
        GetOutlinerView()->ShowCursor();
    Invalidate();
}
```

- **Delete Comment:**

```
void CommentsPanel::Delete()
{
    collectUIInformation("DELETE",get_id());
    SwWrtShell* pWrtShell = mrView.GetWrtShellPtr();
    if (!(pWrtShell && pWrtShell->GotoField(*mpFormatField)))
        return;

    if ( mrMgr.GetActiveSidebarWin() == this)
    {
        mrMgr.SetActiveSidebarWin(nullptr);
        // if the note is empty, the previous line will send a delete event, but we are already there
        if (mnDeleteEventId)
        {
            Application::RemoveUserEvent(mnDeleteEventId);
            mnDeleteEventId = nullptr;
        }
    }
    // we delete the field directly, the Mgr cleans up the PostIt by listening
    GrabFocusToDocument();
    pWrtShell->ClearMark();
    pWrtShell->DelRight();
}
```

- **Reply to Comment:**

```
void SwAnnotationWin::InitAnswer(OutlinerParaObject const & rText)
{
    // If tiled annotations is off in lok case, skip adding additional reply text.
    if (comphelper::LibreOfficeKit::isActive() && !comphelper::LibreOfficeKit::isTiledAnnotations())
        return;

    //collect our old meta data
    SwAnnotationWin* pWin = mrMgr.GetNextPostIt(KEY_PAGEUP, this);
    if (!pWin)
        return;

    const SvtSysLocale aSysLocale;
    const LocaleDataWrapper& rLocalData = aSysLocale.GetLocaleData();
    SwRewriter aRewriter;
    aRewriter.AddRule(UndoArg1, pWin->GetAuthor());
    const OUString aText = aRewriter.Apply(SwResId(STR_REPLY))
            + " (" + rLocalData.getDate( pWin->GetDate())
            + ", " + rLocalData.getTime( pWin->GetTime(), false)
            + "): \"";
    GetOutlinerView()->InsertText(aText);

    // insert old, selected text or "..."
    // TODO: iterate over all paragraphs, not only first one to find out if it is empty
    if (rText.GetTextObject().HasText(0))
        GetOutlinerView()->GetEditView().InsertText(rText.GetTextObject());
    else
        GetOutlinerView()->InsertText("...");
    GetOutlinerView()->InsertText("\"\n");

    GetOutlinerView()->SetSelection(ESelection(0,0,EE_PARA_ALL,EE_TEXTPOS_ALL));
    SfxItemSet aAnswerSet( mrView.GetDocShell()->GetPool() );
    aAnswerSet.Put(SvxFontHeightItem(200,80,EE_CHAR_FONTHEIGHT));
    aAnswerSet.Put(SvxPostureItem(ITALIC_NORMAL,EE_CHAR_ITALIC));
    GetOutlinerView()->SetAttribs(aAnswerSet);
    GetOutlinerView()-
>SetSelection(ESelection(EE_PARA_MAX_COUNT,EE_TEXTPOS_MAX_COUNT,EE_PARA_MAX_COUNT,EE_TEXTPOS_MAX_COUNT)
);

    //remove all attributes and reset our standard ones
    GetOutlinerView()->GetEditView().RemoveAttribsKeepLanguages(true);
    // lets insert an undo step so the initial text can be easily deleted
    // but do not use UpdateData() directly, would set modified state again and reentrance into Mgr
    mpOutliner->SetModifyHdl( Link<LinkParamNone*,void>() );
    IDocumentUndoRedo & rUndoRedo(
        mrView.GetDocShell()->GetDoc()->GetIDocumentUndoRedo());
    std::unique_ptr<SwField> pOldField;
    if (rUndoRedo.DoesUndo())
    {
        pOldField = mpField->Copy();
    }
    mpField->SetPar2(mpOutliner->GetEditEngine().GetText());
    mpField->SetTextObject(mpOutliner->CreateParaObject());
    if (rUndoRedo.DoesUndo())
    {
        SwTextField *const pTextField = mpFormatField->GetTextField();
        SwPosition aPosition( pTextField->GetTextNode(), pTextField->GetStart() );
        rUndoRedo.AppendUndo(
            std::make_unique<SwUndoFieldFromDoc>(aPosition, *pOldField, *mpField, true));
    }
    mpOutliner->SetModifyHdl( LINK( this, SwAnnotationWin, ModifyHdl ) );
    mpOutliner->ClearModifyFlag();
    mpOutliner->GetUndoManager().Clear();
}
```

7. Implement Connectors

- When a comment is created, also create a comment indicator in the actual document.
- Add a hover listener to the comment indicator
- Create a new widget for comment pop-up if something suitable is not already available

Project Timeline

Pre GSoC(upto May 1):

- Continue understanding the codebase
- Solve some EasyHacks
- Make plans for implementing the proposed functionalities
  Note: I have my End-Semester exams from April 25 to May 3

Community Bonding period(May 1 to 26)

- Continue understanding the codebase
- Make plans for implementing the proposed functionalities
- Engage with my mentor and the community to learn more about the project

Week 1-2:

- Create class for comment panel
- Create a new .ui file for the panel
- Work on Comment widget

Week 3-4

- Add functionalities like write/edit , delete comments

Week 5-6

- Add support for threads for comments widget
- Make the widget expandable(and closeable)

Week 7-8

- Add the feature to search or filter by author and time
- Add the option to sort the comments by time and position

Week 9-10

- Add comment indicator on the reference text
- Add comment pop-up feature

Week 11-12

- Continue working on some pending tasks
- Thoroughly test all the features for bugs
- Work on final GSoC report

Post GSoC

- To increase the scope of my contributions; apart from Writer, I would also like to contribute to Calc and Impress

Other Commitments

My university end semester exams are scheduled from 25th April 2023 to 3rd May 2023. **So I have no other commitments from May 4 to July 15 due to my summer holidays.** After my holidays, I will still be able to manage 30 hours per week, as I usually have on average 4 hours of college per day from Monday to Friday.