

Application Planning

Section 1: Project Description

The application will be a text editor designed for application development. It will not be a full fledged IDE but simple to the point. It won't contain unnecessary features that will get in the way of core developing.

Want to provide a more powerful IDE for elementary OS developers and maintainers.

This document is subject to change at any time. All the features will always be under considration. If anyone wishes to contribute to the development of this program this document should be followed until a proper tracker is put into place.

Section 1.1: What tools will be used to develop the software

- Using GTK+3.0 libraries
- Elementary OS granite library
- Will be written in the Vala programming language
- Designed with Elementary OS in mind (<http://elementaryos.org/docs/human-interface-guidelines>)

Section 1.2: Desired Features

- Syntax highlighting in this order
 - Vala
 - PHP (may not ever include PHP support)
- Autocomplete, intellisense, suggestions
 - Project wide suggestions
 - API suggestions
- Project groups
 - Featured for providing autocomplete. Don't think intellisense should look beyond the project scope for files or methods.

Section 1.3: Versions

Versioning will happen as per the specification of Semantic Versioning 2.0.0.

<http://semver.org/>

Section 1.4: Design Philosophy

Closely resemble stock eOS Freya applications with icons, searches merged into the top window bar. The ultimate goal will be to make the application indistinguishable from stock Elementary OS applications. If developing on Luna, items will not be placable within the top window bar (will have to

be developed on a newer OS, like Ubuntu 14.04 Linux mint 17).

References to the Elementary OS Developer UI Design guide will be made. This is for future reference for myself and for those new to this project.

Section 4: Phase One

Very general part of the development phase. The goal: create a simple text editor. The bench mark for this phase is to be similar to Windows' notepad in functionality or leafpad app.

Section 4.1: Starting the Application

A new instance of the application will show the welcome screen. The welcome screen will help the user make use of the application. In this case 2 options, Opening an existing document or creating a brand new one.

See <http://elementaryos.org/docs/human-interface-guidelines/welcome-screen>

It may be desired to open the application in the exact same state that it was closed in. I.e. If there were files open when the application closed it should attempt to reopen them. That is without compromising speed.

See <http://elementaryos.org/docs/human-interface-guidelines/normal-launch>

There may be an issue while trying to load a document that isn't found. Simply test for it and if it doesn't exist display a notification informing the user the file no longer exists. Information info bar is probably the best to use.

See <http://elementaryos.org/docs/human-interface-guidelines/infobars>


Section 4.2: Opening a file

Should try to open any kind of file. But will include the following file types:

- PHP files (Pending, might put this in if expanding to PHP files)
- Vala files
- Text files
- All files.

All files will be opened in a new tab. If the file is already open, which the tabbed view to that file instead of opening a new instance in a new file.


Define Opening a new file as shortcut Ctrl + O

Using this icon: 

Section 4.3: Make a new file

When making a new file the program will make a new tab view.

Define new file as keyboard shortcut Ctrl + N

Using this icon: 

Section 4.4: Saving a file.

If not saved before save as new file

If saved before override the save.


Each file will always be saved. There will be some periodic save mechanism that will save the file with the proper formatting.

See <http://elementaryos.org/docs/human-interface-guidelines/always-saved>

Save frequency might be a preferences variable that can be manually set by the user.

Define 'save' as keyboard shortcut Ctrl + S

Define 'save as' as keyboard shortcut Ctrl + Shift + S

Using this icon: 

Section 4.5: Exiting the program

Exit program gracefully. If the file has been edited then prompt to save it before exit. Should be done for both File -> Exit and window close button.

Define close as keyboard shortcut Ctrl + Q

Section 4.6: Text actions

Section 4.6.1: Undo and Redo

Provide undo and redo functionality that will allow the user to undo any changes that have been done. Basic Idea: take a snapshot of the document for an iteration and store this in memory. A stack will provide the right data structure to pop off the top to reveal the last incremental memory save. This information should then be pushed onto a 'redo' stack to provide redo functionality in the exact same way.

Both stacks should have a limited number of snapshots to reduce the applications memory footprint. A good potential stacksize could be about 50.

The interval for triggering a snapshot and pushing it onto the undo stack could either be a time, lets say 5 seconds or a pause timer to determine how long its been since the user entered any text.

Should not push onto the undo stack if there are no changes so a quick check must be made to ensure the new snapshot is not the same as the previous one.

The redo stack will only be pushed onto if the undo stack had a snapshot popped off. If any other action takes place, such as text being typed in or pasted into the view the redo stack is immediately destroyed.

Define undo as the keyboard shortcut Ctrl + Z

Define redo as the keyboard shortcut Ctrl + Shift + Z

Using this icon as undo: 

Using this icon as redo: 


Section 4.6.3: Copy, cut and Paste

Copy and paste information from the system clipboard manager. Cut will remove the highlighted text from the textarea. Paste will insert text into the text area if the clipboard contains correctly encoded information such as ASCII.

Define cut as the keyboard shortcut Ctrl + X

Define paste as the keyboard shortcut Ctrl + V

Define copy as the keyboard shortcut Ctrl + C


Using this icon for copy: 

@TODO Needs a paste and cut icon.

Section 4.6.4: Select all

Should select all text for the currently selected tab.

Define select all as shortcut Ctrl + A

Using this icon: 


Section 4.6.5: Search and Replace

There will be two text boxes. First one will be used for finding text in the current tab. The second will be a replacement text.

Find next instance of search parameter, in the currently selected tab.


Replace that instance with the string specified. Replace the currently selected text with the contents that are inside the box. If the find box does not match what is selected then the replace will not work.

Find/Replace button. Perform the action of finding the next instance of what is specified in the find text box and replace it with what is contained in the replace text box.

Using this icon: 

Section 4.6.6: Font

Change font in textarea. This will be a persistent setting, therefore saved as soon as its set and then loaded again by default.

Using this icon: 

Section 4.6.7: Main view

Main view will be a full a fully scrollable text area.

Each file will be contained within its own tab. This most likely will use the granite dynamicnotebook api calls.

Word wrap will be an option. If word wrap is on the text will not extend beyond the borders to reveal scroll bars, despite the text area being completely scrollable.

Line numbers should can be switched on and off in the preferences. The line numbers should always be visible.

Section 4.7: Application Menu

There will be an icon in the top right corner of the application that will be a cog. This will be the application menu.

See <http://elementaryos.org/docs/human-interface-guidelines/appmenu>

Section 4.7.1: Preferences

Preferences dialog box will contain a few options to allow the user to customize their experience with the application.

@TODO Preferences Dialog needs to be expanded and clarified.

Section 4.7.2: About Dialog

Provide basic about menu with program information inside.

Section 5: Phase Two

After phase one is complete, tested and in a fairly stable release the project should be branched and a phase two will begin development. A specific version will not be defined but the program will not receive a full version number until all the phases are complete.

This phase will focus on project management. Ideas for how this phase should be done will probably not come until after phase 1 is complete. (Hard to imagine).