Høgskolen i Østfold

NoteIt
ITF11012-1 21V .NET
2021 Spring

Vladislav Jakovlev
Student number: 152414
Date: 28.05.2021

Høgskolen i Østfold

Table Of Contents

1 Introduction

NoteIt is a Universal Windows Platform (UWP) application made for taking simple notes. It includes both online and offline saving and loading capabilities, as well as Rich Text text manipulation. The application also allows the user to draw sketches using the Ink Draw functionality, also with local saving functionality.

2 Description

There aren't any special requirements to be able run or use the application apart from the standard requirements: internet access and VPN/connection to the university's network. Apart from the CRUD operations towards Donau database NoteIt runs perfectly fine offline as well.

The following paragraphs are a description of every functionality aspect in the NoteIt application, as well as a description of design.

2.1 Navigation

The navigation in NoteIt consists of three main pages: Main page with notes, Sketch page with InkDraw and a Settings page with theme-changing possibilities as well as various information about the app. Every page in this application has navigation caching enabled in every page's onload method, so that the data inputted by the user doesn't disappear while browsing different pages. All of the pages are accessible through the navigation panel, which is in the far left side of the app. The navigation panel has also a "back" button which allows the user to easily browse previously opened pages.

2.2 Writing notes

On the applications' main page you will find a big textbox right in the middle, this is a RichTextEdit box with predefined functionality, as well as several additions. In the far left side above the text box are basic text manipulation options, available through corresponding buttons. These include: text size, text underlining, text bolding and text inclining. As predefined in the class RichTextEdit it also has the standard "undo" and "redo" actions available through the corresponding CTRL Z/Y keyboard shortcuts. The text box also has a custom right click pop-up menu which includes the above mentioned actions. In addition to the main note textbox there is a title box and an id box, both of these are simple TextBoxes.

2.3 CRUD actions towards Donau

The requirements of this project are met by using the NotesController class which is a REST API web controller with methods that represent the CRUD actions, these methods are as follows: PostNotes (POST) which creates an new note in the database, GetNotes (GET) which reads a specified note id from the database, PatchNotes (PUT) which updates either the title or the note text of the specified note id in the database, and DeleteNotes (DELETE) which deletes the specified note id from the database. The other controller is called UserController, and it only uses the GET-method which displays the current username and is only for display purposes; to give a glimpse of the idea of user handling.

Figure 1 - CRUD-actions

All of the below mentioned CRUD method calls check for user input first, if no ID is provided then it returns and displays a Toast Notification informing the user about missing input. Unfortunately POSTing and GETing note text to and from the database does not preserve rtf text formatting. I also found out by reading the documentation that the RichEditBox is not the ideal tool for working with a database in such a manner, it is designed to be used in synergy with .rtf documents,mostly offline. Because of this, getting the text value of a RichTextEdit box was challenging and was handled by instead getting the text selection/highlighted text value by setting the property *"noteEdit.Selection.Text"*. CRUD requests are sent to this address *"http://localhost:65076/api/Notes"* towards Donau in NoteIt and are handled as follows:

CREATE
The POST action is represented as a plus button and is the third icon on the database action bar. Upon clicking the button it calls the PostDatabaseButton_Click method which then calls an async task PostNotes, which either posts the current ID with note title and selected note text to the database or returns a bad request if the note ID is already present in the database.

READ
The GET action is represented as an arrow pointing down and is the first icon on the database action bar. Upon clicking the button it calls the GetDatabaseButton_Click method which then calls an async task GetNotes. If the ID is present in the database, then this method returns a string which then gets parsed to a JSON object, where it further gets deconstructed into variables and gets trimmed of unnecessary symbols. If the ID does not exist in the database then it returns a bad request and informs the user. If the request is successful then it fills the title box and note box with corresponding data relative to the given ID.

UPDATE
The PUT action is represented as a synchronisation icon and is the second icon on the database action bar. Upon clicking the button it calls the SyncDatabaseButton_Click method which then calls an async task PatchNotes, which either updates the current given ID's title or note text to the database or returns a bad request if the ID does not exist. Note text must also be selected in order for it to be sent to the database.

DELETE
The DELETE action is represented as a rubbish bin icon and is the final and fourth icon of the database action bar. Upon clicking the button it calls the DeleteDatabaseButton_Click method which then calls an async task DeleteNotes. If the provided ID exists in the database, then it deletes the entry, returns successfully and clears the title and note boxes. If the ID is not present in the database, then it will return a bad request.

2.4 Sketching

The second page, also called SketchNote, is an InkDraw integration with drawing capabilities. This particular page represents a canvas on which the user is able to draw or "sketch" their notes. In a scenario where one would

take notes and the need to sketch a figure or any sort of other visual object would happen to occur, it is a handy addition. A result of such a scenario is displayed in the example figure 2 below.

InkDraw comes "prepackaged" with many great tools, such as; highlighting, erasing, a selection tool, switching input between a mouse and a touchscreen, a ruler, saving functionality with several file options and zoom tool. All in all this is a very effective addition which synergizes well with the text based notes on the main page. This page also has caching enabled, so that the drawings don't disappear upon opening a different view.

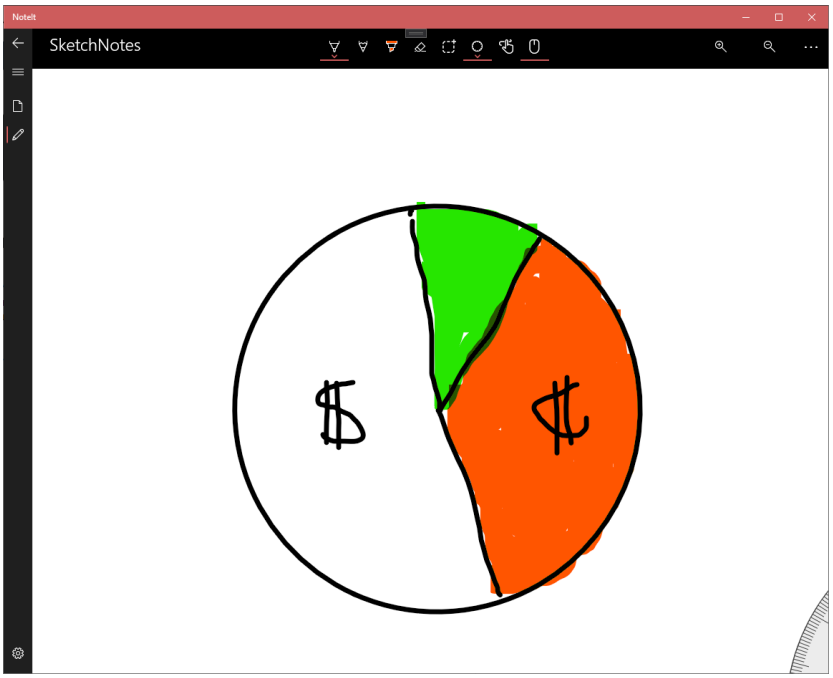


Figure 2 - sketch example

2.5 Loading and saving files offline

As part of the offline functionality the app also has saving files with notetext and opening files. These options are available through the buttons on the far top left side of the main page. This functionality allows the user to save and load files in .rtf format. Text formatting is also preserved and saved upon these actions.

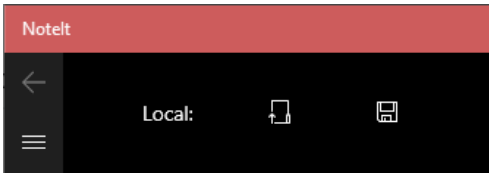


Figure 3 - local file loading and saving.

2.6 Toast Notifications and Tooltips

Messages upon user interaction and error message handling is displayed with so-called Toast Notifications, which is a Singleton AppService added through Windows Template Studio. Other information is displayed in form of ToolTips, for example upon hovering over buttons an explanation pops up, which explains what the specific button performs, as shown in the figure 4 below. Toast notifications are being called through corresponding methods in code and are being displayed upon; successful requests and actions, bad requests with missing or already existing IDs , missing user input and other error messages.

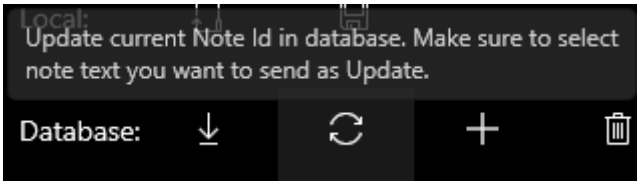


Figure 4 - sync button tooltip

2.7 Design

In order for the application to have a unique look I decided to change up the base colors to Indian Red and by that give NoteIt a slightly minimalistic design. In addition to the predefined themes which come with Windows Template Studio all text boxes and other interactable objects of the app now have a slight red tone to them. To further improve the look of NoteIt, the splash screen, logo and mini icons were changed through the asset folder of the project. This was done with Syncfusion Metro Studio, a tool recommended by our lecturer, in addition to the integrated image modification tool in Visual Studio. The outcome of this is displayed below, figure 5.

Figure 5 - NoteIt logo.

3 Deviations

In order to discuss the results of this project, I would like to compare the initial sketch of NoteIt with the final product. Below, Appendix A.1 represents the initial wireframe of the application, whereas A.2 represents the final product. The major difference between the initial sketch and the current application is the lacking login and registration functionality. Unfortunately, as a result of me having my bachelors project at the same time with this subject, led to time constraints and I was forced to choose not to implement login and registration functionality. Apart from that, both the idea and the product are quite similar and show that the initial idea for the app is in general achieved and implemented as planned. The final product includes additional buttons and text boxes in comparison to the sketch, which were required for the app to function as intended. In addition to the lacking user handling, the avatar was not implemented. Displaying the current user was moved from the main page to the settings page as well. Other than that the final state of NoteIt has all the planned text manipulation tools and a text box with notes, which are the main attraction of the app. In general due to button reorganisation and the chosen color palette, the end result looks better than intended in the initial sketch.

If you take a further look at the initial sketch of NoteIt, there is a *"+ add new"* button which was meant to add additional tabs with notes. Although, because of CRUD and the chosen approach of loading notes from the database by providing a specific ID, new tabs were not necessary, in my opinion.

4 Known issues

There are no known major issues in the application. All of the functionality has been tested prior to delivery and is working as intended. The only minor bug that you may experience is that sometimes the Application Bar buttons get a strange margin offset and get misplaced in the Relative Panel, as shown in figure 6 below. This is simply fixed by restarting the application.

Figure 6 - margin bug

There was also a warning which stated as follows *"Build property AppxBundlePlatforms is not explicitly set and is calculated based on currently building architecture"*. Upon researching and trying a few methods I found a solution where it was recommended[1] to put the following line in the .csproj file: *<AppxBundlePlatforms>x86|x64|arm</AppxBundlePlatforms>*. Upon rebuilding this warning was no longer present.

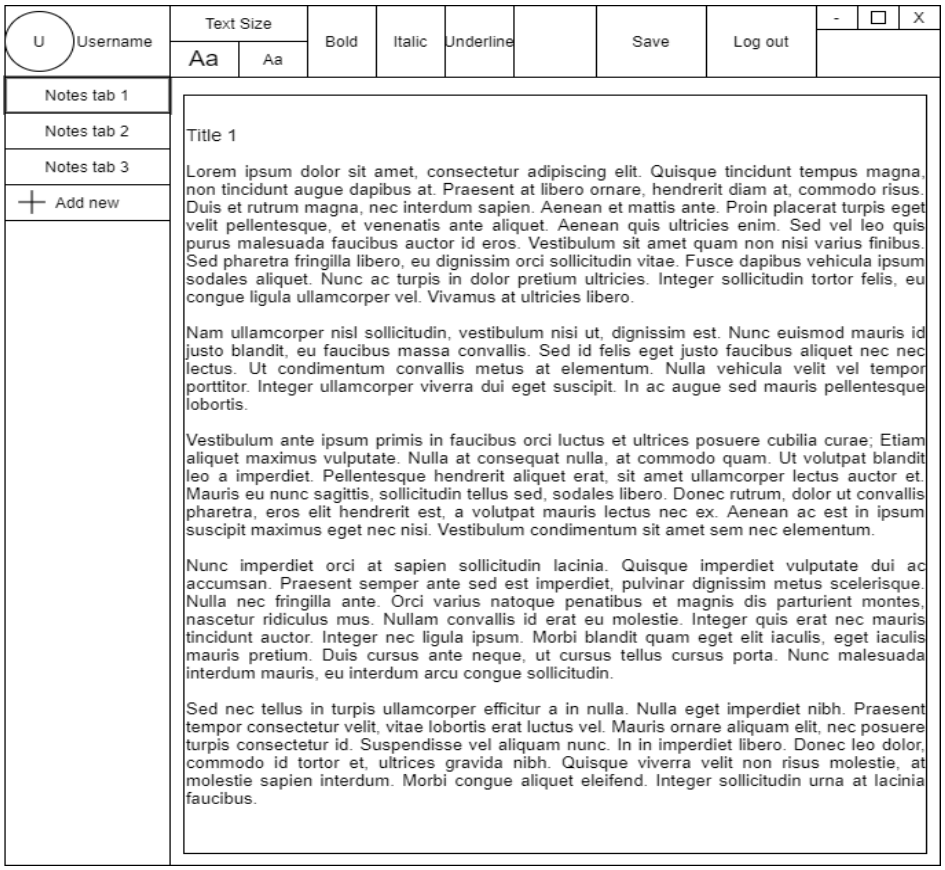
5 Conclusion

In conclusion I would like to state that the general aspects of the project were successfully achieved and that the application is a good representation of my initial idea. Although, ideally there could be login and registration functionality, as well as small quality of life changes, but due to previously mentioned reasons it was not implemented.

[1] Spritehand blog: http://www.spritehand.com/2015/10/appxbundleplatforms-and-windows-uwp-app.html

A. Appendix

A.1. Initial sketch of NoteIt



A.2 Final product