Requirements and Analysis Document for Proton Text

Version: 1.1

Date: May 28, 2017

This version overrides all previous versions.

Ludvig Ekman

eludvig@student.chalmers.se

Institutionen för Informationsteknik

Anton Levholm

levholm@student.chalmers.se

Institutionen för Informationsteknik

Mickaela Södergren

micsod@student.chalmers.se

Institutionen för Informationsteknik

Stina Werme

stinawe@student.chalmers.se

Institutionen för Informationsteknik

Contents

1 Introduction

1.1 Purpose of application

This project intends to create a text editor for IT-students, that can be used for taking notes on lectures. Several features have been planned for the editor, i.e markdown syntax and exporting to pdf, to mention only a few.

1.2 General characteristics of application

The application will be a text editor created for desktop (OSX, Windows and Linux). It will have a ”save”-function, as well as an ”open document”-function. It will also have a number of ways to format the text, to make it easier to structure one's notes and texts.

2 Requirements

A Model, View and Controller-system is the main foundation of this application. For such a system it is required that the model does not get any information directly from the control or view.

2.1 User Interface

The program uses a fixed GUI designed for a standard screen. See appendix for screenshots of the program and its GUI.

2.2 Functional requirements

The user should be able to:

- Write a text
- Save a document
- Open a document
- Format the text to their liking
- Resize the editor window.

2.3 Non-functional requirements

2.3.1 Usability

The application should be easy to use and understand. It should follow the standards of previous text editors, as to not confuse experienced users. Any keyboard-shortcuts should also follow the standards.

2.3.2 Reliability

The application should warn an user if they try to close an unsaved document.

2.3.3 Performance

The text that the user writes should appear in the document when they write it; there should not be any noticable delay of the appearing of the text.

2.3.4 Testability

The application will be tested to see that it correctly opens and saves documents, as well as make sure that the typed text is written out correctly in the document. All the implemented features will be tested as well, to make sure that they work as promised.

2.3.5 Supportability

The application is set to run on all standard desktop operating systems, i.e. Linux, OSX and Windows.

2.3.6 Implementation

The application will run using Java Environment (JRE) that needs to be installed on the host. The application will need to be built through terminal or a Java IDE before it can run.

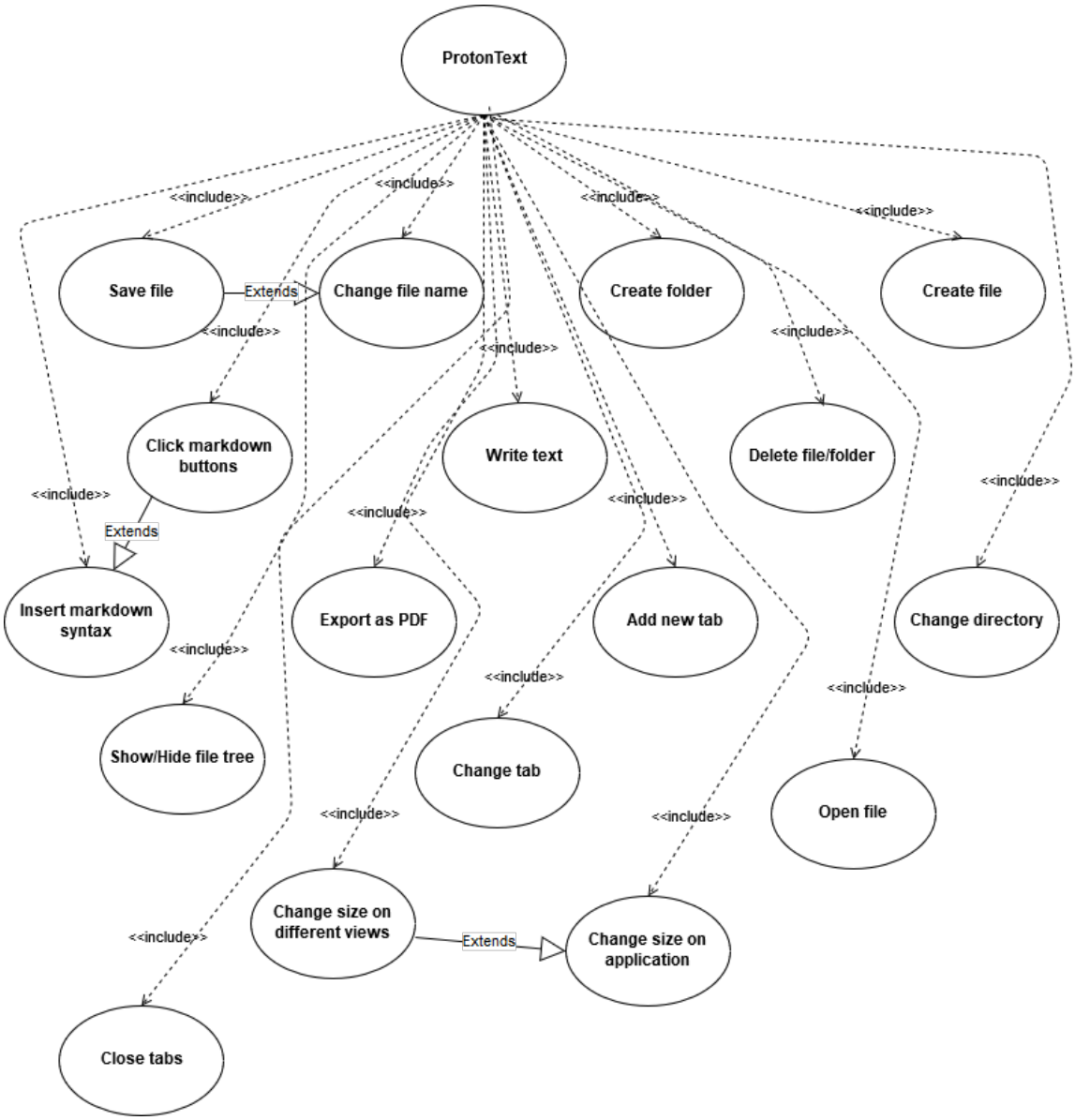
2.3.7 Packaging and installation

The program will be downloaded from git and run using the command 'gradle run'. Tests are run using 'gradle test'. A README.MD file will be avaiable in the text directory with examples of how to use markdown syntax in the application.

2.3.8 Legal

There are no legal issues.

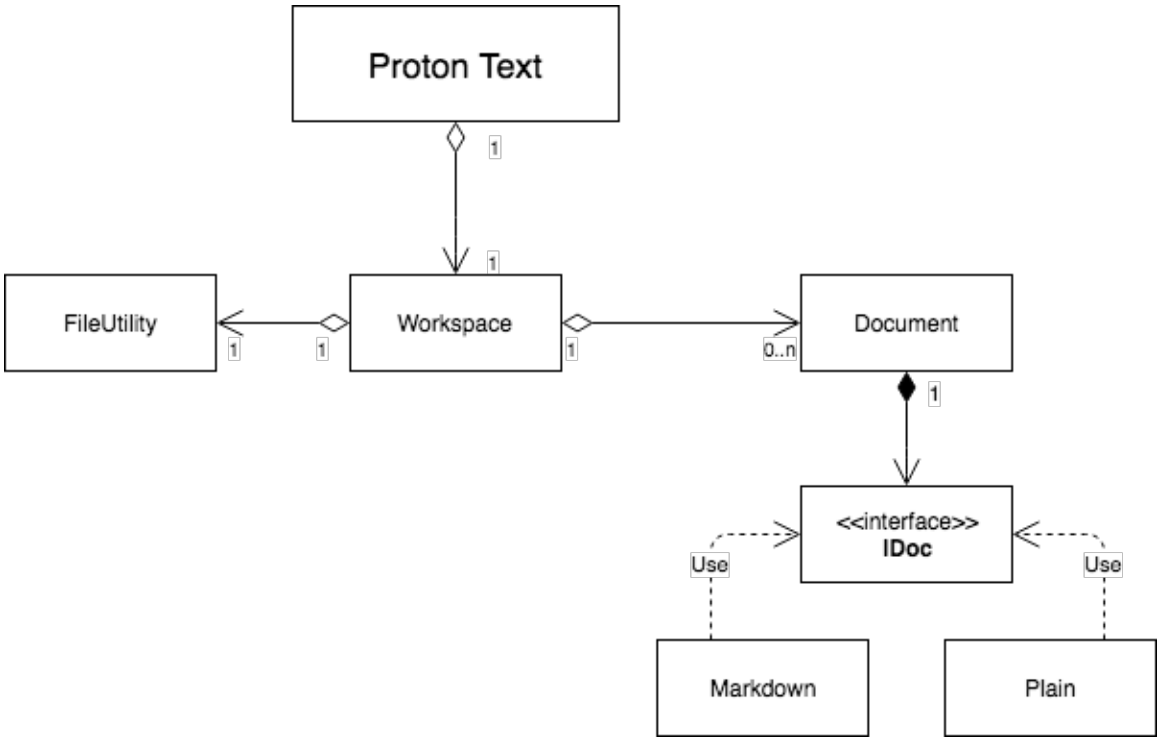
3 Use cases



3.0.1 Use case listing

See appendix.

4 Domain model



Class responsibilities

ProtonText is responsible for creating an instance of the whole application. Workspace has the responsibility to talk to the rest of the model. FileUtility extends Java's class File and has the responsibility to keep track of the file. Document has a responsibility to keep track of the document. Markdown and Plain are different types of document.

Use cases

Proton Text

Innehållsförteckning

UC: Open file

Summary: The user opens a file

Priority: high

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Clicks File -> Open... in menu	
2		Displays file browser
3	Chooses a file	
4		Opens the file in a new tab

Alternate flow

	Actor	System
2.1	Clicks on a file in the file tree	
2.2		Opens the selected file in a new tab

UC: Save file

Summary: The user saves a file

Priority: high

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Clicks ‘Save’ in menu	
		Checks if file is saved before
2		The file is not saved before. Displays pop-up window.
3		User is prompted to enter a name.
4		Saves file under the new name

Alternate flow

	Actor	System
2.1	Clicks ‘Save as...’ in menu	
2.2		Displays pop-up window.
2.3		User is prompted to enter a name.

2.4		Saves file under the new name.

	Actor	System
3.1	Clicks ‘Save’ in menu	
3.2		Check if file is saved before
3.3		File is saved before. Saves file under old name.

UC: Create folder

Summary: The user creates a folder in the file tree

Priority: high

Extends: none.

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Right click on a folder in file tree	
2		Displays a context menu
3	Chooses “New folder”	
4		Creates a new folder in the folder that was clicked on

UC: Create file

Summary: The user creates a new file

Priority: high

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Clicks “File” in the menu	
2		Open menu option
3	Clicks on new	
4		Creates a new file and opens it in a new tab

Alternate flow

Flow 2 Create new file in the file tree

	Actor	System
2.1	Right click on a folder in file tree	
2.2		Displays a context menu
2.3	Choose “New file”	
2.4		Creates a new file in the folder that was clicked on

UC: Delete file/folder

Summary: The user deletes a file/folder

Priority: high

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Right clicks on a file or folder	
2		Display context menu
3	Clicks on “Delete”	
4		Delete the selected file/folder

UC: Write text

Summary: The user writes in a document.

Priority: high

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Presses a symbol key	
2		Checks what key it is
3		Key is symbol key.
4		Key is written in the document.

Alternate flow - delete text

	Actor	System
2.1	Presses backspace	
2.2		Checks what key it is
2.3		Key is backspace.
2.4		Removes character before cursor.

Alternate flow - new line

	Actor	System
3.1	Presses enter	
3.2		Checks what key it is
3.3		Key is enter
3.4		Adds a new row in document. Cursor is moved to the beginning of new row.

UC: Click markdown buttons

Summary: The user clicks a markdown button in the toolbar in order to create text templates for markdown

Priority: low

Extends: Insert markdown syntax

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Clicks a button in toolbar	
2		Checks what button is pressed.
3		Adds a string with the syntax where cursor was.

UC: Insert markdown syntax

Summary: The user writes markdown syntax in the document.

Priority: high

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Writes markdown syntax in document.	
2		Checks document for syntax.
3		Syntax found. Syntax is highlighted.
4		Syntax is converted to HTML and displayed in preview window.

UC: Export as PDF

Summary: Convert translated Markdown to a PDF file

Priority: medium

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Clicks ‘To PDF’-button	
2		Adds a popup window and asks user for a file path
3	Writes file path and clicks ‘Submit’	
4		Converts HTML into a PDF file

UC: Add new tab

Summary: The user adds a new tab

Priority: high

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Clicks File -> New	
2		Creates a new document and adds a tab

UC: Change directory

Summary: Opens a new current directory to work in

Priority: medium

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Clicks ‘File-Change Directory.’ in menu	
2		Displays directory browser
3	Chooses a Directory	
4		Directory is displayed

UC: Change file name

Summary: Change the name of a file

Priority: high

Extends: Save file

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Clicks ‘File->Rename’ in menu	
2		Displays existing file name in an editable window
3	Writes new file name	

Alternate flow

Flow 2 Change name from file tree

	Actor	System
1.1	Double clicks on a file in the file tree	
1.2	Writes new name	
1.3	Presses enter	
1.4		Changes file name

UC: Change tab

Summary: The user changes selected tab

Priority: high

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Clicks on a tab that’s not selected	
2		Selects the clicked tab

Alternate flow

	Actor	System
2.1	Clicks View -> Next tab	
2.2		Selects the tab to the right of the current tab
3.1	Clicks View -> Previous tab	
3.2		Selects the tab to the left of the current tab

UC: Show or hide file tree

Summary: Displays or hides file tree to make use of space

Priority: middle

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Clicks ‘View -> Show/Hide File Tree’ in menu	
2		Shows or hides file tree

UC: Change size on application

Summary: To change the windows size in order to manage space

Priority: medium

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Drags the windows edges	
2		Change size accordingly

UC: Change size on different views

Summary: To change the size a certain field (File tree-view, HTML-view and Web-view) takes in the application

Priority: middle

Extends: change size on application

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Drags the views edges	
2		Change size accordingly

UC: Close tabs

Summary: The user closes the current tab

Priority: high

Extends: none

Includes: none

Participators: The user

Normal flow of events

	Actor	System
1	Clicks the ‘x’ on a tab	
2		Closes the tab

Alternate flow

	Actor	System
2.1	Clicks View -> Close current tab	
2.2		Closes the current tab and selects the previous tab
3.1	Clicks View -> Close all tabs	
3.2		Closes all tabs