Analysis, Design and Software Architecture-E2012 Exam Project: Slice of Pie

Appendix

Kewin Bent Pedersen (kbep@itu.dk)

Christian Martin Henriksen (cmah@itu.dk)

Kasra Tahmasebi Shahrebabak (ktah@itu.dk)

Table of Contents

Appendix 1: Use cases

1.0 Revision Table

Date	Description	Author
04-12-2012	Initial version	All
04-12-2012	Details and diagram added for Use case 1 and 2.[1]	Christian
13-12-2012	Diagrams edited for use case 1 and 2. Details and diagram added for use cases 2-8.	Kewin
16-12-2012	Major overhaul of how use cases are written.	Kewin
16-12-2012	Added use cases 9 to 15, and rewrote others.	Kewin

1.1: Create new document

The user wants to create a new document.

Basic flow

1. The user chooses a place where the document should be and specifies a name for it.
2. The program saves the document to the storage, and updates the list of documents to include the new document.

Preconditions:

- The user must be logged into a client.

1.2: Change the name of a document

The user wants to change the name of a document.

[1] Diagram added in visual studio project

Basic flow

1. The user chooses which document he wants to rename, and specifies the new name for it.
2. The program edits the document and saves the change to the storage, while also adding the change to the document's logs. And then finally updates the list of documents to show the changed name.

Preconditions:

- The user must be logged into a client and be working in a project.
- The user must have access and rights to a document.

1.3: Delete a document

The user wants to delete a document.

Basic flow

1. The user specifies which document he wants to delete.
2. The program deletes the document from the storage, and updates the list of documents is show that the document is no longer there.

Preconditions:

- The user must be logged into a client and be working in a project.
- The user must have access and rights to a document.

1.4: Open a document

The user wants to open a document he has selected in the explorer.

Basic flow

1. The user chooses which document he wants to open.
2. The program reads all info about the document from the storage, and shows it to the user.

Preconditions:

- The user must be logged into a client and be working in a project.
- The user must have access and rights to a document.

1.5: Save a document

The user wants to save a document.

Basic flow

1. The user has changed attached pictures and or changed the documents text.
2. The program saves the document to the storage.

Preconditions:

- The user must be logged into a client and be working in a project.
- The user has opened the document, and made changes to it.

1.6: Create a project

The user wants to create a project.

Basic flow

1. The user specifies he title for his new project.
2. The program saves the new project to the storage, and makes sure the user has access to his new project through the explorer.

Preconditions:

- The user must be logged into the web client.

1.7: Choose a project to work in

The user wants to choose which project he would like to work in.

Basic flow

1. The user logs in with his username.
2. The program presents the user with a list of projects available to him.
3. The user selects which program he wants to work in.
4. The program gets the information about the project from the storage, and presents the project in the explorer.

Preconditions:

- The user must be logged into a client.
- The user must either own or have projects be shared with him.

1.8: Share a project with another user

The user wants to share a project.

Basic flow

1. The user writes the name of the person he wants to share his project with.
2. The program updates the storage to reflect that the project is now shared with the person the user specified.

Preconditions:

- The user must either be the owner of the project, or the project has to be shared with him.
- The user must have opened the project in the web client.

1.9: Insert picture to a document

The user wants to attach a picture to a document.

1. The user chooses a picture from his local file system to add to the document.
2. The program saves the picture to the storage, and updates the document to reflect that the picture is attached to it.

Preconditions:

- The user must be logged into a client.
- The user must have opened the document he wants to insert the picture to.

1.10: View a picture attached to a document

The user wants to see a picture that is attached to a document.

Basic flow

1. The user selects the picture he wants to view.
2. The program shows the picture to the user in a separate window.

Preconditions:

- The user must be logged into a client and be working in a project.
- The user must have opened a document that has a picture attached to it.

1.11: Remove picture attached to a document

The user wants to remove a picture attached to a document.

Basic flow

1. The user chooses which picture in the document he wants to delete.
2. The program will remove the picture from the storage, and update the document view to reflect that the picture is no longer attached to it.

Preconditions:

- The user must be logged into a client and be working in a project.
-
- The user must have opened a document that has a picture attached to it.

1.12: Rename folder

The user wants to rename a folder.

1. The user chooses which document he wants to rename, and specifies the new name for it.
2. The program edits all documents in the folder and underlying folders' paths and saves the changes to the storage, while also adding the change to all impacted document's logs. And then finally updates the list of documents to show the changed name.

Preconditions:

- The user must be logged into a client and be working in a project containing a folder.

1.13: Move object in explorer

The user wants to move either a document or a folder to another folder in the explorer.

Basic flow

1. The user chooses which folder or document should be moved.
2. The program asks the user where the object should be moved to.
3. The user chooses which folder the object should be moved to.

4. The program changes the single documents path if the object was a document, or all documents the folder contained if the object was a folder. The program then saves the changes to the storage, and updates the explorer to reflect the changes.

Preconditions:

- The user must be logged into a client and be working in a project..

1.14: Synchronize local project with server

The user wants to synchronize his local project with the servers version of the project.

Basic flow

1. The user tells the program he wants to synchronize the project he is working in.
2. The program contacts the server and handles all synchronization with the server, and saves the changes made to the local storage, and updates the explorer to reflect changes made to local content.

Preconditions:

- The user must be logged into the offline client and be working in a project.

1.15: Add project from server to offline client.

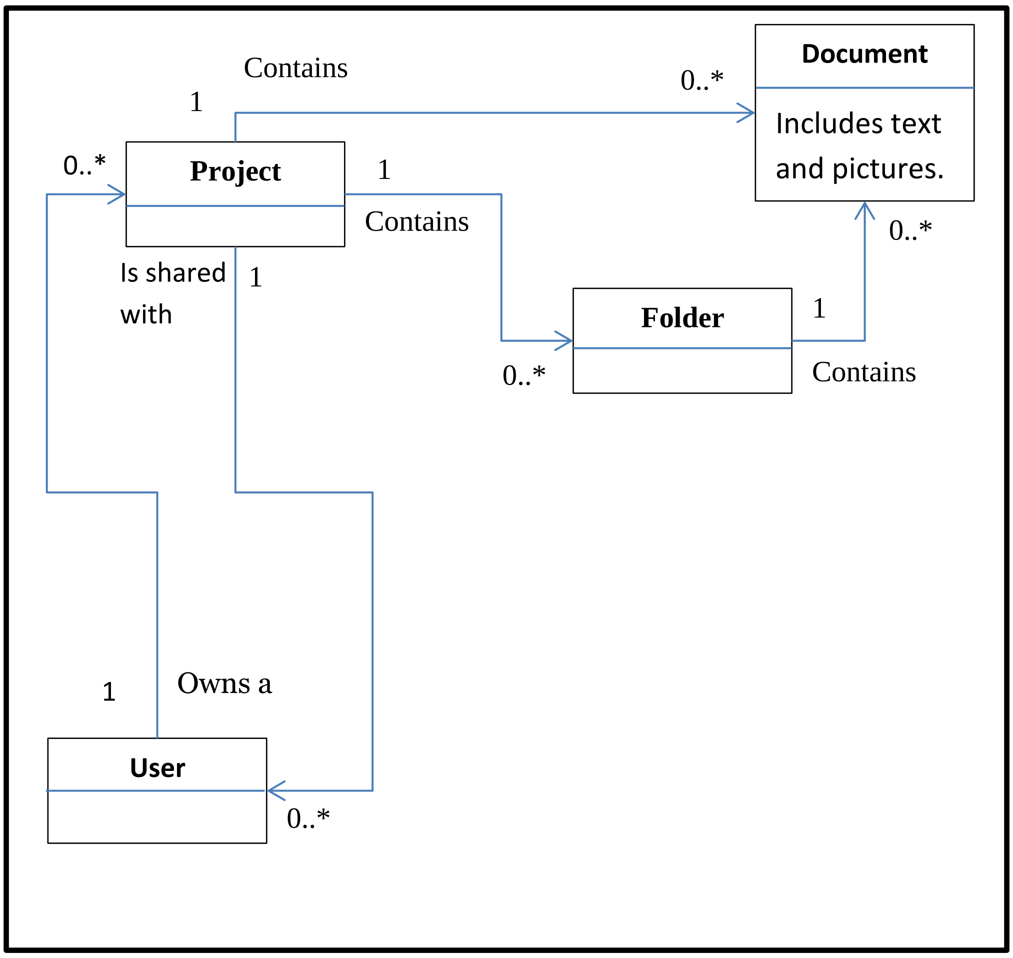
The user wants to add a project that is shared with him on the server to his local client.

1. The user tells the program he would like to add a remote project to his local client.
2. The program contacts the server which supplies it with a list of project titles on the server that the user does not have in his local client, that he either owns or is shared with him.
3. The user chooses which project he wants to add to his local client from the list.
4. The program gets the project and all its content from the server, and saves it to the storage, and then finally updates the list of projects available on the local client to reflect the change.

Preconditions:

- The user must be logged into the offline client.

Appendix 2: Domain Model



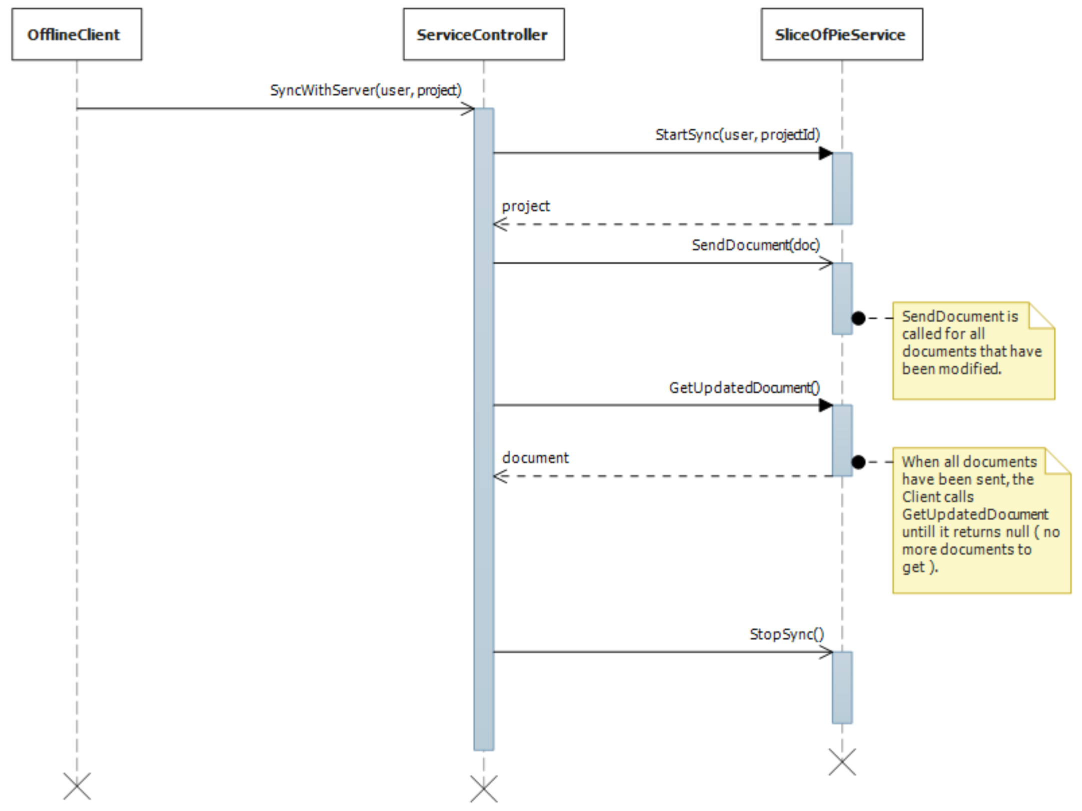
2.0 Revision Table

Date	Description	Author
04-12-2012	Initial version	Kewin & Kasra
16-12-2012	Updated model, added folder and project	Kewin

Appendix 3: System Sequence Diagrams

These diagrams were all created in our visual studio solution, and can also be found there in the project "UML Slice of Pie".

3.1 OfflineClient-Server SSD

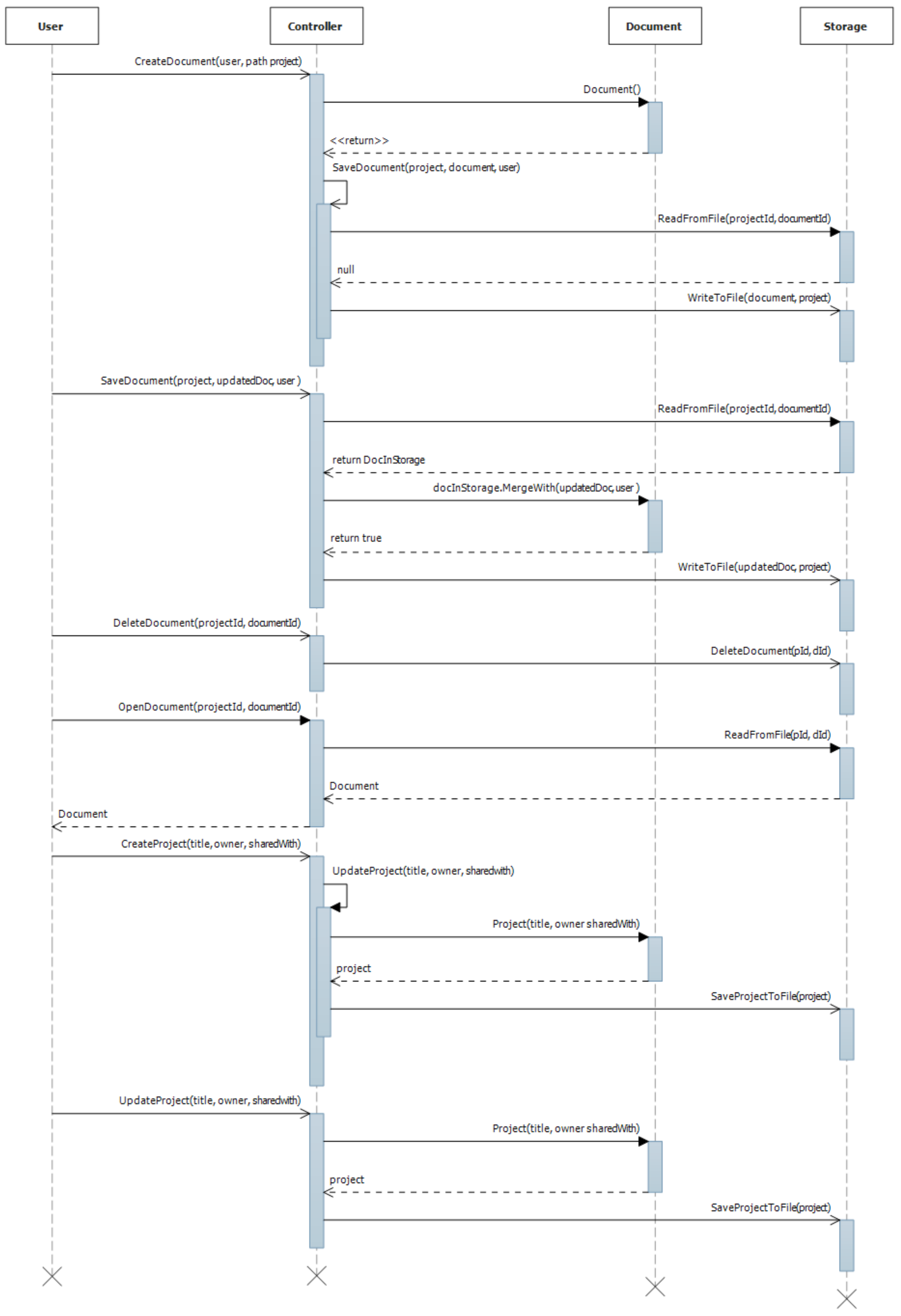

OfflineClient
ServiceController
SliceOfPieService
SyncWithServer(user, project)
StartSync(user, projectId)
project
SendDocument(doc)
SendDocument is called for all documents that have been modified.
GetUpdatedDocument()
document
When all documents have been sent, the Client calls GetUpdatedDocument untill it returns null (no more documents to get).
StopSync()

3.2

OfflineClient-Server SSD diagram Revision Table

Date	Description	Author
16-12-2012	Initial version	Kewin

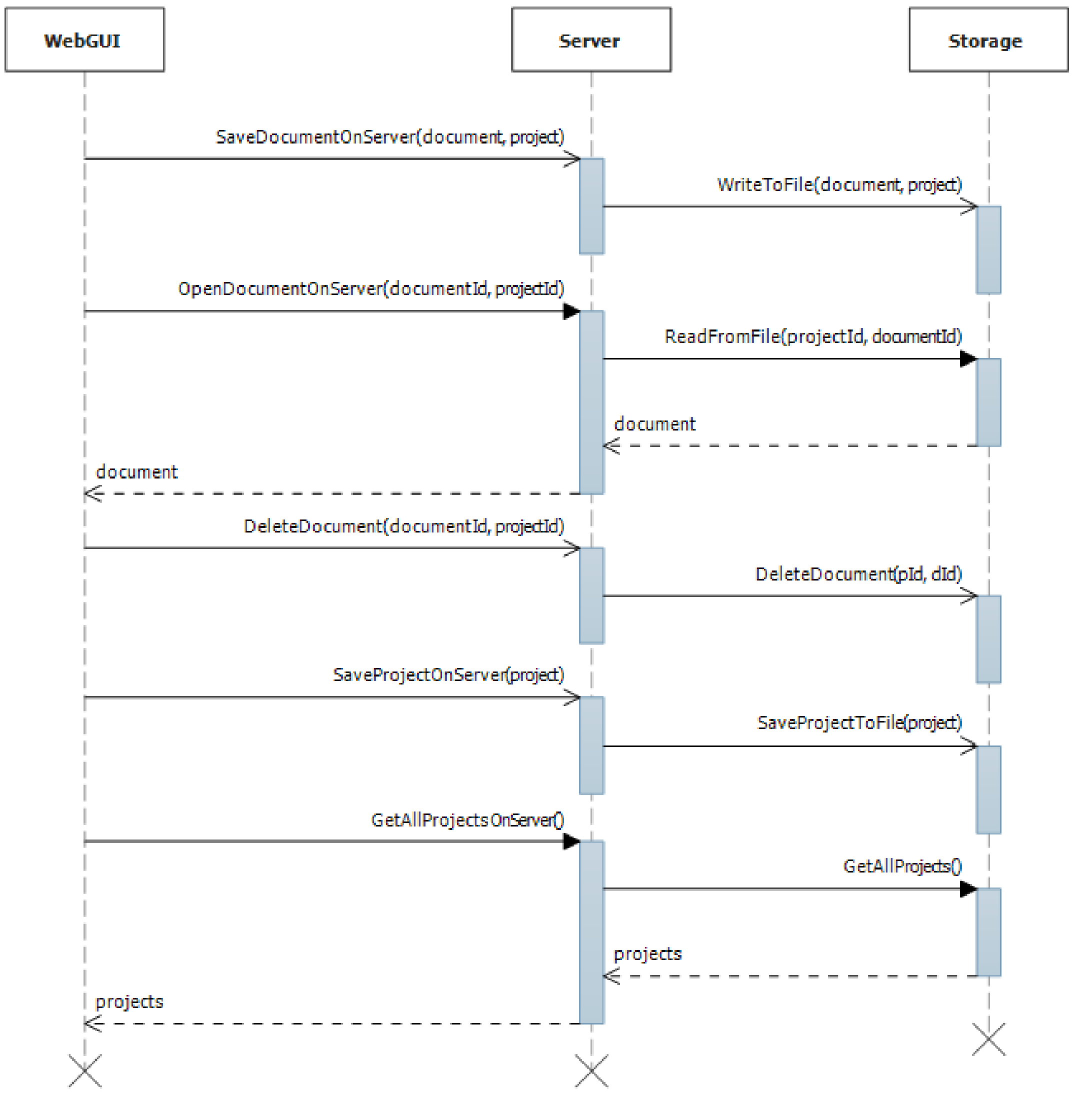
3.3 OfflineClient-Storage SSD


User
Controller
Document
Storage
CreateDocument(user, path project)
Document()
<<return>>
SaveDocument(project, document, user)
ReadFromFile(projectId, documentId)
null
WriteToFile(document, project)
SaveDocument(project, updatedDoc, user)
ReadFromFile(projectId, documentId)
return DocInStorage
docInStorage.MergeWith(updatedDoc, user)
return true
WriteToFile(updatedDoc, project)
DeleteDocument(projectId, documentId)
DeleteDocument(pId, dId)
OpenDocument(projectId, documentId)
ReadFromFile(pId, dId)
Document
Document
CreateProject(title, owner, sharedWith)
UpdateProject(title, owner, sharedwith)
Project(title, owner sharedWith)
project
SaveProjectToFile(project)
UpdateProject(title, owner, sharedwith)
Project(title, owner sharedWith)
project
SaveProjectToFile(project)

3.4 OfflineClient-Storage SSD Revision Table

Date	Description	Author
04-12-2012	Initial version	Christian
14-12-2012	Changed according to new architechture.	Kewin
16-12-2012	Small changes	Kewin

3.5 WebClient-Server SSD


WebGUI
Server
Storage
SaveDocumentOnServer(document, project)
WriteToFile(document, project)
OpenDocumentOnServer(documentId, projectId)
ReadFromFile(projectId, documentId)
document
document
DeleteDocument(documentId, projectId)
DeleteDocument(pId, dId)
SaveProjectOnServer(project)
SaveProjectToFile(project)
GetAllProjectsOnServer()
GetAllProjects()
projects
projects

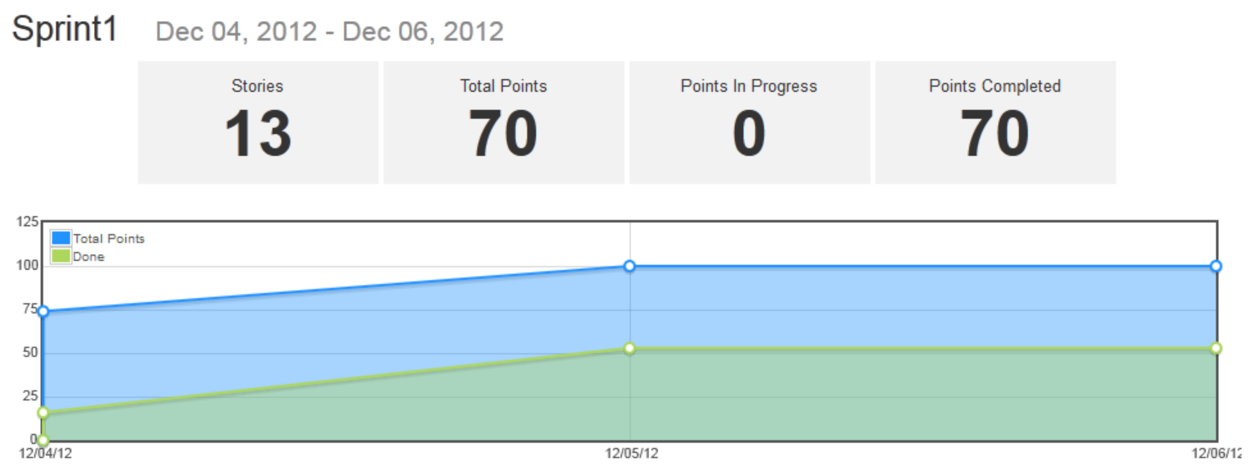
3.6 Web-Server diagram SSD Revision Table

Date	Description	Author
16-12-2012	Initial version	Kewin

Appendix 4: Scrum Burndown charts

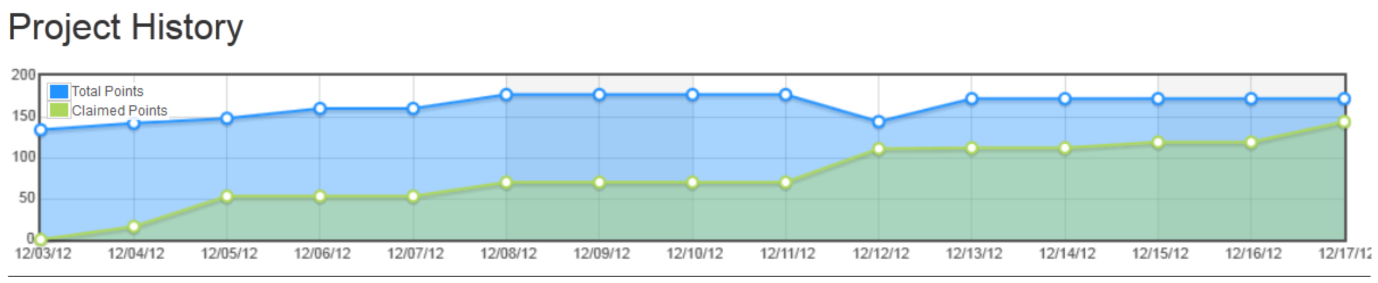
4.1 Example from sprint 1

Example of a burndown-chart from a concluded sprint, in this case we forgot to put the last stories to status: Done before the sprint was done, so that wasn't calculated properly, resulting in the green field not reaching the top.



4.1 Burndown for entire project

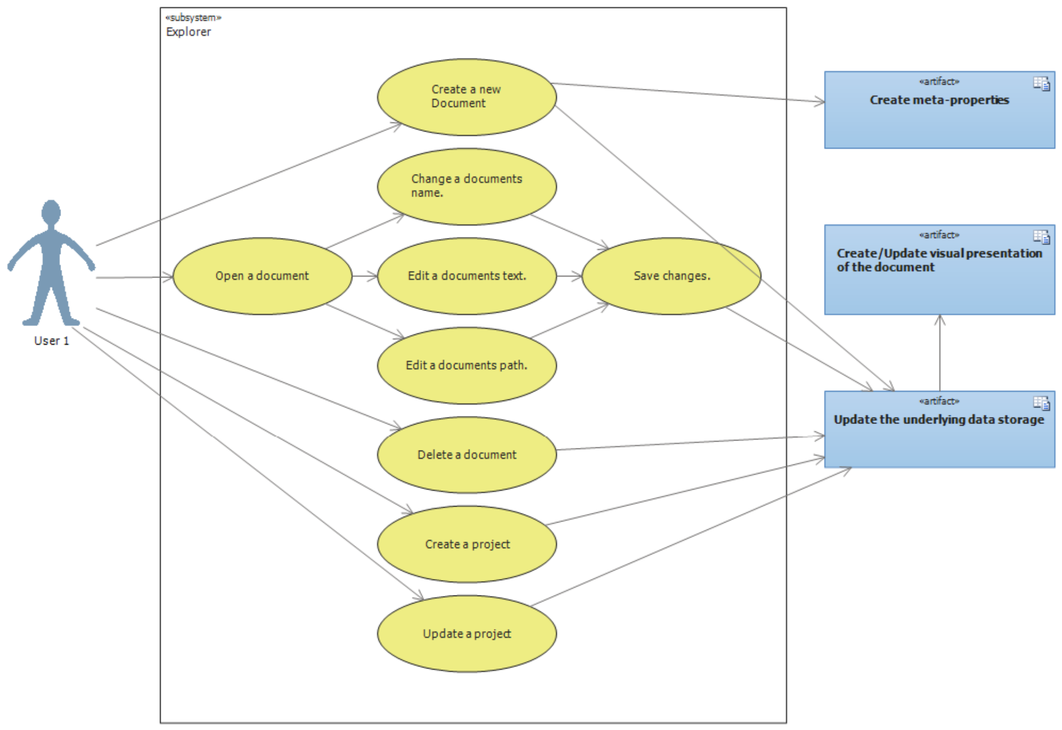
This is our burndown-chart for the entire project duration.



Appendix 5: Use case diagram

This diagrams was created in our visual studio solution, and can also be found there in the project "UML Slice of Pie".

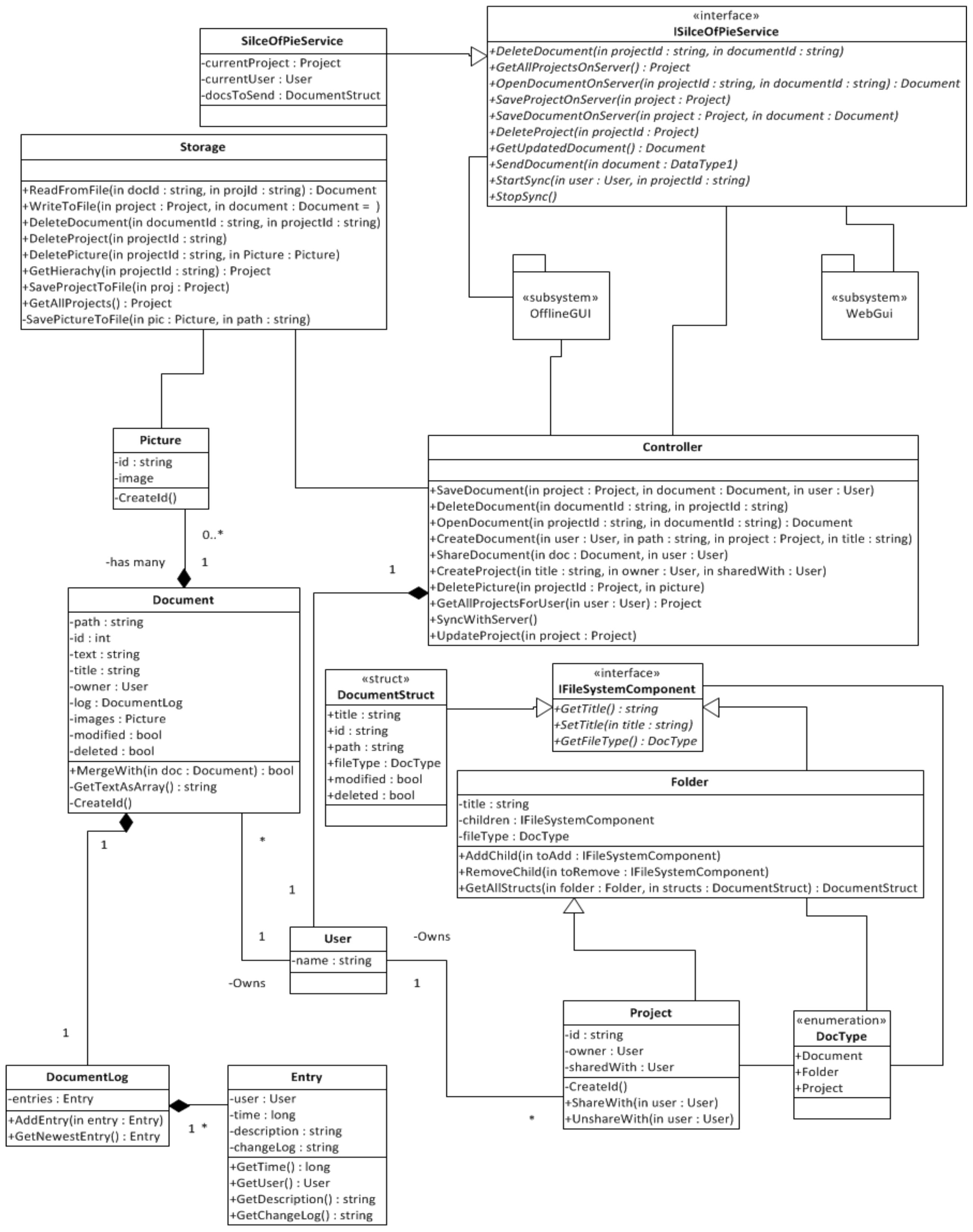
5.1 Use case Model



5.2 Use case diagram Revision Table

Date	Description	Author
04-12-2012	Initial version	Christian
14-12-2012	Added many use cases	Kewin

Appendix 6: Class diagram


SilceOfPieService
-currentProject : Project
-currentUser : User
-docsToSend : DocumentStruct
«interface»
ISilceOfPieService
+DeleteDocument(in projectId : string, in documentId : string)
+GetAllProjectsOnServer() : Project
+OpenDocumentOnServer(in projectId : string, in documentId : string) : Document
+SaveProjectOnServer(in project : Project)
+SaveDocumentOnServer(in project : Project, in document : Document)
+DeleteProject(in projectId : Project)
+GetUpdatedDocument() : Document
+SendDocument(in document : DataType1)
+StartSync(in user : User, in projectId : string)
+StopSync()
Storage
+ReadFromFile(in docId : string, in projId : string) : Document
+WriteToFile(in project : Project, in document : Document =)
+DeleteDocument(in documentId : string, in projectId : string)
+DeleteProject(in projectId : string)
+DeletePicture(in projectId : string, in Picture : Picture)
+GetHierachy(in projectId : string) : Project
+SaveProjectToFile(in proj : Project)
+GetAllProjects() : Project
-SavePictureToFile(in pic : Picture, in path : string)
«subsystem»
OfflineGUI
«subsystem»
WebGui
Picture
-id : string
-image
-CreateId()
0..*
-has many
1
1
Controller
+SaveDocument(in project : Project, in document : Document, in user : User)
+DeleteDocument(in documentId : string, in projectId : string)
+OpenDocument(in projectId : string, in documentId : string) : Document
+CreateDocument(in user : User, in path : string, in project : Project, in title : string)
+ShareDocument(in doc : Document, in user : User)
+CreateProject(in title : string, in owner : User, in sharedWith : User)
+DeletePicture(in projectId : Project, in picture)
+GetAllProjectsForUser(in user : User) : Project
+SyncWithServer()
+UpdateProject(in project : Project)
Document
-path : string
-id : int
-text : string
-title : string
-owner : User
-log : DocumentLog
-images : Picture
-modified : bool
-deleted : bool
+MergeWith(in doc : Document) : bool
-GetTextAsArray() : string
-CreateId()
«struct»
DocumentStruct
+title : string
+id : string
+path : string
+fileType : DocType
+modified : bool
+deleted : bool
«interface»
IFileSystemComponent
+GetTitle() : string
+SetTitle(in title : string)
+GetFileType() : DocType
Folder
-title : string
-children : IFileSystemComponent
-fileType : DocType
+AddChild(in toAdd : IFileSystemComponent)
+RemoveChild(in toRemove : IFileSystemComponent)
+GetAllStructs(in folder : Folder, in structs : DocumentStruct) : DocumentStruct
1
*
1
1
User
-name : string
-Owns
-Owns
1
1
Project
-id : string
-owner : User
-sharedWith : User
-CreateId()
+ShareWith(in user : User)
+UnshareWith(in user : User)
*
«enumeration»
DocType
+Document
+Folder
+Project
1
DocumentLog
-entries : Entry
+AddEntry(in entry : Entry)
+GetNewestEntry() : Entry
1 *
Entry
-user : User
-time : long
-description : string
-changeLog : string
+GetTime() : long
+GetUser() : User
+GetDescription() : string
+GetChangeLog() : string

Appendix 6.0: Class diagram

Date	Description	Author
04-12-2012	Initial version	All
04-12-2012	Small additions to controller and storage	Kasra
08-12-2012	Completely reworked class diagram to conform to new decisions.	Kasra & Kewin
11-12-2012	Reworked class diagram again.	Kewin
16-12-2012	Reworked again	Kewin
17-12-2012	Adjusted to fit final implementation	Kasra

Appendix 7: Sprint reviews and retrospectives

7.1 Sprint 1

7.1.1 Review

We have made a first draft of the offline editor for the client.

At this point we can write documents to the hard drive, but we can't read from it just yet.

7.1.2 Retrospective

What went well?

We were quick and efficient.

We were good at working independently.

We were good at creating a manageable overview of what should be done, which eased the development process.

What could have been better?

Very little progress on the last day of the sprint.

We kept adding new stories, making the sprint too heavy for us to complete within the timeframe.

Some stories were too loosely defined.

7.2 Sprint 2

7.2.1 Review

Our offline client is almost completely done, all that is missing for it to be done is picture support and slightly improved functionality with buttons and interface.

The server is set up and can be run, both with connection and contract. Its behavior needs to be defined for it to be done though.

We have made several changes to our programs design, including adding "Project" and "DocumentStruct", and editing Document and Storage.

Our stories were not very describing of the work that had to be done, and their velocity did not reflect the amount of work that needed to be put in to them.

7.2.2Retrospective

What went well?

We were good at identifying problems and figuring out ways to solve the immediately.

What could have been better?

We were late each day on starting the actual work, and instead put in more hours in the evenings/nights, which made is more tired.

7.3 Sprint 3

7.3.1 Review

Our offline client is completely done, it has a single bug that we are aware of[2] Our web gui is well underway, but not done at this time, it holds a big majority of functionality needed, but not all just yet.

Our server is correctly set up, and just needs to be implemented into the two different types of clients.

We lack some testing on few parts of the system as well.

[2] When deleting pictures attached to a document, if two pictures are removed within the same session, the second picture will not be deleted from the physical storage in 9/10 tests.

All the work that was done in this sprint was VERY poorly documented by stories on ScrumDo, as it was neglected in the wake of immense coding sessions.

7.3.2 Retrospective

What went well?

We were very efficient at implementing new features to our program, and very few errors were encountered in this sprint, making it very efficient.

A lot of coding/work was done.

What could have been better?

We worked way more than the time allotted to this sprint, spending nearly 50 % more time working each day than what was planned.

Appendix 8: Sprint documentation

8.1 Sprint 1

8.1.1 Description of each work day

Day 1

We spent all working hours on creating various design artifacts, including domain model, SSD's and more. At the end of the day we had a good idea of how to start the coding the next day.

Day 2

We started the day reviewing our design artifacts with a small meeting, and then sat down individually and coded separate parts of the program.
At the end of the day a lot of coding had been successfully done, and we a lot of working functionality that had been tested.

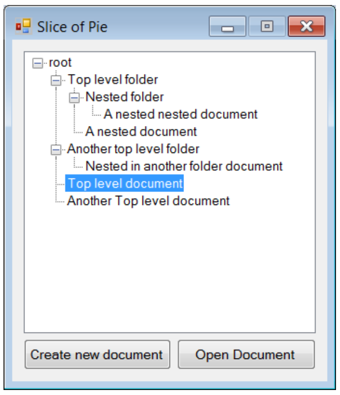
Day 3

On day 3 there was not a lot of work done due to other real life activities coming in the way of the work. Some slight coding, documenting and testing was done, but not at a satisfactory level.

8.1.2 The programs functionality at the end of this sprint

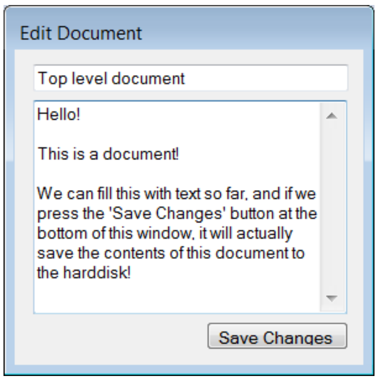
GUI

The user interface is an initial version with basic functionality. The main window shows a tree hierarchy of the folders and documents created, and the user can open one of the documents and start editing. For now it only reads in a set of test data already defined in code, but it will save to the hard disk when the 'Save Changes' button is pressed in the 'Edit Document' window. (The 'Create New Document' button is currently just a placeholder)



Slice of Pie

- root
  - Top level folder
    - Nested folder
      - A nested nested document
    - A nested document
  - Another top level folder
    - Nested in another folder document
  - Top level document
  - Another Top level document



Edit Document

Top level document

Hello!

This is a document!

We can fill this with text so far, and if we press the 'Save Changes' button at the bottom of this window, it will actually save the contents of this document to the harddisk!

Controller

The controller is used as a gateway to the functionality that lies in the storage class, so it is not directly accessed from the GUI.

Document

The document class is at this stage done unless new requirements for it arise. It holds all information related to a document, including a Log which holds entries on changes made to the document.
It also holds functionality to merge the document with a newer version of the same document.

Folder

The folder class holds information relevant to a folder in the file system, and functions to add and remove children, as well as getting all children.

IFileSystemComponent

Our interface that covers over both folders and documents, not much to say other than it at this point serves its functionality.

IFileSystemComponentEnum

An enum describing whether a component is a folder or a document.

Storage

The storage at its current stage can perform most functionality needed from it, but still needs further development.
At this stage it is able to create documents on the file system from a document object without a change log. It can also create folders from folder objects, even if they have a document inside or another folder it will still create a representation of it on the harddrive, but it still needs some optimization before it's done.
We can also read from the harddrive and make new documents, but not folders yet.

8.2 Sprint 2

8.2.1 Description of each work day

Day 1

Day one we started out looking at our diagrams to see what updated had to be made, when that was done, we started coding on our separate parts of the program, extending functionality in all ends.

Day 2

We did a lot of testing on day 2, to make sure everything was working as intended, found several minor issues which were corrected.

We also realized that the structure of our program had to be slightly readjusted to cope with requirements as well as just make more sense in the way data was stored and handled within the program.

Day 3

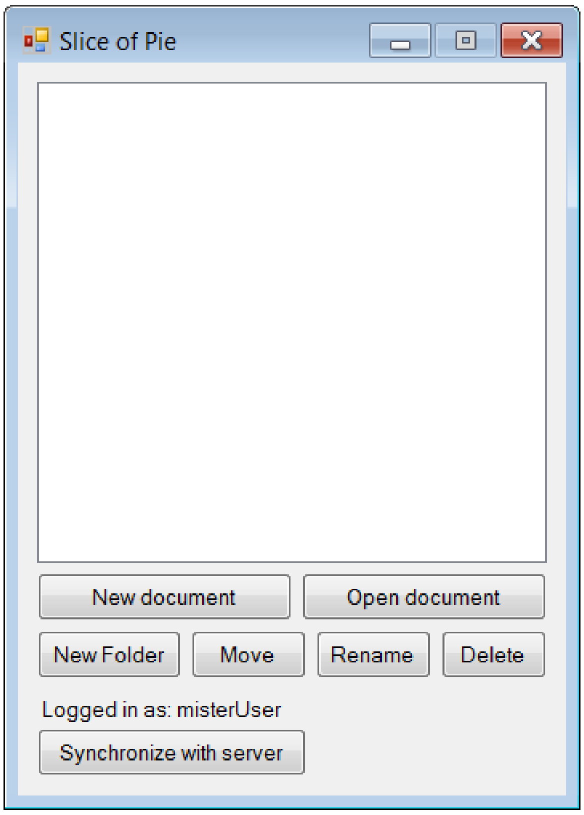
We did a lot of researching on server functionality, and started implementing it as well as being busy with including new readjustments from day 2, the day was very productive and a lot of work was put into the program, resulting in good progress of the program.

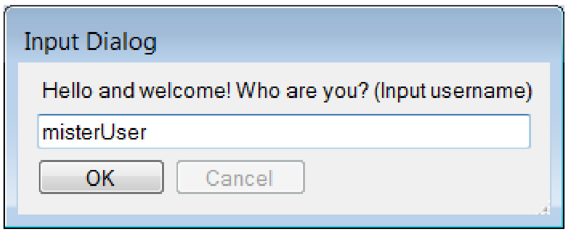
8.2.2 The programs functionality at the end of this sprint

GUI

The User interface is nearing a finished state. All the controls the user needs to fully operate the program has been added. Some of these controls are not functional yet, and only serve as placeholders until the back end is ready to be tied up to these buttons.

This version of the User Interface does not include the recent design change our program underwent, that being the introduction of projects.




Input Dialog

Hello and welcome! Who are you? (Input username)

misterUser

Controller

The Controllers functionality is unchanged since the last sprint, but will be updated to allow interaction with server, as well as the remaining functionality that is required from our readjustment of the structure.

Document

Our Document Class was changed in several ways since last sprint. The biggest change being the MergeWith function, along with several others.
The MergeWith function now takes all changes in the document into account, and generates a fitting changeLog according to the changes.

A document is no longer "shared", as that is handled by our new class "Project".

Folder

The folder class has not been changed since the last sprint.

IFileSystemComponent

The IFileSystemComponent has not been changed since the last sprint.

Doctype

The enum "IFileSystemComponentEnum" was renamed to Doctype, and a new third value was added, which is "Project".

DocumentStruct

This sprint also included the introduction of the DocumentStruct, which is used to store enough information to create a suitable Gui from, with information taken from the storage, without pulling out and handling large amounts of data, of which very little is needed. The documentStruct also inherits from IFileSystemComponent.

Project

The Project class was added to allow users to shared entire "folders", which are named projects. Projects function in almost the same manner as folders, being that they can contain children that are IFileSystemComponents, which is why it extends the folder class.
Projects also include an owner and a list of users it is shared with.

Storage

Our storage class can now save an object of our new Project class on the file system, it creates a folder and some metainformation(a .txt file) which says something about the owner of the project and who it shared with. We can also at this moment, given a project id, create a new project from the files the file system, and return the project object as it should look like for gui representation.

The storage was also updated to read and write documents with their entire log to and from the file system.

Server

A lot of time during this sprint was spent doing reserach and considering the specifics of the implementation of the server. In its current state, the server is runnable and the client application is able to contact the server correctly. The server has no behavior implemented yet. The server is a WCF service hosted in a separate application.

8.3 Sprint 3

8.3.1 Description of each work day

Day 1

We sat down looking at diagrams and models, checking which should be updated, and which are as they should, we also started looking at our webgui.

Day 2

Almost the entire day was devoted to coding, spending more time than allotted, but spending the time well on implementing many features across the program.

Day 3

Day 3 was very much like day 2, we spent way more time than was allotted to the day, but the time was spent doing hardcore coding implementing new features every single hour.

8.3.2 The programs functionality at the end of this sprint

OfflineGui

The "offlineGui" for clients working offline is completely done, all it needs is to be hooked up with the server.

Controller

The controller has been expanded with a lot of functionality to cope with all the new features that were added during the sprint, including pictures, projects and more.

Document

Our document Class has been updated to be able to contain pictures, as well as slight fixes to the way the documents log is written.

Folder

The folder class has not been changed since the last sprint.

IFileSystemComponent

The IFileSystemComponent has not been changed since the last sprint.

Doctype

The enum “IFileSystemComponentEnum” was renamed to Doctype, and a new third value was added, which is “Project”.

DocumentStruct

The folder class not been changed since the last sprint.

Project

The Project class not been changed since the last sprint.

Storage

Our storage has received its final additions, including server functions, improved functionality to functions like reading and writing to files, getting the hierarchy. New functions include adding and deleting pictures.

Picture

The picture class was added to allow documents to hold images as well as their Id.
The picture Class simply has two properties, an Id and a System.Drawing.Bitmap, being the picture object.

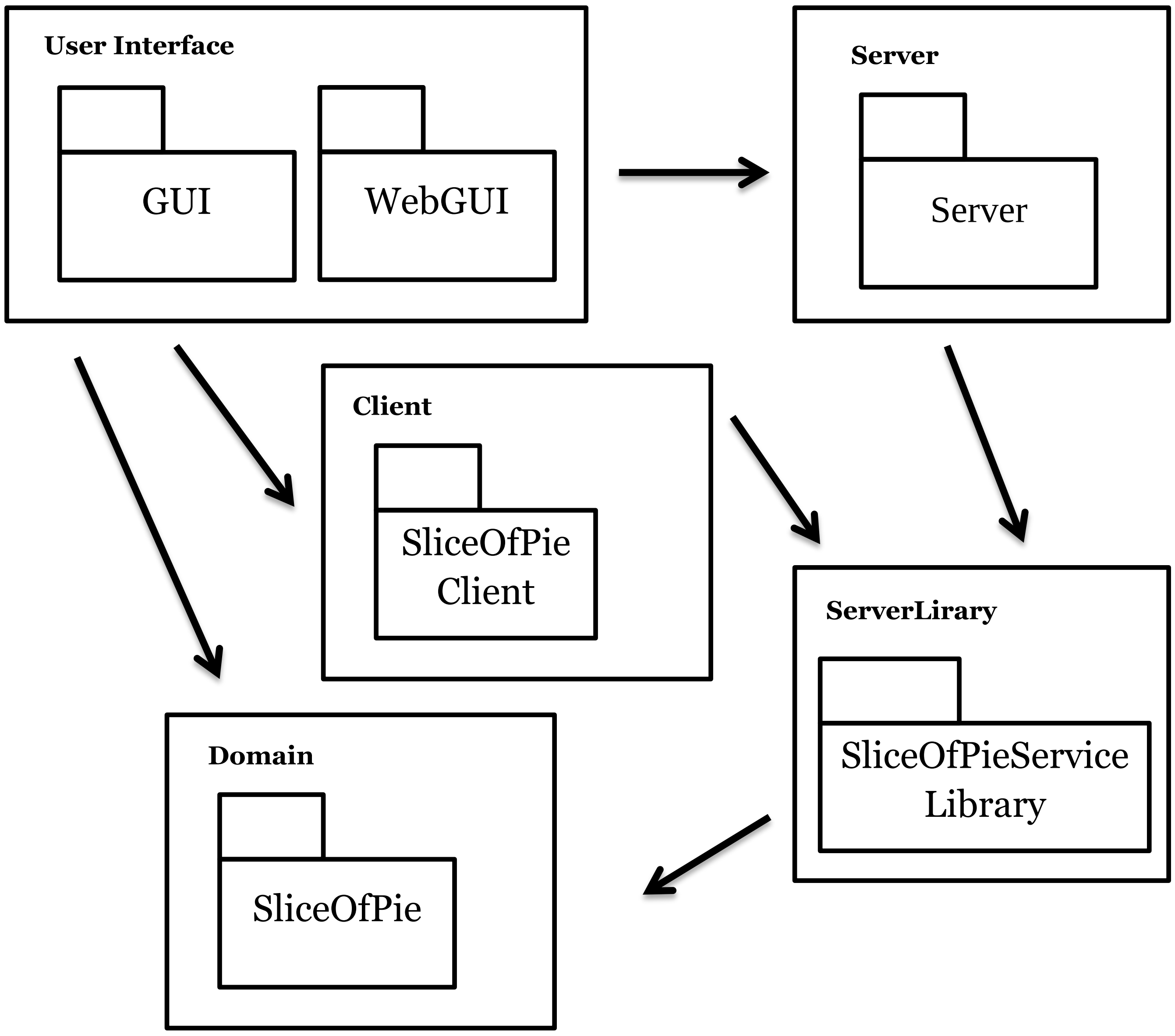
Server

The server has been deployed and is fully operational, all that’s needed is for it to be implemented by our various clients.

WebGui

Our program now also has the online client, being the WebGui. At this stage it holds functionality to open and edit both folders and documents. For it to be completed it still needs picture support and possibly some styling, as well as allowing editing and creation of projects.

Appendix 9: Package diagram

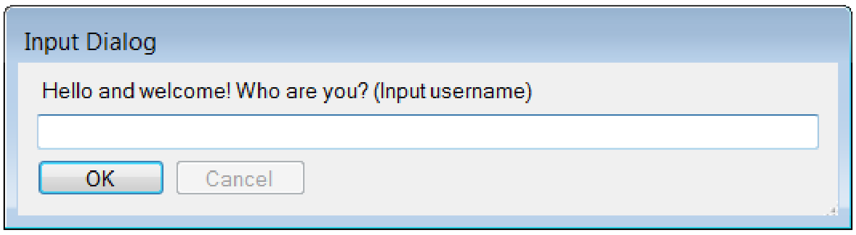


9.0 Revision Table

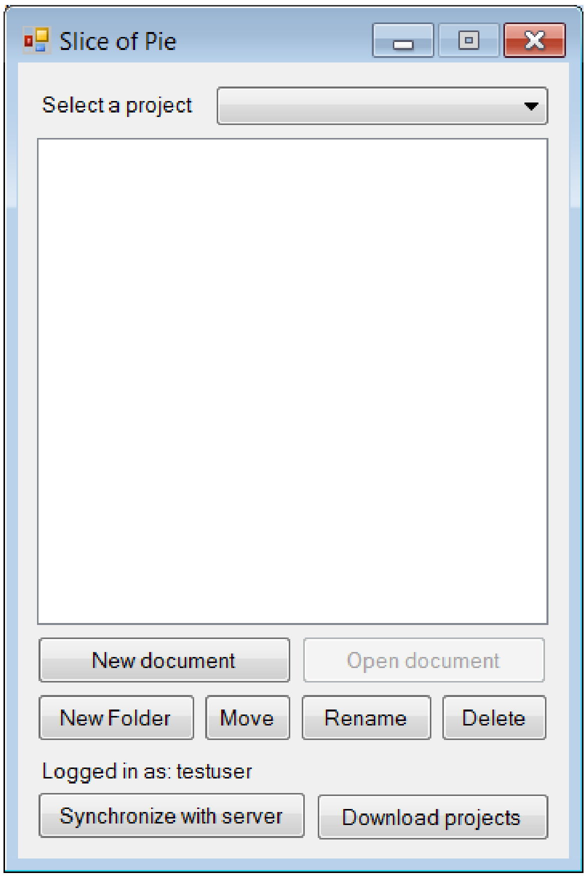
Date	Description	Author
16-12-2012	Initial version	Kewin

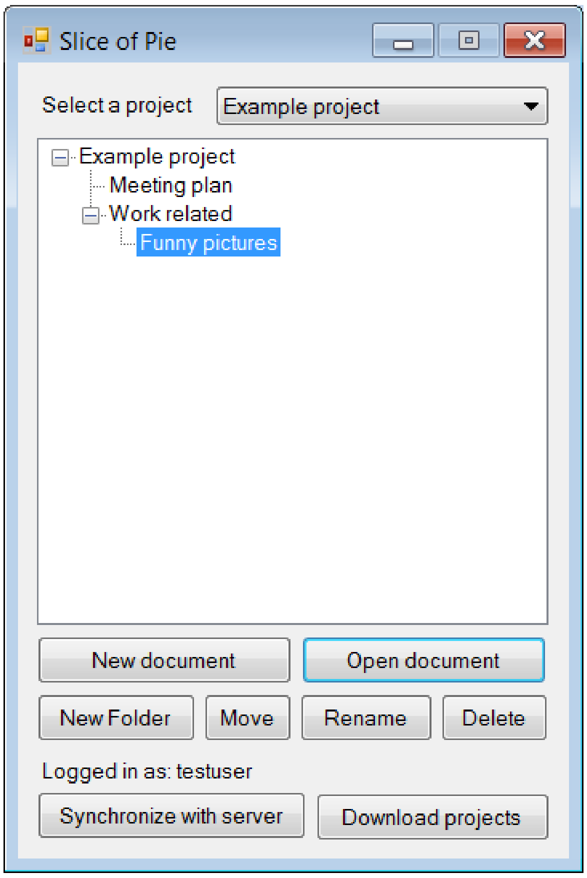
Appendix 10: User manual for offline client

When you open the program, you will be presented with a dialog where you can type in your username.

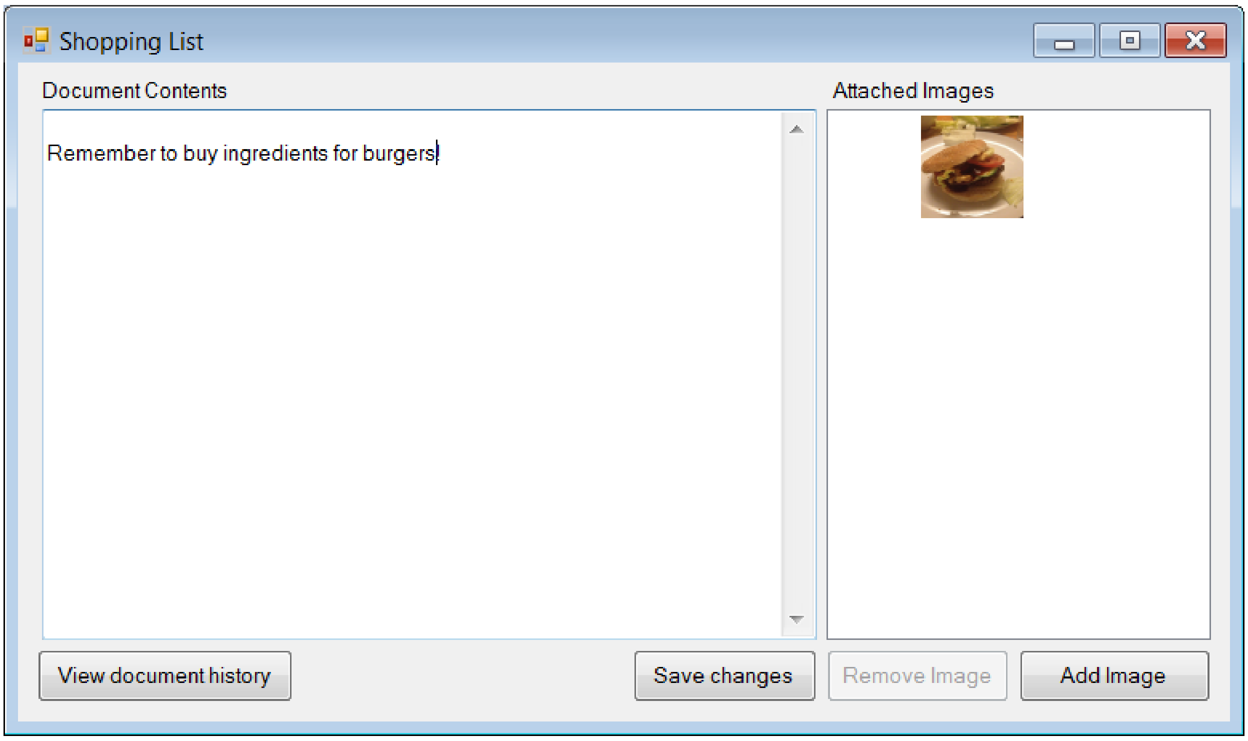


You will then be shown the main window of the offline program. You can select the projects you have downloaded in the dropdown box in the top of the window. When you select a project, the documents and folders of the project will be shown in the window. From here you can manage your existing documents and folders or create new documents and folders.





The “Synchronize with server” button will contact the server and upload your changes to the project to other users, as well as get the changes to the project from the other users. If you need to download a project that was created using the web interface, use the “Download projects” button.



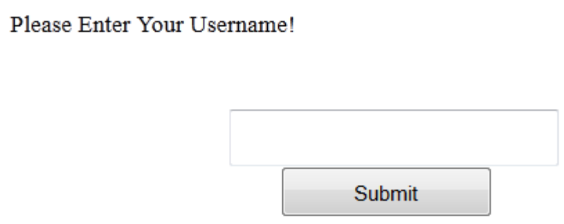
Remember to buy ingredients for burgers

This is the screen that will show when you open a document. You can edit the contents of it, as well as manage the attached pictures. Remember to press “Save changes” after you are done editing the document.

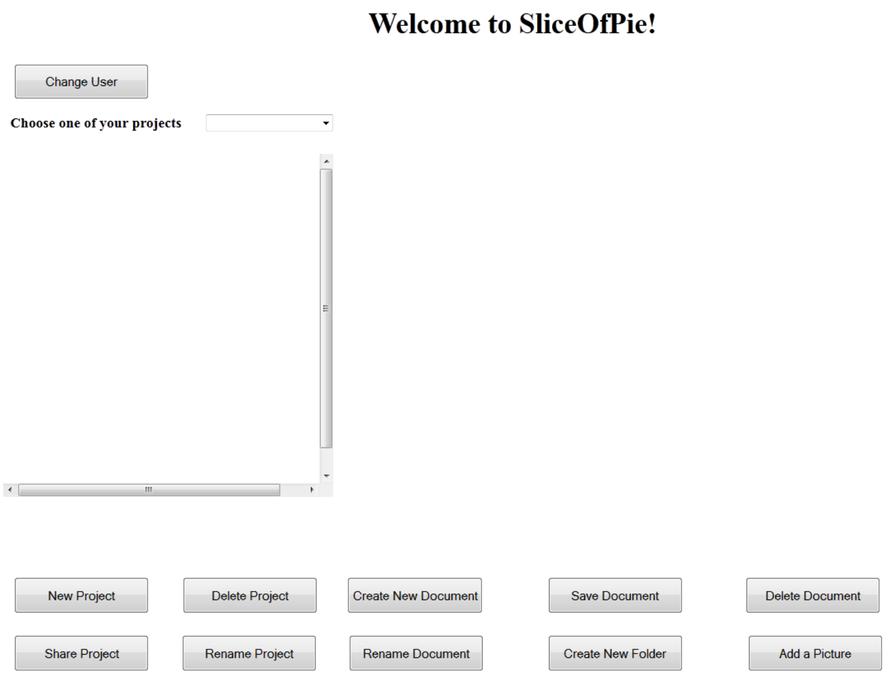
If you need to see what changes that have been made to the document, use the “View document history” button. In here you can see a list of changed all users have made to this document.

Appendix 11: User Manual for web client

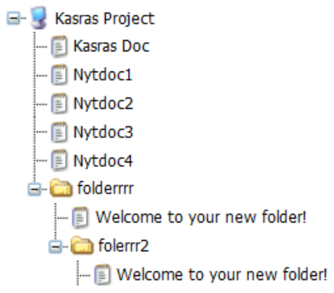
In the start you will as a user be presented with the following scenario:



In which you write your name in and then you are taken to the main interface of the online program:



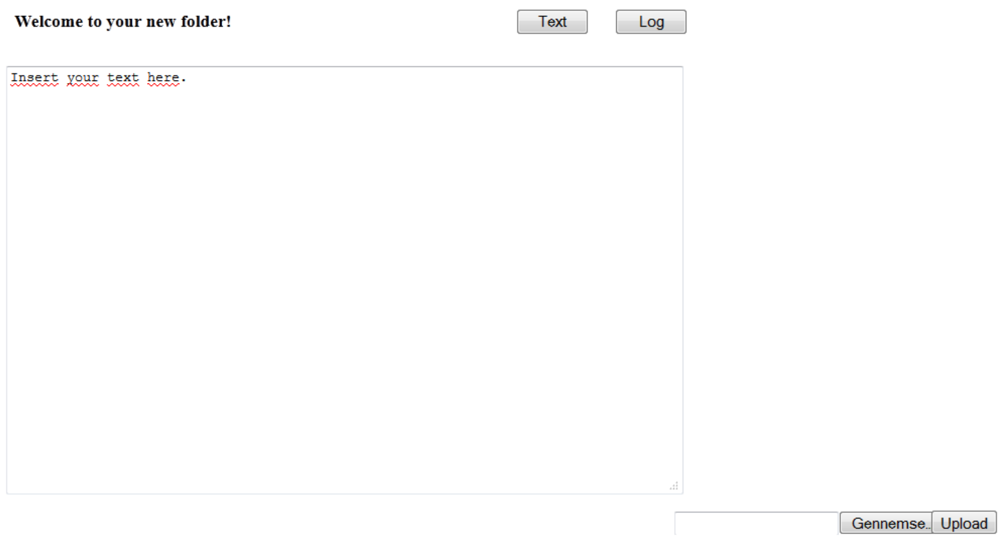
In here you are given a lot of opportunities, in the dropdown you can see all the projects that you own or that is shared with you. If you don't see any projects you can create a new Project and it should pop up instantly when it is created.



When you selected a project you can now choose a document on the dropdown, which will include folders and documents, clicking on a folder will bring up the text editing panel, for you to edit.

Clicking on the Log button will show you the Log of the document.

Clicking any of the 10 buttons in the button, will bring forth some other elements that is associated with the purpose of the button and clicking another button will hide it again, and display the new elements. In this next picture we clicked on a document, so the text window is popped brought forward and we clicked on the add picture button which shows us these panels to carry out the task:



Appendix 12: Project Glossary

12.0 Revision Table

Date	Description	Author
04-12-2012	Initial version	All
04-12-2012	Added: User, Text and Picture. Edited: Document.	Kewin & Kasra
16-12-2012	Added: Folder, Project Implementation Glossary. Edited: Document. Share.	

12.1 User Glossary

Document

The text document the user can insert images and text into.

A document is owned by a single user.

Folder

A document can be arranged into folders, and folders can contain other folders.

Project

A project is the top level folder in which there can be both folder and documents.

A project is owned by a single user, but can be shared with any amount of users.

Explorer system

Like the explorer from windows where you can browse through directories and files.

User

A user of our program, he acts as a client to our server.

A user can own documents, and share these documents with other users.

Owner

Each document has exactly one owner. The owner is a user who created the document, and he can decide who can access that document or not.

Text

Text describes the content of a document that is words and sentences.

Picture/Image

Picture is an image embedded in a document.

Share

A project can be shared with other users in the system. When these users are shared with, they gain access to both read from and write to these documents.

Client

The client is the part of the program that is run on the user machine locally. The user can edit and create documents without having to be online. The user can afterwards decide to go online and synchronize his changes on the documents with the server.

Server

The server is a program that keeps track of all documents created by all users and all changes done to these. Clients can connect to the server and they can exchange changes made on the documents.

Synchronize

The action of bringing the contents of the documents shared between several systems up to date and identical to eachother.

12.2 Implementation glossary

Document

An class that holds information about everything that is in a document the user sees, as well as other fields deciding path, whether or not it has been changed and more.
It also holds functionality to merge with other documents, and other functions relevant to the documents data.

DocumentStruct

A custom Struct created by us, which is a “light” version of our documents, as they contain less information than a complete document, but enough to be used in various parts of our program.

OfflineGui

The client that is not directly connected to the server, but instead allows the user to work with his documents locally.
The OfflineGui is located in the Project called “GUI”.

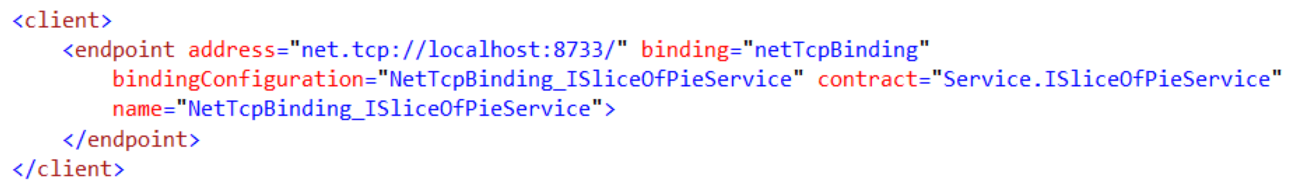
WebGUI

The client that is directly tied to our server, which allows the user to work on his documents online in our ASP.net interface.

Appendix 13: Guide to run server and local client

Enclosed in the final build you will find two directories, Server and GUI. Inside these folders you will find their respective executables, “Server.exe” and “GUI.exe”. You can run these independently of eachother, either on the same machine or on different machines. A test-project is included which can be accessed by using the usernames “testuser”, “friend” or “coworker”.

If you run the server on another computer, be sure the change the IP-address in the configuration file “GUI.exe.config”, which is in the same directory as “GUI.exe”. The attribute to be changed can be found contained within the client element. Instead of “localhost”, type in the IP of the computer running “Server.exe”.

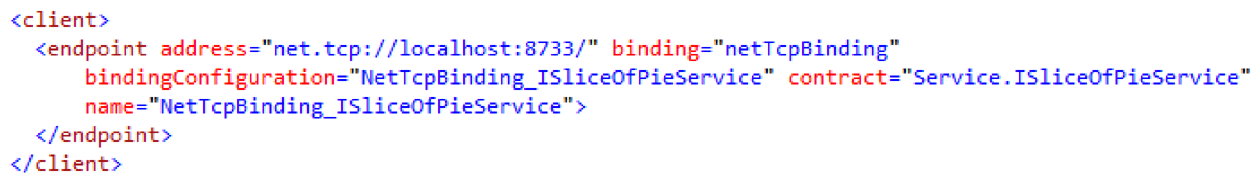


```
<client>
    <endpoint address="net.tcp://localhost:8733/" binding="netTcpBinding"
        bindingConfiguration="NetTcpBinding_ISliceOfPieService" contract="Service.ISliceOfPieService"
        name="NetTcpBinding_ISliceOfPieService">
    </endpoint>
</client>
```

(It might be necessary to have administrative rights when running the server program.)

Appendix 14: Manual for hosting the asp.net website

The asp.net website can be run using a server either on the same pc, or another pc on the local network, in both cases you have to update the web.config file in the visual studio project called WebGUI, in the tag called client:



```
<client>
  <endpoint address="net.tcp://localhost:8733/" binding="netTcpBinding"
      bindingConfiguration="NetTcpBinding_ISliceOfPieService" contract="Service.ISliceOfPieService"
      name="NetTcpBinding_ISliceOfPieService">
  </endpoint>
</client>
```

The address must be set to the IP on which the server is running, in this case its running on the same machine.

14.1 Running the application outside visual studio

If you don't want to run the asp.net website inside visual studio you can easily follow the next steps to run it with IIS.

When that is in order, you have to add the project WebGUI into "C:\inetpub\wwwroot "(assuming IIS is installed), then you have to open IIS and right click "Default Web Site" and choose "add application" then choose the folder called WebGUI you just put into "inetpub/wwwroot", make sure the AppPool is set to .Net 4.0, and the application set to run as x86 mode, finally you can press the "browse project *.80(http)" on the right in the IIS interface. It should then open the application in the browser and you can now add projects and get started.
Also for pictures support, you must rightclick the program in the IIS interface, choose change permissions and set the security for the IIS user to full control. This is not a problem if you run the application in Visual Studio.

Appendix 15: Responsibillities

15.1 Kewin

15.1.1 Main responsible for:

Code:

Document and all inner classes of Document.

Picture.

Controller.

Unit tests.

Artifacts:

Design patterns

Class diagram

Doman Model

System Sequence Diagrams

Class diagram.

Use cases

Report sections:

Interaction diagrams

Report compilation, styling.

Appendix.

Use cases

Domain model

System Sequence Diagrams

Class diagram.

Testing.

Architecture Description/analysis.

15.1.2 Assisted substantially with:

Code:

OfflineGui.

Storage.

Server.

15.2 Kasra

15.2.1 Main responsible for:

Code:

The contents of the following visual studio projects:

- GUI
- The relevant WCF service projects, consisting of:
  - SliceOfPieServiceLibrary
  - SliceOfPieService
  - SliceOfPieClient
  - Server
  - The associated app.config/web.config files for WCF

Artifacts:

- Class Diagram

Report sections:

- Design Patterns
- User manual descriptions
- User manual for offline client

15.2.2 Assisted substantially with:

Code:

The contents of the following visual studio projects:

- SliceOfPie

Artifacts:

- Glossary
- Package Diagram
- Use case descriptions
- GRASP description

Report sections:

- Thing we would have improved had we more time

15.3 Christian

15.3.1 Main responsible for:

Code:

- Everything regarding the WebGUI
    - ASP.Net WebForms/the underlying layout in html
    - C# classes to support the asp.net Controls

- Storage Class

Artifacts:

- GRASP

Report sections:

- Scrum segment
- Manual on setting up asp.net page and connecting to server
- GRASP
- Vision
- Usermanual for WebGui
- Thing we would have improved had we more time
    - WebGUI section
- FURPS+

15.3.2 Assisted substantially with:

Code:

Controller

SliceOfPieServices

Artifacts:

Use Case Model

Sequence Diagram

Glossary

Report sections:

Testing

15.4 All (done as a team)

- Class Diagram
- Conclusion

Product demonstration

- Preparation and demonstration of the product