Contents

1 Vision

1.1 Introduction

Our goal is to make an interactive document sharing system, Slice of Pie, which allows multiple users to share and edit documents both online and offline.

1.2 Problem statement

Sharing and editing documents can be cumbersome.
Sending a document back and forth between multiple users can lead to a lot of errors. Users can overwrite what another user has done, and if they aren't all using the same text editing system this can lead to formatting issues in the document.

1.3 Summary of system features

1. Multiple users must be able to share and edit documents online.
2. Synchronization for offline usage.
3. Merging of documents.
4. History. Which allows the user to see all recent changes made to the document.
5. Documents can be categorized into folders or projects in order to get a better overview when working on a larger project with multiple files.

2 Use cases

1. UC1: Create new document
2. UC2: Edit document
3. UC3: Delete document
4. UC4: Merging documents (resolve conflict)
5. UC5: Offline sync
6. UC6: New folder
7. UC7: New project
8. UC8: Find old version of document
9. UC9: Share Document

10. UC10: Accept invite

11. UC11: Log in

Use case UC1: Create new document
Scope: Slice-of-pie application
Level: User goal
Primary actor: Regular user
Stakeholders and Interests:
- Regular user: Wants to create a new document without any complications.
Preconditions: none.
Postconditions: The document will be created

Basic flow:

1. The user needs a document that can be shared with others.
2. The user chooses a client program (web or desktop client).
3. The user presses the "New Document" button.
4. The client presents a dialogue box.
5. The user types information about the document and press OK.
6. The client sends a request to the server and the document gets stored by the server.
7. The client displays the document for the user.

Extensions:

6. The server is not responding

    (a) The document is saved locally.
    (b) The client requests the server in regular intervals to see if it has recovered.

Special requirements:
- The document should be displayed properly on any devices such as smart phones and tablets.

Use case UC2: Edit document
Scope: Slice-of-pie application
Level: User goal
Primary actor: Regular user
Preconditions: A document must have been created
Postconditions: The document must be edited the way the user desires and the change must be recorded by the server too

Basic flow:

1. The user opens a document.
2. The user edits the content of the document.
3. The client sends a request to the server.
4. The document is changed on the server.

Extensions:

3. The server is not responding

    (a) The document is saved locally.
    (b) The client requests the server in regular intervals to see if it has recovered.

Special requirements:
- The document should be displayed properly on any devices such as smart phones and tablets.

Use case UC3: Delete document
Scope: Slice-of-pie application
Level: User goal
Primary actor: Regular user
Preconditions: A document must have been created. The user trying to delete a document must have sufficient rights to delete selected document
Postconditions: The document is now deleted from the users local repository, and will be deleted when user commits

Basic flow:

1. The user selects a document
2. The user tries to delete the document 3. The client sends a request to the server
3. Server checks for authentication
4. Document is deleted

Extensions:

3. The server is not responding
    - The document is saved locally
    - The client requests the server in regular intervals to see if it has recovered
4. The user lacks the right to delete the selected document
    - User is told that he has not got the rights to delete selected document

Special requirements:

Use case UC4: Merging documents (Resolve conflict)
Scope:
Level:
Primary actor:
Preconditions:
Postconditions:

Basic flow:
Extensions:
Special requirements:

Use case UC5: Offline synchronization
Scope: Slice-of-pie application
Level: User goal
Primary actor: Regular user
Preconditions: User must be logged in. Client must be connected to the server.
Postconditions: The users offline repository will be synchronized with his online repository

Basic flow:

1. The user selects offline synchronization
2. The client sends a request to the server.
3. The server copies the user's online repository and sends it to the client
4. The client sends verification upon receiving the documents

Extensions:

2. The server is not responding
    - The user gets an error message (no connection)
4. The server is not responding
    - The client tries to re-establish a connection to the server at regular intervals, and then sends the verification.

Special requirements:

Use case UC6: Create new folder
Scope: Slice-of-pie application
Level: User goal
Primary actor: Regular user
Preconditions: none.
Postconditions: The folder will have been created

Basic flow:

1. User clicks on "create new folder" button
2. User fills in valid folder information and clicks "OK"
3. Client sends a request to server and server stores the folder
4. Client displays the new folder

Extensions:

2. Information is not valid
   (a) User gets an error message describing what information is invalid
3. Server does not respond
   (a) folder is stored locally
   (b) The client requests the server in regular intervals to see if it has recovered, then resends the request to create a folder.

Special requirements:

Use case UC7: Create new project
Scope: Slice-of-pie application
Level: User goal
Primary actor: Regular user
Preconditions: none.
Postconditions: a Project will have been created

Basic flow:

1. User clicks on "Create New Project" button
2. User fills in valid Project information and clicks "OK"
3. Client sends a request to server and server stores the Project
4. Client displays the new folder

Extensions:

2. Information is not valid
   (a) User gets an error message describing what information is invalid
3. Server does not respond
   (a) Project is stored locally
   (b) The client requests the server in regular intervals to see if it has recovered, then resends the request to create a Project.

Special requirements:

Use case UC8: View old version of document
Scope:
Level:
Primary actor:
Preconditions:
Postconditions:

Basic flow:
Extensions:
Special requirements:

Use case UC11: Log in
Scope: Slice of Pie application
Level: User goal
Primary actor: Regular user
Preconditions: User must be registered in the system *(see "extensions *2").*
Postconditions: User is logged into the system.

Basic flow:

1. When the user starts the system (web client or stand-alone client) he's prompted with a login screen.
2. User must log in by typing username and password.
3. System checks if information matches what is stored in the system.
4. User is logged into the system.

Extensions:

*2. User is not registered in the system.
   (a) User is prompted with a sign up screen (web-client only).
   (b) User fills out required information for signing up.

*4. Username and password are rejected.
   (a) User gets an error message on the screen saying that the username and password don't match the system.
   (b) User gets prompted back to the login screen.

Special requirements:

3 Glossary

- **Response Time:** The time it takes for the system to respond to a request from the user.

4 Supplementary Specification (FURPS+)

- Functionality
  - The system must be able to store documents.

  - The system must support multiple users.

- Usability
  - The system must have a clean and simple user interface.
  - Users must be able to use the application without training.
  - System features must be simple and straight forward. No complicated work flows.
  - The overall system must be easy to use. 8 out of 10 users must be able to use the system without any training.

- Reliability
  - The system must be able to restore it's state in case of a server failure.

- Performance
  - The system must be fast and responsive. A request to the system can't take longer than 3 seconds. (not counting the external internet connection)

- Supportability

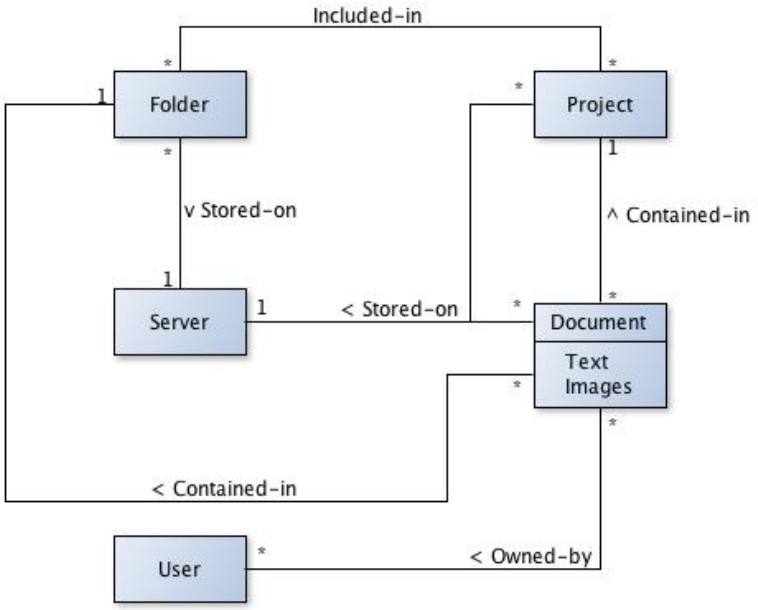
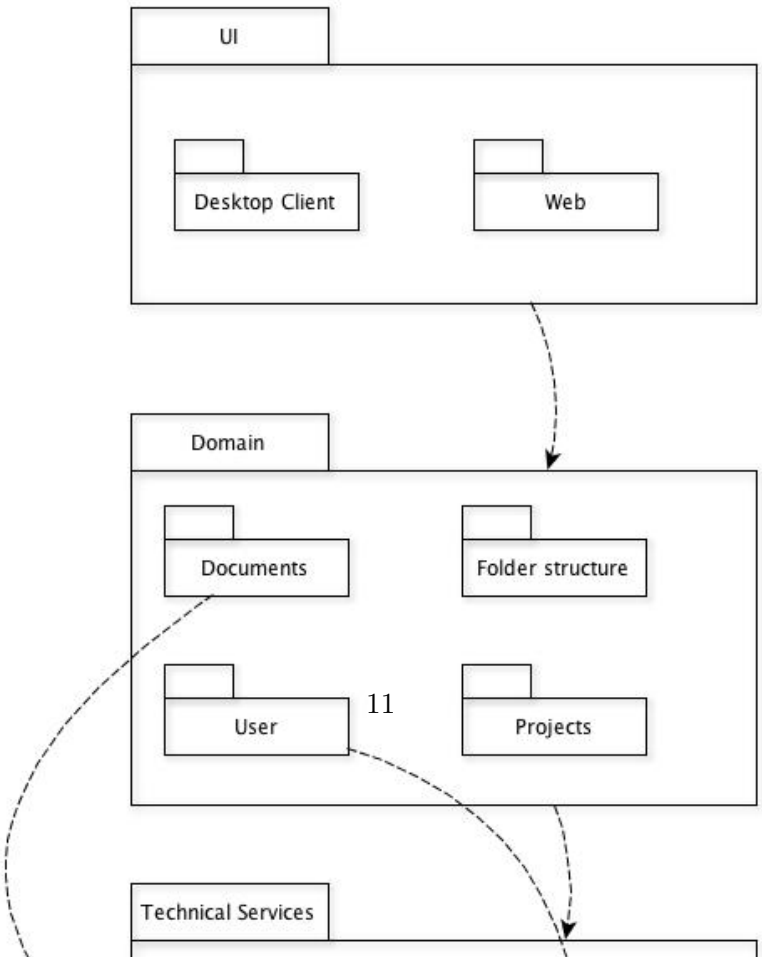


Figure 1: Domain Model

5 Domain Model

6 Logical Architecture



7 System Sequence Diagram

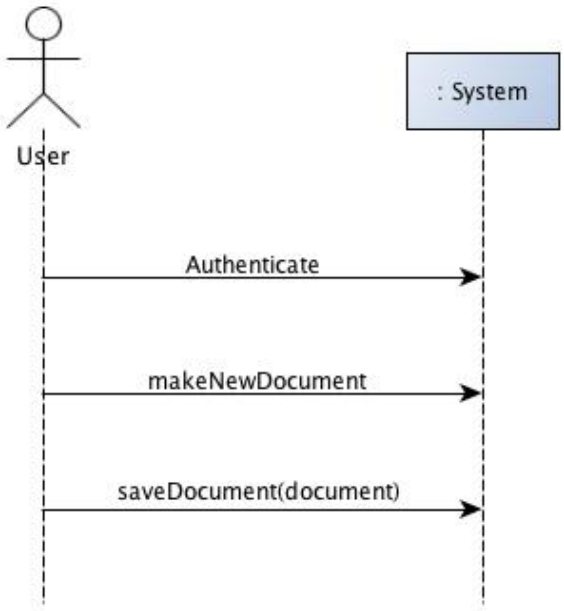


Figure 3: SSD - New document

8 Operation Contracts

Contract CO1: Synchronize
Operation: Synchronize
Cross References: UC1, UC2, UC5
Preconditions: Some documents and/or must have been created on the system.
Postconditions:
- All documents and folders on the client must be uploaded to the server.
- All documents and folders on the server must be downloaded to the client.
- Document version must be the same on the client and server.

9 Use Case Diagram

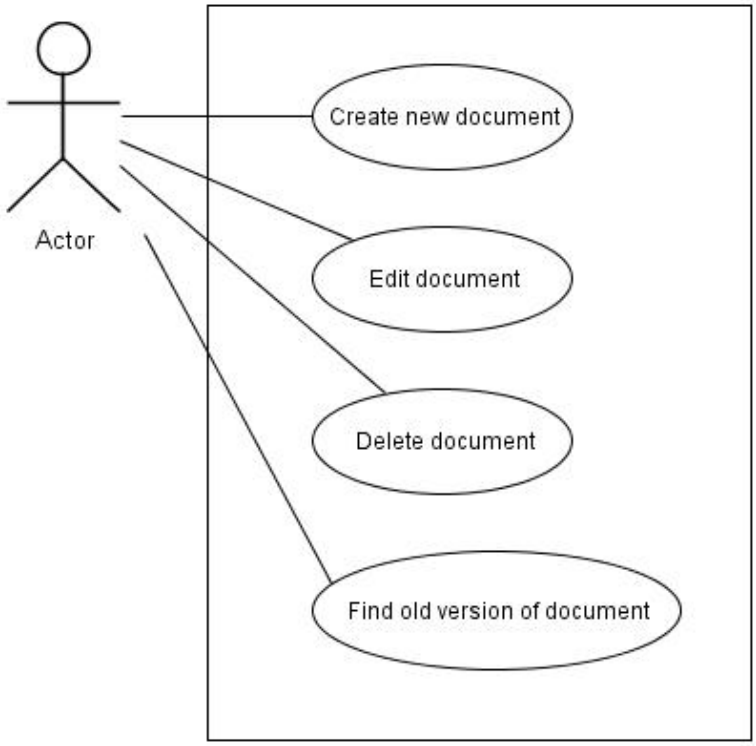


Figure 4: Use case diagram

10 UML Package Diagram

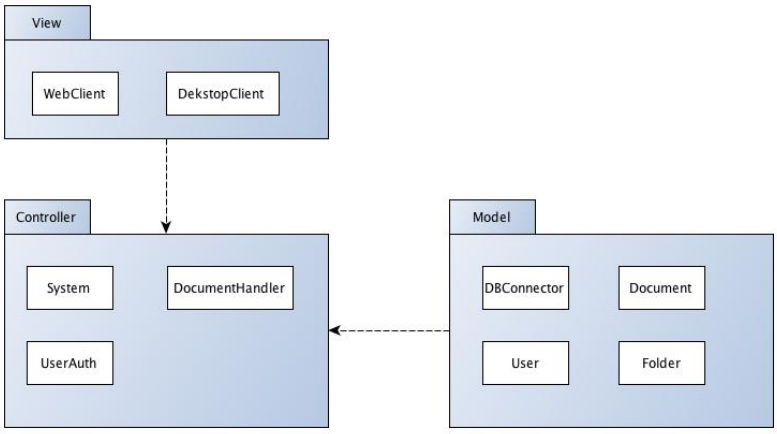


Figure 5: Package Diagram Overview

11 UML Class Diagram

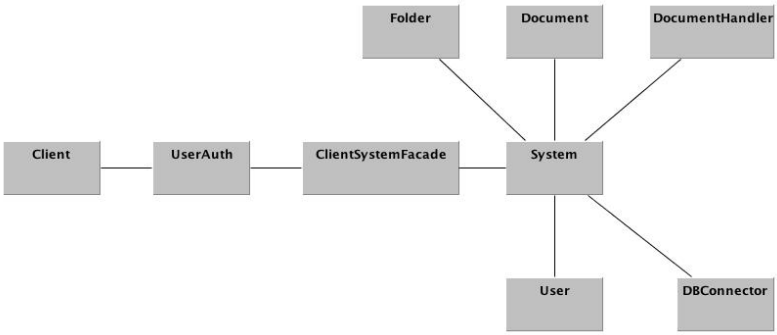


Figure 6: Class Diagram Overview

12 Communication Diagram

Diagrams/Delete Communication Diagram.jpg

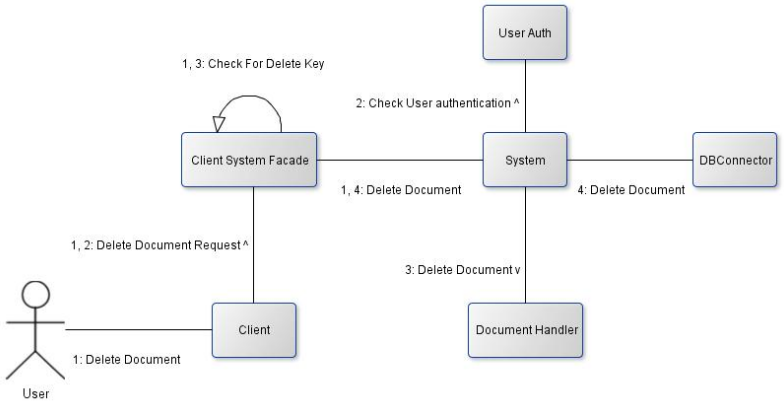


Figure 7: Communication Diagram

13 Technical Memo

Technical Memo
Issue: Files format - Which file format to use

Solution Summary: Use HTML for our file format.
Factors

- Must be able to contain both text and images.

Solution
We chose to use HTML for our file format because it's simple to construct, and can contain text and images seamlessly.
Motivation
We needed a file format that can contain images and text as well as being easy to construct. in addition, HTML can easily be extended to other content. Lastly, HTML can be opened with any browser, so the users isn't tied to SliceOfPie if he just want's to view the content of a file.
Alternatives considered
We considered using a .txt file format, but .txt can only contain plain text. We also considered using our own file format (since the format itself isn't important to the application). But if we use our own format the user is stuck with using SliceOfPie, so he can't view the content of a file with any other application.

14 ER-Diagram

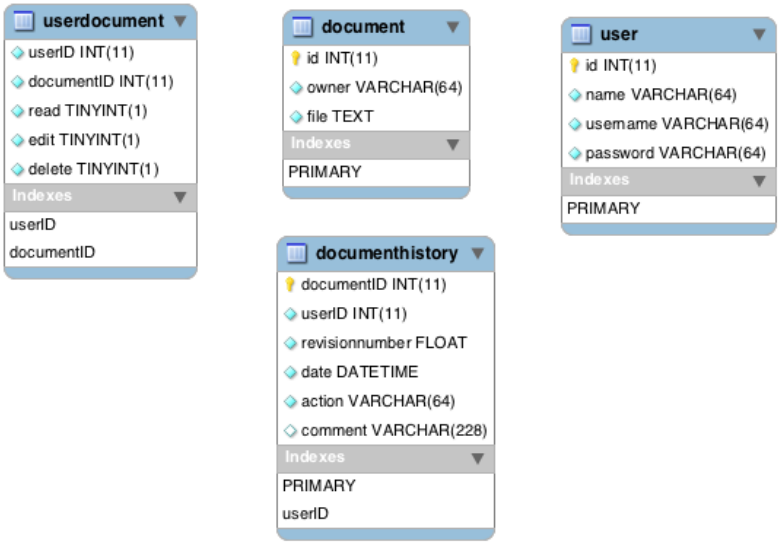


Figure 8: ER-Diagram

15 Revision Table

Table 1: Revision Table

Revision	Changes	Section	Author
21-11-12	Created Vision	1	Bergar
21-11-12	Created Use cases	2	Bergar
21-11-12	Use case 1 and 2	2	Filip
21-11-12	Created Glossary	3	
21-11-12	Created Supplementary Specifications	4	Bergar
21-11-12	Created Domain Model diagram	5	Bergar and Filip
21-11-12	Created Logical Architecture diagram	6	Bergar and Filip
22-11-12	System Sequence Diagram	7	Bergar
22-11-12	Operation Contracts	8	Bergar
22-11-12	Use cases 3, 4, 6 and 7	2	Morten
22-11-12	Use case diagram	9	Filip
23-11-12	UML Package Diagram	10	Bergar
23-11-12	UML Class Diagram	11	Bergar
27-11-12	UML Communication Diagram	12	Morten
04-12-12	Use case 11	2	Bergar
04-12-12	Update class diagram	12	Bergar
04-12-12	Technical memo for file format	13	Bergar
04-12-12	ER-Diagram	14	Bergar