Contents

1 Report **2**
- 1.1 **DONE** Vision & Business case . . . 2
- 1.2 **DONE** Use cases . . . 2
  - 1.2.1 **DONE** Use cases (text) . . . 2
  - 1.2.2 **DONE** Use Case Model (UML) . . . 3
- 1.3 **DONE** Supplementary specification (FURPS+) . . . 3
- 1.4 **DONE** Glossary . . . 3
- 1.5 **DONE** System sequence diagram . . . 3
- 1.6 **DONE** Domain Model . . . 3
- 1.7 **DONE** Logical architecture . . . 3
- 1.8 **DONE** Operation contract . . . 3
- 1.9 **DONE** Interaction Diagram . . . 3
  - 1.9.1 **DONE** Communication Diagram . . . 3
  - 1.9.2 **DONE** Sequence Diagram . . . 3
- 1.10 **DONE** Software Attributes / Qualities . . . 4
- 1.11 **DONE** Package Diagram . . . 4
- 1.12 **DONE** Class diagram . . . 4
- 1.13 **DONE** SAD . . . 4
- 1.14 **DONE** N+1 . . . 4
- 1.15 **DONE** ER-Diagram . . . 4
- 1.16 **DONE** Document GRASP . . . 4
- 1.17 **DONE** HTML Format . . . 4

2 C# **4**
- 2.1 **DONE** C# Basic architecture . . . 4
- 2.2 **DONE** System . . . 4
  - 2.2.1 **DONE** Define central methods. . . . 4
- 2.3 **DONE** Client . . . 5
- 2.4 **DONE** Database . . . 5
- 2.5 **DONE** System architecture . . . 5
- 2.6 **DONE** Test . . . 5
- 2.7 **DONE** User Directory . . . 5
- 2.8 **DONE** User Authentication . . . 5
- 2.9 **DONE** Sharing documents / folders . . . 6
- 2.10 **DONE** Offline synchronization . . . 6

3 Class Diagrams **7**

4 Package diagram **10**

5 Sprint tables **10**

6 Use Cases **12**

1 Report

1.1 DONE Vision & Business case

CLOSED: *2012-11-21 Wed 13:00*

- Introduction
- Problem statement
- Summary of system features

1.2 DONE Use cases

CLOSED: *2012-12-16 Sun 01:34*

1.2.1 DONE Use cases (text)

CLOSED: *2012-12-16 Sun 01:34*

- **DONE** UC1: Create new document
  CLOSED: *2012-11-21 Wed 13:00*
- **DONE** UC2: Edit document
  CLOSED: *2012-11-21 Wed 13:00*
- **DONE** UC3: Delete document
  CLOSED: *2012-11-22 Thu 11:45*
- **DONE** UC4: Merging documents (resolve conflict)
  CLOSED: *2012-12-16 Sun 01:21*
- **DONE** UC5: Offline sync
  CLOSED: *2012-11-22 Thu 11:45*
- **DONE** UC6: New folder
  CLOSED: *2012-11-22 Thu 12:47*
- **DONE** UC7: New project
  CLOSED: *2012-11-22 Thu 13:05*
- **DONE** UC8: Find old version of document
  CLOSED: *2012-12-16 Sun 01:33*
- **DONE** UC9: Share Document
  CLOSED: *2012-12-16 Sun 01:26*
- **DONE** UC10: Log in
  CLOSED: *2012-12-16 Sun 01:27*

1.2.2 DONE Use Case Model (UML)

CLOSED: *2012-11-22 Thu 12:47*

1.3 DONE Supplementary specification (FURPS+)

CLOSED: *2012-12-17 Mon 03:09*

- ☐ Functionality
- ☐ Usability
- ☐ Reliability
- ☐ Performance
- ☐ Supportability

1.4 DONE Glossary

CLOSED: *2012-11-21 Wed 13:01*

1.5 DONE System sequence diagram

CLOSED: *2012-11-22 Thu 12:05*

1.6 DONE Domain Model

CLOSED: *2012-11-21 Wed 13:29*

1.7 DONE Logical architecture

CLOSED: *2012-11-21 Wed 13:58*

1.8 DONE Operation contract

CLOSED: *2012-11-22 Thu 12:47*

1.9 DONE Interaction Diagram

CLOSED: *2012-12-17 Mon 03:09*

1.9.1 DONE Communication Diagram

CLOSED: *2012-11-27 Tue 11:30*

1.9.2 DONE Sequence Diagram

CLOSED: *2012-12-17 Mon 03:09*

1.10 DONE Software Attributes / Qualities

CLOSED: *2012-12-17 Mon 08:28*

1.11 DONE Package Diagram

CLOSED: *2012-11-23 Fri 15:16*

1.12 DONE Class diagram

CLOSED: *2012-11-23 Fri 15:17*

1.13 DONE SAD

CLOSED: *2012-12-17 Mon 08:28*

1.14 DONE N+1

CLOSED: *2012-12-17 Mon 08:28*

1.15 DONE ER-Diagram

CLOSED: *2012-12-12 Wed 18:17*

1.16 DONE Document GRASP

CLOSED: *2012-12-17 Mon 08:28*

1.17 DONE HTML Format

CLOSED: *2012-12-12 Wed 18:17*

2 C#

2.1 DONE C# Basic architecture

CLOSED: *2012-11-22 Thu 14:11*

2.2 DONE System

CLOSED: *2012-11-27 Tue 11:56*

2.2.1 DONE Define central methods.

CLOSED: *2012-12-13 Thu 10:17*

- Method signature
- Document

2.3 DONE Client

`CLOSED:` *2012-12-12 Wed 18:18*

2.4 DONE Database

`CLOSED:` *2012-12-16 Sun 01:42*

- ☒ ER-Diagram
- ☒ User table
- ☒ Document table
- ☒ User-Document table

2.5 DONE System architecture

`CLOSED:` *2012-12-17 Mon 03:10*

- ☒ Implement GRASP
- ☒ Create/Edit/Delete Document
- ☒ Create/Edit/Delete Folder
- ☒ Database Connection
- ☒ GUI Client

2.6 DONE Test

`CLOSED:` *2012-12-17 Mon 08:28*

- TEST EVERYTHING!!

2.7 DONE User Directory

`CLOSED:` *2012-12-16 Sun 01:42*

- ☒ Personal Root directory

2.8 DONE User Authentication

`CLOSED:` *2012-12-17 Mon 03:10*

- ☒ Username and password relationship
- ☒ Storage of username and password
- ☒ Fetch and compare

2.9 DONE Sharing documents / folders

`CLOSED:` *2012-12-16 Sun 01:42*

- ☒ Server code
- ☒ Share with permissions (view/edit/delete)
- ☒ Permisions DB: Add permisions to userdocument

2.10 DONE Offline synchronization

`CLOSED:` *2012-12-17 Mon 03:11*

- ☒ Client to server connection
- ☒ Server handling new data
- ☒ Client accepts all
- ☒ Server handling new data (Where permission meets requirements) (ice cold overwrite)
- ☒ Server simple comparision of document history
- ☒ Server merge of documents

3 Class Diagrams

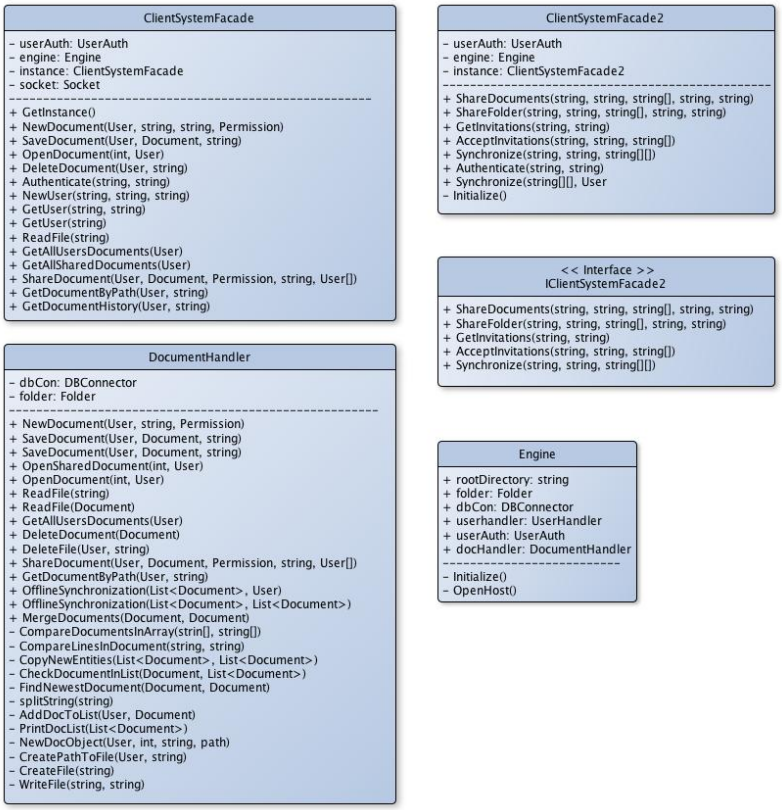

Figure 1: Server class diagram

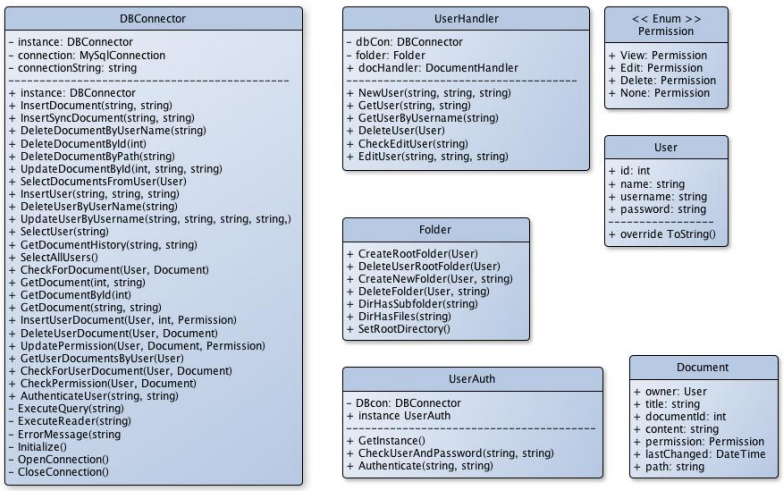

Figure 2: Server class diagram

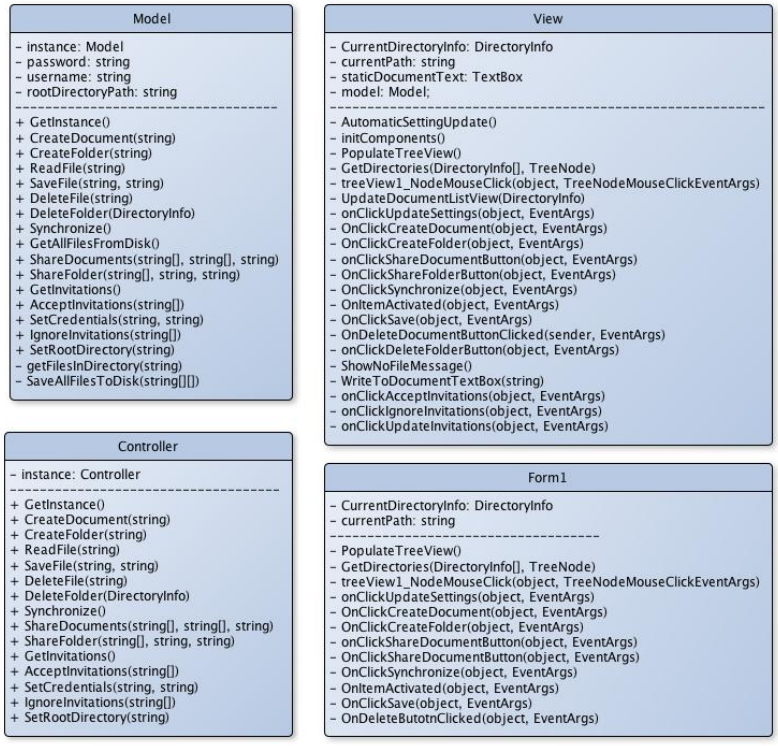

Figure 3: StandAlone Client class diagram

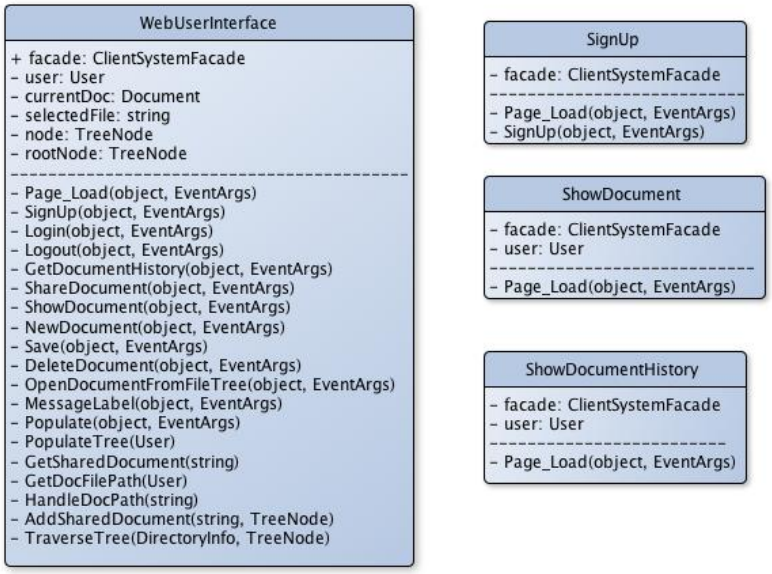

Figure 4: WebUI class diagram

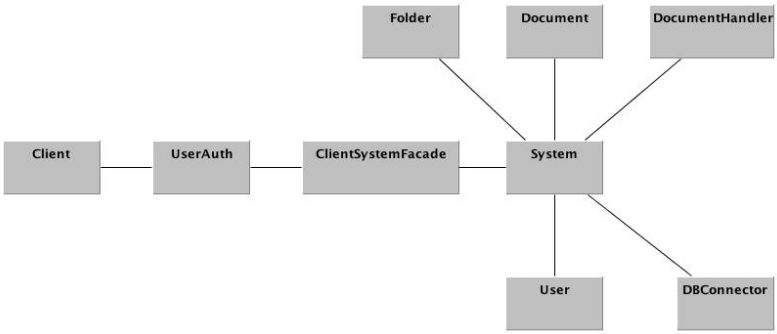

Figure 5: Old version of class diagram

4 Package diagram

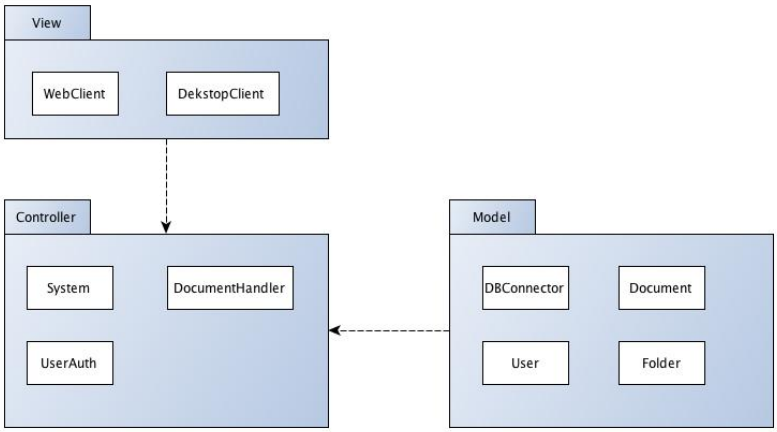

Figure 6: System package diagram

5 Sprint tables

						Remaining effort			
	Product backlog item	Sprint task	Volunteer	Status	Initial est. of effort	day1 (22/11)	day1 (23/11)	day1 (26/11)	day1 (27/11)
Sprint 1					36	22	5	0	0
	Use cases				17	8	0	0	0
		UC1	Filip	Done	1	0			
		UC2	Filip	Done	1	0			
		UC3	Morten	Done	2	0			
		UC4	Bergar	Done	3	0			
		UC5	Morten	Done	3	3	0		
		UC6	Morten	Done	1	0			
		UC7	Morten	Done	1	0			
		UC8	Bergar	Done	2	2	0		
		UC9	Bergar	Done	3	3	0		
	Use case model		Filip	Done	2	1	0		
	SSD		Bergar	Done	2	0			
	Operation contract		Bergar	Done	3	2	0		
	Interaction diagram				4	4	0		
		Communication Diagram	Morten	Done	2	2	0		
		Sequence Diagram	Filip	Done	2	2	0		
	Package Diagram		Bergar	Done	2	2	0		
	Class Diagram		Bergar	Done	2	2	2	0	
	C# Basic architecture		Morten	Done	2	1	1	0	
	System		Bergar	Done	2	2	2	0	

Figure 7: 1. Sprint table

					day 1 (27/11)	day 2 (28/11)	day 3 (29/11)	
Sprint 2					39	9	9	0
	Database				15	0		
		ER-Diagram	Morten	Done	3	0		
		User Table	Morten	Done	3	0		
		Document Table	Morten	Done	3	0		
		User-Document table	Morten	Done	3	0		
		Document-History table	Morten	Done	3	0		
	System Architecture							
		Implement GRASP			9	6	3	0
		C/E/D Document	Bergar	Done	3	1	0	
		Database Connection	Morten	Done	3	3	3	0
		GUI Client	Filip	Done	3	2	0	
	Test							
		test everything	All	Done	6	3	2	0

Figure 8: 2. Sprint table

						day 1 (4/12)	day 2 (6/12)	day 3 (7/12)	day 4 (10/12)	day 5 (11/12)	
Sprint 3					116	98	82	65	55	38	
	User directory				5	2	0				
		Personal root dir	Bergar	Done	3	0					
		shared root dir	Bergar	Done	2	2	0				
	User authentication				6	3	0				
		Username & Password relationship	Morten	Done	2	0					
		Storage of username & password	Morten	Done	1	0					
		Fetch and compare	Morten	Done	3	3	0				
	Sharing document				10	6	0				
		File input	Filip	Done	3	1	0				
		Server Code	Morten	Done	3	1	0				
		Share with permission	Morten	Done	2	2	0				
		Permission db (add field)	Morten	Done	2	2	0				
	Offline sync				28	26	21	13	7	0	
		Client to server connection	Filip	Done	5	5	3	1	1	0	
		Server to Client connection	Bergar	Done	5	3	2	1	1	0	
		Server handling new data (Simple)	Morten	Done	4	4	2	0	0		
		Client accepts all	Filip	Done	3	3	3	3	2	0	
		Server handling new data (Advanced)	Morten	Done	4	4	4	3	0		
		Server simple comparision of doc history	Morten	Done	2	2	2	1	0		
		Server merge of documents	Bergar & Morten	Done	5	5	5	4	3	0	
	Web client	Implement web client	Bergar	Done	5	5	5	5	3	0	
	ER-Diagram	ER-Diagram	Bergar	Done	2	2	2	0			
	File format	Documentation	Bergar	Done	2	2	2	0			
	Class diagram	Update class diagram	Bergar	Done	2	2	2	2	0		
	Testing & Documenting										
		Test Everything	All	NOT DONE	30	30	30	30	30	25	
		Document everything	All	NOT DONE	20	20	20	15	15	10	

Figure 9: 3. Sprint table

6 Use Cases

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: Slice of Pie application
Level: User goal
Primary actor: Regular user
Preconditions: A document must exist on the server and another (edited) version of the same document must exist on the stand-alone client
Postconditions: Both versions (client and server) must have matching content

Basic flow:

1. The user opens an existing document on the client.
2. The user edits the document and saves the lates changes.
3. The stand-alone client and the server are synchronized.
4. Both documents merge and contain both the original and the new content.

Extensions:

*4. There is a conflict when merging the documents.
   (a) The user resolves the conflict manually by editing the document.
   (b) The user synchronizes again.

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: Slice of Pie application
Level: User goal
Primary actor: Regular user
Preconditions: A document with multiple changes must exist
Postconditions: None
Basic flow:

1. A user wants to see an overview of all changes made to a document.
2. User opens a document.
3. User presses a button and is presented with a list of all changes made to the specific document.

Extensions:
Special requirements:

Use case UC9: Share document
Scope: Slice of Pie application
Level: User goal
Primary actor: Regular user
Preconditions: A document must be created in the system and at least two users must be registered.
Postconditions: The document is shared between two (or more) users.

Basic flow:

1. The users creates a new document or opens an existing one.
2. The user selects which users he wants to share the document with.
3. The user shares the document with the selected user(s).

Extensions:
Special requirements:

Use case UC10: 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: