Slice of Pie

Dunno

November 29, 2012

1 Preface

Contents

2 Abstract

3 Scrum

3.1 Definition of Done

3.1.1 General

- Must be reviewed and accepted by another team member
- Story must be completed from beginning to end

3.1.2 Documentation

- Must be in digital form

3.1.3 Program

- Proper documentation has been written. This includes full documentation for public methods
- Tests have been written and they they do not fail
- New code must not break previously written tests

4 Software Analysis

4.1 System Requirements

5 Use Cases

ID	Description	Priority
1	CreateDocumentOnline	High
2	SynchronizeChanges	High
3	MultipleUserEdit	High
4	ViewVersionHistory	Medium

Table 1: Our use cases

5.1 CreateDocumentOffline

Use Case #1: CreateDocumentOffline
ID: U#1
Primary Actor: User / Writer
Stakeholders:
1. User: Wants effective access to a bunch of documents. Wants to share these documents with to other users. Want to be able to access documents in his/her home computer, but still access them elsewhere through a web interface
2. Other users: Potential other users, who wants to use the same document on their home computer or through the web interface.

Precondition: The user is authenticated into the system.
Postcondition: The document is created and saved locally.

Main success scenario:
1. User creates a new document on his local computer.
2. System saves the document offline and online.
3. User edits the newly created document to his liking. When hes done editing, hell request a save from System.
4. System saves the changes made offline and online.
   Steps 3-4 are repeated until User is satisfied with the document.
5. User quits the system repeats steps 1-6 until satisfied.
6. System commits changes offline and online, recording a session of editing to the history of the document.

Extensions
2. System has no online connection

1. System alerts User that his changes will not be saved online. System then defers synchronization to processes shown in U#3.
2. User continues editing as in the main scenario until done.
3. The System makes an offline commit, recording a session of editing that differs from another, possible offline editing session.

5.2 SynchronizeChanges

Use Case #2: SynchronizeChanges
ID: U#2
Primary Actor: User
Stakeholder:
1. User: Wants his files to be available both offline and online. Being available offline permits the user to always access them and edit them even when there is no connection. When online, he can access them from all computers.

Precondition: The user is authenticated into the system.
Postcondition: The local folder is synchronized with the online folder

Main success scenario:
1. User opens the program
2. System synchronizes the local folder with the online folder.
3. User edits a document from the folder locally.
4. User saves the document.
5. System synchronizes the local folder with the online folder.

Extension:
2. System fails to synchronize with the online folder

1. User tries to synchronize using a button.
2. User repeat 2b until 2d.
3. System synchronizes with the online folder.

5.3 MultipleUserEdit

Use Case #3: MultipleUserEdit
ID: U#3
Primary Actor: User1, User2
Stakeholder:
1. User: Wants to edit in a document, that is shared with another user.

Wants the document to be saved online and synchronized with both his and the other user(s) local document folders.

Precondition: User is authenticated. User has created a document as shown in U#1.
Postcondition: The two documents are merged and saved online.

Main success scenario:
1. User1 shares his document with user2 online.
2. User2 synchronizes his local folder with the online folder.
3. User1 edits the document locally.
4. User1 synchronizes with the online document folder.
5. User2 edits the document locally.
6. User2 saves the document locally.
7. User2 synchronizes with the online folder.
8. System merges the online document with the changes made in user2s document.
9. System presents the merged document and the original document to user2.
10. User2 accepts the merged document.
11. System saves the new merged document online.

Extensions:
2. User2 has no internet connection

1. User2 fails to synchronize with the system
2. User2 does not receive the file from the online folder

10. User2 does not accept the merged document

1. User2 edits in the merged document
2. User2 accepts the edited, merged document.
3. System saves the new merged document online.

5.4 ViewVersionHistory

Use Case #4: ViewVersionHistory
ID: U#4

Primary Actor: User
Main:
The user wants to view a history for a file in the Slice Of Pie system. Hell select a file from the graphical user interface in the system and click on a

show version history button. The system will retrieve version history for the file and display it in a new window.

Alternate:

The user wants to revert document to an earlier state. This use case has yet to be defined and elaborated on.

5.5 Domain Model

5.6 System Sequence Diagrams

5.7 FURPS+

6 Software Design

6.1 Class Diagram

6.2 Interaction Diagram

7 Database Design

We use a database on our server, to keep track of who owns which files, who has access and who made what changes to them.
Our relational database contains eight tables:
User describes a user. Email is used as primary key
File describes a file on the server. We use serverpath to describe the path to the folder where the file is located, and name as file name. We don't keep track of folders. By specifying the path to the folder of the file, describing folders become unnecessary. Only downside is that we are unable to handle empty folders as an empty folder is not described by any files. The File table can also hold a Project ID, if the file is a part of a project.
FileMetaData holds meta date for a file such as resolution for pictures. Has a reference to MetaDataType and holds a value.
MetaDataType describes different kinds of meta data. One type may be used by vast amount of files.
FileInstance is a relation between User and File. This describes the local path for a specific user, which allows different users to store their copy of a file, in different locations.
Change is used to keep track of who changed what in which file at what time.
Project holds the title of the project.
ProjectHasUser keeps track of which projects a user has.

7.1 Merge Policy

7.2 Client GUI

7.3 Web GUI

8 Software Architecture

8.1 Architecture Analysis

8.2 Scenarios

8.3 Factor Tables

8.4 Logical Views

8.5 Deployment Views

9 Testing

9.1 Test strategy and test results

9.1.1 Definition of thorough testing

- All public methods must be tested
- All fail scenarios must be tested
- Boundary testing

10 Conclusion

10.1 Reflection Upon Result

11 Appendix

11.1 MySQL Schema

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE SCHEMA IF NOT EXISTS `SliceOfLife` DEFAULT CHARACTER SET
latin1 COLLATE latin1_swedish_ci ;
USE `SliceOfLife` ;

-- -----------------------------------------------------
-- Table `SliceOfLife`.`Project`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SliceOfLife`.`Project` (
  `id` INT NOT NULL ,
  `title` VARCHAR(400) NOT NULL ,
  PRIMARY KEY (`id`) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SliceOfLife`.`File`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SliceOfLife`.`File` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(400) NOT NULL ,
  `serverpath` VARCHAR(400) NOT NULL ,
  `deleted` TINYINT UNSIGNED NULL DEFAULT 0 ,
  `Project_id` INT NOT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_File_Project1_idx` (`Project_id` ASC) ,
  CONSTRAINT `fk_File_Project1`
    FOREIGN KEY (`Project_id` )
    REFERENCES `SliceOfLife`.`Project` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SliceOfLife`.`User`
-- -----------------------------------------------------
```

```
CREATE  TABLE IF NOT EXISTS `SliceOfLife`.`User` (
  `email` VARCHAR(400) NOT NULL ,
  PRIMARY KEY (`email`) ,
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SliceOfLife`.`FileInstance`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SliceOfLife`.`FileInstance` (
  `id` INT UNSIGNED NOT NULL ,
  `User_email` VARCHAR(400) NOT NULL ,
  `path` VARCHAR(400) NOT NULL ,
  `deleted` TINYINT UNSIGNED NULL DEFAULT 0 ,
  `File_id` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`id`, `User_email`, `File_id`) ,
  INDEX `fk_FileInstance_User1_idx` (`User_email` ASC) ,
  INDEX `fk_FileInstance_File1_idx` (`File_id` ASC) ,
  CONSTRAINT `fk_FileInstance_User1`
    FOREIGN KEY (`User_email` )
    REFERENCES `SliceOfLife`.`User` (`email` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_FileInstance_File1`
    FOREIGN KEY (`File_id` )
    REFERENCES `SliceOfLife`.`File` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SliceOfLife`.`Change`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SliceOfLife`.`Change` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `User_email` VARCHAR(400) NOT NULL ,
  `timestamp` BIGINT NULL ,
  `change` TEXT NULL ,
  `File_id` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`id`, `User_email`, `File_id`) ,
  INDEX `fk_Change_User1_idx` (`User_email` ASC) ,
  INDEX `fk_Change_File1_idx` (`File_id` ASC) ,
```

```
  CONSTRAINT `fk_Change_User1`
    FOREIGN KEY (`User_email` )
    REFERENCES `SliceOfLife`.`User` (`email` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Change_File1`
    FOREIGN KEY (`File_id` )
    REFERENCES `SliceOfLife`.`File` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SliceOfLife`.`MetaDataType`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SliceOfLife`.`MetaDataType` (
  `Type` VARCHAR(400) NOT NULL ,
  PRIMARY KEY (`Type`) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SliceOfLife`.`FileMetaData`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SliceOfLife`.`FileMetaData` (
  `id` INT NOT NULL ,
  `value` VARCHAR(400) NULL ,
  `MetaDataType_Type` VARCHAR(400) NOT NULL ,
  `File_id` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`id`, `MetaDataType_Type`) ,
  INDEX `fk_FileMetaData_MetaDataType1_idx` (`MetaDataType_Type`
ASC) ,
  INDEX `fk_FileMetaData_File1_idx` (`File_id` ASC) ,
  CONSTRAINT `fk_FileMetaData_MetaDataType1`
    FOREIGN KEY (`MetaDataType_Type` )
    REFERENCES `SliceOfLife`.`MetaDataType` (`Type` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_FileMetaData_File1`
    FOREIGN KEY (`File_id` )
    REFERENCES `SliceOfLife`.`File` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SliceOfLife`.`ProjectHasUser`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SliceOfLife`.`ProjectHasUser` (
  `User_email` VARCHAR(400) NOT NULL ,
  `Project_id` INT NOT NULL ,
  PRIMARY KEY (`User_email`, `Project_id`) ,
  INDEX `fk_ProjectHasUser_User1_idx` (`User_email` ASC) ,
  INDEX `fk_ProjectHasUser_Project1_idx` (`Project_id` ASC) ,
  CONSTRAINT `fk_ProjectHasUser_User1`
    FOREIGN KEY (`User_email` )
    REFERENCES `SliceOfLife`.`User` (`email` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_ProjectHasUser_Project1`
    FOREIGN KEY (`Project_id` )
    REFERENCES `SliceOfLife`.`Project` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```