NOTEPAD

Brian Smith, Nathan Bergen, Trip Volpe,
Sky Magnuson, Ertan Dogrultan, Daniel Crowell

Software Requirements Specification

Draft 1
April 18, 2008

CSE 403 - CSRocks Inc.

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1	All	First draft	04/08/08
1.1	Nathan	Scope, Features	04/13/08
1.2	Ertan	Revised the document & Added use case diagrams and tables	04/17/08
Final	Trip	Cleaned up structure and text, expanded scope & added UI images	4/18/08

Table of Contents

Description

High-Level Description

Notepad will be a collaborative musical score editor and player. Notepad will not conquer the world. Notepad will conquer the musical world. Notepad will allow multiple composers to create and edit musical scores simultaneously. Notepad will have an easy-to-use interface connected directly to a central server. On the server, users can create and log in to projects, allow others to access partially completed songs, and also save changes directly to the main server to allow future editing and access by others. Notepad will be capable of playing back the created works.

Notepad will be delivered with documentation describing each of the features. This documentation will be developed with the product so there is no massive push at the end to produce documentation. The current goal is to make the user interface simple enough that using the program will not require extensive documentation. A help page will be built into the program to briefly explain the various features. Using the program effectively will require a basic understanding of musical notation. An explanation of this will not be included in the final product.

Target Customers

Our target customer base comprises two main categories: 1) casual, amateur composers, and 2) music teachers and students. Amateur composers will find the feature set adequate for comparatively simple compositions, and will be able to collaborate with peers to expand their creative horizons. Music teachers will find the interactivity useful to give hands-on demonstration of basic musical principles to beginning students without requiring physical proximity.

Scope

At the highest level, Notepad's scope covers the creation, storage, sharing, retrieval, collaborative editing, and playback of musical score projects. The scope also covers the creation of user accounts, user logons, and project permission management. It is similar to Google Documents in that users' projects are stored primarily on a single-point server, and editing is done through a web browser application.

The scope of the editor itself is intended to cover relatively simple musical notation; it will not provide for the use of some of the more sophisticated musical constructs, a full list of which falls outside even the scope of this document. Among the notation that we hope to support: notes and rests of varying lengths (*breve* through *semihemidemisemiquaver*), accidentals, clefs, time signatures, key signatures, tempo, measure bars, ties, and slurs.

Platform

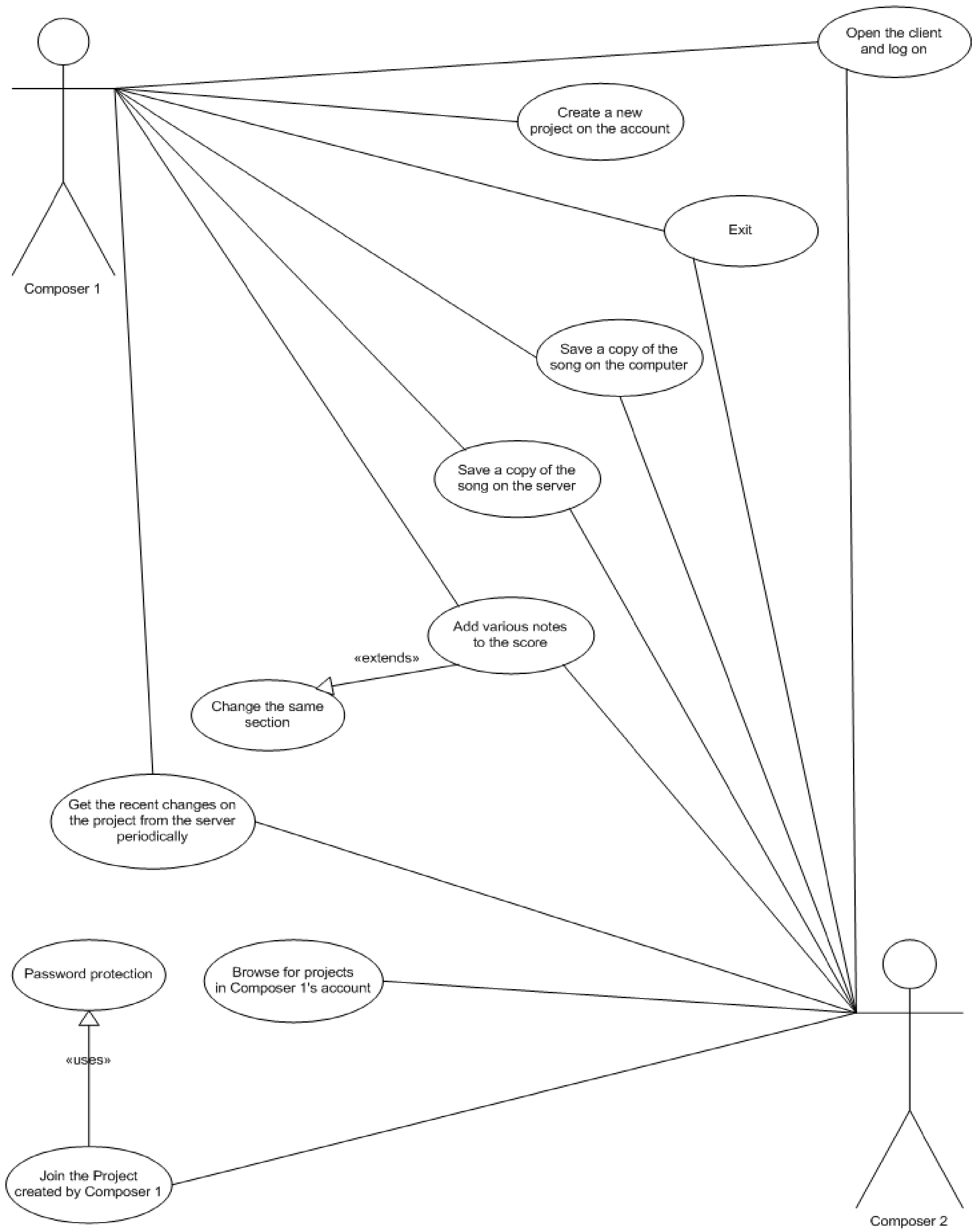
Our goal is to develop the Notepad frontend as a Microsoft Silverlight web application using the C# language and the Windows Presentation Foundation (WPF) component of the .NET Framework 3.0. Silverlight is a very new technology and still not particularly refined. However, its support of WPF provides us with an alternate development path should issues with Silverlight become limiting, as WPF can be used to develop a standalone Windows application. The use of WPF will effectively limit user deployment to Windows environments for the time being, as support on other platforms through Mono is still experimental.

The server component will preferably also be developed with C# and .NET, although other options including ASP.NET are being considered.

Use Case #1

Goal	Create a new score and edit it with multiple participants
Level	Sub function
Actor	Composer2
Primary Actor	Composer1
Precondition	Composer1 and Composer2 both are both logged in to Notepad
Success end condition	The final song represents the combined additions from composer1 and composer2
Failure end condition	The final song does not represent combined additions from composer1 and composer2
Trigger	Composer1 and Composer2 open clients on two different computers
Main success scenario	1. Composer1 clicks on the "Create a New Score" tab to start a project 2. The server creates a new project under Composer1's account 3. Composer1 selects to share the project with “Composer2” 3. Composer2 selects "Join Project" 4. Composer2 browses through list of current projects to find Composer1's project 5. Server verifies that Composer2 is allowed to edit Composer1’s project 6. The two program instances begin sending back changes over the Internet to the server 7. Composer1 and Composer2 add various notes to the song 8. Each user's changes are made on the server copy 9. Composer1 and Composer2 get updated song from the server (6-9 repeat) 10. Composer1 exits the program 11. Composer2 adds more to the song 12. Server copy updates with more changes made by Composer2 14. Composer2 downloads final score to computer 13. Composer2 exits the program 14. Server retains final copy under Composer1's account
Extensions	8a. Composer1 and Composer2 change the same section of the song 8b. Changes from the user with most recent copy of song are added to server copy
Variations	5a. Composer2 joins unlocked project without the need of verification

Summary Diagram of Use Case #1


Composer 1
Open the client and log on
Create a new project on the account
Exit
Save a copy of the song on the computer
Save a copy of the song on the server
Add various notes to the score
«extends»
Change the same section
Get the recent changes on the project from the server periodically
Password protection
Browse for projects in Composer 1's account
«uses»
Join the Project created by Composer 1
Composer 2

Use Case #2

Goal	Edit a new or existing score by adding notation
Level	Subfunction
Primary Actor	Typical Notepad User
Precondition	Notepad is running
Success end condition	Score reflects changes made to it by the user
Failure end condition	Score does not reflect changes made to it by the user
Trigger	User creates a new score or joins an existing project
Main success scenario	1. User selects a musical component button from a toolbar (e.g., note, rest, 2. If applicable, user selects time duration of the component. 3. User moves the editing caret to the desired position in the score using the mouse and/or arrow keys. 4. User presses “Enter” to place the component. 5. Repeat 1-5 until done.
Extensions	
Variations	1-4a. User drags and drops a component from the toolbar to place it. The application 'shadows' what the score will look like when the object is placed; a 50% opacity preview, etc. 2b. Selecting certain components, such as a Key Signature, will show an additional window or panel in which other properties of the object can be set before placement. 4a. User deletes a component by moving the caret adjacent to it and pressing some 'delete' meta-key (right-click, delete, backspace, etc)

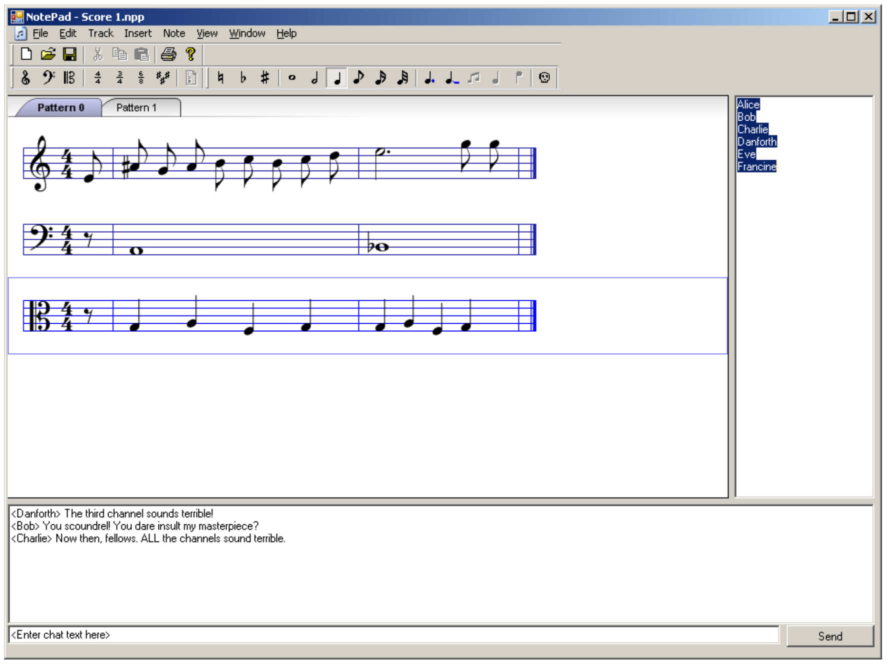
Use case #3

Goal	Play Music
Level	Subfunction
Primary Actor	Typical Notepad User
Precondition	The user has speakers connected to their computer, and has a score with content already loaded.
Success end condition	The user hears an accurate, audio representation of the file's contents
Failure end condition	The user does not hear an accurate audio representation of the file's contents
Trigger	The user presses the “Play” button.
Main success scenario	1. The user presses the “Play” button 2. The system locks/caches the file locally 3. The system converts the music score into a playable audio file 3. The system begins playing the audio file. 4. As the notes are being played, each note is highlighted on the screen by the system. 5. Playback ends, and the system unlocks the file. 6. The user has heard how the composition sounds.
Extensions	2a. The composition contains no sounds. 2a.1. The system aborts, no locking or playing is done
Variations	3a. As the audio file is played, each note in the score is highlighted on the screen at the appropriate time. (This is a stretch goal.)

Feature List

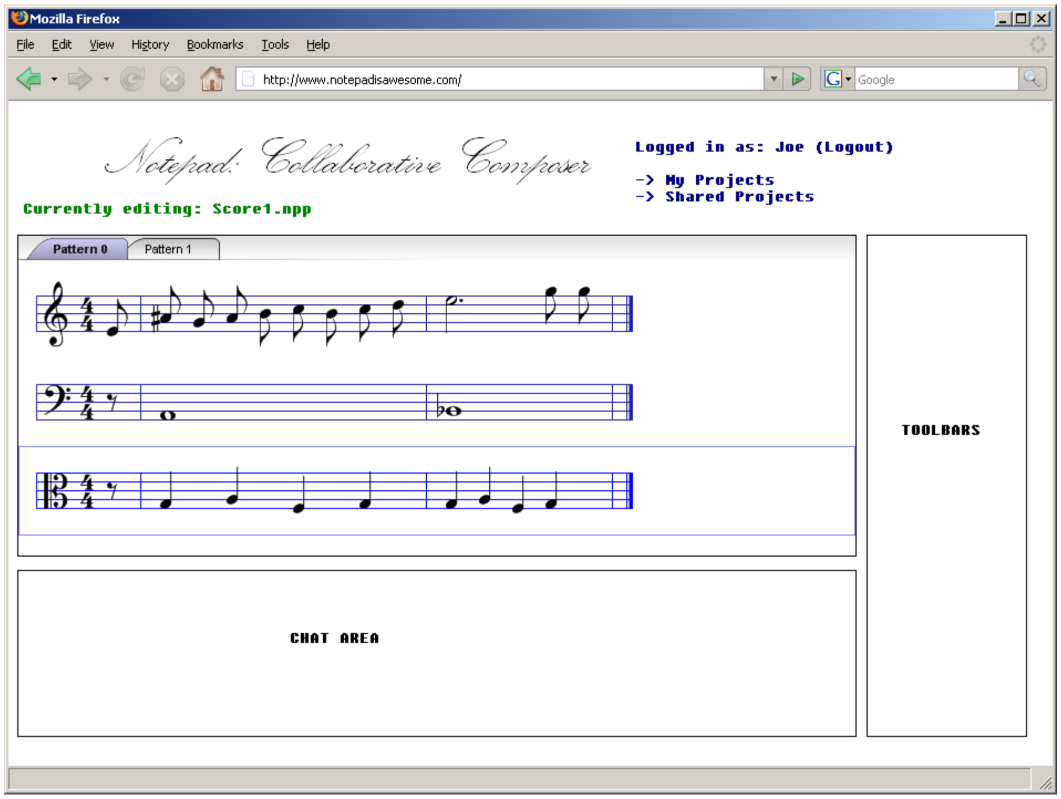
Feature	**Target**
Create accounts and login to the server	Beta
Create / delete / rename a new score	Beta
Grant other uses the ability to alter scores you own	Beta
Add / delete musical notes / rests to a score with mouse	Beta
Add Sharps, Flats to music with mouse	Beta
Add multiple lines of music	Beta
Support Key Signatures	Beta
Add support for altering scores with the keyboard	Final
Allow and gracefully handle simultaneous edits by multiple contributors	Final
Automatically delineate measures	Final
Chat with other users	Final
Change the tempo of the music	Final
Allow different clefs	Final
Support Chords	Final
"Legality" checks on user note placement	Final
"Grammar" checks on user note placement (slur note & rest)	Final
Convert the music to a playable format	Final
Highlight “current” notes during playback	Stretch
Temporarily color coordinated highlighting of locks and changes of other users	Stretch
Read in Midi Files	Stretch
Export to a visual format	Stretch
Print Musical Scores	Stretch
Allow multiple sheets per score (different instruments)	Stretch
Side by side comparison view (by measure)	Stretch

UI Drawings



<Danforth> The third channel sounds terrible!
<Bob> You scoundrel! You dare insult my masterpiece?
<Charlie> Now then, fellows. ALL the channels sound terrible.
<Enter chat text here>

UI Drawing 1. Showing main editing pane, logged-in user list, and chat window. Note that the window frame is no longer relevant, as our goal is to develop a web-based application.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.notepadisawesome.com/
Google
Notepad: Collaborative Composer
Logged in as: Joe (Logout)
-> My Projects
-> Shared Projects
Currently editing: Score1.npp
Pattern 0
Pattern 1
TOOLBARS
CHAT AREA

UI Drawing 2. A possible web interface layout.