Team X

Implementation Document: Project Scherzo

Rebecca Young, Lineker Tomazeli, Alicia Bendz

March 31, 2012

COMP361 Software Engineering Project

School of Computer Science

McGill University

Table of Contents

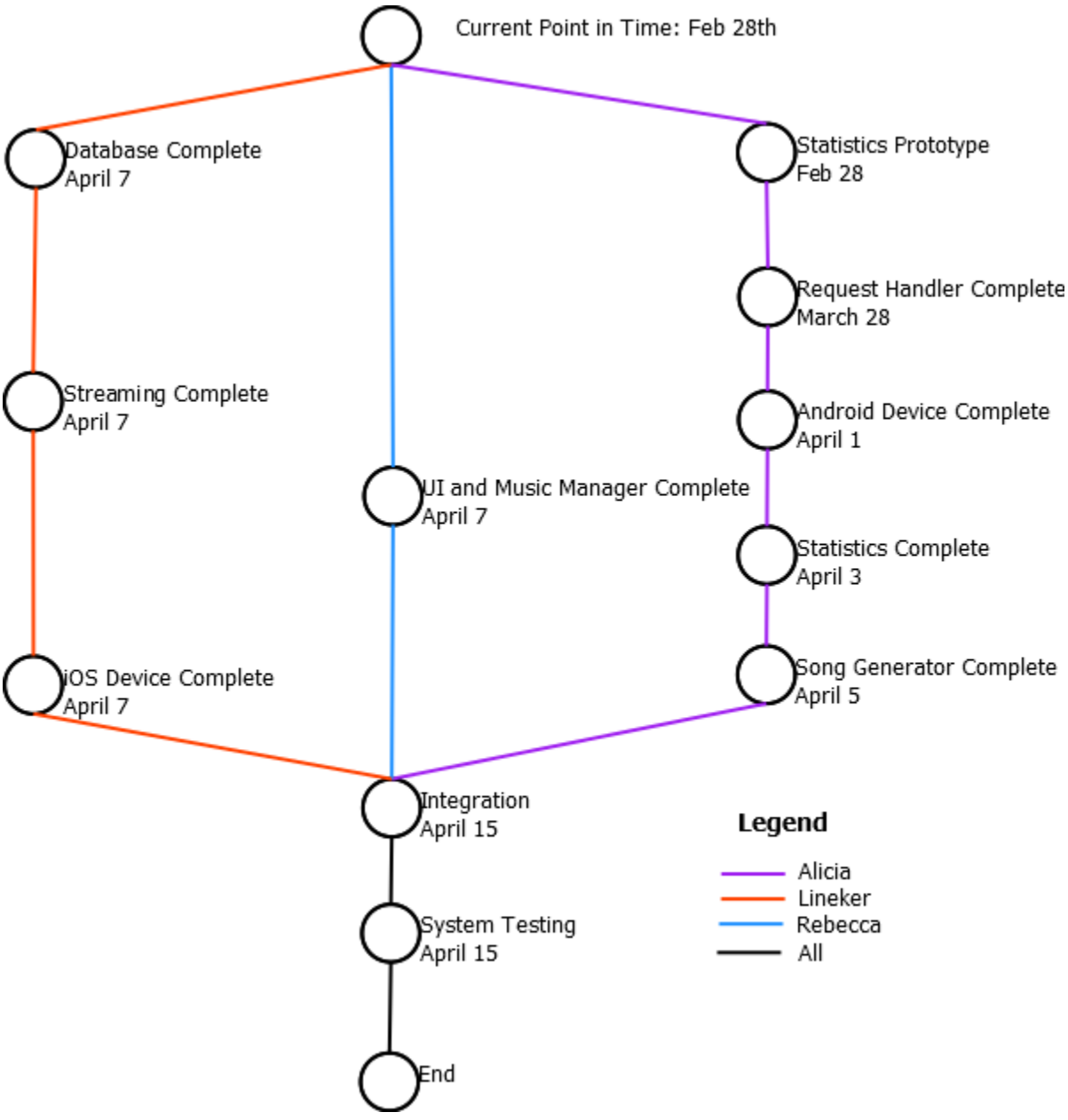
1. Programming Responsibility

1.1 Programming Responsibility Table

Module	Assigned Programmer	Due Date	Date Completed
Request Handler	Alicia Bendz	Mar 28, 2012	March 28, 2012
Android Device	Alicia Bendz	Apr 1, 2012	N/A
Statistics	Alicia Bendz	Apr 3, 2012	N/A
Song Generator	Alicia Bendz	Apr 5, 2012	N/A
Database	Lineker Tomazeli	Apr 7, 2012	N/A
iOS Device	Lineker Tomazeli	Apr 7, 2012	N/A
Main UI	Rebecca Young	Apr 7, 2012	N/A
Music Manager	Rebecca Young	Apr 7, 2012	N/A
Streaming	Lineker Tomazeli	Apr 7, 2012	N/A

2. Milestones and Deliverables

2.1 Dated Activity Chart



2.2 Milestones and Deliverables Table

Event	Assigned Programmer	Description	Due Date	Date Completed
Request Handler Prototype	Alicia Bendz	A Request Handler implementation with all functions but no testing.	Jan 25, 2012	Jan 25, 2012
Android Prototype	Alicia Bendz	An Android application fully implemented but not tested.	Jan 25, 2012	Jan 25, 2012
Music Manager Prototype	Rebecca Young	A Music Manager implementation prototype with all basic functions.	Feb 1, 2012	Feb 1, 2012
Database Prototype	Lineker Tomazeli	A Database with all functions implemented but untested.	Feb 10, 2012	Feb 10, 2012
iOS Prototype	Lineker Tomazeli	An iOS application fully implemented but not tested.	Feb 15, 2012	Feb 15, 2012
Song Generator Prototype	Alicia Bendz	A Song Generator implementation without testing.	Feb 21, 2012	Feb 21, 2012
Main UI Prototype	Rebecca Young	A Main UI implementation without testing.	Feb 24, 2012	Feb 24, 2012
Statistics Prototype	Alicia Bendz	A Statistics implementation without testing.	Feb 28, 2012	Feb 26, 2012
Request Handler Complete	Alicia Bendz	A Request Handler complete with testing and integration.	Mar 28, 2012	March 28, 2012
Android Complete	Alicia Bendz	An Android application with testing and integration.	Apr 1, 2012	N/A
Statistics Complete	Alicia Bendz	The Statistics Module complete with testing and integration.	Apr 3, 2012	N/A
Song Generator Complete	Alicia Bendz	The Song Generator complete with testing and integration.	Apr 5, 2012	N/A
Music Manger Complete	Rebecca Young	A Music Manager module with complete functioning, testing, and integration.	Apr 7, 2012	N/A
Main UI Complete	Rebecca Young	The Main UI complete with integration and testing.	Apr 7, 2012	N/A
Database Complete	Lineker Tomazeli	The Database complete with integration and testing.	Apr 7, 2012	N/A
Streaming Complete	Lineker Tomazeli	The Streaming module complete with integration and testing.	Apr 7, 2012	N/A
iOS Complete	Lineker Tomazeli	The iOS application complete with integration and testing.	Apr 7, 2012	N/A

4. Testing

Method Overview

The method selected for the included test case is addSong(Request r), which has the following function signature:

public synchronized boolean addSong(Request r);

When excuted, this method attempts to add the song in the given Request object to the current list of songs playing. Any Request object may be received. Request objects consist of a Song and an optional specified time in hours and minutes. Songs consist of minutes and seconds, and the method returns a boolean indicating whether the song was successfully inserted. The complete set of test cases involving this method are detailed in Section V of the Design Document.

Selected Test Case

Cases 1-3 in Section V of the Design Document are selected for this logging.

Let T=0 be the current time, and Tf be the latest time for which there is a song scheduled to be played (on the queue). Let Tr be the requested time of the Request r if specified.

Set-up

```
@Before
public void setUp() throws Exception {
	MusicManager man = new MusicManager(false);

	mQueue1 = new ActiveQueue(MIN, man);
	mQueue2 = new ActiveQueue(MIN, man);

	// Block any client requests
	mQueue1.terminateWatcher();
	mQueue2.terminateWatcher();

	// Insert songs onto the list
	Song s1 = new Song("files/02 We're In The Club Now.mp3");
	Song s2 = new Song("files/03 Married Life.mp3");
	Song s3 = new Song("files/04 Carl Goes Up.mp3");
	Song s4 = new Song("files/05 52 Chachki Pickup.mp3");

	LinkedList<Song> list = new LinkedList<Song>();
	list.add(s1);
	list.add(s2);
	list.add(s3);
	list.add(s4);
	mQueue1.setSongs(list);
```

```
		mQueue2.setSongs(list);
}
```

Boundaries Testing

```
/**
 * Test method for {@link MusicManager.ActiveQueue#addSong(MusicManager.Request)}.
 */
@Test
public void testAddSongBoundaries() {
		Song testSong1 = new Song("files/01 Up With Titles.mp3");
		Song testSong2 = new Song("files/22 Up With End Credits.mp3");

		File session = new File("session.ser");

		// Destroy any saved playlist/active queue settings
		if(session.exists()) {
			session.delete();
		}

		DateTime date = new DateTime();
		int hour = date.getHourOfDay();
		int min = date.getMinuteOfDay();

		// Boundary case 1: Request r = null;
		assertFalse(mQueue1.addSong(null));
		// Boundary case 2: r != null; r.min < 0, r.hour >= 0
		assertTrue(mQueue1.addSong(new Request(testSong1, -1, hour)));
		assertTrue(mQueue1.getSongs().size() == mQueue2.getSongs().size());
		// Boundary case 3: r != null; r.min >= 0, r.hour < 0
		assertTrue(mQueue1.addSong(new Request(testSong2, min, -1)));
		assertTrue(mQueue1.getSongs().size() == mQueue2.getSongs().size());
		// Boundary case 4: r != null; r.song = null;
		assertFalse(mQueue2.addSong(new Request(null, 2, 2)));
}
```

Case 1: Tr != null, Tf > Tr.

```
/**
 * Test method for {@link MusicManager.ActiveQueue#addSong(MusicManager.Request)}.
 */
@Test
public void testAddSongCase1() {
		Song testSong1 = new Song("files/Sleep Away.mp3");
		DateTime date = new DateTime();
		int hour = date.getHourOfDay();
		int min = date.getMinuteOfHour();

		// Find time the latest existing song will be played
		int eHour = hour;
		int eMin = min;
		LinkedList<Song> list = mQueue1.getSongs();
		for(int i = 0; i < list.size(); i++) {
			eMin += list.get(i).getSongLength();
		}
```

```
		while(eMin >= 60) {
			eHour++;
			eMin -= 60;
		}
		// Insert at a position smaller than that time
		assertTrue(mQueue1.addSong(new Request(testSong1, eHour, eMin - 10)));
		// There should have been a swap assuming normal range of song length
		assertTrue(mQueue1.getSongs().size() == list.size());
		assertTrue(mQueue1.getSongs().contains(testSong1));
}
```

Case 2: Tr != null, Tf < Tr.

```
/**
 * Test method for {@link MusicManager.ActiveQueue#addSong(MusicManager.Request)}.
 * @throws SQLException
 * @throws ClassNotFoundException
 */
@SuppressWarnings("static-access")
@Test
public void testAddSongCase2() throws ClassNotFoundException, SQLException {
		Song testSong1 = new Song("files/Bleed.mp3");
		DateTime date = new DateTime();
		int hour = date.getHourOfDay();
		int min = date.getMinuteOfHour();
		// Load a test playlist for generating filler
		int pid =
		Global.getSRVInstance().PlaylistService().GetPlaylistIdByName("rUp");
		Collection<Song> coll =
		Global.getSRVInstance().SongService().getSongsByPlaylistId(pid);
		LinkedList<Song> link = new LinkedList<Song>();
		link.addAll(coll);
		Playlist playlist = new Playlist();
		playlist.setSongs(link);

		mMan.changePlaylist(playlist);
		mQueue1 = new ActiveQueue(MIN, mMan);
		mQueue1.setSongs(mList);
		int origLength = mQueue1.getSongs().size();

		assertTrue(mQueue1.addSong(new Request(testSong1, hour, min + 40)));
		int newLength = mQueue1.getSongs().size();
		// The last element should be the added song
		assertTrue(mQueue1.getSongs().get(newLength - 1).equals(testSong1));
		// Additions; the length of the song list should be longer by > 1
		assertFalse(mQueue1.getSongs().size() == origLength + 1);
		// Check that insertion was +-3 minutes
		int sum = 0;
		int target = hour*60 + min;
		Song temp;
		LinkedList<Song> songs = mQueue1.getSongs();
		for(int i = 0; i < songs.size(); i++) {
			temp = songs.get(i);
			if(temp == testSong1) {
				System.out.println("Found it.");
```

```
                    break;
                }
                sum += temp.getSongLength();
        }
        System.out.println("Sum: " + sum);
        System.out.println("Target: " + target);
        assertTrue(Math.abs(sum - target) <= 3);
        // Depending on sample songs used, this may fail.
        // With this set it always passes. Failing occasionally is OK.
}
```

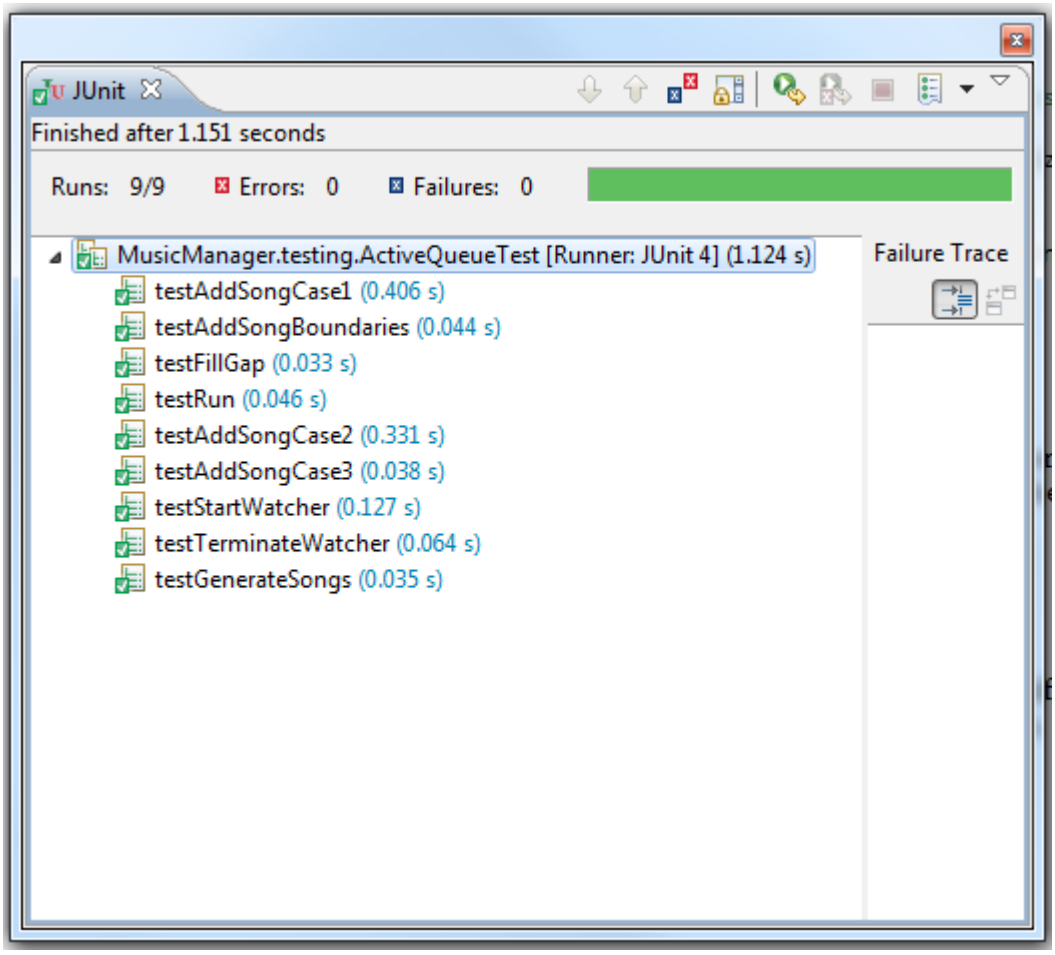
Case 3: Tr == null.

```
/**
 * Test method for {@link MusicManager.ActiveQueue#addSong(MusicManager.Request)}.
 */
@Test
public void testAddSongCase3() {
        Song testSong1 = new Song("files/Bird Song.mp3");
        mQueue1 = new ActiveQueue(MIN, mMan);
        mQueue1.setSongs(mQueue2.getSongs());

        // The length of the song list should be equal after a successful insert
        assertTrue(mQueue1.addSong(new Request(testSong1, -1, -1)));
        assertTrue(mQueue1.getSongs().size() == mQueue2.getSongs().size());
        assertTrue(mQueue1.getSongs().contains(testSong1));
        // It should be the third song on the list
        assertTrue(mQueue1.getSongs().get(2).getTitle().equals(testSong1.getTitle()));
}
```

Results

The results for all test cases were as expected, after a few failures due to errors in the test case assertions. These were not in the actual code, so are not logged here. With these corrected, the results were:



Note that as stated in the Design Document, since all tests are based in specific instances of Song objects (with distinct lengths), extended testing using varying playlists, songs, and times of day is also being done to thoroughly test this method.

6. Modification Log

Revision	Time	File Count	Author	Message
171	Yesterday 10:18 PM	1	ryoung11	Merge.
170	03/27/2012 09:41 PM	4	abendz	Changed song gen to use new song gap from music manager.
169	03/26/2012 09:24 PM	2	ryoung11	Constraint changed to SongReplayGap.
168	03/26/2012 12:52 PM	5	ltomaz	changed the way is opening connections to db
167	03/26/2012 12:51 PM	1	ltomaz	db working
166	03/23/2012 04:06 PM	3	abendz	Changed stats colours.
165	03/23/2012 03:00 PM	6	abendz	Minor changes. Mostly tiny UI fixes.
164	03/22/2012 04:21 PM	1	abendz	Post peer review rework.
162	03/22/2012 11:18 AM	22	abendz	Add multicasting android and request handler.
160	03/21/2012 01:02 AM	3	ryoung11	Testing for ActiveQueue. Commented out some println's in MusicManager and AQ.
157	03/14/2012 12:52 PM	1	ryoung11	Updating play count before song finishes playing.

Revision	Time	File Count	Author	Message
156	03/12/2012 12:48 PM	3	ryoung11	Play playlist error corrected.
155	03/12/2012 11:54 AM	28	abendz	Some fixes.
154	03/12/2012 10:54 AM	1	ryoung11	UI update
153	03/08/2012 09:27 PM	11	abendz	UI and Android changes.
152	03/08/2012 04:30 PM	2	ltomaz	fixed SSL issue grooveshark
151	03/08/2012 04:02 PM	2	ltomaz	fixed SSL issue grooveshark
150	03/07/2012 07:26 PM	1	ryoung11	update.
149	03/07/2012 07:22 PM	1	ryoung11	Main with Req handler
148	03/07/2012 07:13 PM	1	ltomaz	new DB connection
147	03/07/2012 03:06 PM	3	ryoung11	Can remove songs from playlists.
146	03/07/2012 02:35 PM	1	ryoung11	Merge.
145	03/04/2012	5	ryoung11	Pause for streaming implemented.

Revision	Time	File Count	Author	Message
	08:58 PM			
144	03/04/2012 03:09 PM	7	ryoung11	Plays streaming songs. Pause command not implemented for a streamed song yet.
143	03/04/2012 01:26 PM	2	ltomaz	add method to update LastPlayed song property
142	03/03/2012 12:03 AM	7	ryoung11	Sets past playlist as default when adding songs.
141	03/02/2012 12:59 AM	8	ryoung11	Streaming integrated minus playing them. Can add them to playlists. Pause implemented. Now loads your last session's pla…
140	03/01/2012 10:40 PM	22	abendz	Odds and ends with android and request handler.
138	02/29/2012 02:21 AM	1	ltomaz	add method addToCurrentPlaylist
137	02/29/2012 02:15 AM	5	ltomaz	Grooveshark is working again. Added Button to add selected songs from UIStreaming.
136	02/29/2012 02:10 AM	1	ltomaz	empty database, just copy and replace the old one if necessary
135	02/28/2012 09:23 PM	4	ryoung11	Integrated Statistics and SongGenerator into UI.
134	02/28/2012 04:52 PM	6	abendz	Updated testing for SongGen.

6. Programmer's Journal

6.1. Lineker Tomazeli

Entry 1	
Date:	Comments:
January 22, 2012	Database is up and running. A lot of encapsulation so it becomes simple to query the database (max refactoring)
Entry 2	
Date:	Comments:
February 15, 2012	The team keeps asking me for new queries. I think they should try writing it.
Entry 3	
Date:	Comments:
February 25, 2012	Grooveshark changed their API. The streaming module is not working anymore.
Entry 4	
Date:	Comments:
February 30, 2012	Found an working round for Grooveshark. Streaming is back to LIVE. Hopefully we won't have problems again
Entry 5	
Date:	Comments:
March 15, 2012	We are having many problems when too many request are sent to the sqlite (db). We either have "SQLite busy" message or "Database Locked"
Entry 6	
Date:	Comments:
March 17, 2012	I did some changes to the database module. Now instead of a singleton connection each request opens a new connection. Sqlite seems to prefer this way. No more error messages.
Entry 7	
Date:	Comments:
March 19, 2012	I think we are a team of doers. This can make any project successful. Is good man everyone does their part.
Entry 8	
Date:	Comments:
March 30, 2012	I think we can improve our UI.

6.2. Rebecca Young

Entry 1	
Date:	Comments:
February 19, 2012	We discovered that the GUI for the server application did not work on Lineker's Mac (the frame appeared but was frozen; no user-interaction at all) because of a .jar file in the build path. Apparently on a Mac, this file changed the launch configurations in Eclipse and due to some bug it disabled Swing. This discovery is good since it works now but now we have to figure out what to do about that jar since we need it for the stats module.
Entry 2	
Date:	Comments:
February 21, 2012	Scherzo v0.0 is rivaling iTunes in number of threads at runtime. We are up to 31; iTunes runs between 33 - 40
Entry 3	
Date:	Comments:
March 7, 2012	Integration for the first time of all the modules together was simple, since we had all been testing already using each other's modules (albeit sometimes incorrectly).
Entry 4	
Date:	Comments:
March 20, 2012	All the failures so far from running jUnit tests on ActiveQueue methods came from errors in the test case set-up.
Entry 5	
Date:	Comments:
March 26, 2012	Made a small change as a result of our post-peer review meeting. Instead of asking users to specify the maximum number of times a song may be played per session, we will ask them how much time should pass between song plays; ie, frequency. This makes much more sense from a user POV.
Entry 6	
Date:	Comments:
March 28, 2012	As a whole I like our team dynamics. I definitely learned things from my teammates. In my opinion, when you can trust the quality and timeliness of your teammates' work, and there is good method of communication between each member, then the team works.

6.3. Alicia Bendz

Entry 1	
Date:	Comments:
Feb 14, 2012	Working in a team is great when you can trust that the work will be done by your teammates and you can see a significant increase in throughput of work for the team vs. just one person.
Entry 2	
Date:	Comments:
Feb 15, 2012	There is some difficulty in coordinating deadlines and making sure they are kept. Since there is no supervisor, team members need to monitor each other, which is too time consuming, or we just hope that others are doing their work correctly. Without constant communication this is worrisome occasionally.
Entry 3	
Date:	Comments:
Feb 16, 2012	Some time is wasted when not everyone is on the same page in terms of project development. Hacks and temporary fixes need to be done in order to keep developing but then everything needs to be reworked after.
Entry 4	
Date:	Comments:
Feb 21, 2012	Often assumptions are made about what team members know or how they will work. When these assumptions are incorrect, this leads to going back and reworking or trying to patch up problems that occurred from assuming someone would work a certain way and didn't. Example, unfamiliarity with SVN technology resulted in wasted time trying to fix conflicts and repository problems.
Entry 5	
Date:	Comments:
Feb 24, 2012	It would be easier if there were more scheduled update meetings. Of course, we're all restricted by time and can't have frequent meetings but often I can't really tell where the other members are in terms of progress or when they plan to work on certain things.
Entry 6	
Date:	Comments:
March 7, 2012	It's hard to synchronize work when everyone is at a different stage of development.
Entry 7	
Date:	Comments:
March 8, 2012	Sometimes you realize when implementing something and encountering a problem, it would have saved some trouble if the team had just decided upon something before instead of waiting until it came up.

Entry 8	
Date:	Comments:
March 13, 2012	Integration seems to take a long time when you need to wait for other team members to catch up or to catch up to their stage. It's hard to determine when things need to be done when there aren't many fine grained due dates.
Entry 9	
Date:	Comments:
March 14, 2012	Work goes so quickly when everyone has a complete understanding of the project. Integration is done easily when everyone understands where each part goes and what it's supposed to do.
Entry 10	
Date:	Comments:
March 19, 2012	Without an "official" client, it's difficult to gauge when something is "done" or "good". It's hard to see the project as a client would since I've been working on it for so long.

INSPECTION SUMMARY

System: DP _____ Rel. No. _____ Date: March 21, 2012

Subsystem Song Generator _____ Rel. No. _____

Module name Song Generator _____ ID _____

Moderator: Lineker Tomazeli _____ Room: 513 Phone: _____

Inspection type:

- ☐ Requirements
- ☒ Code
- ☐ High Level Design
- ☐ Test Plan
- ☐ Detail Design
- ☐ Test Case

Error	MINOR ERRORS				MAJOR ERRORS			
	M	W	E	Total	M	W	E	
IF: Interface								
DA: Data		1		1				
LO: Logic		1	1	2				
IO: Input/Output								
PF: Performance								
HF: Human Factors								
ST: Standards								
DC: Documentation	2	1		3				
OT: Other		1		1				
			Total	7			Total	0

McGill

COMP 361
Software Engineering Project

INSPECTION REPORT

System: DP Rel. No. _____ Date: March, 21, 2012

Subsystem Song Generator Rel. No. _____

Module name Song Generator ID _____

Moderator: Lineker Tomazeli Room: 513 Phone: _____

Size of material: 87 lines

Inspection Meeting Number: 1 Inspection Duration: 1 hr

Number of Inspectors: 2 Total Preparation Time: 10 min

Unit disposition: ☑ meet ☐ re-examine ☐ re-inspect

Estimated rework effort: 1/2 (hours)

Rework to be completed by: March, 28, 2012

Reinspection scheduled for: March 28, 2012

Other inspectors:

Lineker Tomazeli

Rebecca Young

Moderator certification: L Tomazeli Date: 21 March 2012

Additional Comments: _____

System: DP Rel. No. _____ Date: March 21, 2012
Subsystem: Song Generator Rel. No. _____
Module name: Song Generator ID _____
Moderator: Lindker Room: 513 Phone: _____

Inspection type:

- ☐ Requirements
- ☐ High Level Design
- ☐ Detail Design
- ☑ Code
- ☐ Test Plan
- ☐ Test Case

Location	Error Description	Error Type:	Error Class:	Severity:
76	Missing Javadoc	Doc	M	Minor
89	this should be a class constant	Data	W	Minor
114	Should be SONG_LISTS/2	Funct	W	Minor
129-130	Unnecessary lines of code	Funct	E	Minor
135	Wrong comment.	Doc	W	Minor
155	Change to get(0)	Syntx	W	Minor
74-75	Specify units of these parameters	Doc	M	Minor
~~76-100~~				

Error type: IF=Interface DA=Data LO=Logic IO=Input/Output PF=Performance HF=Human Factors ST=Standards DC=Documentation OT=Other
Error class: M=Missing W=Wrong E=Extra

Type

- Interface
- Data
- Logic
- I/O
- Performance
- Standards
- Documentation
- Syntax
- Functionality
- Test Environment
- Test Coverage
- Other

Severity

- Minor
- Major

Class

- M – missing
- E – extra
- W – wrong