Team X

Preliminary Design Report

Rebecca Young, Lineker Tomazeli, Alicia Bendz

October 7, 2011

COMP361 Software Engineering Project

School of Computer Science

McGill University

Table of Contents

I. Project Overview 2

- **i.** Team information 2
- **ii.** Motivation and description 2
- **iii.** Baseline 3
- **iv.** Success factors 5

II. System Diagram 6

- **i.** Block diagrams 6

III. Formal Phased Project Breakdown 8

IV. Milestones and Deliverables 11

V. Dated Activity Graph 14

VI. Resource Requirements 14

VII. Cost Estimate 16

VIII. Engineering Methods 16

IX. Risk Identification 18

X. Legal Disclaimer 19

PROJECT OVERVIEW

Team Information

Rebecca Young	260413151	Software Engineering
Lineker Tomazeli	260348949	Software Engineering
Alicia Bendz	260307158	Computer Science

Motivation

Our goal is to create a program that will allow the clientele of an establishment to request the background music they listen to on-site. The establishments we have in mind could be restaurants, hospitals, or any organization with a waiting room or public area. Downloading an application onto an iPhone or Android device will allow a client to request songs from a pre-selected playlist. This request will be sent to the server component, processed, and feedback will be returned to the client regarding the success of the request.

Overview Description

The project has two components: the desktop or server software and the smart phone or client application. The components will be monitored by a server user and clients respectively.

The desktop component will play music according to a playlist and act as a server, processing all song requests. There will be a general database storing song files and playlists which will be accessible only to this component. The software will handle client requests for songs, return queue request feedback, and intelligently generate playlists based on song request statistics and user defined settings. The manager will be able to specify settings including which playlist to make viewable to the clients, whether to stream external songs via cloud, set constraints on song request frequency, and view a graphical presentation of request statistics including the song popularity and time of day. An additional feature will be the receiving and displaying of client feedback.

The mobile application will make requests to the server to queue particular songs queried from the approved list provided by the server administrator. The client will also be able to give certain specifications, such as the preferred time at which the requested song should be played. The phone application will also allow text feedback to be sent to the server. Supported systems will be iOS and Android.

Baseline

We have observed several popular ways that service providers currently provide background music for their clients: a multi-CD player with a sound system, an iTunes playlist on a computer hooked up to speakers, or simply playing from a radio station. All of these involve some kind of speaker system, so for our project we will assume it is present. The first two scenarios require that the user already has all the music files, either on CDs or as local files. *See Resource Requirements, p. 14.*

As an analysis of the baseline models:

	Model	Pros	Cons
1	Multi-CD player	Complete control over content and play order.	Input must be present and in CD form; there is an upper bound on the maximum number of CDs that may be loaded; there must be at least one CD. Difficult to modify and view order. Clients have no role.

2	iTunes playlist	Complete control over content and play order; songs may be shuffled randomly for variety. Active list can be viewed and modified easily. With available network, playlist can be set as sharable and playlists of devices within range can be played.	All files must be owned and on local computer; or CDs must be ripped. Clients have no role.
3	Radio	Files do not need to be owned; no real upper bound on number of songs; can choose a general genre of music by selecting a station. Selections generally correspond to the season and the popular songs.	Little to no control over content and order. Clients have no role.

For each of these three major models, the clients have no way to participate in choosing music aside from making a request in person, or without the server user manually allowing other playlists to be played, in the case of iTunes.

Success Factors: Baseline Derived

Our software hopes to draw from the current baselines, keeping certain features, dropping ones which complicate the system, and adding the crucial element which allows clients to actively choose their background music and request its playing time.

While we assume the existence of a sound system and a computer connected to a network, our software will encapsulate the radio concept by having the option of streaming music from the internet. In the case that the user has no local files, he will be able to select a playlist of streamed music. We will research several options for streaming and determine what kind of general control the server user can have.

The server administrator will also have the ability to completely control the content of the playlist, by pre-selecting a playlist from which clients can select music. This means that the server user can still ensure that all music will be appropriate for the environment and the clients, but clients will also have some freedom to choose.

All of these are already incorporated into our software design, so success will be measured by the following criteria:

1. **Functionality:** if each module works according to our specified standards. This will be measured by checking that, at the very least, clients are able to request local music to be played on the server.

2. **Minimal changes:** that the number of large revisions is minimal (less than half the number of modules); “large” being a change that affects functionality.

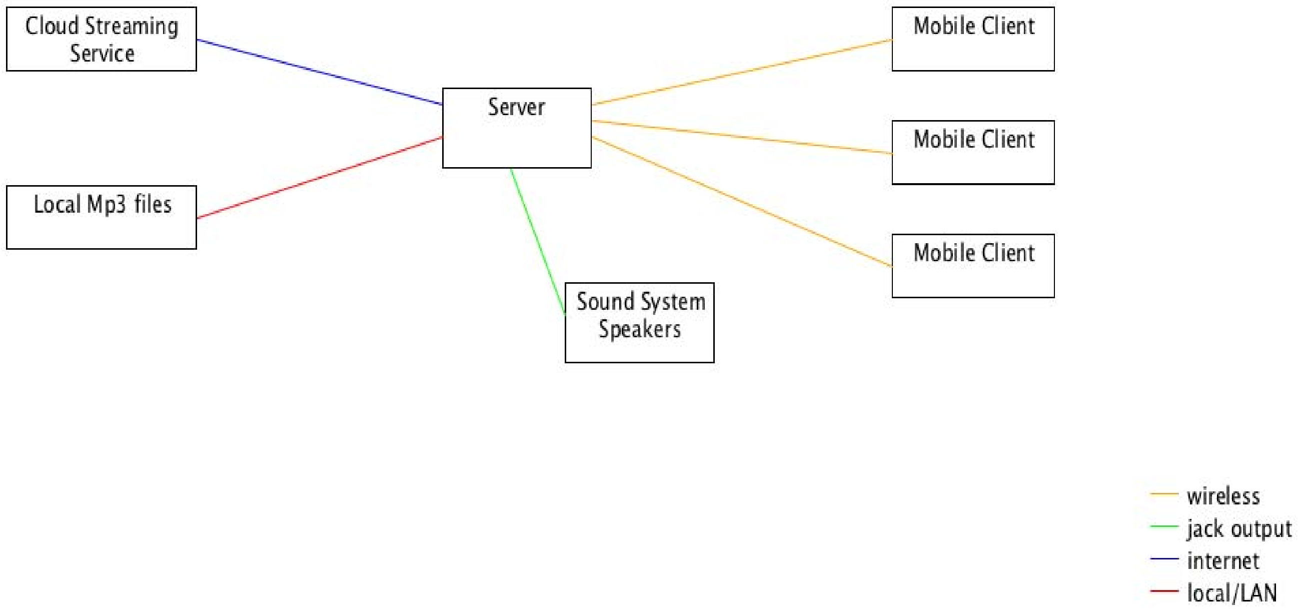
Success Factors: General

As we do plan to make our proposed software available to the public, we are also considering general success factors, of which these four are predominant:

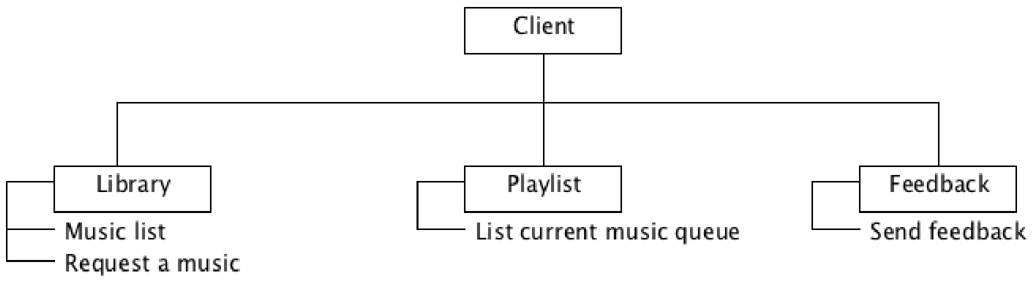
ID	Factor	Description	Measurement
1	**Usability**	Due to our target audience's relatively low experience with technology, this factor is crucial. Both the service provider staff and the clients must be able to intuitively understand and navigate the software and application; the software interface must be simple, and clear yet attractive.	We will release a beta version and try them with test users (these may include small interested companies, classmates, friends, and relatives), and get feedback on usability. At least 70% of the responses should be positive. We will make improvements based on this for the real release.
2	**Reliability**	Software malfunctioning would be an immediate deterrent, as it could damage the client's experience with the service provider.	We will have extensive unit testing, and will manually perform tests in many different locations. Our test users will also give feedback on this. Success will be >90% reliability, meaning that there will be at most 1 crash out of 10 times the software is used.
3	**Robustness**	The software must be capable of functioning in environments where there may not be good network connection or the hosting computer has problems.	Same as above, but with success measured as >80%.
4	**Compatibility**	The software needs to function on standard operating systems; specifically, Windows, Mac, and Linux.	We will run three single tests on each of these systems and confirm that they all have full functionality.

BLOCK DIAGRAMS

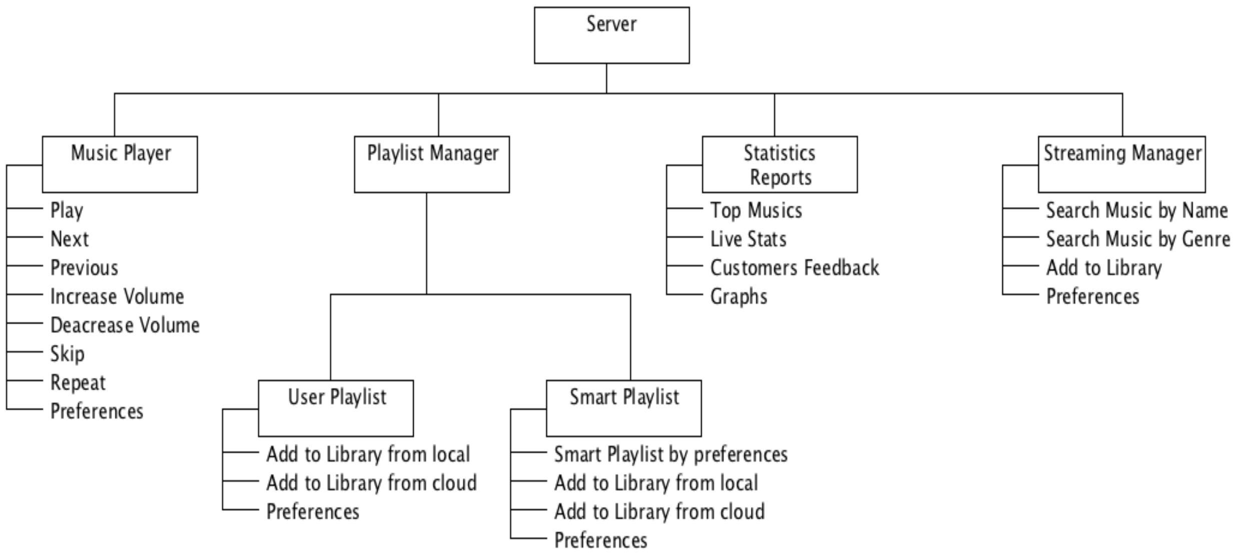
General Structure



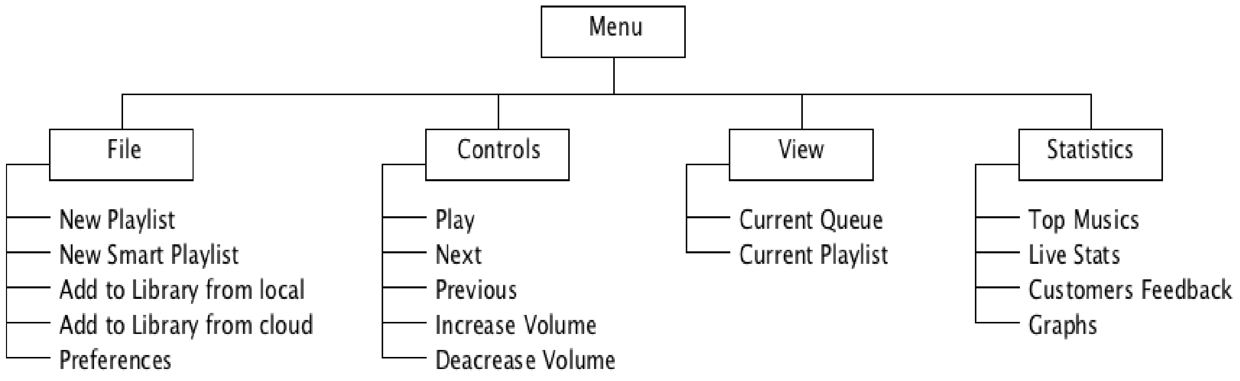
Client View



Server View



Menu View



Formal Phased Project Breakdown

Phase/Activity	**Duration (days)**	**Due date/ Milestone**
1.Research and Design		
1.1 Decide upon project and create summary	3	Sept. 16 - Milestone 1 - Deliverable 1
1.2 Create Preliminary Design document	12	Oct. 7 - Milestone 2 - Deliverable 2
1.3 Research and decide upon music playing library	1	Oct 21 - Milestone 3
1.4 Determine what files are acceptable or if conversion will be possible (if so... research libraries)	1	Oct 21 - Milestone 3
1.5 Decide what kind of statistics and reporting will be done and research graphing/plotting libraries	1	Oct 21 - Milestone 3
1.6 Figure out what preferences and settings will be configurable and where they will be stored	1	Oct 21 - Milestone 3
1.7 Research and decide upon streaming APIs	1	Oct 21 - Milestone 3
1.8 Research and decide upon method of implementing distributed system	1	Oct 21 - Milestone 3
1.9 Determine traits of songs which playlist generator can base selection on; decide on algorithms to use (hill climbing)	1	Nov 4 - Milestone 3
1.10 Decide on classes and responsibilities for playlists	1	Nov 4 - Milestone 3
1.11 Decide upon how to implement database	1	Nov 4 -Milestone 3
1.12 Create UI designs	5	Nov 4 - Milestone 4
1.13 Requirements Document	6	Nov 4 - Milestone 4 - Deliverable 3

1.14 Design Document	12	Jan 27 – Milestone 8 - Deliverable 4
1.15 Implementation Document	12	Feb 17 – Deliverable 5 – Milestone 10
2. Music Player		
2.1 Experiment with library and create simple music playing program	2	Nov 18 - Milestone 5
2.2 Implement a music queue to hold a configurable number of songs to play in sequence	2	Nov 18 - Milestone 5
2.3 Extend music playing program to have all desired actions (skip, play, pause, stop, repeat, etc.)	3	Nov 18 - Milestone 5
2.4 Extend music playing program to accept streaming songs; perhaps detect quality of streaming songs to switch to local files if quality is low	4	Dec 2 - Milestone 6
2.5 Create and run unit tests	2	Dec 2 - Milestone 6
2.6 Integrate music player	1	Dec 2 - Milestone 6
3. Streaming Manager		
3.1 Implement manager to accept requests from request manager and direct to player	4	Nov 18 - Milestone 5
3.2 Unit tests	1	Dec 2 - Milestone 6
3.3 Integrate	1	Dec 2 - Milestone 6
4. Playlist Manager		
4.1 Implement playlist, playlist manager, and song classes according to design	1	Nov 18 – Milestone 5
4.2 Unit test	0.5	Nov 18 – Milestone 5
4.3 Integrate	0.5	Dec 2 – Milestone 6

5. Request Handler		
5.1 Set up simple server that detects requests and returns simple response	3	Nov 18 - Milestone 5
5.2 Generate more complex interface for server to allow song requests, feedback submission, and song list request	2	Nov 18 - Milestone 5
5.3 Implement streaming music requests	4	Dec 2 - Milestone 6
5.4 Unit tests	1	Dec 2 - Milestone 6
5.5 Integrate	1	Dec 2 - Milestone 6
6. Database Interface		
6.1 Create program to set up database	1	Jan 11 - Milestone 7
6.2 Create interface and querying classes for database	1	Jan 11 - Milestone 7
6.3 Unit tests	1	Jan 11 - Milestone 7
6.4 Integrate	2	Feb 17 - Milestone 9
7. Intelligent Playlist Generator		
7.1 Implement simple generator based on hard coded preference	2	Jan 11 - Milestone 7
7.2 Make weights or evaluation functions configurable	1	Jan 11 - Milestone 7
7.3 Unit tests	1	Jan 11 - Milestone 7
7.4 Create playlist generator UI for configuration	3	Feb 17 - Milestone 9
7.5 Integrate	2	Feb 17 - Milestone 9
8. Statistics and Reports		
8.1 Implement according to design	2	Jan 11 - Milestone 7
8.2 Create unit tests	1	Jan 11 - Milestone 7

8.3 Integrate to get data from other modules	1	Jan 11 - Milestone 7
8.4 Create image/report viewing UI/Viewer	3	Feb 17 - Milestone 9
8.5 Integrate UI	2	Feb 17 - Milestone 9
9. Phone Application		
9.1 Create simple program to send requests to request module and process feedback	2	Mar 5 - Milestone 10
9.2 Extend to display song list; allow selection of song and sending of request; extend to send user feedback	2	Mar 5 - Milestone 10
9.3 Implement remaining features along with desired UI (assuming it was designed)	5	Mar 19 - Milestone 11
9.4 Unit tests	2	Mar 19 - Milestone 11
10. User Interface		
10.1 Create main menu interface	4	Mar 5 - Milestone 10
10.2 Unit test	1	Mar 5 - Milestone 10
11. Integrate & Test		
11.1 Integrate all components into main menu	4	Mar 19 - Milestone 11
11.2 Improve overall integration	21	Apr 26 - Milestone 13
11.3 Test fully integrated software	21	Apr 26 - Milestone 13

Milestones and Deliverables

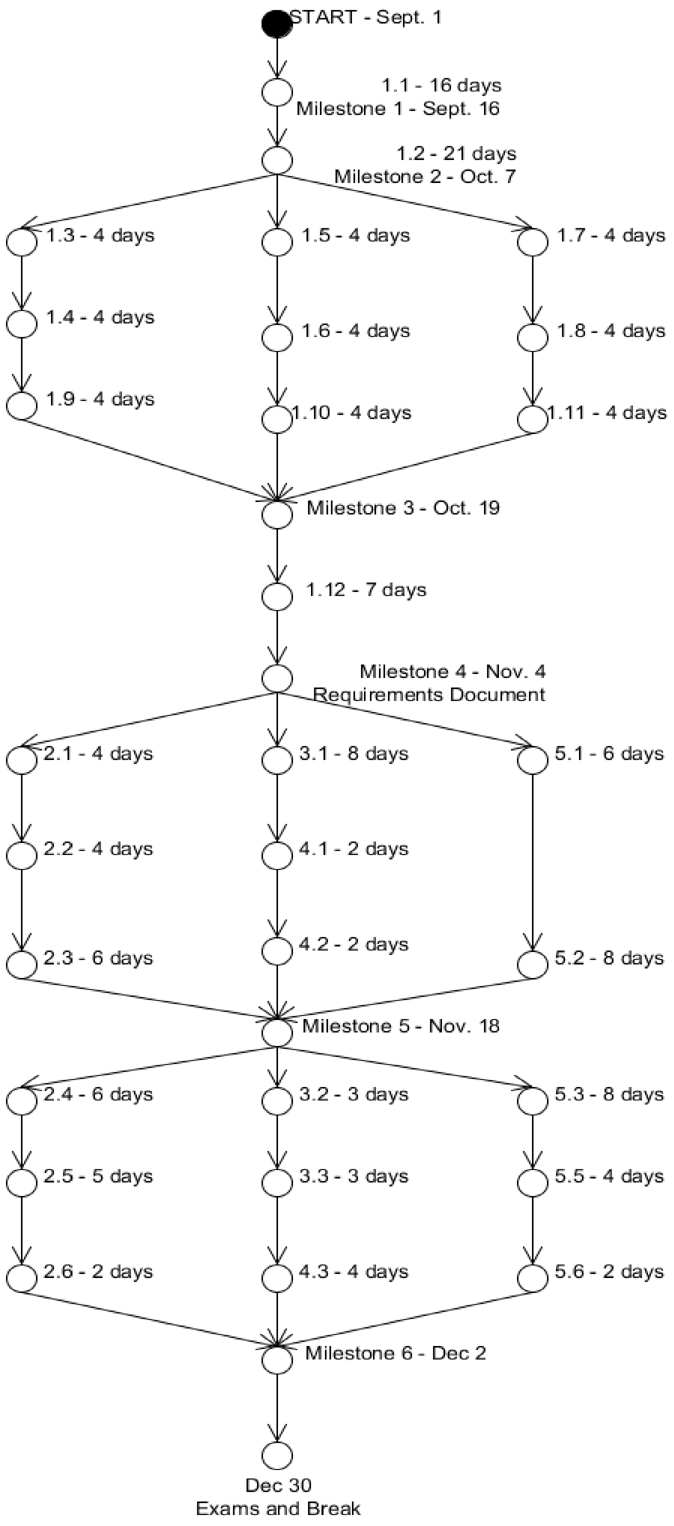
	Title	Result	Date	Dependencies	Deliverables
1	Group Formation	Group formed and project determined	09/16/11	M1	1. Project summary
2	Preliminary Design	Create Preliminary Design Document	10/07/11	M1	2. Preliminary Design
3	Research	Research necessary libraries and software for modules; decide upon most requirements for implementation	10/21/11	M1	
4	Requirements Document	Create Requirements Document	11/04/11	M1, M3	3. Requirements Document
5	Primary Modules I	Start music player, streaming manager, playlist manager and request handling modules	11/18/11	M1, M3	
6	Primary Modules II	Finish music player, streaming manager, and request handling modules	12/02/11	M1, M3, M5	
7	Secondary Modules I	Start statistics, preferences, and AI modules	01/11/12	M1, M3	
8	Design document	Create design document	01/27/12	M1, M3, M4	4. Design Document
9	Secondary Modules II	Finish statistics, preferences, and AI modules; implementation document	02/17/12	M1, M3, M5, M6, M7	5. Implementation Document

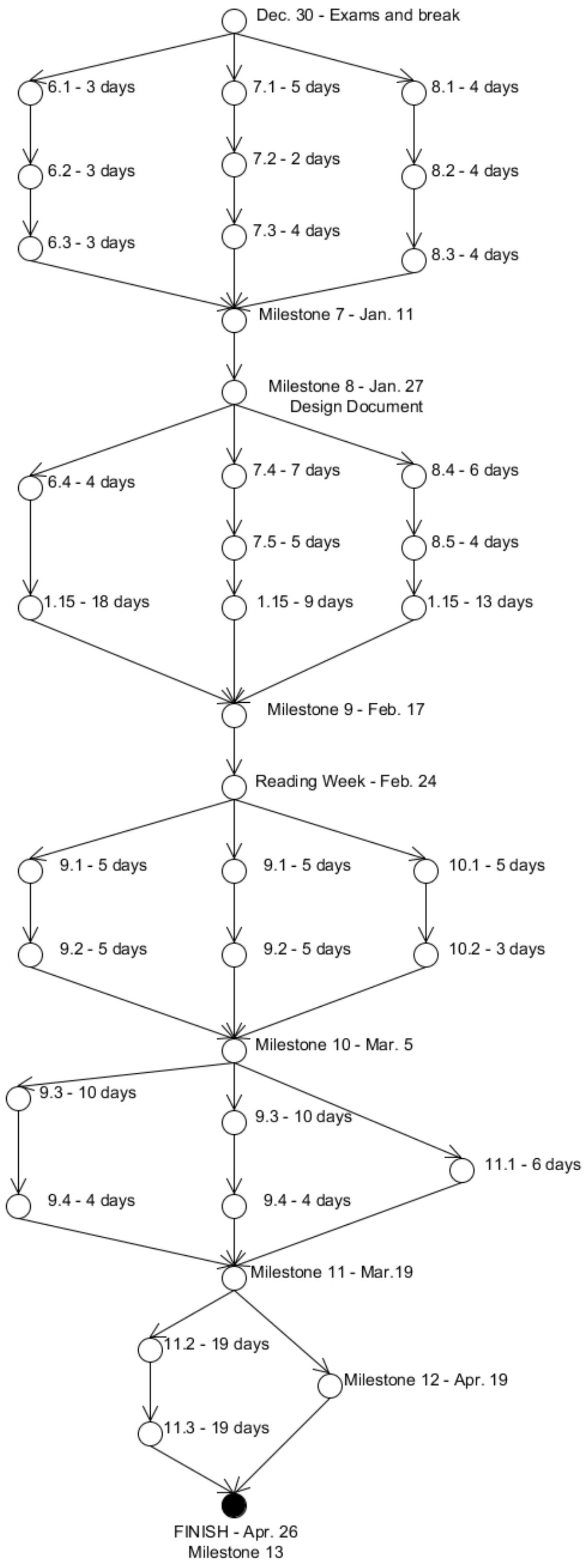
10	Phones and UI	Start phone applications and create main UI	03/05/12	M1, M3	6. Milestone Demo
11	Phones and Integrate	Finish phone applications and complete integration	03/19/12	M1, M3, M5, M6, M7, M9, M10	
12	Presentation	Class presentation	04/09/12	M1, M3, M5, M6, M7, M9, M10, M11	7. Class Presentation
13	Integrate and Test	Integrate and test; hand in final project	04/26/12	M1, M3, M5, M6, M7, M9,M10,M11	8. Application Acceptance Test

Meetings

Meetings with Joseph Vybihal, will ideally take place within a few days of each milestone.

DATED ACTIVITY GRAPH




Dec. 30 - Exams and break
6.1 - 3 days
7.1 - 5 days
8.1 - 4 days
6.2 - 3 days
7.2 - 2 days
8.2 - 4 days
6.3 - 3 days
7.3 - 4 days
8.3 - 4 days
Milestone 7 - Jan. 11
Milestone 8 - Jan. 27
Design Document
6.4 - 4 days
7.4 - 7 days
8.4 - 6 days
7.5 - 5 days
8.5 - 4 days
1.15 - 18 days
1.15 - 9 days
1.15 - 13 days
Milestone 9 - Feb. 17
Reading Week - Feb. 24
9.1 - 5 days
9.1 - 5 days
10.1 - 5 days
9.2 - 5 days
9.2 - 5 days
10.2 - 3 days
Milestone 10 - Mar. 5
9.3 - 10 days
9.3 - 10 days
11.1 - 6 days
9.4 - 4 days
9.4 - 4 days
Milestone 11 - Mar.19
11.2 - 19 days
Milestone 12 - Apr. 19
11.3 - 19 days
FINISH - Apr. 26
Milestone 13

Resource Requirements

Developer

ID	Resource	Cost
01	Computers with network capabilities – minimum one per programmer.	Approximately $2,000 total yearly upgrade and maintenance cost.
02	Smart devices for development and testing: iPhone or another iOS device, and Android	Approximately $300 total purchase and maintenance price.
03	Internet	$30/mo throughout the duration of the project; approximately $240.

Cost Estimate

Hypothetical Total Cost

Programmers	Hourly Rate	Total Hours	Sum
3	$25	1416	$35,400
		Resources:	$2,540
		Total Cost:	$37,940

Engineering methods

Software Process Model

We are not working with any specific client, and while we do not foresee large developmental changes during the process, we do have several components which will require experimentation and where we will need to make design adjustments. The software is divisible into well-defined modules which can be integrated in stages. Taking this into account along with the dynamics of our team, we will use a modified waterfall process. We will develop each module, sometimes concurrently with others, such that there is an integration point at the completion of each one. If necessary, the integration phase could also involve re-working the module in a loop. Once this is complete, the process will continue to the next stage of the waterfall model. Testing results may also require us to loop to previous steps until everything is functioning correctly.

Programming Development Method

It is crucial that the basic functionality – the client being able to view the server playlist and send the request across the network – is implemented well and tested. This relies mostly on the back-end of the server software, so we will concentrate on this first by using an incremental programming development method. We will develop the modules in a certain order to first build an initial, basic functioning model with very low complexity. This will be followed by a phase of testing and reviewing, and then with each additional module we will add complexity incrementally.

Each milestone phase will always begin with a team meeting. We will discuss the requirements of the next milestone and the corresponding deliverables, look at what has been accomplished in the past phase and if major changes need to be made, then designate the separate work paths that each team member will take for the new phase. Based on our activity graph, our assessment of workloads, and experience, we will divide up the work for any revisions or integrations that remain from the previous phases. We will be in communication throughout the phase, and likely will have several meetings. The phase will end in a team meeting, followed by a period of integration and testing.

Team Organization

We have a project leader but are working together as a group, making adjustments through discussion and group decisions. We will designate modules based on individual expertise of team members, and perform merges either together or individually depending on the complexity involved. The way this functions is similar to that in the SCRUM model, with sprints of relatively short time spans between sub-milestones and regular team meetings at the start of each.

Process Assessment

The assessment of our selected modified waterfall process will involve the following checks:

1. **General complexity increase.** Because our process is a waterfall, and our modules are of reasonable size, we expect the program complexity to increase incrementally as time goes on. If the complexity remains the same for more than two milestones this will indicate that our process should be more noise-dependent, and the current process model is not effective.
2. **Number of loops.** The process is a not a direct waterfall, but we still do not expect to make too many iterations. Our maximum number of loops for a module should not exceed three. We will not count making minor changes as a loop – only changes that require significant re-writing of code, restructuring of classes or methods, or changes in functionality.
3. **Milestones and deliverables.** The process will also be assessed by our ability to follow our dated activity graph, and the planned milestones with their deliverables. Due to the iterative nature of our process, we will account for a certain amount of buffer space around each milestone: approximately +/- a fifth of the time designated for that particular phase. Our assessment will only take place at the end of the project, to allow

our days in excess or deficit to balance each other out. A total of over 20 days in excess will indicate an ineffective process method.

Change Control Method

We have split the work into multiple modules, some of which can be developed concurrently. This is indicated in the *Formal Phased Project Breakdown, p. 8*. Each module is relatively independent, and we will merge them during the integration steps with the help of SVN for Eclipse.

RISK IDENTIFICATION

Our identification of risks is below; all specific action and probability and exposure calculations will follow in further documents. Included is a brief description of how we plan to handle these risks, in general terms.

ID nomenclature: N – network, D – development, S – scheduling, C – client.

ID	Title	Description	Brief handling methods
01N	Network-related security	Because our software is heavily dependent on a local network connection and sending messages across it, it is important that the database of songs and playlists on the local computer are secure, and that clients may only view the pre-selected playlist contents, without having access to the actual files.	We will assume this risk since it is crucial to our functionality. We will focus on this risk while structuring our database, and run unit testing specifically to make sure it is secure.
02D	Development process risk: Streaming	We are not very familiar with this area, and will need to choose some site(s) from which we can stream music, account for potential network problems and deal with them, and take care to respect all copyrights. The other possibility is that research reveals this to be impossible.	The risk will be assumed; extra time has been designated before the third milestone research streaming API's. We will also anticipate the possibility of needing to cut it out, and will prepare other options.
03D	Development process risk: Graphic user interface	Because we are using Java to ensure compatibility with multiple operating systems, our interface is limited to Java libraries. Creating an interface we are satisfied with, that is user-friendly and intuitive, could require research and/or more advanced designing.	Special time has been designated for the fourth milestone to develop UIs. There is also time at the end during final integration and testing where interface polishing can be done.

04D	Development process risk: Server to device communication	Our smart device applications need to be able to locate and communicate with the server across the local network. Problems could include network strength, reliability, identification, and connection between device and server.	This functionality is essential so we will assume this risk, and try to create simple prototypes during the research phase of our project to anticipate these types of problems.
05S	Scheduling related issues; time constraints	It is possible that our estimations of work and effort, and thus the corresponding portions of time allotted to work on phases were not feasible. We are working under several external deadlines, and multiple self-imposed internal ones. Getting too far behind could result in quality and functionality problems.	At this point, we can anticipate only by allowing buffer time for our milestones, which may potentially cut into time we designated for vacation, exam period, etc. We will learn from experience as the project progresses, and if needed will make changes to our activity graph accordingly.
06D	Development process risk: Library limitations	Based on the libraries we choose, development could be inhibited later on in the process. We could also find that our assumptions about multi-platform compatibility for certain libraries are not correct.	We will do our library-related research carefully before the third milestone, but will also need to use buffer time to make changes if large problems arise.
07C	Client feedback	After our initial stage of testing with clients, feedback might indicate that we did not correctly gage user needs and opinions. Time permitting, this would require us to loop back to development stages and make adjustments.	This risk must be assumed; we plan to complete our project as designed for the purpose of the course, and then make extensions to our timeline before releasing it to the public.

Legal Disclaimer

We plan to have our completed software and portable device applications closed-source.

With the streaming option, we will choose methods which respect copyright laws and streaming requirements. As any other music-managing software such as Windows Media Player or iTunes, we disclaim all responsibility related to the source of local music files.