LinkList Process Plan Document

1. Introduction

LinkList is an Android application intended to provide a platform for Streamlined Collaborative Playlists (SCP). Users will join a lobby started from a host device where they will select songs to create a queue to be played through the audio channel of the host.

LinkList Will:

- Allow groups of users to connect together and create a shared playlist.
- Allow users to add songs to a single playlist.
- Allow one host to play songs from the queue.

LinkList Will Not:

- Infringe on musical copyright laws
- Allow illegal media sharing

1.1 Definitions and Acronyms

SCP - In the context of this project, a Streamlined Collaborative Playlist is a music queue that is created dynamically by multiple users across multiple devices, but is played through one host device.

LL - LinkList

LoC- Lines of Code

2. Process Description

2.1 Project Lifecycle

We will be using waterfall development as an overall design choice, but each individual part of the cycle will be coded in small chunks similar to rapid prototyping. We will be making small bits of the project at a time and remaking them until we are happy. Once two parts are complete, integration will start. Integration will happen concurrently with development of other parts of the app.

2.2 Process Activities

First Iteration:

Activity Name	Description	Input Criteria	Output Criteria
Basic System Outline	Discuss and decide on the basic system architecture for the key features of the project	Should include server/client communication, and all basic elements of the application	Each element will have all of the most basic functionalities designed
Basic User Backend Functionality	write program(s) that includes the most basic functions of the user side	This will be based off of the system outline, and should be able to send and receive packets to and from a particular server and to play audio on the device	This completed element(s) will be tested to show accurate two-way server communication and demonstrable audio channel control
Basic Server Functionality	begin server-side coding, set groundwork for creating lobbies and accepting requests	The server should be able to process requests sent to it from devices	Server should accept inputs from devices, and act on simulated inputs to create lobbies

<u>Additional Iterations:</u> (as we are redoing many of these steps, the overall flow of code will follow this diagram, but some steps may be repeated)

Extended User Back-End Functionality	Add more advanced functions for user backend	Implement functions for creating and customizing lobbies, joining lobbies, inviting other users, voting, text integration, and other functions	Each iteration will provide new testable functions, and improve upon previous functions
Extended Server Functionality	Add more advanced functions for the server	Implement functions for accepting server customization, finding music, organizing playlists, accepting votes, handling duplicate requests, dynamic resource allocation for lobbies, and other functions	Each iteration will provide new testable functions, and improve upon previous functions
UI Development	Create and iterate a user interface to interact with the user back-end code	Add a front-end to run on devices to safely interact with back-end code and utilize user functions	UI will be easy and intuitive to use, and will give users access to all backend functions that they need without providing access to unsecure back-end methods
Integration	As elements are completed, they will be tested for integrated functionality	Given iterations from server, back-end, and UI, test these together to see that all communication happens smoothly and as expected	Each iteration of server, back-end, and UI will be shown to function together, and any bugs will be documented for removal in future iterations

3. Roles

3.1 Team member names

Steven Gurney, George Puliafico, Vincent DiBlasio, Christopher Bartoli

3.2 Roles Table

Role	Responsibility
Project Planner	Must understand how all parts of the project fit together.
Time Manager	Must allocate personnel to appropriately use time, increasing time efficiency through parallelization.
Lead UI	Designs the UI and has final say on front-end design decisions.
Historian	After each progression in rapid prototyping, must gather each individual's ideas for improvement for use in next prototype.
User back-end Coder	Codes primarily on systems used in-app.
Server Coder	Codes primarily on server-use systems.
Integration Tester	Makes sure all individual prototypes work well together.

3.3 Role Assignment Table

People:	Roles:
George P.	Project Planner, Time Manager, Server Coder
Vincent D.	Lead UI, Integration Tester, User back-end Coder
Chris B.	Historian, User back-end Coder
Steve G.	Integration Tester, Server Coder

*Each individual will be responsible for testing code in their unique environment before submitting to larger project

4. Estimates

4.1 Effort Estimate

-Lines of code: We expect to have about 6 different modules of code working together, and expect each module to both integrate with at least two others, but also use about 500-1500 LoC each. With each module's integration code to be about 50 LoC each. Overall this should add to about 9550 LoC. With about 550 of that being code to help modules correctly send/receive information.

-Defects: We expect to encounter 4 bugs per 115 lines of code, and expect to find an extra 1 bug per 50 LoC of integration code. Overall that should be about 78 bugs from standard code plus 11 bugs in integration. Overall, about 89 bugs seems like a good first estimate.

-Effort Hours: We expect to spend 2 hours of planning and writing per 100 lines of code, and 1 hour on debugging for every 2 bugs. Thus with 9550 LoC and 89 bugs, it should take about 191 Hours of work on Code, and 44.5 Hours on debugging, for a total of about 235.5 hours of work on average, per prototype, and we estimate 45% time improvement per prototype, and 3 prototypes total, equaling 235.5 + 129.5 + 71 = 436 Hours of effort total, or 109 effort hours each. ~4.54 person days each.

4.2 Schedule

#	Task	Estimated Effort (man-hours)	Start Date	End Date	Person/ People	Actual Effort (man-hours)
S1	Design Server Side Architecture	150	9/27/16	10/20/16	GP,SG	134
S2	Initial Socket to Socket Connection	45	10/20/16	11/11/16	GP,SG	38
S3	Setup Google Cloud Server	5	11/11/16	11/12/16	GP,SG	8
B1	Client Connection to Server Lobbies/Client Back-end	75	11/11/16	11/21/16	CB,VD	92
G1	Barebones GUI	75	9/27/16	10/19/16	VD,CB	20
I1	Integration of Prototype client and Server	120	10/19/16	12/4/16	VD,SG,GP, CB	137
B2	Host End Spotify Integration	180	11/11/16	12/4/16	CB,VD	160