**Software Design Document**

Project: Senior Project - DJukebox

Author: Alexander Mitchell

Advisor: Franz Kurfess

**Introduction**

* Purpose

The purpose of this project is to investigate a mobile application to handle disk jockey requests by patrons in various settings. The mobile application will solve problems relating to this and serve as a testament to my knowledge and skills in the field of computer science.

* Scope

This project will consist of the research, design, and development of a mobile application on the Android platform. This mobile application will be split into two sides or versions; one for the Disk Jockeys (often called the server) and one for the patrons (often called the client). These two versions will interact with each other offer the application’s functionality. The mobile application will not be published to the Google Play store. The project has a single researcher, designer, developer, and tester; that is, the team consists of only Alexander Mitchell.

* Definitions, Acronyms, Abbreviations
  + Definitions
    - Client side/application/version: Refers to the portion of the application that is used by patrons and the general public, all non-disk jockey users. This is the part of the application that makes requests.
    - Server side/application/version: Refers to the portion of the application that is used by disk jockeys, karaoke jockeys, party and event hosts, etc. This is the part of the application that takes requests.
  + Acronyms
    - DJ: Disk Jockey
  + Abbreviations
    - DJ: Disk Jockey
    - App: application, as in mobile application

**Design Overview**

* Description of Problems
  + DJ’s have a hard time hearing requests.
  + DJ’s generally find taking requests annoying.
  + Patrons do not know if a DJ heard their request.
    - Even if the request was heard, patrons don’t know if the request will be accepted or rejected.
* Technologies Used
  + Platform

This project will be created for the Android Mobile platform only.

* + Development Environment

The Android Studio development environment will be used to do all the research as well as the development for this project.

* + Testing

Testing will be done on the Android device emulators available on Android studio as well as physical Android devices. The actual Android devices used will be listed here when they are acquired.

* Testing Devices Used
  + Various different devices on the Android Studio Device Emulator as previously stated
  + A borrowed Motorola Droid running API 16
    - Note: Project had to be altered to target this older API in order to test on this device, as such it was much less aesthetically pleasing on it.
* System Architecture
  + Backend
    - Real-time database system hosted on Google servers called Google Firebase.
  + Structure is a tree as follows
  + ROOT: DJukebox
    - Users
      * <Username1>
        + Bio

Name: <DJ Name>

Venue: <Venue Name>

EndTime: <End Time of Current Session>

Paragraph: <Paragraph of biography of DJ>

Password: <Password of this user>

* + - * + RequestList

Request0

Artist: <Request’s Artist>

Song: <Song Requested>

VersionRemix: <Version/Remix of Song>

Status: <Accepted | Rejected | Created>

Request1

…

…

* + - * <Username2>
        + …
      * …
* System Operation
  + Deploys on any Android device running Android version 17 or newer.

**User Interface Design**

* Client
  + See sketches and images in appendix.
  + Client side referred to as “Patron” in graphical user interface.
* Server
  + See sketches and images in appendix.
  + Client side referred to as “DJ” in graphical user interface.

**Data Model and Storage**

* Data Objects
  + DJ
    - Object to represent a user, specifically a user, and their profile and associated session
    - Fields:
      * name: A string of the name the DJ would like displayed on their session
      * venue: A string of the venue the DJ is performing at during the session
      * endTime: A string of the time the DJ’s current session ends at
      * paragraph: A string of a paragraph to describe the DJ; a biography paragraph
      * key: A string of the key of this DJ’s node in the database; its username
      * password: A string of the password for this DJ’s login
    - Defined Values:
      * NOT\_IN\_SESSION\_VENUE: A code string to indicate that a DJ is not currently in session to be stored as the venue when a DJ is not in session, used in combination with NOT\_IN\_SESSION\_ENDTIME.
      * NOT\_IN\_SESSION\_ENDTIME. A code string to indicate that a DJ is not currently in session to be stored as the end time when a DJ is not in session, used in combination with NOT\_IN\_SESSION\_VENUE
      * BIO: A string to represent the key of the Bio nodes in the database.
      * END\_TIME: A string to represent the key of the endTime nodes in the database.
      * NAME: A string to represent the key of the name nodes in the database.
      * PASSWORD: A string to represent the key of the Password nodes in the database.
      * VENUE: A string to represent the key of the Venue nodes in the database.
      * USERS: A string to represent the key of the Users nodes in the database.
      * PARAGRAPH: A string to represent the key of the Paragraph nodes in the database.
  + Song Request:
    - Object to represent a song request and its associated status
    - Fields:
      * artist: A string of the name of the requested song’s artist.
      * song: A string of the name of the song requested.
      * versionRemix: A string of the specific version or remix of the song that is being requested.
      * status: An enumerator of the current status of this request; accepted, rejected, or created
      * key: A string of the key of this song request nodes in the database; “request<position in list of requests>”.
    - Defined Values
      * REQUEST\_LIST: A string to represent the key of the RequestList node in the database.
      * REQUEST: A string to represent the key, minus the position, of the node of each request in the database.
      * ARTIST: A string to represent the key of the Artist nodes in the database.
      * SONG: A string to represent the key of the Song nodes in the database.
      * STATUS: A string to represent the key of the Status nodes.
      * VERSION\_REMIX: A string to represent the key of the VersionRemix nodes in the database.
      * accepted: A string to represent that accepted status value in the database.
      * rejected: A string to represent that rejected status value in the database.
      * created: A string to represent that created status value in the database.

**References**

* Sams Teach Yourself Android Application Development Fourth Edition
  + Use this book to learn how to develop Android applications. Use as a reference during development as well.