**Cloud Based Music Application**

Software Design Document

Name (s): Sahil Narula (12BCE0416), Poorva Arora (12BCE0277)

Submitted to: Prof. Lydia Jane

Date: 21-February-2015

**TABLE OF CONTENTS**

[1. INTRODUCTION 1](#_Toc412194340)

[1.1 Purpose 1](#_Toc412194341)

[1.2 Scope 2](#_Toc412194342)

[1.3 Overview 2](#_Toc412194343)

[1.4 Reference Material 2](#_Toc412194344)

[1.5 Definitions and Acronyms 2](#_Toc412194345)

[2. SYSTEM OVERVIEW 2](#_Toc412194346)

[3. SYSTEM ARCHITECTURE 2](#_Toc412194347)

[3.1 Architectural Design 2](#_Toc412194348)

[3.2 Decomposition Description 3](#_Toc412194349)

[3.3 Design Rationale 3](#_Toc412194350)

[4. DATA DESIGN 3](#_Toc412194351)

[4.1 Data Description & Data Dictionary 3](#_Toc412194352)

[5. COMPONENT DESIGN 3](#_Toc412194354)

[6. HUMAN INTERFACE DESIGN 4](#_Toc412194355)

[6.1 Overview of User Interface 4](#_Toc412194356)

[6.2 Screen Images 4](#_Toc412194357)

[7. REQUIREMENTS MATRIX 4](#_Toc412194359)

[8. APPENDICES 5](#_Toc412194360)

# INTRODUCTION

## Purpose

The purpose of the Detailed Design Document is to define the detailed design for all components of the **Cloud Based Music Application** which are specified in the [SRS]. The low-level components are designed, coded and tested

## Scope

The application implements server based optimal storage for the playlist of different users. The application back-end is able to execute multiple evented IO using a single thread. By hiding the complexity of the technology and providing users fast and efficient platform to access their playlist from the server, the system will be easy to use.

## Overview

This DDD is organized as follows. Chapter 2 of this document is a short introduction to the general context of the application to be made and to the background of this project. Chapter 3 describes the overall architecture and all the components in a consistent way. Chapter 4 consists of a dictionary for related keywords. Chapter 5 consists of Various components description for the project. Chapter 6 includes the User Interface for the project

## Reference Material

*This section is optional.*

**CLOUD COMPUTING**: <http://en.wikipedia.org/wiki/Cloud_computing>

**NODEJS**: <http://nodejs.org/>

**ANGULARJS**: <https://angularjs.org/>

**SAILSJS:** <http://sailsjs.org>

**EXPRESSJS:** <http://expressjs.com>

**MONGOOSE CONNECTOR**: <http://mongoosejs.com>

**MONGODB**: <http://www.mongodb.org/>

## Definitions and Acronyms

|  |  |
| --- | --- |
| NODEJS | Server side scripting language in javascript |
| ANGULARJS | Client side javascript MVC framework |
| EXPRESS | Nodejs MVC framework for web |
| SAILSJS | Framework built over express, waterline orm, sockets to increase workflow productivity. |
| MONGODB | Database that stores data in form of JSON also called as BSON |
| MONGOOSE | Is a connector for mongodb. ODM middleware for express |

|  |  |
| --- | --- |
| Req | Req given to node web server |
| Res | Response provided by server controllers |
| Module.export | Used to export a function in node application |
| Module | Used to define angularjs module |
| $routerProvider | Provides routes to angular application |
| Require() | Used to require a npm module |
| Node\_modules | Folder consisting of all of the node dependencies for the project |
| Config.js | Configuration file for different node environments |
| NODE\_ENV | Describes the node environment currently working on |
| $locationProvider | Angular dependency for changing the url location |
| $http | Angular dependency to do ajax operations |
| $resource | Angular dependency to connect to rest api optimally. |
| Socket | Consists of node websockets to provide real-time communication from backend to frontend |

# SYSTEM OVERVIEW

The project is divided into two modules frontend and backend. Frontend consists of all the assets to improve user experience. Back-end consists of restful apis to provide optimal routes to send data back and forth. Both frontend and backend are further divided into sub modules. Frontend application consists of various assets and client side controllers which includes resources to connect to remote rest apis. Backend consists of sub modules of various controllers and models for database and also consists of a separate node\_modules folder which consists of application dependencies listed in package.json file.

# SYSTEM ARCHITECTURE

## Architectural Design

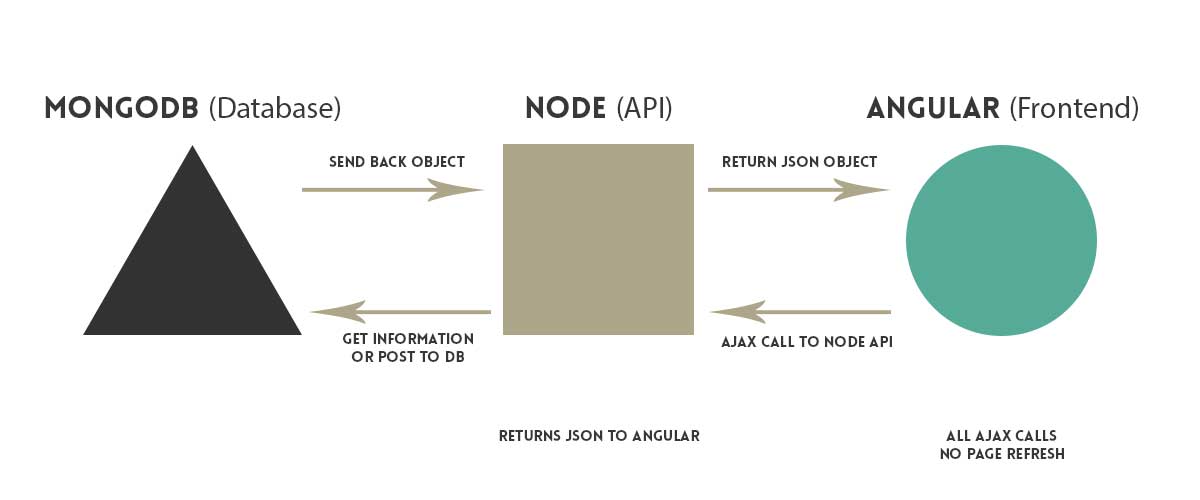
This includes the higher level overview of the modules used in the application.

Each section in the frontend is assigned a controller which is defined in angular app, Each controller has its scope to list data to the user. Scope of one controller doesn’t affect scope of other controllers. Apart from scope application also has rootScope which consistes of operations common to all views in the application. Each controller connects to the backend rest apis using $resource. $resource is a dependency injection to angular application using ng-resource. $resource consists of http requests like get, post, put, delete. Apart from http application also connects to the backend using sockets to provide real-time buffering of the songs.

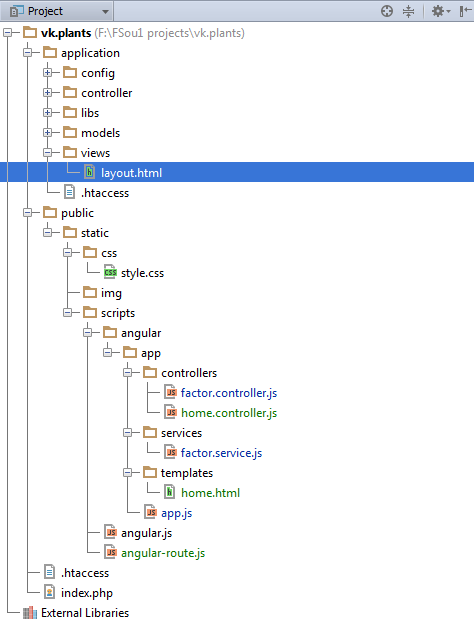
Backend has a similar structure which consists of different node modules that are injected to the application using require() method. Structure of the backend follows controllers, views, models. Controllers provide overall functionality happening in the backend and also sending json data back and forth. The data is stored in mongodb. It consists of various environments to work with. Development environment consists of all the configuration for localhost and production environment consists all the configurations for azure server.

## Decomposition Description

Following images provide the overall structure of the application



Overall folder structure is



## Design Rationale

There are various alternatives to nodejs. The only reason to choose nodejs is because it runs on single thread. And the asynchronous design of nodejs provides blazingly fast speed. Nodejs provides connectors for sql but mongodb is javascript based orm which is compiled by the common V8 engine. So the whole stack is based on javascript which removes the overhead from the server to compile different functions to deliver one webpage. The stack consists of MEAN. MEAN – Mongo, express, angular, node.

# DATA DESIGN

## Data Description & Data Dictionary

Various collections for the data storage are:

|  |  |  |
| --- | --- | --- |
| User | Consists of user entities to uniquely identify the user while authentication | Email, username, password, updated\_at, created\_at |
| Songs | Consists of the online repository of the songs. | songname, metadata, genreid, categoryid, filelocation, updated\_at, created\_at |
| Genre | Consists of various genre categories. | Genrename, genreid, updated\_at, created\_at |
| Uploads | Consists the track of various uploads done by either the user or the admin | Filelocation, filename, uploaderid, admin, created\_at, updated\_at |
| Still working on it while implementation | | |

# COMPONENT DESIGN

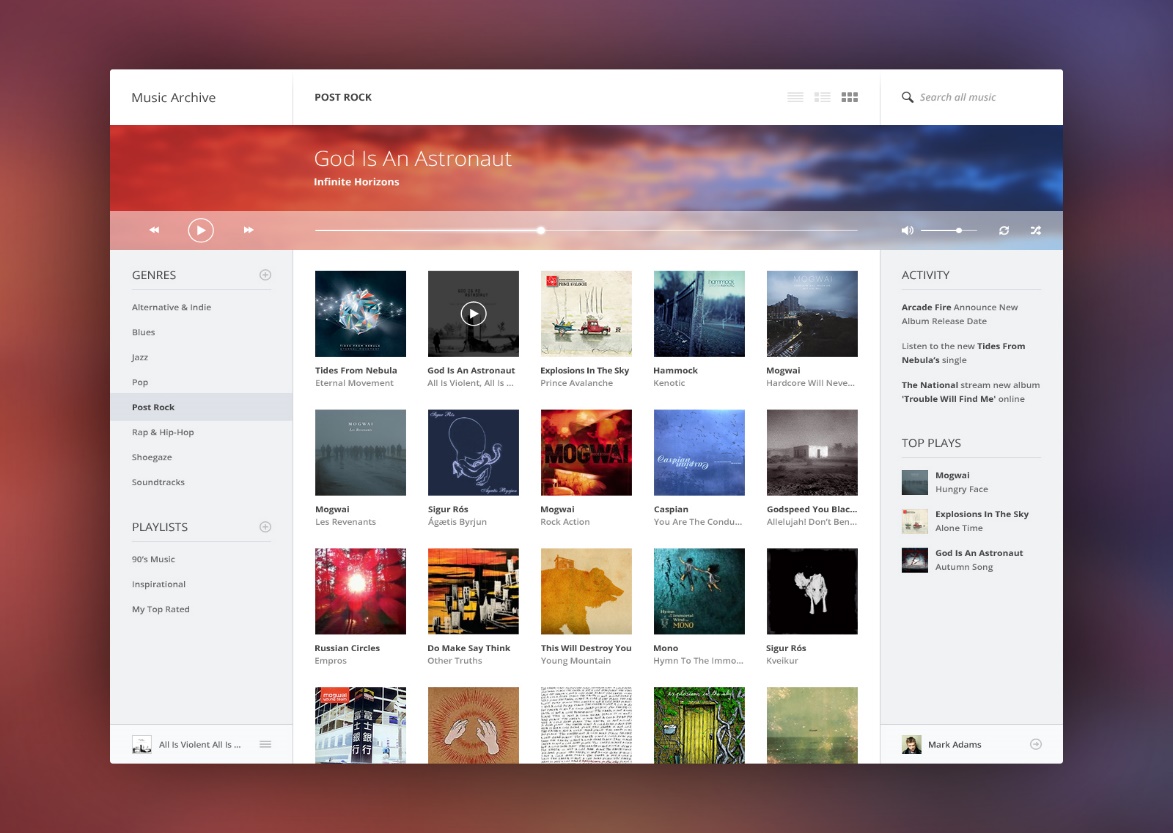
In this section, we take a closer look at what each component does in a more systematic way. If you gave a functional description in section 3.2, provide a summary of your algorithm for each function listed in 3.2 in procedural description language (PDL) or pseudocode. If you gave an OO description, summarize each object member function for all the objects listed in 3.2 in PDL or pseudocode. Describe any local data when necessary.

# HUMAN INTERFACE DESIGN

## Overview of User Interface

Frontend of the applications is being implemented on the best technology stack to improve the user experience of the user. Following this section consists of screen image of the expected outcome of the application.

## Screen Images



# REQUIREMENTS MATRIX

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Local Playlist access | Remote Playlist access | Song Upload | Remote files | Local files |
| Owner |  |  |  |  |  |
| Admin |  |  |  |  |  |
| First time User |  |  |  |  |  |
| Returning User |  |  |  |  |  |
| Loggedin user |  |  |  |  |  |

# APPENDICES

*Nodejs:* [*http://nodejs.org*](http://nodejs.org)

*Angularjs:* [*http://angularjs.org*](http://angularjs.org)

*Mean-stack:* [*http://meanjs.io*](http://meanjs.io)

*Expressjs:* [*http://expressjs.com*](http://expressjs.com)

*Sailsjs:* [*http://sailsjs.org*](http://sailsjs.org)

*Cloud Computing:* <http://en.wikipedia.org/wiki/Cloud_computing>

*Mongoose connector:* <http://mongoosejs.org>

*Mongodb:* http://mongodb.com