# WKLOG0010-6

The World Internet Security Company

**PINNA SERVER**

Date: September 2014  
Version: 1.2  
Authors: WISeKey SA

Table of Contents

I. Overview 5

1.1. Introduction 5

1.2. Scope of work 5

II. Data Model 5

2.1. User Information 5

2.2. Music Information 6

2.3. Description 7

III. REST API design 7

3.1. Request Format 7

3.2. Response Format 7

3.3. Http Response 8

3.4. HTTP Response Status codes 8

IV. REST API References 8

4.1. User 8

4.1.1. Create User (DONE) 8

4.1.2. Login (DONE) 9

4.1.3. Logout (DONE) 9

4.1.4. Get User (DONE) 9

4.1.5. Get User by ID (DONE) 10

4.1.6. Update User (DONE) 10

4.1.7. Change Password (DONE) 11

4.1.8. Get Followers (DONE) 11

4.1.9. Follow (DONE) 11

4.1.10. Unfollow (DONE) 12

4.1.11. Followed 12

4.1.12. Add PINNA coins (DONE) 12

4.2. Music 13

4.2.1. Upload track (DONE) 13

4.2.2. Get track (DONE) 13

4.2.3. Delete track (DONE) 14

4.2.4. Like a track (DONE) 14

4.2.5. Unlike a track (DONE) 15

4.2.6. Is liked a track (DONE) 15

4.2.7. Download track (DONE) 15

4.2.8. Purchase track (DONE) 16

4.2.9. Update track (DONE) 16

4.2.10. List genres (DONE) 16

4.2.11. List moods (DONE) 17

4.2.12. Search tracks by name (DONE) 17

4.2.13. List playlists of user (DONE) 18

4.2.14. Create playlist (DONE) 18

4.2.15. Get playlist’s tracks (DONE) 18

4.2.16. Delete playlist (DONE) 19

4.2.17. Edit playlist name (DONE) 19

4.2.18. Add tracks to playlist (DONE) 20

4.2.19. Remove tracks from playlist (DONE) 20

4.2.20. List all tracks in My Library (DONE) 20

4.2.21. List all the stations in My Stations (DONE) 21

4.2.22. List all the stations in a user’s My Stations by UserID (DONE) 21

4.2.23. List trending stations (DONE) 22

4.2.24. Create station (DONE) 22

4.2.25. Get station detail by ID (DONE) 23

4.2.26. List all tracks of a station (DONE) 23

4.2.27. Update station (name + photo) (DONE) 24

4.2.28. Delete station (DONE) 25

4.2.29. Subscribe a station OR Add a station to My Stations(DONE) 25

4.2.30. Unsubscribe from a station OR Remove a station from My Stations(DONE) 25

4.2.31. Search stations by name (DONE) 26

4.2.32. List all tracks in My Tracks (DONE) 26

4.2.33. Get Ads (DONE) 27

V. Update Notes 27

5.1. 2014/09/19 27

5.2. 2014/09/30 27

5.3. 2014/10/03 27

5.4. 2014/10/07 28

5.5. 2014/10/08 28

5.6. 2014/10/20 28

5.7. 2014/10/21 28

5.8. 2014/10/22 28

5.9. 2014/10/24 28

5.10. 2014/10/28 28

5.11. 2014/11/13 29

5.12. 2014/11/18 29

5.13. 2014/11/20 29

# Overview

## Introduction

PINNA is an Internet radio mobile application. This app has two parts: client side (mobile app) and server side. This document is the server side document.

## Scope of work

* The server side will be implemented using Python programming language with Django framework.
* Web services:
  + Use RESTful
  + Profile Information will be stored in MySQL database
  + Media files (display photos, tracks) will be stored in Amazon S3 and accessible via Amazon CloudFront.

# Data Model

## User Information

* User :
  + - Username: string
    - Firstname: string
    - Lastname: string
    - Email: string
    - Password: string
    - Is\_superuser: boolean: Super admin
    - Is\_staff: boolean: Admin
    - Is\_active: boolean: Active user.
    - Last\_login: datetime
    - Date\_joined: datetime
  + Profile:
    - User: Foreign Key: refers to User
    - Display photo: string: URL to Amazon CloudFront
    - Followers: n-n relationship to other Users: people which follow a user.
    - income: float: incoming money amount of a user.
    - MyTracks: Foreign Key: refers to a Library model
    - MyLibrary: Foreign Key: refers to a Library model

Django has two models of User information: default User model and extended Profile model which references to User. We choose the User extended Profile model for server side.

## Music Information

* Track:
  + User: Foreign Key: refers to an User model
  + Name: string
  + URL: string: refers to the track’s Amazon CloudFront URL
  + Genre: Foreign Key: refers to a Music Meta model
  + Mood: Foreign Key: refers to a Music Meta model
  + BPM: integer
  + Hashtags: n-n relationship to Hashtag model: refers to a list of Hashtags
  + Like\_count: integer: number of likes
  + Download\_count: integer: number of downloads
  + Author: String: Author of the track
  + Artist: String: Artist of the track
  + Duration: float: Duration of the track
  + Price: float
* Music Meta model:
  + Name: string
  + Date\_created: datetime
  + Type: string: Genre, Mood
* Hashtag
  + Name: string
  + Date\_created: datetime
* Station
  + User: Foreign Key: refers to an User model
  + Name: string.
  + Genre**:** string
  + Mood**:** string
  + Hahstags**:** string
  + Date\_ created: datetime
  + Tracks: n-n relationship to Track model: refers to a list of Tracks
* Playlist
  + User: Foreign Key: refers to an User model
  + Name: string
  + Date\_ created: datetime
  + Tracks: n-n relationship to Track model: refers to a list of Tracks
* Library
  + Date\_ created: datetime
  + Tracks: n-n relationship to Track model: refers to a list of Tracks

## Description

Genres, Styles, Moods, and BPMs will be predefined by the PINNA admin. There will be a web console for admin to manage (add/remove/edit) these types of music metadata.

When a user is registered successfully, a Profile , a My Library (Library model) and a My Tracks (Library model) will be created for this user. All tracks of the user will go to My Tracks (tracks which user uploads). All purchased tracks of the user will go to My Library. At the beginning My Tracks and My Library will be empty.

When user uploads a track, the client will call the upload api exposed by the server. If uploading is successful, the server will return track\_id and track\_url to the client. Track\_url is an Amazon CloudFront URL.

Name of track/station/playlist must be unique. The server will check the uniqueness when it executes create/update actions.

# REST API design

## Request Format

For POST and PUT requests, the request body must be JSON, with the Content-Type header set to application/json.

In case of uploading file, the Content-Type should be multipart/form-data.

Authentication is done via HTTP headers where:

* The PINNA\_APPLICATION\_ID header identifies which application you are accessing.
* The PINNA\_APPLICATION\_KEY header authenticates the endpoint.
* The PINNA\_AUTHENTICATED\_TOKEN header identifies the logged in user. This value is given by the server via create/login action.

## Response Format

For all requests the response is a JSON object.

Whether a request is successful is indicated by the HTTP status code. A 2xx status code indicates success, whereas 4xx and 5xx status code signifies failure. Response body is always JSON-encoded string and contains the following fields:

* + - code: 1 OR 0: indicates whether the request is successful or failed
    - msg\_code: a predefined message code for a specific message. I.e: *user\_found* for “*User is found”*. This code will be used for customizing message on the client side in the case that the client side doesn’t want to show the server message.
    - message: message content. I.e: *User is found*
    - Other fields: token, track\_url …

Example:

*{*

*“code”:0,*

*“msg\_code”:”user\_not\_found”,*

*“message”:”User is not found”,*

*}*

## Http Response

Http Response status must always be set. Below is an example:

*Status: 201 Created*

*Location: https://api.parse.com/1/users/g7y9tkhB7O*

Where:

* 201: Http Status value;
* Location: Endpoint of the new object created such as a USER object, a COMMENT, etc.

For API calls that require user-authentication, the 401 status should be used for un-authenticated API call.

## HTTP Response Status codes

* 200: OK <no error>
* 400: Bad request
* 401: Unauthorization
* 405: Method not allow
* etc …

# REST API References

## User

**Note: msg\_code in the response will be lowercase. All request parameters must be in lowercase.**

### Create User (DONE)

* Description: This method will create new user on server
* Service URL: **https://{BaseURL}/api/user/create**
* HTTP Method: **POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* Request parameters:
* Display\_name
* email
* password
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Failed
    - Email\_invalid
    - User\_existed
    - Empty\_fields (client should check this)
  + Message:
  + Token: Authenticated token
  + Userid
* Sample call using curl:

### Login (DONE)

* Description: This method will login user to server
* Service URL: **https://{BaseURL}/api/user/login**
* HTTP Method:**POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* Request parameters:
* Username **or** email
* password
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Failed
    - Invalid\_credentials
    - Empty\_fields (client should check this)
  + Message:
  + Token: Authenticated token
  + userid

### Logout (DONE)

* Description: This method will logout user from server
* Service URL: **https://{BaseURL}/api/user/logout**
* HTTP Method: **POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Failed
    - invalid\_auth\_token
  + Message:

### Get User (DONE)

* Description: This method will get user information from server
* Service URL: **https://{BaseURL}/api/user/profile**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - OK
    - invalid\_auth\_token
    - invalid\_user
    - Failed
  + Message:
  + Display\_name
  + Email
  + Photo
  + Userid
  + Num\_followers
  + Num\_followings
  + Amount: Number of PINNA coins of the user

### Get User by ID (DONE)

* Description: This method will get user information from server
* Service URL: **https://{BaseURL}/api/user/<userID>/profile**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - OK
    - invalid\_auth\_token
    - invalid\_user
    - Failed
  + Message:
  + Display\_name
  + Email
  + Photo
  + Is\_followed
  + Num\_followers
  + Num\_followings
  + Amount: Number of PINNA coins of the user

### Update User (DONE)

* Description: This method will update user information
* Service URL: **https://{BaseURL}/api/user/update**
* HTTP Method: **PUT**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters:
* Display\_name
* Photo
* *Note: Empty field means no change.*
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Failed
    - invalid\_auth\_token
  + Message:
  + Display\_name
  + Photo\_url
  + Email

### Change Password (DONE)

* Description: This method will change user passowrd. After changing password, user is logged out and have to login again using new password.
* Service URL: **https://{BaseURL}/api/user/changepassword**
* HTTP Method: **POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters
* Old\_password
* New\_password
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Invalid\_password
    - Ok
    - Empty\_fields
    - Failed
    - invalid\_auth\_token
  + Message:

### Get Followers (DONE)

* Description: This method will get all followers of an user
* Service URL: **https://{BaseURL}/api/user/followers/<userID>**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters:
  + Page: The page number of the results set. Default is 1
  + Per\_page (OPTIONAL): The items number per page. Default is 100
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Invalid\_userid
    - Failed
    - invalid\_auth\_token
  + Message:
  + users: [{user1 details}, {user2 details}, … ]
  + total: Total number of users
  + num\_pages: Number of pages
  + next\_page: Url of next page
  + previous\_page: Url of previous page

### Follow (DONE)

* Description: This method will make an user follow another one.
* Service URL: **https://{BaseURL}/api/user/follow/<UserID>**
* HTTP Method: **POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Invalid\_userid
    - Empty\_fields
    - Failed
    - invalid\_auth\_token
  + Message:

### Unfollow (DONE)

* Description: This method will make an user unfollow a followed user.
* Service URL: **https://{BaseURL}/api/user/unfollow/<UserID>**
* HTTP Method: **POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Failed
    - Invalid\_userid
    - Empty\_fields
    - invalid\_auth\_token
  + Message:

### Followed

* Description: This method will check if an user followed another one.
* Service URL: **https://{BaseURL}/api/user/unfollow**
* HTTP Method: **POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters
* Followed\_user
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - already
    - not\_yet
    - failed
    - invalid\_auth\_token
  + Message:

### Add PINNA coins (DONE)

* Description: This method will increase the number of coins of the current user
* Service URL: **https://{BaseURL}/api/user/add\_coins**
* HTTP Method: **POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters:
  + Purchase\_data: (required): INAPP\_PURCHASE\_DATA from Google In-App Purchase Service. Must be a string in json format.
  + Signature: (required): Google In-app purchase signature for the above purchase
  + Coins: (required): Number of PINNA coins will be added to the user account
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Failed
    - invalid\_auth\_token
    - empty\_fields
    - invalid\_purchase\_data: purchase data is not well formed format
    - order\_existed: The OrderID was used before.
  + Message:
  + Amount: current user number of coins

## Music

### Upload track (DONE)

* Description: This method will create a track in the My Library of user
* Service URL: **https://{BaseURL}/api/music/upload**
* HTTP Method: **POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters
* File
* Name
* Genre <genreid>
* BPM
* Mood <moodID>
* Hashtags: <hashtagName> : “hashtag1, hashtag2, hashtag3”
* Author
* Artist
* Duration
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Name\_existed (Name is existed)
    - Failed
    - Empty\_fields
    - invalid\_auth\_token
  + Message:
  + Url: amazon cloudfront signed url
  + Id: track id in database

### Get track (DONE)

* Description: This method will get track’s information
* Service URL: **https://{BaseURL}/api/music/<trackid>**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Failed
    - invalid\_auth\_token
  + Message:
  + Url: amazon cloudfront url
  + Name
  + Genre <ID>
  + Mood <ID>
  + BPM
  + Hashtags
  + Author
  + Artist
  + Duration
  + Download\_count
  + Like\_count
  + Price
  + Is\_liked: true/false (use for checking if a track is liked by current user)
  + Owner\_id: UserID of the track’s owner
  + Is\_purchased

### Delete track (DONE)

* Description: This method will delete a track
* Service URL: **https://{BaseURL}/api/music/<trackid>**
* HTTP Method: **DELETE**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Failed
    - invalid\_auth\_token
  + Message:

### Like a track (DONE)

* Description: This method will make current user like a track
* Service URL: **https://{BaseURL}/api/music/like/<trackid>**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Failed
    - invalid\_auth\_token
  + Message:

### Unlike a track (DONE)

* Description: This method will make current user unlike a liked track
* Service URL: **https://{BaseURL}/api/music/unlike/<trackid>**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Failed
    - invalid\_auth\_token
  + Message:

### Is liked a track (DONE)

* Description: This method will check if current user liked a liked track
* Service URL: **https://{BaseURL}/api/music/is\_liked/<trackid>**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Liked: Liked
    - Not: Not like
    - Not\_found
    - Failed
    - invalid\_auth\_token
  + Message:

### Download track (DONE)

* Description: This method will increase download count of a track by 1
* Service URL: **https://{BaseURL}/api/music/download/<trackid>**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Failed
    - invalid\_auth\_token
  + Message:

### Purchase track (DONE)

* Description: This method will purchase a track
* Service URL: **https://{BaseURL}/api/music/purchase/<trackid>**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found: track is not found
    - Not\_enough: current user’s amount is not enough to buy this track.
    - Failed
    - invalid\_auth\_token
  + Message:
  + Amount: the current user number of PINNA coins

### Update track (DONE)

* Description: This method will update track information
* Service URL: **https://{BaseURL}/api/music/<trackid>**
* HTTP Method: **PUT**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters:
* Name
* Genre
* BPM <ID>
* Mood <ID>
* Hashtags: : “hashtag1, hashtag2, hashtag3”
* Author
* Artist
* Duration
* *Note: Empty field means not updating*
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Failed
    - invalid\_auth\_token
    - not\_found
  + Message:

### List genres (DONE)

* Description: This method will list all available genres in database
* Service URL: **https://{BaseURL}/api/music/genres**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Failed
    - Not\_found
    - invalid\_auth\_token
  + Message:
  + Genres: [[genre1\_id, genre1\_name], [genre2\_id, genre2\_name]…]

### List moods (DONE)

* Description: This method will list all moods in database
* Service URL: **https://{BaseURL}/api/music/moods**
* HTTP Method: **GET**
* Request parameters:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Failed
    - invalid\_auth\_token
  + Message:
  + Moods: [[mood1\_id, mood1\_name], [mood2\_id, mood2\_name]…]

### Search tracks by name (DONE)

* Description: This method will search tracks by name
* Service URL: **https://{BaseURL}/api/music/search**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters:
  + Search\_term
  + Page: The page number of the results set
  + Per\_page: The number of items per page (OPTIONAL). Default is 100
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Empty\_fields
    - Not\_found
    - Failed
    - invalid\_auth\_token
  + Message:
  + Tracks: [{track1 details}, {track2 details} …]
  + total: Total number of tracks
  + num\_pages: Number of pages
  + next\_page: Url of next page
  + previous\_page: Url of previous page

### List playlists of user (DONE)

* Description: This method will list all playlists of a user
* Service URL: **https://{BaseURL}/api/music/playlists**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - not\_found
    - failed
    - invalid\_auth\_token
  + Message:
  + Playlists: [{‘id’:’xxx’, ‘playlist\_name’: ‘xxx’, ‘num\_tracks’:’xxx’} …]

### Create playlist (DONE)

* Description: This method will create a playlist
* Service URL: **https://{BaseURL}/api/music/playlist/create**
* HTTP Method: **POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters:
* Playlist\_name
* Tracks: [track1\_id, track2\_id …] //*This can be empty*
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Name\_existed
    - empty\_fields
    - Failed
    - invalid\_auth\_token
  + Message:
  + Playlist\_id

### Get playlist’s tracks (DONE)

* Description: This method will list all tracks of a playlist
* Service URL: **https://{BaseURL}/api/music/playlist/<playlistID>**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters:
  + Page: The page number of the results set
  + Per\_page (OPTIONAL): The items number per page. Default is 100
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Invalid\_playlist
    - Failed
    - invalid\_auth\_token
  + Message:
  + tracks: [{track1 details}, {track2 details} …]
  + total: Total number of tracks
  + num\_pages: Number of pages
  + next\_page: Url of next page
  + previous\_page: Url of previous page

### Delete playlist (DONE)

* Description: This method will delete a playlist
* Service URL: **https://{BaseURL}/api/music/playlist/<playlistID>**
* HTTP Method: **DELETE**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Invalid\_playlist
    - Failed
    - invalid\_auth\_token
  + Message:

### Edit playlist name (DONE)

* Description: This method will update a playlist’s information (name)
* Service URL: **https://{BaseURL}/api/music/playlist/<playlistID>**
* HTTP Method: **PUT**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters:
* Playlist\_name
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Invalid\_playlist
    - Name\_existed
    - Failed
    - invalid\_auth\_token
  + Message:

### Add tracks to playlist (DONE)

* Description: This method will add a track to a playlist
* Service URL: **https://{BaseURL}/api/music/playlist/<playlistID>/add**
* HTTP Method: **POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters
* Tracks: [track\_id1, track\_id2, …]
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - invalid\_playlist
    - Failed
    - invalid\_auth\_token
  + Message:

### Remove tracks from playlist (DONE)

* Description: This method will remove a track from a playlist
* Service URL: **https://{BaseURL}/api/music/playlist/<playlistID>/remove**
* HTTP Method: **POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters:
* Tracks: [track\_id1, track\_id2, …]
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - invalid\_playlist
    - Failed
    - invalid\_auth\_token
  + Message:

### List all tracks in My Library (DONE)

* Description: This method will list all tracks in My Library
* Service URL: **https://{BaseURL}/api/music/mylibrary/tracks**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request Parameters:
  + Page: The page number of the results set. Default is 1
  + Per\_page (OPTIONAL): The items number per page. Default is 100
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found (empty)
    - Failed
    - invalid\_auth\_token
  + Message:
  + tracks: [{track1 details}, {track2 details} …]
  + total: Total number of tracks
  + num\_pages: Number of pages
  + next\_page: Url of next page
  + previous\_page: Url of previous page

### List all the stations in My Stations (DONE)

* Description: This method will list all stations in user’s My Stations
* Service URL: **https://{BaseURL}/api/music/stations**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request Parameters:
  + Page: The page number of the results set. Default is 1
  + Per\_page (OPTIONAL): The items number per page. Default is 100
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Failed
    - invalid\_auth\_token
  + Message:
  + Stations: [{‘id’:xx, ‘name’:’xxx’, ‘num\_tracks’:xx, ‘genre’:[x,y,z], ‘mood’:[x,y,z]}, ‘author’:’display name’, ‘recent\_num\_subscribers’:xx, ‘photo’: ‘photo url’, ‘privacy\_level’:’0 OR 1’, ‘bpm’: ‘xx to yy’, ‘hashtags’: [x, y, z …], ‘owner\_id’:xx, ‘is\_subscribed’:’true or false’} ….]
  + total: Total number of stations
  + num\_pages: Number of pages
  + next\_page: Url of next page

### List all the stations in a user’s My Stations by UserID (DONE)

* Description: This method will list all stations in a user’s My Stations by userID
* Service URL: **https://{BaseURL}/api/music/station/my\_stations/<userID>**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request Parameters:
  + Page: The page number of the results set. Default is 1
  + Per\_page (OPTIONAL): The items number per page. Default is 100
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Failed
    - invalid\_auth\_token
  + Message:
  + Stations: : [{‘id’:xx, ‘name’:’xxx’, ‘num\_tracks’:xx, ‘genre’:[x,y,z], ‘mood’:[x,y,z]}, ‘author’:’display name’, ‘recent\_num\_subscribers’:xx, ‘photo’: ‘photo url’, ‘privacy\_level’:’0 OR 1’, ‘bpm’: ‘xx to yy’, ‘hashtags’: [x, y, z …], ‘owner\_id’:xx, ‘is\_subscribed’:’true or false’} ….]
  + total: Total number of stations
  + num\_pages: Number of pages
  + next\_page: Url of next page

### List trending stations (DONE)

* Description: This method will list trending stations which are stations popular in a recent period of time.
* Service URL: **https://{BaseURL}/api/music/station/trending**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Failed
    - invalid\_auth\_token
  + Message:
  + Stations: [{‘id’:xx, ‘name’:’xxx’, ‘num\_tracks’:xx, ‘genre’:[x,y,z], ‘mood’:[x,y,z]}, ‘author’:’display name’, ‘recent\_num\_subscribers’:xx, ‘photo’: ‘photo url’, ‘privacy\_level’:’0 OR 1’, ‘bpm’: ‘xx to yy’, ‘hashtags’: [x, y, z …], ‘owner\_id’:xx, ‘is\_subscribed’:’true or false’} ….]

### Create station (DONE)

* Description: This method will create a station
* Service URL: **https://{BaseURL}/api/music/station/create**
* HTTP Method: **POST**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters
* Name: Required.
* Genre: Required. <genreID>
* Mood: Required.<moodID>
* Bpm: Optional. “min\_bpm, max\_bpm” or NOT SET. This is a range of bpm. i.e: “120, 300”
* Hashtags: Optional. “tag1, tag2, tag3 …”. Free texts which are separated by comma.
* Photo: Required. Display photo of a station.
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Name\_existed
    - Empty\_fields
    - Failed
    - invalid\_auth\_token
  + Message:
  + id:
  + name
  + num\_tracks
  + num\_subcribers
  + recent\_num\_subscribers: number of subscribers in the last period of time which (time) is used for selecting trending stations.
  + genre
  + mood
  + bpm: “min\_bpm to max\_bpm”
  + author:
  + privacy\_level: 0 is private and 1 is public
  + photo: station photo url

### Get station detail by ID (DONE)

* Description: This method will list all tracks of a station
* Service URL: **https://{BaseURL}/api/music/station/<stationID>/details**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Invalid\_station
    - Permission\_denied
    - Failed
    - invalid\_auth\_token
  + Message:
  + id:
  + name
  + num\_tracks
  + num\_subcribers
  + recent\_num\_subscribers: number of subscribers in the last period of time which (time) is used for selecting trending stations.
  + genre
  + mood
  + bpm: “min\_bpm to max\_bpm”
  + author:
  + privacy\_level: 0 is private and 1 is public
  + photo: station photo url
  + thumbnail:
  + owner\_id
  + is\_subscribed: Is current user subscribe this station. True if it is his stations

### List all tracks of a station (DONE)

* Description: This method will list all tracks of a station
* Service URL: **https://{BaseURL}/api/music/station/<stationID>**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request Parameters:
  + Page: The page number of the results set. Default is 1
  + Per\_page (OPTIONAL): The items number per page. Default is 100
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Invalid\_station
    - Permission\_denied
    - Failed
    - invalid\_auth\_token
  + Message:
  + id:
  + name
  + num\_tracks
  + num\_subcribers
  + recent\_num\_subscribers: number of subscribers in the last period of time which (time) is used for selecting trending stations.
  + genre
  + mood
  + bpm: “min\_bpm to max\_bpm”
  + author:
  + privacy\_level: 0 is private and 1 is public
  + photo: station photo url
  + thumbnail:
  + owner\_id
  + is\_subscribed: Is current user subscribe this station. True if it is his stations
  + tracks: [{track1 details}, {track2 details} …]
  + total: Total number of tracks
  + num\_pages: Number of pages
  + next\_page: Url of next page

### Update station (name + photo) (DONE)

* Description: This method will update a station’s information (name)
* Service URL: **https://{BaseURL}/api/music/station/<stationID>**
* HTTP Method: **PUT**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request parameters:
* Name
* photo
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Permission\_denied
    - Name\_existed
    - Failed
    - invalid\_auth\_token
  + Message:

### Delete station (DONE)

* Description: This method will delete a station
* Service URL: **https://{BaseURL}/api/music/station/<stationID>**
* HTTP Method: **DELETE**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Permission\_denied
    - Failed
    - invalid\_auth\_token
  + Message:

### Subscribe a station OR Add a station to My Stations(DONE)

* Description: This method will add a station to My Stations
* Service URL: **https://{BaseURL}/api/music/station/<stationID>/subscribe**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Failed
    - invalid\_auth\_token
  + Message:

### Unsubscribe from a station OR Remove a station from My Stations(DONE)

* Description: This method will remove a station from My Stations
* Service URL: **https://{BaseURL}/api/music/station/<stationID>/unsubscribe**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Failed
    - invalid\_auth\_token
  + Message:

### Search stations by name (DONE)

* Description: This method will search stations via name by a search\_term
* Service URL: **https://{BaseURL}/api/music/station/search**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request Parameters:
  + Search\_term (required):
  + Page: The page number of the results set. Default is 1
  + Per\_page (OPTIONAL): The items number per page. Default is 100
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found
    - Failed
    - invalid\_auth\_token
  + Message:
  + Stations: [{‘id’:xx, ‘name’:’xxx’, ‘num\_tracks’:xx, ‘genre’:[x,y,z], ‘mood’:[x,y,z]}, ‘author’:’display name’, ‘recent\_num\_subscribers’:xx, ‘photo’: ‘photo url’, ‘privacy\_level’:’0 OR 1’, ‘bpm’: ‘xx to yy’, ‘hashtags’: [x, y, z …], ‘owner\_id’:xx, ‘is\_subscribed’:’true or false’} ….]
  + total: Total number of stations
  + num\_pages: Number of pages
  + next\_page: Url of next page

### List all tracks in My Tracks (DONE)

* Description: This method will list all tracks in My Tracks
* Service URL: **https://{BaseURL}/api/music/mytracks/tracks**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Request Parameters:
  + Page: The page number of the results set. Default is 1
  + Per\_page (OPTIONAL): The items number per page. Default is 100
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Not\_found (empty)
    - Failed
    - invalid\_auth\_token
  + Message:
  + tracks: [{track1 details}, {track2 details} …]
  + total: Total number of tracks
  + num\_pages: Number of pages
  + next\_page: Url of next page
  + previous\_page: Url of previous page

### Get Ads (DONE)

* Description: This method will remove a track from My Tracks. Use
* Service URL: **https://{BaseURL}/api/music/get\_ads**
* HTTP Method: **GET**
* Headers:
* **PINNA\_APPLICATION\_ID**
* **PINNA\_APPLICATION\_KEY**
* **PINNA\_AUTHENTICATED\_TOKEN**
* Response: is a JSON including
  + Code:
  + Msg\_code:
    - Ok
    - Failed
    - invalid\_auth\_token
  + Message:
  + ads: [{ad1 details: id, name, url}, {ad2 details} …]
  + total: Total number of tracks
  + num\_pages: Number of pages
  + next\_page: Url of next page
  + previous\_page: Url of previous page
  + play\_after: Play an ad after {this value} songs

# Update Notes

## 2014/09/19

* Removed STYLE from the Music Meta model. It is no longer in use.
* Removed BPM from the Music Meta model. Added BPM as an attribute of the Track model.
* Corrected Station workflow.
* Edited reponse of the login/logout/create API.
* Added My Tracks APIs.

## 2014/09/30

* Added more msg\_code to user APIs
* Enabled logging in via email
* Renamed getting/updating user profile APIs
* Separated headers and request parameters

## 2014/10/03

* Added (DONE) in the APIs which is done
* Added more fields to Track: author, artist, duration
* Added searching tracks API

## 2014/10/07

* Changed the response of getting track(s) APIs. Will return full details of a track.
* Added pagination for listing tracks

## 2014/10/08

* Added increasing like/download count for a track
* Updated docs about pagination for listing playlist tracks API
* Added changing password. User have to login again after changing password.

## 2014/10/20

* Added thumb to station
* Added like/unlike/is\_liked a track

## 2014/10/21

* Added a field “is\_liked” to a track details. This field is used for checking if the track is liked by current user. It has 3 values:
  + Is\_my\_track: The track belongs to current user
  + Liked: This track is liked by the current user
  + Not: This track has not been liked by the current user yet.

## 2014/10/22

* Updated is\_liked to be True/False only
* Added owner\_id to Track/Station details
* Added is\_subscribed to Station details
* Added getting My Stations by UserID API

## 2014/10/24

* Added getting station details by ID

## 2014/10/28

* Added follow/unfollow/getting followers APIs
* Added is\_followed to getting profile by UserID API

## 2014/11/13

* Added Purchasing track API
* Added “amount” attribute to get profile APIs
* Added adding PINNA coins API

## 2014/11/18

* Updated adding PINNA coins API: moved to USER APIs part. Added verifying Google In-App purchase signature for each order. Saved the order information in the server database.
* Added ‘is\_purchased’ to Track details
* Implemented increasing Track price every 500 likes OR 100 purchases

## 2014/11/20

* Added Ad server: API + Settings
* Clear some not-in-use APIs