adderNet

By

Arpan Pathak
Subhamoy Sarkar
Tirthamouli Baidya

UNDER THE GUIDANCE OF
Pramit Ghosh

PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING
RCC INSTITUTE OF INFORMATION TECHNOLOGY
Session 2015-2016

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY
[Affiliated to West Bengal University of Technology]
CANAL SOUTH ROAD, BELIAGHATA, KOLKATA-700015

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RCC INSTITUTE OF INFORMATION TECHNOLOGY

TO WHOM IT MAY CONCERN

I hereby recommend that the Project entitled **adderNet** prepared under my supervision by **Arpan Pathak** (Reg. No. 141170110017, Class Roll No.CSE/2014/044), **Tirthamouli Baidya** (Reg. No. 141170110090, Class Roll No. CSE/2014/045),**Subhamoy Sarkar** (Reg. No. 141170110081, Class Roll No. CSE/.2014/038) of B.Tech (7^{th} /8^{th} Semester), may be accepted in partial fulfillment for the degree of **Bachelor of Technology in Computer Science & Engineering** under West Bengal University of Technology (WBUT).

.

……

Project Supervisor
Department of Computer Science and Engineering
RCC Institute of Information Technology

Countersigned:

……

Head
Department of Computer Sc. & Engg,
RCC Institute of Information Technology
Kolkata – 700015.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

CERTIFICATE OF APPROVAL

The foregoing Project is hereby accepted as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it is submitted.

FINAL EXAMINATION FOR
EVALUATION OF PROJECT

1. _____

2. _____

(Signature of Examiners)

ACKNOWLEDGEMENT

I, Subhamoy Sarkar (Class Roll No. : CSE/2014/038, Univ. Roll No: 11700114081) have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them, especially our mentor Sir Pramit Ghosh and my project partners, Arpan Pathak and Tirthamouli Baidya.

_____

ACKNOWLEDGEMENT

I, Arpan Pathak (Class Roll No. : CSE/2014/044, Univ. Roll No: 11700114017) have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them, especially our mentor Sir Pramit Ghosh and my project partners, Subhamoy Sarkar and Tirthamouli Baidya.

s

_____

ACKNOWLEDGEMENT

I, Tirthamouli Baidya (Class Roll No. : CSE/2014/045, Univ. Roll No: 11700114090) have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them, especially our mentor Sir Pramit Ghosh and my project partners, Arpan Pathak and Subhamoy Sarkar.

_____

Table of Contents

1 Introduction

At the point when utilized legitimately, online networking can be a profitable expansion to a division's interchanges system. Since numerous workers have communicated an enthusiasm for creating and keeping up a web-based social networking nearness in individual and expert limits, the Office of University Communications and Marketing has made the accompanying prologue to web-based social networking. It would be ideal if you read this diagram before building up your web-based social networking nearness.

Online networking is a web based type of correspondence. Online networking stages enable clients to have discussions, share data and make web content. There are numerous types of web-based social networking, including websites, smaller scale web journals, wikis, interpersonal interaction locales, photograph sharing destinations, texting, video-sharing destinations, podcasts, gadgets, virtual universes, and that's just the beginning.

Billions of individuals around the globe utilize online networking to share data and make associations. On an individual level, online networking enables you to speak with loved ones, learn new things, build up your interests, and be engaged. On an expert level, you can utilize web-based social networking to widen your insight in a specific field and fabricate your expert system by interfacing with different experts in your industry. At the organization level, online networking enables you to have a discussion with your gathering of people, pick up client input, and lift your image.

With such huge numbers of new web-based social networking locales propelling every year, choosing which one is ideal for your specialization can be overpowering. It is critical to know about rising web-based social networking locales, and see how they could fit into your correspondences technique. In any case, not every social medium locale will be helpful for your specialization's image or promoting objectives.

Before you dispatch an official record on another online networking webpage for your area of expertise, attempt it on an individual level. Make a record for yourself, and afterward utilize it. Concentrate how different people and organizations utilize the site. What sort of substance is posted on the site? Which posts are the most famous on the site? How regularly are clients and organizations posting?

At that point, consider how your area of expertise would fit in. Because you can be on a web-based social networking website, doesn't really mean you ought to be. Spreading yourself crosswise over an excessive number of online networking locales could weaken your social methodology, keeping you from utilizing any of them successfully. Rather, center on the online networking locales that enable you to impart your substance to the proper crowd.

2 Abstract

The Social Media AdderNet features the expanding significance of web-based social networking and systems in the public arena. It is required to give an important chance to scientists and engineers to share their discoveries in this new region and include increasingly highlight in the years to come.An online networking like adderNet gives heaps of highlights to share data about individuals over the globe. It is extremely inaccurate to state it being just an informal organization that associates with individuals. It has highlights which if fused accurately can possibly tumble a portion of the best netting web-based social networking. The adderNet has totally fused a music relax where individuals can enjoy a reprieve from the informal community and hear some out music, make playlist and download music for positively no charge by any stretch of the imagination. The adderNet likewise may execute some Machine Learning calculation that will recognize some content from a bit of picture and yield the content. It might be actualized utilizing the python modules, however for the present, the element is forgotten. The adderNet has a component of Trending post that enables clients to see which post is inclining. The equations and different references are clarified later. Addernet can possibly aggregate a great deal of client and if oversaw accurately, it might never come up short on client space. Designers at adderNet have created calculations and techniques that may keep a repetitive client from reproducing a record. Generally speaking the AdderNet is a decent to attempt long range interpersonal communication site which is free and dependably would be. The project is open source and available at github, anaybody can fork this project and contribute to it, here is the link to download the project : - https://github.com/arpanpathak/adderNet/,

3 Review of Literature

A social media like adderNet provides lots of features to share information about people across the globe. It is very incorrect to say it being only a social network that connects to people. It has features which if incorporated correctly has the potential to tumble some of the top grossing social media.

The adderNet has completely incorporated a music lounge where people can take a break from the social network and listen to some music, create playlist and download music for absolutely no charge at all.

The adderNet also may implement some Machine Learning algorithm that will detect some text from a piece of picture and output the text. It may be implemented using the python modules, but for now, the feature is yet to be implemented.

The adderNet has a feature of Trending post that allows users to see which post is trending. The formulas and other references are explained later.

Addernet has the potential to accumulate a lot of user and if managed correctly, it may never run out of user space. Developers at adderNet have developed algorithms and methodologies that may prevent a redundant user from recreating an account.

Overall the AdderNet is a good to try social networking site which is free and always would be.

3 Project Objectives

- Primary objective is to create an account which is secured and there is no information leak of any sort from the website
- Registration can be done through existing Google or Facebook or GitHub ids.
- The registered users will be allowed to connect to the other users. There would be a follow option which allows users to follow each other's posts.
- There is a messaging option that allows two users to connect and converse using plain text or even using media like pictures and videos.
- Posts may be of some text type, or pictures or videos or all combined.
- Posts can be followed or, liked and reacted.
- adderNet features a free music lounge that allows registered users to listen and download unlimited music. Customizable playlists can also be created.
- It will implement character detection through machine learning so that documents and such artifacts could be easily transported between organizations, just with a click of a button.
- There is a newsfeed which will show all the trending posts as well as posts of all followers/following
- This project may help others to collect real user data for Machine Learning classifier and many useful predictions can be performed such as sentiment analysis, spam detection and filtering, fake user detection and so on.
- There is an OTP authentication API which prevents fake account creation.

4 System Design

4.1. ER Diagrams

- adderNet Social Media Models:

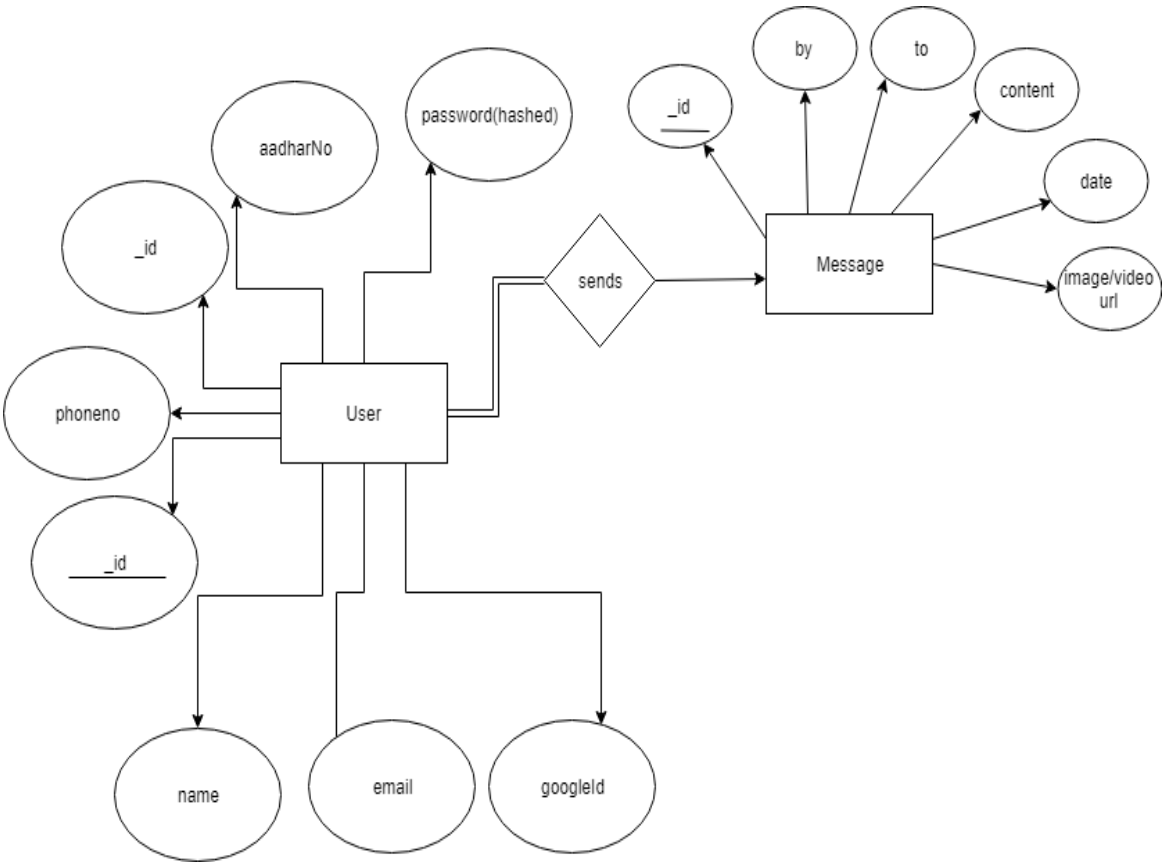


Fig 1. User And Message model implementations

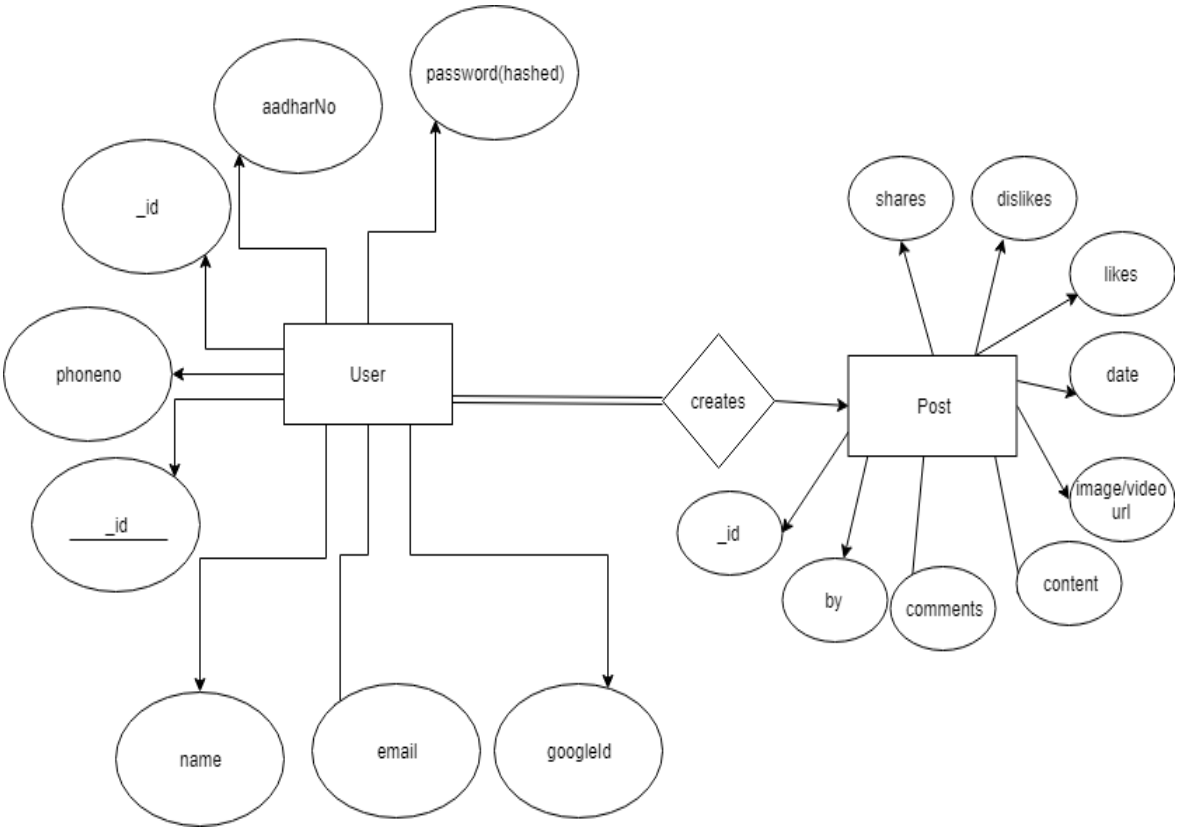


Fig 2. User and Posts Model Implementations

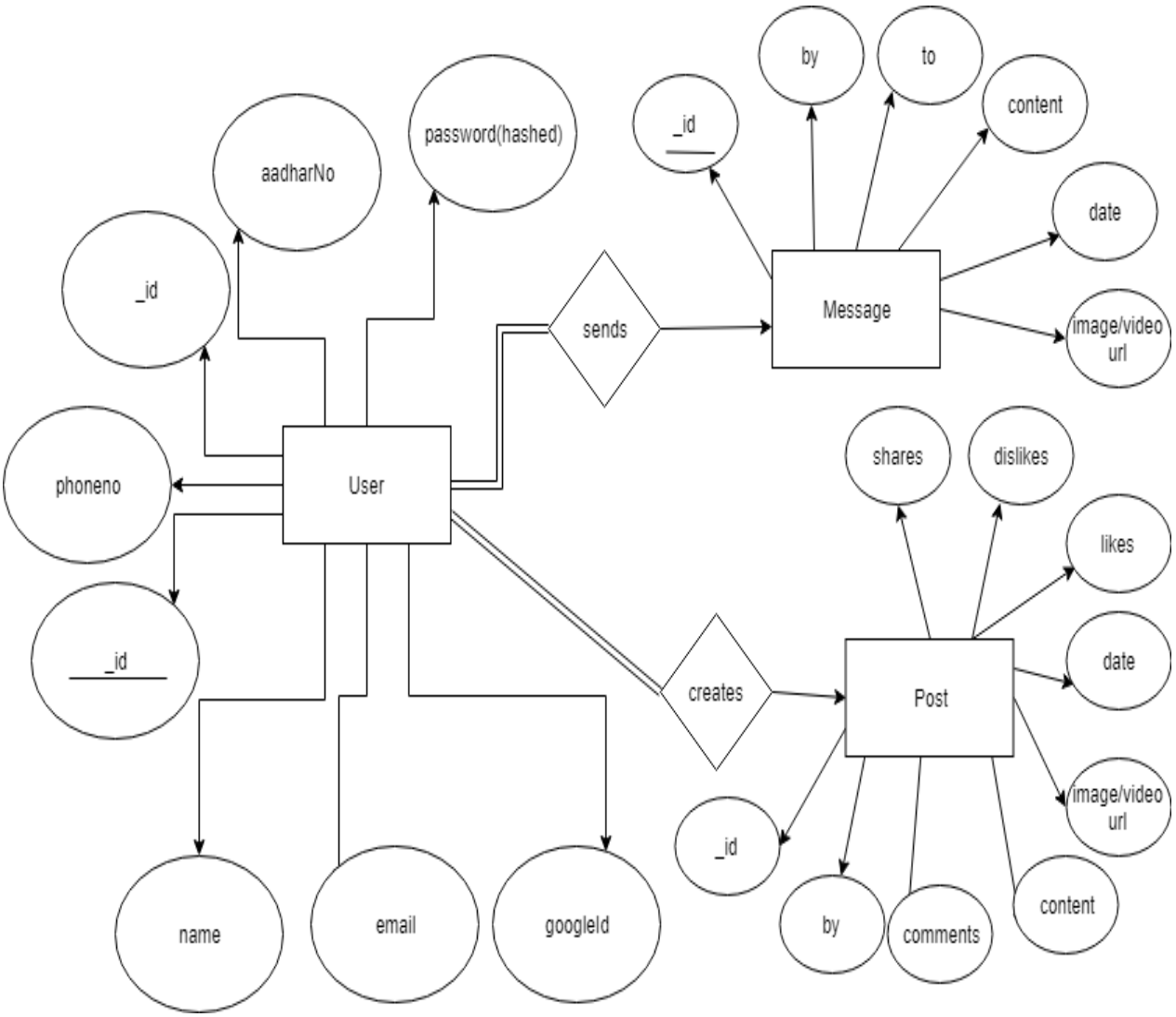


Fig 3. A total ER Diagram for the social media aspect of the adderNet.

- AdderNet Music app Models

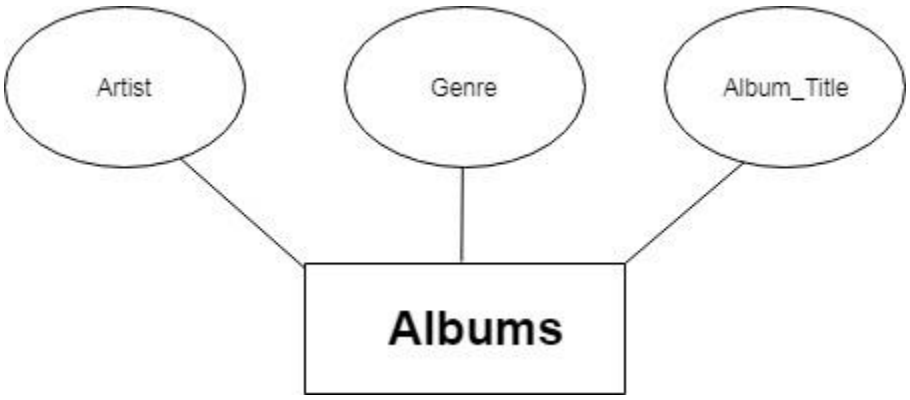


Fig 4. Albums Model

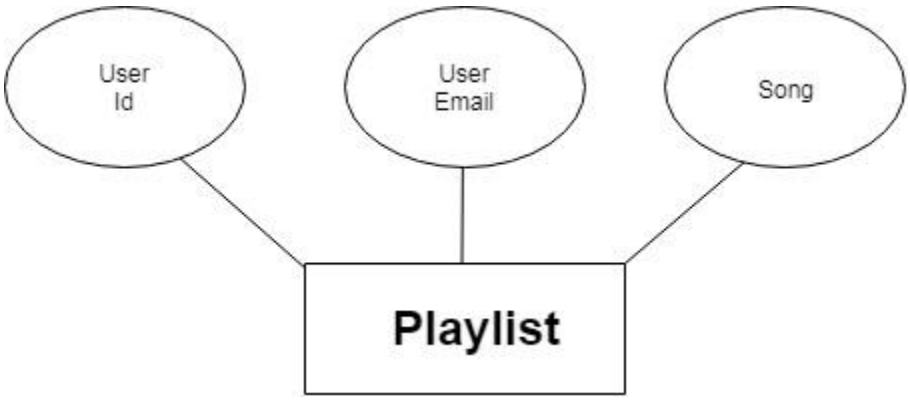


Fig 4. Playlist Model

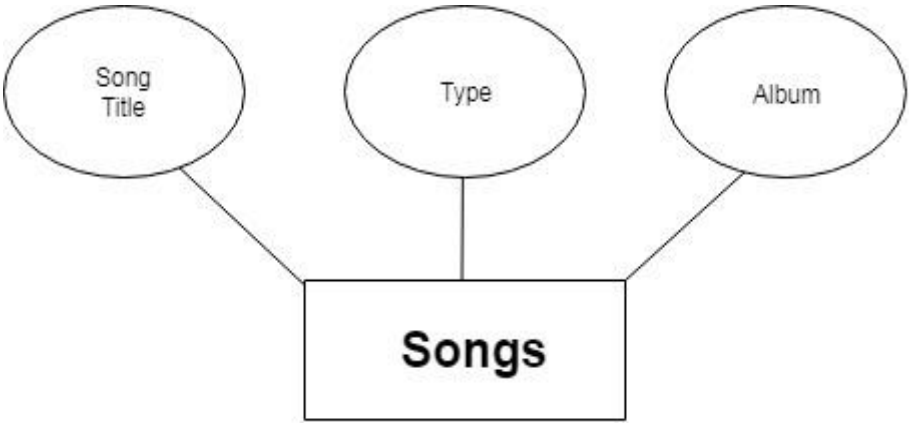


Fig 4. Songs Model

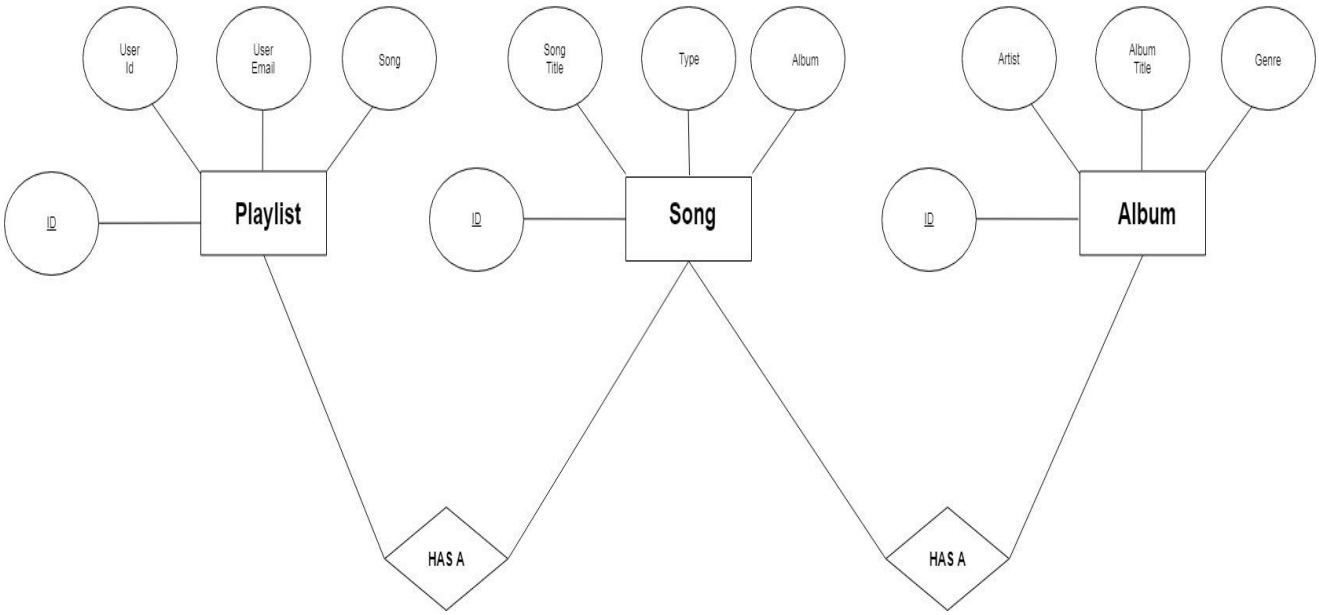
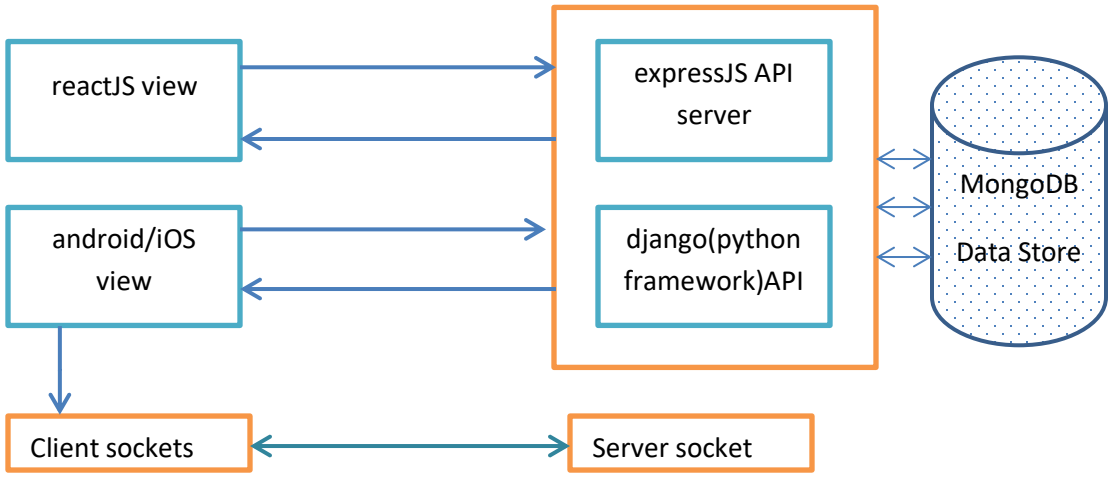


Fig 5. Complete Entity Relation Diagram of the adderNet Music Implementation

Implementation Details

1. AdderNet Social Media

We followed MVC(Model View Controller) design pattern for implementing adderNet. The view layer is completely isolated from everything and it only renders the component/UI to input and output data. Each element in view is called an abstract component. We used reactJS for creating each and every component. Each component has its own controller. The controller communicates to expressJS API server through AJAX request API calls and data is send in form of JSON back and forth. In this way, we are completely following the restAPI paradigm. The advantage of using restAPI is it makes other native app development like android/iOS a lot easier task. The reactJS User Interface made in a way that it'll properly work in mobile/tablets too. The UI is mobile friendly/responsive.
We are using two API servers one is written is expressJS(nodejs framework) and the other is written in django(python framework). All the user data, online users and session data is stored in remotely hosted MongoDB cluster. We are using Mongoose ODM for database management. ODM is very useful because we can add schema to a database like NoSQL.



List of all the functionalities and their implementation : -

- **LOGGING IN STRATEGY**

The implementation is pretty straight forward. Since the user has many ways to login, care had to be taken that the models should have correct values irrespective of the type of login he/she uses. If the user uses normal registration through addernNet and not using Google or Facebook, the information is migrated to the remote MongoDb database. If using Google OAuth or Facebook OAuth, using the reacts passport functionality we can get the information from Google or Facebook in a form of JSON object that gets pushed into the database.Once a user is logged in,they can use any API calls which requires authentication. The user session is persistent and stored in

remotely hosted MongoDB. MongoDB is a NoSQL database which provides scalability. We can easily increase no of clusters to distribute user data into as many clusters as we want. It enforces scalability and great performance in data query.

- **CREATING POSTS**

In creating posts, a dedicated React component pops up and it presents the users with an option of posting only texts or pictures or videos with caption. This was implemented using multipart AJAX submission/ multipart form data. There is a dedicated model in MongoDB that will contain the post after it has been pushed into the database.The uploade files are stored in separate expressJS static file hosting server.Currently there is a limit of 25mb.

Create post uses a '/main/createPost' API call to expressJS server which creates a post as par given data with the help of AJAX.

here is the create post API implementation snippet :-

```
app.post('/main/createPost',upload.fields([{name: 'image',maxCount:1},{name: 'video',maxCount: 1}]),
  (req,res) => {
    console.log(req.files);
    console.log(req.body);
    if(req.body.postContent.trim()=="")
        res.send({error: 'post can not be empty'});
    User.findById(req.user._id).
        then((user)=>{
        if(user)
            new Post({content: req.body.postContent, by: req.user._id,
                image: req.files['image']?req.files['image'][0].filename.toString() : "",
                video: req.files['video']?req.files['video'][0].filename.toString() : "" })
                .save((err,post)=>{

                user.posts.push(post._id);
                user.save();
                res.send( {'_id': post._id });
        });

    });

});
```

- **IMPLEMENTATION OF MESSENGER**

The messenger allows user to chat with each other. User can chat with the people they are following. We used socket.io for message send and received notification which helps us to create a relative instant messaging service which is socke.io event driven. It's not required to refresh the page to see the new message; socket.io is always listening for new events. A technology called ajax polling is used to update a part of a web page on any change without refreshing the entire page. The same socket can use used by all the other component for live notification which is not completed yet so far because nodejs socket.io has a memory leak issue. Currently there are better websockets but they works only on secured socket layer or SSL. We don't have fund to buy a webhosting server with https (SSL certificate).

- **IMPLEMENTATION OF TIMELINE**

The timeline is a component which displays all the information and activity of a user. Anybody can goto his timeline url and see their posts, shares, profile picture, cover picture, followers.

The timeline component is using 4 small AJAX api calls to server to fetch user information and they are :-

a), fetching basic user information like name,email,date created and so on,

b) fetching all the post sorted by time stamp ,

c) fetching all the shared post shorted by timestamp,

d) fetching all the followers

- Implementation of search USER

Users can search and follow other users in adderNet. Our search query matches all the fields such as user id, name, surname, email id, phone no,

- Implementing the news feed
  Newsfeed is currently display all the post of the users currently the user following sorted by time stamp.
- Nearest Friend Notification **[unable to implement without SSL Certificate]**
  We planned to implement finding out nearest friend within 1 km radius and inform user according to their location if they want with the help of google geolocation api. But google geolocation/map API requires a secured server with https which is not possible without buying a domain, server and SSL Certificate.

2. AdderNet Music App

- **HOME PAGE AND REDIRECTION**

The adderNet music app has a home page that is shown when a logged in user from adderNet Social media is access this section. The whole music application is developed using Django framework, that is quite different from how React and NodeJs works. Hence to get the userid and user email correctly, a separate model as shown above for the music system was implemented and the initial parameters to login a user are got as a get request from the adderNet server. We take this userid and email to implement the playlist feature for individual users.

- **IMPLEMENTATION FOR ALBUM LISTINGS**

The page after homepage is a simple display of all the albums along with the album art as thumbnail. This data is preloaded into the database by the admin and the loading url can be accessed only by the admin, due to security issues.

```
<table class="striped">
                    <tr>
                        {% for album in all_albums %}
                        <td>
                            <a href="{% url 'music:detail' album.id %}">
                             <img src="{{album.album_logo}}" class="circle">{{album.album_title}}</a>


                        </td>
                        {% if forloop.counter|divisibleby:3 %}
                    </tr>
                    <tr>
                        {% endif %}
                        {% endfor %}
                    </tr>
</table>
```

IMPLEMENTING THE SONG LISTING

Just as above, the song data is also preloaded into the database by the admin. The song list is simply the set of all the songs that belong to one common album. Every song has an option to be added to the favorites for future playing from the playlist directly. You can also download the song from the server to your local machine from the audio controls provided.

Here is the implementation of the songlist in Django.

```
<table class="striped">
    {% for songs in album.song_set.all %}
        <tr>
            <td>{{ songs.song_title }}</td>
               <td>
                   <audio controls><source src="http://10.11.48.51:8080/Pink Floyd/{{ album.album_title }}/{{ songs.song_title }}.mp3" type="audio/mpeg"></audio>
                   <!--<a href="{% url 'music:plist' album.id songs.id %}">AddToFav</a><br>-->
                    <td><a class="btn-floating pulse cyan tooltipped" data-position="right" data-tooltip="Add To Favourites"
                          href="{% url 'music:plist' songs.id album.id %}"><i class="material-icons">playlist_add</i></a></td>
            </td>
        </tr>
    {% endfor %}
</table>
```

- **FAVORITES OR PLAYLIST**

Every user will have the privilege to make his/her own playlist. Clicking on add to favorites does the job of populating the database with the song and the UserId taken from AdderNet. Now every time a user logs in and requests for his/her playlist, the songs which have the UserID of the particular person is filtered and shown. The user also has the option of removing the particular song from the playlist if he/she wishes to.

Here is the main implementation of the playlist in Django.

```
<table class="striped">
   {% for pl in all_pl %}
       {% ifequal pl.userid cuser %}
         <tr>
         <td>{{ pl.song.song_title }}</td>
             <td> <audio controls><source src="http://10.11.48.51:8080/Pink Floyd/{{ pl.song.album.album_title }}/{{ pl.song.song_title }}.mp3" type="audio/mpeg"></audio></td>
           <td><a class="btn-floating  pulse red tooltipped" data-position="right" data-tooltip="Remove"  href="{% url 'music:delsong' pl.id %}"><i class="material-icons">clear</i>
         </tr>
       {% endifequal %}
    {% endfor %}
</table>
```

Output/Result

1.AdderNet MusicApp

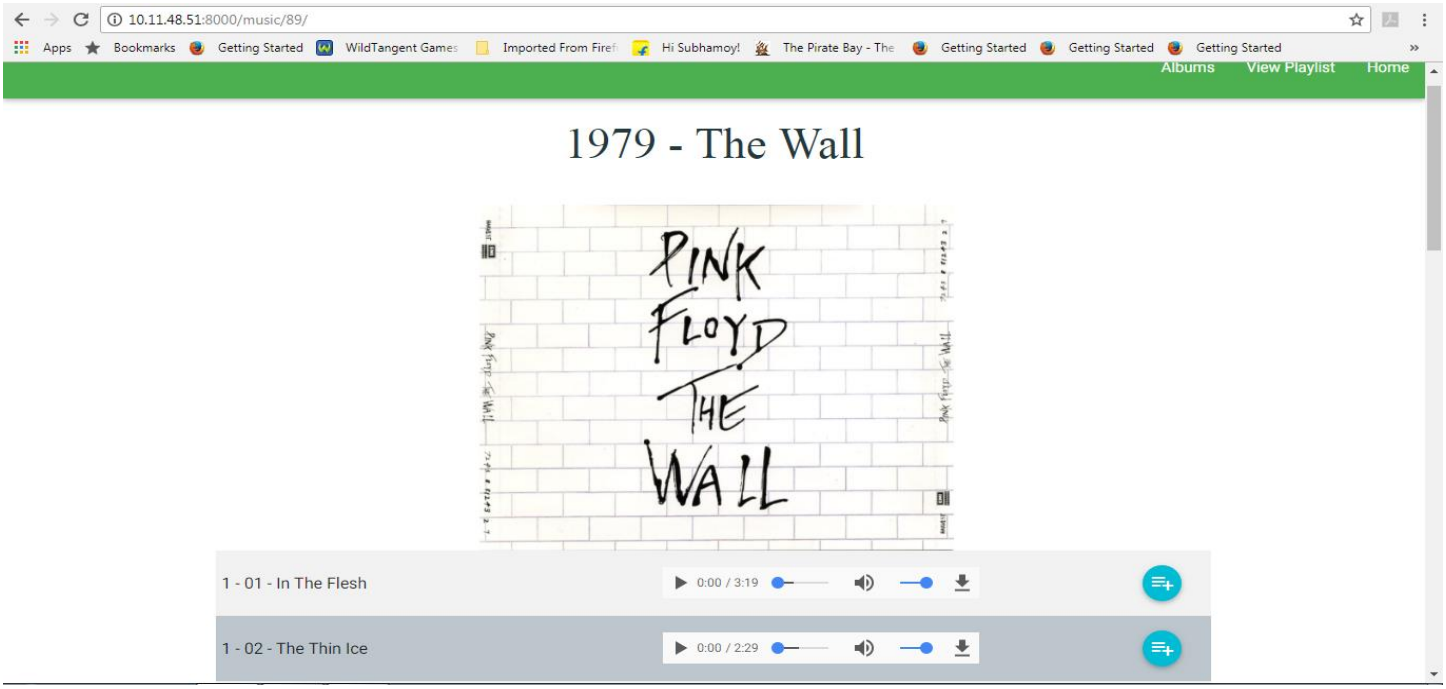


FIG. Songlist Implementation

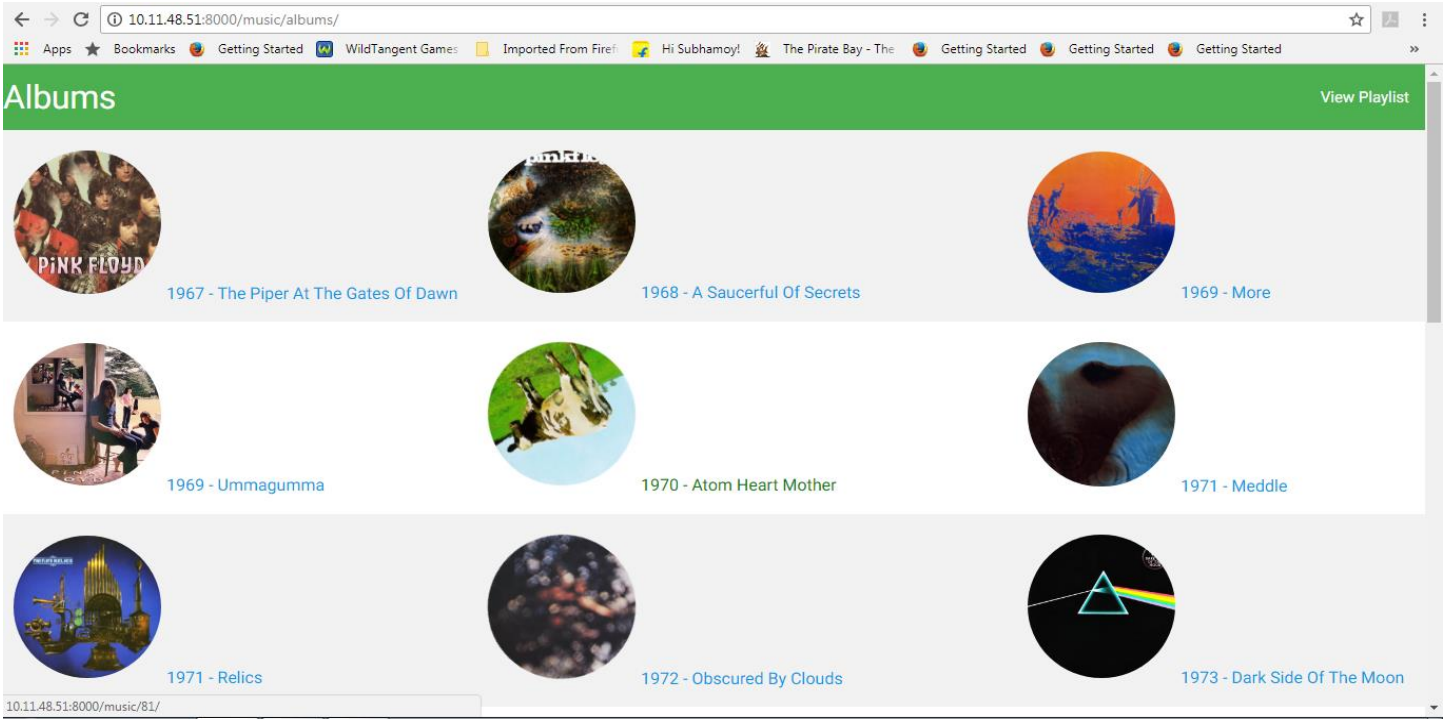


FIG. Implementation of Album listings

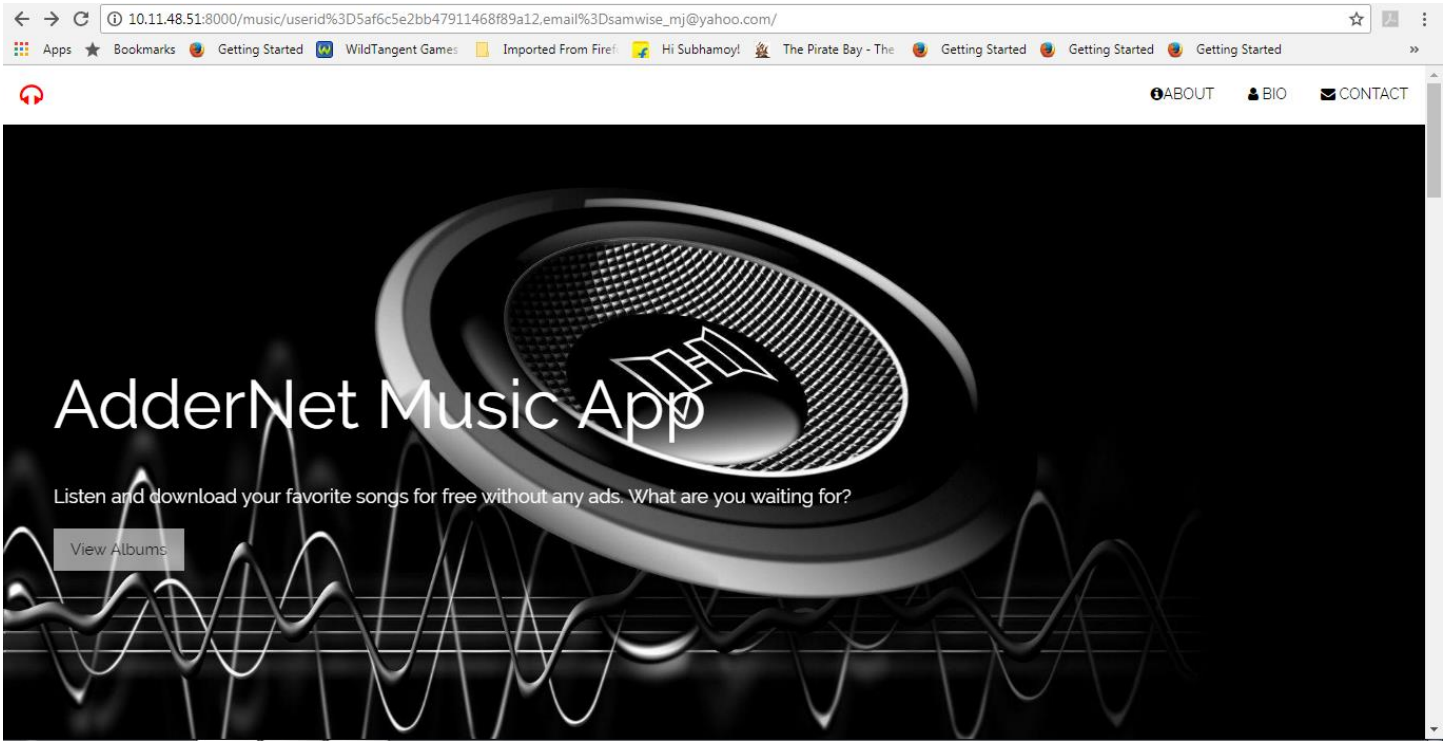


FIG. AdderNet Home page

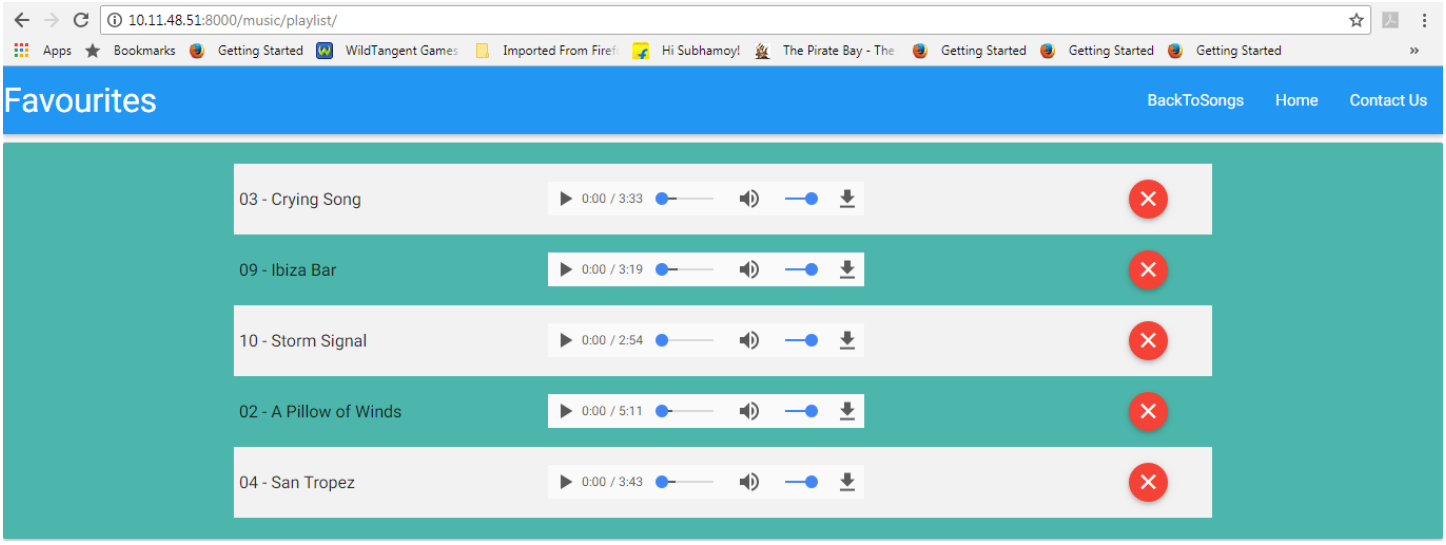


FIG. Implementation of Playlist.

Conclusion

To conclude, inspite of some shortcomings, the project was executed and every aspect of the project was tested. The project after extensive testing was found out to be ok, with some level of fault tolerance and every module and functionality worked as expected. For some more functionality, the SSL certificate must be purchased, so that it can be incorporated into the application and hence allow us to show some special functionality which cannot be displayed now as of yet due to absence SSL certification.

Also, a NodeJS server may be purchased and the project may be deployed after some more testing and fine tuning, to be extensively used by anyone across the globe.

APPENDIX (Code)

Code for API (expressJS)

api.js

```
/*** ALL APIS, @author: arpanp_athak,tirthamouli_baidya,subhamoy_sarkar ***/

/*** import your models here ***/

/***                                                                                                    |
=====================================IMPORTS=========================================
==================    |***/

const mongoose = require('mongoose');


const User=mongoose.model('user');

const Post=mongoose.model('post');

const Comment=mongoose.model('comment'),

        Message=mongoose.model('Message');

const Conversation=mongoose.model('Conversation');

const config=require('./config/keys');

const helpers=require('./lib/helpers');

const multer  = require('multer');

const path = require('path');

var storage = multer.diskStorage({

  destination: function (req, file, cb) {

    cb(null, 'uploads/')

  },

  filename: function (req, file, cb) {

      console.log(file.originalname);

    cb(null, Date.now() + "."+file.mimetype.split("/")[1]) //Appending extension

  }
```

```
})

let upload = multer({ storage: storage });


/***
================================================================================
================== ***/


/* internal API */

 // this piece of code is moved to /lib/helpers.js file

/* end of this section */


// all server API ..

module.exports = (app,passport,io) => {


      require('./sockets/socket-routes.js')(io);

      const Online=require('./models/online-model');

      Online.remove({},()=>{

        console.log("Cleared Online");

      });

      /// NOTE : -- socket.io added to this file.. you can emit any method to connected
socket..

      // an API to get all the details about the developers/creators ...

      app.get('/creators', (req, res) => {

        const creators = [

          {id: 1, firstName: 'Arpan', lastName: 'Pathak'},

          {id: 2, firstName: 'Tirthamouli', lastName: 'Baidya'},

          {id: 3, firstName: 'Subhamoy', lastName: 'Sarkar'},


        ];
```

```
        res.json(creators);
    });


    // an API to see whether the user is authenticated and authorized to view certain api or not...
    app.get('/authenticated',(req,res) => {
        // if user is authenticated then user will not be null in request object...
        res.json({authenticated: req.isAuthenticated(), user: req.user, maxAge: req.session.cookie.maxAge });
    });


    //User Registration API...
    app.post('/registerUser',(req,res) => {
        User.findOne({ $or:[ {email: req.body.email},
                                    {$and: [{aadharNo: {"$ne":""}},{aadharNo: {"$ne":null}},{aadharNo: req.body.aadharNo} ] },
                                    {phone: req.body.phoneNo.toString()} ]} ,
                //check if email Id already registered
                (err,user)=>{
                    if(user){
                        let msg="";
                        if(user.email==req.body.email) msg+='email,';
                        if(user.aadharNo!=null && user.aadharNo!='' && user.aadharNo===req.body.aadharNo.toString()) msg+='aadharNo,';
                        if(user.phone===req.body.phoneNo.toString()) msg+='phone no,';
                        msg+=' already registered';
                        res.json({"status":"failed","error": msg});
                    }
```

```
                        else{
                              //Sanity check
                              var email = req.body.email.match(/^[a-zA-Z0-9.!#$%&’*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/) !== null ? req.body.email : false;
                              var password = typeof(req.body.password) == 'string' && req.body.password.length > 7 ? req.body.password : false;
                              var phone = req.body.phoneNo.toString(),aadharno,aadharNo=req.body.aadharNo.toString(),
                                    name=req.body.firstName+" "+req.body.lastName;

                              // after sanity checking....
                              if(name && email && password && phone ){
                                    //hashing password
                                    var hash = helpers.hash(password);
                                    //Store user
                                    new User({'name': name,'email':email,'password':hash,'aadharNo': aadharNo,
                                          date_created: Date.now(),'phone': phone})
                                       .save((newUser)=>{
                                             // if new uer is null then there is a problem in database...
                                             !newUser? res.json({"status" : "success","error":""}) :

res.json({"error" : "Unable to create user"});
                                       }
                                    );

                              }else{
```

```
                                    res.json({"error"  :  "Invalid  email  and  password
type"});
                                }
                            }


                    }
                );
            });


            // use this as auth failed callback URL....
            app.get('/authFailed',(req,res)=>res.json({'error': 'authentication failed!'}));


            // API to send OTP... currently using free twilio message service...
            app.get('/sendOTP',(req,res)=>  helpers.sendTwilioSms("+919099994016","apni  vogoban
dada, ei nin OTP ="+helpers.RandomStringGenerate(6),
                                                    (err)=> res.send(err?err:'sent') )
                    );
            /***            ================================================================
================================================***/
            /*** authentication API ***/
            // this is the login api, here use passportJS callbacks to authenticate using
            // any strategy you want.. to add OAuth strategy , edit the required setup file...
            // EmailOTP will be added after implementing sendEmail( ) helper function...
            app.post('/login',passport.authenticate(['email-local','phone-
local'],{failureRedirect: '/authFailed' }),
              (req,res) => {
                  res.json( {'loggedin' : true } );
            });
            // logout api
            app.get('/logout',(req,res) => {
```

```
            req.session.destroy();
            req.logout();

            res.json({ message: 'logged out','status': 'logged out'}) ;
            io.emit('logout',{data: 'test'});
      });



      //implementing music api
      app.get('/music',(req,res) => {
            //res.json({authenticated: req.isAuthenticated(), 'usern': req.user.name });
            res.json({usernm: "", userem: req.user.email });
            //res.json({'email':req.user.email});
      });

      // google oauth API
      app.get('/auth/google', passport.authenticate('google', {
          scope: ['profile','email']
      }));

      // google OAuth callback URI
      app.get('/auth/google/callback',
            passport.authenticate('google',{                              failureRedirect:
'/authFailed',successRedirect: 'http://localhost:3000/profile' }),
            // This part of code will only be executed if successRedirect is not set in
passport.authenticate()
            (req,res) => res.send(helpers.json_to_html(req.user))
      );
```

```
      // facebook OAuth API

      app.get('/auth/facebook',passport.authenticate('facebook'));

      // facebook OAuth callback URI...

      app.get('/auth/facebook/callback',

        passport.authenticate('facebook',                {                failureRedirect:
'/authFailed',successRedirect: 'http://localhost:3000/profile' }),

        // This part of code will only be executed if successRedirect is not set in
passport.authenticate()

        (req,res)=> {  req.session.save((err)=>res.send(helpers.json_to_html(req.user)) )
}

      );

      /***
================================================================================
================== ***/



      /*** user info API ***/

      app.post('/user/changeDp',upload.fields([{name: 'image',maxCount:1}]),(req,res)=>{

            User.findByIdAndUpdate(          req.user._id,          {          profilePic:
req.files['image'][0].filename.toString() }).

            then((user)=>res.send(          {          'changed':          true,profilePic:
req.files['image'][0].filename.toString()} ));

      });

      app.post('/user/changeCoverPic',upload.fields([{name:
'image',maxCount:1}]),(req,res)=>{

            User.findByIdAndUpdate(           req.user._id,           {           coverPic:
req.files['image'][0].filename.toString() }).

            then((user)=>res.send(          {          'changed':          true,coverPic:
req.files['image'][0].filename.toString()} ));

      });

      app.post('/user/addFriend',(req,res)=>{

            User.findById(req.user._id).then((user)=>{

                  user.friends.push(req.body._id);
```

```
                  user.friend_requests_send.push(req.body._id);
                  User.findById(req.body._id).
                  then((friend_request_user)=>{
                        friend_request_user.friend_requests.push(req.user._id);
                        friend_request_user.save();
                        user.save().then(res.send({added: true}));
                  });

            });
      });
      /***
==========================================================================end
of this section ***/

      /*** user post APIs ***/
      app.get('/allPost',(req,res)=> {
            // Post.remove({},(err)=>{ res.send({fuck: true}) });
            Post.find({}).then((posts)=>res.send(posts));
      });
      /*** main APIs ***/
      app.post('/main/createPost',upload.fields([{name:        'image',maxCount:1},{name:
'video',maxCount: 1}]),
        (req,res) => {
            console.log(req.files);
            console.log(req.body);
            if(req.body.postContent.trim()=="")
                  res.send({error: 'post can not be empty'});
            User.findById(req.user._id).
                  then((user)=>{
```

```
                if(user)
                      new Post({content: req.body.postContent, by: req.user._id,
                            image:
req.files['image']?req.files['image'][0].filename.toString() : "",
                            video:
req.files['video']?req.files['video'][0].filename.toString() : "" })
                            .save((err,post)=>{


                            user.posts.push(post._id);
                            user.save();
                            res.send( {'_id': post._id });
                });


          });


      });
      app.get('/main/getAllPost',(req,res) => {
            User.findById(req.user._id).populate( [
             { path: 'posts',
               populate:  {
                        path: 'comments',
                        populate: {path: 'by', select: 'name profilePic'}
                    }


             },
             { path: 'posts',
               populate:  {
                        path: 'by',
                        select: 'name profilePic'
```

```
            }


        },


        ]).


            then((user)=>{
            if(user)
                  res.send(user.posts);
            });


});
app.post('/main/getAllSharesById',(req,res) => {
      User.findById(req.body._id).populate( [
       { path: 'shares',
         populate:  {
                  path: 'comments',
                  populate: {path: 'by', select: 'name profilePic'}
            }


        },
        { path: 'shares',
         populate:  {
                  path: 'by',
                  select: 'name profilePic'
            }


        },
```

```
        ]).

             then((user)=>{
             if(user)
                   res.send(user.shares);
             });

});

app.post('/main/getAllPostById',(req,res) => {
      User.findById(req.body._id).populate( [
       { path: 'posts',
         populate:  {
                   path: 'comments',
                   populate: {path: 'by', select: 'name profilePic'}
             }

       },
       { path: 'posts',
         populate:  {
                   path: 'by',
                   select: 'name profilePic'
             }

       },

       ]).
```

```
                then((user)=>{
                if(user)
                      res.send(user.posts);
                });

      });
      app.post('/main/sharePost',(req,res)=>{
            User.findById(req.user._id).then((user)=>{
                  user.shares.push(req.body._id);
                  user.save();
                  Post.findById(req.body._id).then((post)=>{
                        post.shared_by.push(req.user._id);
                        post.save().then(res.send( { 'shared': true}));
                  });
            });
      })
      app.post('/main/addComment',(req,res)=>{
            Post.findById(req.body._id).then((post)=>{

                  var    comment    =    new    Comment({postId:    req.body.id,content:
req.body.content, by: req.user._id});
                  new Comment({content: req.body.content, by: req.user._id})
                        .save((err,comment)=>{

                        post.comments.push(comment._id);
                        post.save().then(()=>
                        res.send({added:    true,    content:    comment.content,    by:
req.user.name})
```

```
                  );
            });


      });
});
app.post('/main/addLike',(req,res)=>{
      Post.findById(req.body._id).then((post)=>{
            post.likes.push(req.user._id);
            post.save().then(()=>res.send({added: 'added'}));
      })
});
app.post('/main/addDislike',(req,res)=>{
      Post.findById(req.body._id).then((post)=>{
            post.dislikes.push(req.user._id);
            post.save().then(()=>res.send({added: 'added'}));
      })
});
app.post('/main/sharePost',(req,res)=>{
      User.findById(req.user._id).then((user)=>{
            user.shares.push(req._id);
            post.save().then(()=>res.send({added: 'added'}));
      })
});


app.post('/main/updatePost',(req,res) => {
      Post.findByIdAndUpdate(req.body._id,{content: req.body.content})
            .then(res.send({updated: 'true'}));
});
```

```
	app.post('/main/updatePost',(req,res) => {
		Post.findByIdAndUpdate(req.body._id,{content: req.body.content})
			.then(res.send({updated: 'true'}));
	});
	app.post('/main/deletePost',(req,res) => {
		Post.findByIdAndRemove(req.body._id,()=> {
			User.findById(req.user._id).then((user)=>{		user.posts.pull({_id: req.body._id})
				user.save();
				Comment.remove({postId: req.body._id});
				res.send({deleted: 'true'})
			});

		});
	});
	/** ================================================================ **/
	/** end of this section **/


	/***		_____message		sending		and		retreiving APIs._____ ***/
	app.get('/main/getAllMessages',(req,res) => {
		User.findById(req.body._id).populate('messages').then((messages)=>{ res.send(messages)});
	});
	app.get('/t',(req,res)=> Conversation.find({}).then(conversation=>res.send(conversation)) );
	app.post('/main/getConversation',(req,res) => {
		let conversation_id= (req.user._id<req.body.to)?
```

```
                                        (req.user._id+"_"+req.body.to)                :
(req.body.to+"_"+req.user._id);
            console.log(conversation_id);
            console.log(req.body.to);

      Conversation.findById(conversation_id).populate('messages').then(conversation=>{
                  if(!conversation){
                        conversation=new Conversation({_id: conversation_id });
                        conversation.save();

                  }
                  res.send(conversation.messages );
            } );
      });
      app.post('/main/sendMessage',(req,res)=>{
            let conversation_id= (req.user._id<req.body.to)?
                                        (req.user._id+"_"+req.body.to)                :
(req.body.to+"_"+req.user._id);

            if(req.body.to.trim()=="")
                  conversation_id=req.user._id; //

            var msg=new Message({data: req.body.data,by: req.user._id,to: req.body.to });
            msg.save();
            Conversation.findById(conversation_id).then(
                  conversation=>{
                        conversation.messages.push(msg);
                        conversation.save().then( ()=>{
                              Online.find({id: req.body.to}).then((onlineData)=>{
```

```
                                        onlineData.forEach((onlinePerson)=>{

                                          io.to(onlinePerson.socket).emit('message-
received',msg);
                                        });
                                 });
                                Online.find({id: req.user._id}).then((onlineData)=>{
                                           console.log('emiting to '+req.user._id);
                                        onlineData.forEach((onlinePerson)=>{
                                          io.to(onlinePerson.socket).emit('sent',msg);

                                        });
                                 });
                                res.send({ sent: true })
                           });

                    }
              )

      });

      /*** ======================END=======================================***/

      app.get('/currentUser',(req,res)=>
User.findById(req.user._id).then((user)=>res.send(user)));

      app.post('/getUserById',(req,res)=>
User.findById(req.body._id).then((user)=>res.send(user)));
```

```
	app.post('/main/getAllFriends', (req,res)=>{
		User.findById(req.body._id).populate({
			path: 'friends',
			select: 'name profilePic coverPic'
		}).then((user)=>res.send(user.friends));
	});



	app.post('/main/getAllFriendsId', (req,res)=>{
		User.findById(req.body._id).
			then((user)=>res.send(user.friends));
	});


	app.get('/newsFeed',(req,res)=>{
		User.findById(req.user._id).populate({
			path: 'friends',
			select: 'posts'
		}).then(user=>res.send(user.friends)); // user.friends is now populated list of posts of all the friends...
	});
	app.get('/searchUser',(req,res)=>{
		let name=req.query.name;
		User.find({$or:[{name: new RegExp(name, "i")},{email: new RegExp(name, "i")},{phone:RegExp(name, "i")} ]},null,{sort:'_id'})
		.select('name').select('profilePic').select('phone')
		.then((users)=>{
			res.send(users);
		} );
	})
```

```
	/*** ..... **/
	// Test API
	//importing the unit test file....
	require('./tests.js')(app);
}


/*
 * Search for friend below here
 *
 */


// app.get('/searchFriend',(req, res)=>{
//    name = typeof(req.query.name) == 'string' && req.query.name.trim().length > 0 ? req.query.name.trim() : false;
//    if(name){
//          helpers.searchFriend(name, (err, data)=>{
//                if(!err && data){
//                      res.json(data);
//                }else{
//                      res.send("No data found");
//                }
//          });
//    }else{
//          res.send("Error: Please enter valid search name");
//    }
// });
```

Code for AdderNet Music App (Django/Python)

Models.py

```
from django.db import models


    class Album(models.Model):
        artist=models.CharField(max_length=250)
        album_title=models.CharField(max_length=500)
        genre=models.CharField(max_length=100)
        album_logo=models.CharField(max_length=1000)

        def __str__(self):
            return self.album_title +" - "+self.artist

    class Song(models.Model):
        album=models.ForeignKey(Album,on_delete=models.CASCADE)
        file_type=models.CharField(max_length=10)
        song_title=models.CharField(max_length=250)

        def __str__(self):
            return self.song_title

    class Playlist(models.Model):
        userid=models.CharField(max_length=100)
        useremail=models.CharField(max_length=100)
        song=models.ForeignKey(Song, on_delete=models.CASCADE)

        def __str__(self):
            return self.song.song_title
```

urls.py

```
from django.conf.urls import url

from . import views

app_name='music'

urlpatterns = [

    # /music/

    url(r'^$', views.index,name='index'),


    #load db

    #userid=5ae41f0c8071e11420dd0327,email=samwise.mj@gmail.com

    url(r'^userid=(?P<user_id>[0-9A-Za-z]+),email=(?P<email_id>[0-9A-Za-
z.@_]+)/$',views.redinit,name="redinit"),

    url(r'^dbloadsong5535/',views.dbloadsong,name="dbloadsong"),

    url(r'^dbloadalbum5535/',views.dbloadalbum,name="dbloadalbum"),

    # url(r'^dbloadalbumart5535/',views.dbloadalbumart,name="dbloadalbum")

    #/music/xxxx

    url(r'^(?P<album_id>[0-9]+)/$',views.detail,name="detail"),

#     url for favourite

    url(r'^(?P<album_id>[0-9]+)/favourite/$',views.favourite,name="favourite"),

    # add_category form

    url(r'^add_category/',views.add_category,name='add_category'),

    url(r'^(?P<song_id>[0-9]+)/(?P<album_id>[0-9]+)$', views.plist, name="plist"),

    url(r'^add',views.add,name='add'),

    url(r'^abc2',views.add2,name='add2'),

    url(r'^playlist/(?P<pl_id>[0-9]+)/$', views.delsong, name="delsong"),

    url(r'^playlist/', views.vplist, name="vplist"),

    url(r'^albums/', views.albums, name="albums"),
```

```
    url(r'^home/', views.home, name="home"),
]
```

views.py

```
from django.http import HttpResponse,Http404
from django.template import RequestContext
from django.shortcuts import render,get_object_or_404,render_to_response
import os.path, os
import requests
from .models import Album, Song, Playlist

def redinit(request, user_id, email_id):
    print(user_id)
    print(email_id)
    # print(222+222)
    request.session['user_id'] = user_id
    request.session['email_id']=email_id

    # return render(request, 'music/index.html', {'all_albums':Album.objects.all()})
    return render(request, 'music/home.html')
    # return render(request,'music/',{})

def home(request):
    return render(request, 'music/home.html')

def index(request):
    all_albums=Album.objects.all()
```

```
    # template=loader.get_template('music/index.html')
    context = {
        'all_albums':all_albums,
    }
    return render(request,'music/index.html',context)

def albums(request):
    return render(request, 'music/index.html', {'all_albums': Album.objects.all()})

def detail(request, album_id):
    # try:
    #     album=Album.objects.get(pk=album_id)
    # except Album.DoesNotExist:
    #     raise Http404("No album")
    album=get_object_or_404(Album,pk=album_id)
    x=request.session.get('email_id')
    # return HttpResponse(str(x))
    print(x)
    return render(request,'music/songlist.html',{"album":album,})

def plist(request, song_id, album_id):
    # print(song_id)
    # print(request.session.get('email_id'))
    # print(request.session.get('user_id'))

    # entry=Playlist.objects.get(Song.id=song_id)
    # print(entry)
    pl=Playlist()
```

```
    song=get_object_or_404(Song, pk=song_id)
    print(song)
    pl.song=song
    pl.useremail=request.session.get('email_id')
    pl.userid=request.session.get('user_id')
    pl.save()
    album = get_object_or_404(Album, pk=album_id)
    return render(request, 'music/songlist.html', {"album": album,"msg":'true' })


def delsong(request,pl_id):
    pl=Playlist.objects.get(pk=pl_id)
    print(pl)
    pl.delete()
    userid = request.session.get('user_id')
    pl = Playlist.objects.all()
    return render(request, 'music/plist.html', {"all_pl": pl, "cuser": userid,
"msg":'true'})


def vplist(request):
    userid=request.session.get('user_id')
    pl=Playlist.objects.all()
    return render(request, 'music/plist.html',{"all_pl":pl, "cuser":userid,})


def favourite(request,album_id):
    album = get_object_or_404(Album, pk=album_id)
    try:
        selected_song=album.song_set.get(pk=request.POST['song'])
    except(KeyError,Song.DoesNotExist):
```

```
        return render(request,'music/songlist.html',{
            "album":album,
            "error_msg":"Valid song is not selected",
        })
    else:
        selected_song.is_favourite=True
        selected_song.save()
        return render(request, 'music/songlist.html', {"album": album, })


def add_category(request):
    context=RequestContext(request)
    if request.method=='POST':
        form=CategoryForm(request.POST)
        #         CHECKING FORM VALIDITY
        if form.is_valid():
            #save new category to the dbase
            form.save(commit=True)
            return index(request)
        else:
            print (form.errors)
    else:
        form=CategoryForm()
        # Bad form (or form details), no form supplied...
        # Render the form with error messages (if any).
        return render(request,'music/add_category.html',{'form': form})


def add(request):
    context={}
```

```
    if request.method=="POST":
        num1=int(request.POST.get("num1"))
        num2=int(request.POST.get("num2"))
        context["result"]=num1+num2


    return render(request, "music/abc.html", context=context)

def add2(request):
    context=RequestContext(request)

    if request.method=='POST':
        form=User(request.POST)
        if form.is_valid():
            form.save(commit=True)
            saved=True
            return render(request, "music/abc2.html", {"form": form,"saved":saved})
        else:
            print(form.errors)
    else:
        form=User()
    return render(request,"music/abc2.html",{"form":form})


def dbloadalbum(request):
    dir = "C:\\Users\\Samwise\\Desktop\\Music\\Pink Floyd"
    all = Album.objects.all()
    flag = 0
```

```
    for filename in os.listdir(dir):
        for albums in all:
            if albums.album_title==filename:
                flag=1;
        if flag==0:

            a=Album()
            a.artist="Pink Floyd"
            a.album_title=filename
            a.genre='Progressive Rock'
            a.album_logo="http://localhost/Pink Floyd/"+filename+"/"+"Front.jpg"
            a.save()
        flag=0
        a=Album()
        a.artist="Pink Floyd"
        a.album_title=filename
        a.genre='Progressive Rock'
        a.album_logo="http://10.11.48.51:8080/Pink Floyd/"+filename+"/"+"Front.jpg"
        a.save()



    return HttpResponse("<h2>Done!!!!!Check Db</h2>")



def dbloadsong(request):
    dir = "C:\\Users\\Samwise\\Desktop\\Music\\Pink Floyd"
    for filename in os.listdir(dir):
        dirin = dir + "\\" + filename
```

```
        a = Album.objects.get(album_title__contains=filename)
        all = Song.objects.all()

        for song in os.listdir(dirin):
            for songs in all:
                if songs.song_title == song[0:song.find(".mp3")]:
                    if songs.album == a:
                        flag = 1
            if (flag == 0):
                if (song.endswith(".mp3")):
                    s = Song()

                    s.album = a
                    s.file_type = "mp3"
                    s.song_title = song[0:song.find(".mp3")]
                    s.save()
            flag = 0
    return HttpResponse("<h2>Done!!!!!Check Db</h2>")
```

home.html

```
<html>
{% load staticfiles %}
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
```

```
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">

<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Raleway">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

  <!--<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBu-916DdpKAjTmJNIgngS6HL_kDIKU0aU&callback=myMap"></script>-->

<!--<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBu-916DdpKAjTmJNIgngS6HL_kDIKU0aU&callback=myMap"></script>-->


<style>

body,h1,h2,h3,h4,h5,h6 {font-family: "Raleway", sans-serif}

body, html {

    height: 100%;

    line-height: 1.6;

}

/* Full height image header */


.w3-bar .w3-button {

    padding: 16px;

}

.mySlides {display:none;

}

slider

{
```

```
transition: 1s;

}


</style>

<body>


<!-- Navbar (sit on top) -->

<div class="w3-top">

  <div class="w3-bar w3-white w3-card" id="myNavbar">

    <a href="#home" class="w3-bar-item w3-button w3-wide"><i class="fa fa-headphones"
style="font-size:24px;color:red"></i></a>

    <!-- Right-sided navbar links -->

    <div class="w3-right w3-hide-small">

      <a href="#about" class="w3-bar-item w3-button"><i class="fa fa-info-
circle"></i>ABOUT</a>

      <a href="#team" class="w3-bar-item w3-button"><i class="fa fa-user"></i> BIO</a>

      <a href="#contact" class="w3-bar-item w3-button"><i class="fa fa-envelope"></i>
CONTACT</a>

      {% if user.is_authenticated %}

      <a href="/logout" class="w3-bar-item w3-button"><i class="glyphicon glyphicon-log-
out"></i> LOGOUT</a>

      {% endif %}

    </div>

    <!-- Hide right-floated links on small screens and replace them with a menu icon -->


  </div>

</div>



<!-- Header with full-height image -->
```

```
<header class="bgimg-1 w3-display-container w3-grayscale-min" id="home">
  <div id="slider">
<img class="mySlides" src="{% static 'images/11.jpg'%}" style="width:100%">
  <img class="mySlides" src="{% static 'images/22.jpg'%}" style="width:100%">
  <img class="mySlides" src="{% static 'images/33.jpg'%}" style="width:100%">
    <img class="mySlides" src="{% static 'images/lz.jpg'%}" style="width:100%">
</div>
    <script>
var myIndex = 0;
carousel();

function carousel() {
    var i;
    var x = document.getElementsByClassName("mySlides");
    for (i = 0; i < x.length; i++) {
       x[i].style.display = "none";
    }
    myIndex++;
    if (myIndex > x.length) {myIndex = 1}
    x[myIndex-1].style.display = "block";
    setTimeout(carousel, 5000); // Change image every 2 seconds
}
</script>

  <div class="w3-display-left w3-text-white" style="padding:48px">
    <span class="w3-jumbo w3-hide-small">AdderNet Music App</span><br>
    <span class="w3-xxlarge w3-hide-large w3-hide-medium">Pink Floyd Music App</span><br>
```

```
      {% if user.is_authenticated %}
      <span class="w3-large">Welcome {{user.username}}</span>
   <!--<a href="/albums" class="w3-button w3-white w3-padding-large w3-large w3-margin-top w3-opacity w3-hover-opacity-off">View Albums</a>-->
      {% else %}
      <span class="w3-large">Listen and download your favorite songs for free without any ads. What are you waiting for?</span>
    <p><a href="{% url 'music:albums' %}" class="w3-button w3-white w3-padding-medium w3-medium w3-margin-top w3-opacity w3-hover-opacity-off">View Albums</a>

    {% endif %}
  </div>
</header>

<!-- About Section -->
<div class="w3-container" style="padding:128px 16px" id="about">
  <h3 class="w3-center">ABOUT OUR WEBSITE</h3>
  <p class="w3-center w3-large">Key features</p>
  <div class="w3-row-padding w3-center" style="margin-top:64px">
    <div class="w3-quarter">
      <i class="fa fa-music w3-margin-bottom w3-jumbo w3-center"></i>
      <p class="w3-large">HQ Audio</p>
      <p>Listen and download high quality audio files.</p>
    </div>
    <div class="w3-quarter">
      <i class="fa fa-download w3-margin-bottom w3-jumbo"></i>
      <p class="w3-large">Free Download</p>
      <p>Download any number of songs you want for free.</p>
    </div>
```

```
    <div class="w3-quarter">
      <i class="fa fa-list w3-margin-bottom w3-jumbo"></i>
      <p class="w3-large">Playlist</p>
      <p>Create and edit your own playlist and access it from any device.</p>
    </div>
    <div class="w3-quarter">
      <i class="fa fa-cog w3-margin-bottom w3-jumbo"></i>
      <p class="w3-large">Support</p>
      <p>We provide assistance to our user 24*7. </p>
    </div>
  </div>
</div>

<!-- Team Section -->
<div class="w3-container" style="padding:128px 16px" id="team">
  <h3 class="w3-center">THE BAND</h3>
  <p class="w3-center w3-large"></p>
  <div class="w3-row-padding w3-grayscale" style="margin-top:64px">
    <div class="w3-col l3 m6 w3-margin-bottom">
      <div class="w3-card">
        <img src="{% static 'images/RW1.jpg'%}" alt="John" style="width:100%">
        <div class="w3-container">
          <h3>Roger Waters</h3>
          <p class="w3-opacity">Basist & Vocals</p>
          <p>George Roger Waters (born 6 September 1943) is an English singer,
songwriter, bassist, and composer. In 1965, he co-founded the progressive rock band Pink
Floyd with drummer Nick Mason, keyboardist Richard Wright, and guitarist, singer, and
songwriter Syd Barrett. Waters initially served as the bassist, but following the
departure of Barrett in 1968, he also became their lyricist, co-lead vocalist, and
conceptual leader.</p>
```

```
          <p><button class="w3-button w3-light-grey w3-block"
onclick="location.href='https://en.wikipedia.org/wiki/Roger_Waters';"><i class="fa fa-
info-circle"></i> Know more...</button></p>

        </div>

      </div>

    </div>

    <div class="w3-col l3 m6 w3-margin-bottom">

      <div class="w3-card">

        <img src="{% static 'images/DG1.jpg'%}" alt="Jane" style="width:100%">

        <div class="w3-container">

          <h3>David Gilmour</h3>

          <p class="w3-opacity">Lead Guitarist & Vocals</p>

          <p>David Jon Gilmour, CBE (born 6 March 1946) is an English guitarist, singer
and songwriter best known as a longtime member of the progressive rock band Pink Floyd.
He joined the group as guitarist and co-lead vocalist in 1968, replacing founder member
Syd Barrett.</p>

          <p><button class="w3-button w3-light-grey w3-block"
onclick="location.href='https://en.wikipedia.org/wiki/David_Gilmour';"><i class="fa fa-
info-circle"></i> Know more...</button></p>

        </div>

      </div>

    </div>

    <div class="w3-col l3 m6 w3-margin-bottom">

      <div class="w3-card">

        <img src="{% static 'images/RR1.jpg'%}" alt="Mike" style="width:100%">

        <div class="w3-container">

          <h3>Rick Wright</h3>

          <p class="w3-opacity">Rhythm</p>

          <p>Richard William Wright (28 July 1943 – 15 September 2008) was an English
musician, composer, singer, and songwriter. He was a founder member, keyboardist, and
vocalist of the progressive rock band Pink Floyd, performing on all but one of the
```

```
group's albums including The Piper at the Gates of Dawn, The Dark Side of the Moon, Wish You Were Here and The Division Bell, and playing on all of their tours.</p>

          <p><button class="w3-button w3-light-grey w3-block" onclick="location.href='https://en.wikipedia.org/wiki/Richard_Wright_%28musician%29';"><i class="fa fa-info-circle"></i> Know more...</button></p>

        </div>

      </div>

    </div>

    <div class="w3-col l3 m6 w3-margin-bottom">

      <div class="w3-card">

        <img src="{% static 'images/NM1.jpg'%}" alt="Dan" style="width:100%">

        <div class="w3-container">

          <h3>Nick Mason</h3>

          <p class="w3-opacity">Drummer</p>

          <p>Nicholas Berkeley Mason (born 27 January 1944) is an English drummer, best known as a founding member of the progressive rock group Pink Floyd.</p>

          <p><button class="w3-button w3-light-grey w3-block" onclick="location.href='https://en.wikipedia.org/wiki/Nick_Mason';"><i class="fa fa-info-circle"></i> Know more...</button></p>

        </div>

      </div>

    </div>

  </div>

</div>



<!-- Modal for full size images on click-->

<div id="modal01" class="w3-modal w3-black" onclick="this.style.display='none'">

  <span class="w3-button w3-xxlarge w3-black w3-padding-large w3-display-topright" title="Close Modal Image">×</span>

  <div class="w3-modal-content w3-animate-zoom w3-center w3-transparent w3-padding-64">
```

```
    <img id="img01" class="w3-image">
    <p id="caption" class="w3-opacity w3-large"></p>
  </div>
</div>



<!-- Contact Section -->
<div class="w3-container w3-light-grey" style="padding:128px 16px" id="contact">
  <h3 class="w3-center">CONTACT</h3>
  <p class="w3-center w3-large">Lets get in touch. Send us a message:</p>
  <div class="w3-row-padding" style="margin-top:64px">
    <div class="w3-half">
      <p><i class="fa fa-map-marker fa-fw w3-xxlarge w3-margin-right"></i> Kolkata,
WestBengal</p>
      <p><i class="fa fa-phone fa-fw w3-xxlarge w3-margin-right"></i> Phone: +91
8697036339</p>
      <p><i class="fa fa-envelope fa-fw w3-xxlarge w3-margin-right"> </i> Email:
samwise.mj@gmail.com</p>
      <br>
      <form action="/action_page.php" target="_blank">
        <p><input class="w3-input w3-border" type="text" placeholder="Name" required
name="Name"></p>
        <p><input class="w3-input w3-border" type="text" placeholder="Email" required
name="Email"></p>
        <p><input class="w3-input w3-border" type="text" placeholder="Subject" required
name="Subject"></p>
        <p><input class="w3-input w3-border" type="text" placeholder="Message" required
name="Message"></p>
        <p>
          <button class="w3-button w3-black" type="submit">
            <i class="fa fa-paper-plane"></i> SEND MESSAGE
```

```
          </button>
        </p>
      </form>
    </div>
    <div class="w3-half">
      <!-- Add Google Maps -->
      <!--<div id="map" class="w3-greyscale-max" style="width:100%;height:510px;"></div>-
->




    </div>
  </div>
</div>



<!-- Footer -->
<footer class="w3-center w3-black w3-padding-64">
  <a href="#home" class="w3-button w3-light-grey"><i class="fa fa-arrow-up w3-margin-
right"></i>To the top</a>
  <div class="w3-xlarge w3-section">
    <i class="fa fa-facebook-official w3-hover-opacity"></i>
    <i class="fa fa-instagram w3-hover-opacity"></i>
    <i class="fa fa-snapchat w3-hover-opacity"></i>
    <i class="fa fa-pinterest-p w3-hover-opacity"></i>
    <i class="fa fa-twitter w3-hover-opacity"></i>
    <i class="fa fa-linkedin w3-hover-opacity"></i>
  </div>
</footer>
<!-- Add Google Maps -->
```

```
<!--
To use this code on your website, get a free API key from Google.
Read more at: https://www.w3schools.com/graphics/google_maps_basic.asp
-->

</body>
</html>
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<style>

img {
    border: 1px solid #ddd; /* Gray border */
    border-radius: 4px;  /* Rounded border */
    padding: 5px; /* Some padding */
    width: 150px; /* Set a small width */
    opacity: 1.0;
    filter: alpha(opacity=100);
  -webkit-transition: -webkit-transform .8s ease-in-out;
          transition:         transform .8s ease-in-out;
}
img:hover {
  -webkit-transform: rotate(360deg);
          transform: rotate(360deg);
}
```

```
a:hover {
    color: green;
}
a{font-size: 16px;
    }

  </style>



<link rel = "stylesheet" href =
"https://fonts.googleapis.com/icon?family=Material+Icons">
<link rel = "stylesheet" href =
"https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.3/css/materialize.min.css">


<script type = "text/javascript" src = "https://code.jquery.com/jquery-
2.1.1.min.js"></script>
<script src =
"https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.3/js/materialize.min.js"></scrip
t>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>


<body>


  <nav>
    <div class="nav-wrapper green" >
      <a href="#" class="brand-logo">Albums</a>
      <ul id="nav-mobile" class="right hide-on-med-and-down">


        <li><a href="{% url 'music:vplist' %}">View Playlist</a></li>
      </ul>
```

```
    </div>
  </nav>


          <!--{% for album in all_albums %}-->
        <!--{% if forloop.first %}<div class="row">{% endif %}-->
            <!--<div class="column">-->
                <!--<img src="{{album.album_logo}}">-->
                <!--</img>-->
            <!--</div>-->
        <!--{% if forloop.counter|divisibleby:"3" %}</div><div class="row">{% endif %}-->
        <!--{% if forloop.last %}</div>{% endif %}-->
    <!--{% endfor %}-->



<table class="striped">
                    <tr>
                        {% for album in all_albums %}
                        <td>
                            <a href="{% url 'music:detail' album.id %}">
                             <img src="{{album.album_logo}}"
class="circle">{{album.album_title}}</a>



                        </td>
                        {% if forloop.counter|divisibleby:3 %}
                    </tr>
                    <tr>
                        {% endif %}
```

```
                    {% endfor %}
                </tr>
</table>
</body>
</html>
```

songlist.html

```
<html>
<head>
    <style>
    h3
    {
    font-family: Berlin, sans serif;
    color: #25383C;
    }
    h3:hover
    {
    color: blue;
    font-size: 50px;
    transition: 2s;
    }
        img
        {
        margin-left:250px;
        }
        tr:hover
```

```
        {
        background-color: #98AFC7;
        font-weight: bold;
        }
        table
        {
        background-color: #BCC6CC;
    </style>
<link rel = "stylesheet" href =
"https://fonts.googleapis.com/icon?family=Material+Icons">
<link rel = "stylesheet" href =
"https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.3/css/materialize.min.css">


<script type = "text/javascript" src = "https://code.jquery.com/jquery-
2.1.1.min.js"></script>
<script src =
"https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.3/js/materialize.min.js"></scrip
t>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>



<body>
  <nav>
    <div class="nav-wrapper green" >
      <a href="#" class="brand-logo">{{cuser}}</a>
      <ul id="nav-mobile" class="right hide-on-med-and-down">
        <li><a href="{% url 'music:index' %}">Albums</a></li>
        <li><a href="{% url 'music:vplist' %}">View Playlist</a></li>
          <li><a href="{% url 'music:home' %}">Home</a></li>
      </ul>
```

```
    </div>
  </nav>



{#aka details.html#}
{%  if error_msg %}
<p><strong>error_msg</strong></p>
{% endif %}
<script>
    function Replace(e)
    {
        e.value="Added";



    }
</script>
  <script>
      document.addEventListener('play', function(e){
    var audios = document.getElementsByTagName('audio');
    for(var i = 0, len = audios.length; i < len;i++){
        if(audios[i] != e.target){
            audios[i].pause();
        }
    }
}, true);
  </script>
<h3 align="center">{{ album.album_title }}</h3><br>
```

```
<!--<table align="center">-->

<!--<tr align="center"><td>-->

    <!--<img src="{{ album.album_logo }}" style="width:600px;height:300px;">-->

    <!--</td>-->

<!--</tr>-->

<!--</table>-->


  <div class="container">

    <img class="materialboxed" width="450" src="{{album.album_logo}} "></div>


{# <form action="{% url 'music:favourite' album.id %}" method="post">#}

 {% csrf_token %}

      <div class="container">

<table class="striped">

    {% for songs in album.song_set.all %}

        <tr>

            <td>{{ songs.song_title }}</td>

               <td>

                   <audio controls><source src="http://10.11.48.51:8080/Pink Floyd/{{
album.album_title }}/{{ songs.song_title }}.mp3" type="audio/mpeg"></audio>

                   <!--<a href="{% url 'music:plist' album.id songs.id
%}">AddToFav</a><br>-->

                    <td><a class="btn-floating pulse cyan tooltipped" data-
position="right" data-tooltip="Add To Favourites"  href="{% url 'music:plist' songs.id
album.id %}"><i class="material-icons">playlist_add</i></a></td>

            </td>

        </tr>

    {% endfor %}

</table>
```

```
      </div>
    <!--<a href="{% url 'music:vplist' %}">Show Playlist</a>-->
  </form></body></html>
```

playlist.html

```
{#aka details.html#}

{% load staticfiles %}

{#<link rel="stylesheet" href="{% static 'css/sl.css' %}">#}

<link rel = "stylesheet" href =
"https://fonts.googleapis.com/icon?family=Material+Icons">

<link rel = "stylesheet" href =
"https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.3/css/materialize.min.css">



<script type = "text/javascript" src = "https://code.jquery.com/jquery-
2.1.1.min.js"></script>

<script src =
"https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.3/js/materialize.min.js"></scrip
t>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

{#<script type = "text/javascript" src = "{% static 'js/ext.js' %}"></script>#}




<nav>

    <div class="nav-wrapper blue">

        <a href="" class="brand-logo">Favourites</a>

      <ul id="nav-mobile" class="right hide-on-med-and-down">

     <li><a href="{% url 'music:index' %}">BackToSongs</a></li>

        <li><a href="http://localhost:3000/">Home</a></li>

        <li><a href="">Contact Us</a></li>

      </ul>
```

```
    </div>
</nav>
 <div class="card-panel teal lighten-2">
{%  if msg %}
<p><strong>RemovedFromFavs</strong></p>
{% endif %}


{#    <div class="parallax-container">#}
{#      <div class="parallax"><img src="{% static 'images/bgimg.jpg' %}"></div>#}
{#    </div>#}


<div class="container">
<table class="striped">
   {% for pl in all_pl %}
        {% ifequal pl.userid cuser %}
          <tr>
          <td>{{ pl.song.song_title }}</td>
              <td> <audio controls><source src="http://10.11.48.51:8080/Pink Floyd/{{
pl.song.album.album_title }}/{{ pl.song.song_title }}.mp3"
type="audio/mpeg"></audio></td>
            <td><a class="btn-floating  pulse red tooltipped" data-position="right" data-
tooltip="Remove"  href="{% url 'music:delsong' pl.id %}"><i class="material-
icons">clear</i></a></td>
          </tr>
        {% endifequal %}
    {% endfor %}
</table></div></div>
```