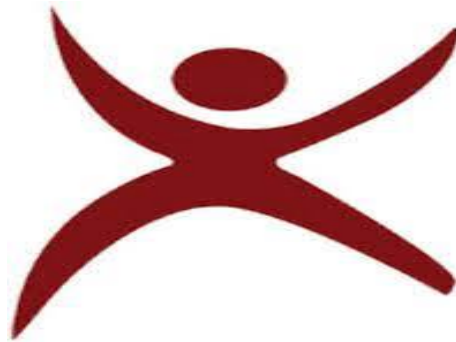


MOOD BASED MUSIC PLAYER

MAJOR PROJECT REPORT

Submitted By

P.SOWJANYA	:R161165
V.KESAVI	:R161521
V.JAYASREE	:R161306



Under the supervision of
Mr. N Satyanandaram
Assistant Professor

Department of Computer Science and Engineering

As a part of

Partial fulfillment of the degree of Bachelor of Technology in
Computer Science and Engineering



CERTIFICATE

This is to certify that the report entitled Mood Based Music Player submitted by V.Jayasree, P.Sowjanya, V.Kesavi in partial fulfilment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out by them under my supervision and guidance.

Mr. N. Satyanandaram

Project Guide,
Assistant Professor,
Computer Science and Engineering,
RGUKT,RK Valley

Mrs. Ch. Ratnakumari,

Head of the Department,
Assistant Professor,
Computer Science Engineering,
RGUKT,RK Valley



DECLARATION

We are hereby declare that this report entitled “Mood Based Music Player” submitted by us under the guidance and supervision of Mr.N Satyanandaram is a bonafide work. we also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma.

Date: 27-04-2022
Place:RK Valley

V.Jayasree(R161306)
P.Sowjanya (R161165)
V.Kesavi(R161521)



ACKNOWLEDGMENTS

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We would like to express my sincere gratitude to **N Satyanandaram sir** for valuable suggestions and keen interest throughout the progress of my project.

At the outset, We would like to thank **Rajiv Gandhi University of Knowledge and Technologies, R.K Valley** for providing all the necessary resources for the successful completion of our course work.

With Sincere Regards,
V.Jayasree
P.Sowjanya
V.kesavi

Table of Contents

1. Abstract-----	6
2. Introduction-----	7
2.1 About Mood Based Music Player	
2.2 Objective	
2.3 Purpose	
3. System Configuration-----	8
3.1 Software Requirements	
3.2 Hardware Requirements	
4. Technologies-----	9-16
4.1 HTML	
4.2 CSS	
4.3 BOOTSRAP	
4.4 PYTHON	
4.5 DJANGO	
5. Feasibility Report-----	17
5.1 Economical Feasibility	
5.2 Technical Feasibility	
5.3 Social Feasibility	
6. System Requirement Analysis-----	18-20
6.1 Existing System	
6.2 Proposed System	
6.3 Functionalities	
6.4 Representation of Framework	
7. Sample Code-----	21-30
8. Output Screens-----	31-32
9. Non-Functional Requirements-----	33
10. Conclusion-----	34
11. Future Scope-----	35

1.ABSTRACT

Music is a vital mood controller and helps in improving the mood. It also helps a person to bounce back to normal mood when he is stressed out. Continuous music play requires creating and managing personalized song playlist which is a time consuming task. It would be very helpful if the music player itself selects a song according to the current mood of the user. The mood of the user can be detected by the inputs given by him about his mood. Our project is aimed to provide people with befitting music using the inputs given by him, saving the time and thereby enhancing user experience.

2.INTRODUCTION

2.1 About Mood Based Music Player

As we all know that music has the potential to change are state of mind and everybody loves listening to music if it matches their current mood but it becomes really frustrating and even time consuming to select music of their choice from a never ending playlist. Our project is based on emotion detection of the user according to his input and then sorting the playlist according to the mood detected. Music is often described as a “language of emotions” While processing music brain’s language Centre, emotional Centre and memory Centre are connected thereby stimulating a thrill obtained by expected beats in a pattern to provide a synesthetic experience. Existing music player satisfies the user’s basic requirements, yet the user has to face the task of manually browsing through the playlist of songs and select songs based on his current mood and behaviour. This project is based on the principle of detection of human emotions according to his mood to play appropriate songs for current emotional state.

2.2 Objective

The objective of this project is to display the playlist of songs according to the input given by the user about his mood.

2.3 Purpose

The purpose of this application is to provide people with befitting music using the inputs given by him as well as data collected from already selected songs by the user, saving the time and thereby enhancing user experience.

3.SYSTEM REQUIREMENTS

3.1 Hardware Requirements

Processor : 1.9 GHz or 64-bit dual core processor

HDD : 250GB

RAM : 4GB

3.2 Software Requirements

Operating System : LINUX, WINDOWS

Coding Language : PYTHON, DJANGO

Front End : HTML, CSS, BOOTSTRAP

IDE Tools : VISUAL STUDIO

4.TECHNOLOGIES

HTML

HTML is used to create electronic documents (called pages) that are displayed on the World Wide Web. Each page contains a series of connections to their pages called hyperlinks. HTML provides the basic structure of sites, which is enhanced and modified by other technologies like CSS and JavaScript.

CSS

CSS is the language for describing the presentation of Web pages, including Colours, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is used to presentation, formatting, and layout.

BOOTSTRAP

Bootstrap is free and opensource, css framework directed at responsive, mobile- first front end web development. It contains css and js based design templates for typography, forms, buttons, navigations and other interface components.

PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

DJANGO

Django is an MVT web framework that is used to build web applications. The huge Django web-framework comes with so many “batteries included” that developers often get amazed as to how everything manages to work together. The principle behind adding so many batteries is to have common web functionalities in the framework itself instead of adding latter as a separate library. One of the main reasons behind the popularity of Django framework is the huge Django community. The community is so huge that a separate website was devoted to it where developers from all corners developed third-party packages including authentication, authorization, full-fledged Django powered CMS systems, e-commerce add-ons and so on.

Features of Django

Rapid Development

Django was designed with the intention to make a framework which takes less time to build web application. The project implementation phase is a very time taken but Django creates it rapidly.

Secure

Django takes security seriously and helps developers to avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery etc. Its user authentication system provides a secure way to manage user accounts and passwords.

Scalable

Django is scalable in nature and has ability to quickly and flexibly switch from small to large scale application project.

Fully loaded

Django includes various helping task modules and libraries which can be used to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds etc.

Versatile

Django is versatile in nature which allows it to build applications for different-different domains. Now a days, Companies are using Django to build various types of applications like: content management systems, social networks sites or scientific computing platforms etc.

Open Source

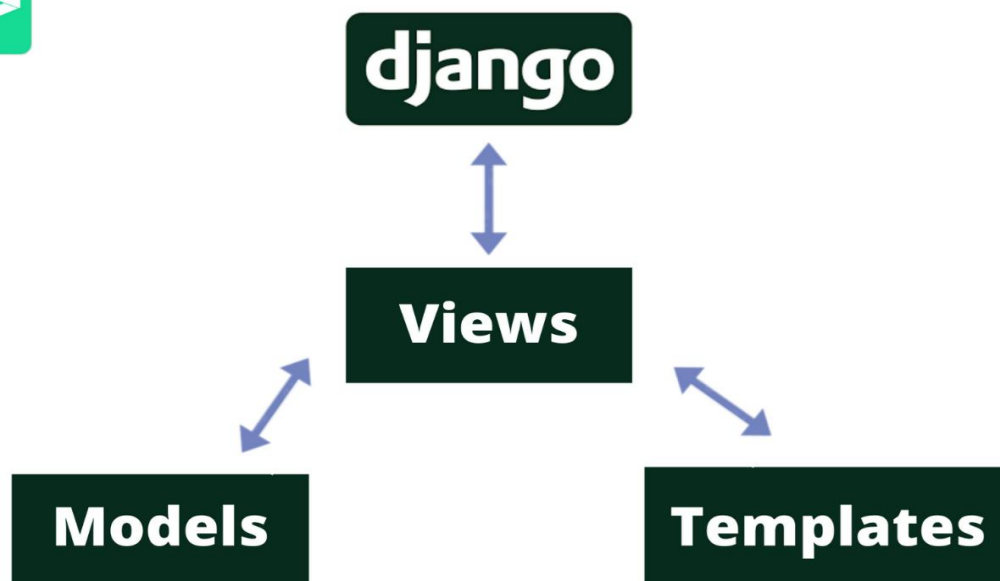
Django is an open source web application framework. It is publicly available without cost. It can be downloaded with source code from the public repository. Open source reduces the total cost of the application development.

Vast and Supported Community

Django is an one of the most popular web framework. It has widely supportive community and channels to share and connect.

Django URLs And Views

A request in Django first comes to urls.py and then goes to the matching function in views.py. Python functions in views.py take the web request from urls.py and give the web response to templates. It may go to the data access layer in models.py as per the query set.



If we look at the 3-tier architecture of an app. Views are like the business logic layer. It is the controller in a typical MVC(Model View Controller) design but Django has a slightly different naming convention called MVT (Model View Template) where:

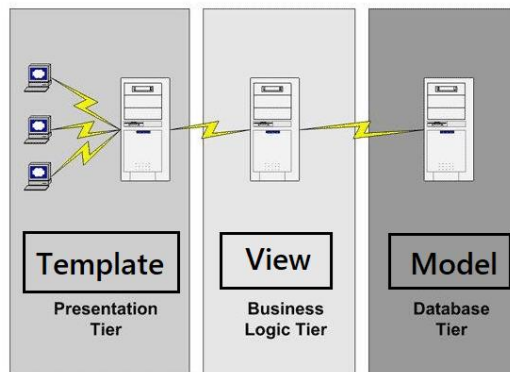
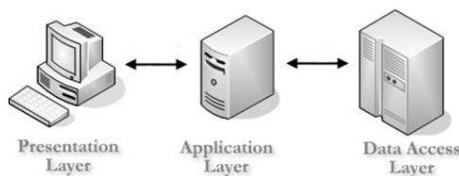
Model is the data access layer,

View is the business logic layer and

Template is the presentation layer.

Django MVT

3 - tier architecture



Django url's Path

Django has a **urls.py** file under the project by default. It also has a pre-defined path for the *admin* app. However, Django recommends mapping all resources via another *urls.py* newly created under the app. The below explains it:

```
urls.py
songs > app > urls.py > ...
1
2 from django.urls import path
3 from . import views
4 urlpatterns=[
5     path('',views.index),
6 ]
7
```

Django View Function

The URL mapping will redirect requests from project URLs to app URLs and then to the respective view function. A sample view function code may look like this:

Here, the **request** is the URL request mapping and calling the view function. **Render** combines a given template with a given context dictionary. **{}** denotes the dictionary of values that can be added to the template context.

```
urls.py  views.py 1
songs > app > views.py > ...
6 from django.shortcuts import render
7
8 def index(request):
9     return render(request,'homepage.html')
10
11
```

GITHUB

GitHub is a Git repository hosting service. It is a web-based service. It is a file or code-sharing service to collaborate with different people. GitHub is a

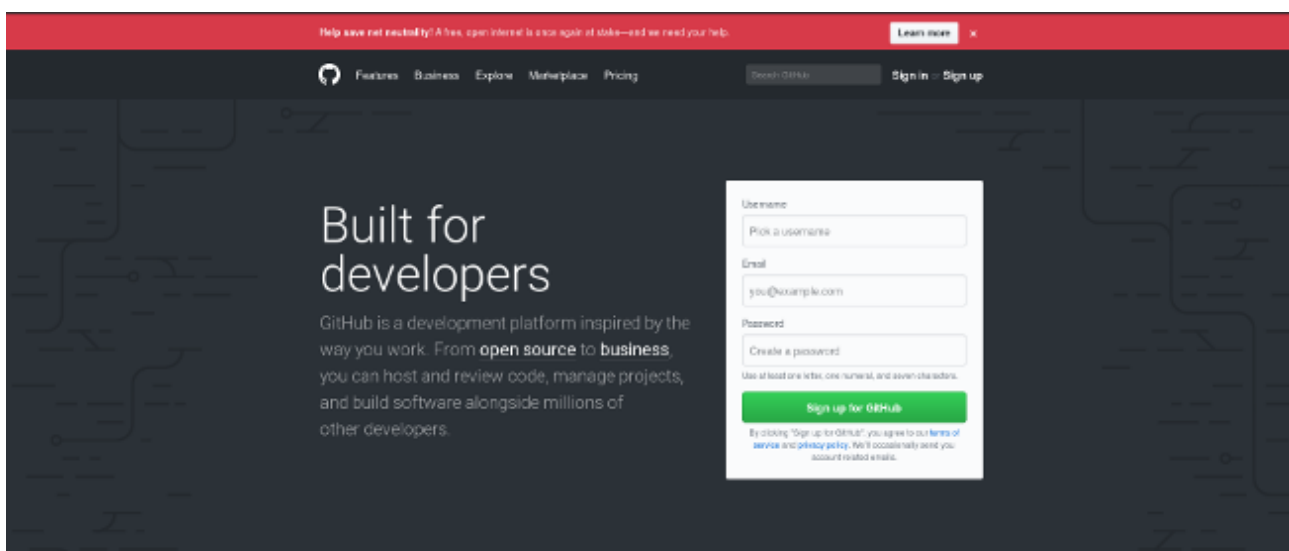
highly used software that is typically used for version control. It is helpful when more than just one person is working on a project.

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

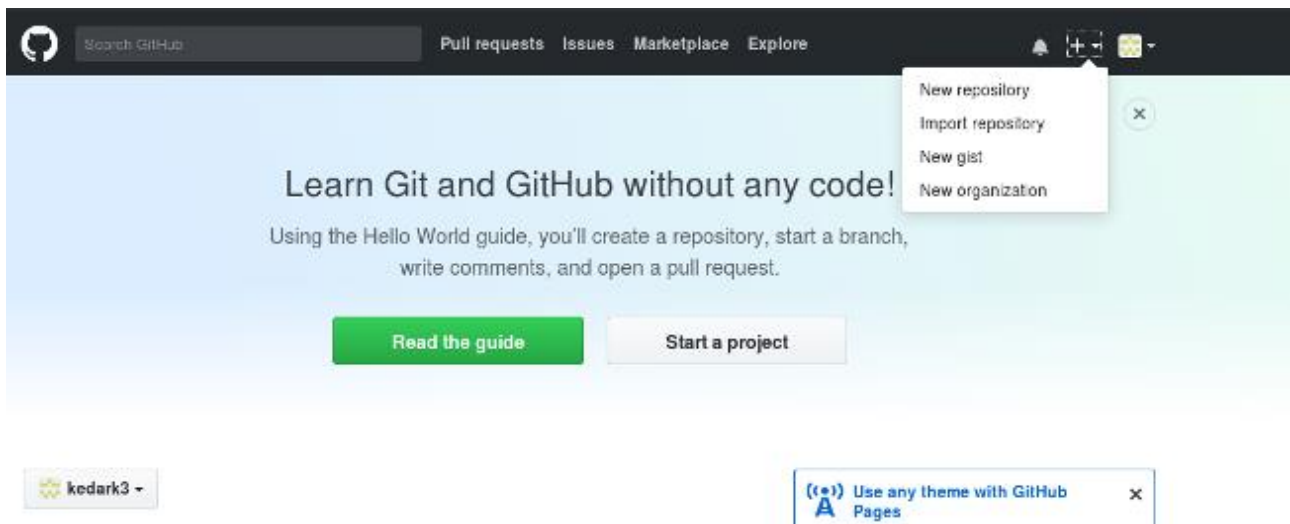
Example: whenever a developer develops some project (like an app) or something, he/she constantly update it catering to the demands of users, technology, and whatsoever it maybe. Version control systems keep these revisions straight, storing the modifications in a central repository. It allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute.

Git is used to storing the source code for a project and track the complete history of all changes to that code, while GitHub is a cloud-based platform built around the Git tool. So it's necessary to upload your android project on GitHub.

Step 1: Create a GitHub account

A screenshot of the GitHub website's sign-up page. The page has a dark background with a faint, light-colored branching diagram. At the top, there is a red banner with white text that reads "Help save net neutrality! A free, open internet is once again at stake—and we need your help." and a "Learn more" link. Below the banner, there is a navigation bar with links for "Features", "Business", "Explore", "Marketplace", and "Pricing". To the right of these links are buttons for "Search GitHub", "Sign in", and "Sign up". The main content area features the text "Built for developers" in a large, white font. Below this, it says "GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside millions of other developers." To the right of this text is a white sign-up form. The form has four input fields: "Username" with a placeholder "Pick a username", "Email" with a placeholder "you@example.com", and "Password" with a placeholder "Create a password". Below the password field is a small note: "Use at least one letter, one numeral, and seven characters." At the bottom of the form is a green button that says "Sign up for GitHub". Below the button is a small disclaimer: "By clicking 'Sign up for GitHub', you agree to our terms of service and privacy policy. We'll occasionally send you account related emails."

Pick a username (e.g., octocat123), enter your email address and a password, and click Sign up for GitHub . Once you are in, it will look something like this:



Step 2: Create a new repository

A repository is like a place or a container where something is stored; in this case we're creating a Git repository to store code. To create a new repository, select **New Repository** from the + sign dropdown menu (you can see I've selected it in the upper-right corner in the image above).

A screenshot of the 'Create a new repository' form on GitHub. The form is titled 'Create a new repository' and includes a subtitle 'A repository contains all the files for your project, including the revision history.' The form has two main sections: 'Owner' and 'Repository name'. The 'Owner' section shows a dropdown menu with 'kedark3' selected. The 'Repository name' section has a text input field. Below these fields, there's a note: 'Great repository names are short and memorable. Need inspiration? How about **fuzzy-sniffle**.' The 'Description (optional)' section has a text area. The 'Public/Private' section has two radio buttons: 'Public' (selected) and 'Private'. The 'Initialize this repository with a README' section has a checkbox that is checked. Below this, there are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None'. At the bottom, there is a green button labeled 'Create repository'.

Enter a name for your repository (e.g, "Demo") and click **Create Repository**. Don't worry about changing any other options on this page.

Congratulations! You have set up your repository on GitHub.com.

To tell your computer that Demo is a directory managed by the Git program, enter the following commands in Terminal

- **git init**
- **git add README.md**
- **git commit -m "first commit"**
- **git branch -M main**
- **git remote add origin https://github.com/<your_username>/Demo.git**
- **git push -u origin main**

That's it! You have created your first GitHub repo, connected it to your computer, and pushed (or uploaded) a file from your computer to your repository called *Demo* on GitHub.com

5.FEASIBILITY REPORT

The feasibility of the project is analysed in this phase and a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential. They are

- 1.ECONOMICALFEASIBILITY
- 2.TECHNICALFEASIBILITY
- 3.SOCIALFEASIBILITY

5.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available.

5.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

5.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user.

6.SYSTEM ANALYSIS

System analysis is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements on the system. System analysis is a problem-solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is viewed as a whole, the inputs are identified and the system is subjected to close study to identify the problem areas. The solutions are given as a proposal. The proposal is reviewed on user request and suitable changes are made. This loop ends as soon as the user is satisfied with the proposal.

6.1 EXISTING SYSTEM

There are several applications that provides facilities and services for music playlist generation or play a particular song and in this process all manual work is involved. Currently, there are many existing music player applications. Some of the interesting applications among them are:

- Saavn and Spotify -These applications give good user accessibility features to play songs and recommends user with other songs of similar genre.
- Wynk Music-This application is a free app that provides unlimited streaming and free unlimited song downloads across Bollywood, regional and international music. Users can search for music by mood, by artist or simply tune into live radio.

Disadvantages:

A person in a bad mood would prefer to listen to music to sooth himself in order to bounce back to normal mood. Though there exist few music players that plays music, it would be irritating to the person to select each song manually and add them to the playlist.

6.2 PROPOSED SYSTEM

Here in this Project we tried to customize the playlist of songs according to the persons mood. The mood of the user can be detected by the inputs given by him about his mood. Our project saves the time of the user by automatically displaying the playlists according to his mood and thereby enhancing user experience.

6.3 FUNCTIONALITIES:

There are two major functionalities to this system.

- 1.Select songs
- 2.Search songs

1.Select Songs

The functionality select songs can further be classified into three categories.

a)Displaying songs according to time

This will take the current local time and display songs according to the time. If the time is between 1:00 AM and 6:00 AM, the system will display “Devotional songs”. If the time is between 6:00 AM and 12:00 AM, the system will display “Love songs”. If the time is between 12:00 AM and 6:00 PM, the system will display “Sad songs”.

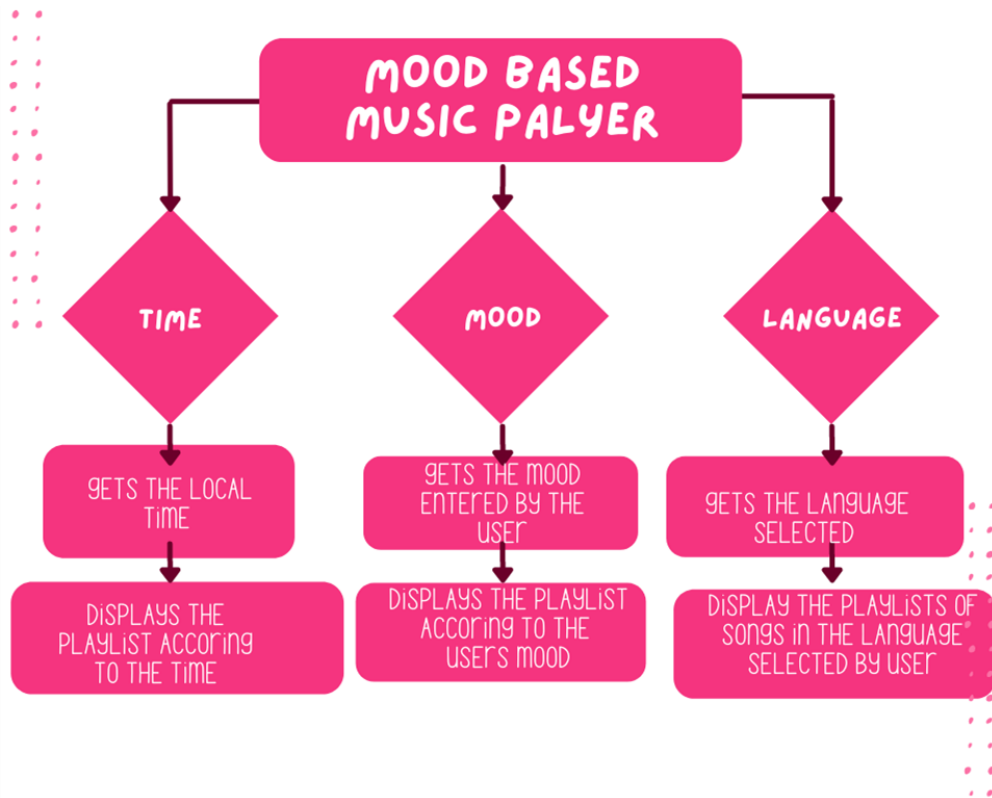
b)Displaying songs according to mood selected by the user

Here the user will input his mood by selecting one from the provided dropdown, then according to the input the list of songs are displayed.

c)Displaying songs according to language selected by the user

Here the user will select language he wanted to listen songs in from the provided drop down, then according to the input the list of songs in that particular language are displayed.

6.4. REPRESENTATION OF FRAMEWORK



7.SAMPLE CODE

BASE.HTML

```
{% extends 'base.html' %}
{% load static %}
{% block style %}
<style>
  header {
    background: linear-gradient(-45deg, #3f51b1 0%, #5a55ae 13%, #7b5fac
25%, #8f6aae 38%, #a86aa4 50%, #cc6b8e 62%, #f18271 75%,
#f3a46987%, #f7c978 100%);
    background-size: 400% 400%;
    animation: gradient 10s ease infinite;
    /* background-size: cover; */
    height: 100vh;
    /* background-position: center;
    background-attachment: fixed;
    background-repeat: no-repeat */
  }
  @keyframes gradient {
    0% {
      background-position: 0%;
    }
    50% {
      background-position: 100%;
    }
    100% {
      background-position: 0%;
    }
  }
}

.content
{
  /* margin-top: 50px; */
  overflow: auto;
  height: calc(100vh - 50px);
}
.youtube-frame
```

```
{
  display:flex;
  justify-content:center;
  margin-top:10vh;
}
```

```
.youtube-frame iframe
{
  width:50vw;
  height:50vh;
  border-radius:50px;
  box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
  transition: all 0.3s cubic-bezier(.25,.8,.25,1);
}
.youtube-frame iframe:hover {
  box-shadow: 0 14px 28px rgba(0,0,0,0.25), 0 10px 10px rgba(0,0,0,0.22);
}
```

```
.navbar .navbar-brand,
.navbar-nav .nav-item .nav-link {
  color: white;
  font-size: 20px;
  margin-right: 10px;
  border-bottom: 2px solid transparent;
  transition: border-bottom 0.5s
}
.navbar-nav .nav-item .nav-link:hover {
  color: white;
  border-bottom: 2px solid black;
}
```

```
</style>
```

```
{% endblock %}
```

```
{% block content %}
```

```
<header>
```

```
<nav class="navbar navbar-expand-lg navbar-dark" style="background-color: transparent; height:50px;">
```

```
<a class="navbar-brand" href="/">Songs Recommended System</a>
```

```
<button class="navbar-toggler" type="button" data-toggle="collapse" data-
```

```
target="#navbarSupportedContent"ariacontrols="navbarSupportedContent
"aria-expanded="false"aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
```

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
<ul class="navbar-nav ml-auto">
<li class="nav-item active">
<a class="nav-link" href="{ % url 'app:time'% }">Time</a>
</li>
<li class="nav-item dropdown active">
<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">weather</a>
<div class="dropdown-menu" aria-labelledby="navbarDropdown">
<a class="dropdown-item" href="/weather/Rain">Rain</a>
<a class="dropdown-item" href="/weather/Summer">Summer</a>
<a class="dropdown-item" href="/weather/Winter">Winter</a>
</div>
</li>
<li class="nav-item dropdown active">
<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">Face expressions</a>
<div class="dropdown-menu" aria-labelledby="navbarDropdown">
<a class="dropdown-item" href="{ % url 'app:sad'% }">Sad</a>
<a class="dropdown-item" href="{ % url 'app:happy'% }">Happy</a>
<a class="dropdown-item" href="{ % url 'app:angry'% }">Anger</a>
</div>
</li>
<li class="nav-item dropdown active">
<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">Language</a>
<div class="dropdown-menu" aria-labelledby="navbarDropdown">
<a class="dropdown-item" href="/change_launage/Telugu">Telugu</a>
<a class="dropdown-item" href="/change_launage/English">English</a>
<a class="dropdown-item" href="/change_launage/Hindhi">Hindhi</a>
```

```

<a class="dropdown-
item"href="/change_launguage/Malayalam">Malayalam</a>
<a class="dropdown-item" href="/change_launguage/Tamil">Tamil</a>
</div>
</li>
<form class="form-inline my-2 my-lg-0" method="POST" action="{ % url
'app:song_search' % }">
  { % csrf_token % }
  <input class="form-control mr-sm-2" type="search" placeholder="Search
For songs" aria-label="Search" name="theme">
  <button class="btn btn-outline-light my-2mysm0"type="submit">Search
</button>
</form>
</ul>
</div>
</nav>
<div class="content">
{ % for link in page_obj % }
<div class="youtube-frame">
<iframe src="{ { link} }" frameborder="0" allow="autoplay; encrypted-
media;" allowfullscreen></iframe>
</div>
{ % endfor % }
</div>
</header>
{ % endblock % }

```

INDEX.HTML

```

{ % extends 'base.html' % }
{ % load static % }
{ % block style % }
<style>
  header {
    background: linear-gradient(-45deg, #3f51b1 0%, #5a55ae 13%,
#7b5fac 25%, #8f6aae 38%, #a86aa4 50%, #cc6b8e 62%, #f18271 75%,
#f3a469 87%, #f7c978 100%);

```



```
background-size: 400% 400%;
animation: gradient 10s ease infinite;
/* background-size: cover; */
height:100vh;
/* background-position: center center;
background-attachment: fixed;
background-repeat: no-repeat */
}
@keyframes gradient {
0% {
background-position: 0%;
}
50% {
background-position: 100%;
}
100% {
background-position: 0%;
}
}
```

```
.content
{
/* margin-top:50px; */
overflow:auto;
height:calc(100vh - 50px);
}
```

```
.youtube-frame
{
display:flex;
justify-content:center;
margin-top:10vh;
}
```

```
.youtube-frame iframe
{
width:50vw;
height:50vh;
border-radius:50px;
box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}
```

```

    transition: all 0.3s cubic-bezier(.25,.8,.25,1);
  }
.youtube-frame iframe:hover {
  box-shadow: 0 14px 28px rgba(0,0,0,0.25), 0 10px 10px rgba(0,0,0,0.22);
}
.navbar .navbar-brand,
.navbar-nav .nav-item .nav-link {
  color: white;
  font-size: 20px;
  margin-right: 10px;
  border-bottom: 2px solid transparent;
  transition: border-bottom 0.5s
}
.navbar-nav .nav-item .nav-link:hover {
  color: white;
  border-bottom: 2px solid black;
}
</style>
{% endblock %}
{% block content %}
<header>
<nav class="navbar navbar-expand-lg navbar-dark" style="background-
color: transparent; height:50px;">
<a class="navbar-brand" href="/">Songs Recommended System</a>
<button class="navbar-toggler" type="button" data-toggle="collapse"
data-
target="#navbarSupportedContent"ariacontrols="navbarSupportedContent
" aria-expanded="false" aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span> </button>
<div class="collapse navbar-collapse" id="navbarSupportedContent">
<ul class="navbar-nav ml-auto">
<li class="nav-item active">
<a class="nav-link" href="{ % url 'app:time'% }">Time</a>
</li>
<li class="nav-item dropdown active">
<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">weather</a>
<div class="dropdown-menu" aria-labelledby="navbarDropdown">

```

```

<a class="dropdown-item" href="/weather/Rain">Rain</a>
<a class="dropdown-item" href="/weather/Summer">Summer</a>
<a class="dropdown-item" href="/weather/Winter">Winter</a>
</div>
</li>
<li class="nav-item dropdown active">
<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">Face expressions</a>
<div class="dropdown-menu" aria-labelledby="navbarDropdown">
<a class="dropdown-item" href="{ % url 'app:sad'% }">Sad</a>
<a class="dropdown-item" href="{ % url 'app:happy'% }">Happy</a>
<a class="dropdown-item" href="{ % url 'app:angry'% }">Anger</a>
</div>
</li>
<li class="nav-item dropdown active">
<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">Language</a>
<div class="dropdown-menu" aria-labelledby="navbarDropdown">
    <a class="dropdown-item"
href="/change_launguage/Telugu">Telugu</a>
    <a class="dropdown-item"
href="/change_launguage/English">English</a>
    <a class="dropdown-item"
href="/change_launguage/Hindhi">Hindhi</a>
    <a class="dropdown-item"
href="/change_launguage/Malayalam">Malayalam</a>
    <a class="dropdown-item"
href="/change_launguage/Tamil">Tamil</a>
</div>
</li>
<form class="form-inline my-2 my-lg-0" method="POST" action="{ % url
'app:song_search' % }">
    { % csrf_token % }
<input class="form-control mr-sm-2" type="search" placeholder="Search
For songs" aria-label="Search" name="theme">
<button class="btn btn-outline-light my-2 my-sm-0"
type="submit">Search</button>

```

```

</form>
</ul>
</div>
</nav>
<div class="content">
{% for link in page_obj %}
  <div class="youtube-frame">
    <iframe src="{{link}}" frameborder="0" allow="autoplay; encrypted-
media;" allowfullscreen></iframe>
  </div>
{% endfor %}
</div>
</header>
{% endblock %}

```

VIEWS.PY

```

import requests
import time as get_time
import json
from django.core.paginator import Paginator
from django.shortcuts import render
from django.http import HttpResponseRedirect

song_language="Telugu"

def get_data(theme):
    # Now using youtube v3 apis
    url
    "https://www.googleapis.com/youtube/v3/search?maxResults=30&q={ }&t
ype=video&key=AIzaSyD8OIFQvf8f6dXuYRSCekWHy9x9gTJJjDyU".fo
rmat(theme+" songs in "+song_language)
    response = requests.get(url)

```

```
data=json.loads(response.text).get("items")
```

```
video_ids=["https://www.youtube.com/embed/"+video_data.get("id").get("videoId") for video_data in data]  
return video_ids
```

```
def index(request):  
    return render(request,'homepage.html')
```

```
def time(request):  
    theme=""  
    t = get_time.localtime()  
    current_time =int(get_time.strftime("%H", t))  
    if current_time>=0 and current_time<6:  
        theme="Devotional"  
    elif current_time>=6 and current_time<12:  
        theme="Love"  
    elif current_time>=12 and current_time<18:  
        theme="Sad"  
    else:  
        theme="Item"  
    links=get_data(theme)  
    return  
render(request,'index.html',{ 'theme':theme,'page_obj':links,'language':song  
_language})
```

```
def weather(request,weather_type):  
    links=get_data(weather_type)  
    return  
render(request,'index.html',{ 'theme':weather_type,'page_obj':links,'language':song_language})
```

```
def happy(request):  
    theme="Happyness"  
    links=get_data(theme)
```

```
    return  
render(request,'index.html',{ 'theme':theme,'page_obj':links,'language':song  
_language})
```

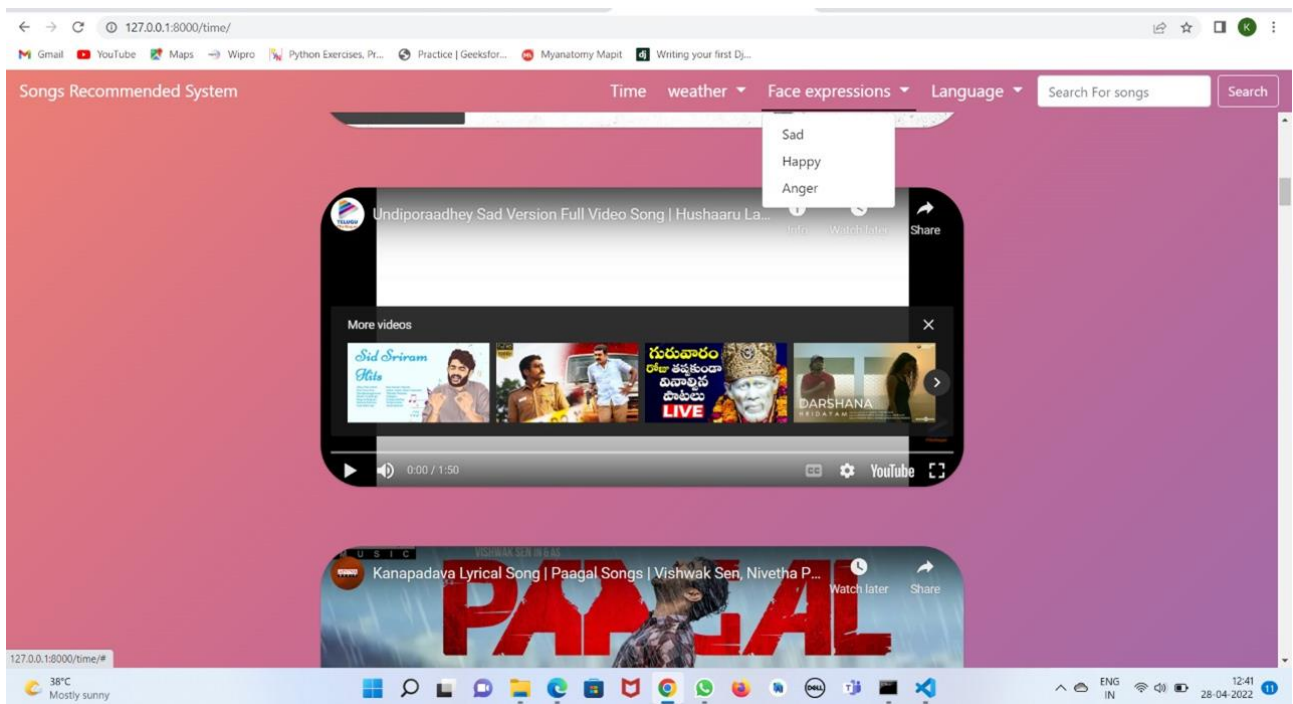
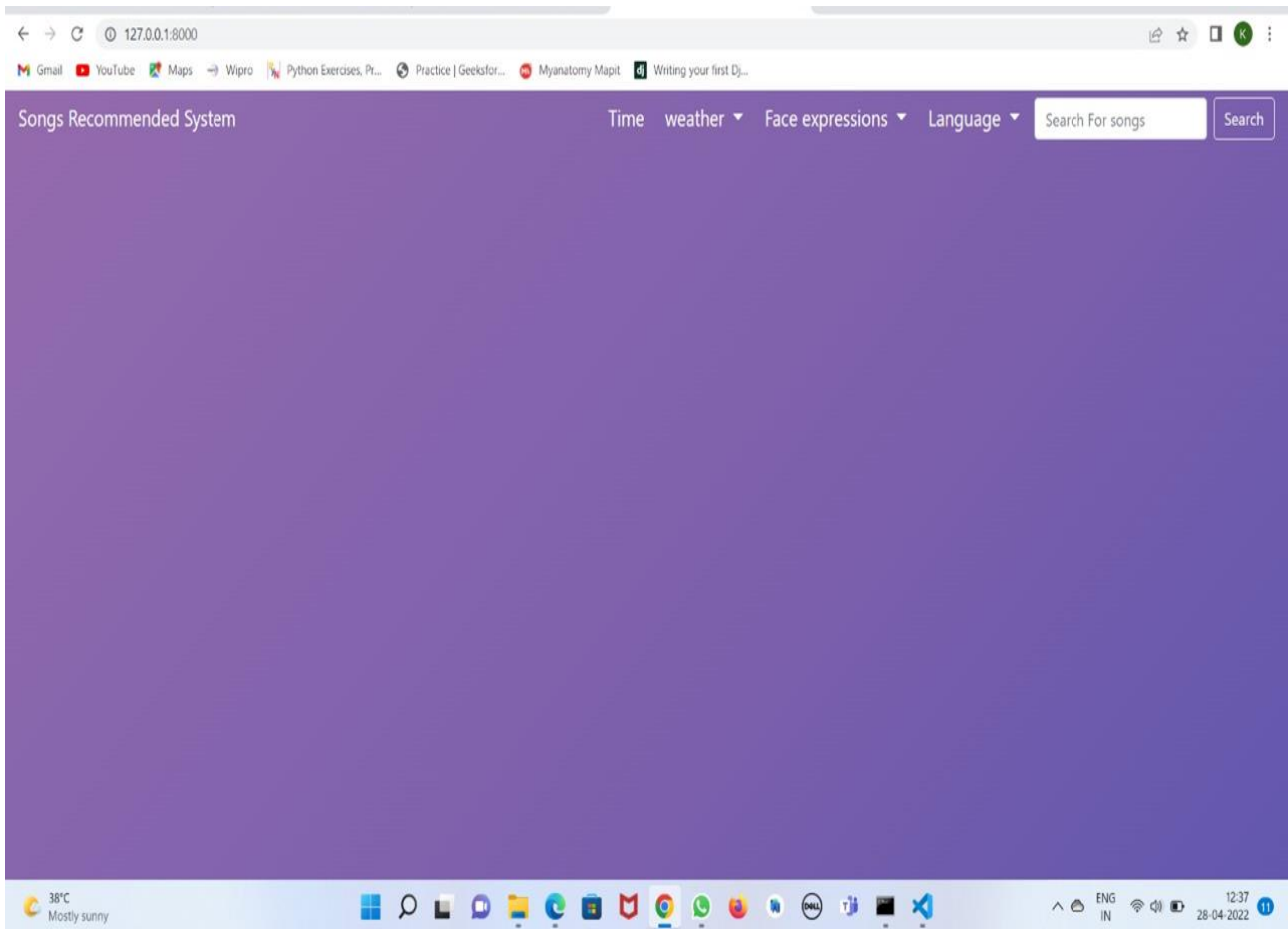
```
def sad(request):  
    theme="Sad"  
    links=get_data(theme)  
    return  
render(request,'index.html',{ 'theme':theme,'page_obj':links,'language':song  
_language})
```

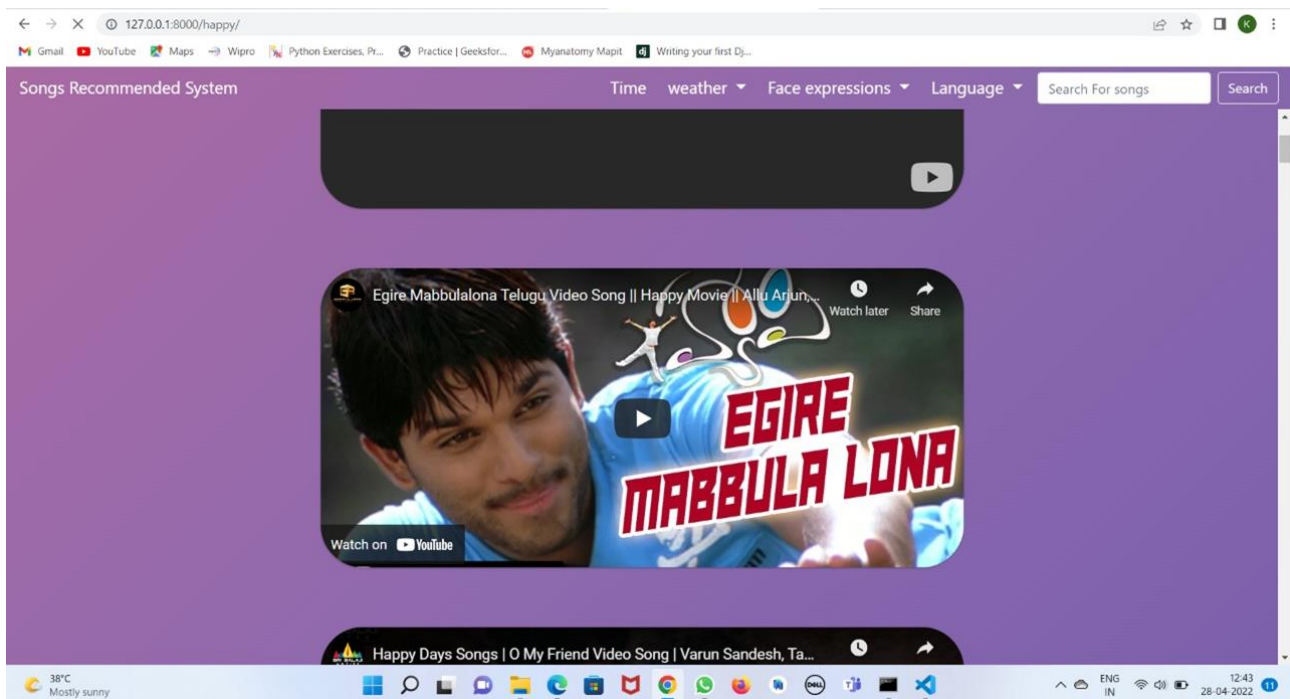
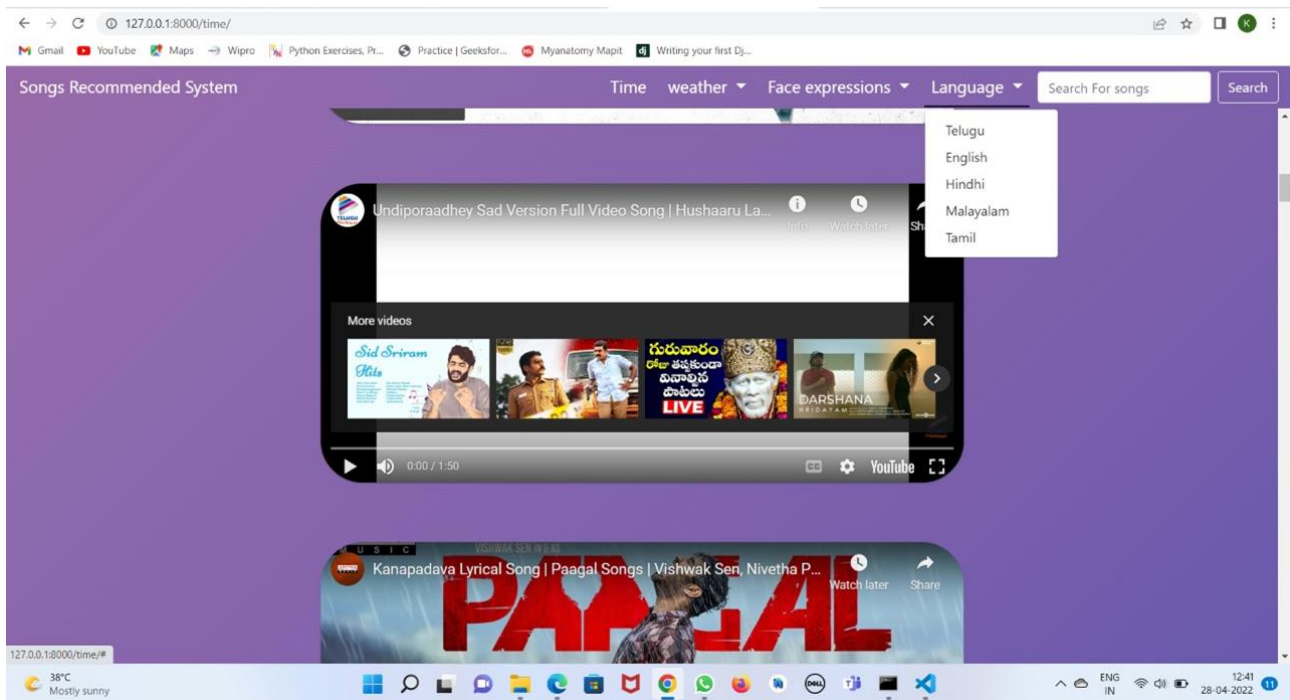
```
def angry(request):  
    theme="Comedy"  
    links=get_data(theme)  
    return  
render(request,'index.html',{ 'theme':theme,'page_obj':links,'language':song  
_language})
```

```
def change_launauge(request,lan):  
    global song_language  
    song_language=lan  
    if(request.META.get('HTTP_REFERER')):  
        return HttpResponseRedirect(request.META.get('HTTP_REFERER'))  
    else:  
        return HttpResponseRedirect("/")
```

```
def song_search(request):  
    theme=""  
    if request.method=="POST":  
        theme=request.POST.get("theme")  
        links=get_data(theme)  
        return  
render(request,'index.html',{ 'theme':theme,'page_obj':links,'language':song  
_language})
```

8.OUTPUT SCREENS





9. NON-FUNCTIONAL REQUIREMENTS

Reliability:

User should get appropriate information about his complaint

Usability:

This tool should have user friendly GUI. User can use it effectively.
Availability: User should get information 24x7. User can access at any time with this tool

Accessibility:

This tool supports multi user accessing. Any user can access the system from different places to use the tool.

Performance:

User should have fast access to get the information from the help centre. User should retrieve the information from help centre database very quickly.

Security:

As it is a web-based application it should be more secure in order to save help centres confidential data from hackers.

Platform Compatibility:

This tool has to work on any kind of operating system without modifying it.

10.CONCLUSION

The Mood-Based Music Player is used to automate and give a better music player experience for the end user. The application solves the basic needs of music listeners without troubling them as existing applications do: it uses increase the interaction of the system with the user in many ways. It eases the work of the end – user by taking the input from the user determining their emotion, and suggesting a customized play-list through a more advanced and interactive system. This application can be improved by modifying and adding few functionalities like adding additional feature of recognizing humans through facial expressions using deep learning.

11.FUTURE SCOPE

- More accurate playlist can be generated.
- Even more compact device can be designed.
- Facial expressions are a great indicator of the state of a mind for a person. Indeed, the most natural way to express emotions is through facial expressions. Humans tend to link the music they listen to; to the emotion they are feeling. The song playlists though are, at times too large to sort out automatically. It can be a great relief if the music player was “smart enough” to sort out the music based on the current state of emotion the person is feeling.
- In future this project sets out to use various techniques like emotion recognition through facial expressions using deep learning.