

EMOTITUNES

A Mini-Project Report Submitted
For
Partial Fulfilment of the Requirements of the
Degree of Bachelor of Engineering

In

COMPUTER ENGINEERING

(Semester VI)
By

Crystal Fernandes (9539)
Sanika Patankar(9563)

Under the guidance of

Prof. Roshni Padate



DEPARTMENT OF COMPUTER ENGINEERING

Fr. Conceicao Rodrigues College of Engineering
Bandra (W), Mumbai - 400050

University of Mumbai

2023-2024

This work is dedicated to my family.

I am very thankful for their motivation and support.

CERTIFICATE

This is to certify that the mini-project entitled “ **EMOTITUNES** ” is a bonafide work of Crystal Fernandes (9539) and Sanika Patankar(9563) submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering in Computer Engineering** (Semester- V).

Prof. Roshni Padate
Guide/ Supervisor

Dr. Sujata Deshmukh
HOD Computer Engg.

Dr. S. S. Rathod
Principal

Approval Sheet

Mini Project Report Approval for T.E. (Semester-VI)

This mini-project report entitled **EMOTITUNES** submitted by Crystal Fernandes (9539), Sanika Patankar(9563) is approved for the degree of Bachelor of Engineering in **Computer Engineering** (Semester-VI).

Examiner 1._____

Examiner 2._____

Date:

Place: Mumbai

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date:

Crystal Fernandes (9539)
Sanika Patankar(9563)

Abstract

This study describes an innovative project that harnesses the capabilities of cloud computing and machine learning techniques to deliver highly personalised music recommendations based on users' facial expressions. The system is designed to analyse and interpret facial expressions in real-time, captured through a camera stream, utilising a sophisticated combination of HTML, CSS, and JavaScript for the frontend interface, Django for robust backend functionalities, TensorFlow for advanced machine learning algorithms, and OpenCV for precise facial image processing. Through this comprehensive integration, the system accurately predicts the user's emotional state, which serves as a pivotal factor in curating music selections tailored to the individual's mood. This approach not only enhances the user experience by offering music that resonates with their current emotional state but also showcases the seamless integration of cutting-edge technologies to deliver a sophisticated and engaging application.

Keywords: Cloud computing, machine learning, personalised recommendations, facial expressions, HTML, CSS, JavaScript, Django, TensorFlow, OpenCV, user experience

Acknowledgments

We have great pleasure in presenting the report on “**EMOTITUNES**”. I take this opportunity to express my sincere thanks towards the guide Prof. Roshni Padate, C.R.C.E, Bandra (W), Mumbai, for providing the technical guidelines, and the suggestions regarding the line of this work. We enjoyed discussing the work progress with him/her during our visits to the department.

We thank Dr. Sujata Deshmukh, Head of Computer Engineering department, Principal and the management of C.R.C.E., Mumbai for encouragement and providing necessary infrastructure for pursuing the project.

We also thank all non-teaching staff for their valuable support, to complete our project.

Date:

Crystal Fernandes (9539)
Sanika Patankar (9563)

Table of Contents

Chapter No	Topic	Page No.
	Abstract	v
1	Introduction	1
2	Objective	2
3	Scope	3
4	Proposed System	4
4.1	Problem Statement	4
5	System Design	5
6	Implementation	6
7	Results	16
8	Conclusion	18

List of Figures

Figure No.	Figure Name	Page No.
1	AWS High Level Architecture	5
2	RDS Connection	6
3	AWS S3 Connection	7
4	AWS Cognito [getTokens and verification]	7
5	AWS Cognito User pool	8
6	Viewing Our Authenticated Users	8
7	GCP OAuth Credentials for Single-sign on	9
8	AWS Cognito Login Page	9
9	AWS Cognito Signup Page	9
10	Single-Sign on Page	9
11	AWS Cognito Verification Code	10
12	AWS S3 Bucket containing all static and media files	11
13	S3 Song Files	11
14	Songs in Happy Mood Folder	12
15	Bucket Policy Updated allowing GetObject	12
16	Created Access Key after creating user	12
17	AWS RDS	13
18	AWS RDS Security Groups	13
19	AWS VPC Security Groups	14
20	AWS VPC Security Groups (Inbound rules)	14
21	AWS VPC Security Groups (Outbound rules)	14
22	AWS EC2 INSTANCE	15
23	EC2 INSTANCE	15
24	Login Page	16
25	Happy Mood	16
26	Sad Mood	17
27	Neutral Mood	17

Chapter 1

INTRODUCTION

In today's digital age, the intersection of technology and human emotion presents exciting opportunities for innovative applications. Our project embodies this convergence by leveraging a combination of advanced technologies to create an interactive web experience that responds to users' moods in real-time.

At its core, our web application utilises the Django web framework to provide a solid foundation for development, offering essential features such as authentication, routing, and database management. Building upon this foundation, we integrate cutting-edge technologies including TensorFlow and OpenCV to analyse and interpret facial expressions, enabling the system to detect users' moods accurately.

Furthermore, we employ Amazon Web Services (AWS) to enhance scalability, reliability, and security. By utilising Amazon S3 for storing static files such as images and audio, we ensure seamless access to multimedia resources while optimising performance. Additionally, our application leverages Amazon RDS for database management, enabling efficient data storage and retrieval.

Moreover, we enhance the user experience by integrating Amazon Cognito for secure authentication, providing users with a seamless login experience across devices. Finally, our application is hosted on Amazon EC2, ensuring high availability and scalability to meet user demand.

In this report, we provide an overview of the technologies utilized, the architecture of our system, and a detailed explanation of the implementation process. Additionally, we discuss the challenges encountered and the strategies employed to overcome them, as well as future enhancements and potential applications of our project.

Chapter 2

OBJECTIVES

1. Implement cloud computing infrastructure to host and manage the system, ensuring scalability, availability, and efficient resource utilisation.
2. Provide personalised music recommendations based on users' emotional states, enhancing the overall user experience by delivering content that aligns with their current mood.
3. User Authentication: Implement user authentication and authorization using Amazon Cognito, providing a secure login mechanism for users and ensuring access control to the application's features and resources.
4. Static File Management: Store static files such as song images and MP3 files in an Amazon S3 bucket, allowing for efficient storage, retrieval, and management of media assets.
5. Database Connectivity: Establish connectivity to an RDS database to store and manage application data, such as user preferences, song metadata, and mood detection results.
6. Emotion Detection Integration: Integrate TensorFlow and OpenCV libraries into the Django website to accurately detect the mood of a person in real-time from images or video streams.
7. Scalable Hosting: Ensure the Django website's scalability and availability, enabling seamless user access without specifying the hosting platform.

Chapter 3

SCOPE OF THE PROJECT

The "Emotitunes" initiative intends to improve music recommendations by making individualised suggestions based on users' emotional states, ultimately improving the entire user experience with content that matches their present mood.

This project includes a thorough integration of AWS Services such as S3, RDS, Cognito, and EC2, which will allow for more effective administration of static file storage, database operations, user authentication, and deployment procedures.

The system uses TensorFlow to perform real-time analysis of users' facial expressions to accurately predict their emotional states, while OpenCV is used to precisely extract key features and information from captured facial expressions, resulting in efficient and effective facial image processing.

"Emotitunes" seeks to provide a smooth and personalised music experience based on each user's emotional environment by integrating innovative technologies.

Chapter 4

PROPOSED SYSTEM

4.1 PROBLEM STATEMENT

In contemporary web development, achieving personalized user experiences that adapt to users' emotions poses a challenge. Existing systems often struggle to integrate real-time emotion detection with dynamic content adaptation, resulting in static and impersonal interactions. Key obstacles include:

- Cloud-based Scalability: Systems lack flexibility in scaling resources based on demand, leading to performance bottlenecks during peak usage.
- Reliability and Availability: Maintaining high availability and reliability in distributed environments requires robust cloud infrastructure to mitigate downtime and service disruptions.
- Cost Management: Balancing performance with cost-effectiveness is challenging, as over-provisioning leads to unnecessary expenses, while under-provisioning compromises performance.

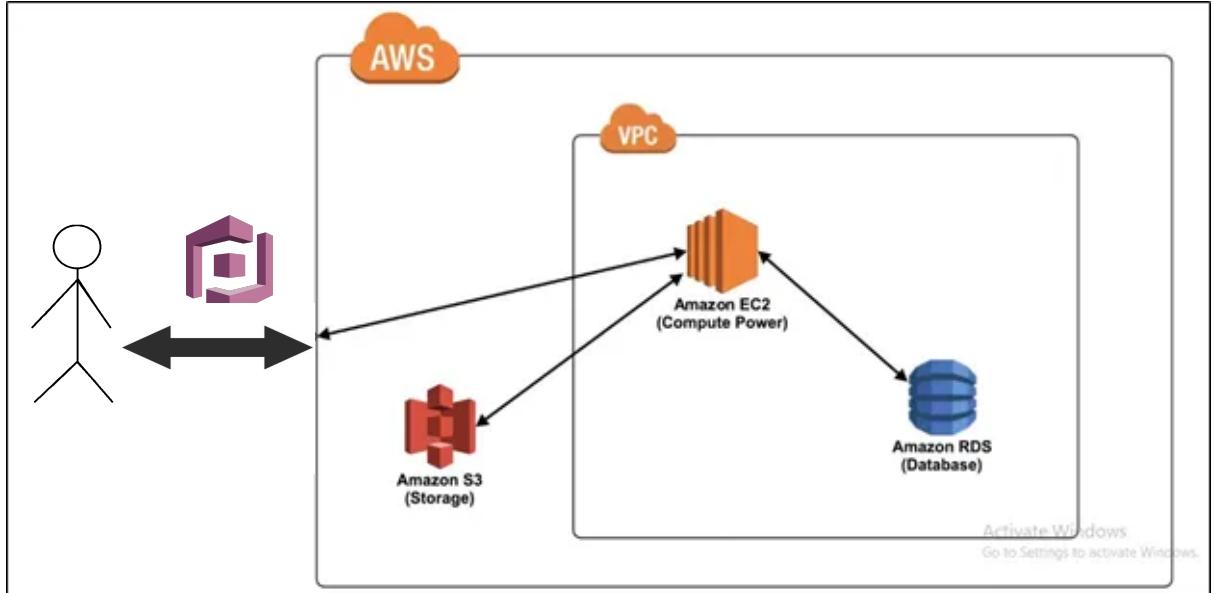
Our Solution Approach:

Our project addresses these challenges by leveraging Amazon Web Services (AWS) to build a scalable and resilient web application infrastructure. By utilizing AWS services such as Amazon EC2 for scalable compute resources, Amazon S3 for reliable storage, Amazon RDS for database management, and Amazon Cognito for secure authentication and user management, we aim to:

- Design a scalable architecture capable of dynamically provisioning compute resources to ensure optimal performance.
- Establish a robust data storage infrastructure for maintaining data integrity and availability.
- Provide a seamless and secure login experience for users, adhering to best practices in user authentication and authorization.
- Through the integration of AWS services, our project aims to create a foundation for delivering personalised and engaging user experiences while addressing the technical challenges of scalability, reliability, and cost management.

Chapter 5

SYSTEM DESIGN



5 AWS High Level Architecture

- **EC2 Instances:** The EC2 instances hosting the web application code are typically deployed within one or more subnets of the VPC. Public-facing EC2 instances serving web traffic are usually placed in public subnets, while backend instances may reside in private subnets for enhanced security.
- **RDS Database:** The RDS database is deployed within a private subnet of the VPC to restrict direct access from the Internet. This ensures that the database remains secure and isolated from unauthorized access.
- **S3 Bucket:** S3 buckets are not directly deployed within subnets, as they are accessed over the Internet by default. However, you can control access to S3 buckets using bucket policies, IAM policies, and VPC endpoints for enhanced security.
- **Cognito User Pool:** While Cognito itself does not reside within subnets, the applications using Cognito (such as web servers hosted on EC2 instances) can be deployed within the VPC's subnets to control inbound and outbound traffic and enforce security policies.

IMPLEMENTATION

CODE:

To connect our RDS to our Django application, we followed these steps:

- We began by configuring PostgreSQL in the RDS console.
- Then, within Django settings, we updated the default database from SQLite to PostgreSQL. Included details like name, user, password, host and port number.
- Migrated all tables to database by using the command makemigration

The screenshot shows a code editor interface with the file 'settings.py' open. The code defines a 'DATABASES' section with a 'default' entry for PostgreSQL, specifying the engine, name, user, password, host, and port. Below this, the 'makemigrations' and 'migrate' commands are run in the terminal, showing successful execution. The terminal output includes migration names like '0001_initial' and '0002_logentry_remove_auto_add'. The code editor has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The terminal tab shows the command line and its output.

Fig. 6.1. RDS Connection

To connect our S3 bucket to our Django application, we followed these steps:

Generating Access Key ID and Secret Key:

- Created a new user in the IAM console with programmatic access and assigned the necessary policy to access the S3 bucket, enabling interaction with AWS services.
- Obtained the Access Key ID and Secret Access Key from the IAM console. These credentials serve as the authentication mechanism for securely accessing AWS resources.

Integration with Django settings.py:

- Integrated the access credentials into the Django application's settings.py file.
- Defined the AWS access key ID and secret key as environment variables within settings.py:

Declared the default file storage and static file storage and ran the command collectstatic

```

126 MEDIA_URL = '/media/'
127
128 # Change the storage backend for media files
129 DEFAULT_FILE_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
130
131 # Change the storage backend for static files
132 STATICFILES_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
133
134 # Remove the "STORAGES" setting
135 # You don't need to define storage backends individually unless you have custom requirements
136
137 # AWS Configuration
138 AWS_ACCESS_KEY_ID = 'AKIA4ERGV3UNH9M7'
139 AWS_SECRET_ACCESS_KEY = 'LuhWqgqVbFrz85nGg7fmGp03NH58r1ZowP3g'
140 AWS_STORAGE_BUCKET_NAME = 'emotitunes-bucket'
141 AWS_S3_CUSTOM_DOMAIN = f'{AWS_STORAGE_BUCKET_NAME}.s3.amazonaws.com'
142 AWS_S3_FILE_OVERWRITE = False
143

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

KeyboardInterrupt
PS C:\Users\VP\Desktop\EmotiTunes> python manage.py collectstatic

You requested to collect static files at the destination location as specified in your settings.

This will overwrite existing files!

Are you sure you want to do this?

Type 'yes' to continue, or 'no' to cancel: yes

128 static files copied.

Script | In 139. Col 56 | Spaces:4 | UTF-8 | CR/LF | Python | 3.11.3 64-bit

Fig 6.2. AWS S3 Connection

To connect our AWS Cognito to our Django application, we followed these steps:

Setting up Cognito User Pool:

- We configured a Cognito User Pool, defining user attributes and authentication methods ensuring that the appropriate client app settings were configured.

Implementing Cognito Authentication in Django:

- We utilized the lambda_handler function to interact with AWS Cognito to verify user tokens and retrieve user information. We integrated the necessary configuration parameters into our Django views.py including the Cognito region name, user pool ID, and client ID.

Handling Authentication Flow in Views:

- The getTokens function handles the token exchange process with Cognito, exchanging the authorization code for user tokens.

Maintaining User Sessions:

- We maintained user sessions through the getSession function which retrieves the user session token from cookies, allowing users to remain authenticated.

```

129
130 # URL that handles the media served from MEDIA_ROOT
131 MEDIA_URL = '/media/'
132
133 # Change the storage backend for media files
134 DEFAULT_FILE_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
135
136 # Change the storage backend for static files
137 STATICFILES_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
138
139 # Remove the "STORAGES" setting
140 # You don't need to define storage backends individually unless you have custom requirements
141
142 # AWS Configuration
143 AWS_ACCESS_KEY_ID = 'AKI44FRHG8YRUX36CMW'

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Token headers: {'kid': '1CKQMyPw4d2PI8HYT6Nd90Ms7EG0I9pbPaonINgqJ=', 'alg': 'RS256'}

Signature verification - SUCCESS

Token claims: {'at_hash': 'AXWu-51hN40_HRk0ymlw', 'sub': '74582438-90a1-70c1-d515-291ce3300ad5', 'cognito:groups': ['us-east-1_FQ0QB1vX_Google'], 'email_verified': False, 'iss': 'https://cognito-idp.us-east-1.amazonaws.com/us-east-1_FQ0QB1vX', 'cognito:username': 'google_103956099659596936203', 'nonce': '55KWh-yXw/20Op95gwFR0nhhf2vZP0x8c81uyvbc3fR_P80h_v2LHRFLDncTRKAzab9AHuUJ1-71-DtEkQs5M-ypWv2IhmfFfholba41QQhWj5Cjyow28qeE7PfpHlR2u_uHPajXJ5/51nPnLyjeECv-Usigk64', 'origin_jti': '3e819fc-a7-61d0-9f92-0cf3f3435b', 'aud': 'ib0s4dn19persenk2ektsb16', 'identities': [{"dateCreated": 1713261839, 'userId': '103956099659596936203', 'providerName': 'Google', 'issuer': None, 'primary': 'true'}], 'token_use': 'id', 'auth_time': 1713261839, 'name': 'sanika patanakar', 'exp': 1713348239, 'iat': 1713261839, 'jti': 'aa3c0637-6023-4af0-bed3-070cb4f708a7', 'email': 'sanik.a912@gmail.com'}

Script | In 137. Col 25 | Spaces:4 | UTF-8 | CR/LF | Select Interpreter | Go Live

Fig 6.3. AWS Cognito [getTokens and verification]

LOGIN/SIGNUP USING AWS COGNITO

After seamlessly integrating AWS Cognito with our Django app, we leveraged its hosted UI domain to provide users with a branded and intuitive authentication experience. Offering seamless login options, users could effortlessly sign in or opt for single sign-on via Google. Configuring OAuth settings with Cognito's domain name and creating OAuth credentials ensured smooth integration for Google sign-in. Additionally, Cognito facilitated email verification upon registration, increasing security measures.

To enhance usability, we maintained user sessions and securely stored user information using cookies. With the flexibility of customizable authentication flows, we tailored the login experience to suit our app's specific requirements, ensuring a seamless and secure user journey.

The screenshot shows the AWS Cognito User Pools service. A user pool named 'pool_for_emotitures' is selected. The 'User pool overview' section displays details such as the pool name, token signing key URL, ARN, and creation time. Below this, the 'Getting started' section provides links for users, groups, sign-in experience, sign-up experience, messaging, app integration, and user pool properties. The 'Users' tab is active, showing 5 users listed in a table with columns for User name, Email address, Email verified, Confirmation status, and Status. The users are: 544834b8-1041-709e-f54..., google_110754794829656..., b4583438-b0d1-7091-f2e..., 541884c8-9011-7039-a26..., and google_1029311976600870... All users are confirmed and enabled. At the bottom, there is an 'Import users (0)' section with a table showing imported users, skipped users, failed users, CloudWatch logs, and created time.

Fig 6.4. AWS Cognito User pool

This screenshot shows the same AWS Cognito User Pools service interface, specifically the 'Users' section. It lists the same five users as in Fig 6.4, all of whom are confirmed and enabled. Below the user list, there is an 'Import users (0)' section which is currently empty, indicating no user import jobs have been created.

Fig 6.5. Viewing Our Authenticated Users

Additional information

Client ID: 298089744498-fau...
Creation date: 9 April 2024 at 19:47:01 GMT+5
Status: Enabled

Client secrets

If you are in the process of changing client secrets, you can manually rotate them without downtime. [Learn more](#)

Client secret: 0CCSPX...
Creation date: 9 April 2024 at 19:47:01 GMT+5
Status: Enabled

+ ADD SECRET

Authorised JavaScript origins

For use with requests from a browser

URIs 1: https://emotitunes.auth.us-east-1.amazoncognito.com

+ ADD URI

Authorised redirect URIs

For use with requests from a web server

URIs 1: https://emotitunes.auth.us-east-1.amazoncognito.com/sso2/idpresponse

+ ADD URI

Note: It may take five minutes to a few hours for settings to take effect

Fig 6.6. GCP OAuth Credentials for Single-sign on

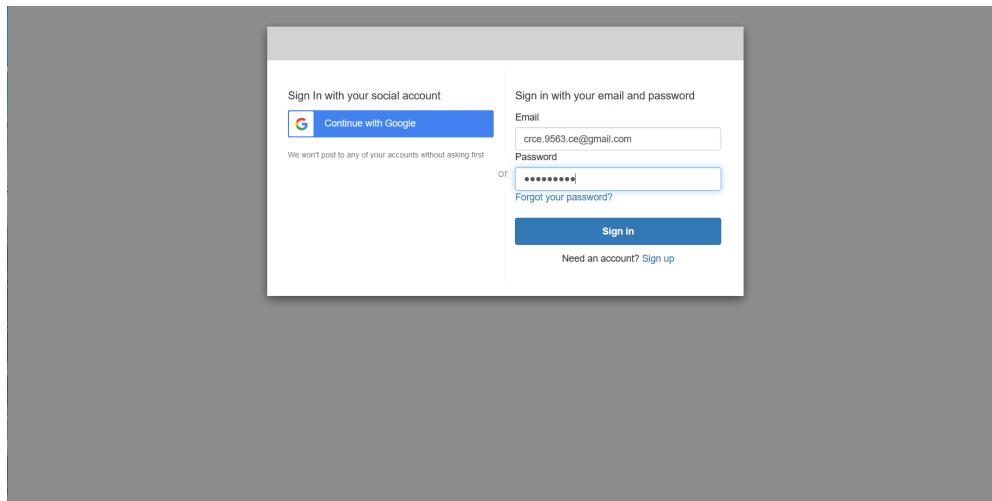


Fig 6.7. AWS Cognito Login Page

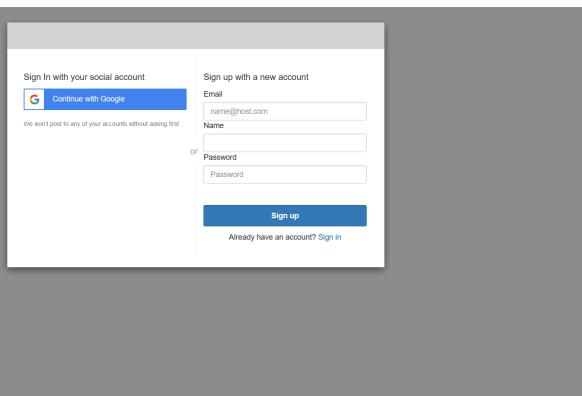


Fig 6.8.a. AWS Cognito Signup Page

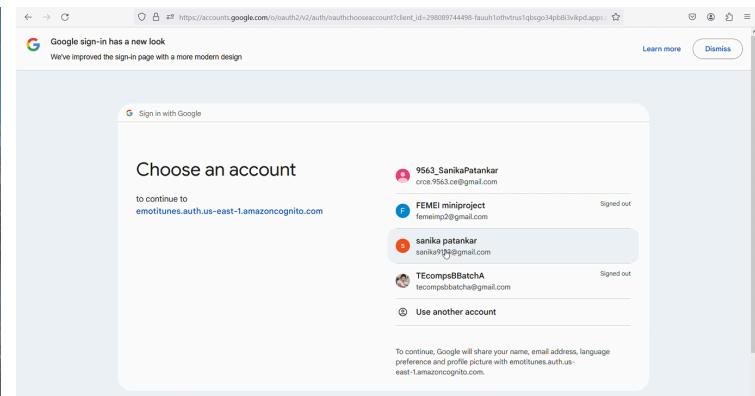


Fig 6.8.b. Single-Sign on Page

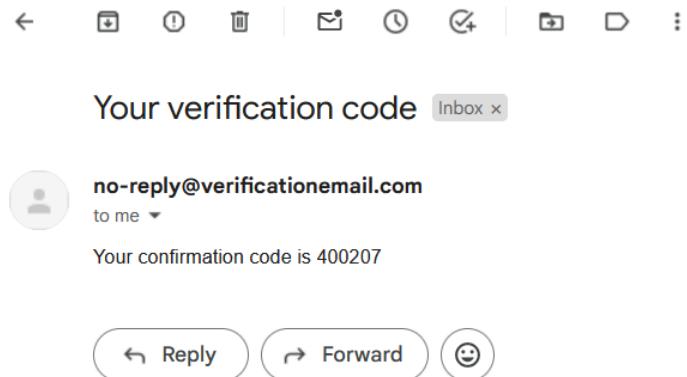


Fig 6.9. AWS Cognito Verification Code

S3 BUCKET

Following the configuration of our Django website to interface with the S3 bucket, administrators were equipped with a seamless process for uploading songs and album images. During this process, MP3 files were automatically organized within the "song_mp3" directory, with each file sorted into subdirectories based on mood. Similarly, album images were efficiently stored in the "song_images" directory within the S3 bucket.

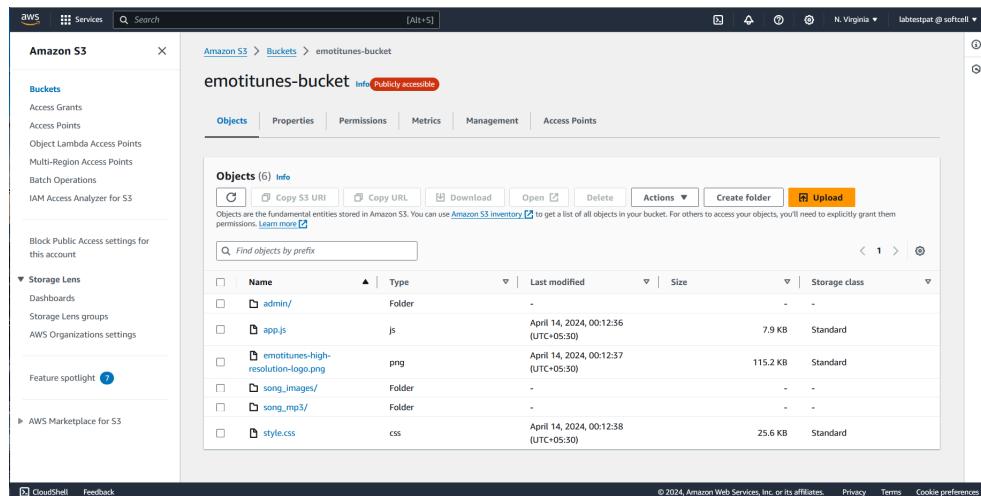


Fig 6.10. AWS S3 Bucket containing all static and media files

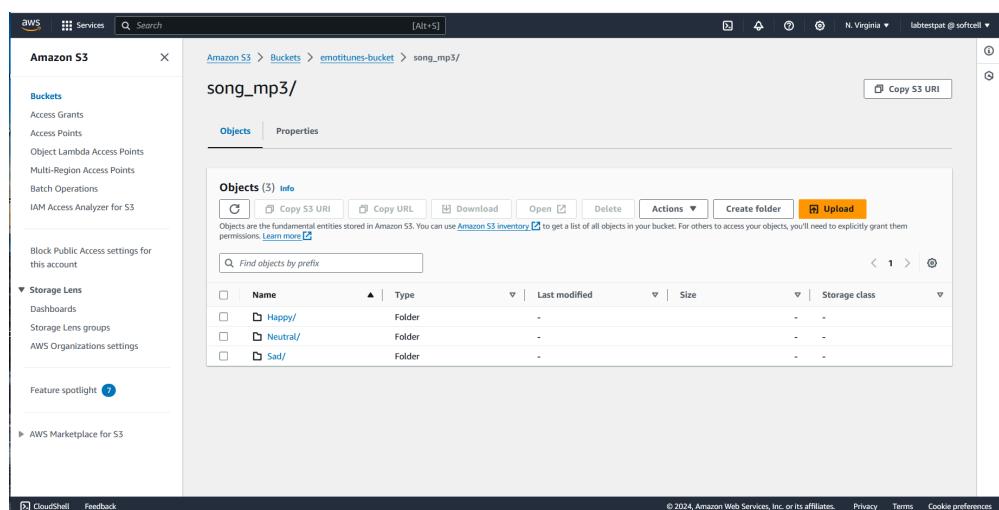


Fig 6.11. S3 Song Files

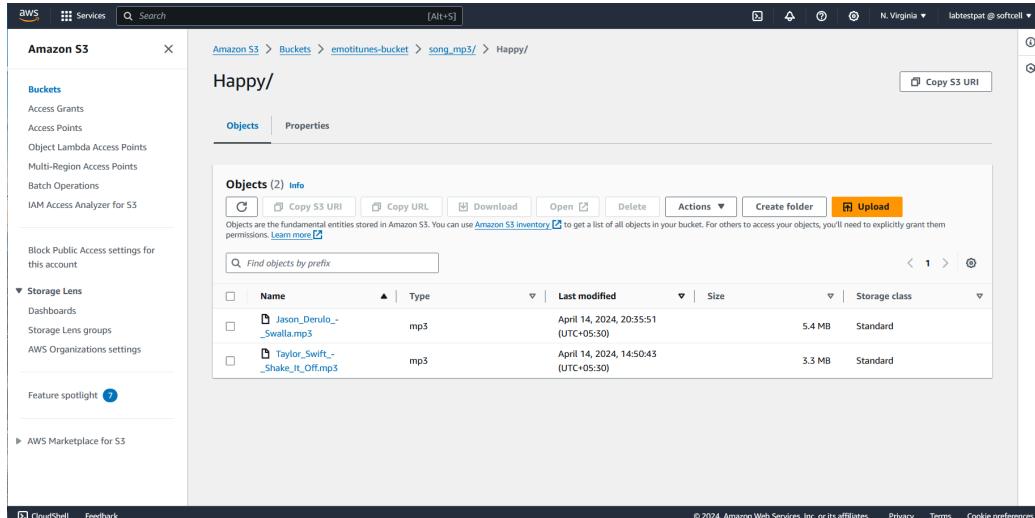


Fig 6.12. Songs in Happy Mood Folder

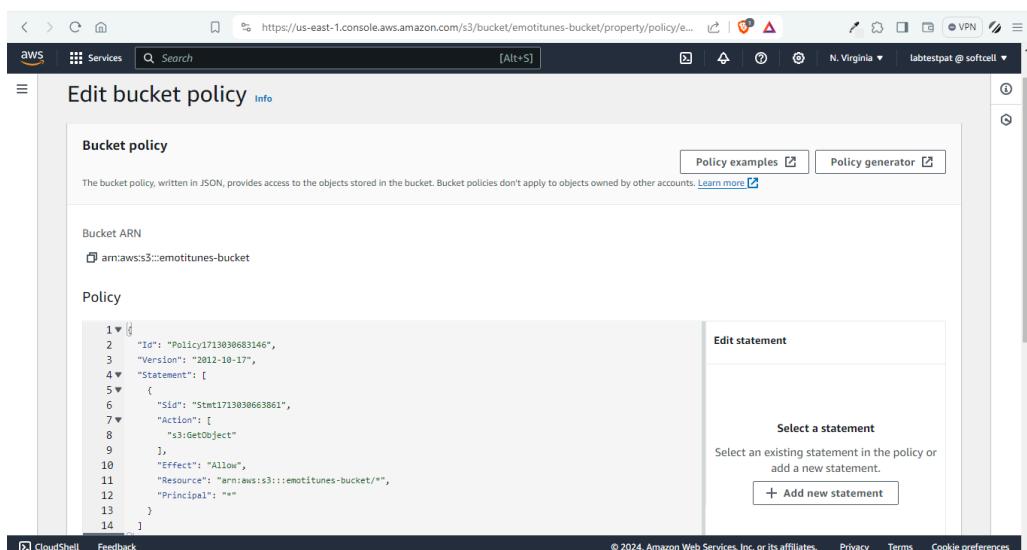


Fig 6.13. Bucket Policy Updated allowing GetObject

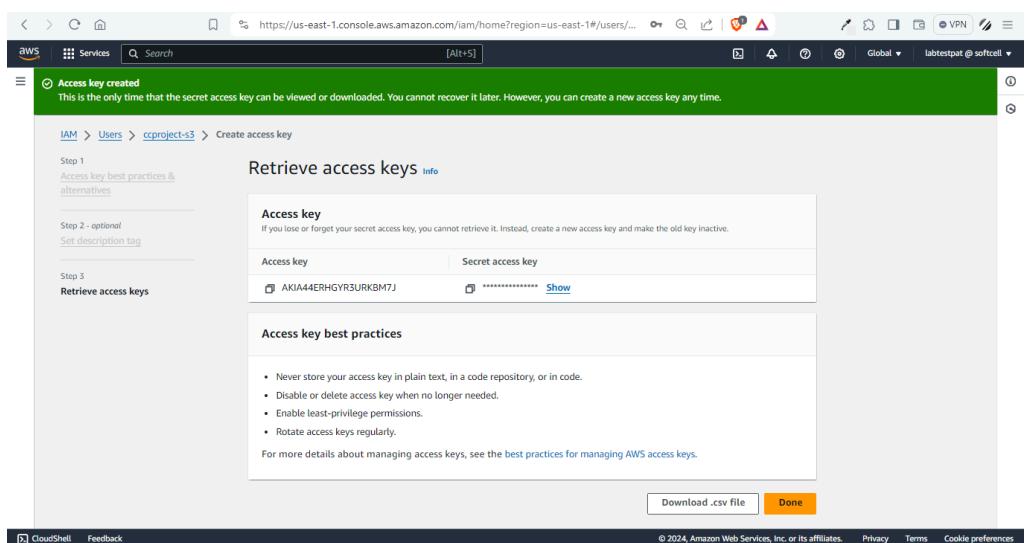


Fig 6.14. Created Access Key after creating user

RDS (DATABASE)

Facing issues with RDS database creation, we initiated VPC configuration, setting up subnets and attaching an Internet Gateway to facilitate outbound and inbound traffic. Creating a new route table, we associated subnets and configured a route for Internet-bound traffic. Additionally, DNS resolution and DNS hostname settings were enabled for seamless resource communication within the VPC. These actions resolved connectivity issues, allowing successful RDS database creation and operation.

The screenshot shows the AWS RDS console. On the left, the navigation pane is open with 'Databases' selected. The main area displays the 'Summary' tab for the database 'cc-project-db'. Key details shown include:

DB identifier	Status	Role	Engine	Recommendations
cc-project-db	Available	Instance	PostgreSQL	2 Informational
CPU	Class	Current activity	Region & AZ	
5.91%	db.t3.micro	0.00 sessions	us-east-1f	

Below the summary, there are tabs for 'Connectivity & security', 'Monitoring', 'Logs & events', 'Configuration', 'Maintenance & backups', 'Tags', and 'Recommendations'. The 'Connectivity & security' tab is active, showing details about the endpoint, networking, and security groups. The endpoint is cc-project-db.cbxhnvcphdr8.us-east-1.rds.amazonaws.com, port 5432. The VPC is cc-project-vpc (vpc-0280aa1f6ea59fcde). The subnet group is default-vpc-0280aa1f6ea59fcde. The security group is cc-project-sec-grp (sg-0a485913a28211d1c), which is active. The certificate authority is Info.

Fig 6.16. AWS RDS

The screenshot shows the AWS RDS console. The left navigation pane is visible with 'Databases' selected. The main area displays the 'Security group rules' and 'Replication' sections for the database 'cc-project-db'.

Security group rules (5):

Security group	Type	Rule
cc-project-sec-grp (sg-0a485913a28211d1c)	CIDR/IP - Inbound	105.226.87.86/52
cc-project-sec-grp (sg-0a485913a28211d1c)	CIDR/IP - Inbound	49.36.100.207/32
cc-project-sec-grp (sg-0a485913a28211d1c)	CIDR/IP - Inbound	110.224.127.225/32
cc-project-sec-grp (sg-0a485913a28211d1c)	CIDR/IP - Inbound	49.32.241.102/32
cc-project-sec-grp (sg-0a485913a28211d1c)	CIDR/IP - Outbound	0.0.0.0/0

Replication (1):

DB identifier	Role	Region & AZ	Replication source	Replication state	Lag
cc-project-db	Instance	us-east-1f	-	-	-

Manage IAM roles:

Add IAM roles to this instance: Choose an IAM role to add

Feature: Choose a feature to add

Add role

Fig 6.16. AWS RDS Security Groups

SECURITY GROUPS(VPC)

The screenshot shows the AWS VPC Security Groups details page. The security group name is "sg-0a485913a28211d1c - cc-project-sec-grp". Key details include:

- Security group name:** cc-project-sec-grp
- Security group ID:** sg-0a485913a28211d1c
- Description:** Created by RDS management console
- VPC ID:** vpc-0280aa16ea59fcde
- Owner:** 885067822627
- Inbound rules count:** 5 Permission entries
- Outbound rules count:** 1 Permission entry

The Inbound rules section lists five PostgreSQL TCP rules on port 5432.

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-0d946cbc69a9326...	IPv4	PostgreSQL	TCP	5432
-	sgr-0259739d9ef16d6...	IPv4	PostgreSQL	TCP	5432
-	sgr-04192105433269...	IPv4	PostgreSQL	TCP	5432
-	sgr-0c5a1db26fc1baec	IPv4	PostgreSQL	TCP	5432
-	sgr-0e74d8b9290ee55...	IPv4	PostgreSQL	TCP	5432

Fig 6.17. AWS VPC Security Groups

This screenshot is identical to Fig 6.17, but the "Outbound rules" tab is selected. It shows one outbound rule allowing all traffic on port All from the security group to All destinations.

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-0dbdd1028f1cdcb25	IPv4	All traffic	All	All

Fig 6.18. AWS VPC Security Groups (Inbound rules)

This screenshot is identical to Fig 6.17, but the "Outbound rules" tab is selected. It shows one outbound rule allowing all traffic on port All from the security group to All destinations.

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-0dbdd1028f1cdcb25	IPv4	All traffic	All	All

Fig 6.19. AWS VPC Security Groups (Outbound rules)

EC2 INSTANCES

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like EC2 Dashboard, EC2 Global View, Events, Console-to-Code, Instances, Images, Elastic Block Store, Network & Security, and more. The main area displays a table of instances. One instance is selected, showing its details in a modal window.

Instances (1/1) Info							
<input type="text"/> Find Instance by attribute or tag (case-sensitive)		Clear filters		All states			
Instance ID	Name	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
i-03a85a29432c1e073	cc_emotlunes	Running	t2.micro	Initializing	View alarms	us-east-1d	ec2-3-88-24-24.compute...

Instance: i-03a85a29432c1e073 (cc_emotlunes)

Details | Status and alarms [New](#) | Monitoring | Security | Networking | Storage | Tags

Instance summary [Info](#)

Updated 12 minutes ago

Instance ID i-03a85a29432c1e073 (9539_CC_Exp3)	Public IPv4 address 3.88.24.24 [open address]	Private IPv4 addresses 172.31.13.225
IPv6 address -	Instance state Running	Public IP DNS ec2-3-88-24-24.compute-1.amazonaws.com [open address]
Hostname type IP name: ip-172-31-13-225.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-13-225.ec2.internal	Elastic IP addresses -
Answer private resource DNS name -	Instance type t2.micro	AWS Compute Optimizer finding No recommendations available for this instance.
Auto-assigned IP address 5.88.24.24 [Public IP]	VPC ID vpc-bf52b6db	Auto Scaling Group name -
IAM Role -	Subnet ID subnet-03fb4614cdac215 (my-subnet-01)	Monitoring disabled
IMDSv2 Required		Termination protection Disabled

Networking

Storage

Tags

Fig 6.20. AWS EC2 INSTANCE

The screenshot shows the AWS EC2 Instances page. The navigation sidebar is identical to Fig 6.20. The main area displays a table of instances. One instance is selected, showing its details in a modal window.

Instances (1/1) Info							
<input type="text"/> Find Instance by attribute or tag (case-sensitive)		Clear filters		All states			
Instance ID	Name	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
i-03a85a29432c1e073	cc_emotlunes	Running	t2.micro	Initializing	View alarms	us-east-1d	ec2-3-88-24-24.compute...

Instance summary for i-03a85a29432c1e073 (cc_emotlunes)

Updated 12 minutes ago

Instance ID i-03a85a29432c1e073 (9539_CC_Exp3)	Public IPv4 address 3.88.24.24 [open address]	Private IPv4 addresses 172.31.13.225
IPv6 address -	Instance state Running	Public IP DNS ec2-3-88-24-24.compute-1.amazonaws.com [open address]
Hostname type IP name: ip-172-31-13-225.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-13-225.ec2.internal	Elastic IP addresses -
Answer private resource DNS name -	Instance type t2.micro	AWS Compute Optimizer finding No recommendations available for this instance.
Auto-assigned IP address 5.88.24.24 [Public IP]	VPC ID vpc-bf52b6db	Auto Scaling Group name -
IAM Role -	Subnet ID subnet-03fb4614cdac215 (my-subnet-01)	Monitoring disabled
IMDSv2 Required		Termination protection Disabled

Details | Status and alarms [New](#) | Monitoring | Security | Networking | Storage | Tags

Instance details [Info](#)

Platform
[Amazon Linux \(Inferred\)](#)

Platform details
[Linux/UNIX](#)

Stop protection
Disabled

AMI ID
[ami-05ffba213df8bc089](#)

AMI name
[al2023-ami-2023.4.20240401.1-kernel-6.1-x86_64](#)

Launch time
[Thu Apr 11 2024 17:04:36 GMT+0530 \(India Standard Time\) \(54 minutes\)](#)

AMI location
[amazon/al2023-ami-2023.4.20240401.1-kernel-6.1-x86_64](#)

Fig 6.21. EC2 INSTANCE

Chapter 7

RESULTS

The images below depict the final website structure of the application. The photographs also offer distinct mood predictions based on their facial expressions, as well as music recommendations for that specific mood.

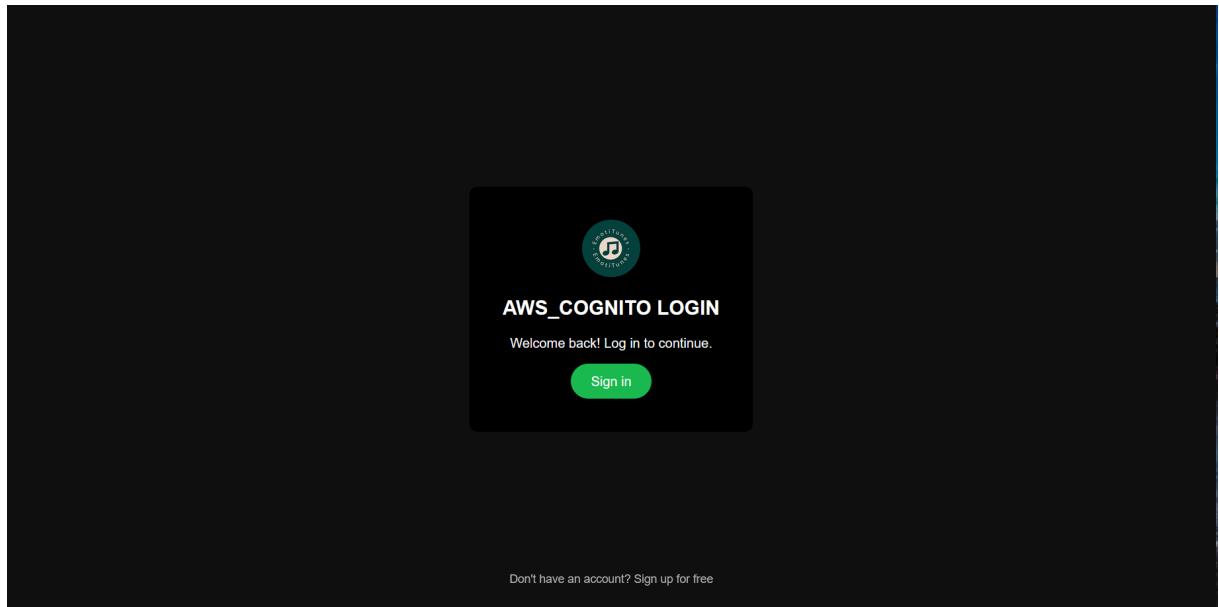


Fig 7.1. Login Page

A screenshot of a mobile application interface. On the left, there's a camera viewfinder showing a person's face with a large yellow smiley face overlay. Below it, the text "Songs for Mood: Happy" is displayed. To the right, there's a "Newly Added Songs" section with four cards: "Shake It Off" by Taylor Swift, "Swalla" by Jason Derulo, "Champagne Problems" by Taylor Swift, and "Party Rock Anthem" by LMFAO. Below this is a "Songs According To Your Mood" section with the same four songs. At the bottom, a playback control bar shows a song by Taylor Swift titled "Shake It Off" at 0:10, with a progress bar ending at 3:35.

Fig 7.2. Happy Mood

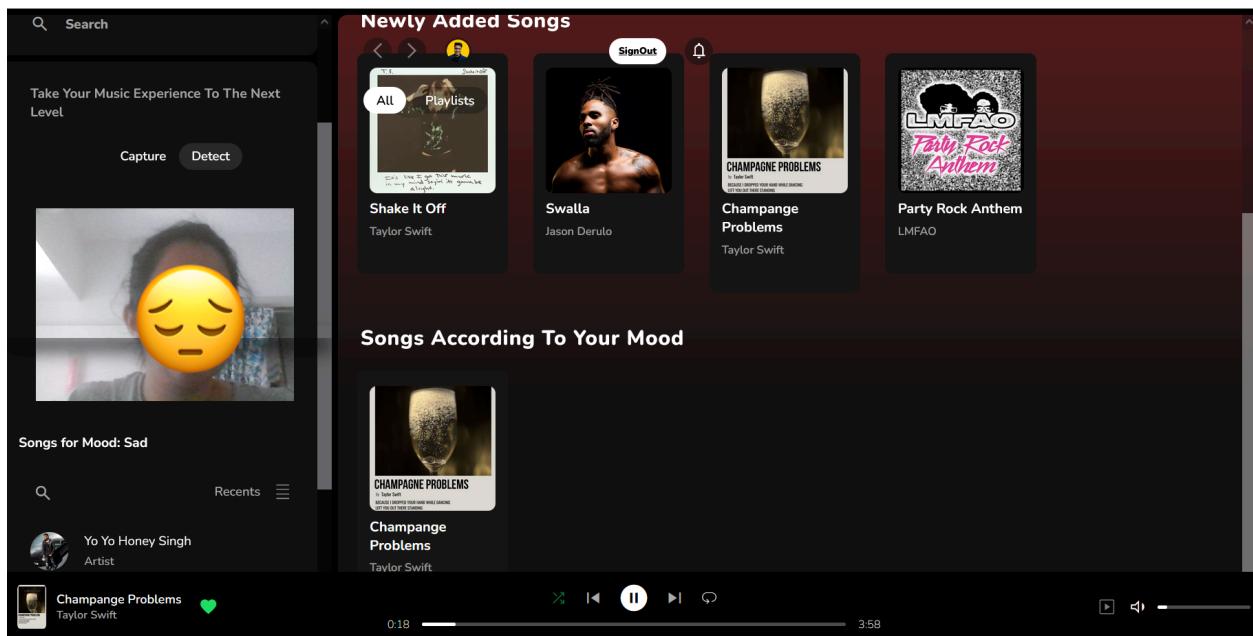


Fig 7.3. Sad Mood

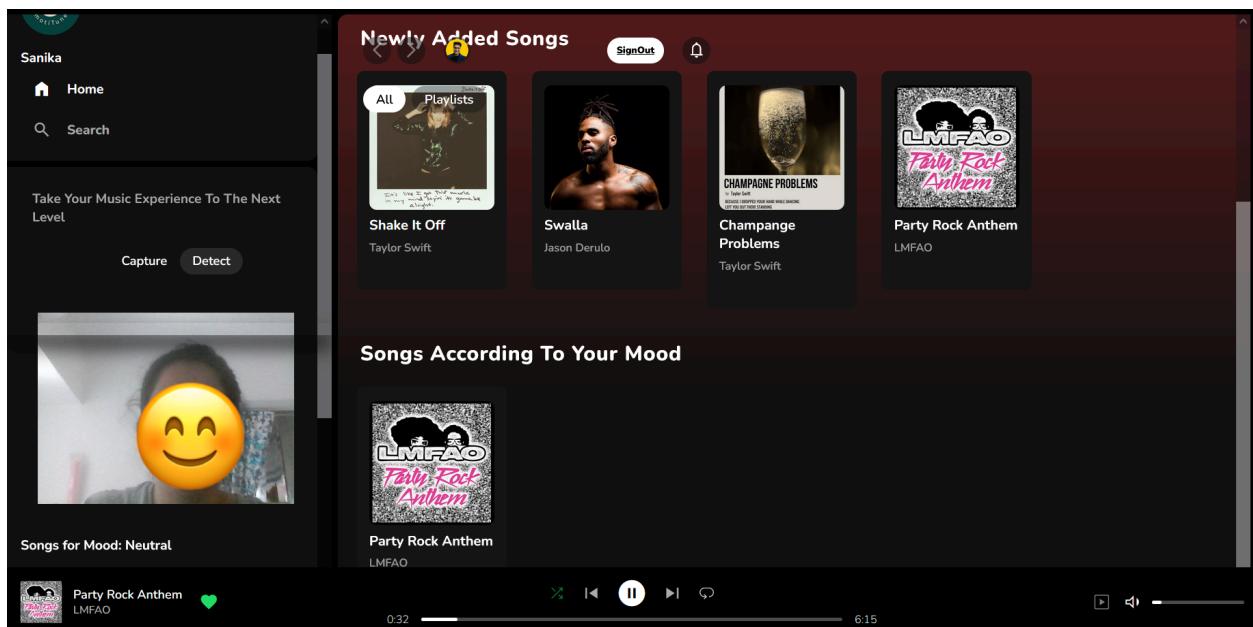


Fig 7.4. Neutral Mood

Chapter 7

CONCLUSION

By establishing a cloud-based architecture, we provide scalability, availability, and effective resource use for the system's continuous operation. The Django-based backend framework supports data processing and communication between the user interface and machine learning models. TensorFlow enables the system to create and deploy real-time facial expression analysis models, which result in accurate predictions of user moods. Furthermore, OpenCV integration enables accurate and efficient face image processing, as well as the extraction of crucial information from facial emotions. Finally, by adapting music suggestions to users' emotional states, the system customises the user experience and offers information that is relevant to their present mood.

In short, this research presents a complete and resilient system that uses cutting-edge technology to generate individualised and emotionally-aware music.