Dr. AMBEDKAR INSTITUTE OF TECHNOLOGY

Near Jnana Bharathi Campus, Bengaluru-560 056.

(An Autonomous Institution, Aided by Government of Karnataka)



Project Report on

"EPHEMORA"

Submitted by

Sameera P Nithin Sivakumar

(1DA21CS097) (1DA21CS095)

Under the Guidance Of

Dr. Asha

Assoc Prof., Dept of CSE

Dr. AIT

Department of Computer Science & Engineering

2023-24

Dr. AMBEDKAR INSTITUTE OF TECHNOLOGY

Near Jnana Bharathi Campus, Bengaluru-560 056.

(An Autonomous Institution, Aided by Government of Karnataka)



This is to certify that the project entitled "EPHEMORA" submitted in the partial fulfilment of the requirement of the 5th semester DBMS laboratory curriculum during the year 2021-2022 is a result of bonafide work carried out by

Sameera P Nithin Sivakumar

(1DA21CS097) (1DA21CS095)

Signature of the guide:

**Dr . Asha**

**Assoc.Prof., Dept of**

**CSE Dr. AIT**

1.Internal Examiner

2.External Examiner\_\_\_\_

**ACKNOWLEDGEMENT**

The satisfaction that accompanies to this project would be incomplete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made our efforts go in vain.

We consider ourselves privileged to express our gratitude and respect towards all those who guided us through the project, “EPHEMORA”.

We would like to express our gratitude to **Dr. C Nanjundswamy, Principal, Dr. A.I.T.,** for providing us the congenial environment to work in.

We would like to express our profuse gratitude to **Dr. Siddaraju, HOD, Dept. of Computer Science & Engineering, Dr. AIT,** for giving us the support, encouragement and providing us the required lab facilities that was necessary for the completion of this project.

As a token of gratitude, we would like to acknowledge our sincere gratefulness to our internal guide **Mr./Mrs. Asha, Assoc. Prof., Dept. of CSE, Dr.A.I.T.,** for her unlimited support and encouragement provided throughout the process.

We also express our gratitude and sincere thanks to all the teaching and non-teaching staff of **Computer Science & Engineering Department.**

Finally, yet importantly, we would like to express our heartfelt thanks to our beloved **Parents** for their blessings and our **Friends** for their help and wishes for the successful completion of this project report.

Sameera P

Nithin Sivakumar

**ABSTRACT**

"Ephemora" is a comprehensive Node.js application designed to enhance personal well-being and productivity through diverse modules catering to various aspects of daily life. The project integrates MongoDB with Mongoose for efficient data management, ensuring scalability and flexibility.

The application encompasses modules such as Administration, Breathe, Quote, User, and others, each tailored to address specific functionalities. The Administration module enables seamless management of admin accounts, facilitating creation, retrieval, update, and deletion operations.

The Breathe module focuses on mindfulness, allowing users to record and monitor their breathing durations. It provides insights into daily breathing patterns, fostering mental and emotional well-being.

The Quote module aims to inspire and motivate users by offering a curated collection of quotes. Users can contribute their own quotes, fostering a sense of community engagement.

The User module centralizes user management, allowing for the creation, retrieval, update, and deletion of user accounts. It empowers users to personalize their profiles and maintain their information securely.

Other modules such as Journal and Playlist offer avenues for self-expression and entertainment, enabling users to document their thoughts and curate playlists for relaxation or productivity.

Overall, "Ephemora" serves as a multifaceted platform designed to enrich users' lives, promoting self-care, creativity, and personal growth in a digital environment.

CONTENTS

|  |  |  |
| --- | --- | --- |
| Chapter No | Title | Page No |
| Chapter 1 | Introduction | 2 |
| Chapter 2 | Requirement specification | 4 |
| 2.1 | Frontend Technologies | 4 |
| 2.2 | Backend Technologies | 5 |
|  |  |  |
| Chapter 3 | Description | 7 |
| Chapter 4 | Coding | 9 |
| Chapter 5 | Snapshots | 18 |
|  |  | 22 |
|  | Conclusion | 26 |
|  | Bibliography | 29 |

Chapter 1

**INTRODUCTION**

In today's fast-paced world, where individuals are constantly juggling various responsibilities and facing numerous stressors, maintaining personal well-being and productivity can often become challenging. In response to this modern-day dilemma, the Ephemora project emerges as a beacon of solace and support, offering a holistic solution to address these concerns.

Ephemora is a dynamic Node.js application meticulously crafted to empower users in their journey towards better mental, emotional, and physical health. Rooted in the belief that small, intentional actions can lead to significant positive changes, Ephemora provides a suite of modules designed to cater to diverse aspects of daily life.

At its core, Ephemora is built upon the principles of simplicity, accessibility, and effectiveness. Whether it's managing administrative tasks, cultivating mindfulness through breathing exercises, seeking inspiration through quotes, or fostering community engagement, Ephemora offers a comprehensive platform to fulfill these needs seamlessly.

Through this introduction, we embark on a journey to explore the intricacies of Ephemora, delving into its various modules, functionalities, and the underlying philosophy driving its development. By embracing Ephemora, users can embark on a transformative experience, navigating life's challenges with resilience, creativity, and a renewed sense of purpose. Join us as we embark on this enriching journey towards holistic well-being and personal growth with Ephemora.

Chapter 2

**REQUIREMENT SPECIFICATION**

Frontend Technologies:

 **React:** EPHEMORA's frontend is developed using React, providing a foundation for building interactive and responsive user interfaces.

 **Tailwind CSS:** Tailwind CSS is employed for styling, contributing to a modern and visually appealing design.

 **Seamless Animations:** The platform incorporates seamless animations to enhance user engagement and overall usability.

Backend Technologies:

 **Node.js:** The backend of EPHEMORA is powered by Node.js, offering a scalable and efficient runtime environment for server-side development.

 **Express.js:** EPHEMORA utilizes Express.js as a robust web application framework to streamline backend development.

 **MongoDB:** Data storage is managed using MongoDB, a NoSQL database, providing flexibility and

scalability for handling diverse data structures.

 **Mongoose:** Mongoose, an ODM (Object Data Modelling) library for MongoDB and Node.js, facilitates interaction with the database.

Chapter 3

**DESCRIPTION**

1. Breathe Section

 **Description:** This section focuses on promoting relaxation and mindfulness through breathing exercises.

 **Model:** Breathe

2. Social Section

 **Description:** Users can connect with each other in a social environment, fostering community engagement.

 **Models:** Admin, Answer

3. Journal Section

 **Description:** Users can express their thoughts and emotions by maintaining a journal.

 **Model:** Journal3

4. I'm Bored Section

 **Description:** Users can engage in activities when feeling bored, including answering fun questions and exploring curated playlists.

 **Models:** Answer, Playlist, Question, Forums, Advices.

 **Interactive Questions:** Users can answer interactive and entertaining questions to beat boredom.

 **Curated Playlists:** Users have access to curated playlists to enhance their mood and

entertainment.

5. Ancient Bharath Design Inspiration

 **Description:** The website design is inspired by the rich cultural heritage of ancient Bharath (India).

 **Implementation:** Elements of Bharath's history, art, and symbolism are incorporated into the website's visual design.

6. Quotes from Ramayana and Mahabharata

 **Description:** Quotes displayed on the platform are sourced from ancient Bharath's epics, Ramayana and Mahabharata.

 **Implementation:** Users can submit their quotes, which are scrutinized by admins and

approved. Approved quotes are displayed for 24 hours, with a new quote generated daily.

CRUD Operations:

1. Breathe Section

 **Create (C):** Users can record their breathing exercises, including the date and duration.

 **Read (R):** Users can view their breathing history and track progress over time.

 **Update (U):** Duration of breathing exercises can be updated.

 **Delete (D):** Previous breathing records can be deleted.

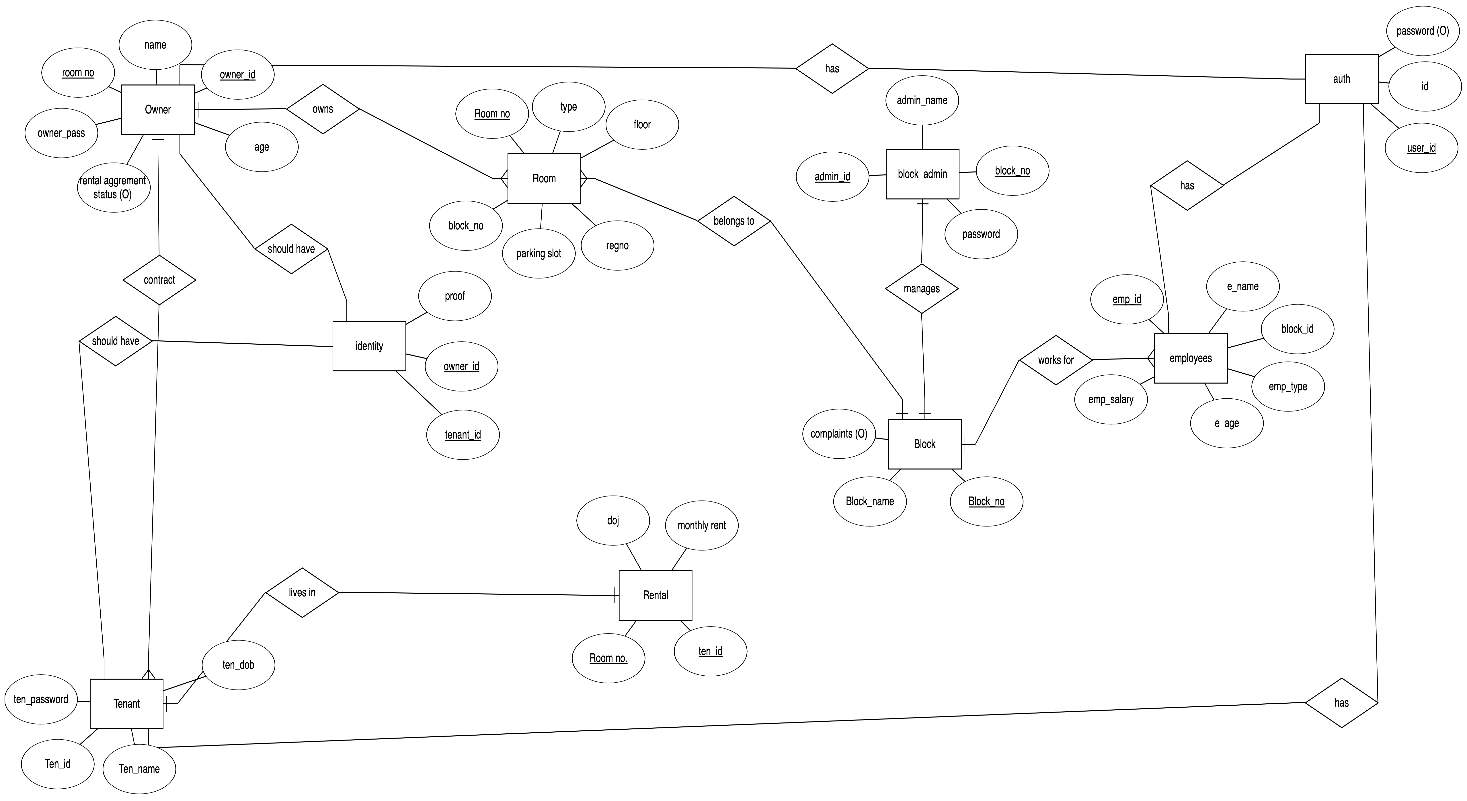
 **History:** The history of your exercise is also stored

1. I'm Bored Section

 **Read (R):** Users can view and answer questions, listen to music from playlists, engage in discussions in forum section and seek for knowledgeable advices .

 **Update (U):** Users can update their answers or playlist details.

 **Delete (D):** Answers or playlists can be deleted.



E-R Diagram used for “EPHEMORA”

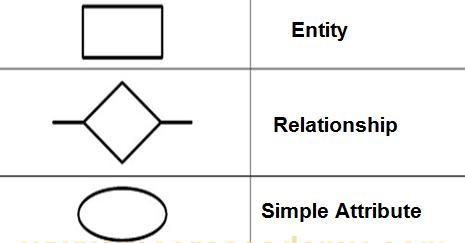
E-R (Entity-Relationship) Diagram is used to represents the relationship between entities in a table. ER diagrams represent the logical structure of databases. ER Diagram represent relationship between two database tables.

E-R diagram means Entity Relationship diagram. Entity is an object of system, generally, we refer entity as database table, the e-r diagram represents the relationship between each table of database. E-R diagram represent entity with attributes, attributes are a property of entity. If we assume entity is a database table then all the columns of table are treated as attributes.

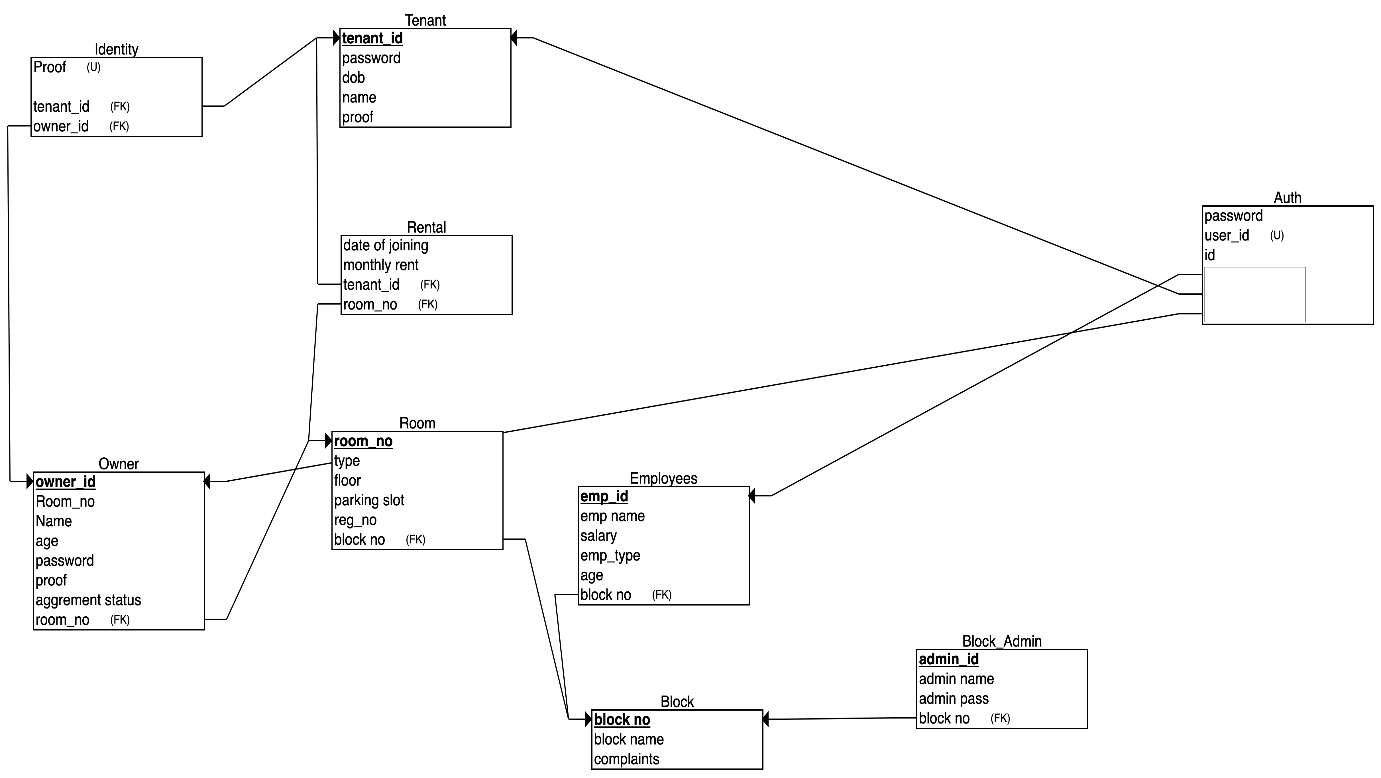
**Entity :** Entities are represented by **rectangle**. All table of database are treated as entity.

**Attributes :** Attributes are represented by **ellipses**. Attributes are properties of

entities.



Schema Diagram:



Schema Diagram used for “EPHEMORA.”

Chapter 4

**CODING**

1. **ADMIN MODULE:**

Admin Controller Operations

Create Admin Account

Route: POST /api/v1/admin

Body:

json

{

"name": "Admin",

"email": "admin@demo.com",

PLAYLIST SECTION IN I'm BORED PAGE11

"password": "admin"

}

**Description:**

This operation creates a new admin account with the provided name, email, and

password.

Get Admin Accounts

Route: GET /api/v1/admin

Body:

json

{}

**Description:**

Retrieves a list of all admin accounts available.

Update Admin Account

Route: PUT /api/v1/admin/:id

Body:

json

{

"name": "Admin",

"email": "admin@demo.com",

"password": "admin"

}

**Description:**

Updates the details of the admin account with the provided ID.

Delete Admin Account

Route: DELETE /api/v1/admin/:id

Body:12

json

{}

**Description:**

Deletes the admin account with the specified ID.

Examples

**1. Create Admin Account:**

json

POST /api/v1/admin

{

"name": "Admin",

"email": "admin@demo.com",

"password": "admin"

}

**2. Get Admin Accounts:**

json

GET /api/v1/admin

{}

**3. Update Admin Account:**

json

PUT /api/v1/admin/:id

{

"name": "Admin",

"email": "admin@demo.com",

"password": "admin"

}

**4. Delete Admin Account:**

json

DELETE /api/v1/admin/:id

{}

These operations enable the management of admin accounts, ensuring functionalities for

creation, retrieval, update, and deletion. The provided examples illustrate the usage of each

operation.13

**2. BREATHE MODULE:**

Breathe Controller Operations

Record Breathing Duration

Route: POST /api/v1/breathe

Body:

json

{

"userId": "65b6015405856a15301a661b",

"duration": "42"

}

**Description:**

Records the breathing duration for a user. If no record exists for the current day, a new record is created. Otherwise, the existing record is updated.

Get Current Day's Breathing Duration

Route: GET /api/v1/breathe/today/:userId

Body:

json

{}

**Description:**

Retrieves the breathing duration recorded for the current day for the specified user.

Examples

**1. Record Breathing Duration:**

json

POST /api/v1/breathe

{

"userId": "65b6015405856a15301a661b",

"duration": "42"14

}

**2. Get Current Day's Breathing Duration:**

json

GET /api/v1/breathe/today/65b6015405856a15301a661b

{}

These operations facilitate the recording and retrieval of breathing duration data. The provided

examples demonstrate the usage of each operation.

**3. QUOTE MODULE:**

Quote Module Operations

Create Quote

Route: POST /api/v1/quote

Body:

json

{

"quote": "This is an inspiring quote.",

"author": "John Doe",

"submittedBy": "user123"

}

**Description:**

Creates a new quote with the provided content, author, and submission information.

Get Active Quote

Route: GET /api/v1/quote/active

Body:

json

{}

**Description:**

Retrieves the currently active quote. If no active quote is found, it falls back to

fetching a random quote from an external API.

Examples:

**1. Create Quote:**

json

POST /api/v1/quote

{

"quote": "This is an inspiring quote.",

"author": "John Doe",

"submittedBy": "user123"

}

**2. Get Active Quote:**

json

GET /api/v1/quote/active

{}

These operations handle the creation of new quotes and fetching the currently active

quote. The provided examples demonstrate the usage of each operation.

**3. USER MODULE:**

User Module Operations

Create User

Route: POST /api/v1/user

Body:

json

{

"fullName": "John Doe",

"username": "johndoe123",

"email": "johndoe@example.com",

"password": "securepassword",

"phone": "1234567890",

"age": 25,16

"gender": "Male"

}

**Description:**

Creates a new user with the provided details, including full name, username, email,

password, phone, age, and gender.

Get Users

Route: GET /api/v1/user

Body:

json

{}

Description:

Retrieves all users from the system.

Update User

Route: PUT /api/v1/user/:id

Body:

json

{

"fullName": "Updated Name",

"username": "updatedUsername",

"email": "updatedemail@example.com",

"password": "updatedpassword",

"phone": "9876543210",

"age": 30,

"gender": "Female"

}

Description:

Updates the user details with the provided data for the user identified by the given ID.

Delete User17

Route: DELETE /api/v1/user/:id

Body:

json

{}

**Description:**

Deletes the user identified by the given ID.

Examples

**1. Create User:**

json

POST /api/v1/user

{

"fullName": "John Doe",

"username": "johndoe123",

"email": "johndoe@example.com",

"password": "securepassword",

"phone": "1234567890",

"age": 25,

"gender": "Male"

}

**2. Get Users:**

json

GET /api/v1/user

{}

**3. Update User:**

json

PUT /api/v1/user/123456789

{

"fullName": "Updated Name",

"username": "updatedUsername",

"email": "updatedemail@example.com",

"password": "updatedpassword",18

"phone": "9876543210",

"age": 30,

"gender": "Female"

}

**4. Delete User:**

json

DELETE /api/v1/user/123456789

{}

These operations handle the creation, retrieval, updating, and deletion of user

accounts. The provided examples demonstrate the usage of each operation.

MODELS[SCHEMAS]:

Here are the schemas that our project comprises of. These models provides information about

the columns of the table of each module those are Admin, User, Questions, Quotes, Answers,

Playlists sections.

Admin Model

javascript

import mongoose, { Schema } from "mongoose";

const adminSchema = new Schema(

{

name: {

type: String,

required: [true, "Please enter the name"],

},

email: {

type: String,

required: [true, "Please enter the email"],

unique: true,

},

password: {

type: String,

required: [true, "Please enter the password"],

},19

role: {

type: String,

default: "Admin",

},

},

{

timestamps: true,

}

);

export const Admin = mongoose.model("Admin", adminSchema);

Answer Model

javascript

import mongoose from "mongoose";

const answerSchema = new mongoose.Schema({

user: {

type: mongoose.Schema.Types.ObjectId,

ref: "User",

required: [true, "Field not found! Please enter the user ID"],

},

content: {

type: String,

required: [true, "Field not found! Please provide a valid answer"],

},

relatedQuestion: {

type: mongoose.Schema.ObjectId,

ref: "Question",

},

});

export const Answer = mongoose.model("Answer", answerSchema);

Breathe Model

javascript

import mongoose from "mongoose";

const breathingHistorySchema = new mongoose.Schema(20

{

user: {

type: mongoose.Types.ObjectId,

ref: "User",

required: [true, "Please enter the user ID"],

},

date: {

type: Date,

required: true,

},

duration: {

type: Number,

default: 0,

},

},

{ timestamps: true }

);

export const Breathe = mongoose.model("Breathe", breathingHistorySchema);

Certainly! Here are three entries for each of the remaining models:

Journal Model

javascript

import mongoose, { Schema } from "mongoose";

const detail = new Schema({

title: {

type: String,

},

content: {

type: String,

},

});

const journalSchema = new Schema(

{

user: {

type: Schema.Types.ObjectId,

ref: "User",21

required: [true, "Please enter the user ID"],

},

mood: {

type: String,

enum: ["Excited", "Happy", "Neutral"],

required: [true, "Please enter the mood"],

},

reason: {

type: String,

},

details: [detail],

images: [

{

type: String,

},

],

},

{ timestamps: true }

);

export const Journal = mongoose.model("Journal", journalSchema);

Playlist Model

javascript

import mongoose, { Schema } from "mongoose";

const playlistSchema = new Schema({

user: {

type: Schema.Types.ObjectId,

ref: "User",

required: [true, "Please enter the user ID"],

},

name: {

type: String,

required: [true, "Please enter the playlist"],

},

url: {

type: String,

required: [true, "Please enter a valid URL"],

},22

});

export const Playlist = mongoose.model("Playlist", playlistSchema);

Question Model

javascript

import mongoose, { Schema } from "mongoose";

const questionSchema = new Schema({

content: {

type: String,

required: [true, "Field not found! Please ask a question"],

},

postedBy: {

type: String,

required: [

true,

"Field not found! User ID (the person who asks the question) has to be provided",

],

},

});

export const Question = mongoose.model("Question", questionSchema);

Quote Model

javascript

import mongoose, { Schema } from "mongoose";

const quoteSchema = new Schema({

quote: {

type: String,

required: [true, "Field not found! Quote is required!"],

},

author: {

type: String,

required: [true, "Field not found! Author is required!"],

},

image: {

type: String,23

required: [true, "Field not found! Image is required!"],

},

active: {

type: Boolean,

default:

false,

},

submittedBy: {

type: String,

required: [true, "Field not found! Username is required!"],

},

});

export const Quote = mongoose.model("Quote", quoteSchema);

User Model

javascript

import mongoose, { Schema } from "mongoose";

import bcrypt from "bcryptjs";

import jwt from "jsonwebtoken";

const userSchema = new Schema(

{

fullName: {

type: String,

required: [true, "Field not found. Name is required!"],

trim: true,

index: true,

},

username: {

type: String,

required: [true, "Field not found. Username is required!"],

unique: true,

lowercase: true,

trim: true,

index: true,

},

email: {

type: String,24

required: [true, "Field not found. Email is required!"],

unique: true,

lowercase: true,

trim: true,

},

password: {

type: String,

required: [true, "Field not found. Password is required!"],

},

phone: {

type: Number,

required: [true, "Field not found. Phone is required!"],

},

age: {

type: Number,

required: [true, "Field not found. Age is required!"],

},

gender: {

type: String,

enum: ["Male", "Female", "Others"],

required: [true, "Please enter the gender"],

},

currentMood: {

type: String,

},

exists: {

type: Boolean,

default: true,

},

refreshToken: {

type: String,

},

},

{ timestamps: true }

);

userSchema.pre("save", async function (next) {

console.log("just before saving");

const rounds = 10;25

const hash = await bcrypt.hash(this.password, rounds);

this.password = hash;

next();

});

userSchema.methods.verifyPassword = async function (password) {

return await bcrypt.compare(password, this.password);

};

userSchema.methods.generateAccessToken = function () {

return jwt.sign(

{

\_id: this.\_id,

fullName: this.fullName,

username: this.username,

email: this.email,

},

process.env.ACCESS\_TOKEN\_SECRET,

{ expiresIn: process.env.ACCESS\_TOKEN\_EXPIRY }

);

};

userSchema.methods.generateRefreshToken = function () {

return jwt.sign(

{

\_id: this.\_id,

},

process.env.REFRESH\_TOKEN\_SECRET,

{ expiresIn: process.env.REFRESH\_TOKEN\_EXPIRY }

);

};

export const User = mongoose.model("User", userSchema);

Chapter 5

SNAPSHOTS

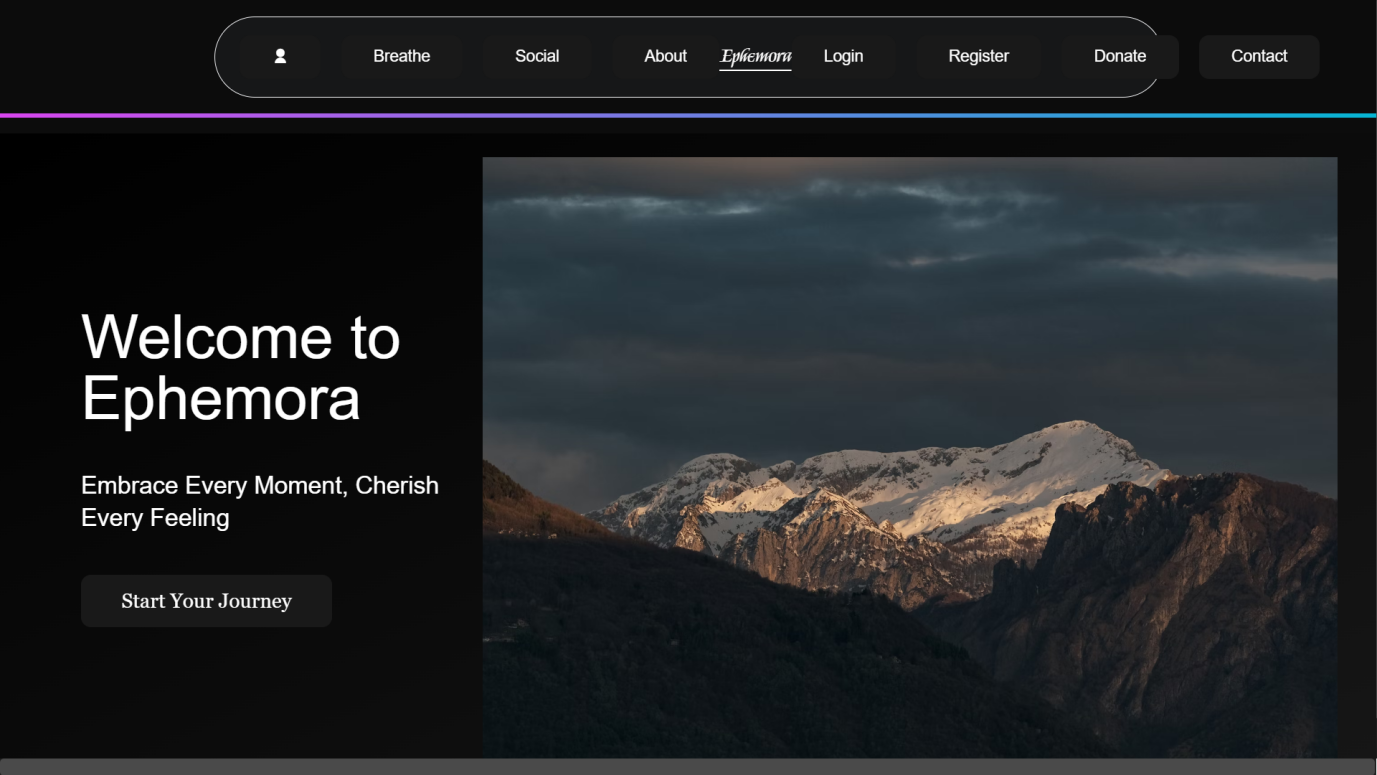
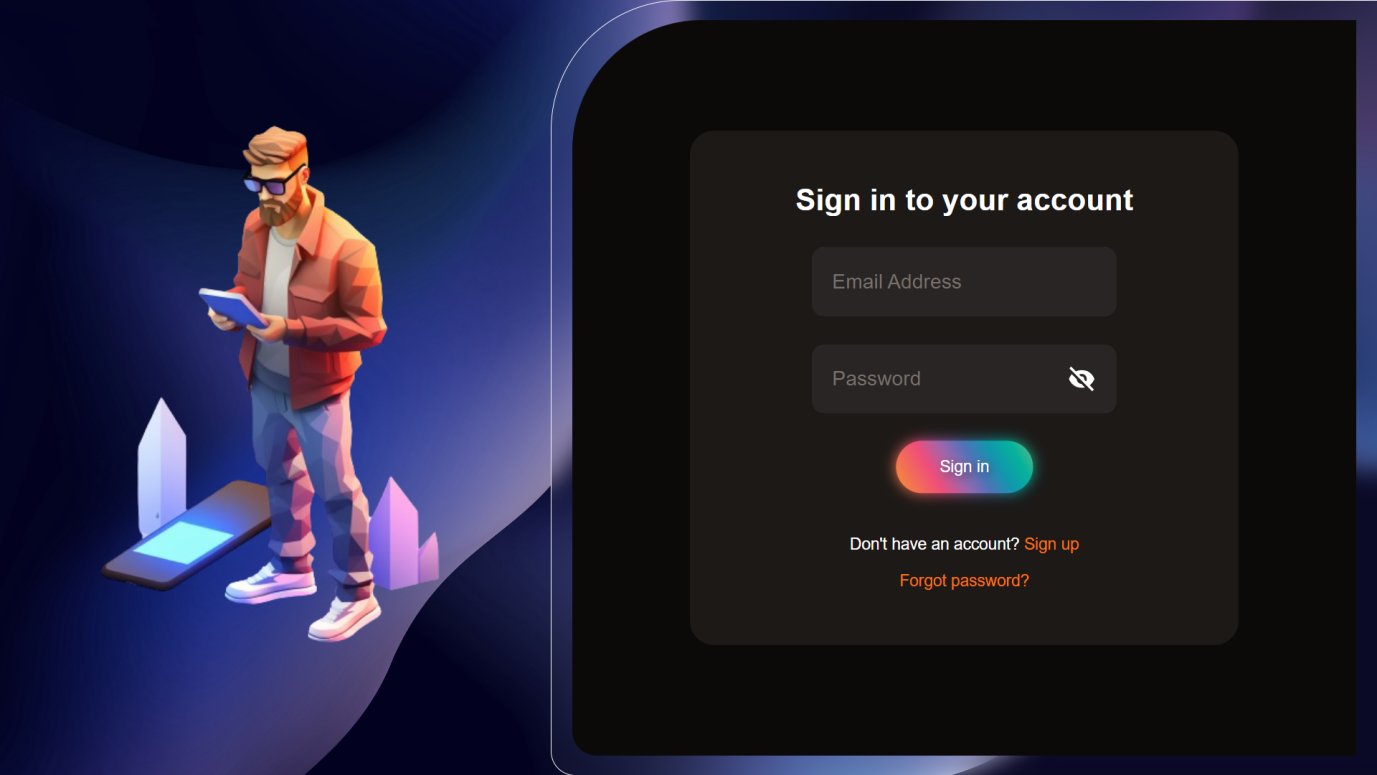


Figure 5.1: Home page of this website

Figure 5.2:Here you can Sign in

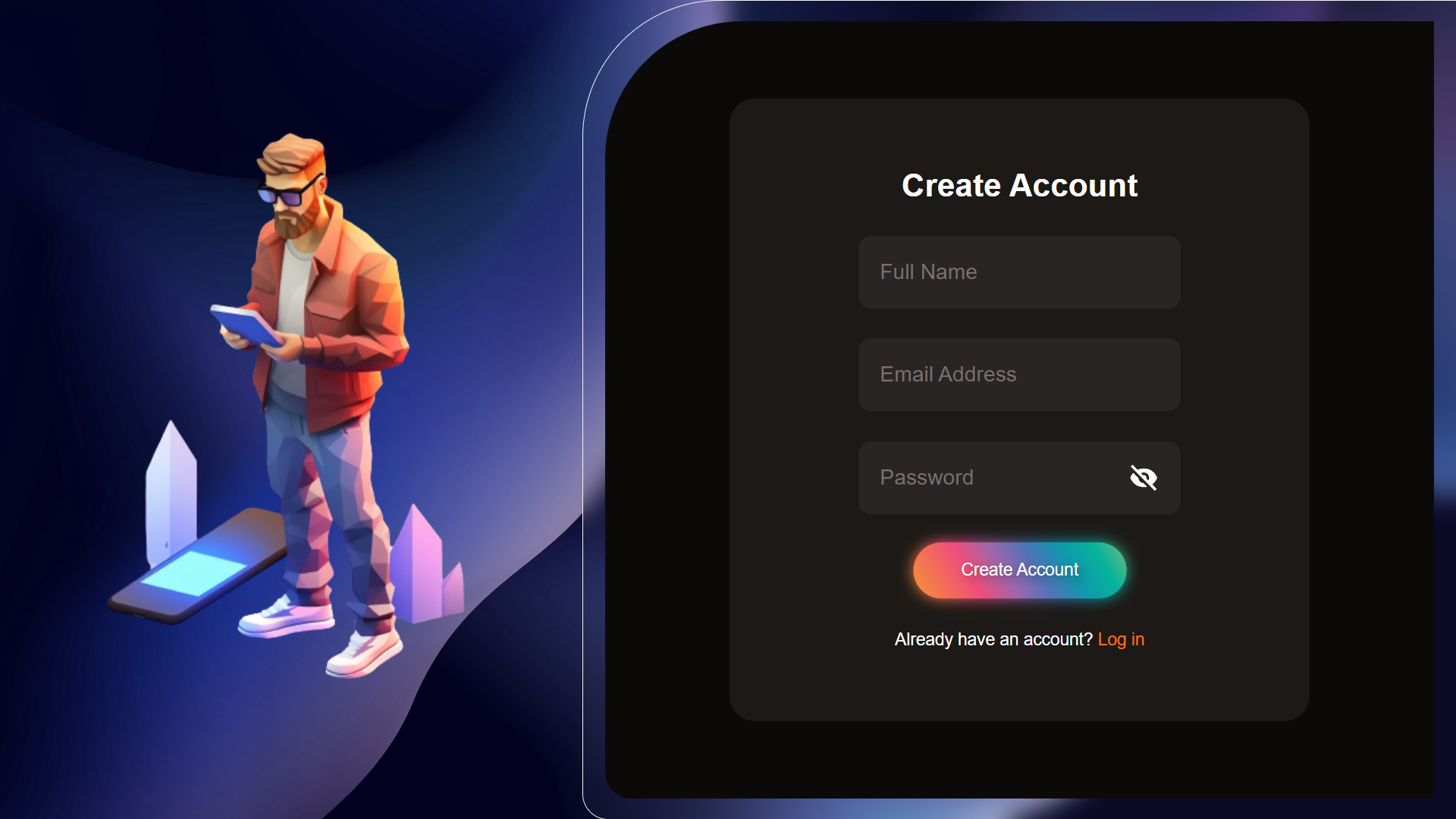


Figure 5.3:Here you can create account

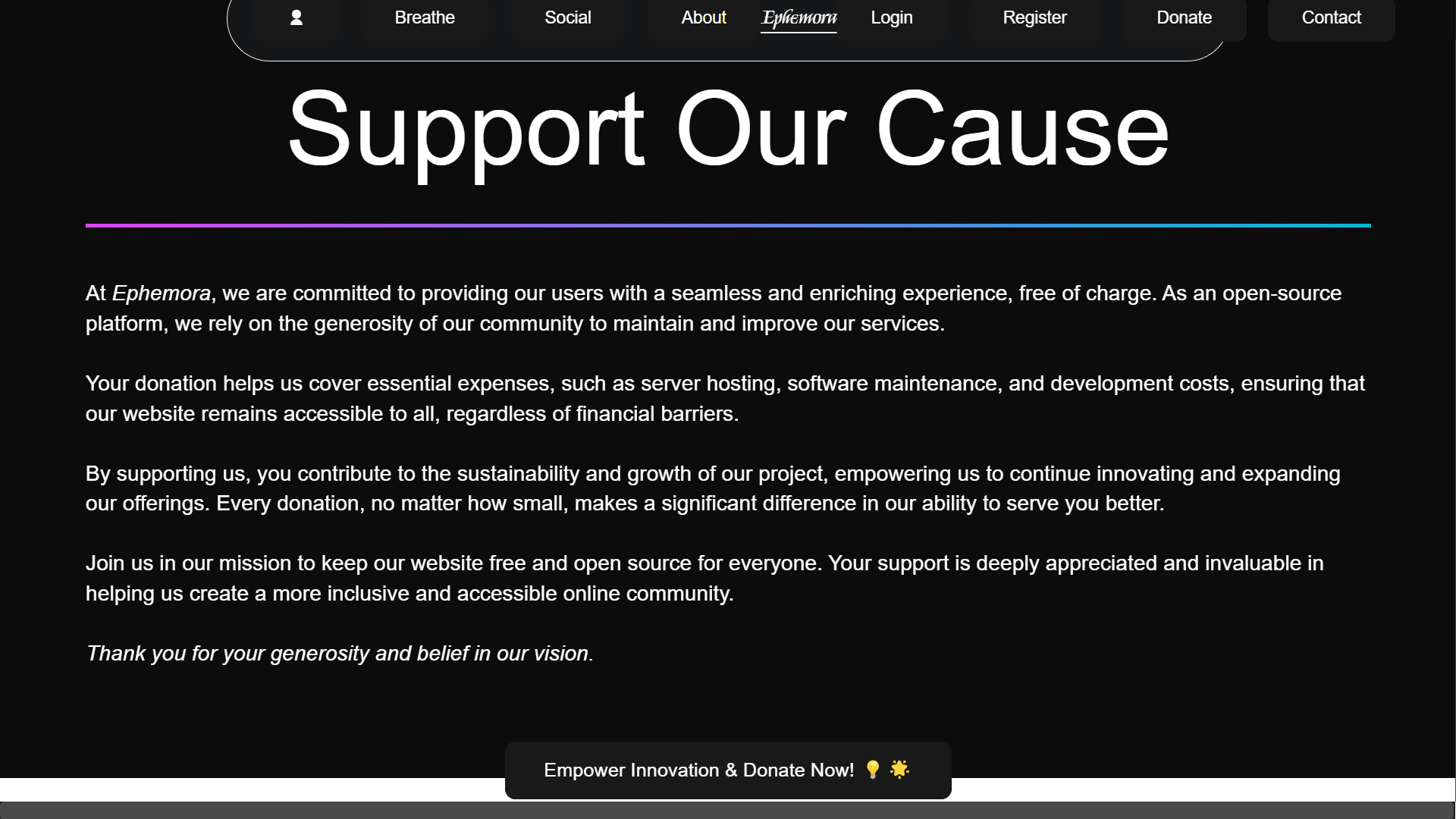


Figure 5.4:You can support us here

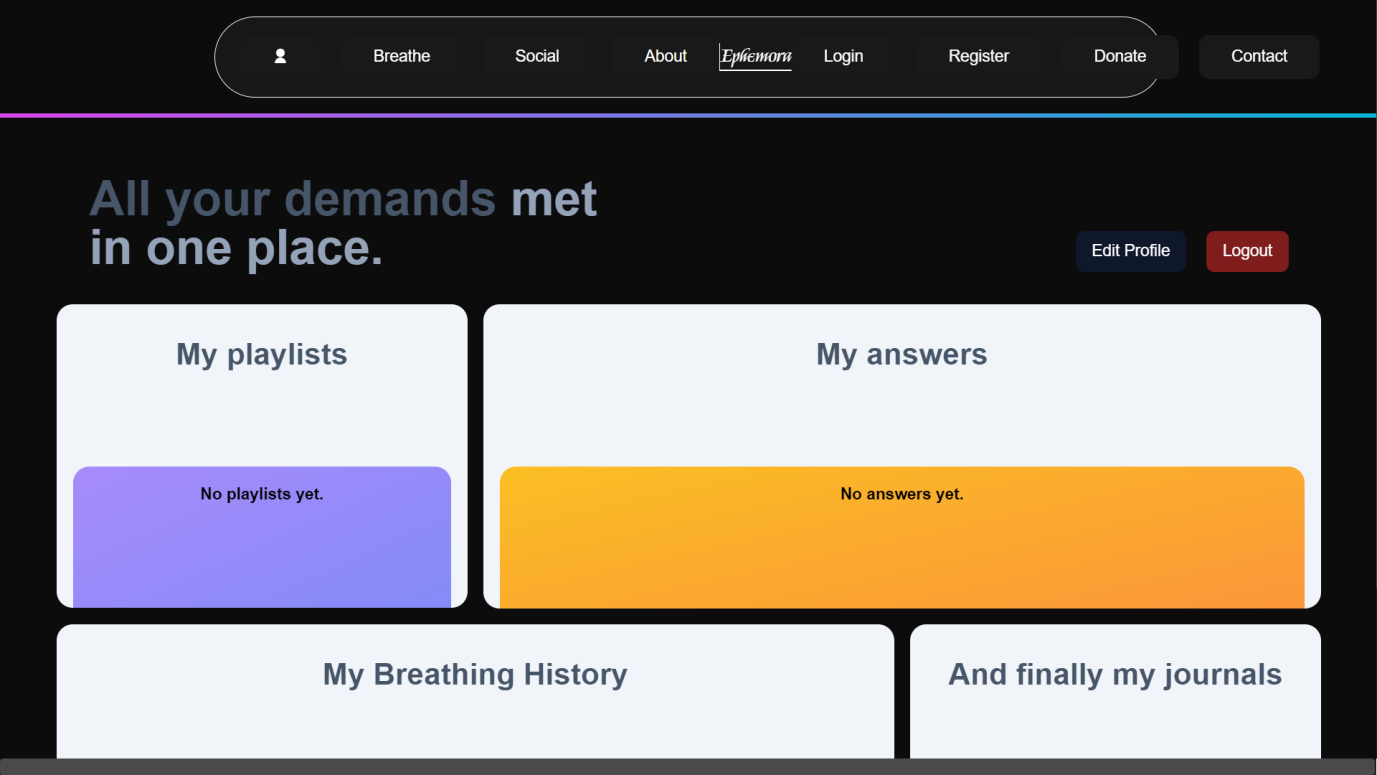


Figure 5.5:Playlist

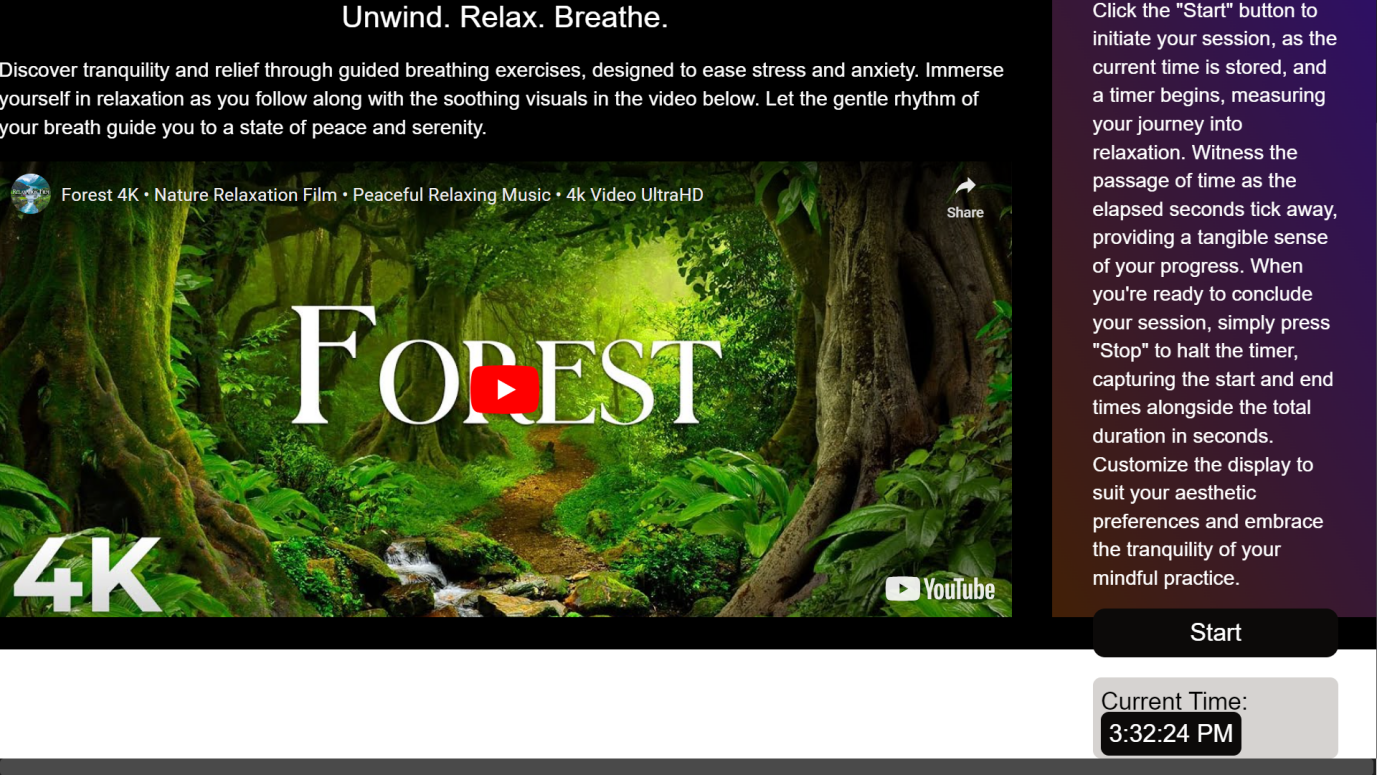


Figure 5.6:Advice Section in boared page

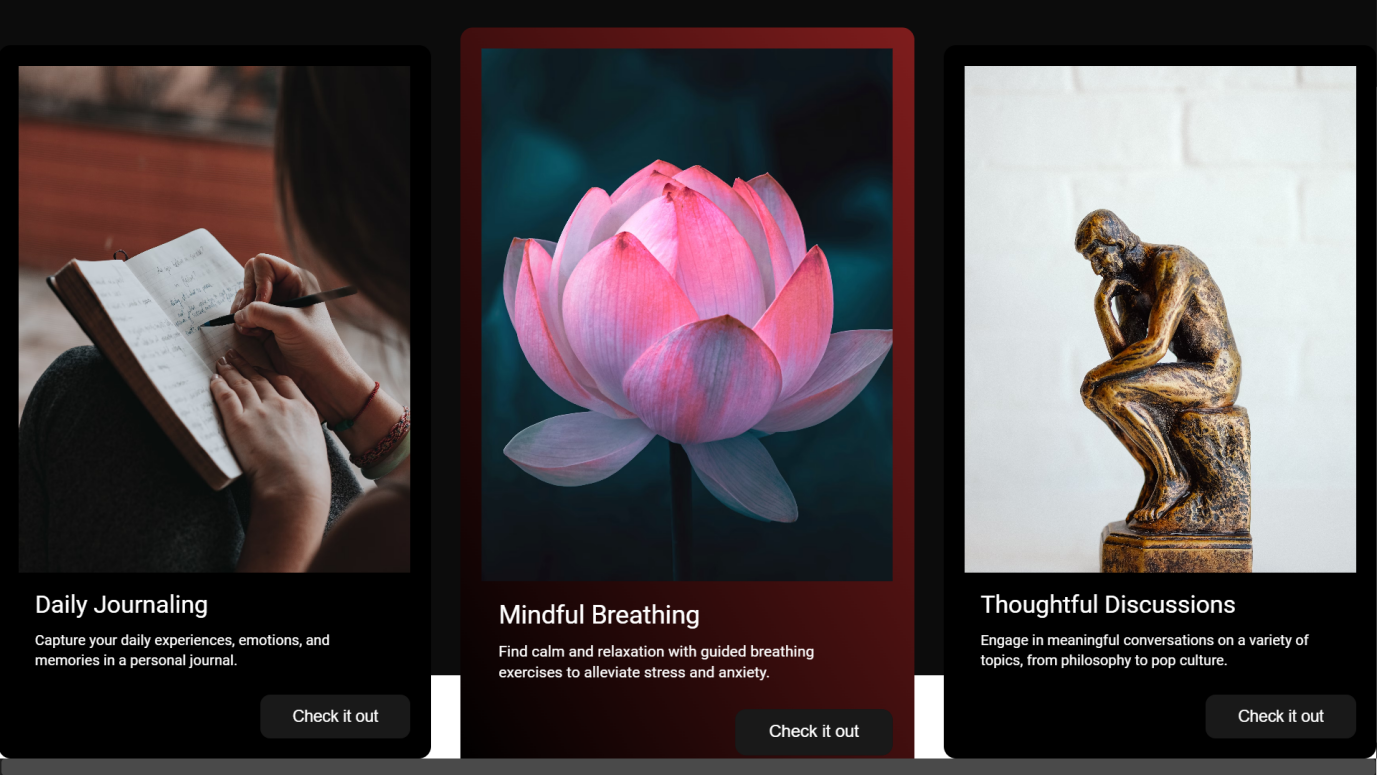


Figure 5.7: Queestion section in boared page

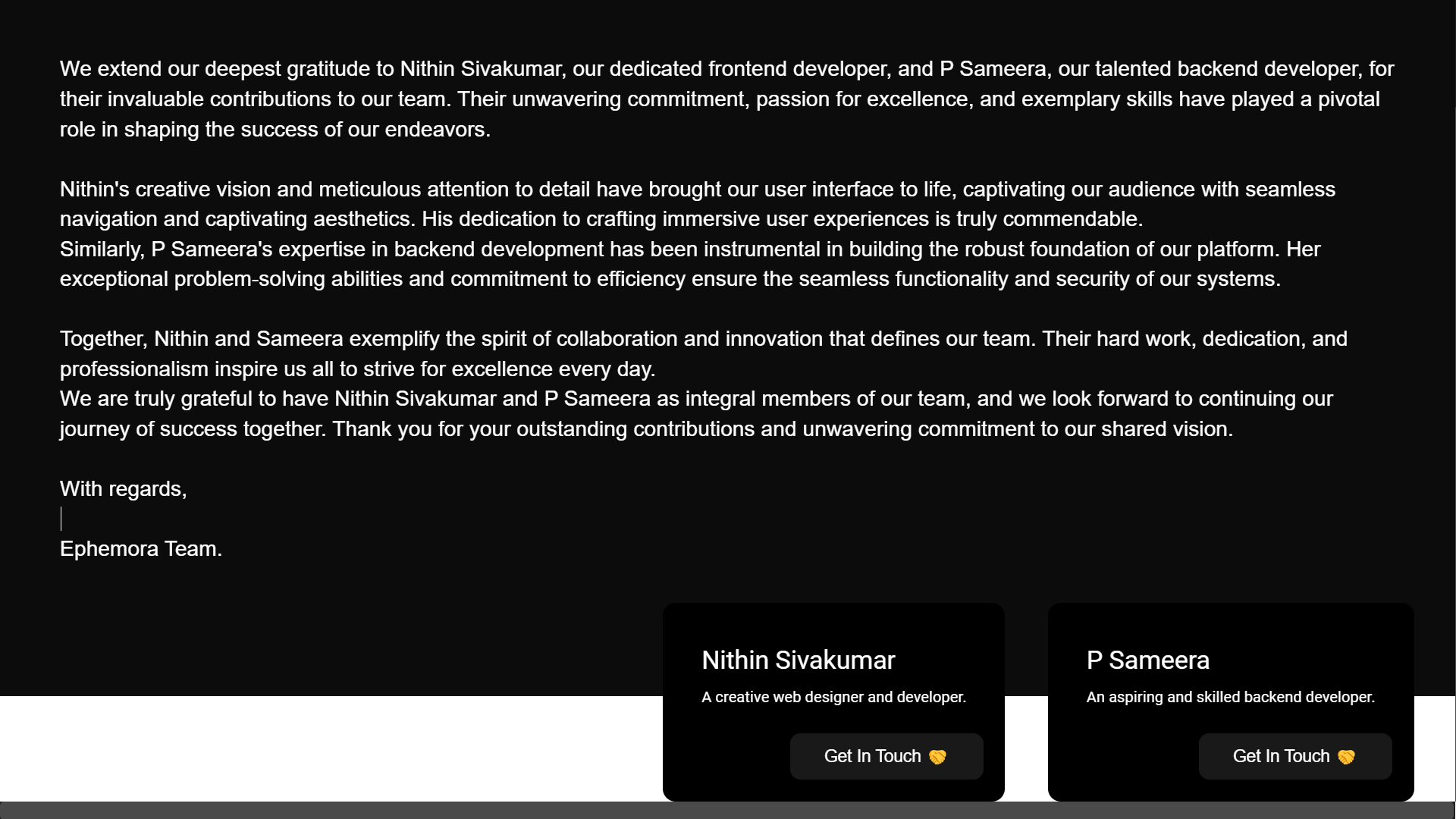


Figure 5.8:You can get in touch with us here

Conclusion

In conclusion, EPHEMORA stands as a testament to the harmonious fusion of

technology, culture, and individual well-being. It has been meticulously designed

to offer a multifaceted experience that transcends the conventional boundaries of

digital platforms.

The features embedded within EPHEMORA—ranging from guided breathing

exercises in the Breathe section to the vibrant connectivity fostered in the Social

section—aim to provide users with a space for introspection, interaction, and

cultural immersion. The Journal section, inspired by ancient Bharath, becomes a

canvas for users to paint their daily narratives, creating a bridge between past

traditions and contemporary self-expression.

The I'm Bored section, with its engaging questions and soon-to-be-introduced

music feature, adds layers to the overall user experience, ensuring that

EPHEMORA remains dynamic and relevant to the evolving needs of its users.

The seamless execution of CRUD operations, detailed in the backend operations

section, underscores the platform's reliability and scalability. EPHEMORA is not

just a product; it is a living, evolving entity that adapts to user needs and

aspirations.

As users navigate through the various sections, they are invited to breathe, connect,

reflect, and engage. The design, inspired by ancient Bharath, coupled with daily

quotes reflecting the ethos of Ramayana and Mahabharata, encapsulates a holistic

approach to user well-being.

In essence, EPHEMORA is more than a digital sanctuary—it is an evolving

ecosystem that invites users to embrace the essence of an ancient civilization in a

modern context. As the platform continues to grow, it remains steadfast in its

commitment to providing a captivating and enriching digital. experience, where

users can escape, engage, and find solace in the harmonious blend of tradition and

technology.

Bibliography

Fundamentals of Database Systems by Elmasri and Navathe,7th Edition, Addison

Wesley,2015.

URLS:

www.wikipedia.org

www.youtube.com

www.mysql.com/