

Project Report

On

## **“NOVAFIT a Fitness App”**

Submitted in partial fulfilment of the requirement for the Award of degree of

**Master of Computer Applications (MCA)**

By

Abhishek Singh (202120045)

**Under the Guidance of**

**Dr. Sanjay Sharma**



**Session 2020-23**

**Department Of Mathematics & Bioinformatics Computer Applications**

**MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY,**

**BHOPAL - 462003**

## Declaration

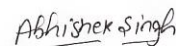
I, hereby declare that project entitled **NOVAFIT a Fitness App** submitted by me in partial fulfilment for the award of the degree of Master of Computer Applications in the Department of Mathematics and Computer Applications, Maulana Azad National Institute of Technology, Bhopal is an authentic work carried out from **10 Jan 2023 to 12 May 2023** under the guidance of **Dr. Sanjay Sharma**.

The matter embodied in this project has not been submitted by me or anybody else to any institution for award of any other degree or diploma.

(Signature of Student)

Abhishek Singh

202120045



**Counter Signed by:**

**Supervisor:**

**Dr. Sanjay Sharma**

**Department of Mathematics, Bioinformatics & Computer Applications**

**MANIT, Bhopal**

**Dr. Namita Shrivastava**

**Head, Department of Mathematics, Bioinformatics & Computer Applications**

**MANIT, Bhopal**

## **Acknowledgement**

Here, I gladly present this project report on **NOVAFIT a Fitness App** as part of the 6th semester MCA (Master's in computer applications). I take this occasion to thank God, almighty for blessing me with his grace and taking our endeavour to a successful culmination. I extend my sincere and heartfelt thanks to me esteemed guide **Dr. Sanjay Sharma** for providing me with the right guidance and advice at the crucial junctures and for showing me the right way. I extend my sincere thanks to my respected head of the department **Dr. Namita Shrivastava** for allowing me to use the facilities available.

I am highly indebted to **Dr. Ghanshyam Singh Thakur, Dr. Sujoy Das, Dr. Amit Bhagat and Dr. Vishnu Priya** for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. I would also like to thank **Enquero Global.** for providing me the opportunity to work on this project and I express my sincere gratitude to **Mr. Abhishek Chatterjee and Ms. Suparna Sen** for their affectionate coordination as a team in the work.

Last but not the least, I would like to express my gratitude towards my parents & my friends for the support and encouragement they have given me during the course of my work

Submitted By-  
Abhishek Singh (202120045)

# Internship Letter

Save & Return Later Submit



## INTERNSHIP LETTER

Dear Abhishek Singh,

On behalf of Enquero Global LLP, I am very pleased to offer you the position of Intern, reporting to Shanmugapriya Gnanasekaran. If you accept this offer, you will begin your internship with the Company on 06-Mar-2023 and will be expected to work around 40 hours per week. As we discussed, your internship will end on or around 06-Jul-2023.

The corporate office address is:

**Enquero Global LLP**  
**3rd Floor, No. 130, Indique - Grape Garden**  
**1st A Cross, 18th Main, Koramangala, 6th Block, Bengaluru-560095**

You will be paid ₹25,000 less all applicable taxes and withholdings, payable.

As an intern, you are specifically ineligible for any employee-related benefits, and you will not receive any of the employee benefits that regular Company's full-time employees receive, including but not limited to, health insurance, vacation or sick pay, paid holidays, participation in the Company's benefits package, etc. You agree and acknowledge that this internship does not create an employer-employee relationship, and you will not claim employer-employee relationship on any account whatsoever.

However, your internship with the Company is "at-will," which means that the Company may terminate your internship at any time, with or without notice. You are expected to provide a notice of least 7 days in case you wish to terminate your internship with the Company.

Any offer extended during campus hiring is subject to successful completion of your internship and graduation.

During your internship, you will have access to trade secrets and confidential business information belonging to the Company and that of the Company's customers. Any and all such information relating to the Company, its customers, its business, finances, operations, etc. shall be deemed to be confidential information and shall be kept strictly confidential by you; and you shall refrain from using it for your own purposes or from disclosing it to anyone outside the Company or for any other purpose (other than in connection with your internship with the Company).

Further, all services rendered by you and all intellectual property generated (including rights of copyrights, patents, designs, etc.) during the course of your internship with the Company shall be a work-made-for-hire for the Company, and solely of the benefit of the Company. The Company shall have all ownership rights, title and interest in all such work-made-for-hire by you, and the Company shall be entitled to use it in the manner it deems fit. All intellectual property rights and rights relating to any work-made-for-hire shall vest solely with the Company; and you shall refrain from using it for your own purposes or from disclosing it to anyone outside the Company or for any other purpose (other than in connection with your internship with the Company).

In addition, you agree that, upon conclusion of your employment, you will immediately return to the Company all of its property, equipment, and documents, including electronically stored information.

We are very excited about the prospect of you joining our team and staff as an Intern at Enquero Global. We look forward to helping you continue your education outside the classroom. To learn more about Enquero, please visit our website at [www.enquero.com](http://www.enquero.com).

Sincerely,  
Enquero Global LLP

**Surbhi Gupta**

Title: AVP- Talent Acquisition

☒ **Authorized Signatory Signature** Surbhi Gupta 14/12/2022 10:46 AM  
(checking the box above is equivalent to a handwritten signature)

My signature below indicates my acceptance of the offer as outlined above.

Abhishek Singh

☐ **Employee Signature** Abhishek Singh 26/12/2022 1:29 AM  
(checking the box above is equivalent to a handwritten signature)

ENQUERO GLOBAL LLP [www.enquero.com](http://www.enquero.com)  
Regd. Office: 136, 2nd Cross, 6th Block, Koramangala, Bangalore, Karnataka 560095, India LLPIN: AAH-4516 Registered with Limited Liability

## **Abstract**

Industrial training is an important phase of a student life. A well planned, properly executed and evaluated industrial training helps a lot in developing a professional attitude. It develops an awareness of industrial approach to problem solving, based on a broad understanding of process and mode of operation of organization. The aim and motivation of this industrial training is to receive discipline, skills, teamwork and technical knowledge through a proper training environment, which will help me, as a student in the field of Information Technology, to develop a responsiveness of the self-disciplinary nature of problems in information and communication technology. During a period of four month(s) training at **Enquero Global**, I was assigned to a team to develop and design NOVAFIT a Fitness App. Throughout this training, I have been learned various programming language that required for the development and designing of module, the process of the production lines and able to implement what I have learnt for the past three years as an **MCA** student in **MANIT Bhopal**.

# **TABLE OF CONTENTS**

- Company Profile..... 8

## **Project - NOVAFIT a Fitness App**

- Introduction ..... 9
- Purpose..... 10
- Scope..... 10
- Process Flow/App Transition ..... 11

## **System Requirement and Technology Used**

- System Requirement.....14
- MERN.....15
- Mongo Db.....16
- Express.....17
- React Js.....18
- Node Js .....18
- Code Snippet.....
- Mongo Db Database ..... 37
- Mongo Db Database Structure .....38

## **Future Scope**..... 42

## **Conclusion**.....43

## **Bibliography** .....44

## **LIST OF FIGURES**

• Figure 1 . Trainer Login Page.....	12
• Figure 2 . Trainee Signup Page.....	12
• Figure 3 . Trainee Dashboard.....	13
• Figure 4 . Trainee Profile.....	13
• Figure 5 . Checkout Page.....	14
• Figure 6 . Razorpay Success Page.....	14
• Figure 7 . Database Schema.....	37
• Figure 8 . Novafit Database Table.....	38
• Figure 9 . Trainee Signup Table.....	38
• Figure 10 . Trainer Signup Table.....	39
• Figure 11 . Payment Table.....	39
• Figure 12 . Admin Signup Table.....	40
• Figure 13 . Review Table.....	40
• Figure 14 . Faq Table.....	41
• Figure 1 . Contact Us Table.....	41

## **Company Profile**



Global Enquero Global is a technology solutions and consulting firm that provides digital transformation services to businesses across a range of industries. The company was founded in 2014 and is headquartered in Milpitas, California. Enquero's services include digital transformation consulting, data analytics, cloud computing, and software engineering. The company's solutions are tailored to each client's specific needs, and its team works closely with clients to understand their business objectives and design solutions that will help them achieve success. Enquero has a strong focus on innovation and collaboration, and its team is made up of talented and experienced professionals who are passionate about using technology to solve complex business challenges. The company has been named to the Inc. 5000 list of fastest-growing private companies in America for three consecutive years (2018-2020).

## **Organization History**

The company was founded in 2014 by CEO Arjun Devgan and COO Giri Devanur. Prior to founding Enquero, both Devgan and Devanur held leadership roles in various technology companies. Enquero started as a small company with just a handful of employees, but quickly grew to a team of over 1,500 people in multiple locations across the globe. In 2016, the company established its headquarters in Milpitas, California. Enquero has been recognized for its rapid growth and innovative solutions, including being named to the Inc. 5000 list of fastest-growing private companies in America for three consecutive years (2018-2020).

. The company has also formed partnerships with leading technology companies such as Amazon Web Services and Microsoft to provide its clients with the latest and most advanced technologies.

Overall, Enquero Global has a relatively short but impressive history, with a focus on delivering cutting-edge technology solutions and services to help its clients achieve their digital transformation goals.



# **NOVAFIT a Fitness App**

## **Introduction**

NOVAFIT a Fitness App is a collective effort of our team, aimed to ease the process of attendance tracking of employees, and calculate allowances for them accordingly. It is aimed to be used for all employees, from CTO to Interns to contract based employees. It has potential to be segregated to a generalised NOVAFIT application for customization and quick deployment for any company.

The best part of this application, that it is rigid enough, that no employee can lie about their attendance data, while flexible enough that allows allowances to be updated by the concerned HR or Manager. To achieve this, this application takes its input data from “Microsoft Dynamics” instead of manual input; removing redundancy in the process possibility of error in the data.

### Overall Features:

- Trainers and Trainee can Signup and Login.
- Trainee can purchase any subscription plan and can have complementary sessions.
- Subscription plan will be of 3 types silver ,gold ,platinum.
- Logging of all activities and data manipulation.
- Trainer will be allocated based on their experience and field
- Trainee can update their attendance
- A shop is also included in the app but only subscribed user can purchase ..
- Trainee can post the Review and can Contact us by filling the contact form.
- Admin can view all the trainee and trainers and can update the Faq , shop items
- Trainer can view how many trainee are associated with them..
- Authentication via JSONWEBTOKEN removing possibility of unauthorised access.

## **Purpose**

The Novafit app is a fitness app that is designed to help people improve their physical health and fitness levels. The app provides users with a range of features and tools to help them track their fitness progress, set fitness goals, and access a variety of workout programs and exercises.

Some of the main features of the Novafit app include personalized workout plans, tracking of daily activity levels, calorie tracking, meal planning and tracking, progress tracking, and social sharing. The app also provides users with access to a community of like-minded individuals who can offer support and motivation.

The ultimate purpose of the Novafit app is to help users achieve their fitness goals and improve their overall health and wellbeing. Whether you are looking to lose weight, build muscle, or simply maintain a healthy lifestyle, the Novafit app can be a valuable tool in helping you achieve your fitness goals.

This application runs on low tier devices with limited functionalities, and with caching in space, this serves the data just right, even without any connection to the internet.

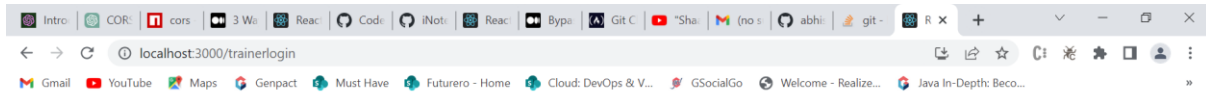
## **Scope**

1. This is a good platform for employees who want to see the allowance they will be receiving. The scope of the Novafit app is quite broad, as it is designed to cater to the needs of a wide range of fitness enthusiasts, from beginners to advanced users. Here are some of the key areas that the app covers:

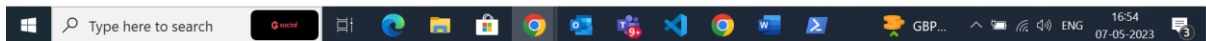
2. Personalized fitness plans: The Novafit app provides users with personalized workout plans based on their fitness level, goals, and preferences. These plans can be tailored to suit a wide range of fitness goals, from weight loss to muscle building.

3. Tracking and monitoring: The app enables users to track their daily activity levels, monitor their progress, and analyze their fitness data. Users can track metrics such as steps taken, calories burned, and distance covered. Overall, the Novafit app has a wide scope and is designed to provide users with a comprehensive set of tools and resources to help them achieve their fitness goals and improve their overall health and wellbeing.

## Trainer Login Page



### Trainer Login

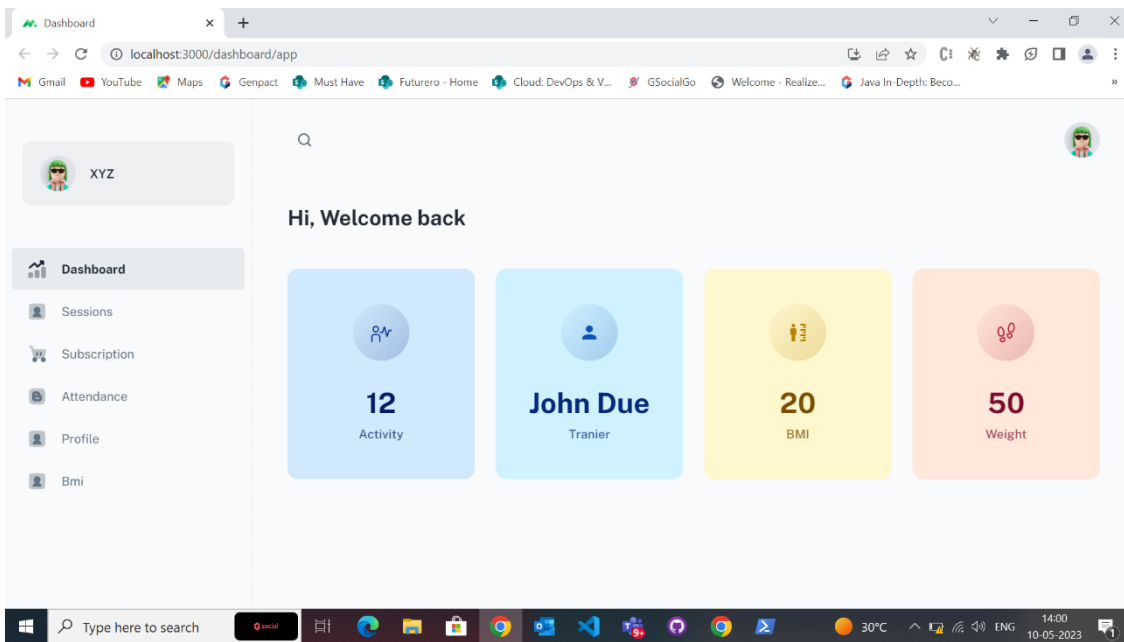
## Trainee Signup Page



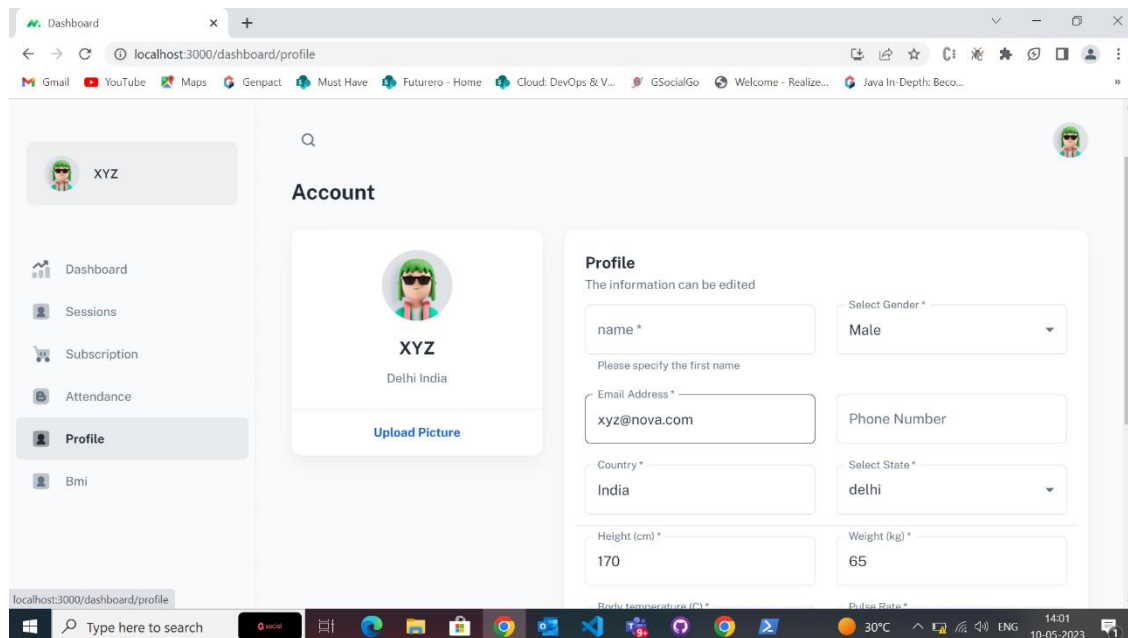
### Sign Up


# Trainee Dashboard



# Trainee Profile



## Checkout Page

The screenshot shows a web browser window with the title 'RazorPay Custom Checkout'. The address bar shows 'localhost:5000/checkout'. The page contains a 'Payment Details' form with the following fields:

- Plan:** A dropdown menu with 'Select Plan' and a placeholder text 'Please select an item in the list.'
- Person Name:** A text input field with 'Enter Your Name' as a placeholder.
- Card Number:** A text input field containing '4111 1111 1111 1111'.
- Expiry:** A text input field with 'MM/YY' as a placeholder.
- CVV/CVC:** A text input field with '\*\*\*' as a placeholder.
- Pay:** A blue button at the bottom of the form.

The Windows taskbar at the bottom shows the search bar, task view button, and several application icons. The system tray on the right shows the temperature (30°C), network status, and the date/time (14:03, 10-05-2023).

## Razorpay Success page

The screenshot shows the same web browser window as the previous one, but with a success modal displayed over the payment form. The modal contains:

- A green checkmark icon inside a circle.
- The text 'Success!' in bold.
- The text 'Your payment is sucess' (note the typo).
- An 'OK' button in the bottom right corner.

The background form is dimmed. The Windows taskbar and system tray are visible at the bottom, showing the same date and time (14:04, 10-05-2023).

## **System Requirement**

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements. A second meaning of the term of System requirements, is a generalisation of this first definition, giving the requirements to be met in the design of a system or subsystem. Typically, an organisation starts with a set of Business requirements and then derives the System requirements from there. Let us see some related aspects of system requirements (hardware and software) used in development of our project.

### **Hardware Specifications:**

- **Processor: Intel® Core™ i5.**
- **RAM: 8.0 GB**
- **Hard Disk: 50GB**
- **Processor Speed: 3.20GHz**
- **Internet Speed: 10Mbps**

### **Software Specifications:**

- **Operating System: Windows 10(64-bit)**
- **Development IDE: Vs Code**
- **Testing Tool: Postman, Thunder Client**
- **For Database Access : Mongo Db**
- **Other Tools: Swagger, Git, Github**

## Technology Used

Various Technologies used during Designing and Development of project are listed below.

There are many categories of technologies, frameworks and database used in development of the application. They are listed below.

### MERN



The Java Development Kit (JDK) is a distribution of Java Technology by Oracle Corporation. MERN is a popular acronym for a full-stack development framework consisting of four technologies: MongoDB, Express.js, React, and Node.js. Together, these technologies provide developers with a powerful and flexible toolset for building modern web applications.

Here is a brief overview of each technology in the MERN stack:

1. MongoDB: A popular NoSQL database that provides a flexible and scalable data storage solution.
2. Express.js: A web application framework for Node.js that provides a set of tools for building HTTP servers and APIs.
3. React: A JavaScript library for building user interfaces that allows developers to create reusable UI components.
4. Node.js: A JavaScript runtime that allows developers to run JavaScript code on the server-side.

MERN stack has gained a lot of popularity among web developers due to its flexibility, scalability, and ease of use. Developers can use this stack to build web applications that can handle large amounts of data, provide real-time updates, and scale easily as traffic grows.architecture.

## **Mongo DB**



MongoDB is a popular NoSQL document-oriented database system that was developed by MongoDB Inc. It is a highly scalable, open-source, and cross-platform database that allows developers to store and manage unstructured data.

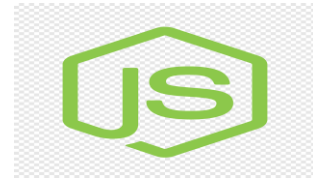
One of the main advantages of MongoDB is its ability to store data in flexible, JSON-like documents. This makes it easy to represent complex hierarchical relationships and makes it well-suited for storing data in object-oriented programming languages. MongoDB also supports dynamic schemas, which means that you can add new fields to a document without having to alter the existing schema.

Other key features of MongoDB include:

1. High availability and scalability: MongoDB supports horizontal scaling across multiple servers and provides automatic failover and load balancing.
2. Rich query language: MongoDB's query language supports complex queries and allows developers to retrieve and manipulate data in a flexible manner.
3. Document-based data model: MongoDB stores data in a format that is easy to understand and manipulate, making it well-suited for web applications.
4. Built-in replication and sharding: MongoDB provides built-in replication and sharding capabilities, which makes it easy to scale and distribute data across multiple nodes.

Overall, MongoDB is a powerful and flexible database system that is well-suited for modern web applications that require high scalability, flexibility, and ease of use. Lesser production time





## **Express**

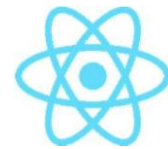
Eclipse is an integrated development environment (IDE) used in computer programming. Express.js is a web application framework for Node.js, designed to provide developers with a set of tools for building web applications and APIs. It provides a minimalistic and flexible approach to building web applications and provides a wide range of features and functionality to help developers build high-performance and scalable applications.

Some of the key features of Express.js include:

1. Routing: Express provides a simple and intuitive way to define application routes and HTTP methods, making it easy to handle requests and responses.
2. Middleware: Express allows developers to use middleware functions to modify incoming requests and outgoing responses. This allows for powerful features like authentication, error handling, and logging.
3. Templating engines: Express supports a wide range of templating engines, making it easy to create dynamic and responsive views for web applications.
4. Error handling: Express provides a robust error handling mechanism that allows developers to handle errors gracefully and return meaningful error messages to clients.
5. Security: Express provides a range of security features, including HTTPS support, cookie parsing, and cross-site scripting (XSS) protection, to help ensure the security of web applications.

Overall, Express.js is a powerful and flexible web application framework that provides developers with a wide range of features and functionality to build high-performance and scalable web applications and APIs. It is widely used in the development of Node.js applications and is highly popular in the web development community.

## **React JS**



React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page, mobile, or server-rendered applications with frameworks like Next.js. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

## **Node Js**



Node.js is an open-source, cross-platform, JavaScript runtime environment that allows developers to run JavaScript code on the server-side. It was built on the Google Chrome V8 JavaScript engine and provides a rich set of APIs and modules for building web applications and command-line tools.

Some of the key features of Node.js include:

1. **Asynchronous I/O:** Node.js is designed to handle I/O operations asynchronously, which means that it can handle a large number of concurrent requests without blocking the event loop. This makes it ideal for building real-time applications that require fast and efficient data transfer.
2. **Scalability:** Node.js is highly scalable and can handle large amounts of data and traffic. It uses an event-driven, non-blocking I/O model that allows it to handle multiple concurrent connections without using threads or processes.

Overall, Node.js is a powerful and versatile platform that allows developers to build fast, scalable, and efficient web applications and command-line tools using JavaScript. It is widely used in the development of modern web applications, including real-time applications, streaming services, and APIs.

# CODE SNIPPETS

## Schema Models Of App

### Trainee Signup Schema

```
const mongoose=require('mongoose');

const { Schema } = mongoose;

const TraineeSchema = new Schema({
  name:{
    type:String,
    required:true
  },
  email:{
    type:String,
    required:true,
    unique:true
  },
  password:{
    type:String,
    required:true
  },
  isSubscribed:{
    type:Boolean,
    default:false
  },
  date:{
    type:Date,
    default:Date.now
  },
});
const Trainee=
mongoose.model('trainee',TraineeSchema)
module.exports=Trainee
```

### Trainer Signup Schema

```
const mongoose = require('mongoose');
const TrainerSchema = new
mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
  typeOfTraining: {
    type: String,
    enum: ['swimming', 'zumba', 'gym',
'power yoga','fat
buring','cycling','strength training'],
    required: true,
  },
  experience:{
    type:Number,
    required:true
  },
  location: {
    type: String,
    required: true,
  },
});

const Trainer =
mongoose.model('trainer',
TrainerSchema);

module.exports = Trainer;
```

## Admin Signup Schema

```
const mongoose=require('mongoose');
const { Schema } = mongoose;

const AdminSchema = new Schema({
  name:{
    type:String,
    required:true
  },
  email:{
    type:String,
    required:true,
    unique:true
  },
  password:{
    type:String,
    required:true
  },
  date:{
    type:Date,
    default:Date.now
  },
});
const Admin= mongoose.model('admin',AdminSchema)
module.exports=Admin
```

## Review Schema

```
const
mongoose=require('mongoose');
const { Schema } = mongoose;

const ReviewSchema = new
Schema({
  trainee:{

type:mongoose.Schema.Types.ObjectId,
    ref: 'trainee'
  },
  title:{
    type:String,
    required:true
  },
  description:{
    type:String,
    required:true,
  },
  tag:{
    type:String,
    default:"general"
  },
  date:{
    type:Date,
    default:Date.now
  },
});

module.exports=
mongoose.model('review',ReviewS
chema)
```

## Faq Schema

```
const mongoose=require('mongoose');

const { Schema } = mongoose;

const FaqSchema = new Schema({
  admin:{

type:mongoose.Schema.Types.ObjectId,
    ref: 'admin'
  },
  question:{
    type:String,
    required:true,
  },
  answer:{
    type:String,
    required:true
  },
  tag:{
    type:String,
    default:"general"
  }
});
const Faq=
mongoose.model('faq',FaqSchema)
module.exports=Faq
```

## Trainee Profile Schema

```
const mongoose = require("mongoose");
const { Schema } = mongoose;
const ProfileSchema = new Schema({
  userId:{
    type: mongoose.Schema.Types.ObjectId,
    ref: 'trainee',
    required:true
  },
  name:{
    type:String,
    required:true
  },
  email:{
    type:String,
    required:true
  },
  gender:{
    type:String,
    required:true
  },
  dob:{
    type:Date,
    required:true
  },
  height:{
    type:Number,
    required:true
  },
  weight:{
    type:Number,
    required:true
  },
  program:{
    type:String,
    required:true
  },
  },
  vitals: [{
    body_temp:
    { type: Number, required: true, },

    pulse_rate:
    { type: Number, required: true, },

    resp_rate:
    { type: Number, required: true, },

    blood_pressure:
    { type: Number, required: true, },
  }],
});
module.exports = mongoose.model('Profile',ProfileSchema);
```

## Subscription Schema

```
const mongoose =
require('mongoose');
const subscriptionSchema = new
mongoose.Schema({
  name: {
    type: String,
    required: true,
    enum: ['platinum', 'gold',
'silver']
  },
  price: {
    type: Number,
    required: true,
    min: 0
  },
  duration: {
    type: Number,
  },
  sessions: {
    type: Number,
  },
  date: {
    type: Date,
    default: Date.now
  },
  trainer: {
    type:
mongoose.Schema.Types.ObjectId,
    ref: 'trainer'
  },
  trainee: {
    type:
mongoose.Schema.Types.ObjectId,
    ref: 'trainee',
    required: true
  },
  expiry: {
    type: Date
  },
  isActive: {
    type: Boolean,
    default: true
  },
  status: {
    type: String,
    enum: ['pending',
'success', 'failed'],
    default: 'failed'
  }
});
module.exports =
mongoose.model('Subscription',
subscriptionSchema);
```

## Sessions Schema

```
const mongoose = require("mongoose")
const sessionSchema = new
mongoose.Schema({
  userId: {
    type:
mongoose.Schema.Types.ObjectId,
    ref: "trainee",
    default: null,
  },
  trainerId: {
    type:
mongoose.Schema.Types.ObjectId,
    ref: "trainer",
  },
  activity: {
    type: String,
    required: true,
  },
  slot: {
    type: String,
    required: true,
  },
  date: {
    type: Date,
    required: true,
  },
  isAvailable: {
    type: Boolean,
    default: true,
  },
});
module.exports =
mongoose.model("Sessions", sessionSchem
a);
```

## **Payment Schema**

```
const mongoose = require('mongoose');

const paymentSchema = new mongoose.Schema(
  {
    orderId: {
      type: String,
      required: true,
    },
    paymentId: {
      type: String,
      required: true,
    },
    signature: {
      type: String,
      required: true,
    },
    status: {
      type: String,
      enum: ["success", "fail"],
      required: true,
    },
    name: {
      type: String,
    },
    plan: {
      type: String,
    },
    email: {
      type: String,
      required: true,
    }
  },
  {
    timestamps: true,
  }
);

module.exports = mongoose.model("Payment", paymentSchema);
```



## Admin Auth Controller

```
const express = require('express');
const {body, validationResult} = require('express-validator');
const router = express.Router()
const Admin = require('../models/Admin')
const bcrypt = require('bcryptjs')
const jwt = require('jsonwebtoken');
// const {is} = require('express/lib/request');
const fetchadmin=require("../middleware/fetchadmin")
const JWT_SECRET = "Abhiisgood$hi"

// Route 1 create a user doesn't require auth api/auth/createuser no login required
router.post('/createuser', [
  body('name', 'Name must be 3character Long').isLength(
    {min: 3}
  ),
  body('email', 'Enter a Valid Email').isEmail(),
  body('password', 'Password must be 3 character Long').isLength(
    {min: 3}
  )
], async (req, res) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
    return res.status(400).json({errors: errors.array()});
  }
  try {
    let admin = await Admin.findOne({email: req.body.email});
    if (admin) {
      return res.status(400).json({error: "Sorry email already exists"})
    }
    const salt = await bcrypt.genSalt(10);
    const secPass = await bcrypt.hash(req.body.password, salt);
    admin= await Admin.create({name: req.body.name, password: secPass, email:
req.body.email})
    const data = {
      admin: {
        id: admin.id
      }
    }

    const authtoken = jwt.sign(data, JWT_SECRET);

    res.json({authtoken})
  } catch (error) {
    console.error(error.message)
    res.status(500).send("Internal Server Error")
  }
})
})
```

```

// Route 2 Authenticate a user doesn't require auth api/auth/login, no login required
router.post('/login', [

  body('email', 'Enter a Valid Email').isEmail(),
  body('password', 'Password can not be blank').exists()
], async (req, res) => {

  const errors = validationResult(req);
  if (! errors.isEmpty()) {
    // console.log("jhi")
    return res.status(400).json({errors: errors.array()});
  }
  const {email, password} = req.body;
  try {
    let admin = await Admin.findOne({email});
    if (! admin) {
      return res.status(400).json({error: "please try to login with correct
credintails"})
    }
    const passwordCompare = await bcrypt.compare(password, admin.password);
    if (! passwordCompare) {
      return res.status(400).json({error: "please try to login with correct
credintails"})
    }
    const data = {
      admin: {
        id: admin.id
      }
    }
    // console.log("hii")
    const authToken = jwt.sign(data, JWT_SECRET);
    res.json({authToken})

  } catch (error) {
    console.error(error.message)
    res.status(500).send("Internal Server Error")
  }
})

// Route 3 Get Logged user details using POST auth api/auth/getuser, Login required
router.post('/getuser', fetchadmin, async (req, res) => {
  try {
    adminId=req.admin.id
    const admin =await Admin.findById(adminId).select("-password")
    res.send(admin)
  } catch (error) {

    console.error(error.message);
    res.status(500).send("Internal Server Error")
  }
})

```

## Trainee Auth Controller

```
const express = require('express');
const {body, validationResult} = require('express-validator');
const router = express.Router()
const Trainee = require('../models/Trainee')
const bcrypt = require('bcryptjs')
const jwt = require('jsonwebtoken');
// const {is} = require('express/lib/request');
const fetchtrainee=require("../middleware/fetchtrainee")
const JWT_SECRET = "Abhiisgood$hi"

// Route 1 create a user doesn't require auth api/auth/createuser no login required
router.post('/createuser', [
  body('name', 'Name must be 3character Long').isLength(
    {min: 3}
  ),
  body('email', 'Enter a Valid Email').isEmail(),
  body('password', 'Password must be 3 character Long').isLength(
    {min: 3}
  )
], async (req, res) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
    return res.status(400).json({errors: errors.array()});
  }
  try {
    let trainee = await Trainee.findOne({email: req.body.email});
    if (trainee) {
      return res.status(400).json({error: "Sorry email already exists"})
    }
    const salt = await bcrypt.genSalt(10);
    const secPass = await bcrypt.hash(req.body.password, salt);
    trainee= await Trainee.create({name: req.body.name, password: secPass, email:
req.body.email,isSubscribed:req.body.isSubscribed})
    const data = {
      trainee: {
        id: trainee.id
      }
    }

    const authToken = jwt.sign(data, JWT_SECRET);

    res.json({authToken})
  } catch (error) {
    console.error(error.message)
    res.status(500).send("Internal Server Error")
  }
})
})
```

```

// Route 2 Authenticate a user doesn't require auth api/auth/login, no login required
router.post('/login', [

  body('email', 'Enter a Valid Email').isEmail(),
  body('password', 'Password cant not be blank').exists()
], async (req, res) => {

  const errors = validationResult(req);
  if (! errors.isEmpty()) {
    // console.log("jhi")
    return res.status(400).json({errors: errors.array()});
  }
  const {email, password} = req.body;
  try {
    let trainee = await Trainee.findOne({email});
    if (! trainee) {
      return res.status(400).json({error: "please try to login with correct
credintails"})
    }
    const passwordCompare = await bcrypt.compare(password, trainee.password);
    if (! passwordCompare) {
      return res.status(400).json({error: "please try to login with correct
credintails"})
    }
    const data = {
      trainee: {
        id: trainee.id
      }
    }
    // console.log("hii")
    const authtoken = jwt.sign(data, JWT_SECRET);
    res.json({authtoken})

  } catch (error) {
    console.error(error.message)
    res.status(500).send("Internal Server Error")
  }
})

// Route 3 Get logged user details using POST auth api/auth/getuser, Login required
router.post('/getuser', fetchtrainee, async (req, res) => {
  try {
    traineeId=req.trainee.id
    const trainee =await Trainee.findById(traineeId).select("-password")
    res.send(trainee)
  } catch (error) {

    console.error(error.message);
    res.status(500).send("Internal Server Error")
  }
})
module.exports = router

```

## Auth Trainer Controller

```
const express = require('express');
const {body, validationResult} = require('express-validator');
const router = express.Router()
const Trainer = require('../models/Trainer')
const bcrypt = require('bcryptjs')
const jwt = require('jsonwebtoken');
// const {is} = require('express/lib/request');
const fetchtrainer=require("../middleware/fetchtrainer")
const JWT_SECRET = "Abhiisgood$hi"

// Route 1 create a user doesn't require auth api/authtrainer/createuser no login required
router.post('/createuser', [
  body('name', 'Name must be 3character Long').isLength(
    {min: 3}
  ),
  body('email', 'Enter a Valid Email').isEmail(),
  body('password', 'Password must be 3 character Long').isLength(
    {min: 3}
  )
], async (req, res) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
    return res.status(400).json({errors: errors.array()});
  }
  try {
    let trainer = await Trainer.findOne({email: req.body.email});
    if (trainer) {
      return res.status(400).json({error: "Sorry email already exists"})
    }
    const salt = await bcrypt.genSalt(10);
    const secPass = await bcrypt.hash(req.body.password, salt);
    trainer = await Trainer.create({name: req.body.name, password: secPass, email:
req.body.email,experience:req.body.experience,typeOfTraining:req.body.typeOfTraining,Location:req.body.Location})
    const data = {
      trainer: {
        id: trainer.id
      }
    }
    const authtoken = jwt.sign(data, JWT_SECRET);

    res.json({authtoken})
  } catch (error) {
    console.error(error.message)
    res.status(500).send("Internal Server Error")
  }
})

// Route 2 Authenticate a user doesn't require auth api/auth/login, no login required
```

```

// Route 2 Authenticate a user doesn't require auth api/auth/login, no login required
router.post('/login', [

  body('email', 'Enter a Valid Email').isEmail(),
  body('password', 'Password can not be blank').exists()
], async (req, res) => {

  const errors = validationResult(req);
  if (! errors.isEmpty()) {
    return res.status(400).json({errors: errors.array()});
  }
  const {email, password} = req.body;
  try {
    let trainer = await Trainer.findOne({email});
    if (! trainer) {
      return res.status(400).json({error: "please try to login with correct
credintails"})
    }
    const passwordCompare = await bcrypt.compare(password, trainer.password);
    if (! passwordCompare) {
      return res.status(400).json({error: "please try to login with correct
credintails"})
    }
    const data = {
      trainer: {
        id: trainer.id
      }
    }
    const authToken = jwt.sign(data, JWT_SECRET);
    res.json({authToken})
  } catch (error) {
    console.error(error.message)
    res.status(500).send("Internal Server Error")
  }
})

// Route 3 Get Logged user details using POST auth api/auth/getuser, Login required
router.post('/getuser', fetchtrainer, async (req, res) => {
  try {
    trainerId=req.trainer.id
    const trainer =await Trainer.findById(trainerId).select("-password")
    res.send(trainer)
  } catch (error) {

    console.error(error.message);
    res.status(500).send("Internal Server Error")
  }
})
module.exports = router

```

## Trainee Profile Controller

```
const express = require('express');
const Profile = require('../models/TraineeProfile');
const Trainee = require('../models/Trainee');
const router = express.Router();
const { body, validationResult } = require('express-validator');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
var fetchuser = require('../middleware/fetchtrainee');
const fetchtrainee = require('../middleware/fetchtrainee');
const JWT_SECRET = "Abhiisgood$hi"

//Route 1: Creating user profile :POST method Login required

router.post('/createtraineeprofile', fetchtrainee, async (req, res) => {
  const {dob, gender, weight, height, program, vitals} = req.body;

  try{
    const traineeId = req.traineeId;
    console.log(traineeId);
    const trainee = await Trainee.findById(traineeId);

    if(!trainee){
      return res.status(404).send("user not found");
    }

    const profile = new Profile({
      userId:trainee._id,
      name:trainee.name,
      email:trainee.email,
      dob,
      gender,
      weight,
      height,
      program,
      vitals
    });

    await profile.save();

    return res.status(201).send({ message: 'profile created successfully'});
  }catch(err){
    console.log(err);
    return res.status(500).send({message: 'Server error'});
  }
});
```

```

//Route 2: update user profile using PUT: Login Required

router.put('/updatetraineeprofile/:id',fetchtrainee,async(req,res) =>{
  const profileId = req.params.id;
  const{ dob,gender,weight,height,vitals } = req.body;

  try{
    const profile = await Profile.findById(profileId);

    if(! profile){
      return res.status(404).send({message: 'Profile not found'});
    }

    profile.dob = dob || profile.dob;
    profile.gender = gender || profile.gender;
    profile.weight = weight || profile.weight;
    profile.height = height || profile.height;
    profile.vitals = vitals || profile.vitals

    await profile.save();

    return res.status(200).send({message: 'profile updated Successfully'});
  }catch(err){
    console.error(err);
    return res.status(500).send({message: ' Server error'});
  }
})

// Route 3 Get Trainee profile Login required
router.get('/gettraineeprofile/:id',fetchtrainee,async (req,res) => {
  const profileId = req.params.id;

  try{
    const profile = await Profile.findById(profileId);

    if(! profile){
      return res.status(404).send({message: 'Profile not found'});
    }

    res.json({
      name:profile.name,
      email:profile.email,
      gender:profile.gender,
      dob:profile.dob,
      height:profile.height,
      weight:Profile.weight,
      userId:profile.userId,
      vitals:profile.vitals,
      program:profile.program
    });

  }catch(err){
    console.error(err);
    res.send(500).send("Internal Server Error");
  }
});

module.exports = router;

```



## Subscription Payment Controller

```
const express = require("express");
const router = express.Router();
const Razorpay = require("razorpay");
const mongoose = require("mongoose");
const { stringify } = require("querystring");
const Payment = require('../models/Payment');
const handleSubscription = require('../subscription');

const instance = new Razorpay({
  key_id: "rzp_test_YLxTdTGvq82Ceb",
  key_secret: "DL8A64vNjo1YPs03zp5cMgTs",
});

router.get("/", (req, res) => {
  let amount;
  const plan = req.query.plan; // Get the selected plan from the query parameter

  if (plan === "Gold") {
    amount = 1200 * 100; // Set amount to 1200 INR for Gold plan
  } else if (plan === "Silver") {
    amount = 500 * 100; // Set amount to 500 INR for Silver plan
  } else if (plan === "Platinum") {
    amount = 2500 * 100; // Set amount to 2500 INR for Platinum plan
  } else {
    amount = 500 * 100; // Default amount is 500 INR
  }

  var options = {
    amount: amount,
    currency: "INR",
  };
  instance.orders.create(options, function (err, order) {
    if (err) {
      console.log(err);
      return res.status(500).send("Unable to create order");
    } else {
      console.log(order);
      return res.render("checkout", {
        amount: order.amount,
        order_id: order.id,
      });
    }
  });
});
```

```

router.post("/pay-verify", async (req, res) => {
  // console.log(req.body);
  body = req.body.razorpay_order_id + "|" + req.body.razorpay_payment_id;
  var crypto = require("crypto");
  var expectedSignature = crypto
    .createHmac("sha256", "DL8A64vNjo1YPs03zp5cMgTs")
    .update(body.toString())
    .digest("hex");
  // console.log("sig" + req.body.razorpay_signature);
  // console.log("sig" + expectedSignature);

  if (expectedSignature === req.body.razorpay_signature) {
    console.log("Payment Success");
    // console.log(req.body);
    const payment = new Payment({
      orderId: req.body.razorpay_order_id,
      paymentId: req.body.razorpay_payment_id,
      signature: req.body.razorpay_signature,
      status: "success",
      name: req.body.razorpay_name,
      email: req.body.razorpay_email,
      plan: req.body.razorpay_plan
    })
    // console.log(payment.plan)
    await payment.save();
    const paymentId = payment.paymentId;
    const email = payment.email;
    handleSubscription(email, paymentId);
    return res.send("Payment successful");
  } else {
    console.log("Payment Fail");
    const payment = new Payment({
      orderId: req.body.razorpay_order_id,
      paymentId: req.body.razorpay_payment_id,
      signature: req.body.razorpay_signature,
      status: "fail",
      name: razorpay_name,
      email: razorpay_email,
      plan: razorpay_plan,
    });
    await payment.save();
    return res.status(500).send("Payment failed");
  }
});
module.exports = router;

```

## Sessions Controller

```
const express = require('express')
const mongoose = require("mongoose");
const router = express.Router();
const Trainer = require('../models/Trainer');
const Subscription = require('../models/SubscriptionSchema');
const Session = require('../models/Sessions');
const fetchtrainee = require('../middleware/fetchtrainee')
//const { computeHeadingLevel } = require('@testing-library/react');
router.post('/booksession', async (req, res) => {
  try {
    const { userId, activity, slot, date } = req.body;

    // Find the subscription for the user
    const subscription = await Subscription.findOne({trainee:userId,isActive:true });
    if (!subscription) {
      return res.status(400).json({ message: "User does not have an active subscription"
});
    }

    // Check if the user has any remaining sessions in their subscription
    if (subscription.sessions < 1) {
      return res.status(400).json({ message: "User does not have any remaining sessions
in their subscription" });
    }

    // Find the session to book
    const sess = await Session.findOne({userId,activity, slot, date });
    if (sess) {
      return res.status(400).json({ message: "you have already booked the session" });
    }

    // Check if the session has reached its maximum capacity
    const sessionCount = await Session.countDocuments({ activity: activity, slot: slot,
date: date});
    if (sessionCount >= 2) {
      return res.status(400).json({ message: "Slot not available" });
    }

    // Update the session with the user and decrement the subscription's sessionsLeft count

    const session = new Session({
      userId :userId,
      activity :activity,
      slot : slot,
      date :date,
      trainerId :null,
    })
```

```

await session.save();
subscription.sessions -= 1;
await subscription.save();

res.json({ message: "Session booked successfully" });
} catch (err) {
console.log(err);
res.status(500).json({ message: "Server Error" });
}
})

router.post('/cancelSession' , async (req, res) => {
  try {
    const { userId, sessionId } = req.body;

    // Find the session to cancel
    const session = await Session.findById(sessionId);
    if (!session) {
      return res.status(400).json({ message: "Session not found" });
    }

    // Check if the user owns the session
    if (session.userId.toString() !== new mongoose.Types.ObjectId(userId).toString()) {
      return res.status(400).json({ message: "User does not own the session" });
    }

    // Calculate the time difference between the session start time and the current time
    const sessionStart = new Date(`${session.date} ${session.slot}`);
    const timeDiff = sessionStart.getTime() - Date.now();

    // Check if the time difference is Less than 48 hours (172800000 milliseconds)
    if (timeDiff < 60000) {
      return res.status(400).json({ message: "Session cannot be cancelled within 48 hours of its start time" });
    }

    // Reset the session properties and save the changes to the Session collection
    //session.userId = null;
    //session.trainerId = null;

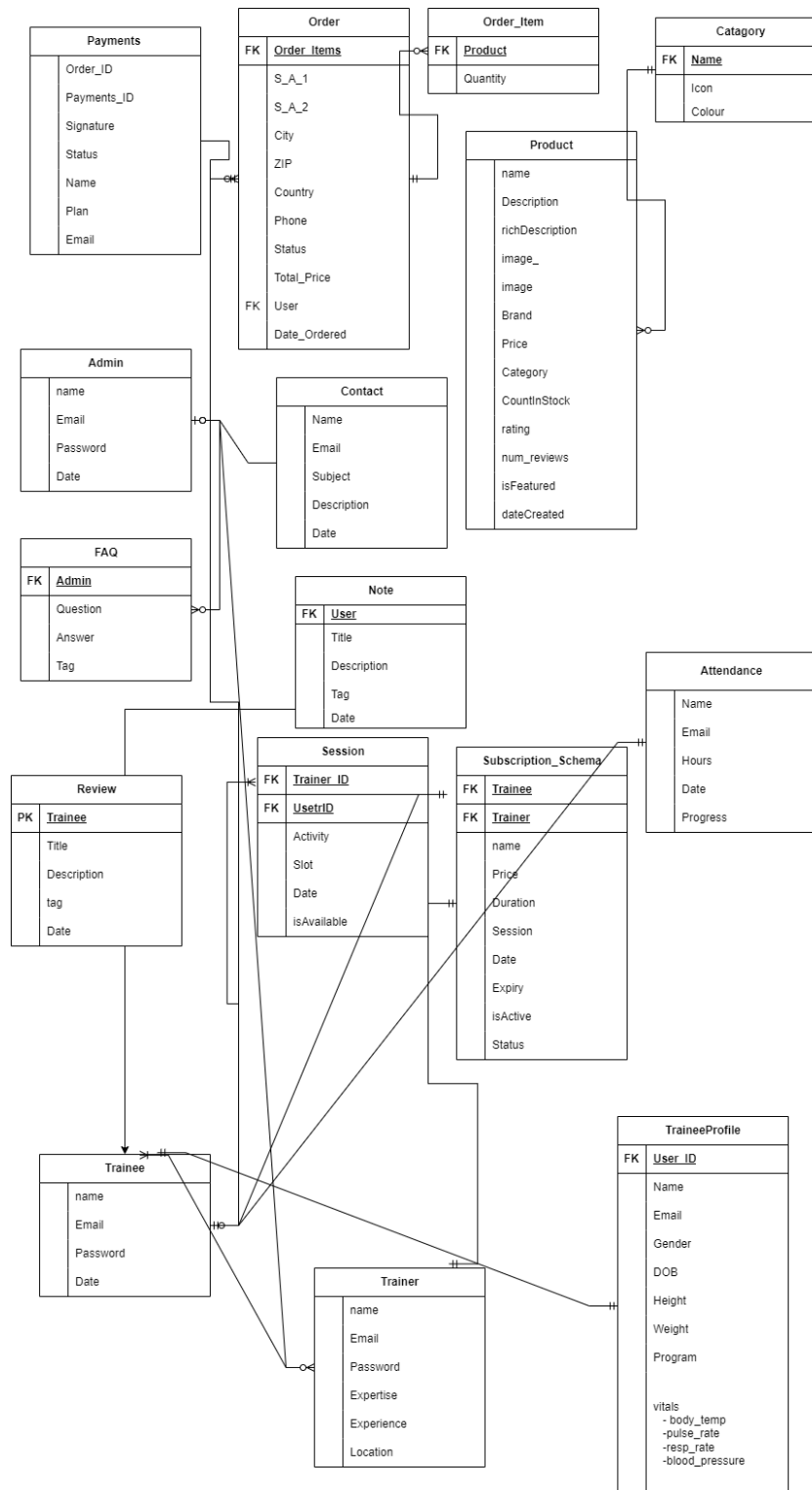
    session.isAvailable = true;
    await session.save();

    // Increment the subscription's sessionsLeft count and save the changes to the Subscription collection
    const subscription = await Subscription.findOne({ user: userId, isActive: true });
    subscription.sessions += 1;
    await subscription.save();

    res.json({ message: "Session cancelled successfully" });
  }
});

```

## Database Schema



# Table Screenshots

## Novafit DataBase

novafit-backend

LOGICAL DATA SIZE: 7.79KB   STORAGE SIZE: 228KB   INDEX SIZE: 336KB   TOTAL COLLECTIONS: 9

CREATE COLLECTION

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
admins	3	562B	188B	36KB	2	72KB	36KB
contacts	6	1.02KB	175B	36KB	1	36KB	36KB
faqs	12	2.46KB	211B	36KB	1	36KB	36KB
profiles	0	0B	0B	4KB	1	4KB	4KB
reviews	2	318B	159B	36KB	1	36KB	36KB
sessions	0	0B	0B	4KB	1	4KB	4KB
subscriptions	0	0B	0B	4KB	1	4KB	4KB
trainees	10	1.85KB	190B	36KB	2	72KB	36KB

Trainee Signup Table

novafit-backend.trainees

STORAGE SIZE: 36KB   LOGICAL DATA SIZE: 1.85KB   TOTAL DOCUMENTS: 10   INDEXES TOTAL SIZE: 72KB

Find   Indexes   Schema Anti-Patterns   Aggregation   Search Indexes   Charts

Filter

Type a query: { field: 'value' }

Reset   Apply

QUERY RESULTS: 1-10 OF 10

```
_id: ObjectId('644620715e5b2bbfd5e2ef68')
name: " Abhishek Singh"
email: "abhishekNovaFit123@gmail.com"
password: "$2a$10$q7K2C79pxgwQjR/cIbI.AuERuv.Z1KDs0bMQRcu.zRRNxzwHndW$6"
role: "user"
date: 2023-04-24T06:23:45.872+00:00
__v: 0
```

## Trainer Signup Table

### novafit-backend.trainers

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 1.59KB TOTAL DOCUMENTS: 7 INDEXES TOTAL SIZE: 72KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes Charts ●

Filter Type a query: { field: 'value' }

Reset

Apply

QUERY RESULTS: 1-7 OF 7

```
{
  "_id": "ObjectId('6446236e3c9bdfd11b30fb6')",
  "name": "Abhishek Singh",
  "email": "abhishekNovaFirainer@gmail.com",
  "password": "$2a$10$U4tw9gML6rYrZeSGPemZxuZo2PuvVgsiBGxH0aqp4/0/GM2FBe0M.",
  "typeOfTraining": "gym",
  "location": "bangalore",
  "__v": 0
}
```

## Payment Table

### razorpay.payments

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 2.48KB TOTAL DOCUMENTS: 9 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes Charts ●

Filter Type a query: { field: 'value' }

Reset

Apply

```
{
  "_id": "ObjectId('6454ad808b47f7f79f46414f')",
  "orderId": "order_LltbRiT5nmXLS",
  "paymentId": "pay_LltbivnWqHE4Z",
  "signature": "380a2eefd09d2596b2152d54d16c34b03fce21508ae68e16627a3377b1862860",
  "status": "success",
  "name": "Abhishek Singh",
  "plan": "platinum",
  "createdAt": "2023-05-05T07:17:20.943+00:00",
  "updatedAt": "2023-05-05T07:17:20.943+00:00",
  "__v": 0
}
```

## Admin Signup Table

### novafit-backend.admins

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 562B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 72KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes Charts ●

Filter  Type a query: { field: 'value' }

Reset

Apply

QUERY RESULTS: 1-3 OF 3

 `_id: ObjectId('644b98e2dca628f7f3214096')  
name: " Anigdha Bansal"  
email: "abcdefgh@gmail.com"  
password: "$2a$10$1hRw2k9ewkezPQB0aM3Th.u6VijRRPNkNW1PVirhyFNYW6begDboe"  
date: 2023-04-28T09:58:58.090+00:00  
__v: 0`  

## Review Table

### novafit-backend.reviews

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 318B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 36KB



Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes Charts ●

Filter  Type a query: { field: 'value' }

Reset

Apply

QUERY RESULTS: 1-2 OF 2

 `_id: ObjectId('644b93f5206e0601d131c2af')  
trainee: ObjectId('644b619bfa6d46edb6bf6723')  
title: "about Sessions"  
description: "its good your sesssions are good"  
tag: "general"  
date: 2023-04-28T09:37:57.801+00:00  
__v: 0`  



## Faq Table

### novafit-backend.faqs

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 2.46KB TOTAL DOCUMENTS: 12 INDEXES TOTAL SIZE: 36KB

Find

Indexes

Schema Anti-Patterns 0

Aggregation

Search Indexes

Charts

Filter

Type a query: { field: 'value' }

Reset

Apply

QUERY RESULTS: 1-12 OF 12

```
_id: ObjectId('644b9cc8be80194773d5c235')
admin: ObjectId('644b98e2dca628f7f3214096')
question: "how to book a session??"
answer: "you have to go to home page and then from there you can find a button ..."
tag: "general"
__v: 0
```

## ContactUs Table

### novafit-backend.contacts

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 1.16KB TOTAL DOCUMENTS: 7 INDEXES TOTAL SIZE: 36KB

Find

Indexes

Schema Anti-Patterns 0

Aggregation

Search Indexes

Charts

Filter

Type a query: { field: 'value' }

Reset

Apply

QUERY RESULTS: 1-7 OF 7



```
_id: ObjectId('644610188969740b71f9e49e')
name: "Abhishek"
email: "abhi@gmail.com"
subject: "for membership program"
description: "hey for 3 months any good plan for me as i can come on weekdays only"
date: 2023-04-24T05:14:00.381+00:00
__v: 0
```



## **Future Scope of Project**

- **2-way Authentication factor for password validation techniques:** With 2-Step Verification (also known as two-factor authentication), you add an extra layer of security to your account in case your password is stolen.
- **Integration with wearables and IoT devices:** Fitness apps will continue to integrate with wearable devices like smartwatches, fitness trackers, and IoT devices like smart scales and heart rate monitors. This will allow NovaFit to track users' activities and health metrics more accurately and provide better recommendations.
- **Gamification:** Fitness apps will increasingly incorporate gamification elements to keep users engaged and motivated. NovaFit could use gamification to create challenges, rewards, and leaderboards to motivate users to stick to their fitness goals.
- **Social features:** Social features like chat rooms, forums, and social media integration will allow users to connect with others who share similar fitness goals, creating a sense of community and support. NovaFit could leverage social features to create a network of users who support and motivate each other in their fitness journeys.
- **Implementing SMTP protocol:** SMTP as a method to transfer mail from one user to another. The end-to-end model is used to communicate between different organizations whereas the store and forward method is used within an organization.

## **Conclusion**

- With the emerging rise of Internet, and its growing usage, this application will serve its purpose of delivering the required data to its stakeholders in time.
- The web is an ever green, growing and trusted platform, and its usage will not cause any issues in usability or accessibility in the foreseeable future.
- The “NOVAFIT a Fitness App” will allow all trainee to signup ,login and book the session and subscription.
- The Trainer can also register and login and be a part of NovaFit family.
- It was a learner project in the company, but its scope is infinitely vast, and it’s very useable in the real life.
- The database is greatly capable of handling multiple concurrent connections/queries and process vast amount of data in no time. It also has capabilities of actions like sharding to further improve the performance.

## **Bibliography**

- [1] <https://owasp.org/www-chapter-belgium/assets/2021/2021-02-18/JWT-Security.pdf>
- [2] Dr Pulkit Sharma, “Login System for Web: Session Management using bcryptjs”, International Research Journal of Engineering and Technology (IRJET) for research analysis, Volume-8, Issue-11, November- 2019, ISSN No 2395-0056
- [3] React Router Author(s): Sagar Ganatra
- [4] “[https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express\\_Nodejs](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs)
- [5] React Tutorials - [https://www.w3schools.com/REACT/react\\_intro.asp](https://www.w3schools.com/REACT/react_intro.asp)
- [6] javascript Tutorials – <https://javascript.info/>
- [7] Node js Tutorial- [https://www.w3schools.com/nodejs/nodejs\\_intro.asp](https://www.w3schools.com/nodejs/nodejs_intro.asp)

Thank You