

AN INDUSTRIAL TRAINING REPORT

At

Writo Education pvt ltd

Date - 25th April 2024 to 25th June 2024

*SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR
THE AWARD OF THE DEGREE OF*

***BACHELOR OF ENGINEERING
(Computer Science)***



Submitted By:

Ritesh Gangwar

UE213075

7th Semester

To

Department of Computer Science Engineering

University Institute of Engineering and Technology

Panjab University, Chandigarh

December, 2024

Sno	Content	Pg-no
	Certificate	2
	Acknowledgment	3
Chapter 1	1. Introduction 1.1 Introduction to the industry/Organization	4
Chapter 2	The technology used	5-6
Chapter 3	Project Details including Snapshots and results/outcome	7-13
Chapter 4	Conclusion	14
Chapter 5	Bibliography	15

CERTIFICATE

Ref No: 2023BKD/3979025



CERTIFICATE OF INTERNSHIP

This certificate is proudly awarded to

Ritesh Gangwar

**For successfully completing his time as an Intern for Role: Full Stack Developer
Intern From 25/04/2024 - 25/06/2024 at WRITO EDUCATION PRIVATE LIMITED.**

CIN:
U80300MP2021PTC058550
Writo Head Office, MP-482005
Contact: 
Support@writo.tech

Omprakash Singh
Founder
Writo Education Pvt. Ltd.

Acknowledgement

I would like to express my sincere gratitude to the entire team at Writo Education pvt ltd, where I had the opportunity to work as a Full Stack Developer. My internship provided me with valuable insights into the world of web development, and I am deeply thankful for the support and guidance I received throughout the experience.

Special Thanks:

- OmPrakash, who guided me through the technical aspects of frontend development, helping me understand the nuances of building user-friendly, responsive web applications.
- The Development Team, for being incredibly supportive and collaborative, creating an enriching environment for learning and growth.

During my time at Writo Education pvt ltd,

=> Developed a secure service to store and retrieve users' payment data via QR code, integrated into the admin dashboard for efficient tracking.

=> Built a registration module featuring Email OTP-based user authentication, strengthening security and streamlining user onboarding.

=> Engineered a real-time chat solution for one-on-one doubt resolution, facilitating effective user support

I am grateful for the opportunity to have been part of this exciting venture, and I look forward to applying the lessons I learned in my future endeavors.

Chapter - 1

Introduction:- 1.1 Introduction to the Industry:-

Writo Tech is a premier technology solutions provider dedicated to transforming the digital landscape through innovative and comprehensive services. We specialize in web development, social media management, education sector solutions, and IT services, ensuring our clients receive tailored solutions that drive success.

Mission

At Writo Tech, our mission is to revolutionize the digital world by delivering cutting-edge technology solutions. We strive to empower businesses, educational institutions, and entrepreneurs with innovative tools, seamless digital experiences, and strategic insights to help them achieve their goals efficiently and effectively.

Chapter - 2

The Technology Used:

The website for this project uses a modern tech stack to provide a fast, reliable, and userfriendly experience for its clients. The technologies used in the development of the platform include:

Node.js:

- **Purpose:** Node.js is a JavaScript runtime that allows developers to build scalable and efficient server-side applications. It is known for its event-driven and non-blocking I/O model, making it ideal for real-time applications.
- **Usage in Project:** Node.js powers the backend of the application, handling API requests, managing WebSocket connections for real-time communication, and facilitating data processing. It ensures smooth interaction between the client and server, enabling real-time features like video calls and user authentication.

Express.js:

- **Purpose:** Express.js is a lightweight web framework for Node.js that simplifies the process of building robust and scalable web applications. It provides a flexible structure for handling routes, middleware, and API endpoints.
- **Usage in Project:** Express.js is used to build the backend APIs that handle user authentication. It also manages WebSocket connections for real-time communication, ensuring efficient data flow between users.

MongoDB:

- **Purpose:** MongoDB is a NoSQL database that stores data in a flexible, JSON-like format. It is designed for scalability and high performance, making it suitable for applications requiring dynamic and unstructured data storage.

- **Usage in Project:** MongoDB is used to store user information, meeting details, and chat messages. It ensures efficient data retrieval and storage, allowing users to join meetings, track meeting history, and manage chat interactions seamlessly.

React.js:

- **Purpose:** React.js is a front-end library for building interactive and dynamic user interfaces. It enables developers to create reusable UI components and manage application state efficiently.
- **Usage in Project:** React.js is used to develop the front-end components of the application, including the home page, meeting interface, and chat functionality. It provides a seamless user experience by handling real-time updates, UI interactions..

JSX:

- **Purpose:** JSX is a syntax extension for JavaScript that is used with React to describe the UI. It allows you to write HTML-like code directly within JavaScript.
- **Usage in Project:** In this project, JSX is used to create the user interface components, making the code more readable and easier to write. It simplifies the process of designing and building the structure of the web pages, such as the home page and meeting interface.

CSS:

- **Purpose:** CSS is used to style the visual appearance of the web application. It controls the layout, colors, fonts, and overall design to ensure a user-friendly and appealing interface.

- Usage in Project: CSS is utilized to style the home page and meeting components, providing a responsive and visually consistent design across different devices. It plays a crucial role in enhancing the user experience by ensuring that the UI elements are well-organized and aesthetically pleasing.

Javascript:

- Purpose: JavaScript is a core programming language for web development that allows you to create interactive and dynamic web applications.
- Usage in Project: It powers the logic for handling video and audio streams, user interactions, and the communication between clients using WebRTC and WebSockets. JavaScript is essential for implementing the real-time features of the video call, like toggling video/audio and managing peer-to-peer connections.

Web Sockets:

- Purpose: WebSockets provide a full-duplex communication channel over a single TCP connection, allowing data to be exchanged between the server and clients in real-time.

Chapter - 3

Project Details including snapshots and Results/outcome

Problem Statement

In today's digital landscape, businesses and individuals require **secure, efficient, and user-friendly** platforms to manage transactions, communication, and user interactions. However, many existing solutions face significant challenges, including:

1. **Lack of Secure Payment Management:** Many platforms struggle with safely storing and retrieving payment data, leading to inefficiencies and security concerns.
2. **Complex User Onboarding:** Traditional authentication methods can be cumbersome, making it difficult for users to register and access services securely.
3. **Limited Real-Time Support:** A lack of instant communication tools prevents users from resolving doubts and issues efficiently, impacting overall engagement and satisfaction.

Project Overview: Secure & Interactive Web Platform

To address these challenges, this project delivers a **comprehensive, secure, and interactive** web application that integrates key functionalities:

- **Secure Payment Data Management:** A robust service to store and retrieve payment data via **QR codes**, seamlessly integrated into an **admin dashboard** for real-time tracking and efficient transaction management.
- **Enhanced User Authentication:** A registration module with **Email OTP-based verification**, ensuring a **secure and streamlined onboarding** experience for users.
- **Real-Time Chat for Support:** A **one-on-one doubt resolution chat** feature, allowing users to get **instant assistance** and enhancing user engagement.

Snapshots:-

```
JS UserSchema.js X
MERN > Writo > Database > Models > JS UserSchema.js > ...
1  const mongoose = require("mongoose");
2  const bcrypt = require("bcryptjs");
3
4  const userSchema = mongoose.Schema(
5    {
6      name: { type: "String", required: true },
7      email: { type: "String", unique: true, required: true },
8      password: { type: "String", required: true },
9      pic: {
10       type: "String", //cloudinary url
11     },
12   },
13   { timestamps: true }
14 );
15
16 userSchema.methods.matchPassword = async function (enteredPassword) {
17   return await bcrypt.compare(enteredPassword, this.password);
18 };
19
20 userSchema.pre("save", async function (next) {
21   if (!this.isModified) {
```

```
JS otpSchema.js X
MERN > Writo > Database > Models > JS otpSchema.js > ...
1  const mongoose = require("mongoose");
2
3  const otpSchema = new mongoose.Schema({
4    email: {
5      type: String,
6      required: ["true", "email is required"],
7    },
8    otp: {
9      type: String,
10     required: ["true", "Otp is required"],
11   },
12   CreatedAt: Date,
13   expiresAt: Date,
14 });
15
16 const otpmodel = mongoose.model("otpmodel", otpSchema);
17
18 module.exports = { otpmodel };
19
```

```
JS ChatSchema.js X
MERN > Writo > Database > Models > JS ChatSchema.js > ...
1  const mongoose = require("mongoose");
2
3  const ChatSchema= mongoose.Schema(
4    {
5      sender:{ type: mongoose.Schema.Types.ObjectId, ref: "User" },
6      receiver:{ type: mongoose.Schema.Types.ObjectId, ref: "User" },
7    },
8    { timestamps: true }
9  );
10
11  const Chat = mongoose.model("Chat", ChatSchema);
12
13  module.exports = Chat;
14

MessageSchema.js X
MERN > Writo > Database > Models > JS MessageSchema.js > ...
1  const mongoose = require("mongoose");
2
3  const MessageSchema = mongoose.Schema(
4    {
5      sender: { type: mongoose.Schema.Types.ObjectId, ref: "User" },
6      content: { type: String, trim: true },
7      Pic: { type: String },//cloudinary url
8      chat: { type: mongoose.Schema.Types.ObjectId, ref: "Chat" },
9    },
10    { timestamps: true }
11  );
12
13  const Message = mongoose.model("Message", MessageSchema);
14  module.exports = Message;
15
```

JS userRoutes.js X



MERN > Writo > routes > JS userRoutes.js > ...

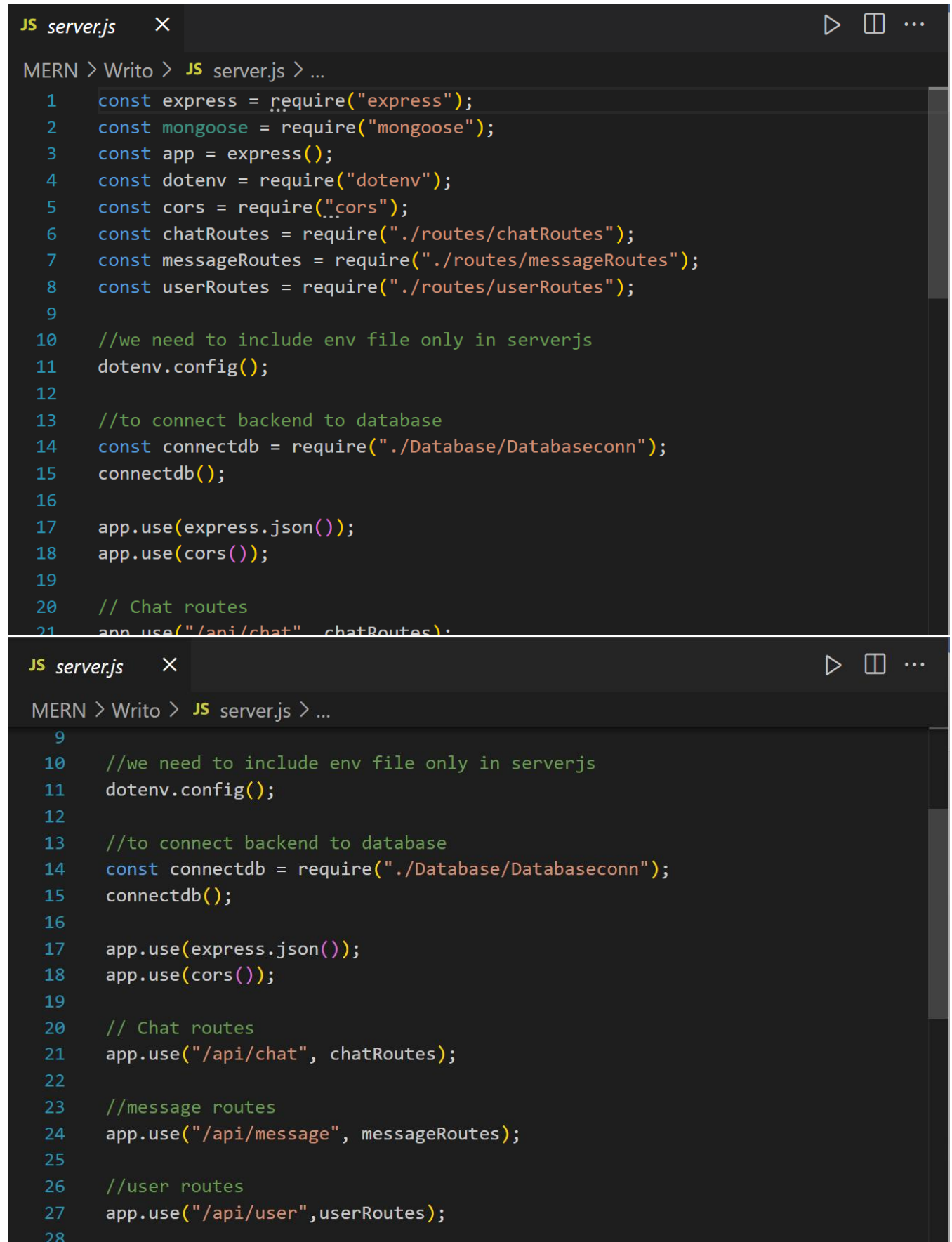
```
1  const express = require("express");
2  const { Login, Signup, verifyOtp } = require("../controllers/user");
3
4  const router = express.Router();
5
6  router.post("/register", Signup);
7  router.post("/login", Login);
8  router.post("/verifyOtp", verifyOtp);
9
10 module.exports = router;
11
```

JS message.js X



MERN > Writo > controllers > JS message.js > ...

```
1  const Message = require("../Database/Models/MessageSchema");
2
3
4  //API to get all the messages
5  const fetchMessages = async (req, res) => {
6
7      try {
8          const id = req.params.chatId;
9          //console.log(id);
10         const messages = await Message.find({ chat: id })
11             .populate("sender", "name pic email");
12
13         //console.log(messages);
14         res.json(messages);
15     } catch (error) {
16         res.status(400).json({ error: "No message found" });
17     }
18 };
19
20
```



```
JS server.js ×
MERN > Writo > JS server.js > ...
1  const express = require("express");
2  const mongoose = require("mongoose");
3  const app = express();
4  const dotenv = require("dotenv");
5  const cors = require("cors");
6  const chatRoutes = require("../routes/chatRoutes");
7  const messageRoutes = require("../routes/messageRoutes");
8  const userRoutes = require("../routes/userRoutes");
9
10 //we need to include env file only in serverjs
11 dotenv.config();
12
13 //to connect backend to database
14 const connectdb = require("../Database/Databaseconn");
15 connectdb();
16
17 app.use(express.json());
18 app.use(cors());
19
20 // Chat routes
21 app.use("/api/chat", chatRoutes);
22
23 //message routes
24 app.use("/api/message", messageRoutes);
25
26 //user routes
27 app.use("/api/user", userRoutes);
28
```

Results/Outcome:

- **User-Friendly Interface:** The application provides an intuitive and engaging user interface that allows users to easily create or join meetings, manage settings, and control video and audio streams.
- **Cross-Platform Compatibility:** Works seamlessly across different devices and browsers, allowing users to join meetings from desktops, laptops, tablets, or smartphones without compatibility issues.
- **Secure Payment Data Management:** Developed a secure service to store and retrieve users' payment data via QR code, integrated into the admin dashboard for efficient tracking.
- **Enhanced User Authentication:** Built a registration module featuring Email OTP-based user authentication, strengthening security and streamlining user onboarding.
- **Real-Time Chat for Support:** Engineered a real-time chat solution for one-on-one doubt resolution, facilitating effective user support.

Chapter - 4

Conclusion:

The development of this application marks a significant achievement in creating a secure, user-friendly, and versatile platform for communication, transactions, and support. Designed with cross-platform compatibility, it ensures seamless accessibility across desktops, laptops, tablets, and smartphones, making it convenient for users to connect from anywhere.

A key highlight of this project is its secure payment data management system, which enables efficient storage and retrieval of payment information via QR codes, integrated into an admin dashboard for streamlined tracking. Additionally, the enhanced user authentication feature, leveraging Email OTP-based registration, strengthens security and simplifies user onboarding.

To further improve user engagement and support, the platform incorporates a real-time chat solution for one-on-one doubt resolution, ensuring quick and effective assistance. Combined with an intuitive user interface, these features enhance the overall user experience, making the application both functional and accessible.

Moving forward, the platform sets the stage for future enhancements, such as advanced security measures, AI-driven user interactions, and deeper integrations with business tools. These improvements will continue to refine the application, ensuring it remains a scalable, reliable, and innovative solution for modern digital interactions.

Chapter -5

Bibliography

React.js

React. (n.d.). A JavaScript library for building user interfaces. Retrieved from React Official Site

Tailwind CSS Documentation : Retrieved from <https://tailwindcss.com/docs>

- Tailwind CSS's official guide to utility-first CSS framework, which was instrumental in creating responsive, mobile-first designs for the website.

GitHub Actions Documentation : Retrieved from <https://docs.github.com/en/actions>

- Documentation on setting up CI/CD pipelines, which automated deployment processes, reducing manual intervention.

Email.js Documentation : Retrieved from <https://www.emailjs.com/docs>

- A guide on integrating Email.js to handle form submissions, ensuring secure communication without requiring a back-end server.

Payment Gateway (Razorpay)

□ Socket.io. (n.d.).