GLOBAL NEXT CONSULTING INDIA LIMITED

FINAL PROJECT REPORT ON

“Global Connect Networking Platform”

Submitted by –

Lakshay ,

Muskan Shrivastava ,

Baishnvi Goswami ,

Suhanee Ahirwar ,

Rohit Sharma,

Akash Kumar

Under the Guidance of –

Mr. Vansh

Submitted to –

Mr. Vansh

Date of Submission – 19/08/2025

Acknowledgment

I sincerely thank my mentor Mr. Vansh, for their constant guidance, patience, and encouragement during every stage of this project. Their feedback and expertise were instrumental in shaping the idea and implementing it successfully.

I also extend my gratitude to my friends and colleagues for their moral support, as well as to Dr. Ambedkar Institute of Technology for Divyangjan for providing the necessary infrastructure.

DECLARATION

I, Lakshay , Muskan Shrivastava , Baishnvi Goswami , Suhani Ahirwar , Rohit Sharma , Akash Kumar , hereby declare that this project report entitled " Global Connect Networking Platform " is the result of my own work and that it has not been submitted to any other institution for any academic award.

All sources of information used have been duly acknowledged.

Signature:

Lakshay ,

Muskan Shrivastava ,

Baishnvi Goswami ,

Suhanee Ahirwar ,

Rohit Sharma ,

Akash Kumar

ABSTRACT

The Global Connect Networking Platform is designed as a **professional networking system that bridges global opportunities.** Unlike existing platforms, it emphasizes affordability, inclusivity, and collaboration. Built using the **MERN** stack **(MongoDB, Express.js, React.js, Node.js)**, it provides modules for user authentication, profile management, job postings, messaging, and AI-based recommendations.

- **Real-time messaging** for instant communication.
- **Job board & recruitment tools** for professional growth.
- **Alumni-student networking** for guidance and mentorship.
- **Social features** like posts, likes, comments, and shares.

The platform addresses gaps in current systems by offering multilingual support, real-time communication, and global reach. The project outcome is a prototype that demonstrates the feasibility of creating a scalable, affordable, and secure global networking platform.

The system is **scalable, secure, and responsive**, making it suitable for web and future mobile applications.

TABLE OF CONTENTS

22. References

INTRODUCTION

Professional networking is essential in today's digital world. Platforms like LinkedIn have transformed the way professionals connect, but challenges remain: premium paywalls, lack of inclusivity, and limited collaboration.

Networking plays a vital role in both personal and professional growth. While platforms like LinkedIn focus on careers and Facebook emphasizes social interaction, there remains a gap for a unified, community-driven space that offers both.

Global Connect aims to bridge that gap, especially for students, alumni, and industry professionals. The platform enables users to share updates, discover job opportunities, connect with mentors, and communicate instantly — all within a single hub.

Global Connect Networking Platform is proposed as a comprehensive system that enables users worldwide to:

Build professional profiles

Connect with recruiters and peers

Apply for jobs

Communicate in real-time

Receive AI-based job and connection recommendations

This project demonstrates a scalable, cost-effective solution for professional networking that can be adopted globally.

PROBLEM STATEMENT

Existing platforms force users to juggle multiple apps for different needs:

High Premium Costs: Advanced features locked behind paid plans.

Limited Inclusivity: Small businesses and students often struggle to use premium services.

Minimal Collaboration: Mostly restricted to posts and simple messaging.

Lack of Regional Focus: Poor multilingual and local job support.

- **Career growth** – LinkedIn, Naukri
- **Social interaction** – Facebook, Instagram
- **Real-time chat** – WhatsApp, Slack

Therefore, there is a need for a global, inclusive networking platform that balances networking, collaboration, affordability, and accessibility.

OBJECTIVES

1. To design and develop a global professional networking platform.
2. To provide modules for profile creation, networking, job posting, and messaging.
3. To implement AI-driven recommendation systems for personalized job and connection suggestions.
4. Unified platform combining career, social, and real-time networking.
5. Secure authentication to protect user data.
6. Post and share opportunities easily within the network.
7. Real-time chat and notifications for instant engagement.
8. Job portal integrated with the community to post and apply for roles.
9. Role-based access so admins can moderate content and manage users.
10. To ensure system scalability, security, and usability.
11. To support global accessibility with multilingual support in future.

BACKGROUND STUDY

Research into similar platforms revealed gaps:

- LinkedIn has strong job networking but lacks casual community engagement.
- Facebook has excellent social engagement but no job-matching features.
- Slack is great for communication but lacks a social feed and public posts.

MERN Stack was chosen because:

- MongoDB – Flexible schema for rapid development.

• Express.js – Robust server framework.

• React.js – Interactive and fast UI.

• Node.js – Event-driven, scalable backend

SYSTEM ARCHITECTURE

[User Device]

|

[React.js Frontend]

|

[REST API (Express.js)] <-> [MongoDB Atlas Database]

|

[Socket.io Server for Real-Time Communication]

|

[Cloudinary for Media Storage]

TOOLS AND TECHNOLOGIES USED

Frontend: React.js, TailwindCSS, Axios, React Router

Backend: Node.js, Express.js

Database: MongoDB Atlas

Real-time: Socket.io

Authentication: JWT, bcrypt.js

Hosting: Render(Frontend), Render (Backend)

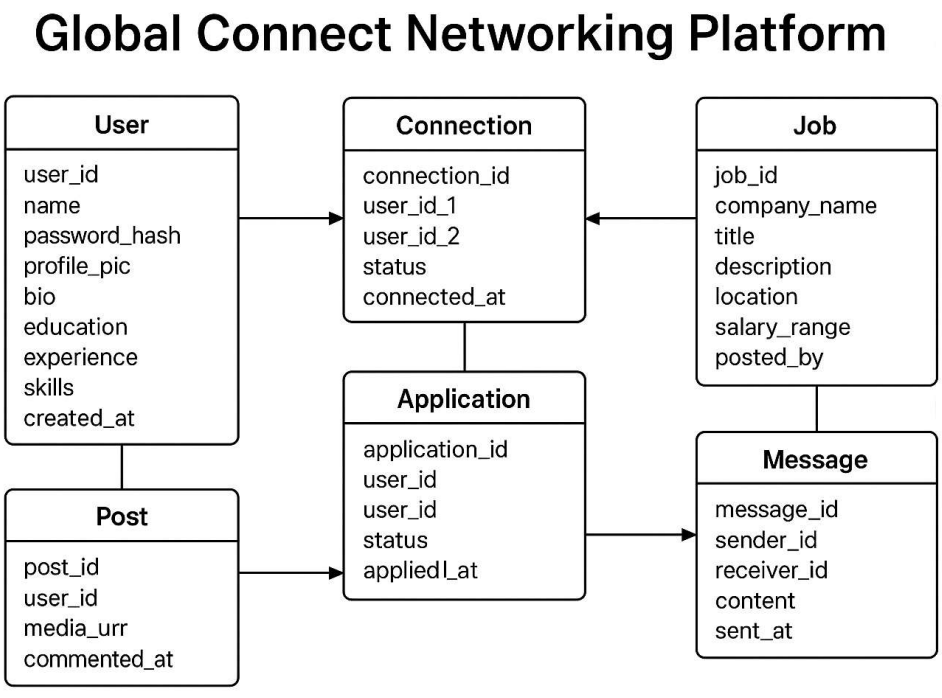
Version Control: Git, GitHub

Media Storage: Cloudinary

MODELS/ FEATURES DESCRIPTION

1. **Authentication & Authorization –** Secure signup/login, password hashing.
2. **Profile Management –** Upload photo, add skills, education, experience.
3. **Post Creation –** Add images, like, comment, share.
4. **Job Board –** Post jobs, apply, track applications.
5. **Chat System –** One-to-one messaging, read receipts, online status.

DATABASE DESIGN



FRONTEND & BACKEND OVERVIEW

Frontend -

The frontend of Global Connect is designed for intuitive navigation, responsive layouts, and accessibility across devices. Built with React.js in the MERN stack, it follows a component-based architecture for reusability and maintainability.

Key Features:

• **Component-Based Design:** Reusable components like Navbar, Sidebar, PostFeed, JobList, ChatWindow.

• **Responsive UI/UX:** Mobile-first layouts using TailwindCSS

• **State Management:** Redux ensures predictable and efficient data handling.

• **Routing:** React Router DOM enables smooth navigation between Home, Jobs, Notifications , and Profile.

• **API Integration:** Secure Axios calls with JSON for backend communication.

• **Real-Time Features:** Socket.io for instant messaging, notifications, and AI chat.

• **Form Handling:** React Form for validation and clean data submission.

Benefits:

• Modular structure for faster development and easy maintenance.

• Engaging, accessible interface.

Backend-

The backend of Global Connect is built to be robust, secure, and scalable, using Node.js and Express.js as the server-side foundation of the MERN stack.

Key Features

• **RESTful API:** Handles CRUD operations for users, posts, jobs, messages, notifications, and comments.

• **Database Layer:** MongoDB for flexible NoSQL storage, ensuring scalability and performance.

• **Data Modeling:** Mongoose for schema definitions, validation, and indexing.

• **Authentication & Authorization:** JWT for secure, role-based access to resources.

• **Real-Time Communication**: Socket.io for instant messaging, live notifications, and AI chat.

• **Error Handling:** Centralized middleware for consistent and clear responses.

• Scalability: Supports horizontal scaling and is microservices-ready for future expansion.

Benefits:

• Clear separation of concerns for maintainability.

• Secure, optimized architecture for high traffic loads.

IMPLEMENTATION

The **Global Connect Networking Platform** project is an AI-powered networking platform built using the **MERN stack** (MongoDB, Express.js, React.js, Node.js) in a modular, phased approach.

Backend: Developed first with **Node.js and Express.js,** it includes RESTful APIs for authentication, posts, jobs, messages, notifications, and AI assistance. JWT authentication with password encryption via bcrypt.js ensures security, while MongoDB with Mongoose handles data modeling and validation. Socket.io enables real-time chat, notifications, and AI-driven interactions. Google Gemini 2.5 is integrated via a dedicated API endpoint to power the AIChat Assistant, providing intelligent guidance, career tips, and personalized recommendations. The backend is deployed on Render, with the database hosted on MongoDB Atlas.

Frontend: Built with **React.js** and a **component-based architecture**, it uses React Router DOM for navigation, Redux for state management, and Axios for API integration. Forms are handled with React Form, while Tailwind CSS ensures a modern, fully responsive design. The frontend is deployed on Render.

Outcome: The platform underwent thorough testing using Postman for API validation and manual checks for UI responsiveness. The result is a fully functional, secure, responsive, and AI-driven platform that enables global user connectivity, job applications, real-time messaging, and intelligent networking.

TEST CASES & RESULTS

The following basic tests were conducted to verify the main features of Global Connect.

Test Action	Expected Result	Result
User Registration	Account created successfully	Pass
User Login	Redirect to dashboard	Pass
Create Post	Post appears in feed	Pass
Apply for Job	Application recorded	Pass
Send Message	Message delivered instantly	Pass
Responsive UI	Layout adjusts properly on mobile	Pass

DEPLOYMENT PROCESS

- **Frontend:** Built React app and hosted on **Render.**
- **Backend:** Hosted Node.js/Express API on **Render.**
- **Database:** Used **MongoDB Atlas** for cloud storage.
- Linked all services, set environment variables, and tested the live site.

LIMITATIONS

- Limited scalability for very high traffic without further optimization.
- No offline access to application features.
- Real-time features may depend on internet speed.
- Limited advanced search and filtering options.
- Basic UI design; not fully customized for all device sizes.

FUTURE ENHANCEMENTS

- Add advanced search and filtering for posts and jobs.
- Improve UI/UX with custom design and animations.
- Implement mobile app version for Android/iOS.
- Add multilingual support.
- Integrate video/audio calls in chat.
- Launch mobile apps (Android/iOS).
- Introduce advanced AI-driven job matching.
- Add video conferencing, webinars, and networking events.
- Provide multilingual support for regional accessibility.
- Introduce premium model at affordable pricing.

CONCLUSION

The **Global Connect Networking Platform** project was developed with the primary aim of creating a platform where users can connect globally, share posts, apply for jobs, and communicate in real-time. Using the **MERN stack**—MongoDB, Express.js, React.js, and Node.js—the project integrates both frontend and backend functionalities seamlessly, providing a smooth and secure user experience.

Throughout the development process, emphasis was placed on building a responsive design, secure authentication, and reliable data management. The system architecture allows scalability and modular expansion, ensuring that future improvements can be incorporated without major restructuring. Real-time features such as messaging, notifications, and AI-driven interactions enhance user engagement, while job application and post creation functionalities make the platform valuable for both professional and social networking.

Testing confirmed that all major features work as intended, and deployment on **Render** makes the application accessible worldwide. While the current version meets its core objectives, it also opens avenues for further improvements such as mobile app integration, advanced search capabilities, and multilingual support.

In conclusion, **Global Connect Networking Platform** is a functional, user-friendly, and scalable platform that fulfills its goal of enabling global connectivity and collaboration. With future enhancements, it has the potential to become a comprehensive networking solution for individuals and organizations worldwide.

APPENDICES

Code Snapshorts:

MODELS

Models User.js

```
const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  bio: { type: String, default: '' },
  skills: { type: [String], default: [] },
  location: { type: String, default: '' },
  profilePic: { type: String, default: 'https://tse3.mm.bing.net/th/id/OIP.1waDZ8Q2eWBkenMscI08qAHaHa?pid=Api&P=0&h=1
  company: { type: String, default: '' },
  connections: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }],
  pendingRequests: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }]
}, { timestamps: true });

const educationSchema = new mongoose.Schema({
  school: String,
  degree: String,
  fieldOfStudy: String,
  startDate: Date,
  endDate: Date,
  description: String
});
const experienceSchema = new mongoose.Schema({
  company: String,
  position: String,
  startDate: Date,
  endDate: Date,
  description: String
});
```

Post.js

```
  content: {
    type: String,
    required: true
  },
  image: {
    type: String
  },
  likes: [
    {
      user: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "User"
      }
    }
  ],
  comments: [
    {
      user: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "User"
      },
      text: {
        type: String,
        required: true
      },
      date: {
        type: Date,
        default: Date.now
      }
    }
  ]
}, { timestamps: true });

module.exports = mongoose.model("Post", postSchema);
```

Message .js

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

const messageSchema = new mongoose.Schema({
  sender: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "User",
    required: true
  },
  receiver: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "User",
    required: true
  },
  message: {
    type: String
  },
  image: {
    type: String // For sending image messages
  },
  seen: {
    type: Boolean,
    default: false
  }
}, { timestamps: true });

module.exports = mongoose.model("Message", messageSchema);
```

Job.js

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

const jobSchema = new mongoose.Schema({
  title: String,
  description: String,
  skills: [String],
  location: String,
  postedBy: { type: mongoose.Schema.Types.ObjectId, ref: "User" },
  applicants: [{ type: mongoose.Schema.Types.ObjectId, ref: "User" }],
  savedBy: [{ type: mongoose.Schema.Types.ObjectId, ref: "User" }],
  status: { type: String, default: "open" } // open, closed
}, { timestamps: true });

module.exports = mongoose.model("Job", jobSchema);
```

Authmiddleware.js

```
const jwt = require('jsonwebtoken');
const User = require('../models/User');

const protect = async (req, res, next) => {
  let token = req.headers.authorization?.split(' ')[1];

  if (!token) {
    return res.status(401).json({ message: 'Not authorized, no token' });
  }

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = await User.findById(decoded.id).select('-password');
    next();
  } catch (error) {
    res.status(401).json({ message: 'Token failed' });
  }
};

module.exports = protect;
```

Upload cloudinary

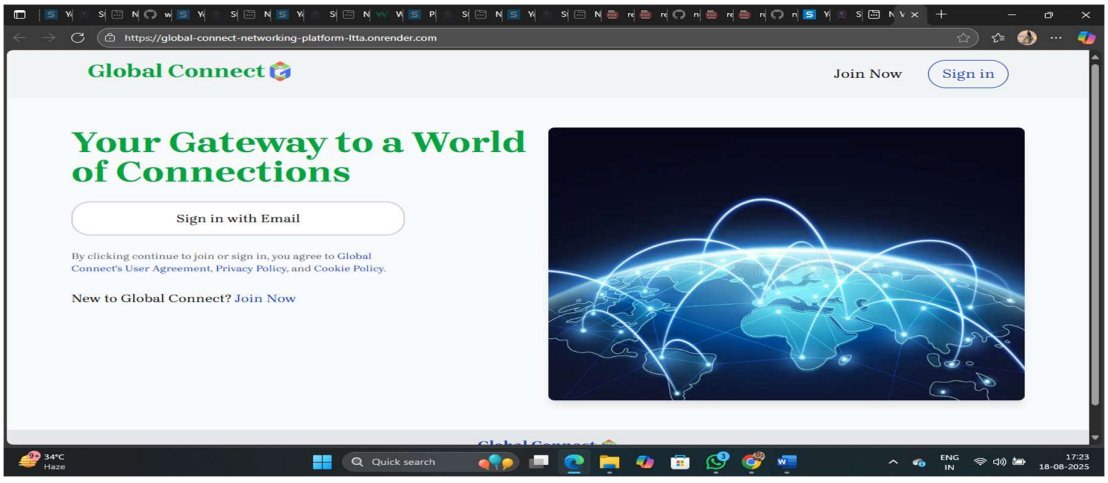
```
const multer = require('multer');
const path = require('path');
const storage = multer.diskStorage({
    destination: function (req, file, callback) {
        callback(null, 'uploads/'); //
    },
    filename: function (req, file, callback) {
        callback(null, Date.now + path.extname(file.originalname))

    }
})


const upload = multer({ storage })

module.exports = upload
```

socket io

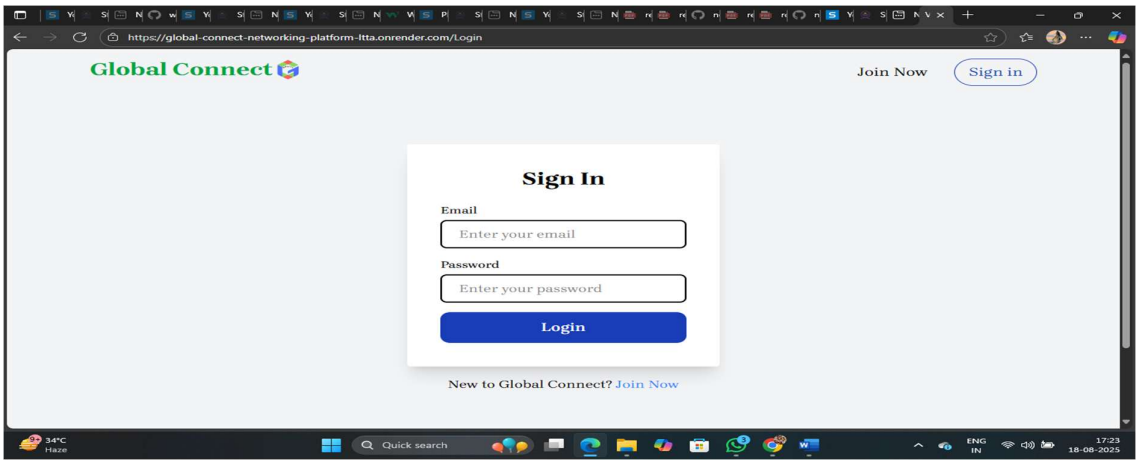
```
const {Server} = require("socket.io")
const http = require("http")
const express = require('express');

// Create HTTP server
const app = express();
const httpServer = http.createServer(app);

// Create Socket.IO server and attach it to HTTP server
const io = new Server(httpServer);

// get socket id from map
const getReceiverSocketId = (userId) => {
    return userSocketMap[userId];
}

//used to store online users
const userSocketMap = {}; // {userId: socketId}

// Handle socket connections
io.on("connection", (socket) => {
  console.log("A user connected:", socket.id);

const userId = socket.handshake.query.userId;
if(userId)  userSocketMap[userId] = socket.id

//io.emit() is used to send events to all the connected clients
io.emit("getOnlineUsers" , Object.keys(userSocketMap));

  socket.on("disconnect", () => {
    console.log("User disconnected:", socket.id);
    io.emit("getOnlineUsers" , Object.keys(userSocketMap));
  });
});

module.exports =  {io, app, httpServer,getReceiverSocketId};
```

SCREENSHOTS

Global connect

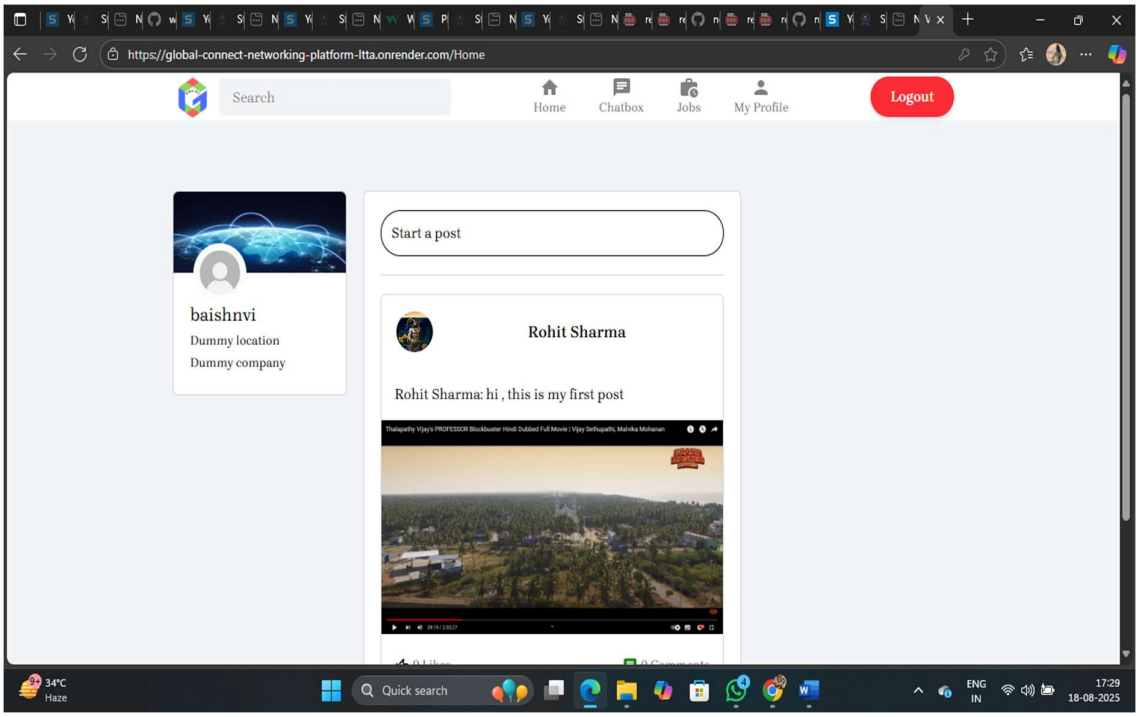


Sign In

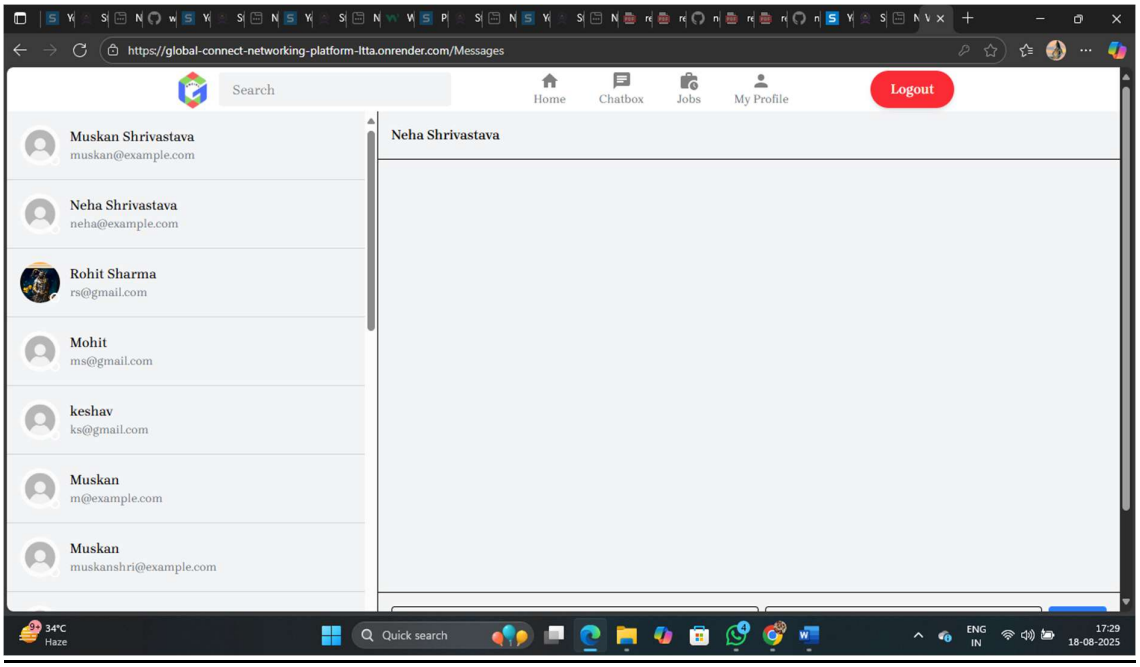


Home

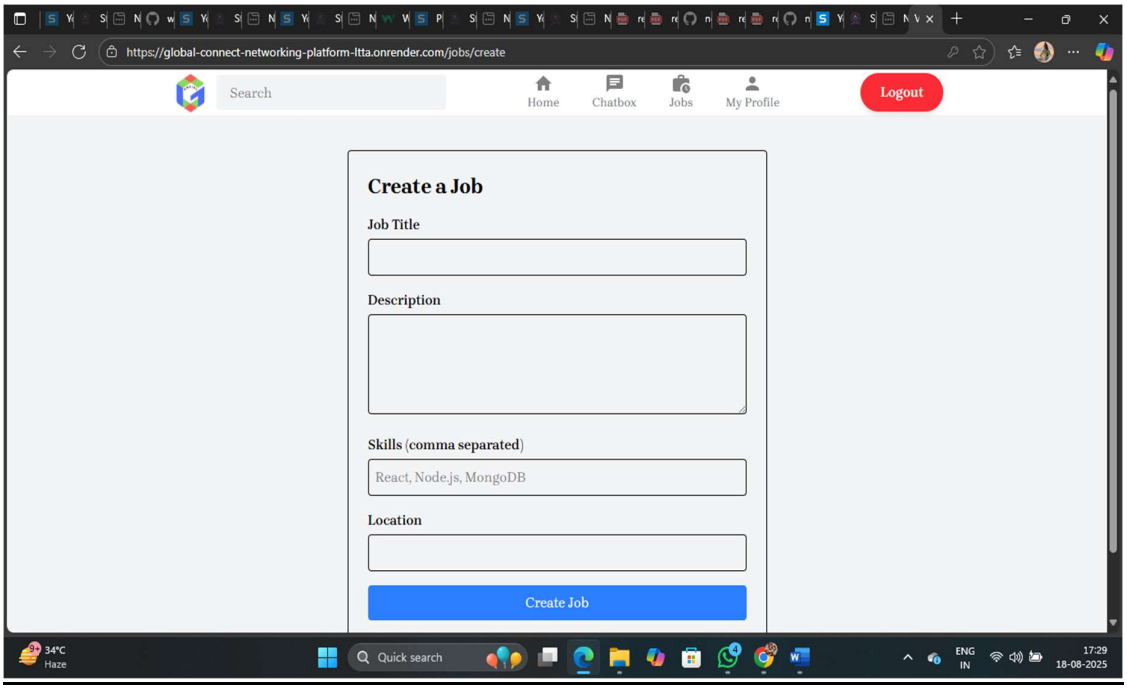
Post


https://global-connect-networking-platform-ltta.onrender.com/Home
Search
Home
Chatbox
Jobs
My Profile
Logout
baishnvi
Dummy location
Dummy company
Start a post
Rohit Sharma
Rohit Sharma: hi , this is my first post
34°C
Haze
Quick search
ENG
IN
17:29
18-08-2025

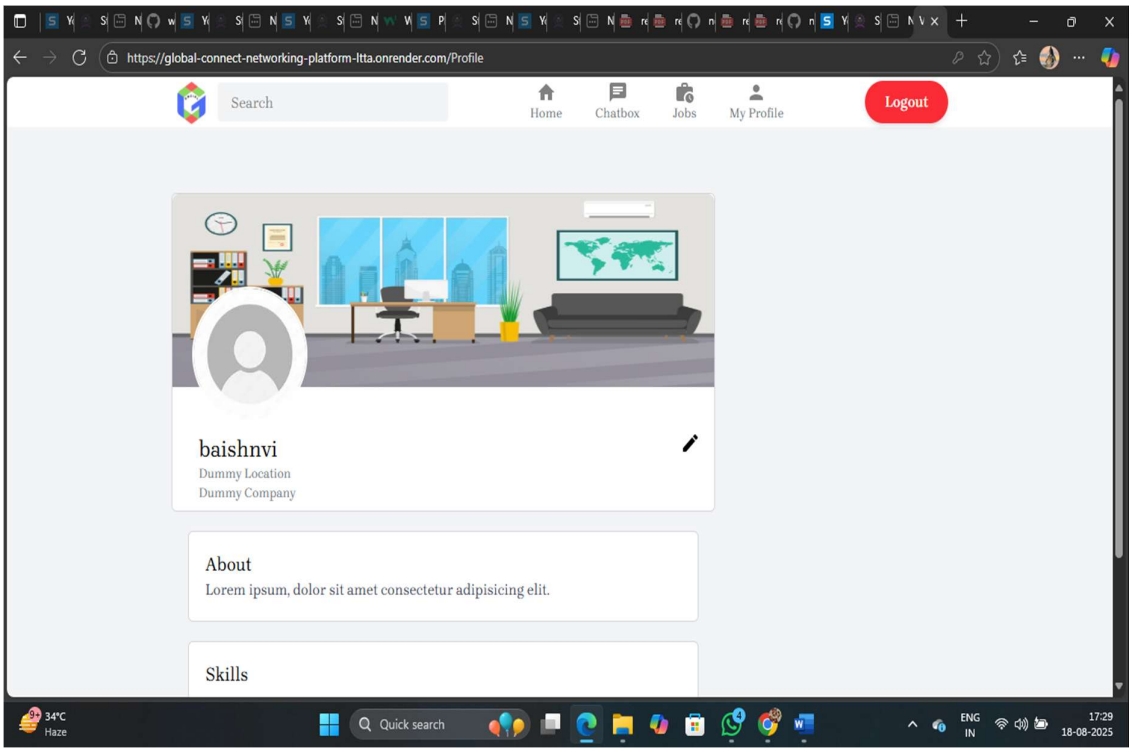
Chatbox


https://global-connect-networking-platform-ltta.onrender.com/Messages
Search
Home
Chatbox
Jobs
My Profile
Logout
Muskan Shrivastava
muskan@example.com
Neha Shrivastava
neha@example.com
Rohit Sharma
rs@gmail.com
Mohit
ms@gmail.com
keshav
ks@gmail.com
Muskan
m@example.com
Muskan
muskanshri@example.com
Neha Shrivastava
34°C
Haze
Quick search
ENG
IN
17:29
18-08-2025

Job



Profile



REFERENCES

- React.js Documentation – https://react.dev
- Node.js Documentation – https://nodejs.org
- Express.js Documentation – https://expressjs.com
- MongoDB Documentation – https://www.mongodb.com/docs
- Socket.io Documentation – https://socket.io/docs
- MDN Web Docs – https://developer.mozilla.org
- W3Schools – https://www.w3schools.com