DOCKERIZE A FRONT-END WEB APP, A BACK-END API, AND A DATABASE COMPONENT. DEPLOY THESE CONTAINERS USING DOCKER COMPOSE OR KUBERNETES.

Enrollment No.s: 21103117, 21103123, 21103131

Name of Students: Vaidik Parashar, Aanvi Varma, Avni Aggarwal

Name of Supervisor: Amarjeet Kaur

December-2023

Submitted in partial full fulfillment of the Degree of

Bachelor of Technology

in

Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

TABLE OF CONTENTS

Chapter No. Topics **Page No.**

Self Declaration

I / We hereby declare the following usage of the open source code and prebuilt libraries in our minor project in 5th Semester with the consent of our supervisor. We also measure the similarity percentage of pre written source code and our source code and the same is mentioned below. This measurement is true with the best of our knowledge and abilities.

1. List of pre build libraries
2. List of pre build features in libraries or in source code.
3. Percentage of pre written source code and source written by us.

Student ID	Student Name	Student Signature
21103117	Vaidik Parashar	
21103123	Aanvi Varma	
21103131	Avni Aggarwal	

Declaration by Supervisor (To be filled by Supervisor only)

I, ……..(Name of Supervisor) declares that I above submitted project with Titled...DOCKERIZE A FRONT-END WEB APP, A BACK-END API, AND A DATABASE COMPONENT.

DEPLOY THESE CONTAINERS USING DOCKER COMPOSE OR KUBERNETES was conducted in my supervision. The project is original and neither the project was copied from External sources not it was submitted earlier in JIIT. I authenticate this project.

(Any Remarks by Supervisor)

Signature (Supervisor)

CERTIFICATE

This is to certify that the work titled DOCKERIZE A FRONT-END WEB APP, A BACK-END API, AND A DATABASE COMPONENT. DEPLOY THESE CONTAINERS USING DOCKER COMPOSE OR KUBERNETES submitted by ..... in partial fulfillment for the award of degree of (Name of programme e.g. B. Tech, M. Tech etc.)..... …….. ……of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor

Name of Supervisor

Designation

Date

ACKNOWLEDGEMENT

Signature of Student:

Name of Student: **Vaidik Parashar**

Enrollment number: **21103117**

Signature of Student:

Name of Student: **Aanvi Varma**

Enrollment number: **21103123**

Signature of Student:

Name of Student: **Avni Aggarwal**

Enrollment number: **21103131**

Date:

SUMMARY

Our project is a comprehensive website dedicated to addressing substance abuse issues. Anchored by a robust blog section utilizing APIs, the platform facilitates user connection through shared experiences. We've seamlessly integrated a volunteer form, empowering individuals to actively contribute and support others. This dynamic and inclusive space aims to break down barriers, foster a sense of community, and provide valuable resources for those navigating the challenges of substance abuse. Our ultimate goal is to inspire positive change, resilience, and a sense of belonging among users, creating a lasting impact on their journeys to recovery.

CHAPTER 1: INTRODUCTION

1.1 GENERAL INTRODUCTION

Welcome to a community like no other, where individuals with a shared curiosity and a commitment to mindful substance exploration come together to connect, learn, and support one another. We proudly introduce Resolute, a social media platform created with the understanding that responsible choices and informed decisions are at the heart of a balanced approach to substances.

Resolute is your safe haven—a place where stigma is replaced with understanding, judgment with empathy, and isolation with a warm, supportive community. We recognize that the spectrum of substance experiences is vast, and we're here to embrace it all, from the therapeutic potential of plant medicines to the mindful use of recreational substances.

Our mission is clear: to provide a platform where you can engage in thoughtful conversations, access reliable information, and connect with like-minded individuals who share your values and interests. We're not just a social media site; we're a movement promoting responsible exploration and harm reduction.

So, whether you're a seasoned explorer, a curious novice, or someone in search of guidance, Resolute is your digital sanctuary, a place to be authentic, share your experiences, and seek guidance from a compassionate and knowledgeable community. Welcome to a world where your journey is celebrated, your questions are answered, and your voice is heard.

Together, we are Resolute—united in our commitment to responsible choices, informed decisions, and a supportive network that understands your path. Join us as we embark on this journey of exploration, empowerment, and enlightenment.

1.2 PROBLEM STATEMENT

DOCKERIZE A FRONT-END WEB APP, A BACK-END API, AND A DATABASE COMPONENT. DEPLOY THESE CONTAINERS USING DOCKER COMPOSE OR KUBERNETES

The prevalence of substance abuse continues to pose a significant public health challenge, impacting individuals and communities globally. Many affected individuals face a lack of accessible and supportive resources, hindering their journey to recovery. Stigma and a dearth of community-driven platforms exacerbate the isolation experienced by those dealing with substance abuse issues.

1.3 SIGNIFICANCE OF THE PROBLEM

Our project addresses the critical need for a comprehensive and inclusive platform to support individuals grappling with substance abuse. By providing a dynamic website with a curated blog section, we aim to break down barriers, foster community, and offer a valuable resource hub. The significance lies in creating a supportive virtual space where individuals can connect, share experiences, and access essential information.

1.4 EMPIRICAL STUDY

Survey on Substance Consumption: Understanding the Landscape

In our ongoing initiative to comprehend the landscape of substance consumption within our community, we designed a survey targeting individuals with diverse backgrounds. The survey delves into key demographic factors such as age, gender,

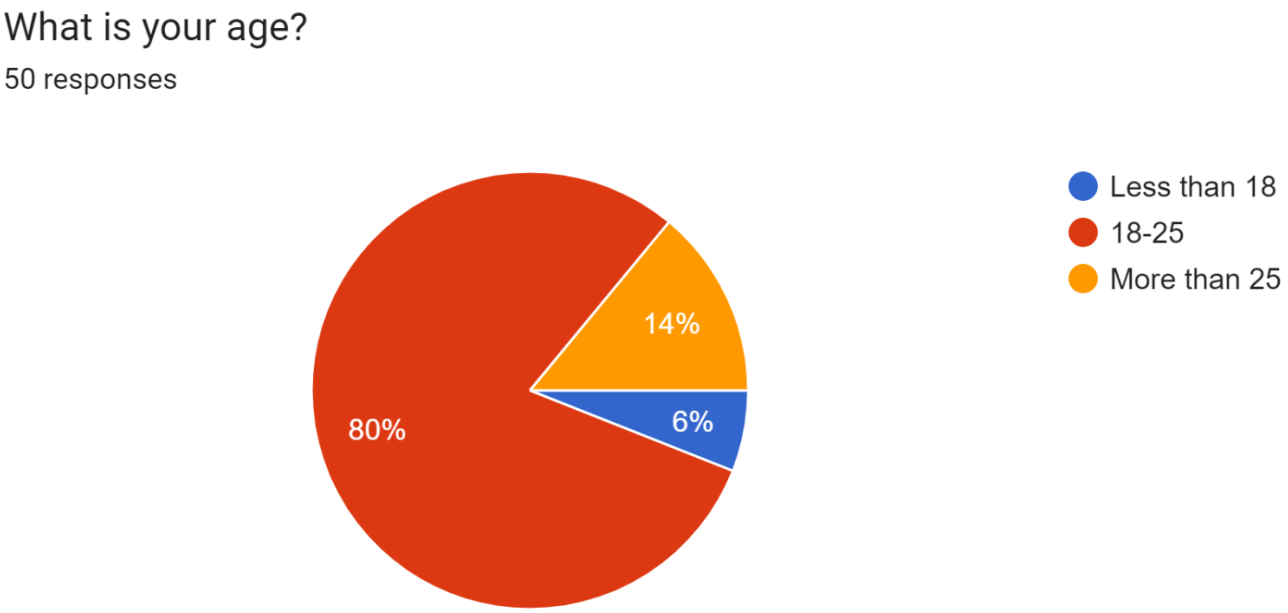
and employment status, providing a comprehensive snapshot of the varied profiles within our community.

Participants were candidly queried about the substances they engage with, the frequency of consumption, and their willingness to consider quitting. This multi-faceted approach aims to unravel the complexities of substance use patterns and the factors influencing them.

One pivotal aspect explored was the participants' perspectives on the role of motivation in the quitting process. Their insights into the significance of motivation serve as a crucial pillar in shaping strategies for support and intervention.

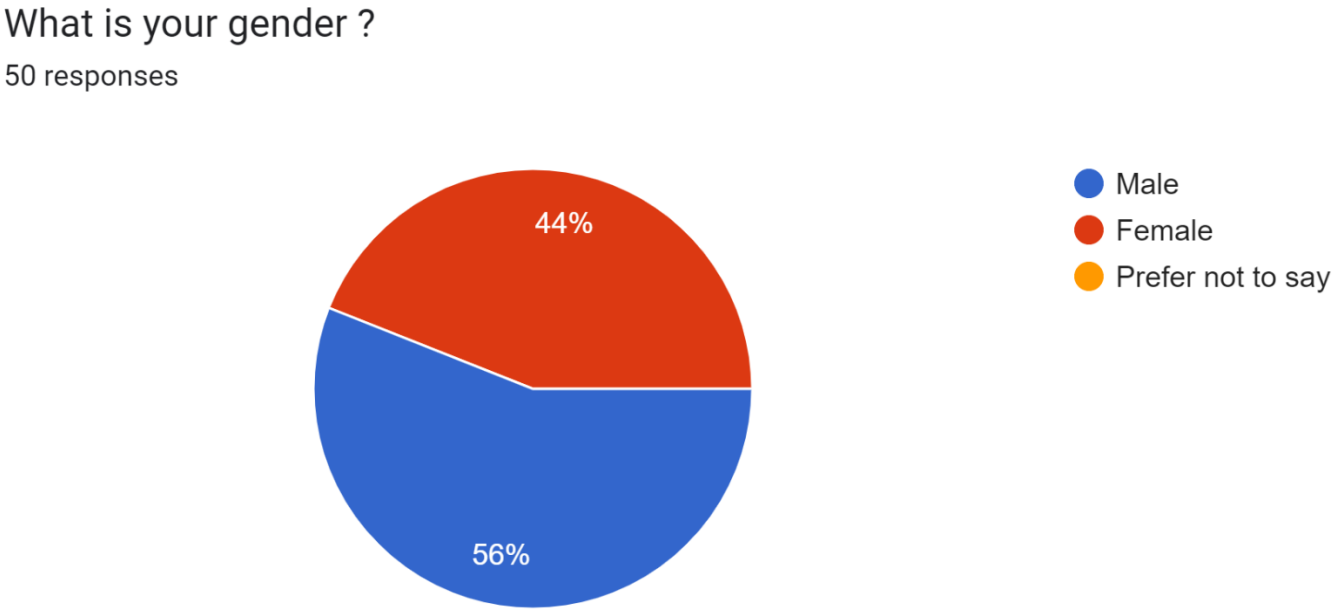
By gathering this nuanced data, we aspire to construct a more profound understanding of the challenges individuals face in combating substance consumption. This survey serves as a foundational step in tailoring our initiatives to better align with the needs and motivations of our community, ultimately fostering a more empathetic and effective approach to supporting those on the path to recovery.

In our recent survey focused on substance consumption, a predominant demographic emerged, with more than 80% of participants falling within the age range of 18-25. This age group represents a critical period of transition and self-discovery, and the prevalence of substance engagement in this cohort demands careful attention.



Gender Distribution:

The survey revealed a gender distribution of 56% males and 44% females. This gender balance emphasizes the need for inclusive and gender-sensitive approaches in developing strategies to address substance consumption within the community.



Substance Usage Patterns:

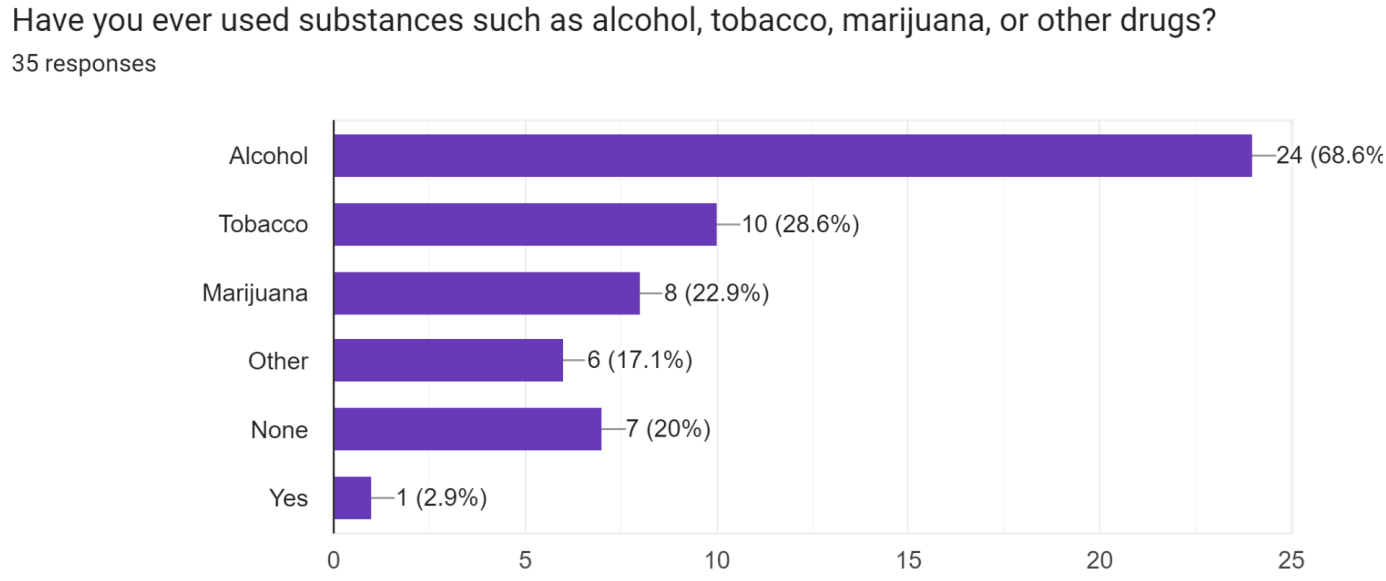
-Alcohol Consumption:A notable 68.6% of respondents acknowledged consuming alcohol. This high prevalence underscores the significance of alcohol-related interventions and education within the community.

- Tobacco Usage:Approximately 28.6% of participants reported tobacco consumption, indicating a concerning proportion engaging in this often addictive behavior.

- Marijuana Use: 22.9% of respondents admitted to using marijuana, shedding light on the prevalence of recreational drug use within the demographic.

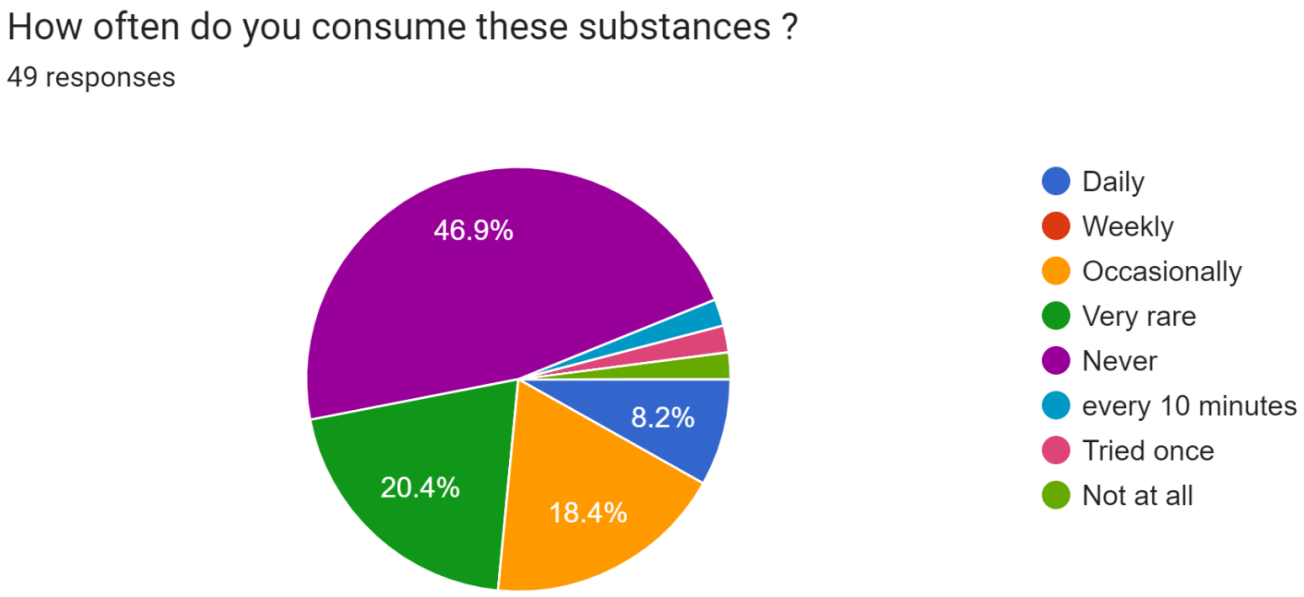
- Other Substances: 17.1% reported consuming substances other than alcohol, tobacco, or marijuana, reflecting the diversity of substances in circulation.

- Abstainers: Encouragingly, 20% of respondents indicated that they had not consumed any of the specified substances, highlighting a significant portion of the demographic making health-conscious decisions.



Consumption Frequency:

- Occasional Consumption: A substantial 18.4% reported occasional substance consumption, reflecting a non-regular but present engagement.
- Daily Consumption: 8.2% admitted to daily substance consumption, indicating a consistent and potentially problematic pattern.
- Frequent Consumption: A concerning 2% reported consuming substances every ten minutes, signifying a subset of individuals engaged in high-frequency, potentially addictive behavior.

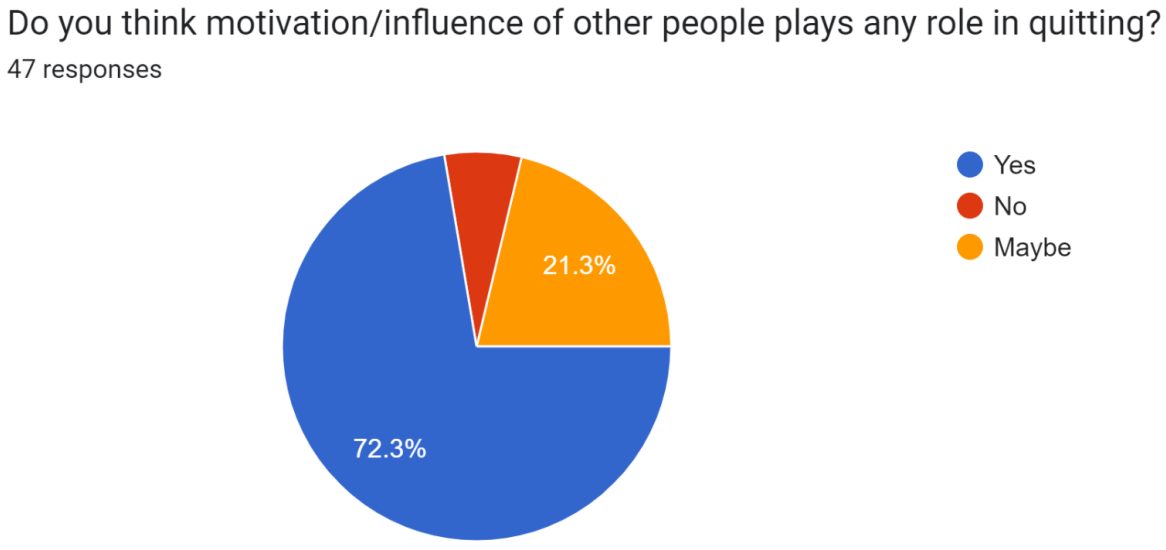


Willingness to Quit:

An essential aspect of our survey was gauging participants' willingness to quit. Preliminary findings suggest a need for targeted intervention strategies, with varying degrees of motivation observed among respondents.

Role of Motivation:

Interestingly, a substantial portion of respondents acknowledged the role of motivation in the quitting process. This insight underscores the importance of incorporating motivational elements in support programs and interventions, aligning with a more holistic approach to substance abuse recovery.



Conclusion:

This survey provides a comprehensive snapshot of substance consumption patterns within the 18-25 age group. The data obtained will serve as a foundation for tailored interventions, educational initiatives, and community-based support programs aimed at addressing the diverse needs of this demographic. As we move forward, it is imperative to leverage these insights to create targeted and effective strategies that foster a healthier and more informed community.

1.5 BRIEF DESCRIPTION OF SOLUTION APPROACH

The survey outcomes underscore a critical insight: a significant 72.6% of participants recognize the pivotal role of motivation in the process of quitting substance abuse. This understanding is not only encouraging but also reinforces the potential impact of our website in supporting individuals on their journey to recovery.

Key Takeaways:

Alignment with User Perspectives: The acknowledgment of motivation as a key factor aligns seamlessly with the ethos of our website. By emphasizing motivational content, success stories, and support networks, we can cater directly to the identified needs and beliefs of our target audience.

Empowering Through Community: The survey results highlight the community's recognition of the intrinsic link between motivation and quitting. Our website can serve as a dynamic hub, fostering a sense of community support where individuals can share motivational experiences, insights, and encouragement.

CHAPTER 2:REQUIREMENT ANALYSIS & SOLUTION APPROACH

2.1 OVERALL DESCRIPTION OF THE PROJECT

Our website is a user-centric platform designed to address the challenges related to substance abuse. Our website features an engaging blog section designed to connect users with shared experiences. Through the integration of APIs, we've curated a space where individuals can stay connected with like-minded people, fostering a sense of community and understanding. The blog section serves as a hub for shared stories, insights, and support.

Additionally, our commitment to inclusivity and collaboration is reflected in the incorporation of a volunteer form. This feature empowers those eager to make a positive impact by providing an avenue for users to contribute and extend help to others in need.

Leveraging MongoDB as the database deployed on MongoDB Atlas, we've prioritized flexibility in data management, allowing for the secure storage and retrieval of information critical to users on their journey to recovery.

To enhance deployment and scalability, we've implemented Docker, ensuring consistent and reliable performance across diverse environments. The result is a user-friendly, informative, and empathetic website that serves as a beacon of support for those seeking guidance and community in their pursuit of a substance-free life.

2.2 REQUIREMENT ANALYSIS

1. Functional Requirements:

1.1 Installation and Local Development:

- Users must have Node.js and Express installed on their local machines.
- The application should be runnable on localhost:3000 after the installation.

1.2 Containerization with Docker:

- The application must be containerized using Docker.
- The Docker container should encapsulate all necessary dependencies for seamless deployment.

1.3 Database Hosting on MongoDB Atlas:

- Application data should be stored on MongoDB Atlas.
- Connection details for MongoDB Atlas (URI, credentials) must be configurable.

1.4 Docker Desktop Dependency:

- Docker Desktop is required to run the containerized web application.

1.5 Deployment on AWS Cloud:

- The application should be deployable on AWS Cloud.
- AWS deployment should eliminate the need for local dependencies, allowing the application to run independently.

2. Non-Functional Requirements:

2.1 Portability:

- The application, once containerized, should run consistently across different environments.

2.2 Performance:

- The web application should have acceptable response times, even under varying load conditions.

2.3 Security:

- Access to MongoDB Atlas should be securely configured, with proper authentication mechanisms.
- Docker containers and AWS deployment should adhere to best practices for security.

2.4 Scalability:

- The application should be designed to scale horizontally on AWS if the user demand increases.

2.5 Availability:

- The deployed application on AWS should have high availability, minimizing downtime.

2.3 SOLUTION APPROACH

1. **Implemented User Authentication:**
   - Integrate user authentication mechanisms to login into user accounts.
   - Implement authorization controls based on user roles to regulate access to different sections of the website.

2. **Created Blog Section:**
   - Develop functionality for users to create, edit, and delete blog posts.

3. **Community Integration:**
   - Integrate APIs to facilitate user connectivity and community building.
   - Provide features for users to interact, comment on blog posts, and share their experiences.

4. **Volunteer Form:**
   - Design and implement a volunteer form with relevant fields to collect information.
   - Develop a backend process to manage and display volunteer opportunities.

5. **Database Integration:**

- Set up MongoDB as the database, considering the flexibility needed for data management.
- Connect the application to MongoDB Atlas to ensure secure storage and retrieval of critical user information.

6. **Security Measures:**

- Implement secure coding practices to protect against common web vulnerabilities.

7. **Deployment with Docker:**

- Containerize the application using Docker to ensure consistency and reliability across various environments.

8. **Scale for Growth:**

- Designed the architecture with scalability in mind, allowing for increased user engagement and data volume.
- Monitor performance metrics and scale resources on the cloud infrastructure as needed.

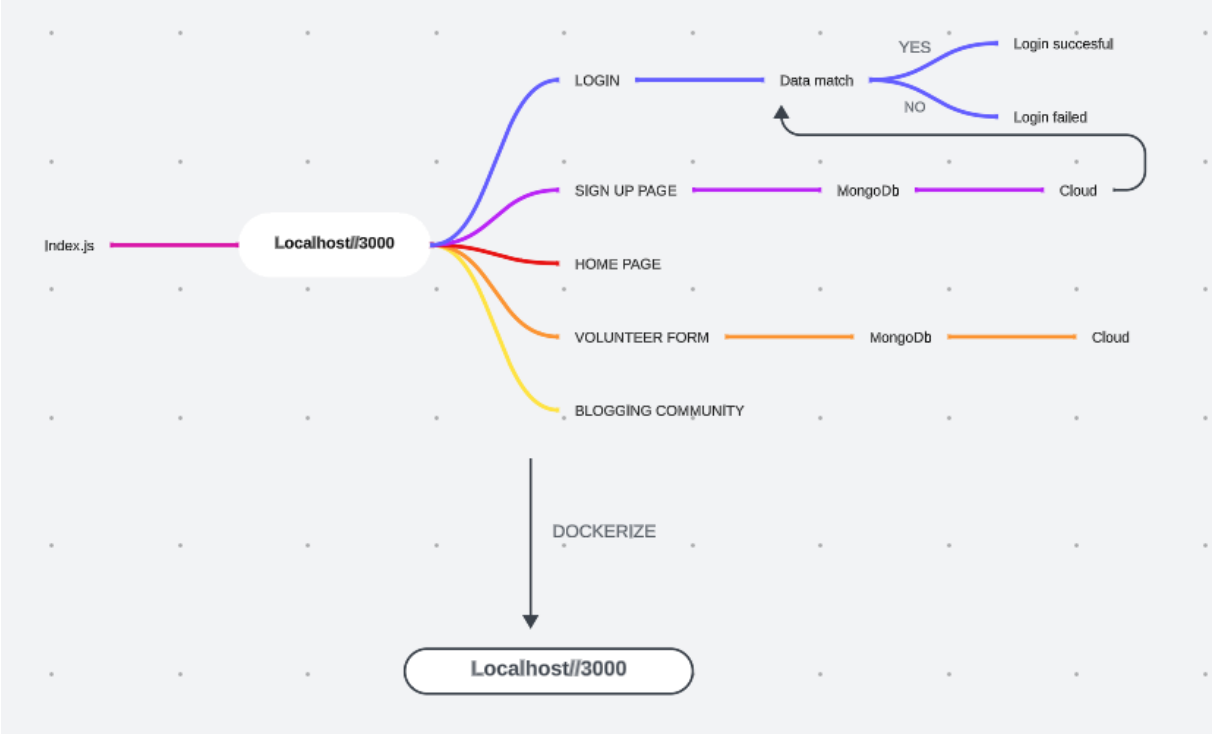
CHAPTER 3:MODELING AND IMPLEMENTATION DETAILS

3.1 DESIGN DIAGRAMS

MONGODB ENTITY DIAGRAM:

3.2 SEQUENCE DIAGRAMS

FLOW OF THE WEB APPLICATION:



3.3 IMPLEMENTATION DETAILS AND ISSUES

1. Frontend Development:

Technologies:

- HTML5, CSS3, and JavaScript for markup, styling, and interactivity.

Steps:

Set Up the Project:

- Initialize a new project using a package manager (npm or yarn).
- Configure the project structure and dependencies.

Design User Interface:

- Create UI wireframes and designs for different sections of the website (blog, volunteer form, user profiles, etc.).
- Implement responsive design for a seamless user experience across devices.

Integrate API for Community Features:

- Connect with relevant APIs to enable user interactions, comments, and sharing of experiences.
- Implement features for users to engage with each other and build a sense of community.

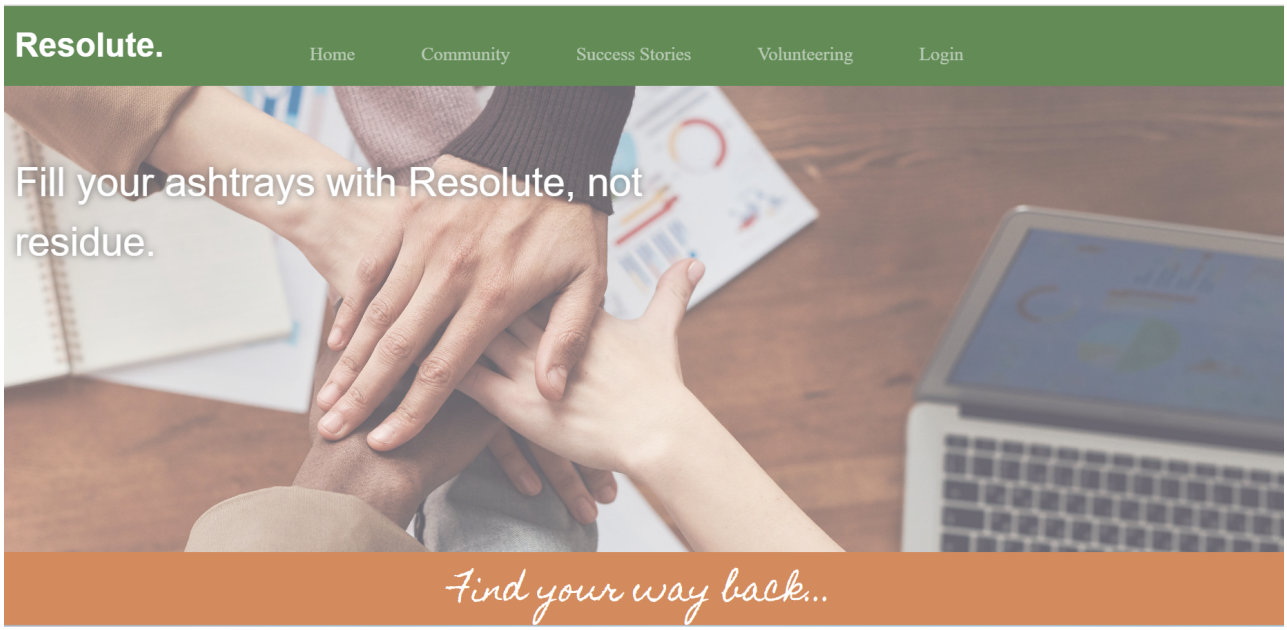
Implement Blog Section:

- Develop pages for creating, editing, and deleting blog posts.
- Display blog posts with appropriate tags and comments.

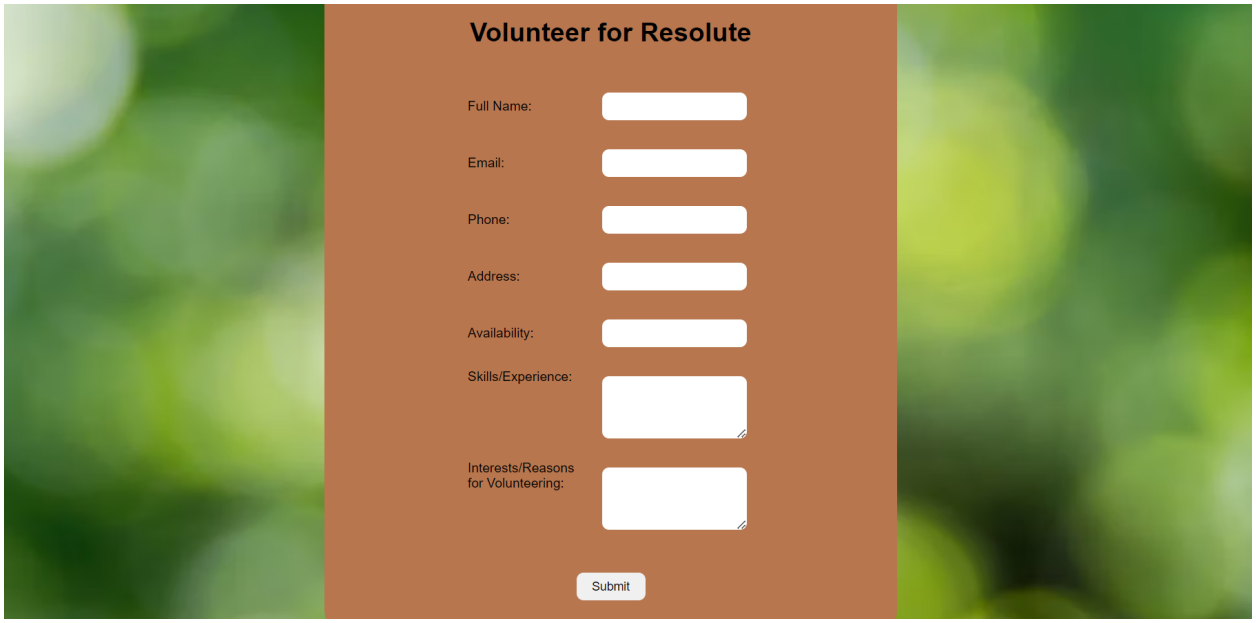
Volunteer Form:

- Design and implement a form for users to submit volunteer information.
- Provide a user-friendly interface for managing and displaying volunteer opportunities.

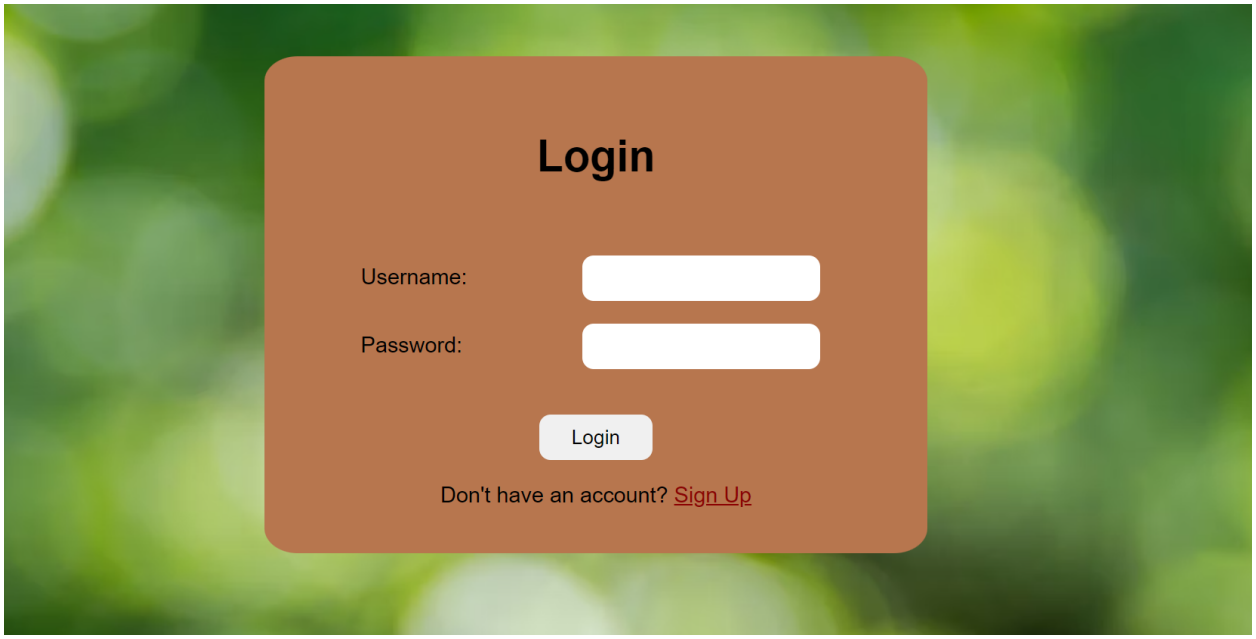
HomePage:



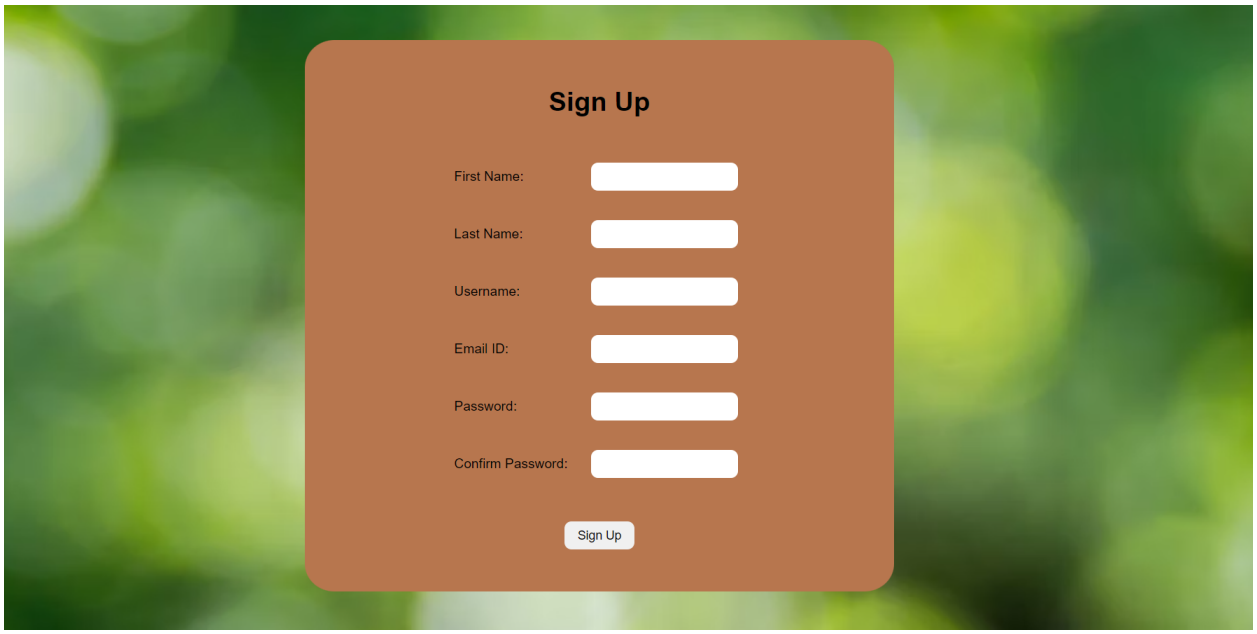
From the home page we can navigate to volunteering form:



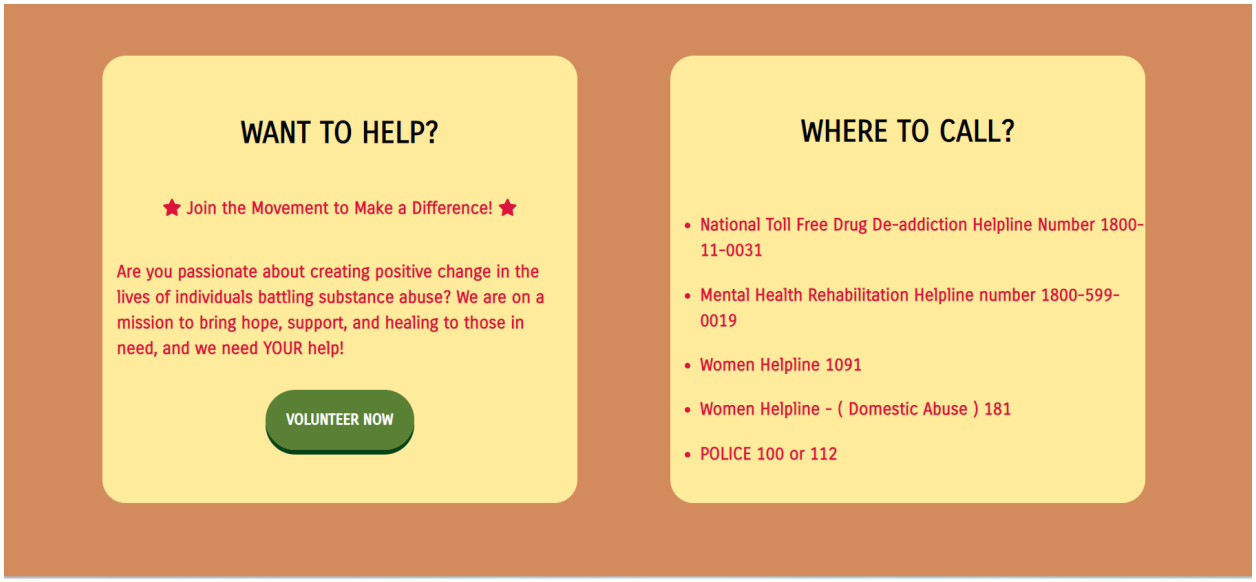
Navigating to Login page:



Navigating to Sign Up page:



Going further down the homepage, we have a link to the volunteering form through a button:



2. Backend Development:

Technologies:

- Node.js with Express or a similar framework for building the backend.
- Use a templating engine (EJS) for rendering dynamic content on the server-side.

Steps:

Database Integration (MongoDB):

- Connect the backend to MongoDB Atlas using the MongoDB Node.js driver.
- Create models for users, blog posts, and volunteer information.

Implement CRUD Operations:

- Develop endpoints for creating, reading, updating, and deleting blog posts.
- Implement CRUD operations for managing volunteer opportunities.

Handle API Integration:

- Integrate with external APIs for community features.
- Implement logic to manage user interactions, comments, and sharing.

Security Measures:

- Implement security best practices, including input validation and protection against common web vulnerabilities.
- Securely store sensitive information, such as API keys.

3. Database Design:

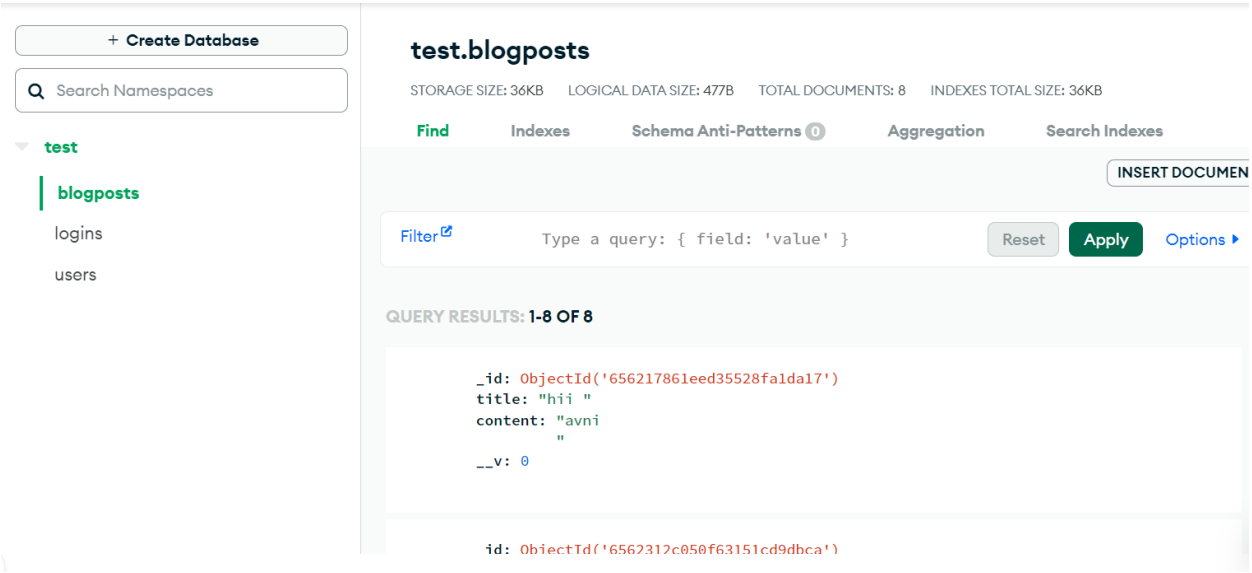
Design the Data Model:

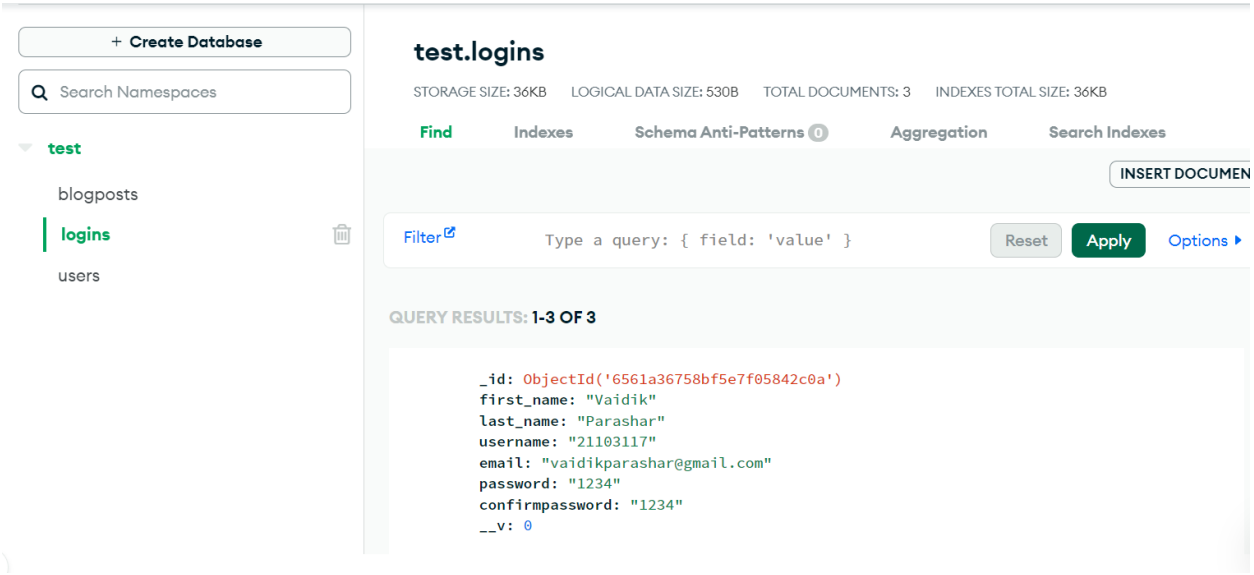
- Translate the logical data model into a MongoDB schema.
- Define collections for users, blog posts, and volunteer information.

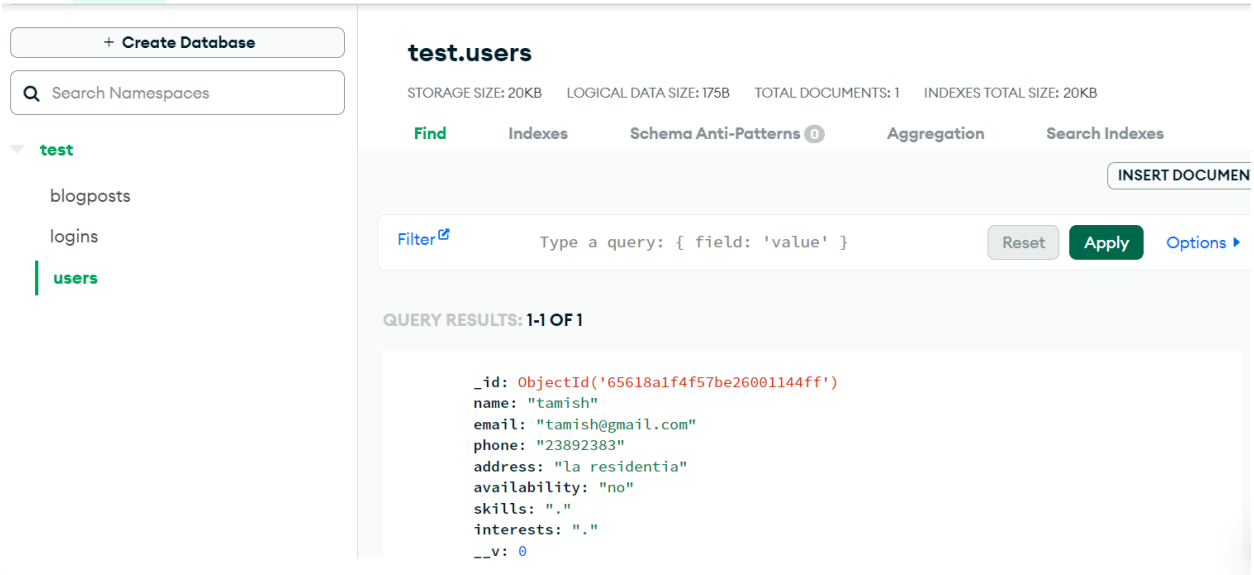
Ensure Data Integrity:

- Establish indexes and unique constraints where necessary.
- Enforce referential integrity in the data model.

Our mongoDB database looks like:







4. Docker and Deployment:

Technologies:

- Docker for containerization.
- AWS (Amazon Web Services) or another cloud provider for deployment.

Steps:

Containerize the Application:

- Write a Dockerfile to package the application and its dependencies into a container.

Deploy on AWS Cloud:

- Deploy the Docker container on AWS using services like EC2(Elastic Container Cloud).

Key features of index.js

Dependencies:

- Utilizes the Express framework for building the web application.
- Relies on MongoDB as the database with Mongoose as the ODM (Object Data Modeling) library.
- Incorporates `body-parser` for parsing request bodies.
- Implements session management using `express-session` for user authentication.

Express App Setup:

- Creates an instance of the Express application (`app`).
- Sets the port for the server to listen on, either from the environment variable `PORT` or the default port 3000.

Middleware Setup:

- Serves static files from the "public" folder, enabling access to client-side assets.
- Configures middleware for parsing URL-encoded form data.
- Initializes session middleware with a secret key for user session management.

Routes for Rendering HTML Pages:

- Defines routes for rendering various HTML pages using the EJS templating engine.
- Pages include the main page (`index.ejs`), login page (`login.ejs`), community page (`community.ejs`), signup page (`signup.ejs`), success page (`success_st.ejs`), and volunteering page (`volunteering.ejs`).

MongoDB Connection:

- Connects to a MongoDB Atlas database using a connection string with credentials and connection options.

Defining MongoDB Schemas and Models:

- Defines Mongoose schemas for entities such as users (`User`), signup information (`signup`), and blog posts (`Post`).

Handling Form Submissions:

- Implements routes for handling form submissions, including:
  - `POST /submit_volunteer`: Handles volunteer form submissions, creating a new user instance and saving it to the MongoDB database.
  - `POST /login`: Handles user login, checking credentials against stored information.
  - `POST /signup`: Handles user signup, creating a new user instance and saving it to the MongoDB database.

API Endpoints for Blog Posts:

- Defines API endpoints for managing blog posts:
  - `GET /api/posts`: Retrieves all blog posts from the MongoDB `Post` collection.
  - `POST /api/posts`: Creates a new blog post, with post data sent in the request body.

Server Start:

- Initiates the server to listen on the specified port.
- Outputs a log message indicating the server is running.

Index.js:

```
JS index.js > ...
1   const express = require('express');
2   const mongoose = require('mongoose');
3   const bodyParser = require('body-parser');
4   const session = require('express-session');
5
6
7   const app = express();
8   const PORT = process.env.PORT || 3000;
9
10  // Reference to public folder for all static files
11  app.use(express.static("public"));
12
13  // Middleware to parse request bodies
14  app.use(bodyParser.urlencoded({ extended: true }));
15  app.use(session({
16      secret: 'your-secret-key',
17      resave: true,
18      saveUninitialized: true
19  }));
```

```
21    // Serve the HTML form
22    app.get('/', (req, res) => {
23        res.render("index.ejs");
24    });
25
26    app.get('/login', (req, res) => {
27        res.render("login.ejs");
28    });
29
30    app.get('/community', (req, res) => {
31        res.render("community.ejs");
32    });
33
34    app.get('/signup', (req, res) => {
35        res.render("signup.ejs");
36    });
37
38    app.get('/success_st', (req, res) => {
39        res.render("success_st.ejs");
40    });
41
42    app.get('/volunteering', (req, res) => {
43        res.render("volunteering.ejs");
44    });
45
```

```
46  // Connect to MongoDB
47  mongoose.connect('mongodb+srv://admin:admin@cluster0.vayxzgs.mongodb.net/?retryWrites=true&w=majority');
48
49  // Define a schema
50  const userSchema = new mongoose.Schema({
51      name: String,
52      email: String,
53      phone: String,
54      address: String,
55      availability: String,
56      skills: String,
57      interests: String
58
59  });
60
61  // Create a model
62  const User = mongoose.model('User', userSchema);
63
64
```

```
64
65  // Handle form submission
66  app.post('/submit_volunteer', async (req, res) => {
67      const newUser = new User({
68          name: req.body.full_name,
69          email: req.body.email,
70          phone: req.body.phone,
71          address: req.body.address,
72          availability: req.body.availability || false,
73          skills: req.body.skills,
74          interests: req.body.interests
75      });
76
77      try {
78          await newUser.save();
79          res.send('Form submitted successfully!');
80      } catch (error) {
81          console.error('Error saving to the database:', error);
82          res.status(500).send('Internal Server Error');
83      }
84  });
85
```

```
87  //Signup vaala form ****************************************
88
89
90  const signupSchema = new mongoose.Schema({
91      first_name: String,
92      last_name: String,
93      username: String,
94      email: String,
95      password: String,
96      confirmpassword: String,
97  });
98
```

```
// Create a model
const signup = mongoose.model('login', signupSchema);

app.post('/login', async (req, res) => {
    const { username, password } = req.body;

    try {
        const user = await signup.findOne({ username });

        // Check if the user exists
        if (user) {
            // Compare the provided password with the password stored in the database (plain text)
            if (password === user.password) {
                req.session.userId = user._id; // No error should occur here now
                res.send('Login successful!');
            } else {
                res.status(401).send('Invalid credentials');
            }
        } else {
            res.status(401).send('Invalid credentials');
        }
    } catch (error) {
        console.error('Error logging in:', error);
        res.status(500).send('Internal Server Error');
    }

});

```

```
// Handle form submission
app.post('/signup', async (req, res) => {
    const {
        first_name,
        last_name,
        username,
        email,
        password,
        confirmpassword
    } = req.body;

```

```

    // Check if password equals confirm password
    if (password !== confirmpassword) {
        return res.status(400).send("Password and confirm password do not match.");
    }

    const newUser = new signup({
        first_name,
        last_name,
        username,
        email,
        password,
        confirmpassword
    });

    try {
        await newUser.save();
        res.send('Form submitted successfully!');
    } catch (error) {
        console.error('Error saving to the database:', error);
        res.status(500).send('Internal Server Error');
    }
});
```

```
164
165  // Define a Mongoose schema
166  const blogpostSchema = new mongoose.Schema({
167      title: String,
168      content: String,
169  });
170
171  // Create a Mongoose model
172  const Post = mongoose.model('blogpost', blogpostSchema);
173
174  app.use(express.static('public'));
175  app.use(bodyParser.json());
176
177  // Get all blog posts
178  app.get('/api/posts', async (req, res) => {
179      try {
180          const posts = await Post.find();
181          res.json(posts);
182      } catch (error) {
183          res.status(500).json({ error: 'Internal Server Error' });
184      }
185  });
186
```

```
187  // Create a new blog post
188  app.post('/api/posts', async (req, res) => {
189      const { title, content } = req.body;
190
191      try {
192          const newPost = await Post.create({ title, content });
193          res.json(newPost);
194      } catch (error) {
195          res.status(500).json({ error: 'Internal Server Error' });
196      }
197  });
198
199
200  // Start the server
201  app.listen(PORT, () => {
202      console.log(`Server is running on http://localhost:${PORT}`);
203  });
```

Issues faced:

- Connectivity of form from database
- In containerizing the web app
- Deploying the container on aws

3.4 RISK ANALYSIS AND MITIGATION

1. Technical Risks:

Risk 1: Technology Compatibility Issues

- Impact: Incompatibility between the chosen frontend and backend technologies.
- Mitigation:
    - Solution: Conduct thorough compatibility testing during development.
    - Contingency: Have a backup plan for alternative technologies if compatibility issues persist.

Risk 2: Third-Party API Reliability

- Impact: Dependency on external APIs for community features.
- Mitigation:
    - Solution: Select reputable and well-documented APIs.
    - Contingency: Develop fallback mechanisms or alternative API integrations.

2. Operational Risks:

Risk 3: Communication Breakdown

- Impact: Lack of effective communication leading to misunderstandings.
- Mitigation:
    - Solution: Establish clear communication channels and protocols.
    - Contingency: Implement regular team check-ins and project status updates.

3. External Risks:

Risk 4: Regulatory Changes

- Impact: Changes in regulations affecting the website's functionality.
- Mitigation:
    - Solution: Stay informed about relevant regulations.

- Contingency: Develop a rapid response plan for adapting to regulatory changes.

4. Security Risks:

Risk 5: Data Breach

- Impact: Unauthorized access to user data.
- Mitigation:
  - Solution: Implement encryption for sensitive data.
  - Contingency: Have a response plan for data breaches, including informing affected users.

Risk 6: API Key Exposure

- Impact: Unauthorized use of API keys.
- Mitigation:
  - Solution: Securely store and manage API keys.
  - Contingency: Rotate API keys regularly and have a process for key revocation.

5. Deployment and Scalability Risks:

Risk 7: Deployment Failures

- Impact: Issues during the deployment of the Docker container on AWS.
- Mitigation:
  - Solution: Implement thorough testing in a staging environment before deployment.
  - Contingency: Have a rollback plan in case of deployment failures.

Risk 8: Inadequate Scalability

- Impact: Unexpected traffic spikes causing performance issues.
- Mitigation:
  - Solution: Design the architecture with scalability in mind.
  - Contingency: Implement auto-scaling capabilities on the cloud infrastructure.

6. Usability and User Engagement Risks:

Risk 9: Poor User Adoption

- Impact: Users not engaging with the community features.
- Mitigation:
    - Solution: Conduct user testing and gather feedback during development.
    - Contingency: Develop marketing strategies to promote user engagement.

Risk 10: Lack of Inclusivity

- Impact: Failure to create an inclusive environment for diverse users.
- Mitigation:
    - Solution: Solicit feedback from a diverse group of users during design.
    - Contingency: Regularly assess user feedback and make iterative improvements.

CHAPTER 4: TESTING

4.1 ERROR AND EXCEPTION HANDLING

Handling Database Operations:

The submit_volunteer route and the signup route include asynchronous operations that interact with the database using Mongoose.

The try...catch blocks are used to capture errors that might occur during these database operations.

For example, in the submit_volunteer route:

```
// Handle form submission
app.post('/submit_volunteer', async (req, res) => {
    const newUser = new User({
        name: req.body.full_name,
        email: req.body.email,
        phone: req.body.phone,
        address: req.body.address,
        availability: req.body.availability || false,
        skills: req.body.skills,
        interests: req.body.interests
    });

    try {
        await newUser.save();
        res.send('Form submitted successfully!');
    } catch (error) {
        console.error('Error saving to the database:', error);
        res.status(500).send('Internal Server Error');
    }
});
```

This block attempts to save a new user to the database. If an error occurs during this operation, it is caught and logged (console.error) and an HTTP response with a 500 status code is sent back to the client.

Handling Login Operation:

In the login route, the try...catch block is used to handle errors that may occur during the login process.

For example:

```
app.post('/login', async (req, res) => {
    const { username, password } = req.body;

    try {
        const user = await signup.findOne({ username });

        // Check if the user exists
        if (user) {
            // Compare the provided password with the password stored in the database (plain text)
            if (password === user.password) {
                req.session.userId = user._id; // No error should occur here now
                res.send('Login successful!');
            } else {
                res.status(401).send('Invalid credentials');
            }
        } else {
            res.status(401).send('Invalid credentials');
        }
    } catch (error) {
        console.error('Error logging in:', error);
        res.status(500).send('Internal Server Error');
    }
```

This block attempts to find a user in the database based on the provided username. If any error occurs during this process (e.g., a database connection issue or a syntax error), it is caught, logged, and an appropriate error response is sent back.

Handling Blog Post Operations:
Similar error handling is applied to the routes handling blog post operations (/api/posts).
For instance, in the get('/api/posts') route:

```
// Get all blog posts
app.get('/api/posts', async (req, res) => {
    try {
        const posts = await Post.find();
        res.json(posts);
    } catch (error) {
        res.status(500).json({ error: 'Internal Server Error' });
    }
});
```

Here, if there is an error while trying to retrieve blog posts from the database, it is caught, and an error response is sent.

4.2 LIMITATIONS OF THE SOLUTION

1. Privacy and Anonymity:

- Substance abuse is a sensitive topic, and users may be concerned about privacy. Ensuring robust privacy settings and options for users to control the visibility of their posts and personal information.

- Implementing strict measures to safeguard user anonymity, especially if the platform is intended for individuals who may not want their identity disclosed.

- Sensitive information in the databases is not secure. We can use Becrypt to ensure that password and other sensitive information is kept encrypted.

2. Content Moderation:

- Managing user-generated content is crucial, especially when dealing with a vulnerable user group. Implementing effective content moderation to filter out inappropriate or triggering content.

- Developing mechanisms for users to report concerning posts, and have a moderation team in place to address issues promptly.

3. Triggering Content:

- Posts related to substance abuse and recovery can be triggering for some users. Implementing content warnings and ensuring users have the option to filter or customize their feed to avoid potentially harmful content.

4. Streak Feature Challenges:

- While streaks can be motivational, they may also create unnecessary pressure for some users. Monitoring user feedback and mental health indicators to ensure the feature is encouraging rather than stressful.

5. Ensuring Authenticity:

- Verify the authenticity of user posts to prevent misuse or false claims. Implementing features to validate users' progress, such as milestones achieved or days sober.

6. Legal and Ethical Considerations:

- Respecting user rights and adhere to laws regarding user data protection and mental health support services.

7. Community Guidelines:

- Addressing potential issues like cyberbullying, discrimination, or any form of harm to users.

CHAPTER 5: FINDINGS,CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

In conclusion, our substance abuse support website stands as a testament to our commitment to making a positive impact on individuals facing the challenges of addiction. Through a meticulous combination of informative content, community engagement, and future-focused enhancements, our project is poised to be a beacon of support, empowerment, and inspiration.

Key Achievements:

1. Community Connection: The survey results have provided valuable insights into the demographics and consumption patterns of our community, laying the groundwork for tailored interventions.

2. User Recognition of Motivation's Role: The survey highlighted that a significant 72.6% of participants acknowledge the pivotal role of motivation in the process of quitting. This insight validates our mission and positions our website as a crucial resource aligning with user beliefs and needs.

3. Future Enhancements for a Holistic Approach: The planned transformation into a social media app, streak challenges, enhanced frontend design, and the addition of a success story tab represent a strategic roadmap. These features will not only enhance user engagement but also provide diverse avenues for motivation, support, and celebration of recovery milestones.

The Road Ahead:

As we move forward, our commitment remains unwavering. We envision a future where our platform continues to evolve, adapt, and respond to the ever-changing needs of our community. By actively integrating user feedback, staying attuned to emerging trends, and consistently refining our resources, we aim to create a lasting and positive impact on the lives of those navigating the challenging path to recovery.

5.2 Future Enhancements Roadmap for Our Website:

1. Transformation into a Social Media App:

- **Objective:** Convert the blogging page into a dynamic social media platform where users can share their journey through photos, fostering a visual and supportive community.

- **Implementation:** Integrate photo-sharing capabilities, comments, and user profiles to enhance engagement and provide a more interactive and visually appealing experience.

2. Streak Feature for Daily Challenges:

- **Objective:**Encourage users to take on challenges like "21 Days Sober" by implementing a streak feature, requiring daily posts to maintain the streak.

- **Implementation:** Develop a user-friendly interface that facilitates easy participation in daily challenges, motivating users through gamification and the satisfaction of maintaining streaks.

3. Frontend Enhancement for User-Friendly Design:

- **Objective:** Elevate the user experience by overhauling the frontend to be more intuitive, aesthetically pleasing, and user-friendly.

- **Implementation:** Utilize modern design principles, responsive layouts, and intuitive navigation to create an inviting and accessible interface, ensuring a positive and seamless user journey.

4. Success Story Showcase:

-**Objective:**Inspire and motivate users by curating a dedicated "Success Story" tab, highlighting individuals who have successfully overcome addiction with the help of our app.

- **Implementation:** Create a compelling section featuring diverse success stories, integrating multimedia elements to provide a comprehensive view of the transformative journeys experienced by our community members.

5. Continuous User Feedback Integration:

- **Objective:** Foster an iterative development process by actively seeking and incorporating user feedback for ongoing improvement.

-Implementation: Implement feedback mechanisms, surveys, and user testing to gather insights and perspectives, ensuring that future enhancements align with the evolving needs and preferences of our user base.

By incorporating these future enhancements, our website will evolve into a multifaceted platform that not only provides valuable resources but also cultivates a dynamic and supportive community. Through visual storytelling, challenges, and an enhanced user interface, we aim to create a space that not only facilitates recovery but celebrates the triumphs of those on their journey to overcoming substance abuse.

REFERENCES

https://youtu.be/tVgnhxb-ScE?si=50KdiZCsSB6HpRXq
https://youtu.be/9FD2ugeS4OU?si=5deicB-MeT7qn28r
https://youtu.be/3c-iBn73dDE?si=RnatFEqNALF4Ry7O
https://youtu.be/qNIniDftAcU?si=YhKeFv6UjtL4jCtQ
https://explore.skillbuilder.aws/learn/signin
https://www.udemy.com/course/the-complete-web-development-bootcamp/learn/lecture/37348056?start=0#overview
https://docs.docker.com/docker-hub/
https://www.mongodb.com/docs/
https://youtu.be/OML9f6LXUUs?si=ZnpeVSu1GCQCh9lu

GITHUB REPOSITORY LINK:

https://github.com/aggavni4/Resolute.git

DOCKER-HUB REPOSITORY LINK:

https://hub.docker.com/repository/docker/vaidikp/resolute-minor-project/general