First created the react template using Vite

npm create vite@latest

Then imported some modules:

npm module axios react

Created a sample app bar for my screen visibility and mobile response

This is the appbar,.jsx file with the backend submitting form thing:

```
// ContactUs.js
import React, { useState } from 'react';
import AppBar from './appbar';
import axios from 'axios';

const ContactUs = () => {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    companyName: '',
    message: '',
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData((prevData) => ({
      ...prevData,
      [name]: value,
    }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    try {
      const response = await axios.post('/api/contact', formData);

      if (response.status !== 201) {
        throw new Error('Failed to submit form');
      }
      // Handle success
      alert('Form submitted successfully!');
      // Optionally, clear form fields
      setFormData({
        name: '',
        email: '',
        companyName: '',
        message: '',
      });

    } catch (error) {
      console.error('Error submitting form:', error);
      alert('Failed to submit form');
    }
  };

  return (
    <div className="w-full min-h-screen flex flex-col font-sans">
      <AppBar />
      <div className="flex-1 bg-[#10102a] text-white px-4 md:px-8
lg:px-16">
        <h1 className="text-4xl font-bold text-center mt-8 mb-8">CONTACT
US</h1>
        <div className="flex flex-col lg:flex-row">
          {/* Left Side Content */}
          <div className="flex-1 p-8 flex justify-center items-center">
            <div className="space-y-4">
              <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Nunc vulputate libero et velit interdum, ac aliquet odio mattis.</p>
              <div className="space-y-4">
                <div className="flex items-center space-x-2">
                  <span className="material-icons h-6 w-6
cursor-pointer">email</span>
                  <a
href="mailto:teamvyadh@vit.ac.in">teamvyadh@vit.ac.in</a>
                </div>
                <div className="flex items-center space-x-2">
                  <span className="material-icons h-6 w-6">phone</span>
                  <a href="tel:+919827036208">+91 9827036208</a>
                </div>
                <div className="flex items-center space-x-2">
                  <span className="material-icons h-6
w-6">location_on</span>
                  <p>VIT, Vellore Campus, Tiruvallam Rd, Katpadi, Vellore,
Tamil Nadu 632014</p>
                </div>
              </div>
            </div>
          </div>

          {/* Right Side Form */}
          <div className="flex-1 p-8">
            <div className="bg-gray-800 bg-opacity-50 p-8 rounded-lg
shadow-lg">
              <form onSubmit={handleSubmit}>
                <div className="mb-4">
                  <label className="block text-sm font-semibold mb-2"
htmlFor="name">Your Name</label>
                  <input
                    type="text"
                    id="name"
                    name="name"
                    value={formData.name}
                    onChange={handleChange}
                    placeholder="Your Name"
                    className="w-full p-3 border border-gray-300 rounded
bg-white text-black focus:outline-none focus:ring-2 focus:ring-gray-500
focus:border-transparent"
                    required
                  />
                </div>
                <div className="mb-4">
                  <label className="block text-sm font-semibold mb-2"
htmlFor="email">Email</label>
                  <input
                    type="email"
                    id="email"
                    name="email"
                    value={formData.email}
                    onChange={handleChange}
                    placeholder="Email"
                    className="w-full p-3 border border-gray-300 rounded
bg-white text-black focus:outline-none focus:ring-2 focus:ring-gray-500
focus:border-transparent"
                    required
                  />
                </div>
                <div className="mb-4">
                  <label className="block text-sm font-semibold mb-2"
htmlFor="companyName">Company Name</label>
                  <input
                    type="text"
                    id="companyName"
                    name="companyName"
                    value={formData.companyName}
                    onChange={handleChange}
                    placeholder="Company Name"
                    className="w-full p-3 border border-gray-300 rounded
bg-white text-black focus:outline-none focus:ring-2 focus:ring-gray-500
focus:border-transparent"
                  />
                </div>
                <div className="mb-4">
                  <label className="block text-sm font-semibold mb-2"
htmlFor="message">Message</label>
                  <textarea
                    id="message"
                    name="message"
                    value={formData.message}
                    onChange={handleChange}
                    placeholder="Enter your Message"
                    className="w-full p-3 border border-gray-300 rounded
bg-white text-black focus:outline-none focus:ring-2 focus:ring-gray-500
focus:border-transparent"
                    rows="6"
                    required
                  />
                </div>
                <button
```

```
                  type="submit"
                  className="w-full bg-gray-700 text-white p-3 rounded-lg
font-semibold hover:bg-gray-600 transition-colors focus:outline-none
focus:ring-2 focus:ring-gray-500 focus:ring-opacity-50"
                >
                  Submit
                </button>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};

export default ContactUs;
```

Sure, here's a description of each Tailwind CSS class used in the `ContactUs.js` file:

1. Container (`<div className="w-full min-h-screen flex flex-col font-sans">`)
   - `w-full`: Full width.
   - `min-h-screen`: Minimum height of 100vh (viewport height).
   - `flex`: Apply flexbox layout.
   - `flex-col`: Arrange children in a column.
   - `font-sans`: Use a sans-serif font.

2. Content Container (`<div className="flex-1 bg-[#10102a] text-white px-4 md:px-8 lg:px-16">`)
   - `flex-1`: Flex-grow to fill remaining space.
   - `bg-[#10102a]`: Background color with the custom hex value.
   - `text-white`: White text color.
   - `px-4`: Horizontal padding of 1rem (16px).
   - `md:px-8`: Horizontal padding of 2rem (32px) for medium screens and up.
   - `lg:px-16`: Horizontal padding of 4rem (64px) for large screens and up.

3. Heading (`<h1 className="text-4xl font-bold text-center mt-8 mb-8">`)
   - `text-4xl`: Text size of 2.25rem (36px).
   - `font-bold`: Bold font weight.
   - `text-center`: Center-aligned text.
   - `mt-8`: Margin-top of 2rem (32px).

   - `mb-8`: Margin-bottom of 2rem (32px).

4. Flex Container for Form and Contact Info (`<div className="flex flex-col lg:flex-row">`)
   - `flex`: Apply flexbox layout.
   - `flex-col`: Arrange children in a column.
   - `lg:flex-row`: Arrange children in a row for large screens and up.

5. Left Side Content Container (`<div className="flex-1 p-8 flex justify-center items-center">`)
   - `flex-1`: Flex-grow to fill remaining space.
   - `p-8`: Padding of 2rem (32px).
   - `flex`: Apply flexbox layout.
   - `justify-center`: Center-align items horizontally.
   - `items-center`: Center-align items vertically.

6. Vertical Space (`<div className="space-y-4">`)
   - `space-y-4`: Vertical space of 1rem (16px) between child elements.

7. Contact Info Item (`<div className="flex items-center space-x-2">`)
   - `flex`: Apply flexbox layout.
   - `items-center`: Center-align items vertically.
   - `space-x-2`: Horizontal space of 0.5rem (8px) between child elements.

8. Right Side Form Container (`<div className="flex-1 p-8">`)
   - `flex-1`: Flex-grow to fill remaining space.
   - `p-8`: Padding of 2rem (32px).

9. Form Wrapper (`<div className="bg-gray-800 bg-opacity-50 p-8 rounded-lg shadow-lg">`)
   - `bg-gray-800`: Background color of gray-800.
   - `bg-opacity-50`: Background opacity of 50%.
   - `p-8`: Padding of 2rem (32px).
   - `rounded-lg`: Large border-radius.
   - `shadow-lg`: Large box shadow.

10. Form Label (`<label className="block text-sm font-semibold mb-2" htmlFor="name">`)
   - `block`: Display block.
   - `text-sm`: Text size of 0.875rem (14px).
   - `font-semibold`: Semi-bold font weight.
   - `mb-2`: Margin-bottom of 0.5rem (8px).

11. Form Input (`<input className="w-full p-3 border border-gray-300 rounded bg-white text-black focus:outline-none focus:ring-2 focus:ring-gray-500 focus:border-transparent">`)
   - `w-full`: Full width.
   - `p-3`: Padding of 0.75rem (12px).
   - `border`: Border.

- `border-gray-300`: Border color of gray-300.
- `rounded`: Border-radius.
- `bg-white`: Background color white.
- `text-black`: Black text color.
- `focus:outline-none`: Remove outline on focus.
- `focus:ring-2`: Apply ring on focus with width of 2.
- `focus:ring-gray-500`: Ring color of gray-500.
- `focus:border-transparent`: Transparent border on focus.

12. Submit Button (`<button className="w-full bg-gray-700 text-white p-3 rounded-lg font-semibold hover:bg-gray-600 transition-colors focus:outline-none focus:ring-2 focus:ring-gray-500 focus:ring-opacity-50">`)
    - `w-full`: Full width.
    - `bg-gray-700`: Background color gray-700.
    - `text-white`: White text color.
    - `p-3`: Padding of 0.75rem (12px).
    - `rounded-lg`: Large border-radius.
    - `font-semibold`: Semi-bold font weight.
    - `hover:bg-gray-600`: Background color gray-600 on hover.
    - `transition-colors`: Transition for color change.
    - `focus:outline-none`: Remove outline on focus.
    - `focus:ring-2`: Apply ring on focus with width of 2.
    - `focus:ring-gray-500`: Ring color of gray-500.
    - `focus:ring-opacity-50`: Ring opacity of 50%.

These are the components used and obviously this is form chatgpt, but the code is researched and implemented by me.

This is the sample appbar.jsx file:

```
import React from 'react';

const AppBar = () => {
  return (
    <nav className="bg-gray-900 text-white p-4">
      <div className="container mx-auto flex justify-between
items-center">
        <div className="text-xl font-bold">Logo</div>
        <div className="space-x-4">
          <a href="#" className="hover:text-gray-400">About us</a>
          <a href="#" className="hover:text-gray-400">Achievements</a>
          <a href="#" className="hover:text-gray-400">Competitions</a>
          <a href="#" className="hover:text-gray-400">Sponsors</a>
          <a href="#" className="hover:text-gray-400">Board</a>
          <a href="#" className="hover:text-gray-400">Contact Us</a>
        </div>
      </div>
    </nav>
  );
};

export default AppBar;
```

And integrate this in the app.jsx file:

```
import React from 'react';
import ContactUs from './components/contactUs';

function App() {
  return (
    <div className="App">
      <ContactUs />
    </div>
  );
}

export default App;
```

Some elements like material icons and font san is implemented in indes.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css2?family=DM+Sans:wght@400;500;700&di
splay=swap" rel="stylesheet">
    <title>Vite + React</title>
  </head>
```

```html
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

Next comes the backend part:

Npm install express cors mongoose

Npm init -y // for initilaizing package.json if it doesn't appear after the above command

This is the server.js file which is our backend logic file:

```javascript
// server.js
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');

const app = express();
const PORT = process.env.PORT || 5000;

// Middleware
app.use(cors({
  origin: 'http://localhost:3000',
}));
app.use(express.json());

// MongoDB Connection
mongoose.connect('mongodb://mantissa6789:Mantis164758@ac-qn0oqwt-shard-00-00.9ramotn.mongodb.net:27017,ac-qn0oqwt-shard-00-01.9ramotn.mongodb.net:27017,ac-qn0oqwt-shard-00-02.9ramotn.mongodb.net:27017/?replicaSet=atlas-w4ijlq-shard-0&ssl=true&authSource=admin', {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    dbName: 'vyaadh-forms'
});

const db = mongoose.connection;
db.on('error', console.error.bind(console, 'MongoDB connection error:'));
db.once('open', () => {
  console.log('Connected to MongoDB');
});

// Define MongoDB Schema and Model
const contactSchema = new mongoose.Schema({
  name: String,
  email: String,
  companyName: String,
  message: String,
});

const Contact = mongoose.model('Contact', contactSchema);

// Routes
app.post('/api/contact', async (req, res) => {
  try {
    const { name, email, companyName, message } = req.body;

    const newContact = new Contact({
      name,
      email,
      companyName,
      message,
    });

    await newContact.save();

    console.log('Contact saved:', newContact);

    res.status(201).json({ message: 'Contact saved successfully' });
  } catch (error) {
    console.error('Error saving contact:', error);
    res.status(500).json({ message: 'Error saving contact' });
  }
});

// Start the server
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Here's a basic overview of each component in your `server.js` file:

1. Importing Required Modules:
```
  javascript
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
```

   - `express`: Web framework for Node.js, used to create the server.
   - `mongoose`: MongoDB object modeling tool, used for interacting with MongoDB.
   - `cors`: Middleware for enabling Cross-Origin Resource Sharing, allowing requests from different domains.

2. Initializing Express Application:
```
  javascript
const app = express();
const PORT = process.env.PORT || 5000;
```

   - `app`: Instance of the Express application.
   - `PORT`: The port on which the server will listen, defaulting to `5000` if not set in the environment variables.

3. Middleware Setup:
```
  javascript
app.use(cors({
  origin: 'http://localhost:3000',
}));
app.use(express.json());
```

   - `app.use(cors({...}))`: Enables CORS for requests coming from `http://localhost:3000`.
   - `app.use(express.json())`: Middleware to parse incoming JSON requests.

4. MongoDB Connection:
```
  javascript

mongoose.connect('mongodb://mantissa6789:Mantis164758@ac-qn0oqwt-shard-00-00.9ramotn.mongodb.net:27017,ac-qn0oqwt-shard-00-01.9ramotn.mongodb.net:27017,ac-qn0oqwt-shard-00-02.9ramotn.mongodb.net:27017/?replicaSet=atlas-w4ijlq-shard-0&ssl=true&authSource=admin', {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    dbName: 'vyaadh-forms'
  });
```

- Connects to a MongoDB database using Mongoose with specified connection options.

5. Database Connection Event Listeners:
   javascript

```
const db = mongoose.connection;
db.on('error', console.error.bind(console, 'MongoDB connection error:'));
db.once('open', () => {
  console.log('Connected to MongoDB');
});
```

- `db.on('error', ...)`: Logs any connection errors.
- `db.once('open', ...)`: Logs a message once connected to the database.

6. Define MongoDB Schema and Model:
   javascript

```
const contactSchema = new mongoose.Schema({
  name: String,
  email: String,
  companyName: String,
  message: String,
});

const Contact = mongoose.model('Contact', contactSchema);
```

- `contactSchema`: Defines the structure of documents in the `Contact` collection.
- `Contact`: Mongoose model based on the schema, used to interact with the `Contact` collection.

7. Define Routes:
   javascript

```
app.post('/api/contact', async (req, res) => {
  try {
    const { name, email, companyName, message } = req.body;

    const newContact = new Contact({
      name,
      email,
      companyName,
      message,
    });
    await newContact.save();

    console.log('Contact saved:', newContact);

    res.status(201).json({ message: 'Contact saved successfully' });
  } catch (error) {
    console.error('Error saving contact:', error);
    res.status(500).json({ message: 'Error saving contact' });
  }
});
```

- `app.post('/api/contact', ...)`: Handles POST requests to `/api/contact`.
- Creates a new `Contact` document from request body data and saves it to the database.
- Sends appropriate responses based on success or error.

8. Start the Server:

```
  javascript
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

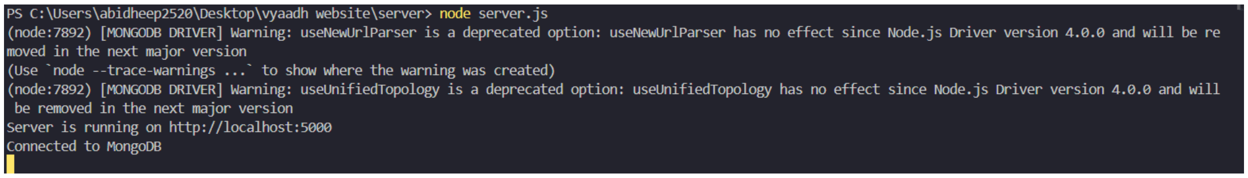
- Starts the server, listening on the specified port, and logs a message to the console.

Please use your own MongoDB ID and credentials. The credentials provided here are for demonstration purposes only and should not be misused. I am not on a premium plan and misuse can lead to database access issues and potential security risks.

So after this we start the backend server first with this command:

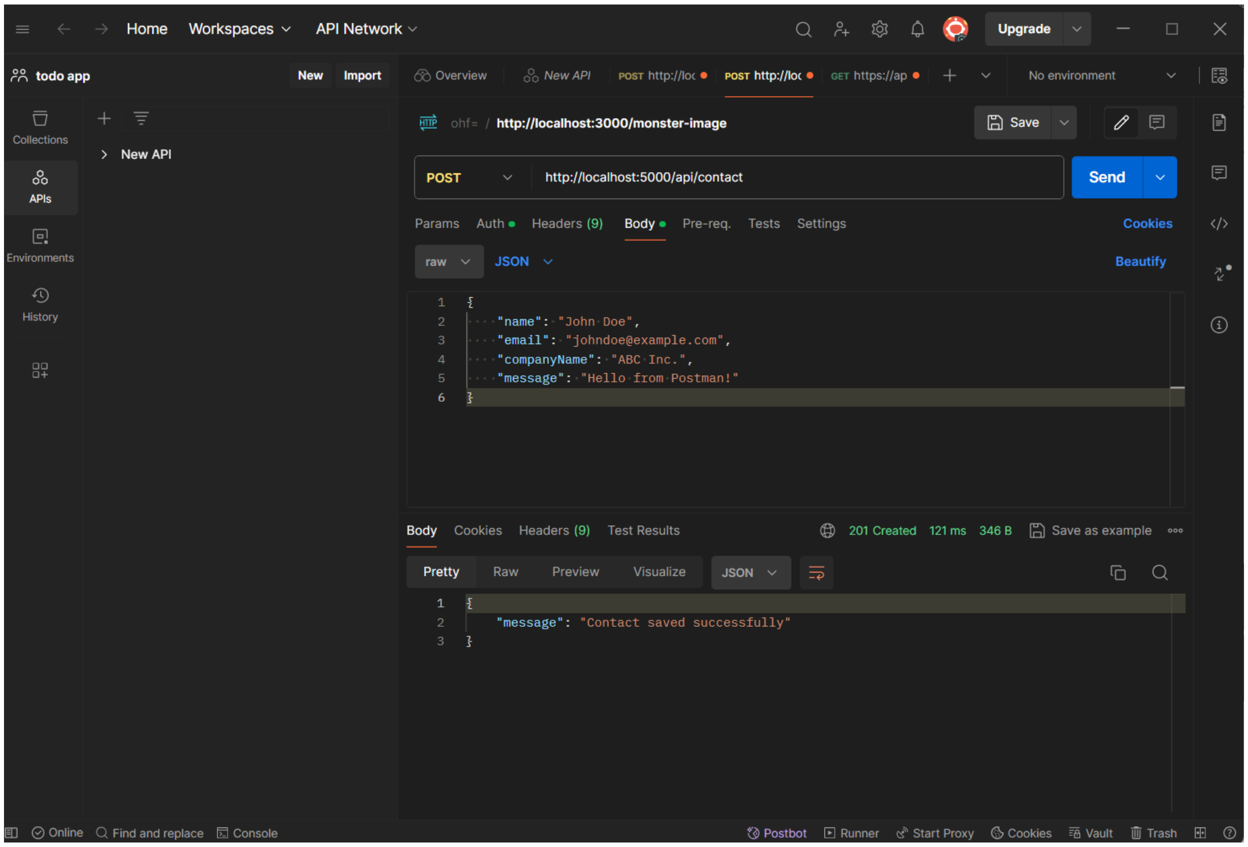
Node server.js

Check for the output like



```
PS C:\Users\abidheep2520\Desktop\vyaadh website\server> node server.js
(node:7892) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be re
moved in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:7892) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will
 be removed in the next major version
Server is running on http://localhost:5000
Connected to MongoDB
```

And also check the postman api if it's running correctly or not:



http://localhost:3000/monster-image

POST http://localhost:5000/api/contact

```
{
    "name": "John Doe",
    "email": "johndoe@example.com",
    "companyName": "ABC Inc.",
    "message": "Hello from Postman!"
}
```

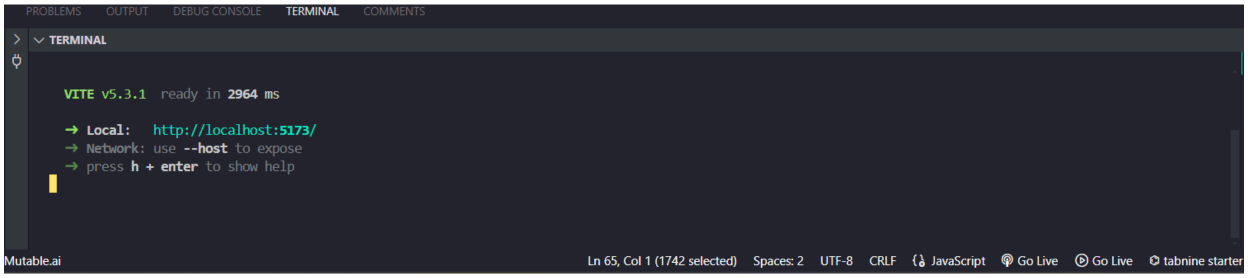
201 Created 121 ms 346 B

```
{
    "message": "Contact saved successfully"
}
```

Then start the frontend by giving this command:

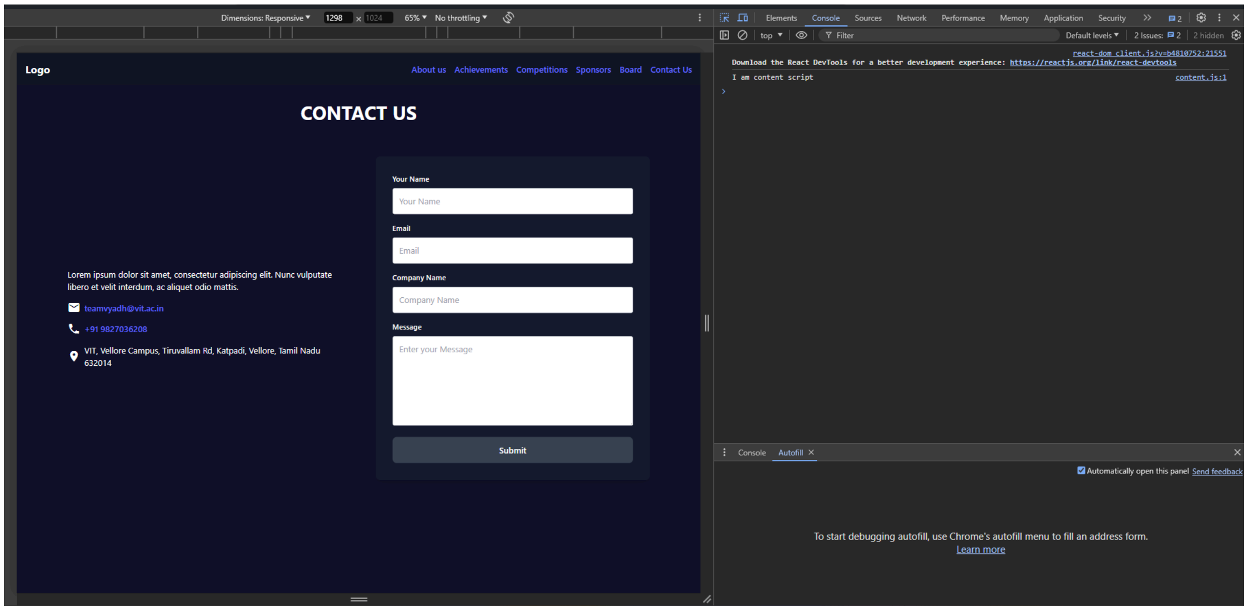
Npm run dev

And the output in the terminal looks something like this:



```
  VITE v5.3.1  ready in 2964 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

And check the output in the website:
For laptops:


Dimensions: Responsive 1298 x 1024 65% No throttling

Logo

About us Achievements Competitions Sponsors Board Contact Us

CONTACT US

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vulputate libero et velit interdum, ac aliquet odio mattis.

teamvyadh@vit.ac.in

+91 9827036208

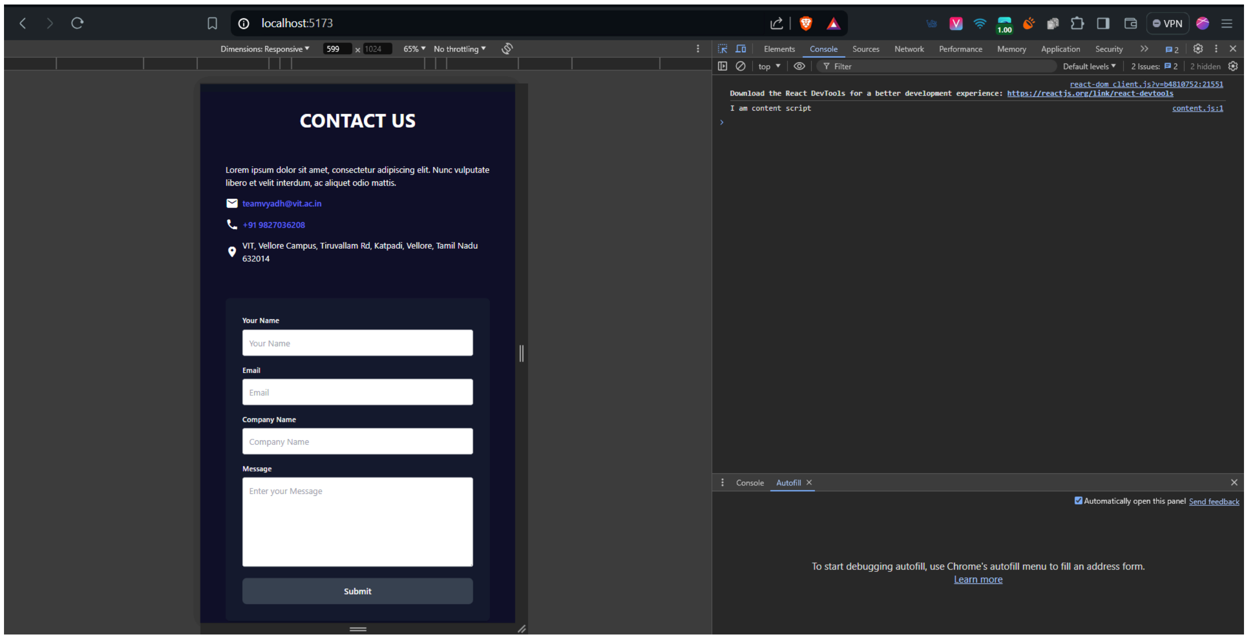
VIT, Vellore Campus, Tiruvallam Rd, Katpadi, Vellore, Tamil Nadu 632014

Your Name: Your Name
Email: Email
Company Name: Company Name
Message: Enter your Message
Submit

```
react-dom_client.js?v=b4810752:21551
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
I am content script                                          content.js:1
```

To start debugging autofill, use Chrome's autofill menu to fill an address form.
Learn more

For mobiles:


localhost:5173

Dimensions: Responsive 599 x 1024 65% No throttling

CONTACT US

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vulputate libero et velit interdum, ac aliquet odio mattis.

teamvyadh@vit.ac.in

+91 9827036208

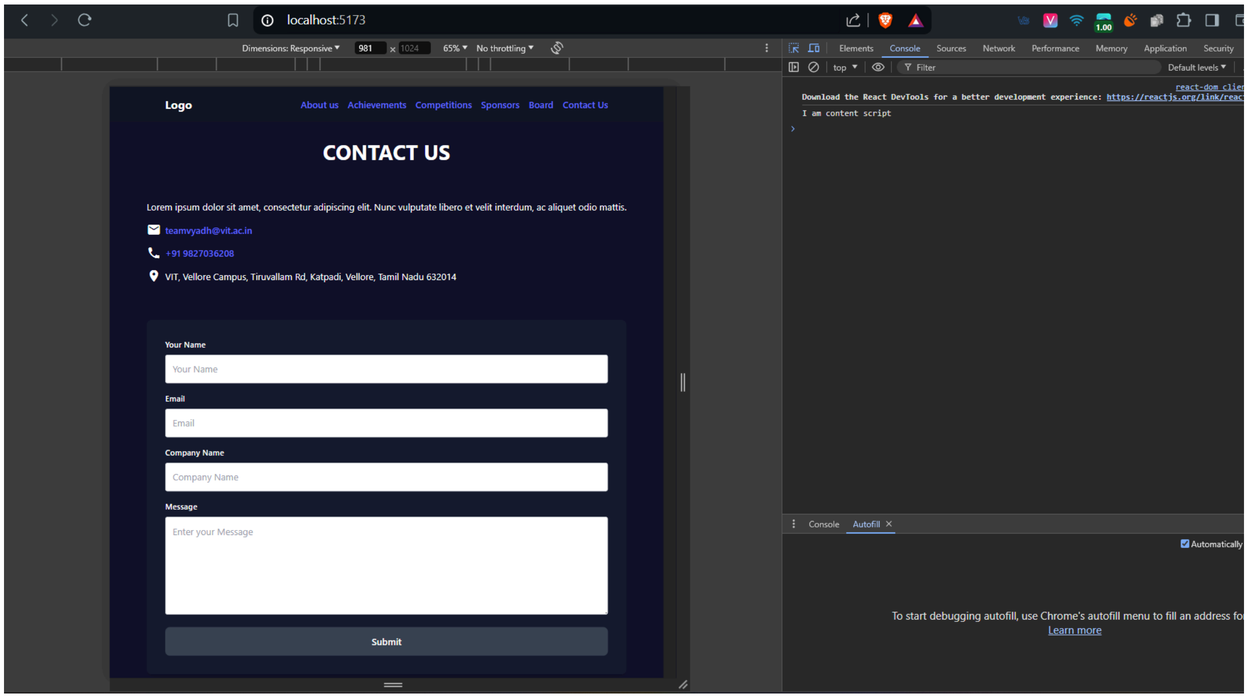
VIT, Vellore Campus, Tiruvallam Rd, Katpadi, Vellore, Tamil Nadu 632014

Your Name: Your Name
Email: Email
Company Name: Company Name
Message: Enter your Message
Submit

```
react-dom_client.js?v=b4810752:21551
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
I am content script                                          content.js:1
```

To start debugging autofill, use Chrome's autofill menu to fill an address form.
Learn more

For tab:


localhost:5173

Dimensions: Responsive ▾ 981 × 1024 65% ▾ No throttling ▾

Logo About us Achievements Competitions Sponsors Board Contact Us

CONTACT US

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vulputate libero et velit interdum, ac aliquet odio mattis.

teamvyadh@vit.ac.in

+91 9827036208

VIT, Vellore Campus, Tiruvallam Rd, Katpadi, Vellore, Tamil Nadu 632014

Your Name: Your Name

Email: Email

Company Name: Company Name

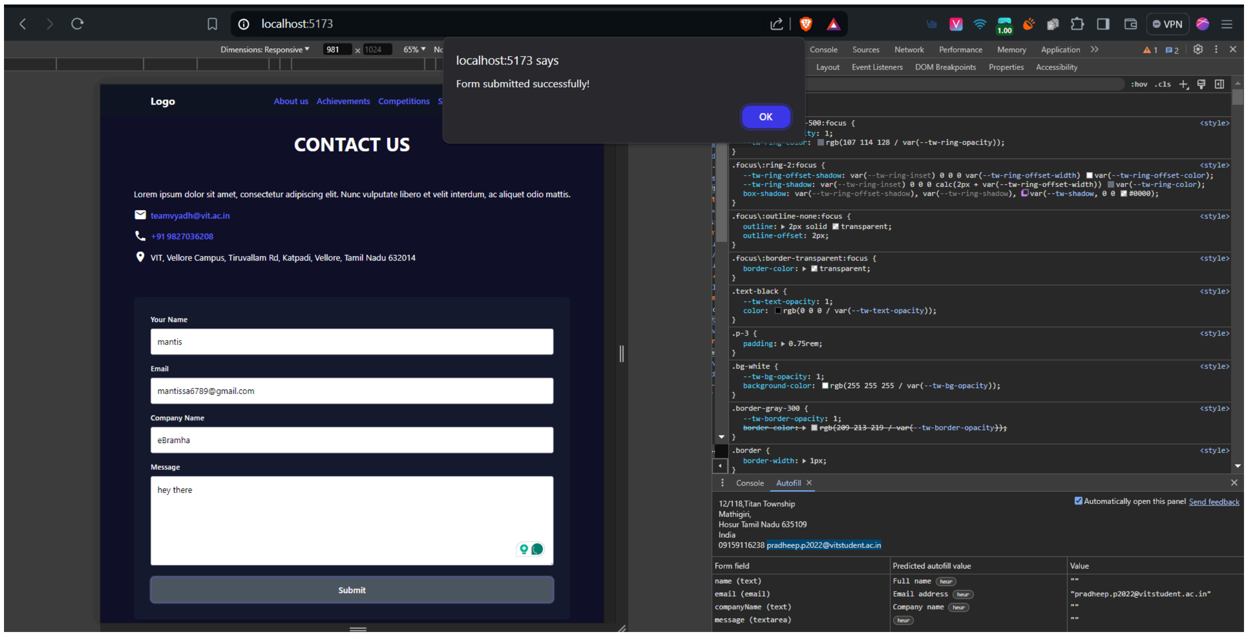
Message: Enter your Message

Submit

```
Download the React DevTools for a better development experience: https://reactjs.org/link/reac
I am content script
```

To start debugging autofill, use Chrome's autofill menu to fill an address fo
Learn more

So this is responsive now, we start to enter the dtails now and check for the connection between mongodb and frontend and backend


localhost:5173

Dimensions: Responsive ▾ 981 × 1024 65% ▾

localhost:5173 says

Form submitted successfully!

OK

Logo About us Achievements Competitions

CONTACT US

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vulputate libero et velit interdum, ac aliquet odio mattis.

teamvyadh@vit.ac.in

+91 9827036208

VIT, Vellore Campus, Tiruvallam Rd, Katpadi, Vellore, Tamil Nadu 632014

Your Name: mantis

Email: mantissa6789@gmail.com

Company Name: eBramha

Message: hey there

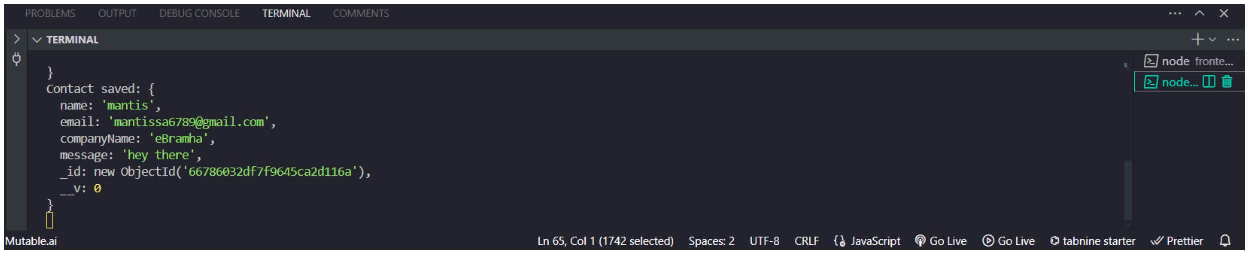
Submit

```
-500:focus {
  ity: 1;
  --tw-ring-color: rgb(107 114 128 / var(--tw-ring-opacity));
}
.focus\:ring-2:focus {
  --tw-ring-offset-shadow: var(--tw-ring-inset) 0 0 0 var(--tw-ring-offset-width) var(--tw-ring-offset-color);
  --tw-ring-shadow: var(--tw-ring-inset) 0 0 0 calc(2px + var(--tw-ring-offset-width)) var(--tw-ring-color);
  box-shadow: var(--tw-ring-offset-shadow), var(--tw-ring-shadow), var(--tw-shadow, 0 0 #0000);
}
.focus\:outline-none:focus {
  outline: ▸ 2px solid transparent;
  outline-offset: 2px;
}
.focus\:border-transparent:focus {
  border-color: ▸ transparent;
}
.text-black {
  --tw-text-opacity: 1;
  color: rgb(0 0 0 / var(--tw-text-opacity));
}
.p-3 {
  padding: ▸ 0.75rem;
}
.bg-white {
  --tw-bg-opacity: 1;
  background-color: rgb(255 255 255 / var(--tw-bg-opacity));
}
.border-gray-300 {
  --tw-border-opacity: 1;
  ~~border-color: ▸ rgb(209 213 219 / var(--tw-border-opacity));~~
}
.border {
  border-width: ▸ 1px;
}
```

12/118,Titan Township
Mathigiri,
Hosur Tamil Nadu 635109
India
09159116238 pradheep.p2022@vitstudent.ac.in

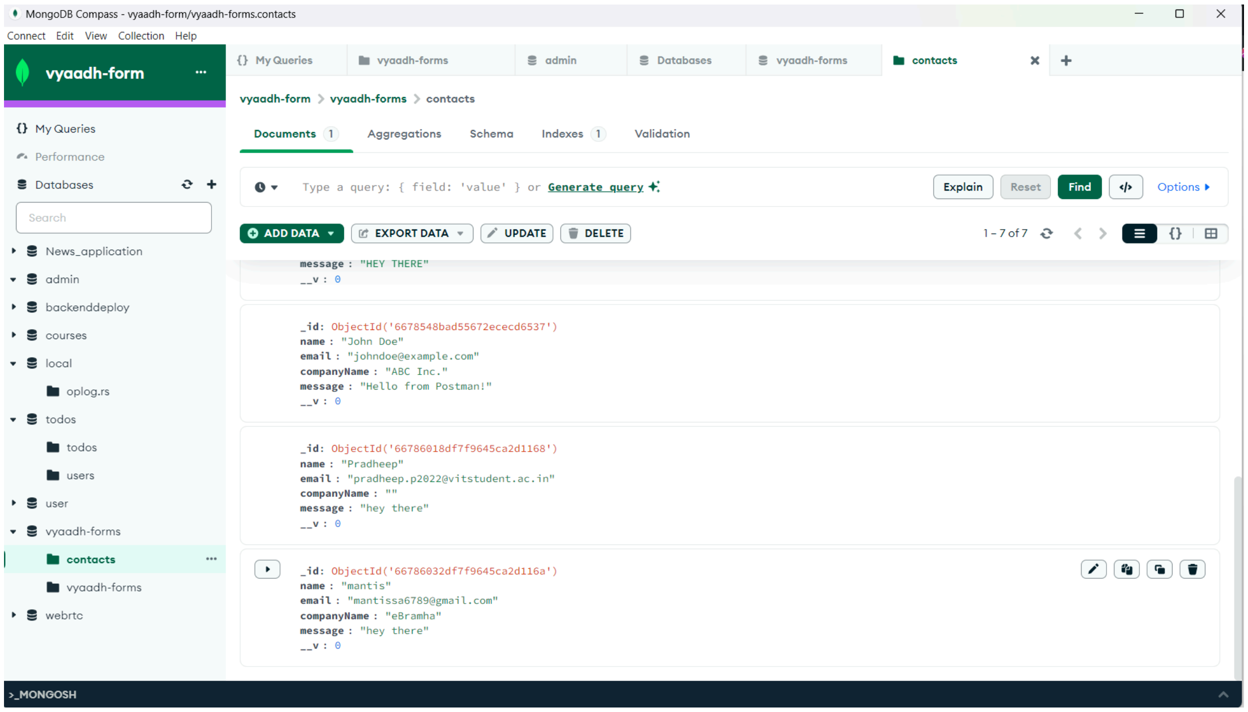
Form field	Predicted autofill value	Value
name (text)	Full name heur	""
email (email)	Email address heur	"pradheep.p2022@vitstudent.ac.in"
companyName (text)	Company name heur	""
message (textarea)	heur	""

this isi the website output



```
}
Contact saved: {
  name: 'mantis',
  email: 'mantissa6789@gmail.com',
  companyName: 'eBramha',
  message: 'hey there',
  _id: new ObjectId('66786032df7f9645ca2d116a'),
  __v: 0
}
```

This is the backend output



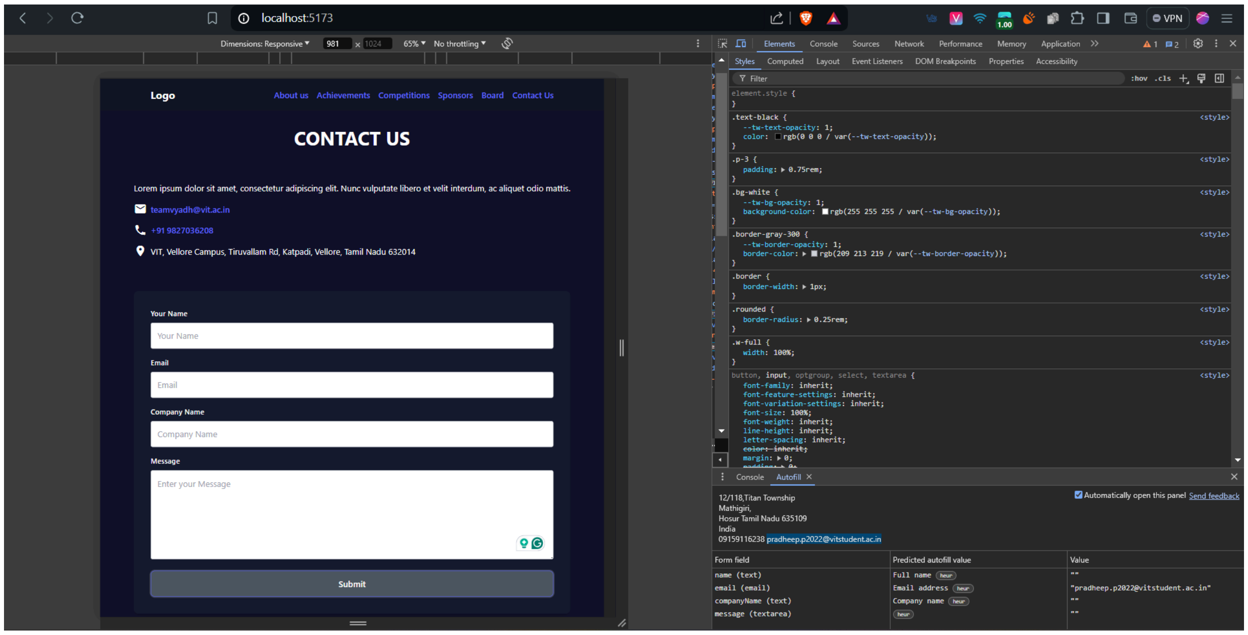
MongoDB Compass - vyaadh-form/vyaadh-forms.contacts

vyaadh-form > vyaadh-forms > contacts

```
message : "HEY THERE"
__v : 0

_id: ObjectId('6678548bad55672ececd6537')
name : "John Doe"
email : "johndoe@example.com"
companyName : "ABC Inc."
message : "Hello from Postman!"
__v : 0

_id: ObjectId('66786018df7f9645ca2d1168')
name : "Pradheep"
email : "pradheep.p2022@vitstudent.ac.in"
companyName : ""
message : "hey there"
__v : 0

_id: ObjectId('66786032df7f9645ca2d116a')
name : "mantis"
email : "mantissa6789@gmail.com"
companyName : "eBramha"
message : "hey there"
__v : 0
```

This is the data stored in the MongoDB(this app is MongoDB compass)

After clicking the okay from the alert with the message “the form has been submitted successfully”, the form gets refreshed



CONTACT US

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vulputate libero et velit interdum, ac aliquet odio mattis.

teamvyadh@vit.ac.in

+91 9827036208

VIT, Vellore Campus, Tiruvallam Rd, Katpadi, Vellore, Tamil Nadu 632014

Your Name: Your Name
Email: Email
Company Name: Company Name
Message: Enter your Message

```
element.style {
}
.text-black {                                              <style>
  --tw-text-opacity: 1;
  color: rgb(0 0 0 / var(--tw-text-opacity));
}
.p-3 {                                                     <style>
  padding: ▶ 0.75rem;
}
.bg-white {                                                <style>
  --tw-bg-opacity: 1;
  background-color: rgb(255 255 255 / var(--tw-bg-opacity));
}
.border-gray-300 {                                         <style>
  --tw-border-opacity: 1;
  border-color: ▶ rgb(209 213 219 / var(--tw-border-opacity));
}
.border {                                                  <style>
  border-width: ▶ 1px;
}
.rounded {                                                 <style>
  border-radius: ▶ 0.25rem;
}
.w-full {                                                  <style>
  width: 100%;
}
button, input, optgroup, select, textarea {                <style>
  font-family: inherit;
  font-feature-settings: inherit;
  font-variation-settings: inherit;
  font-size: 100%;
  font-weight: inherit;
  line-height: inherit;
  letter-spacing: inherit;
  ~~color: inherit;~~
  margin: ▶ 0;
```

12/118,Titan Township
Mathigiri,
Hosur Tamil Nadu 635109
India
09159116238 pradheep.p2022@vitstudent.ac.in

[✓] Automatically open this panel Send feedback

Form field	Predicted autofill value	Value
name (text)	Full name heur	""
email (email)	Email address heur	"pradheep.p2022@vitstudent.ac.in"
companyName (text)	Company name heur	""
message (textarea)	heur	""

So that’s all.