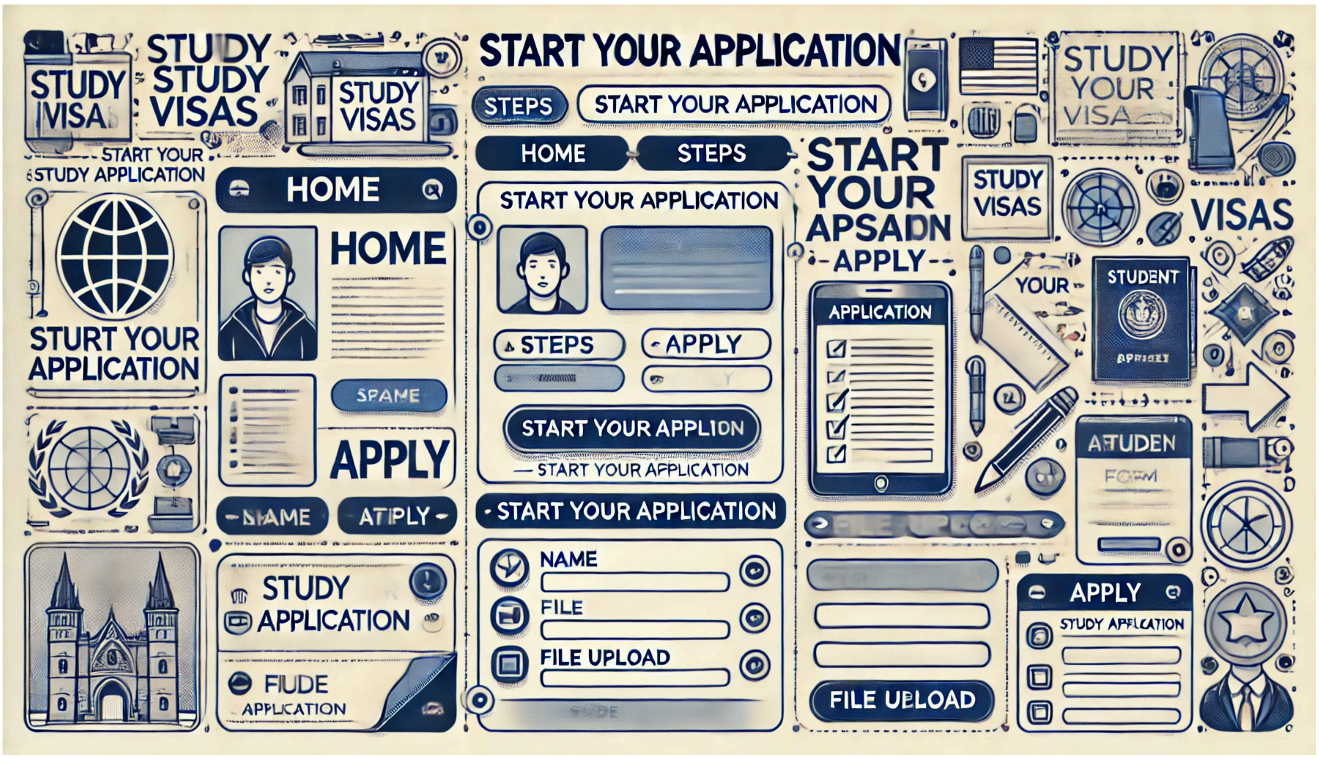
Visa Application Site.NEXT.JS



Steps to Create the Application:

1. ***Set Up the Project:**
   - Install Node.js and *npm* if not already installed.
   - Create a new Next.js project.
   - Set up the project `structure`.
2. ***Create Pages:**
   - Home page: `Introduction` and navigation.
   - Application page: Form for applying for a study visa.
   - Steps page: Detailed explanation of the `steps`.
3. ***Styling:**
   - Use CSS or a CSS-in-JS solution to style the pages.
4. ***Form Handling:**
   - Use state management to handle form data.
   - Implement form validation.
5. ***Deployment:**
   - Deploy the application using *Vercel* (the creators of Next.js) or another hosting service.

Step-by-Step Guide with Code:

1. **Set Up the Project:**

```shell
npx create-next-app@latest study-visa-app
cd study-visa-app
npm run dev
```

1. **Project Structure:**

```
study-visa-app/
├── pages/
│   ├── index.js
│   ├── apply.js
│   ├── steps.js
├── styles/
│   ├── globals.css
│   ├── Home.module.css
│   ├── Apply.module.css
│   ├── Steps.module.css
```

1. **Home Page (`pages/index.js`):**

```
import Link from 'next/link';
import styles from '../styles/Home.module.css';

export default function Home() {
  return (
    <div className={styles.container}>
      <h1>Welcome to the Study Visa Application Portal</h1>
      <p>Apply for a study visa and find detailed steps on how to
do so.</p>
      <nav>
        <ul>
          <li>
            <Link href="/apply">
              <a>Apply for Visa</a>
            </Link>
          </li>
          <li>
            <Link href="/steps">
              <a>Visa Application Steps</a>
            </Link>
          </li>
        </ul>
      </nav>
    </div>
  );
}
```

1. **Application Page (`pages/apply.js`):**

```
import { useState } from 'react';
import styles from '../styles/Apply.module.css';

export default function Apply() {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    passportNumber: '',
    country: '',
  });

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

  const handleSubmit = (e) => {
    e.preventDefault();
    // Handle form submission (e.g., send data to a server)
    console.log('Form submitted: ', formData);
  };

  return (
    <div className={styles.container}>
      <h1>Apply for a Study Visa</h1>
      <form onSubmit={handleSubmit}>
        <div className={styles.formGroup}>
          <label htmlFor="name">Name:</label>
          <input
            type="text"
            id="name"
            name="name"
            value={formData.name}
            onChange={handleChange}
            required
          />
        </div>
        <div className={styles.formGroup}>
          <label htmlFor="email">Email:</label>
          <input
            type="email"
```

```
              id="email"
              name="email"
              value={formData.email}
              onChange={handleChange}
              required
            />
          </div>
          <div className={styles.formGroup}>
            <label htmlFor="passportNumber">Passport Number:</label>
            <input
              type="text"
              id="passportNumber"
              name="passportNumber"
              value={formData.passportNumber}
              onChange={handleChange}
              required
            />
          </div>
          <div className={styles.formGroup}>
            <label htmlFor="country">Country:</label>
            <input
              type="text"
              id="country"
              name="country"
              value={formData.country}
              onChange={handleChange}
              required
            />
          </div>
          <button type="submit">Submit Application</button>
        </form>
      </div>
    );
  }
```

1. **Steps Page (`pages/steps.js`):**

```jsx
import styles from '../styles/Steps.module.css';

export default function Steps() {
  return (
    <div className={styles.container}>
      <h1>Study Visa Application Steps</h1>
      <ol>
        <li>Gather all required documents (passport, photos, proof of funds, etc.).</li>
        <li>Fill out the visa application form.</li>
        <li>Schedule an appointment at the nearest consulate or embassy.</li>
        <li>Attend the interview with all required documents.</li>
        <li>Pay the visa application fee.</li>
        <li>Wait for the visa processing (this can take several weeks).</li>
        <li>Receive your visa and prepare for your studies abroad.</li>
      </ol>
    </div>
  );
}
```

1. **Styling (e.g., `styles/Home.module.css`):**

```css
.container {
  max-width: 800px;
  margin: 0 auto;
  padding: 2rem;
  text-align: center;
}

nav ul {
  list-style-type: none;
  padding: 0;
}

nav ul li {
  display: inline;
  margin: 0 1rem;
}

nav ul li a {
  text-decoration: none;
  color: #0070f3;
  font-weight: bold;
}
```

Deployment:

To deploy your Next.js application, you can use Vercel, which is the easiest way to deploy a Next.js app:

1. Push your code to a GitHub repository.
2. Go to Vercel and sign up.
3. Import your project from GitHub.
4. Follow the prompts to deploy your application.

Complete Code in One File: `pages/index.js`

```
import { useState } from 'react';
import Link from 'next/link';
import Head from 'next/head';
import styles from '../styles/Home.module.css';

export default function Home() {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    passportNumber: '',
    country: '',
  });

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

  const handleSubmit = (e) => {
    e.preventDefault();
    // Handle form submission (e.g., send data to a server)
    console.log('Form submitted: ', formData);
  };

  return (
    <div>
      <Head>
        <title>Study Visa Application Portal</title>
        <meta name="description" content="Apply for a study visa
and find detailed steps on how to do so." />
      </Head>
      <main className={styles.container}>
        <h1>Welcome to the Study Visa Application Portal</h1>
        <p>Apply for a study visa and find detailed steps on how to
do so.</p>
        <nav>
          <ul className={styles.navList}>
            <li>
              <Link href="#apply">
                <a>Apply for Visa</a>
              </Link>
            </li>
            <li>
              <Link href="#steps">
                <a>Visa Application Steps</a>
              </Link>
            </li>
          </ul>
        </nav>

        <section id="apply" className={styles.section}>
          <h2>Apply for a Study Visa</h2>
          <form onSubmit={handleSubmit} className={styles.form}>
            <div className={styles.formGroup}>
              <label htmlFor="name">Name:</label>
              <input
                type="text"
                id="name"
                name="name"
                value={formData.name}
                onChange={handleChange}
                required
              />
            </div>
            <div className={styles.formGroup}>
              <label htmlFor="email">Email:</label>
              <input
                type="email"
                id="email"
                name="email"
                value={formData.email}
                onChange={handleChange}
                required
              />
            </div>
            <div className={styles.formGroup}>
              <label htmlFor="passportNumber">Passport Number:
</label>
              <input
                type="text"
                id="passportNumber"
                name="passportNumber"
                value={formData.passportNumber}
                onChange={handleChange}
                required
              />
            </div>
            <div className={styles.formGroup}>
              <label htmlFor="country">Country:</label>
              <input
                type="text"
                id="country"
                name="country"
                value={formData.country}
                onChange={handleChange}
                required
              />
            </div>
            <button type="submit" className=
{styles.submitButton}>Submit Application</button>
          </form>
        </section>

        <section id="steps" className={styles.section}>
          <h2>Study Visa Application Steps</h2>
          <ol>
            <li>Gather all required documents (passport, photos,
proof of funds, etc.).</li>
            <li>Fill out the visa application form.</li>
            <li>Schedule an appointment at the nearest consulate or
embassy.</li>
            <li>Attend the interview with all required documents.
</li>
            <li>Pay the visa application fee.</li>
            <li>Wait for the visa processing (this can take several
weeks).</li>
            <li>Receive your visa and prepare for your studies
abroad.</li>
          </ol>
        </section>
      </main>
    </div>
  );
}
```

Public code references from 5 repositories

Styling (e.g., `styles/Home.module.css`):

```css
.container {
  max-width: 800px;
  margin: 0 auto;
  padding: 2rem;
  text-align: center;
}

.navList {
  list-style-type: none;
  padding: 0;
}

.navList li {
  display: inline;
  margin: 0 1rem;
}

.navList li a {
  text-decoration: none;
  color: #0070f3;
  font-weight: bold;
}

.section {
  margin-top: 2rem;
  text-align: left;
}

.form {
  max-width: 600px;
  margin: 0 auto;
}

.formGroup {
  margin-bottom: 1rem;
}

.formGroup label {
  display: block;
  margin-bottom: 0.5rem;
}

.formGroup input {
  width: 100%;
  padding: 0.5rem;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.submitButton {
  background-color: #0070f3;
  color: white;
  padding: 0.5rem 1rem;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.submitButton:hover {
  background-color: #005bb5;
}
```

Steps to Run the Application:

1. **Install Node.js and npm**: Make sure you have Node.js and npm installed on your machine.

2. **Create a New Next.js Project**:

   ```shell
   npx create-next-app@latest study-visa-app
   cd study-visa-app
   ```

3. **Replace the Default Code**: Replace the contents of `pages/index.js` with the code provided above.

4. **Add the CSS Styles**: Create a file named `styles/Home.module.css` and add the styling code provided above.

5. **Start the Development Server**:

```
npm run dev
```

6. **View the Application**: Open your web browser and navigate to `http://localhost:3000` to view the application.

Visa Application Site Ts:

JS

```
 // Visa Application Site - Next.js Application in TypeScript

// 1. Setting up the Project
// Run the following commands to initialize the Next.js project:
// npx create-next-app@latest visa-application-site --typescript
// cd visa-application-site

// 2. Installing Dependencies
// Install the required dependencies:
// npm install tailwindcss@latest @prisma/client @types/node bcrypt
js-cookie next-auth react-hook-form axios

// 3. Configure Tailwind CSS
// Run the Tailwind setup command and configure the project:
// npx tailwindcss init

// tailwind.config.js
module.exports = {
 content: [
   './pages/**/*.{js,ts,jsx,tsx}',
   './components/**/*.{js,ts,jsx,tsx}',
 ],
 theme: {
   extend: {},
 },
 plugins: [],
};

// Import Tailwind CSS in styles/globals.css
@tailwind base;
@tailwind components;
@tailwind utilities;

// 4. Database Configuration
// Install Prisma:
// npx prisma init

// prisma/schema.prisma
// Example schema for users and applications
model User {
  id         Int      @id @default(autoincrement())
  email      String   @unique
  password   String
  name       String
  applications Application[]
}

model Application {
  id           Int      @id @default(autoincrement())
  userId       Int
  user         User     @relation(fields: [userId], references: [id])
  status       String   @default("Pending")
  createdAt    DateTime @default(now())
}

// Migrate the schema:
// npx prisma migrate dev --name init

// 5. User Authentication
// pages/api/auth/[...nextauth].ts
import NextAuth, { NextAuthOptions } from 'next-auth';
import CredentialsProvider from 'next-auth/providers/credentials';
import { PrismaClient } from '@prisma/client';
import bcrypt from 'bcrypt';

const prisma = new PrismaClient();

const authOptions: NextAuthOptions = {
  providers: [
    CredentialsProvider({
      name: 'Credentials',
      credentials: {
        email: { label: 'Email', type: 'text' },
        password: { label: 'Password', type: 'password' },
      },
      async authorize(credentials) {
        if (!credentials) return null; // Check if credentials are
provided
        console.log('Authorizing user with email:',
credentials.email);
        const user = await prisma.user.findUnique({
          where: { email: credentials.email }, // Find user by email
        });
```

```
       if (user && bcrypt.compareSync(credentials.password,
user.password)) {
         // Compare provided password with stored hash
         console.log('Authorization successful for user:',
user.email);
         return { id: user.id, email: user.email, name: user.name
}; // Return user data
       }
       console.warn('Authorization failed for email:',
credentials.email);
       return null; // Return null if authentication fails
     },
   }),
 ],
 session: {
   strategy: 'jwt', // Use JWT for session management
 },
 callbacks: {
   jwt: async ({ token, user }) => {
     if (user) {
       console.log('JWT callback: Adding user ID to token:',
user.id);
       token.id = user.id; // Add user ID to JWT payload
     }
     return token;
   },
   session: async ({ session, token }) => {
     if (token) {
       console.log('Session callback: Adding user ID to session:',
token.id);
       session.user = {  ...session.user, id: token.id }; // Add
user ID to session object
     }
     return session;
   },
 },
};

export default NextAuth(authOptions);

// 6. Application Submission Form
// components/ApplicationForm.tsx
import { useForm, SubmitHandler } from 'react-hook-form';
import axios from 'axios';

type FormData = {
 name: string; // User's name
 email: string; // User's email
};

export default function ApplicationForm() {
 const { register, handleSubmit } = useForm<FormData>(); //
Initialize form handling

 const onSubmit: SubmitHandler<FormData> = async (data) => {
   console.log('Submitting application with data:', data); // Log
submitted data
   try {
     await axios.post('/api/applications', data); // Send
application data to API
     console.log('Application submitted successfully');
     alert('Application submitted successfully!'); // Notify user
of success
   } catch (error) {
     console.error('Error submitting application:', error); // Log
errors
   }
 };

 return (
   <form onSubmit={handleSubmit(onSubmit)} className="space-y-4">
     <div>
       <label htmlFor="name">Name</label>
       <input {...register('name')} id="name" className="border p-
2 w-full" />
     </div>
     <div>
       <label htmlFor="email">Email</label>
       <input {...register('email')} id="email" type="email"
className="border p-2 w-full" />
     </div>
     <button type="submit" className="bg-blue-500 text-white px-4
py-2">Submit</button>
   </form>
 );
}
```

```
// 7. API Route for Application Submission
// pages/api/applications.ts
import { NextApiRequest, NextApiResponse } from 'next';
import { PrismaClient } from '@prisma/client';

const prisma = new PrismaClient();

export default async function handler(req: NextApiRequest, res:
NextApiResponse) {
 if (req.method === 'POST') {
   const { name, email } = req.body; // Extract name and email from
request body
   console.log('Received POST request to /api/applications with
body:', req.body);
   try {
     const application = await prisma.application.create({
       data: {
         user: {
           connectOrCreate: {
             where: { email }, // Check if user with this email
exists
             create: { email, name, password: 'placeholder' }, //
Create user if not exists
           },
         },
       },
     });
     console.log('Application created successfully:', application);
// Log success
     res.status(201).json(application); // Respond with the created
application
   } catch (error) {
     console.error('Error creating application:', error); // Log
errors
     res.status(500).json({ error: 'Failed to submit application'
}); // Respond with error
   }
 } else {
   console.warn('Invalid request method to /api/applications:',
req.method);
   res.status(405).json({ error: 'Method not allowed' }); //
Respond with method not allowed
```

```
    }
  }
```