Accounting Wiz: Chartered Accountants

1. Project Overview

Accounting Wiz is an accounting platform designed to showcase professional accountants and their services. It allows users to view profiles, rates, reviews, and contact accountants for their financial needs. Built with modern web technologies, it provides an interactive and user-friendly interface.

---

2. Introduction

Accounting Wiz is a web application designed to connect users with professional accountants. It provides an interface for users to browse accountant profiles, view detailed information, and get in touch with professionals for financial services.

This platform uses a **mock backend** implemented with **JSON Server** and a **mock database** (db.json), providing an API to simulate a real backend during local development.

---

3. Key Features

- **Browse Accountant Profiles**: View a list of accountants with details like rating, services, pricing, and task complexity.
- **Profile Details**: Access detailed information for each accountant, including services offered, client reviews, and contact information.
- **Interactive Navigation**: Easily navigate through different sections such as Home, Profiles, and Contact.
- **Responsive Design**: Fully responsive UI that works seamlessly across various devices (mobile, tablet, desktop).
- **Contact Form**: A form where users can send messages to accountants directly.

---

4. Technology Stack

Frontend

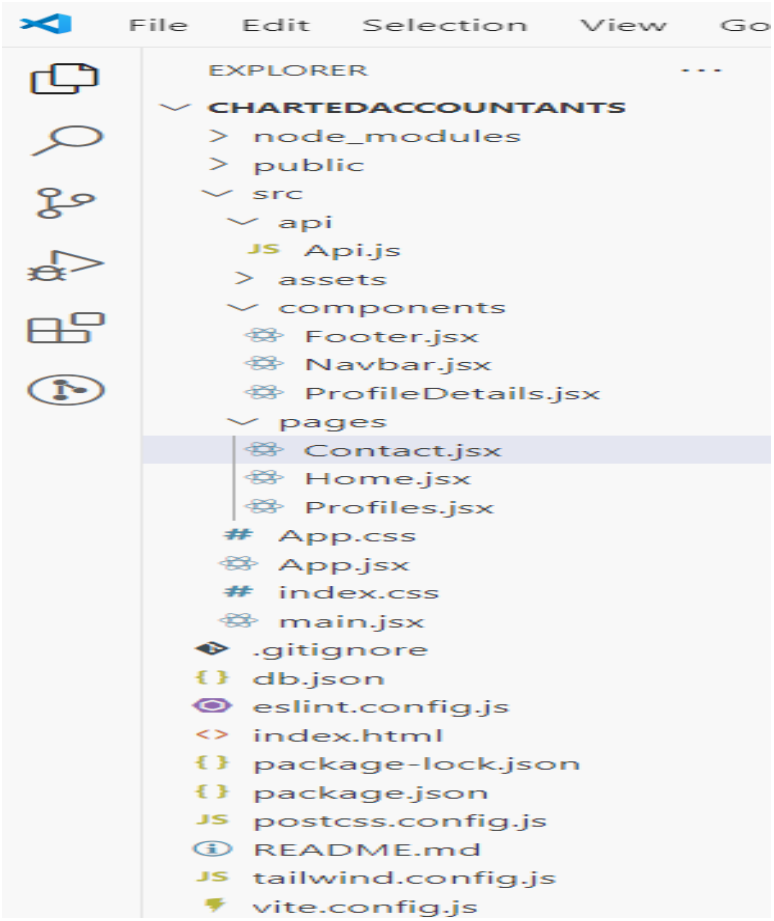
- **React**: For building reusable UI components and managing state.
- **Tailwind CSS**: For styling with utility-first CSS framework.
- **AOS (Animate On Scroll)**: For animations triggered on scroll events.
- **Axios**: For making HTTP requests to the backend.

Backend

- **JSON Server**: Used to simulate a RESTful API during local development with **db.json** as the mock database.
- **db.json**: A local file used as a mock database for storing accountant profiles.

---

5. File Structure



---

6. Backend API Interaction

Mock Database: db.json

The **db.json** file serves as a mock database containing a list of accountant profiles. This file is used by **JSON Server** to simulate backend data.

Example db.json structure:

```
{
  "Accounts" : [
  {
```

"id": 1,

"name": "Michael Jackson",

"image": "https://images.unsplash.com/photo-1567427017947-545c5f8d16ad?fit=crop&w=800&q=80",

"intro": "Expertise in accounting and finance, specializing in financial statements and auditing.",

"rating": 4.8,

"reviewCount": 89,

"taskComplexity": "Basic to complex tasks",

"price": "€4,370",

"deliveryTime": "Delivers within 2 days",

"testimonial": {

"text": "Exceptional service in managing personal finances and deep understanding of financial markets.",

"author": "John Doe"

},

"about": {

"from": "INDIA",

"partnerSince": 2011,

"averageResponseTime": "30 minutes",

"description": "Professional Chartered Accountant offering diverse accounting and financial services.",

"services": ["Financial accounting", "Bookkeeping", "Balance Sheets"],

"benefits": ["One-time delivery", "24/7 support"]

}

}

]

}

Endpoints for the Mock API

- **GET /Accounts**: Retrieves a list of all accountant profiles.
    - **URL**: http://localhost:3000/Accounts

    - **Method**: GET
    - **Response**: A JSON array containing all accountant profiles.
- **GET /Accounts/:id**: Retrieves details of a specific accountant by ID.
    - **URL**: http://localhost:3000/Accounts/:id
    - **Method**: GET
    - **Response**: A single accountant profile based on the provided id.

---

7. Components Overview

1. Navbar.jsx

The Navbar component provides a navigation bar with links to various sections of the application, such as Home, Profiles, and Contact. It also includes a responsive mobile menu that toggles visibility.

2. Footer.jsx

The Footer component displays quick links, social media icons, and contact information.

3. ProfileDetails.jsx

The ProfileDetails component displays detailed information about an individual accountant, including their name, services, benefits, and reviews.

4. Contact.jsx

The Contact component allows users to submit a contact form, sending their name, email, and message. It includes basic validation for form fields.

---

8. API Management

API Calls

The **API.js** file manages all API calls to the mock backend (JSON Server). It uses **Axios** to make **GET** requests to fetch the list of accountants or the details of a single accountant.

Example of API calls in Api.js:

```
import axios from "axios";

const baseURL = "http://localhost:3000/Accounts";

// Fetch all accountants

export const fetchAccountants = async () => {

  try {
```

```
    const response = await axios.get(baseURL);
    return response.data; // Returns the list of accountants
  } catch (error) {
    console.error('Error fetching accountants:', error);
    throw error;
  }
};
// Fetch details of a specific accountant by ID
export const fetchAccountantDetails = async (id) => {
  try {
    const response = await axios.get(`${baseURL}/${id}`);
    return response.data; // Returns details of a specific accountant
  } catch (error) {
    console.error('Error fetching accountant details:', error);
    throw error;
  }
};
```

Mock Backend: db.json

db.json is the mock database used during local development. **JSON Server** serves as the mock API server that reads from this file and returns data for the frontend to consume.

To run the mock server:

```
json-server --watch db.json --port 3000
```

---

9. Key Functionality

- **Fetch Accountant Data**: The frontend interacts with the mock API to fetch the list of accountants and their details.
- **Display Profiles**: The profile details, including images, ratings, and services, are displayed dynamically based on the data retrieved from the mock API.
- **Responsive UI**: The app's UI adjusts automatically based on the device's screen size, ensuring a seamless experience across mobile and desktop.

- **Form Validation**: The contact form includes client-side validation to ensure users provide correct data.

---

10. Future Enhancements

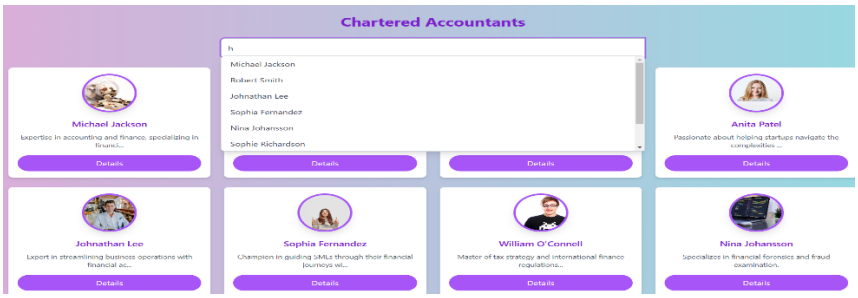
- **Backend Integration**: Replace the mock API with a real backend service to store and manage accountant data.
- **Authentication**: Add user authentication for saving and managing accountant profiles.
- **Advanced Search and Filter**: Implement filtering and sorting capabilities to search accountants based on rating, services, and location.
- **Admin Panel**: Provide an interface for admins to manage the accountant profiles, including CRUD operations.

---

11. Screenshots and UI Preview

Here are some screenshots showcasing the **Accounting Wiz** web app:

- **Home Page**: About accountants Wiz and displayed with basic information.
- **Profiles Page**: Detailed information about a selected accountant.



- **Contact Form**: A form for users to send messages to accountants.

Conclusion:

Accounting Wiz is a simple yet effective platform for browsing and interacting with professional accountants. With its modern stack and the use of mock backend during development, it is easy to test and iterate upon the application's frontend.

Github Repository: https://github.com/Nandini1207/Accounting-Wiz.git