Project Title

Summary

Our project, "Smart Home Automation System," aims to revolutionize the way we interact with our living spaces by integrating cutting-edge IoT technology. The system enables users to control and monitor home appliances remotely via a mobile application, providing convenience, energy efficiency, and enhanced security. Key features include real-time energy consumption tracking, customizable automation schedules, and seamless integration with popular voice assistants. Key Features: - Remote control of home appliances through a mobile app - Real-time energy consumption monitoring - Customizable automation schedules for various devices - Integration with voice assistants like Amazon Alexa and Google Assistant - Enhanced security with real-time alerts and monitoring

Folder Structure

sign
. . .project_Summary.pdf
. . .src
. . .. . .index.js
. . .. . .index.css
. . .. . .components
. . .. .Navbar.js
. . .. .Carousel.js
. . .. . .firebase
. . .. .authsetup.js
. . .. .config.js
. . .. . .pages
. . .. .About.js
. . .. .Home.js
. . .. .Login.js
. . .. .Nopage.js
. . .. .Contact.js

Code Summaries

■ :sign
project_Summary.pdf
Not a code file

. . .■ :src
index.js
- Imported React from 'react' to enable the use of React library in the code
- Imported ReactDOM from 'react-dom/client' to render React elements into the DOM
- Imported './index.css' to apply styling to the components in the App
- Imported App component from './App' to render the main application component
- Imported reportWebVitals function from './reportWebVitals' to measure performance metrics in the app

- Created a root element using ReactDOM.createRoot(document.getElementById('root')) to render the React elements into the DOM
- Rendered the component inside a component for better debugging and warning messages
- Used the reportWebVitals function to measure performance metrics in the app and log the results or send them to an analytics endpoint.

index.css
- The code includes imports for Tailwind CSS base, components, and utilities, which are used for styling the project.
- The code also imports two Google Fonts, Cormorant Infant and Josefin Sans, with different weights and display settings.
- The fonts imported are used for styling text within the project.
- Tailwind CSS classes are used throughout the project for consistent and efficient styling of elements.

. . .. . .■ :components
Navbar.js
- Imports:
- React from "react"
- Outlet, Link from "react-router-dom"
- Login from "../pages/Login"
- Logo from '../assets/Logo.png'
- Functionality:
- Navbar component that takes two props: getdets and userCred
- Contains a list of navigation items with names and links
- Displays a navigation bar with a logo, navigation items, and a login button if userCred is null
- If userCred is not null, it displays the user's display name instead of the login button
- Clicking on a navigation item scrolls to the corresponding section on the page
- Includes a conditional rendering for the login button based on the userCred prop
- Uses the Outlet component from react-router-dom for nested routing.

Carousel.js
- Imports:
- React, { useEffect, useState } from 'react'
- Swiper, SwiperSlide from "swiper/react"
- "swiper/css"
- "swiper/css/effect-coverflow"
- "swiper/css/pagination"
- { EffectCoverflow, Autoplay, Pagination } from "swiper/modules"
- Functionalities:
- The Carousel component takes in a prop 'sPV' which determines the number of slides per view
- The component uses Swiper library to create a carousel with coverflow effect
- It imports images from assets folder to display in the carousel
- The useEffect hook is commented out, but it was intended to handle scroll events for animations
- The Swiper component is configured with coverflow effect, autoplay, and pagination
- The images are mapped over to create individual slides within the carousel
- The Swiper component is rendered with the specified configurations and images, adjusting size based on screen width

. . .. . .■ :firebase

authsetup.js
- The code imports the Firebase and FirebaseUI libraries using the 'require' method.
- It initializes a new instance of the AuthUI class from the Firebase Auth module.
- The code then starts the FirebaseUI authentication flow by specifying the sign-in options, which include email and Google authentication methods.
- It checks if there is a pending email link sign-in and starts the authentication flow if true.
- It also checks if the current URL contains an email link sign-in and starts the authentication flow if true.

config.js
- Imported necessary functions from Firebase SDKs such as initializeApp, getAnalytics, getAuth, GoogleAuthProvider, signInWithPopup, setPersistence, signInWithRedirect, inMemoryPersistence
- Firebase configuration details like apiKey, authDomain, databaseURL, projectId, storageBucket, messagingSenderId, appId, and measurementId are provided in the firebaseConfig object
- Initialized Firebase app using initializeApp function with firebaseConfig
- Retrieved analytics instance using getAnalytics function
- Retrieved auth instance using getAuth function
- Comments are provided for setPersistence and signInWithRedirect functions, which are currently not used in the code

. . .. . .■ :pages
About.js
- Imports:
- Imported img2 from "../assets/about.jpeg"
- Imported ScrollAnimation from "react-animate-on-scroll"
- Imported Pattern from "../assets/Pattern.svg"
- Imported useEffect, useLayoutEffect, useState from "react"
- Functionality:
- The About component is responsible for displaying information about the company or website.
- It uses state variables writerimg, pattern, and aboutus to control animations and styling.
- The useEffect hook is used to trigger animations when the user scrolls to a certain position on the page.
- The handleScroll function is called when the user scrolls, and it updates the state variables to reveal animations.
- The component includes HTML elements to display text and images related to the company or website.
- The component also includes a hidden pattern image that becomes visible with a spinning animation when the user scrolls to a specific position.
- The writerimg state variable controls the visibility and animation of an image displayed on the page.
- The aboutus state variable controls the styling of a text section displayed on the page.

Home.js
- Imports:
- Import img1 from "../assets/img1.jpg"
- Import About from "./About"
- Import Benefits from "./Benefits"
- Import Contact from "./Contact"
- Functionality:
- Renders the Home component with user credentials passed as props
- Displays a background image with text content promoting journaling
- Includes a button linking to the Instagram page for more information

- Displays an image sourced from img1 variable
- Renders the About, Benefits, and Contact components below the main content.

Login.js
- Imported useEffect from react for managing side effects in functional components
- Imported getAuth and onAuthStateChanged from firebase auth sdk for authentication functionalities
- Imported firebase from firebase/compat/app for compatibility with older versions of firebase
- Imported firebaseui and firebaseui styles using the CDN for pre-built UI for firebase authentication
- Imported app and auth from ../firebase/config for firebase configuration
- Imported GoogleAuthProvider from firebase/auth for Google authentication functionality
- Used useEffect hook to initialize the Firebase authentication UI
- Defined a uiConfig object with callbacks for successful sign-in and UI rendering
- Used signInFlow: 'popup' for sign-in using a popup instead of redirect
- Added GoogleAuthProvider as a sign-in option with the clientID provided
- Included terms of service and privacy policy URLs in the uiConfig object
- Started the Firebase UI authentication using ui.start method with the uiConfig and container element
- Rendered a login form with Firebase UI elements in the JSX return statement

Nopage.js
Project Documentation:
- This code includes a function called Nopage that returns a React component.
- The function imports React from the 'react' library.
- The component renders a simple div with the text "Nopage".
- The Nopage component is then exported as the default export of the file.

Contact.js
- This code imports a logo from the assets folder.
- The Contact function creates a component that displays contact information.
- It defines an array of contact buttons with labels "Home", "About", and "Contact Us".
- It also defines an array of SVG paths for social media icons.
- Another array contains URLs for social media profiles.
- The component renders a logo, a list of contact buttons, and social media icons with links.
- It also includes a copyright notice at the bottom of the component.
- The component uses Tailwind CSS classes for styling.

Benefits.js
- Imports:
- Import Carousel component from "../components/Carousel"
- Import useEffect and useState from "react"
- Functionality:
- The Benefits component includes a useEffect hook that triggers an animation when the viewer is at a certain scroll position on the page.
- The useEffect hook listens for scroll events and updates the state variable benefitsClass accordingly to apply the animation class.
- The component renders a section titled "BENEFITS" with a heading "Benefits of Journaling" when the animation is triggered.
- The component conditionally renders a Carousel component based on the screen size, with different slide per view values.

App.js
- The code imports necessary modules from react-router-dom, including BrowserRouter, Routes, and Route for setting up routing in the application.
- It also imports components like Home, Navbar, Nopage, Login, and auth from specific directories for use in the project.
- The code sets up a function called getdets, which uses signInWithPopup method from firebase/auth to authenticate user credentials using GoogleAuthProvider.
- useState hook is used to manage user credentials in the state.
- The main App component is created, which renders different routes based on the URL path.
- The Navbar component is rendered with props like getdets and userCred.
- The Home component is rendered as the default route.
- The Login component is rendered for the '/login' path.
- The Nopage component is rendered for any other path not defined in the routes.
- Finally, the App component is exported as the default export.