GEE'Z BROKER

Developer Manual

Hello Developer Welcome to GEEZ BROKER!

This Broker Site is composed of the following:

- Folders
  - Admin
  - Component
  - CSS
  - Database
  - Image
  - IMG
  - JS
  - Shape
  - Uploaded_file
- Files:
    - About.php
    - Contact.php
    - Dashbors.php
    - Home.php
    - Index.php
    - Login.php
    - My_listing.php
    - Post_property.php
    - Register.php
    - Requests.php
    - Saved.php
    - Search.php
    - Update.php
    - Update_property.php
    - View_property.php

NO	Folder Name	Description
1	admin	• **Admins.php** This PHP script establishes an admin interface where authenticated users can perform CRUD operations on administrator accounts. It ensures secure authentication, enables searching for admins by name, and offers options for updating existing accounts or registering new ones. Additionally, it incorporates feedback mechanisms to inform users of successful actions or empty search results. • **Dashboard.php**

		This PHP script creates a user interface for an admin dashboard, providing insights and navigation options for managing a web application. It fetches admin information from a database, displays it along with a menu for toggling themes and user profile details. The dashboard presents key metrics such as property listings, total users, admins, and new messages, retrieved from corresponding database tables. Additionally, it includes date selection functionality and links to various admin management sections. • **Dashboard1.php** This PHP script generates an admin dashboard interface, displaying various metrics and providing navigation links for managing a web application. It fetches admin information from a database and dynamically populates boxes with metrics such as the number of property listings, total users, admins, and new messages. Each metric box includes a link to its corresponding management section for easy access and navigation. • **Listings.php** This PHP script manages property listings in an admin panel. It includes functionality for deleting listings and searching for specific properties. The script retrieves admin credentials from cookies and redirects to the login page if not authenticated. It also dynamically displays property information fetched from a database, including images, price, name, and address. Each listing is presented in a box format with options to view the property details or delete the listing. The layout is designed to be responsive and visually appealing, with CSS styles for various elements. • **Login.php** This PHP script handles the login functionality for admin users. It checks the submitted username and password against records in the database. If the credentials match, it sets a cookie to remember the user's session and redirects them to the dashboard page. Otherwise, it displays a warning message for incorrect credentials. The login form includes fields for entering the username and password, with client-side validation to remove any white spaces. The layout is styled using CSS for a clean and user-friendly interface. • **Message.php** This PHP script manages the messages received from users. It retrieves messages from the database and displays them in a grid layout. Each message is shown in a box containing the sender's name, email, phone number, and message content. Below each message, there's a button to delete the message. Upon clicking the delete button, it prompts the user to confirm the deletion before proceeding. The script also includes functionality for searching messages based on name, email, or phone number. It allows users to enter search queries in a form input field, and upon submission, it filters the messages accordingly. The layout is styled using CSS to ensure a visually appealing and organized presentation of messages. The design follows a clean and minimalist approach for better readability and user experience. • **Register.php** This PHP script handles the registration of new admin users. It ensures that each admin has a unique username and properly matches the password with its confirmation before registering. Upon submission of the registration form, the script checks if the username is already taken. If not, it verifies that the password matches its confirmation. If all conditions are met, it inserts the new admin's details into the database and displays a success message. The layout of the registration form follows a clean and user-friendly design. It provides input fields for the username, password, and password confirmation.

Additionally, it includes styling to enhance the visual presentation of the form for better readability and user experience.

- **Update.php**

This PHP script allows an admin user to update their profile information, including their username and password. Let's break down the functionality:

1. **Profile Retrieval**: It retrieves the admin's profile information from the database based on their admin ID retrieved from the cookie.

2. **Username Update**: If the user submits a new username, it checks if the username is already taken. If not, it updates the username in the database.

3. **Password Update**: It verifies the old password entered by the user against the one stored in the database. If they match, it checks if the new password and its confirmation match and updates the password in the database.

The layout of the update form provides input fields for the username, old password, new password, and password confirmation. It ensures that inputs are sanitized and passwords are properly matched before any updates are made to the database. Additionally, it displays warning or success messages based on the outcome of the update operation.

- **Users.php**

This PHP script is for managing user accounts in an admin panel. Here's a breakdown of its functionality:

1. User Deletion: When the admin submits the delete button for a user, it first verifies if the user exists in the database. If the user exists, it proceeds to delete associated data such as property listings, requests, and saved items. Then it deletes the user account itself. If the user doesn't exist, it displays a warning message.

2. User Listing: It retrieves user information from the database and displays it in a grid layout. For each user, it shows their name, contact number, email, and the number of properties they have listed. It provides a delete button for each user.

3. Search Functionality: It allows the admin to search for users by name, contact number, or email. The search results are dynamically updated without refreshing the page.

The layout is designed to be responsive and user-friendly, with each user's information displayed in a box format. There's also confirmation before deleting a user to prevent accidental deletions. Additionally, it handles cases where no users are found or if there are no user accounts added yet.

- **View_property.php**

Your PHP code appears to be a part of an admin dashboard for managing users and property listings. Let's break down what each section does:

1. Initialization and Authentication:

- The code starts by including the connection file and checking if the admin is logged in. If not, it redirects to the login page.
- It also checks for a specific property ID (`get_id`) from the URL. If it's not provided, it redirects to the dashboard.

2. Property Deletion:

- When the admin submits a delete request (`POST` request), it sanitizes the property ID and verifies if the property exists in the database.
- If the property exists, it retrieves its images and deletes them from the server.
- Then it deletes the property from the database and displays a success message.

		3. HTML Output: - The HTML section displays property details such as images, name, location, price, contact details, amenities, and description. - It also provides a form to delete the property, which triggers the deletion process described above. **4. Styling:** - The CSS section defines styles for various elements, including fonts, colors, layout, and media queries for responsiveness. Overall, the code provides functionality for viewing and deleting property listings from the admin dashboard. It utilizes PHP for server-side logic and HTML/CSS for front-end presentation and styling. Additionally, it incorporates Swiper.js for image carousel functionality.
2	Component	• **Admin_header.php** This HTML code defines a header section with a fixed position on the left side of the viewport. Within the header, there's a container holding an aside element, which contains the logo and close button at the top, followed by a sidebar navigation menu. The sidebar includes links to different pages such as the dashboard, listings, users, admins, messages, and an option to add an admin or logout. Additionally, there's a menu button with a hamburger icon positioned outside the header. Overall, this code creates a user interface for navigating various sections of a web application. • **Admin_logout.php** This PHP script includes a file named "connect.php" and then proceeds to unset the value of the "admin_id" cookie by setting its expiration time to a past value. This effectively deletes the cookie. After deleting the cookie, the script redirects the user to the "index.html" page located in the parent directory using the header function. In summary, this script is responsible for logging out an admin user by deleting their session cookie and redirecting them to the homepage. • **Connect.php** This PHP script establishes a connection to a MySQL database using PDO (PHP Data Objects). It defines variables for the database name, username, and password. Then, it creates a PDO object named "$conn" using the database name, username, and password provided. Additionally, the script defines a function named "create_unique_id()" that generates a random string of characters. This function creates a unique identifier by randomly selecting characters from a set of alphanumeric characters. The length of the generated string is set to 20 characters. Finally, the function returns the generated random string. • **Footer.php** This HTML code defines a footer section for a web page. It includes branding elements like a logo and tagline, along with social media icons for engagement. Additionally, it features contact details such as phone number and email, along with a link to the admin login page. The copyright information is also displayed at the bottom. • **Message.php** This PHP code block checks for different types of messages (success, warning, info, and error) and displays them using SweetAlert, a JavaScript library for creating alert modals. It loops through each type of message array and generates the corresponding SweetAlert modal to inform the user about the messages. • **Save_send.php** This PHP script manages saving listings and sending requests for properties. **For the "save" action:** - It verifies if the user is logged in. - If the user is authenticated, it generates a unique ID for the saved item and

		retrieves the property ID from the form. - The script checks if the property is already saved by the user. If it is, the property is removed from the saved list; otherwise, it's added. - Appropriate success or warning messages are displayed based on the action taken. **For the "send" action:** - It performs a similar authentication check to ensure the user is logged in. - After authentication, it generates a unique ID for the request and retrieves the property ID from the form. - The script fetches the ID of the property owner to whom the request will be sent. - It verifies if a request for the same property from the same sender to the same receiver already exists. If not, the request is sent. - Relevant success or warning messages are displayed depending on the outcome of the action. • **User_header.php** This HTML code defines a header section for a website. It consists of two navigation bars, each serving different purposes. In the first navigation bar (nav-1), the website logo ("GeezBroker") is displayed along with a link to the home page. Additionally, there is a link to "post property" action. The second navigation bar (nav-2) includes a menu button and a dropdown menu with three main categories: "my listings", "options", and "help". Each category contains sub-items that expand when hovered over or clicked. Additionally, there are links to the "saved" section and the user's account, which includes options such as updating the profile and logging out, depending on whether the user is logged in or not. • **User_header2.php** This HTML code defines a header section for a website with a fixed position. It contains a navigation bar (nav-1) with a semi-transparent background color. The navigation bar includes the website logo ("GeezBroker") and a link to the home page ("Go To Home Page"). The logo is styled with a larger font size, bold weight, and custom font family. • **User_logout.php** This PHP code segment is responsible for logging out a user. It includes a connection to a database using the `connect.php` file. Then, it removes the user's cookie named `"user_id"` by setting its value to an empty string and setting its expiration time to a time in the past (effectively deleting it). Finally, it redirects the user to the `index.html` page using the `header()` function. This action effectively logs out the user by removing their session information and redirecting them to the home page.
3.	CSS	• **Admin_style.css** This CSS code defines various styling properties such as colors, fonts, and layouts for a web page. It uses custom variables for colors and other commonly used values to maintain consistency and ease of maintenance. The `@import` rule is used to import a font from Google Fonts. The `:root` selector is used to define global CSS variables. The `*` selector is used to apply styles to all elements on the page. The `html` and `body` selectors define styles for the overall structure and layout of the page. Selectors like `.header`, `.navbar`, `.box`, `.btn`, etc., define styles for specific components or elements within the page. Media queries are used to apply different styles based on the viewport size, making the design responsive. Overall, this CSS code provides a clean and modern styling for a web page, with attention to detail and responsiveness across different devices.

		• **Dstayle.css** • **Hstayle.css** • **Style.css**
4.	Database	• **Home_db** This SQL script creates several tables for a database named `home_db`. Here's a breakdown of each table: 1. ***admins***: This table stores information about administrators. It includes fields for `id` (unique identifier), `name`, and `password`. 2. ***messages***: This table is used for storing messages submitted by users. It contains fields for `id`, `name`, `email`, `number`, and `message`. 3. ***property***: This table holds details about properties. It includes fields such as `id`, `user_id` (owner's ID), `property_name`, `address`, `price`, `type`, `offer`, and various amenities and features of the property. 4. ***requests***: This table is for storing requests made by users for property viewings or other actions. It includes fields for `id`, `property_id`, `sender` (sender's ID), `receiver` (receiver's ID), and `date`. 5. ***saved***: This table is used to save properties that users have bookmarked or saved for later viewing. It contains fields for `id`, `property_id`, and `user_id`. 6. ***users***: This table stores information about users. It includes fields for `id`, `name`, `number`, `email`, and `password`. Each table has its own unique primary key (`id`) to ensure data integrity and facilitate efficient querying. Additionally, default timestamps are provided for the `date` fields using `current_timestamp()`. Overall, this script sets up the basic structure for a database intended to manage properties, users, administrators, messages, requests, and saved properties. If you have any specific questions or need further clarification on any aspect of the script, feel free to ask!
5.	Images	Some of the image we used are placed here
6.	Img	Also there!
7.	JS	• **Admin_script.js** This JavaScript code snippet controls the behavior of a header element and input fields in a web page. Here's a breakdown of what each part does: **1. Header Control:** - It selects the header element using the class `.header`. - When the element with the ID `menu-btn` is clicked, it adds the class `active` to the header, presumably to open a menu. - When the element with the ID `close-btn` is clicked, it removes the class `active` from the header, presumably to close the menu. - When the window is scrolled, it removes the class `active` from the header. This might be intended to close the menu when the user scrolls. **2. Input Field Control:** - It selects all input fields of type "number". - For each input field, it sets up an `oninput` event listener. - When the user inputs a value, it checks if the length of the value exceeds the `maxLength` attribute of the input field. - If it does, it truncates the value to the maximum length specified by `maxLength`. Overall, this script provides interactivity by toggling the visibility of a menu in the header based on button clicks and scrolling. Additionally, it ensures that input fields of type "number" do not exceed their maximum length specified by the `maxLength` attribute. If you have any further questions or need clarification, feel free to ask! • **D.js** • **Dscript.js** • **Hscript.js**

		• **Script.js**
8.	Shapes	They are a part of our CSS.
9.	Uploaded_files	All the uploaded file will be saved to this directory