Application Overview

This FastAPI application serves as a backend for managing an Electric Vehicle (EV) database. It features functionalities for user authentication, adding, editing, and comparing EVs, and submitting reviews for EVs. The application utilizes Google Firestore for data persistence and Firebase for authentication.

FastAPI Application Initialization

FastAPI Instance: The core of the web application, responsible for handling incoming HTTP requests and routing them to the appropriate handler functions.

app = FastAPI()

Firestore Client Initialization: Connects to Google Firestore, a NoSQL database used to store EV data, user profiles, and reviews.

firestore_db = firestore.Client()

Firebase Request Adapter: Configures the request adapter for Firebase, enabling the application to verify Firebase authentication tokens.

firebase_request_adapter = requests.Request()

Functionality Overview

User Authentication and Management

validateFirebaseToken(id_token): Validates the Firebase authentication token to authenticate users.

getUser(user_token): Retrieves or creates a user document in Firestore based on the provided Firebase user token.

EV Data Management

add_ev_page(request): Renders the page for adding a new EV to the database, available only to authenticated users.

add_ev(request): Processes the submission of a new EV, adding it to the Firestore database.

edit_ev_page(request, ev_id): Displays the edit page for a specific EV, populated with its current data from Firestore.

submit_edit_ev(request, ev_id): Updates the specified EV's data in Firestore based on form submission.

delete_ev(ev_id): Removes the specified EV from the Firestore database.

show_ev(request, ev_id): Displays detailed information about a specific EV, including reviews.

EV Comparison

compare_evs(request): Compares two selected EVs based on their attributes, such as battery size and cost, and displays the comparison results.

Review Management

add_review(request, ev_id): Allows authenticated users to submit reviews for a specific EV.

get_ev_reviews(ev_id): Retrieves all reviews for a specific EV from Firestore.

Detailed Method Documentation

getUser(user_token)

Purpose: Retrieves an existing user document from Firestore or creates a new one if it doesn't exist.

Parameters:

user_token: A dictionary containing the user's Firebase token information.

Returns: A Firestore document reference to the user.

validateFirebaseToken(id_token)

Purpose: Validates the provided Firebase ID token to authenticate users.

Parameters:

id_token: The Firebase ID token obtained from the client.

Returns: A dictionary containing the decoded token information if valid, or None if invalid.

add_ev_page(request)

Purpose: Renders the page for adding a new electric vehicle, accessible only by authenticated users.

Parameters:

request: The request object containing HTTP request information.

Returns: An HTML response rendering the add EV page.

add_ev(request)

Purpose: Processes form data to add a new electric vehicle to the Firestore database.

Parameters:

request: The request object containing HTTP request information, including form data.

Returns: A redirect response to the main page upon successful addition of the EV.

edit_ev_page(request, ev_id)

Purpose: Displays the edit page for an existing electric vehicle, pre-populated with its data.

Parameters:

request: The request object.

ev_id: The unique identifier of the electric vehicle to edit.

Returns: An HTML response rendering the edit EV page.

submit_edit_ev(request, ev_id)

Purpose: Updates the data of an existing electric vehicle in the Firestore database based on form submission.

Parameters:

request: The request object containing updated EV data.

ev_id: The unique identifier of the electric vehicle to update.

Returns: A redirect response to the main page upon successful update.

delete_ev(ev_id)

Purpose: Removes an electric vehicle from the Firestore database.

Parameters:

ev_id: The unique identifier of the electric vehicle to delete.

Returns: A redirect response to the main page upon successful deletion.

show_ev(request, ev_id)

Purpose: Displays detailed information about a specific electric vehicle, including user reviews.

Parameters:

ev_id: The unique identifier of the electric vehicle to display.

Returns: An HTML response rendering the EV details page, including reviews if any.

compare_evs(request)

Purpose: Performs a comparison between two selected electric vehicles based on various attributes like battery size, cost, and power. It highlights the advantages of each vehicle in the comparison.

Parameters:

request: The request object containing the IDs of the two EVs to be compared.

Returns: An HTML response rendering the comparison results, highlighting differences between the two EVs.

add_review(request, ev_id)

Purpose: Allows authenticated users to submit reviews for a specific electric vehicle, including a textual review and a numerical rating.

Parameters:

request: The request object containing the review content and rating.

ev_id: The unique identifier of the electric vehicle being reviewed.

Returns: A redirect response to the EV details page, including the new review.

get_ev_reviews(ev_id)

Purpose: Fetches all reviews associated with a specific electric vehicle from the Firestore database, to be displayed on the EV's details page.

Parameters:

ev_id: The unique identifier of the electric vehicle whose reviews are to be fetched.

Returns: A list of review documents, each containing the review content, rating, and the date it was posted.

Additional Notes

The application uses Jinja2 for HTML template rendering, allowing for dynamic content presentation based on data fetched from Firestore.

Firebase Authentication is used to secure routes that require user authentication, such as adding or editing EVs and submitting reviews.

The application is designed to be scalable, with Firestore allowing for real-time data updates and efficient querying capabilities.

Conclusion

This documentation offers a comprehensive look at the key features of **main.py**, focusing on how it supports the management of an electric vehicle (EV) database. It breaks down the roles and parameters of each function, emphasizing areas such as user authentication, managing EV information, handling reviews, and comparing EVs. These elements are crucial to the app's functionality. The goal of this documentation is to make the application's backend architecture clear and understandable, which should help with its ongoing maintenance and future development efforts.