תיק מתכנת

CryptoPulse – B5 group

Introduction:

"CryptoPulse" emerges as an innovative news aggregator and a comprehensive cryptocurrency trading platform, crafted to empower users within the dynamic realm of digital currencies. It's not just a source of information but also a gateway to action, offering a fully integrated wallet feature that allows users to buy and sell a wide array of cryptocurrencies directly within the app.

CryptoPulse is more than an application—it's an indispensable companion for anyone looking to stay connected to the cryptocurrency market's pulse and engage with it directly through a trusted and versatile trading tool. Whether charting a course through the turbulent seas of market volatility or sailing the calmer waters of long-term investment, CryptoPulse equips you with the necessary resources to navigate the cryptocurrency ecosystem with assurance and agility.

Main Functions:

* Frontend Main Functions:

1. fetchCryptoCurrenciesFromFirestore:
   * Purpose: Fetches cryptocurrency data from a Firestore collection named "cryptocurrencies".
   * Process: It retrieves all documents from the "cryptocurrencies" collection. It filters out a document with the id "updatedTimeStamp" to store its timestamp separately. The rest of the documents are added to an array of currencies.
   * Return: Returns an object containing an array of currencies and the last update timestamp.
2. fetchCryptoNewsFromFirestore:
   * Purpose: Fetches cryptocurrency news data from a Firestore collection named "cryptonews".
   * Process: Similar to fetching cryptocurrencies, it retrieves all documents from the "cryptonews" collection, excluding the "updatedTimeStamp" document, to compile an array of news items and an updated timestamp.
   * Return: Returns an object with an array of news items and the last update timestamp.
3. createNewUser:
   * Purpose: Creates a new user document in Firestore within the "users" collection using provided user and form data.
   * Process: Adds a document with the user's UID as the document ID, containing user information such as UID, email, phone number, display name, photo URL, and creation timestamp.
   * Return: Does not return a value but logs errors to the console if any occur.
4. fetchUserDataFromFirestore:
   * Purpose: Fetches comprehensive user data from Firestore, including data from the "wallet" and "transactions" subcollections under the user's document.
   * Process: Retrieves the main user document, then fetches data from both "wallet" and "transactions" subcollections, combining all data into a single object.
   * Return: Returns a combined data object containing the user's information, wallet, and transaction data, or null if an error occurs.
5. addCryptoToTheWallet:
   * Purpose: Adds a specified amount of a cryptocurrency to a user's wallet in Firestore and records the transaction.
   * Process: Uses a transaction to atomically update the user's wallet with the new amount of a specified currency and records the transaction details, including credit card details (with only the last four digits of the card number saved) in the "transactions" subcollection.
   * Return: Logs a success message or error to the console.
6. tradeCrypto:
   * Purpose: Trades one cryptocurrency for another in the user's wallet and records the transaction.
   * Process: Checks for sufficient funds, then updates the amounts of the sold and bought currencies in the wallet and records the trade transaction in the "transactions" subcollection.
   * Return: Logs a success message, an insufficient funds message, or an error to the console.
7. withdrawCrypto:
   * Purpose: Withdraws a specified amount of cryptocurrency from the user's wallet to a bank account and records the transaction.
   * Process: Checks for sufficient cryptocurrency funds, updates the wallet, and records the withdrawal transaction, including bank account details, in the "transactions" subcollection.
   * Return: Logs a success message, an insufficient funds message, or an error to the console.
8. deleteUserAccount:
   * Purpose: Deletes the user's account and all associated data from Firestore and Firebase Authentication.
   * Process: Runs a transaction to delete the user's Firestore document and then deletes the user's authentication record.
   * Return: Does not return a value but performs deletion operations.
9. updateUserProfile:
   * Purpose: Updates the user's profile data in Firestore and Firebase Authentication, optionally including a password change.
   * Process: Optionally re-authenticates the user if a password change is requested, then updates the user's Firestore document and authentication profile with the new information.
   * Return: Logs a success message or error to the console.
10. sortData:
    * Purpose: This function is designed to sort an array of cryptocurrency data (cryptoCurrenciesData.filterdData) based on a specified attribute (field) and manage the display of sorted data considering pagination. It supports sorting in both ascending and descending orders, with the ability to toggle the sorting order upon consecutive invocations with the same field.
    * Process:
      + Toggle Check: Initially, it checks if the current sorting field (sort.field) matches the requested sorting field. If so, it proceeds to toggle the sorting order.
      + Descending to Ascending: If the current sorting order is descending (sort.asc is false), it sorts the data in ascending order for that field. This involves comparing values within each object in the array, adjusting for numerical or string types accordingly.
      + Reset Sorting: If the sorting field is clicked again after being sorted in ascending order, the function calls handlePageChange() to presumably reset the view to its unsorted state, and setSort with no field and null order.
      + Sorting Logic: If a new field is specified for sorting (or for the first invocation with any field), the function sorts the data based on the field type (numerical or string) in ascending order by default.
      + Pagination: Post sorting, it slices the sorted array to only include a segment of data corresponding to the current page (currentPage) and number of items per page (currenciesPerPage).
    * Return: This function does not explicitly return values. Instead, it updates the state in several ways:
      + setData: Updates the displayed data to a slice of the sorted array according to the current pagination settings.
      + setSort: Updates the sorting criteria (field and asc) to reflect the current state of sorting.

* Backend Main Functions:

1. fetchAndStoreCryptoData:

* Purpose: This function is scheduled to run every 60 minutes. Its primary role is to fetch cryptocurrency data from the CoinGecko API, parse the data, and store it in the Firestore database under the "cryptocurrencies" collection. It also records the timestamp of the update.
* Process:
  + API Request: Constructs a URL with query parameters to fetch data for 100 cryptocurrencies, sorted by market cap in descending order, including their sparkline and price change percentages over 1h, 24h, and 7d.
  + Data Processing: Upon a successful response, it parses the JSON data and uses a Firestore batch operation to update the database. Each cryptocurrency is stored as a document identified by its id, with the data as document fields.
  + Timestamp: Records the current time in Israel's timezone as the update timestamp in a specific document ("updatedTimeStamp") within the "cryptocurrencies" collection.
* Error Handling: Logs an error if the fetch request fails or if there's an issue updating Firestore.

1. fetchAndStoreCryptoNews Function
   * Purpose: Similar to the first, this function runs every 60 minutes to fetch the latest cryptocurrency news from the CryptoCompare API and store it in Firestore under the "cryptonews" collection. It also handles the deletion of existing news before storing the new batch.
   * Process:
     + API Request: Makes a request to the CryptoCompare API to fetch the latest cryptocurrency news in English.
     + Pre-update Cleanup: Before adding new news data, it deletes the existing documents in the "cryptonews" collection to prevent outdated news from persisting.
     + Data Processing: Parses the JSON response and, for each news item, creates or updates a document in the "cryptonews" collection with the news ID as the document key.
     + Timestamp: Adds a timestamp of the update in the "cryptonews" collection, formatted to the current time in Israel's timezone.
   * Error Handling: Logs errors if the fetch request fails or if updating Firestore encounters issues.

We covered the main functions of the frontend and the backend of our code, we have more functions but’s its secondary and deals with user-interface and user-experience.