Final Report

Burger-Crypto | Team068-BurgerMaker

Beside the content of the final report shown as follows, there are several other related files (most are required to satisfy the requirements for the stage 4 checkpoint 2).

Github Repo | SQLs | Discussion

- **Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).**

  No significant changes from the project developed in our initial proposal submitted at stage 1 was made. We implemented only minor, incremental changes that aimed to enhance the overall functionality of the project.

- **Discuss what you think your application achieved or failed to achieve regarding its usefulness.**

  We finally achieved a powerful minimum viable product. The achieved features are as follows.
  1. The user-related management. Log in, authentication, and user info …
  2. The trading. Real-data market, trading simulation, and portfolio-level analysis and visualization …
  3. The community. It is where users learn from each other with published posts. Also related to some incentive mechanisms.
  4. Retrospective of operations. AI-based analysis and recommendation.

  There is still something we failed to achieve.
  1. Low -latency market information update. Since we could only access crypto data from other platforms' API, the precision and update speed is not guaranteed.

- **Discuss if you change the schema or source of the data for your application**

  No, the schema is mostly like the one we design at previous stages.
  We add some additional tables to support advanced features. For example, "holds" table to record users' holding cryptos and "StarPostRecord" table to record thumb-ups for posts.

No, like we designed before, the crypto info is from external API and user info is auto-generated. The AI-driven investment suggestion is generated with LLM (ChatGPT-4 API).

- **Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?**

  Table “hold”: This is a table to record how many cryptos a user holds. It gets us rid of calculating such information over and over again with the “trades” table. It reduces the query overhead of our system.

  Table “StarPostRecord”. This table is to record “star history. Only with it can we display “who likes your post”, implement “unstar” and avoid star repetitively, which is important since we now use star to reward users.

- **Discuss what functionalities you added or removed. Why?**

  Since our Burger Crypto is aimed to act as a learning platform, we provide more incentive mechanisms to encourage the user community. This is based on Burger Coin. Users are rewarded Burger Coins with others “likes” for their posts. And there may be more ways to earn coins and such coins could be used to exchange for some “small prize” or souvenir, like a real burger, in the future.

- **Explain how you think your advanced database programs complement your application.**

  Thanks to the transaction and trigger mechanism, the system achieves consistency on any transaction made by our user, without complex locking designs. Store procedures help us to shorten the length of code we need in the backend, saving much debug time and redundancy.
  For the details of our implementation, we have included all the sql files inside the git directory : **doc/SQL/**, and the text of explanation is formatted as comments.

- **Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future**

team could use this as helpful advice if they were to start a similar project or where to maintain your project.

Yunchao Hou: User logging and authentication design. We adopt token + local storage to store user_id encrypted via MD5 as token in the local storage within the browser. Every time a user is logged in, a new token will be generated and stored in the user's browser. Whenever the user revisits our website, the server will try to verify the token, extract the information about the user, and log the user in without a password. The token is set to expire in 1 hour so that the system is safer.

Yuanze Zheng: When coding the implementation of the transaction logic, due to the Portfolio implementation, there are multiple steps to be performed in a single transaction, including checking the user's cash balance (buy) or the user's virtual currency balance corresponding to the Portfolio (sell), as well as updates to the transaction table, updates to the holdings table, and deletion of columns with a quantity of zero after the sell. We encapsulate this step as a transaction in a stored procedure, ensuring simplicity of the code that triggers the transaction process as well as atomicity of the transaction and consistency of the data across the database.

Ziyang Xie: One technical challenge that I solved is the integration of an external Crypto API in our backend system, which is of great importance to ensure seamless interaction with our internal database for real-time updates of cryptocurrency price and. This was a very critical feature for our product. My contribution was about designing and implementing a robust API integration framework that would fetch and parse dynamic data with a strong focus on handling data inconsistencies and network latency issues.

Bohan Wu: There’s a list-related API. We integrate search, pagination, star … so much functionality into it. Some of the params are even optional. Such design leads to complex SQL string concatenation and many branches. So to avoid bugs, developers should consider all user paths and use complete unit tests.

- **Are there other things that changed comparing the final application with the original proposal?**

  We singled out the Trade table as a relationship between Portfolio and Asset.

We added a Vote table to track the history of post-like events by users, so that we can prevent duplicate likes.

- **Describe future work that you think, other than the interface, that the application can improve on**

1. **Better Token Logic:** Use Token + session to control the expiration datetime of the token to increase the safety.
2. **Advanced Features:** Create a Statistics table to store data aggregated in a certain interval, like networth of a user each day, month, year. It'll be faster and more elegant to fetch data, instead of using SQL aggregation everytime user queries.
3. **Data Management Scalability:** Since the application grows, keeping the volumes of data becomes more critical. Advanced data partitioning and indexing would be necessary with this growth to deal with this growth. This would mean that the system remains responsive to all sorts of data loads without performance degradation.
4. **Performance Optimization with Redis:** Using Redis as a data structure store in memory can speed access to data significantly. The system could act as a cache or as a message broker, improving loading times and enhancing the experience of the user in case of frequent interactions in regard to data, particularly real-time features.
5. **Semantic-Level Data Retrieval Using Vector Databases:** Implementing a vector database could enhance our system's ability to perform semantic-level data retrieval. This would allow for more sophisticated user profiling and the delivery of personalized trading advice based on deep insights into user behavior and preferences. Tools like Milvus or Pinecone could be explored for these purposes.

- **Describe the final division of labor and how well you managed teamwork.**

  Yunchao Hou: Authentication, user login system, first lines of dashboard, OpenAI interface, user info page.

  Yuanze Zheng: User portfolio page and backend CRUD, Pi Chart of user's portfolios' ratio, each Portfolio's risk evaluation(Store Procedure), Trade logic(Transaction).

Ziyang Xie: Crypto Market, Trading, Incorporate real-time crypto price search. Server deployment, Overall Frontend refinement, API design.

Bohan Wu: The community, post publish, post management, list, and star design.

We managed our teamwork perfectly without any conflicts.