FTGP Group [Group Number]: Sprint 4 Report (3rd May)

Sprint 3 Review (Sprint duration 29th April – 3rd May):

This should summarize your sprint review meeting. The meeting should be done at the end of each sprint. You need to identify which tasks from your sprint backlog were completed, which were altered, and which were not completed. Please also use this process to reflect and improve your next sprints.

Completed work, i.e. which tasks were complete and by who:

Freddie Yu：

MongoDB Client:

Location: backend/dbs/mongo_db.py

Features:

-Implements a MongoDB class for managing MongoDB operations.

-Provides various methods for common database actions such as getOne, getAll, and insertOne with error handling.

-Uses __enter__ and __exit__ methods to support context management, ensuring connections are properly closed.

-Allows for switching between tables, performing aggregations, and other advanced operations.

Redis Client:

Location: backend/dbs/redis_db.py

Features:

-Provides a connection to a Redis instance specified by the REDIS_URL from the settings.

-Includes a method for checking the connection (test), and uses __getattr__ to delegate calls to the Redis connection.

Settings Configuration:

Location: backend/settings.py

Features:

-Defines database configurations, including MongoDB and Redis connection details, and specifies constants like database names and paths used throughout the backend.

Yuyao Wang：

Api for users to use：

Location： backend/api/apis/user_api.py

Features:

-User Registration (/register)

Function: Allows new users to register. It receives information such as user name, phone number, and password, and creates a new user record in the database.

Method: POST

-User login (/login)

Function: Allows users to log in. It authenticates the user with a mobile number and password, and may set relevant session information after a successful login.

Method: POST

-User logout (/logout)

Function: Allow users to log out and clear the login status.

Method: Any, but usually POST

-User Information (/info)

Run the following command to obtain information about the current login user, such as the user ID, user name, and whether the user is an administrator.

Method: Usually GET

-User re-authentication (/reauth)

Function: Interface for users to re-authenticate during a session.

Method: POST

-Administrator gets the user list (/user_admin_list)

Function: Get a list of verified or unverified users for administrator review.

Method: POST

-Administrator User authentication (/user_admin_auth)

Function: The administrator can perform authentication operations on users, such as updating the authentication status of users.

Method: POST

-Error handling and user status loading

Features: Contains several functions for Flask-Login integration to handle callbacks for unauthorized access and to load user information.

Jingkun Yang:

Basic blockchain and a simple web application:

Location：backend/api.py

Features:

The code could be divided into three part: Class Blockchain definition and functionality, flask application and route definition, run Flask application. The detailed features are as follows:

- Defines a basic structure and method of a blockchain, include create blockchains, manage transactions, mine blocks and Potential conflicts are resolved through the consensus of network nodes by replacing the current chain with the longest valid chain found in the network.

- Created a network server using Flask.

- Defines routes for Flask applications and for handling HTTP requests. The operating parameters of the Flask application are set, including the host address and port number

Add more comment in previous code:

- Add comment in frontend user mobile part and transform the language to English.

Yunkui Yu：

Api for users to use.

Location：backend/api/apis/task_api.py

Features:

-Add task (/add_task)

Function: Adds a new task to the database. Automatically assigns a UUID and timestamps to the task. Validates the user's authenticity, throwing an exception if the user is not valid. Inserts the task details into the database and returns the newly created task data.

-Get task (/get_my_task)

Function: Retrieves and returns tasks created and invested in by a user based on their user ID. Counts the number of tasks created and invested in by the user

-Get task main (/get_task_main)

Function: Retrieves all tasks that are currently "in progress." Calculates the number of tasks in different statuses, such as ongoing and other statuses.

-Get task list (/get_task_list)

Function: Retrieves a list of tasks based on a specified status (default is "about to crowdsource").

-Get task detail (/get_task_detail)

Function: Fetches detailed information about a task based on its task ID. If the task exists, it returns information about the task and the associated user.

-Get comment (/get_comment)

Function: Retrieves comments associated with a task based on the task ID.

-Add comment (/add_comment)

Function: Adds a comment to a specified task. Logs the creation time of the comment.

-Add complaint (/add_tousu)

Function: Adds a complaint about a task. Logs the creation time and initializes the complaint status.

-Get index of tasks (/get_index_tasks)

Function: Retrieves "in progress" tasks; if not enough tasks are available, it supplements with "about to crowdsource" tasks. Randomly selects tasks for display on the homepage.

- Administration interface of the tasks (/task_admin_main)

Function: Administration interface used to retrieve "about to crowdsource" and "crowdfunding failed" tasks, as well as to count tasks in different statuses.

-Milestone control (/task_admin_lcb)

Function: Retrieves a detailed list of all on-chain business milestone for ongoing tasks, including counts of started and completed.

Aibo Xu：

Blockchain implementation:

Location：backend/blockchain

-The build implements a basic blockchain system with full functionality for creating and managing blockchains. It includes mining operations to define the structure of blocks, initialize the blockchain and create genesis blocks if necessary, and add new blocks to the chain.

-In addition, functions for verifying the difficulty of block hashes, replacing chains to maintain the longest valid chain, and persisting blockchain data are also implemented. The whole system aims to provide a secure and tamper-proof data recording mechanism.

MongoDB-Based Data Management:

Location：backend/api/datas/data_api.py

-Multi-table data management: by defining multiple MongoDB collections (e.g., users, blocks, tasks, etc.), management for different data types is realized. This includes data addition, querying, updating and deletion operations, making data processing more specialized and modular.

-Complex Query and Update Operations: Supports advanced data retrieval functions, including sorting, filtering and complex condition queries, as well as update operations for specific fields. This enables the application to flexibly handle a variety of business requirements, such as user management, task tracking and complaint handling.

-Implementation of auxiliary functions: In addition to basic database operations, the code also implements the management of task-related investment, comment and complaint information to enhance the interactivity and user experience of the application.

Changes, i.e. tasks that have changed/not completed and why:

Delete the forget password part in ueser mobile login part since we need to use the characteristic of blockchain to get user's account back.

Agreed weekly "Equity share", i.e. how this sprint's work was split:

Equally shared.

Sprint 4 Planning (Sprint duration 6th – 22nd May)

This should summarize your sprint planning meeting. The meeting should be done at the beginning of each sprint. You must specify your sprint vision and select which items from the product backlog you plan on completing during the next sprint (sprint backlog). Additionally, you must select the product owner and the scrum master.

Product Owner: Jingkun Yang

Scrum Master: Freddie Yu

Sprint Vision:

Next week, we will delete the invalid part of the front-end according to the feedback from the meeting on Monday, and standardize the front-end code. It also discusses how to connect the front end to the back end. According to the current discussion. Parts of the previous login may need to be removed.

Sprint Backlog:

Freddie Yu：

Further Improve the Database:

Enhance the database clients with additional functionalities and optimize existing code for efficiency and robustness.

Adjust and Optimize Frontend Layout Related to the Database:

Modify and refine frontend interfaces interacting with the databases, ensuring smooth data exchange between the frontend and backend.

Insert Data into the Database to Enable System Operation:

Populate the MongoDB and Redis databases with relevant initial data, ensuring the system functions correctly from data entry to retrieval and frontend integration.

Explore Mature Similar Products or Platforms:

Identify key differences between our system and others in the market.

Uncover unique features that contribute to the system's effectiveness.

Identify market pain points our system can address.

Jingkun Yang:

I will explore the contents of the login section to find more possible combination between web 2.0 and web 3.0 and remove the code that is useless. In addition, I will assist Freddie to modify the contents of index.js file in frontend/mobile api file according to back-end needs.I will also reconsider the usability of some front-end code and make appropriate cuts.

Yuyao Wang：

Our work progress may slow down next week as we have other coursework to complete, so next week I will focus on modifying the front end and trying to improve the user api part of the current back end.

Yunkui Yu：

Modify the front-end code to make the front-end functionality clearer and more complete. And according to the completed api file, discuss the corresponding back-end function implementation with team members.

Aibo Xu：

Improve the front-end investment management board, add dynamic data display, realize real-time update display content. It is expected to add the task list switching function, so that users can view different types of tasks according to their choices.

Plan to realize the project information display part. Access back-end data to realize dynamic content display. Dynamically obtain all project information from the backend, including name, amount, status and description, and display it through data binding to ensure real-time and accurate content. Enhance system interactivity: Dynamically display investment buttons according to the status of the project, provide conditional rendering to adapt to different project status, and increase feedback and guidance for user operation.

Add detailed user information display section, new user's detailed information display and complaint function, so that users can have a more comprehensive understanding of the project creator, and at the same time provide a more friendly way to submit and handle complaints.

Anything else you would like to share:

At present, we can only plan until next week, because we need to arrange and discuss the follow-up work while completing the content of next week. However, we will focus on assigning tasks according to each person's technical characteristics. For example, Freddie is good at front-end, so it is possible to let him do front-end optimization; Yunkui Yu to do the overall DAPP test; Aibo Xu does the modification of the back-end blockchain related parts; Yuyao Wang makes changes to other parts of the back end; Jingkun Yang will write the report. Note that this is just focusing on each person's assignment. In practice, everyone will assign some other content to ensure that everyone is involved in the whole process.