Design Document for Cynance

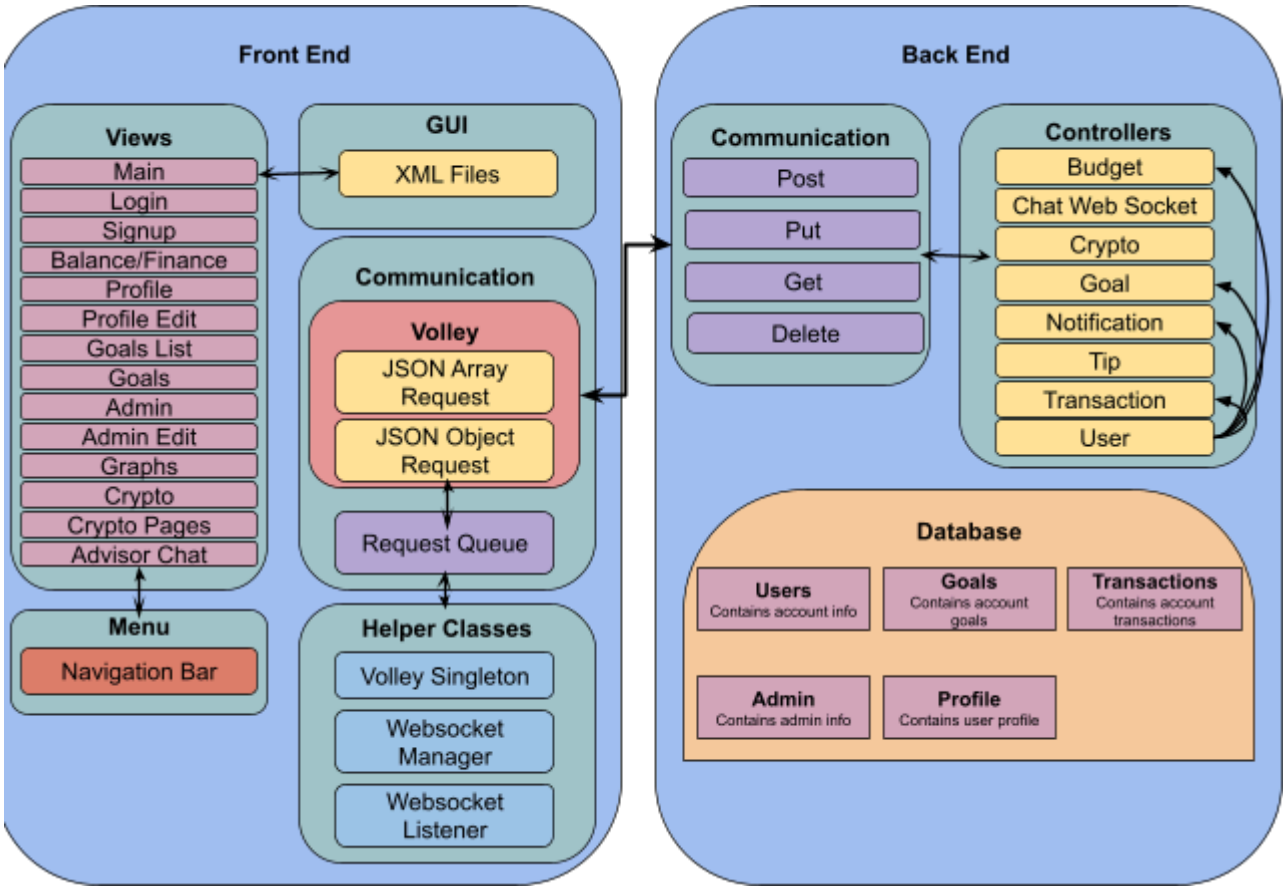
Group 1_Jabir_1

Member 1 Zachary Ahmad: 25% contribution

Member 2 Alec Ferchen: 25% contribution

Member 3 Fadi Masannat: 25% contribution

Member 4 Jacob Mashol: 25% contribution


Front End
Views
Main
Login
Signup
Balance/Finance
Profile
Profile Edit
Goals List
Goals
Admin
Admin Edit
Graphs
Crypto
Crypto Pages
Advisor Chat
Menu
Navigation Bar
GUI
XML Files
Communication
Volley
JSON Array Request
JSON Object Request
Request Queue
Helper Classes
Volley Singleton
Websocket Manager
Websocket Listener
Back End
Communication
Post
Put
Get
Delete
Controllers
Budget
Chat Web Socket
Crypto
Goal
Notification
Tip
Transaction
User
Database
Users
Contains account info
Goals
Contains account goals
Transactions
Contains account transactions
Admin
Contains admin info
Profile
Contains user profile

Use this third page to describe complex parts of your design.

Frontend(Currently Implemented)
Signup(User)

- Signup stores & sends these values:
    - EditText: Username
    - EditText: Password
    - RadioButton: isAdmin
- Upon Clicking the signup button, the values Username, Password, First, Last, isAdmin are sent to the backend server through a POST request

GoalsPage(User)

- Goals page sends data to the backend for each goal
    - EditText: Goal
    - EditText: Date
    - EditText: Description
    - EditText: OldGoal
    - Button: post
    - Button: put
    - Button: delete
- Upon clicking the delete button, the value from OldGoal finds an ID from the server from the old goal name and sends a delete request with this ID. This happens with the put button as well but instead sends a put request then with data from Goal, Date, and Description. The post button sends Goal, Date, and Description to the backend to create a goal.

CreateCrypto & CreateTip(Admin)

- Both crypto creation & tip creation sends these values:
    - Crypto Creation
        - EditText: Price
        - EditText: Name
        - Button: POST
            - Upon Clicking the “Create” button the values of Price and Name are sent to the backend via POST to create a new cryptocurrency that will be displayed in the backend server
    - Tip Creation
        - EditText: Quote/Tip
        - Button: POST
            - Upon Clicking the “Create” button the string value of the quote/tip is sent to the backend via POST to create a new tip that will randomly selected to be displayed

Financial Advisor(User)

- Advisor help sends these values to the backend websocket:
    - TextView: Name
    - Button: Connect
        - TextView: Message
        - Button: Send
- Upon clicking the connect button the name will be concatenated to the end of the websocket link to create a connection. Then you can type a message via the message textview and send that to the backend websocket after clicking the send button

Backend Design Overview

The backend server for this application, built with Spring Boot, Java, and JPA, follows a modular architecture with layered components for handling various functionalities. The backend serves as a RESTful API provider and real-time communication hub with the frontend, focusing on modularity, scalability, and secure data handling.

Modules and Components

1. User Management
    - Entity: The User entity represents users in the system with attributes such as username, password, balance, role, firstName, and lastName. It maintains:
        - One-to-Many Relationship with Goal and Transaction entities, where each user can have multiple goals and transactions associated with them.
        - Many-to-Many Relationship with Crypto, allowing users to manage a portfolio of different cryptocurrencies.
2. Transaction Management
    - Entity: The Transaction entity captures each financial action, including attributes such as amount, type (e.g., deposit, withdrawal, transfer), and timestamp. Relationships include:
        - One-to-Many Relationship with User (sender and recipient), enabling intra-user transactions.
3. Goals Management
    - Entity: The Goal entity tracks individual goals for each user with attributes like title, description, startDate, and deadline. Each goal is associated with:
        - One-to-Many Relationship with User, as a user can have multiple goals, while each goal belongs to only one user.
4. WebSocket Notification System
    - Server: The NotificationServer WebSocket endpoint (/notifications/{username}) is responsible for managing live notifications. It connects each user to a unique WebSocket URL based on their username.
    - Session Management: Maintains active WebSocket sessions using SessionManager, which maps usernames to sessions and manages session lifecycle events (open, close, and error).

Relationships and Data Transfer Objects (DTOs)

The backend relies on a structured relational mapping using JPA annotations to manage entity relationships effectively. For example:

- One-to-Many Relationships: Used for linking users to goals and transactions, allowing each user to manage their goals and track their transaction history.
- Many-to-Many Relationships: Utilized in the cryptocurrency module, where users can hold portfolios of multiple cryptocurrencies.

DTOs like UserSummaryDTO, TransactionDTO, GoalDTO, play a crucial role in encapsulating data. They ensure that only the necessary information is sent to the frontend, enhancing security (e.g., masking sensitive fields) and reducing payload size, which optimizes client-server communication.

- Scalability: Modular design with separated service and repository layers ensures that new features can be added without impacting existing functionality.
- Real-Time Updates: The WebSocket system provides immediate feedback to users, enhancing the interactive experience.
- Data Integrity: Transactions are atomic, ensuring data consistency and accurate balance updates, crucial for a finance-oriented application.

Database Schema