

# INFORMATICS 117, The Crypto Bros

## Deliverable #4





Informatics 117: The Crypto Bros  
Santiago Palomares, Lance Li, Xiaohua Zhang,  
Anthony Wen & Husain Wafaie  
31 May 2024

## Table of Contents

Table of Contents	2
Personas/Scenarios	3
Functional Requirements	4
Non-Functional Requirements	9
Constraints and Assumptions	11
UI Design	13
Software Design	20

# Personas/Scenarios

<https://www.figma.com/board/VQSBZs2T0rf5tJW7RI78qT/Untitled?t=pCLg4mDpZE9ePTsQ-1>

<p><b>Persona</b></p>  <p><b>Tony</b> Student Age 20</p> <p><b>Bio</b></p> <p>Tony is a tech-savvy young professional with a keen interest in cryptocurrency investments. He has been actively trading crypto for the past year and is always on the lookout for new platforms to enhance his trading experience.</p>	<p><b>Interests</b></p> <ul style="list-style-type: none"> <li>He thoroughly enjoys to find new crypto investments</li> <li>He enjoys playing video games</li> <li>Likes to read the news on new developments in technology</li> </ul> <p><b>Needs and expectations</b></p> <ul style="list-style-type: none"> <li>Needs a way to communicate with people about cryptocurrency</li> <li>Have a way to see crypto stocks with their values updated real time</li> <li>He needs a secure way to commit crypto transactions such as withdrawing, selling, trading, and buying</li> </ul> <p><b>Influences</b></p> <ul style="list-style-type: none"> <li>The young tech-savvy student is influenced by current days news on social media such as X</li> <li>Tony is in a campus club regarding cryptocurrency that hold discussions about the possibilities and uses of crypto</li> </ul> <p><b>Motivations</b></p> <ul style="list-style-type: none"> <li>Tony is motivated by the potential for high returns in the crypto market. He enjoys the thrill of trading and the opportunity to learn about emerging technologies</li> </ul> <p><b>Goals</b></p> <ul style="list-style-type: none"> <li>Tony aims to increase his crypto portfolio and diversify his investments. He wants to stay updated on the latest trends in the crypto market and make informed trading decisions</li> </ul> <p><b>Pain points and frustrations</b></p> <ul style="list-style-type: none"> <li>Tony faces challenges in finding user-friendly platforms with low fees and reliable customer support. He also struggles with keeping up with fast-paced and volatile nature of the crypto market.</li> </ul>
<p><b>Persona</b></p>  <p><b>Henry</b> Businessman Age 35</p> <p><b>Bio</b></p> <p>Henry is a moderately tech-savvy businessman based in Shanghai, China. At 38 years old, he has built a successful career in the import-export industry, specializing in electronics and consumer goods. Henry holds a Bachelor's degree in Business Administration from Shanghai Jiao Tong University and has over 15 years of experience in his field.</p>	<p><b>Interests</b></p> <ul style="list-style-type: none"> <li>Global Trade enthusiasts</li> <li>Touring the world</li> <li>Networking</li> <li>Investing</li> </ul> <p><b>Needs and expectations</b></p> <ul style="list-style-type: none"> <li>Needs a website to securely trade cryptocurrency</li> <li>Needs to be able to see crypto assets and price to buy/sell</li> <li>Expects to be able to talk with other users about crypto</li> </ul> <p><b>Influences</b></p> <ul style="list-style-type: none"> <li>WeChat</li> <li>Colleagues in his industry</li> <li>Friends and family</li> </ul> <p><b>Motivations</b></p> <ul style="list-style-type: none"> <li>Earn a profit on cryptocurrency</li> <li>Spread knowledge about cryptocurrency to other users</li> </ul> <p><b>Goals</b></p> <ul style="list-style-type: none"> <li>Talk with people about cryptocurrency in a secure and anonymous manner</li> <li>Increase crypto portfolio</li> <li>Trade directly with other NewU patrons</li> <li>Join Group chats in order to find out the latest exclusive insights from peers.</li> </ul> <p><b>Pain points and frustrations</b></p> <ul style="list-style-type: none"> <li>Unable to talk about cryptocurrency in a secure and safe environment</li> <li>No central location for people interested in talking about cryptocurrency</li> </ul>
<p><b>Persona</b></p>  <p><b>Jessica</b> Janitor Age 48</p> <p><b>Bio</b></p> <p>Jessica is a hardworking janitor based in Guangzhou, China. At 48 years old, he has dedicated the past 20 years to working in various facilities, including office buildings, schools, and hospitals. Jessica completed his secondary education and began working immediately to support his family. He has gained lots of interest in crypto trading through word of mouth and from news of cryptocurrency.</p>	<p><b>Interests</b></p> <ul style="list-style-type: none"> <li>Cryptocurrencies</li> <li>Technology</li> <li>Gaming</li> <li>Travel</li> </ul> <p><b>Needs and expectations</b></p> <ul style="list-style-type: none"> <li>Needs a viable, trusted way to stake a coin</li> <li>Have a system that identifies if a coin can be staked or not</li> </ul> <p><b>Influences</b></p> <ul style="list-style-type: none"> <li>WeChat</li> <li>Twitter</li> <li>News</li> </ul> <p><b>Motivations</b></p> <ul style="list-style-type: none"> <li>Secure a better financial future for his family</li> <li>Diversify income sources beyond his day job</li> <li>Learn about crypto for potentially higher returns</li> </ul> <p><b>Goals</b></p> <ul style="list-style-type: none"> <li>Learn more about the benefits of staking cryptocurrency</li> <li>Identify a reliable and user-friendly platform for staking</li> <li>Have another income through staking besides his regular earnings</li> </ul> <p><b>Pain points and frustrations</b></p> <ul style="list-style-type: none"> <li>Limited investment from minimal earnings to invest</li> <li>He has little time to dedicate learning about staking coins and cryptocurrency due to long working hours</li> <li>Understand the workings of staking well enough to ensure the security of his investments</li> </ul>
<p><b>Persona</b></p>  <p><b>Mark</b> Software Developer Age 72</p> <p><b>Bio</b></p> <p>Mark is a dedicated software developer based in Guangzhou, China. With many years of experience learning new software paradigms is easy for him. Recently, he has developed a keen interest in crypto trading, sparked by discussions with colleagues and news about cryptocurrency. He wishes to use NewU in order to stake his crypto and Trade.</p>	<p><b>Interests</b></p> <ul style="list-style-type: none"> <li>Cryptocurrencies</li> <li>Technology</li> <li>Football</li> <li>Music</li> </ul> <p><b>Needs and expectations</b></p> <ul style="list-style-type: none"> <li>Seamless and secure staking and trading</li> <li>Secure connection to wallet and error handling</li> </ul> <p><b>Influences</b></p> <ul style="list-style-type: none"> <li>Fox News</li> <li>Elon Musk</li> <li>X/Social Media</li> <li>Software Trends</li> </ul> <p><b>Motivations</b></p> <ul style="list-style-type: none"> <li>Secure staking</li> <li>Desire to find Community that can be trusted</li> <li>Building a Network</li> </ul> <p><b>Goals</b></p> <ul style="list-style-type: none"> <li>Trade Crypto among work friends</li> <li>Identify a reliable and user-friendly platform for staking</li> <li>Use BTC-GPT to both research the findings of</li> </ul> <p><b>Pain points and frustrations</b></p> <ul style="list-style-type: none"> <li>Complexity of current trading platforms</li> <li>Lack of reliable information / Security with crypto and many emerging scams</li> </ul>

# Functional Requirements using use case format

## Use Case: Communicate with other users

- Basic Flow
  1. User connects to their wallet
  2. User switches to the community page
  3. System presents communication interface showing detailed view (current conversations, search bar)
  4. User searches up the user they would like to communicate with
  5. User types the message they would like to send
  6. User presses the submit button
  7. Message sent to receiver via supporting server
  8. Message stored in Firebase database
- Alternate Flow
  - 1a. User is already connected to their wallet
    1. System does not require user to have to connect to their wallet, returns to BF at step 2
  - 4a. User has a existing conversation with the user they would like to communicate with
    1. User selects the existing conversation from the communication interface, returns to BF at step 5
  - 4b. User wants to add the user they would like to communicate with as a friend
    1. User presses the avatar of the user they would like to communicate with and selects add friend
    2. System automatically creates a private chat between new friend and current user, returns to BF at step 5
  - 4c. User wants to create a group chat with new friends
    1. Users are able to hit the create group button(+ icon) and search for users or select users they would like to add into a group chat.
    2. User is required to send a message to initialize the group chat
    3. The user will be able to name the group chat or skips and stops at step 8
- Exception Flow
  - 5a. User is unable to send message because they are not friends with the user

1. They are prompted to send a friend request instead. The flow may continue after it is accepted or they return to step 3.

### Use Case: Buying assets

- Basic Flow
  1. Users connects to their wallet
  2. User presses the market button leading to a marketplace page
  3. User scrolls down and moves pages to find desired item
  4. User selects the item and bids on the price
  5. User waits until the bidding is over or the seller accepts the biting price
- Alternate Flow
  - 2a. The user finds the desired item on the home page
    1. Returns to BF at step 4
- Exception Flow
  - 4a. Item is sold at the moment of bidding
    1. System notifies User that the bidding is invalid
  - 5a. Seller rejects the bidding price
    1. System notifies User of rejection of bid
    2. System prompts User to give up or increase bid

### Use Case: Selling assets

- Basic Flow
  1. User connects to their wallet
  2. User presses the profile icon leading to their asset's page
  3. System presents detail view of User's list items (cryptocurrency name, amount, value)
  4. User presses the selling button
  5. User chooses the amount and price they want to sell cryptocurrency for
  6. User presses confirm button
- Alternate Flow
  - 6a. User doesn't press the confirm button
    1. User changes the amount/price of cryptocurrency they want to sell, returns to BF at step 6

- Exception Flow

- 5a. User chooses more cryptocurrency than they own

- 1. System notifies User of an error, cryptocurrency is not listed for selling

## Use Case: Staking/Re-staking

- Basic Flow

- 1. Users connect to their wallet.

- 2. User navigates to the staking page.

- 3. System presents available staking options.

- 4. User selects a staking option.

- 5. User specifies the amount to stake.

- 6. User confirms the staking action.

- 7. System processes the staking transaction and updates the user's staking status.

- Alternate Flow

- 1a. User is already connected to their wallet.

- System does not require the user to connect to their wallet again and continues to step 2 of the basic flow.

- Exception Flow

- 5a. User specifies an amount greater than their available balance.

- System notifies the user of insufficient funds and prompts the user to enter a valid amount. 5b. User specifies an invalid staking amount (e.g., below the minimum required amount).

- System notifies the user of the invalid amount and prompts the user to enter a valid amount.

## Use Case: Lending

- Basic Flow

- 1. Users connect to their wallet.

- 2. User navigates to the lending page.

- 3. System presents available lending options.

- 4. User selects a lending option.

- 5. User specifies the amount to lend.

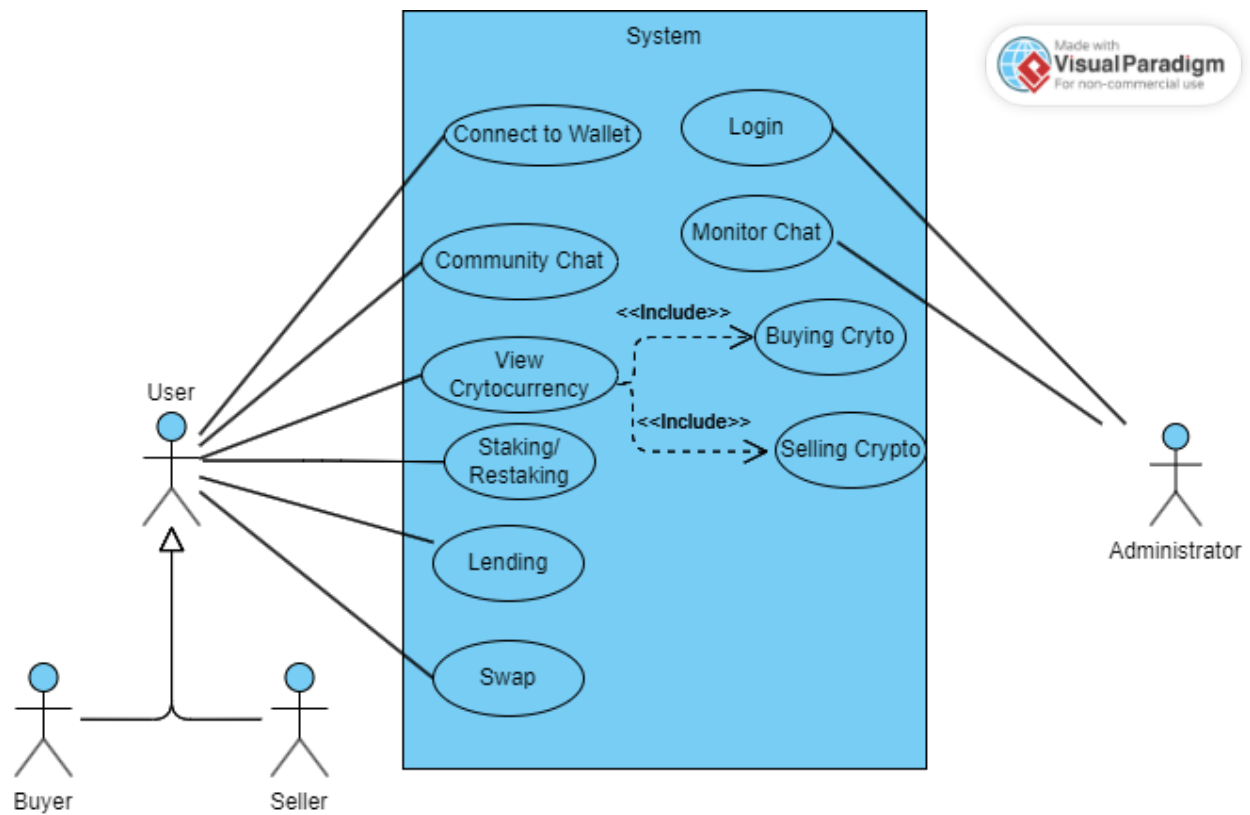
- 6. User confirms the lending action.

7. System processes the lending transaction and updates the user's lending status.
- Alternative Flow
    - 1a. User is already connected to their wallet.
      - System does not require the user to connect to their wallet again and continues to step 2 of the basic flow.
  - Exception Flow
    - 5a. User specifies an amount greater than their available balance.
      - System notifies the user of insufficient funds and prompts the user to enter a valid amount.
    - 5b. User specifies an invalid lending amount (e.g., below the minimum required amount).
      - System notifies the user of the invalid amount and prompts the user to enter a valid amount.

### Use Case: Query BTC-GPT

- Basic Flow
  1. User logs in connecting their wallet
  2. User navigates to the BTC-GPT page
  3. Chat page is displayed
  4. User enters the query into the search bar and sends it to the agent
  5. System processes results and displays them for the user to see
- Alternative Flow
  - 5a. User wants to share results
    - Users are able to share the response with a group chat or direct message.
    - Returned to step 3
  - 5b. User wants to archive or delete chat
    - The user clicks on the chat title on the right chat display and is able to select between a few options (archive or delete chat) then the user is returned to empty chat page at step 3.
- Exception Flow
  - 4a. System deems the user query is irrelevant or inappropriate.
    - System returns a message explaining the exception
    - Returned to step 3

## Use-Case Diagram



## Other Functional Requirements

- The website should support features such as Staking/Re-staking, Lending
- The website should support a search feature for cryptocurrency
- Users should be able to bookmark cryptocurrency
- Users should be able to edit their username



# Non-Functional Requirements

- **Security**

1. *Definition:* Users should be able to connect their wallet to the website in a secure manner. They should also be able to have secure one on one conversations with each other over chat features.
2. *Priority:* High

- **Scalability**

1. *Definition:* Design the system to support future expansions and feature integrations, such as new extensions or services.
2. *Priority:* High

- **Safety**

1. *Definition:* Implement measures to prevent users from accidental transactions, such as confirmation dialogs for trades and transfers.
2. *Priority:* High

- **Maintainability**

1. *Definition:* The system provides comprehensive logging and monitoring tools.
2. *Priority:* Medium

- **Performance**

1. *Definition:* Ensure the system can handle high volumes of transactions quickly, aiming for transaction processing within seconds.
2. *Priority:* Medium

- **Usability**

1. *Definition:* User interface should be intuitive and accessible.
2. *Priority:* Medium

- **Portability**

1. *Definition:* Ensure that the platform is compatible across different operating systems, devices, and browsers. Users should be able to access and interact with the marketplace smoothly.
2. *Priority:* Low

- **Robustness**

1. *Definition:* The system should handle incorrect, unexpected, or corrupt data inputs gracefully, providing useful error messages to the users.
2. *Priority:* Low

- **Sustainability**

1. *Definition:* Optimize resource usage to minimize energy consumption.
2. *Priority:* Low

## Constraints and Assumptions

- **Firestore is Now Required**

1. *Definition:* NewU is now willing to pay for services such as Firestore and intends on using the service long term. It will be used to store most of the data required to run the application.

- **Typescript to Javascript translation**

1. *Definition:* The website is currently built largely using javascript and the inclusion of a tool using typescript has proved to be a dilemma of either translating it or switching over to typescript.

- **Lack of experience in Frontend Design**

1. *Definition:* All 5 of the team members have only a bare minimum of experience with Frontend design, which posed incredible challenges in the current phase.

- **The Figma given is well-designed**

1. *Definition:* The figma design is given by the sponsor. It was developed by another team and is the foundation of our website. One of the major assumptions is that the Figma design is already to the sponsor's liking and at large similar to the final design.

- **Assumption: Avalanche Subnet could be easily integrated into the website**

1. *Definition:* A major portion of the project is the marketplace feature, the assets traded through the Avalanche subnet will have a reduced gas fee, one of the major attractions of our website. However, we have yet to be in contact with the Avalanche team to get a better understanding of the features and how their API works.

- **Assumption: Develop Centralized**

1. *Definition:* There is huge confusion right now on the login and chat feature due to the concept of decentralization being talked about over and over again even

though we're using centralized databases and login options. Our assumption will be that everything will be centralized first.

- **Assumption: Users from China will be able to access our app even though we're using Firebase**

1. *Definition:* China has banned the access of Firebase through their servers. Our assumption is that users will still be using our app through the use of a VPN.

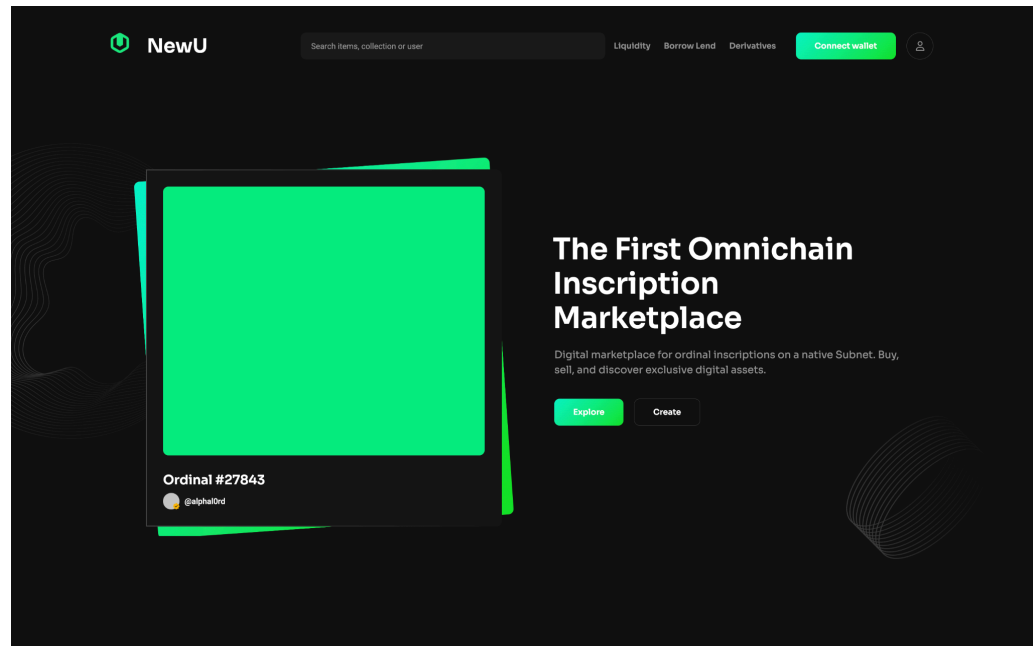
- **Assumption: Users will primarily be using Desktop Chrome**

1. *Definition:* Chrome although illegal in China has many more extensions that can be utilized for features such as logging in. Limiting the scope of the project to a desktop application with Chrome will allow for development to be much quicker.

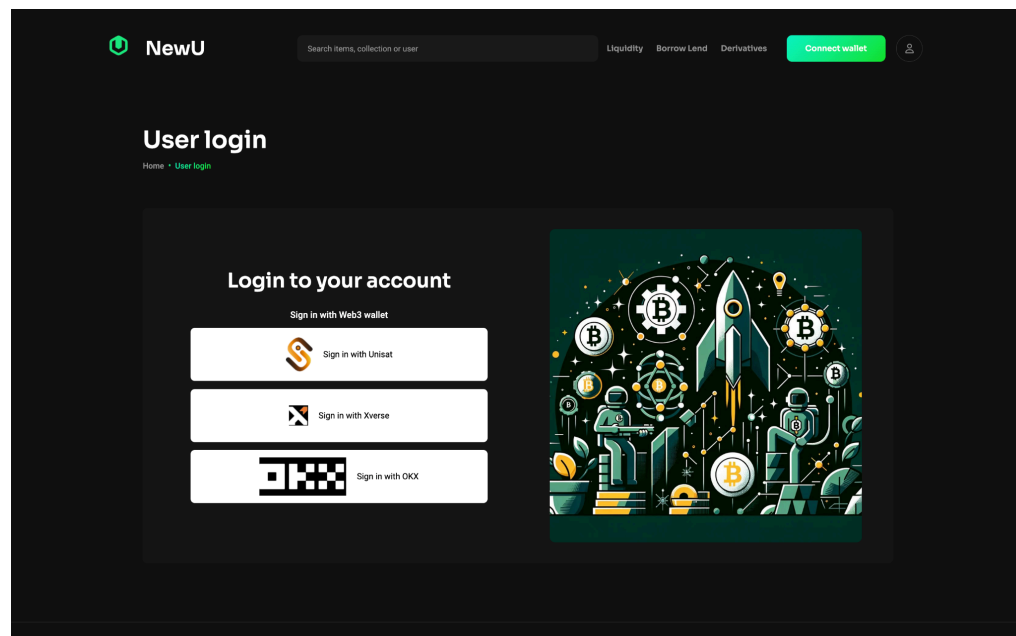
# UI Design

- **UI Mockups/Wireframes/SiteMaps:**

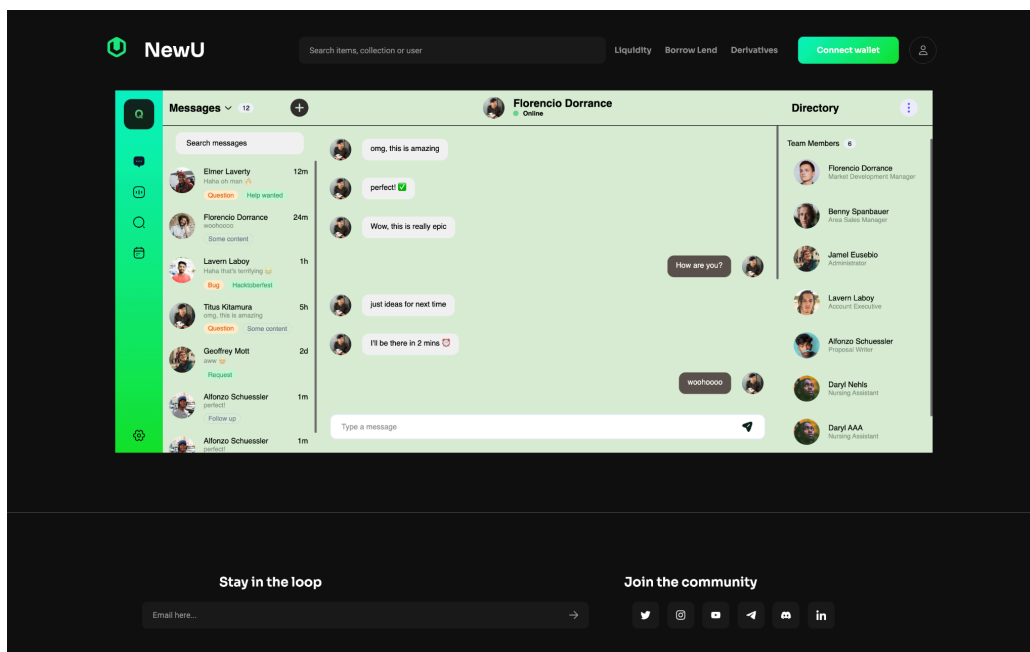
- Home Page



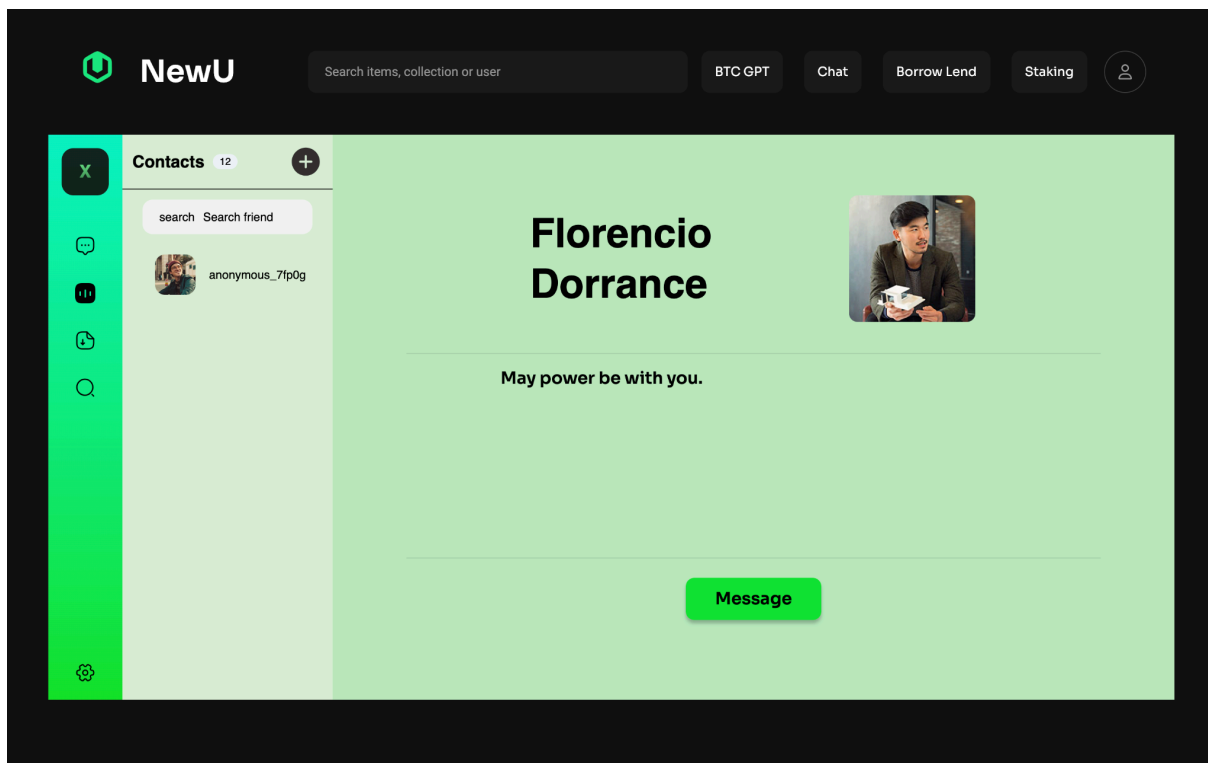
- Login Page



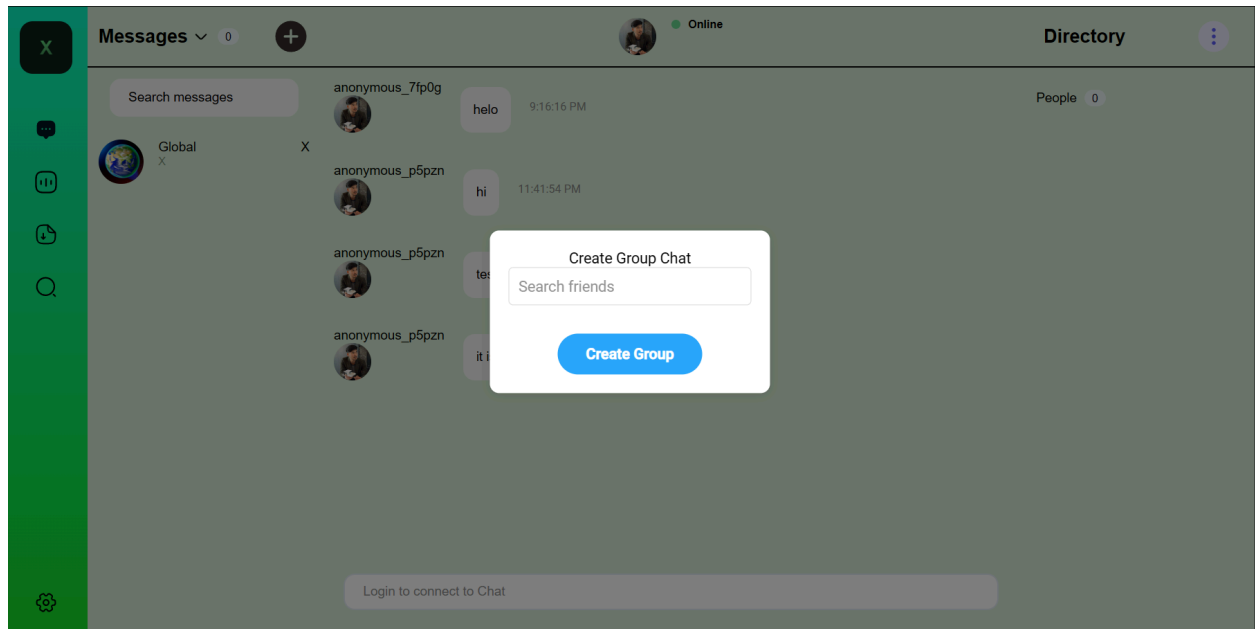
- Chat Tab in Community Page



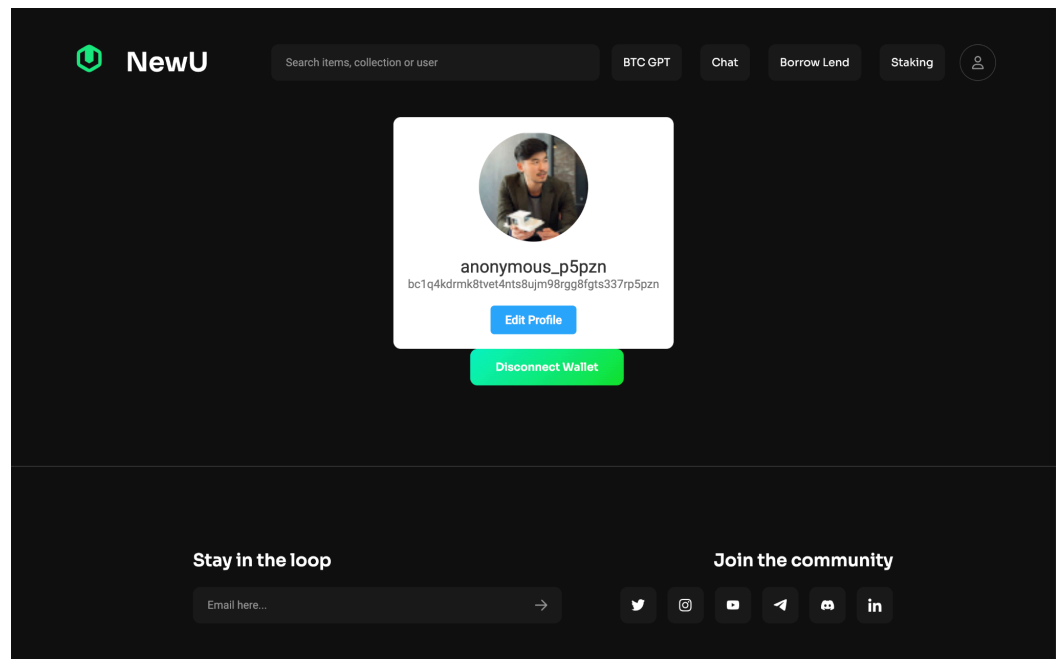
- Friend Tab in Community Page

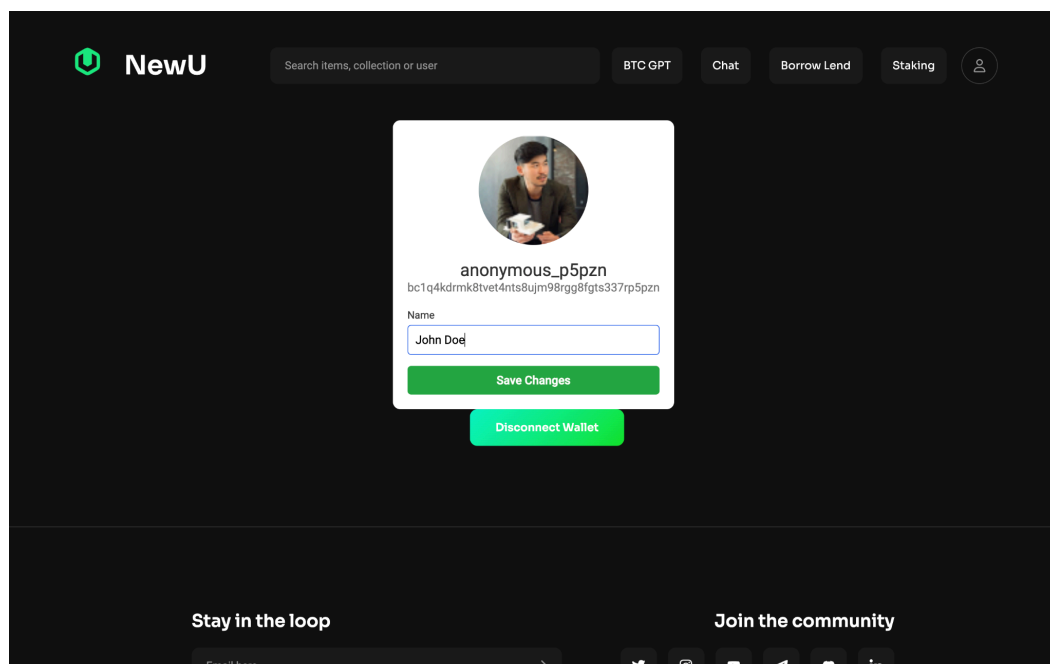


- Create group chat toast

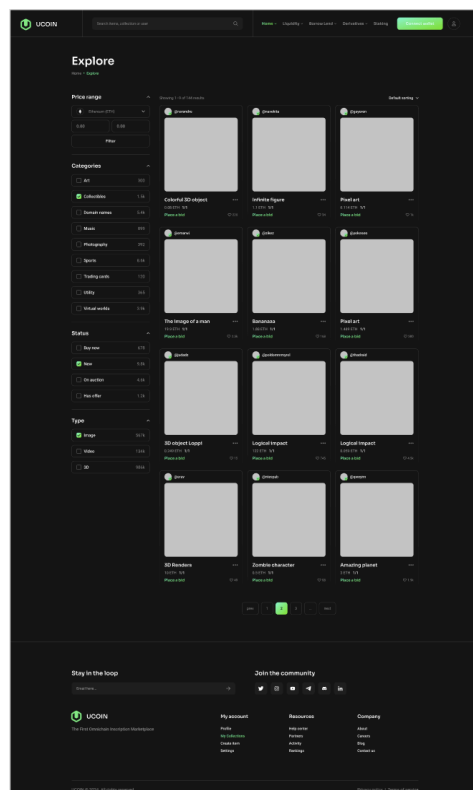


- User Profile Page/Making edit to username



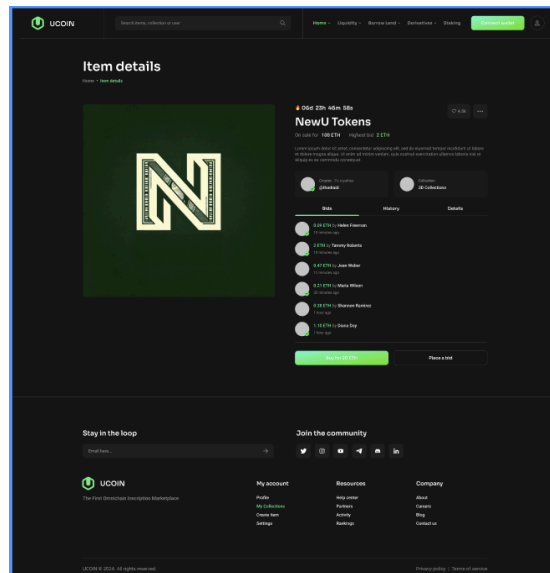


## ○ Marketplace Page

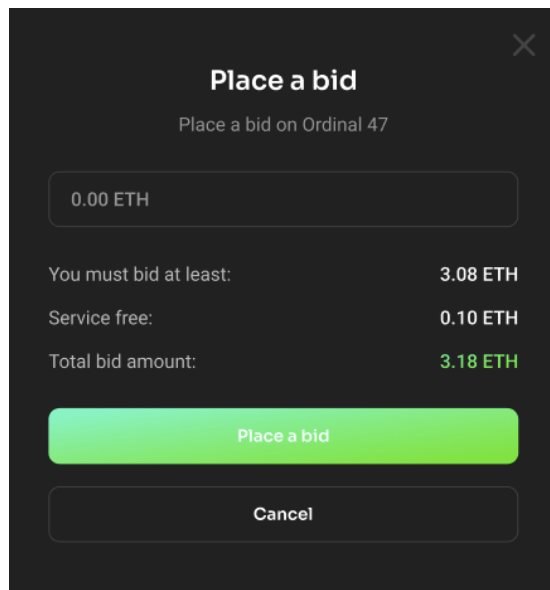




○ Detailed Cryptocurrency Token Page



○ Bid Page



## ● Storyboards:

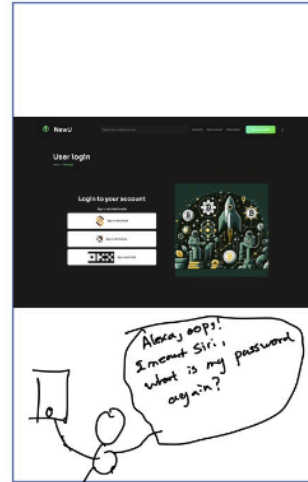
### Chater



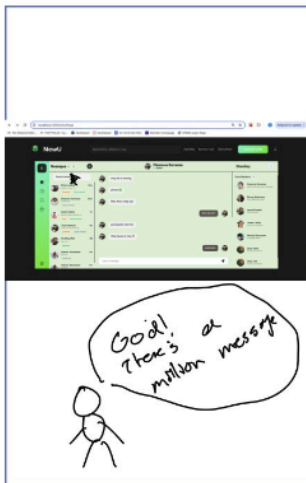
User wakes up in the morning wondering what people in different times have chatted about crypto.



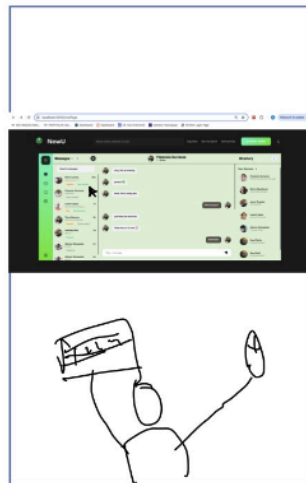
User goes on btcx.club and click on chat to enter the chat screen.



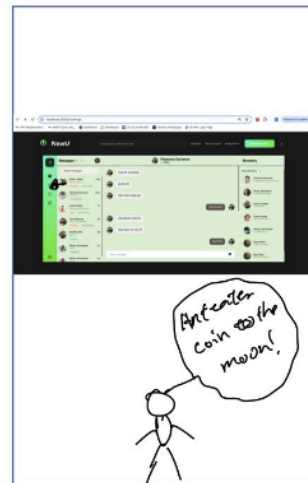
User forgot to log in so the site prompts the login page. User logs in with wallet.



User first sees there are unread messages in his inbox. He checks them.



User then sends greetings to his friends and ask if they trade today.



User goes to global chat to talk about the next meme coin.

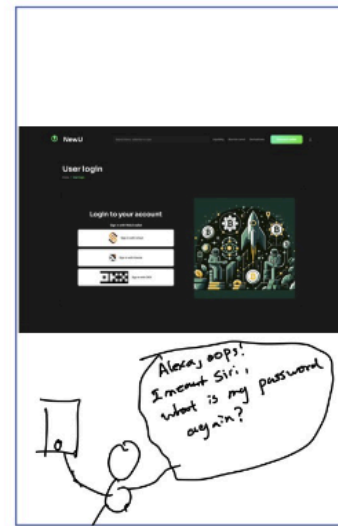
## Placing a bid on a crypto token



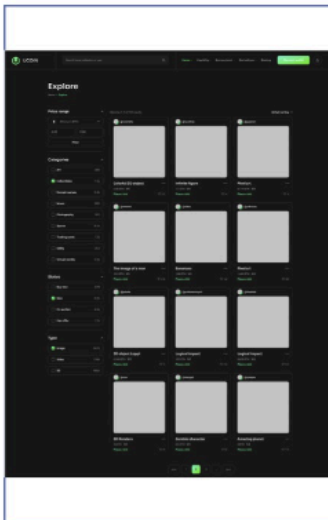
User hears crypto is doing well and decides to purchase some cryptocurrency.



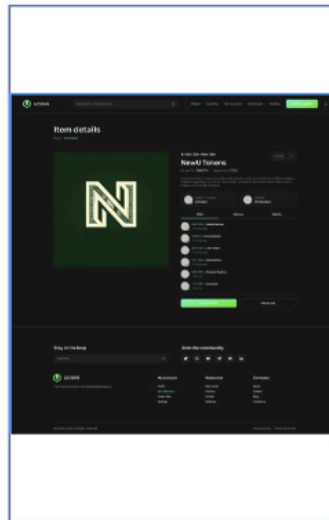
User goes on btcx.club



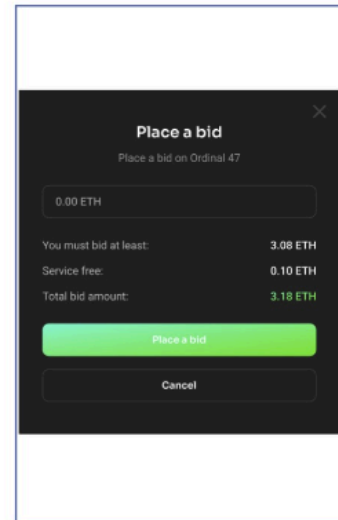
User logs in by connecting to their wallet.



User goes to the marketplace and looks at the available crypto to buy.



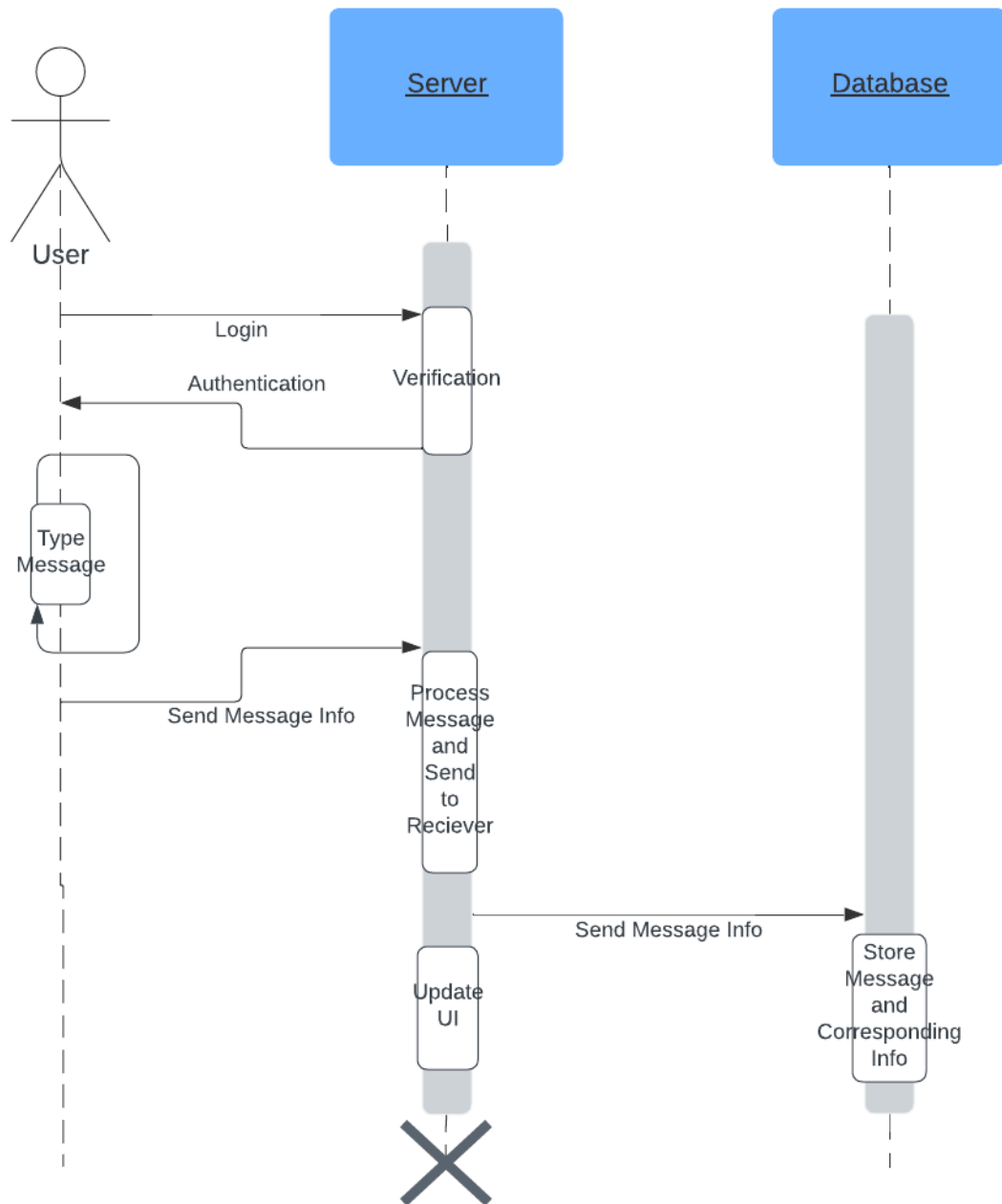
User clicks on a crypto token to see more information about that token.



User decides they would like that token and places a bid.

# Software Design

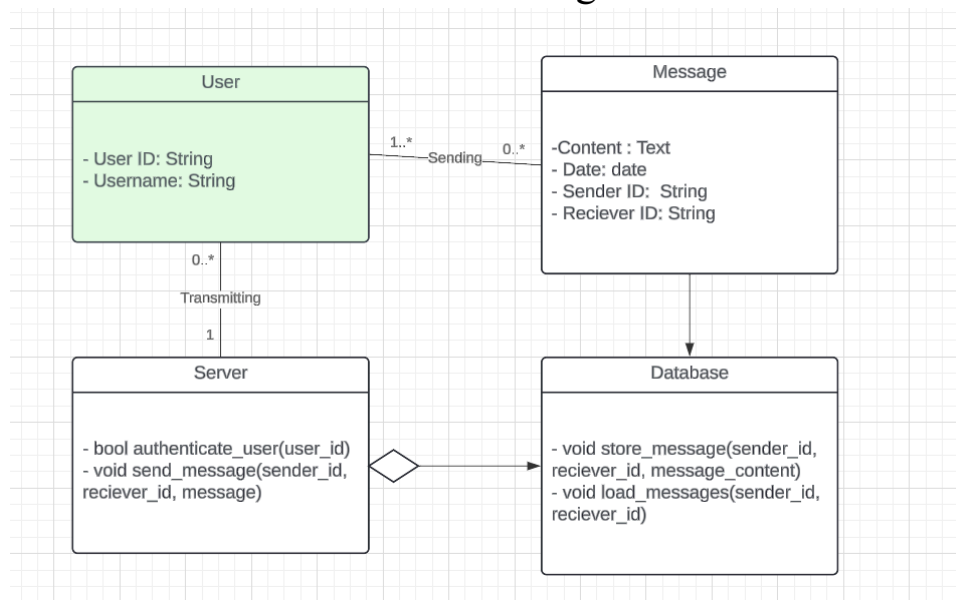
## UML Sequence Diagram:



This UML sequence diagram describes the important use case of communicating with other users. The main actor of this diagram is the user who must first login by connecting to their web3.0 wallet. Otherwise, the user is not able to communicate with other users and is only able to read messages from the global chat. The login is handled with an external extension which currently supports Unisat and OKX. A user's login information which consists of their wallet address and a temporary username is stored inside Firebase.

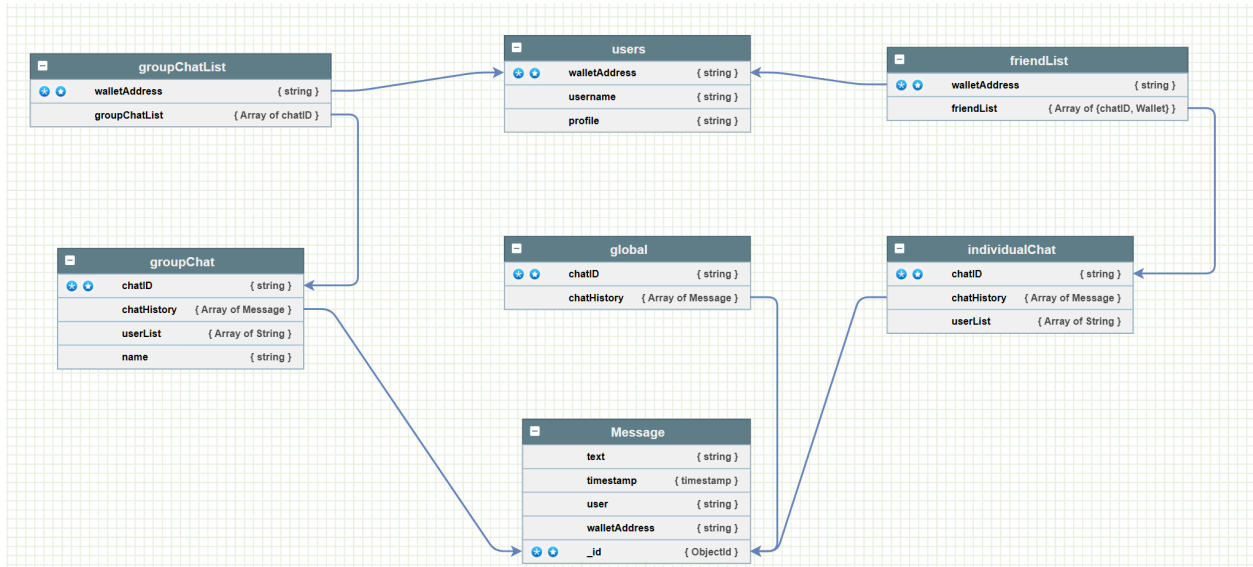
After the user is logged in, they are able to proceed to the community feature by selecting on the Chat button in the header. The user is automatically able to view all the global chats and can now type a message to send to the global chat. Messages are sent to and stored inside Firebase. In addition, the user can now select another user's avatar to add them as a friend or go to the friend tab inside the community page to add friends. Once another user has been added as a friend, an individual chat will open up to allow direct and private communication with the other user. The user's friend(s) are stored inside Firebase.

### UML Class Diagram:

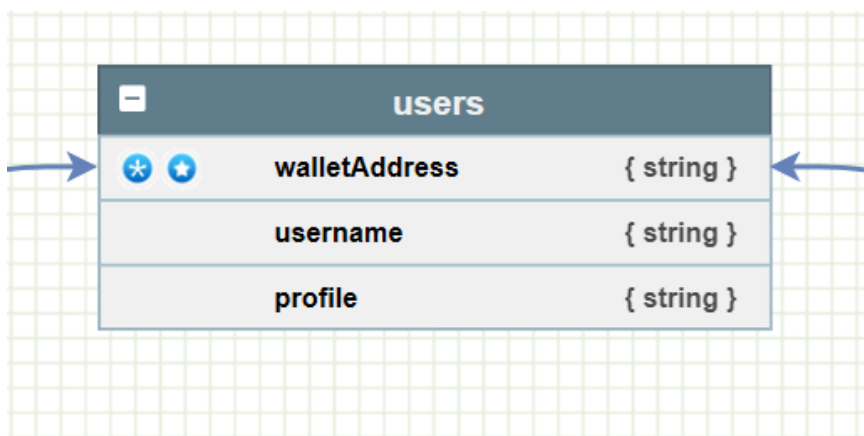


## Database Design NoSQL

(Additional diagram relevant specifically to our project):

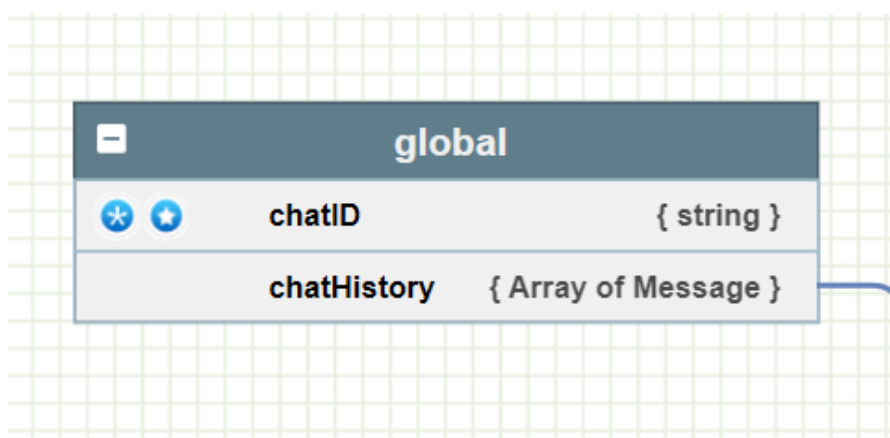


The database is a NoSQL design with Google’s Firebase as our database. Firebase operates as a collection of documents and each document contains information similar to a SQL table to its rows containing information in columns. “By default, Cloud Firestore automatically maintains single-field indexes for each field in a document and each subfield in a map.” (Google Cloud) This characteristic helps for faster retrieval of data, which is one of the main reasons we chose this service as our database.



The **users**' table is where we store our user information. In Firebase, new entries are generated as documents and are automatically given a self-generated unique document for identification. Normally this value has no meaningful purpose other than for identification but we've decided to make our own documentID. The characteristics of web3.0 wallet addresses are they are unique. This is an important characteristic because no one wants their money to go into another's wallet accidentally, both in virtual and in real life. This unique characteristic satisfies the same uniqueness characteristics of the documentID. With a known documentID we can reach each user's information faster instead of querying it. This is extremely convenient because when users log into their web3 wallet, the first set of information we receive contains the walletAddress. The users' unique document is created upon their first login with their wallet. Each wallet will have its own account. While developing this table one of the most important questions was how do we allow users to change username and profile picture. There were two approaches to this problem, either we save it in one location and have API calls from other functions that require profile picture and username, or we save it at all other tables like message and update it periodically. Originally we had the second approach since we were not going to include the update user information functionality until the end. As we approach the end of the development of the chat

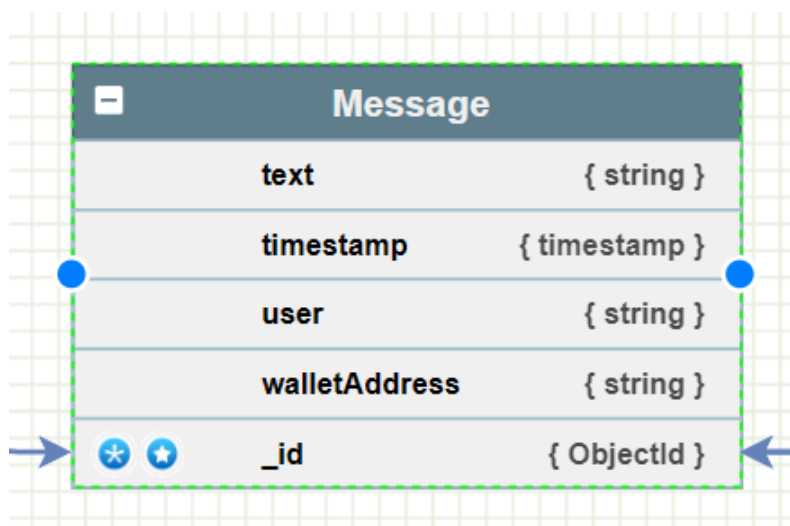
feature, we're starting to tackle this problem. In discussion with our CTO, he suggested using the first approach to ensure the timeliness of the information. We're currently at a 50% milestone on implementing this change. Our biggest setback is that the amount of API calls is too frequent. We want to implement a Redis cache system to save time and money from redundant API calls to the Firebase server. Currently, we've shifted towards finishing all the functionalities before coming back and optimizing with Redis.



The **global** table is where we store all the message information for global chats. Currently, we are only using one global chat but we have a vision that as we include more types of assets/cryptocurrency on our subnet, every coin type will have their community global chat. The global chats are documents which contain only one field, chatHistory, and documentID as the name. Since there is a set amount of global chats and users can not freely create them, it is logical to use the name as the documentID. We are only storing chatHistory because, by the definition of global chat, everyone can type in it. There isn't a meaningful purpose in storing a list of users or other information as it will also slow down the database that is already in high demand. This is not currently a big issue right now but as the number of users increase, it will slow down the loading time of the chat page if every user needs to be displayed for each global chat. The chatHistory is an array

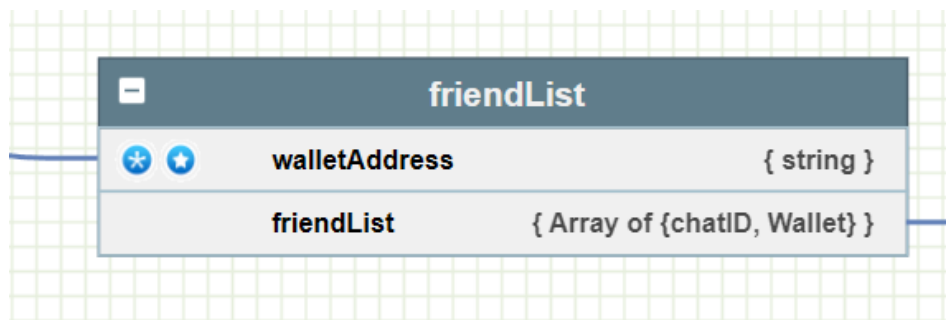


of messages with the format indicated by the message table in the diagram. For future development, there are plans to store only the key identification walletAddress in the message object but it is in low priority since the mass amount of messages in the global chat would already dilute the significance of who wrote the message. In addition, it is also not realistic to implement until a cache system is set up where usernames and profile pictures are easily and cheaply acquired.

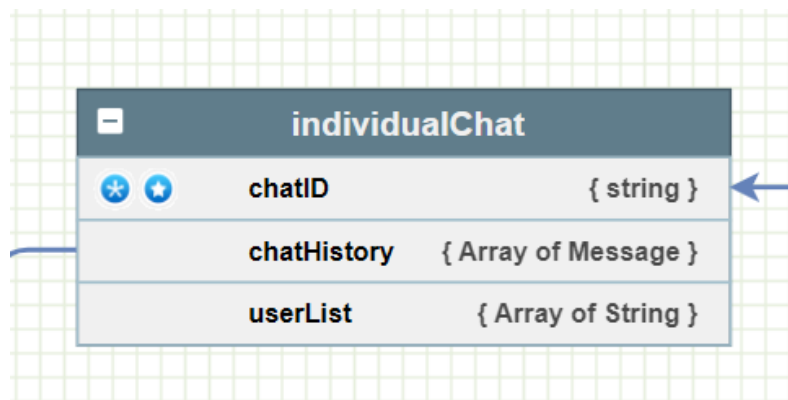


Moving on to the **messages** table which is a placeholder for a message object, we can use it to further develop and brainstorm ideas for creating different types of chats. By default, the database has a single global chat to which everyone has access to. Each individual message object has a few key attributes that simplify development, with one of them being the user. The user attribute of the message class allows for the identification of the sender. This facilitates UI features such as displaying who sent the message, querying the user instance for things like the avatar and username to display, and, later on, the creation of a friends list for each user. Similar to the reasoning for keeping track of a username, we also store the wallet address associated with each message sent. This ensures that when creating relationships between users, there will always be a unique identifier to differentiate and verify security. Otherwise, if multiple people have the same username, we can

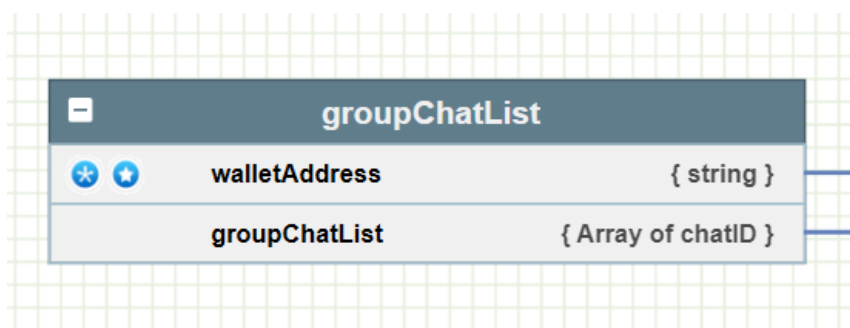
not accurately determine which user they want to add friends with. Other attributes of the message class include text, which is the actual content of the message sent, timestamp, which is the time the message was sent, and id, allowing for the identification of malicious messages later on in development.



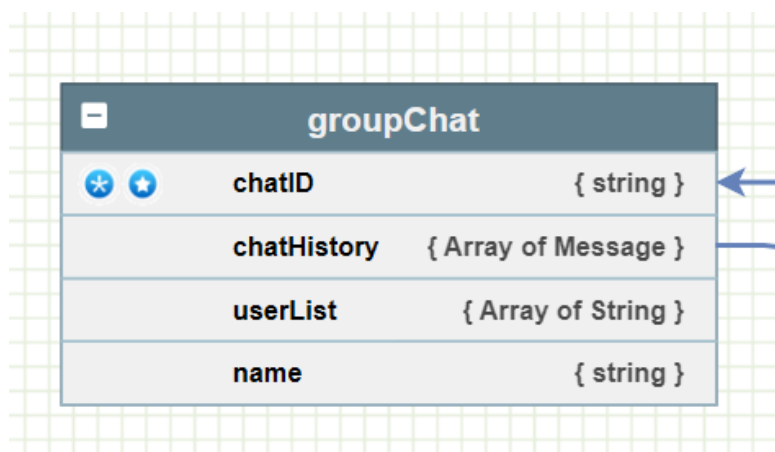
For the **friendsList** table, there are two attributes each allowing for key functionalities related to different types of messaging and later on direct trading. The unique identifier is the **walletAddress** which relates a **friendsList** collection back to its respective user instance. The second attribute is the friend's **walletAddress**, which allows the identification of each friend within the list. This structure enables the system to efficiently manage and retrieve a user's friends, facilitating the creation of private chats and enabling direct transactions between users. Each entry in the **friendList** table represents a unique relationship between the user and their friends, which provides a foundation for personalized communication and trading features. The user's **friendsList** document is generated at their first login and the **friendList** field updates upon adding more users as friends.



Once the friendList table has been created, the user can now create their own chat room. This is related to the **individualChat** table. The chatID is generated through the combination of the walletAddress of users involved in a direct message. The messages here are stored in an array of message objects referenced through their unique associated id. Both of these design decisions allow for the messages to be managed more efficiently for both users. After this class, it will facilitate the development of features such as deleting or editing a message. Despite the name individualChat, an instance of this class would also store a userList which is an array of users. This allows the individualChat class to be reused for instances that require a group chat. However, where they differ is the creation of an instance of a groupChat instance. Individual documents are created upon friend adds, the chatID will also serve as the documentID since two unique values concatenated have a low chance of becoming non-unique. This means that a unique individual chat conversation is created whenever a new user is added as a friend. For future development, developers should look into valid walletAddress characters and find a character that walletAddress will not have and is accepted by Firebase as documentID and insert it in between the two walletAddress to ensure uniqueness.



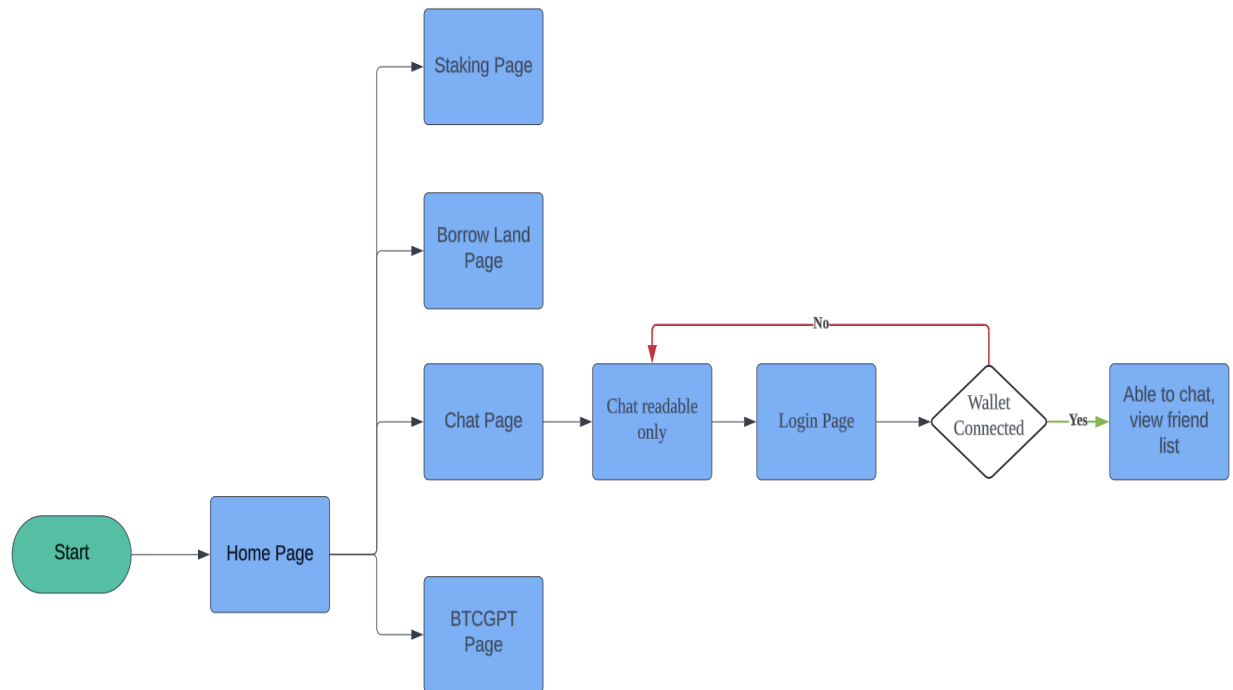
The **groupChatList** table is designed to manage and organize group chats within the website's Chat Page. Each entry in the table is linked to a specific user through their unique **walletAddress**, which serves as the primary identifier. This structure allows the system to efficiently track and manage the group chats for each individual user. Within the **groupChatList** table, the **walletAddress** attribute ties the back to the respective user instance in the **users** table. The **groupChatList** attribute is an array that holds multiple **chatID** values, each representing a unique group chat that the user is a member of. These **chatID** values are linked to the **groupChat** table, where each **chatID** corresponds to a specific group chat entry.



Lastly, the **groupChat** table is utilized to differentiate between a group chat and an individual chat. It includes attributes such as **chatID**, **chatHistory**, **userList**, and **name**. The **chatID** is a unique identifier for each group chat, generated through a library called **UUID** and a **chatHistory** attribute utilized similarly to individual chat. The **chatHistory** is an array of message objects. Each message object contains

details like the sender's walletAddress, the text of the message, the timestamp, the user, and unique message id.

### User Flow Diagram:



The User Flow Diagram illustrates the navigation paths within an application starting from the initial page. The process begins at the "Start" point and transitions to the "Home Page." From the "Home Page," users can navigate to several different pages: Staking Page, Borrow Land Page, Chat Page, BTCGPT Page. When a user navigates to the "Chat Page," the page is only readable by default, which restricts chat functionality until the user logs in. To gain full access to chat features, the user must go through the "Login Page" and connect their wallet.

At the "Wallet Connected" decision point: If the wallet is connected ("Yes"), the user is able to chat, view the " friend list" page, granting full access to chat and

friend list functionalities. If the wallet is not connected ("No"), the user is redirected back to the "Chat readable only" page.