Network Architecture

Users:

1- **Student**:

a. Functions:

i. Register:
   - Name* (To be included in the database only)
   - Age*
   - Gender*
   - University Email*
   - MetaMask address*
   - Phone number
   - Username*
   - (Users are free to use their real names or any fake ones to be visible)
   - Password*
   - Faculty/ department*
   - Year*
   - ID*
   - Bio

ii. Log in: using username and password.

iii. Ask a Question

iv. Answer a Question

v. Vote

vi. Give Awards

vii. Post

b. Attributes:

i. Questions Tofu (reputation)

ii. Answers Tofu (reputation)

2- Academic Staff

a. **Functions:**

i. Register: Same data needed as the students in addition to the following:
   - ➢ ASID* (Academic staff ID Unique ID generated by the university to the academic staff)
   - ➢ Position* (Professor or TA)

ii. Log in: using username and password

iii. Ask a Question

iv. Answer a Question

v. Vote

vi. Give Awards

vii. Post

b. **Attributes:** To be determined

How can I interact with the community?

- **Asking questions**
  All users will be able to post questions to the community. Questions will be tagged as a question with tag (Q). No needed tokens or tofu coins to ask questions. These questions are academically or technically related, meaning that these questions will likely be about courses in the university.
- **Answering questions**
  All users will be able to provide answers to the questions asked based on their knowledge.
- **Voting**
  Voting is a way that determines what is displayed in the website. It's a way to show users insightful contributions to the community. Posts with more votes are what other users think is a good or publicly acceptable contribution. We encourage users to upvote the content that they like, to see more of it on the website.
- **Posting**

Posting is a feature where all community members can share their thoughts or opinion with the community. They can also share materials and non-academic related topics like a closed group in Facebook community. Posts are a fun way to connect with other community members so they will not be awarded with points or tokens but can be awarded with awards.

- **Giving awards**

  Awards are a more sophisticated way of showing appreciation and recognizing the contributions of other users. Questions or answers are highlighted to make it more special and stand out from other contributions.

What is in it for me?

- **Tofu**

  Tofu is the reputation point of the system. Its an indicator to how much the user has contributed to the community whether with questions or answers.
  Users will gain Tofu when others upvote their questions or answers, The rating of the academic staff will also grant the user Tofu points. The number of points gained will depend on other parameters like the age of the questions or answer and the rating of the academic staff.

- **Bateekh**

  Bateekh is the main currency (Token) of our website, it will be used in all transactions. They are also a form of tangible reward, by which users can exchange for services on the website or acknowledgement from the university.

How does being anonymous on the website work?

The website gives users the option to overshare without revealing their real username or identity. When a user signs up, he'll be asked to create a username. He will have the option to interact with the website anonymously or with showing this username.

Calculations behind the Scenes:

In this section we will provide the exact formulas used by our system to calculate the total Tofu points of a user.

Tofu points = Asking points + Answering points

These asking and answering points depend on the voting of the users in addition to the rating of the academic staff as follows:

1. **For upvotes of both questions and answers** where the weight of the upvotes decreases with time, we have:

   U -> number of upvotes for the question,

   W -> weight of the upvotes based on the time decay factor T in addition to the user type Y, and

   P -> total points earned by the user who posted the question.

   Then, the formula for P is:

   P = U * W, where W is calculated using the formula:

   W = [1 - ((current_time - question_post_time) / (T * 86400))] * Y

   Where Y can have the following values:

   = 1 (for students)

   = 1.25 (for TAs)

   = 1.5 (for professors)

   & T will be defined later after testing based on analyzing the users' behavior.

2. **For the rating of the academic staff** where professors have higher weight than TAs, we have:
   R -> the rating gained as a ratio out of 10 (RT for TAs / RP for Professors)

   WT -> weight of TAs

   WP -> weight of Professors

   P -> total rating points earned by the user who posted the question

   Then, the formula for P is:

   P = RP * WP + RT * WT

Advanced Techniques:

PageRank Algorithm

PageRank is an algorithm developed by Larry Page and Sergey Brin while they were graduate students at Stanford University, and it is the foundation of Google's search engine. The algorithm is designed to rank the importance of web pages based on the number and quality of links pointing to them.

The basic idea behind PageRank is that a page is more important if it has many links pointing to it from other pages, and if those pages themselves are important. In other words, a page that is linked to by many other important pages is itself likely to be important.

To calculate the PageRank score for a particular page, the algorithm considers both the number of links pointing to the page, and the quality of those links. Quality is determined by the PageRank scores of the pages that are linking to the page. The higher the PageRank score of the linking pages, the more weight their links will carry.

The algorithm works by starting with an initial estimate of the PageRank scores for each page in the network, and then iteratively updating those scores based on the links between the pages. In each iteration, the algorithm recalculates the PageRank scores for each page based on the updated scores of the pages that link to it. This process continues until the PageRank scores for each page converge to a stable value.

PageRank has been a foundational algorithm in the development of web search engines, and it has inspired many other algorithms for ranking pages and determining the relevance of search results.

Implementation:

To implement the PageRank algorithm for users, we would first need to define what constitutes a "link" between users. One possible definition could be when one user upvotes or downvotes another user's posts or comments. We can consider the user who upvotes/downvotes as the "source" and the user who receives the vote as the "target".

Once we have defined our links, we can build a network graph of users and their links. The PageRank algorithm can then be applied to this network to calculate the importance of each user. The basic idea is that users who receive many votes from important users (i.e., those with high PageRank scores) will themselves have high PageRank scores.

The PageRank algorithm can be adapted to work on this network of users by following these steps:

1. Build the network graph: Each user is represented as a node in the graph, and a directed edge is created from the source user to the target user for each vote. If a user receives multiple votes from the same source, these can be combined into a single edge with a weight equal to the number of votes.
2. Assign initial PageRank scores: Each user is assigned an initial PageRank score of 1/N, where N is the total number of users in the network.
3. Iteratively update the PageRank scores: In each iteration, the PageRank score for each user is updated based on the PageRank scores of the users that link to it. The formula for calculating the new PageRank score for a user i is:

PageRank(i) = (1 - d) / N + d * sum(PageRank(j) * w(j,i) / out_degree(j)), where j are the users that link to i, w(j,i) is the weight of the link from j to i, out_degree(j) is the number of outgoing links from j, d is the damping factor (usually set to 0.85), and N is the total number of users in the network.

4. Repeat step 3 until the PageRank scores converge: The iteration process is repeated until the PageRank scores converge to a stable value.

Once we have calculated the PageRank scores for each user, we can use them to rank users by importance and popularity. The users with the highest PageRank scores are considered to be the most important and popular users on the site.

the damping factor is a parameter that is used to adjust the importance given to the incoming links of a node. The damping factor is usually set to a value of 0.85, although it can be adjusted depending on the specific application.

The purpose of the damping factor is to model the behavior of users who may randomly navigate from one page to another without following a link. The damping factor essentially represents the probability that a user will continue browsing the web by clicking on a link, rather than randomly jumping to a new page.

When the PageRank algorithm is applied, the damping factor is used to determine the probability that a user will follow a link from the current page, versus randomly jumping to a new page. This is done by assuming that each time a user reaches a new page, there is a (1-d) probability that they will randomly jump to another page, and a d probability that they will continue browsing by following a link from the current page.

The damping factor is used to ensure that the PageRank scores converge to a stable value, as without it, the algorithm would not converge for certain types of networks. The damping factor helps to prevent "dead ends" in the network, where a page has no outgoing links, by allowing some of the PageRank value to flow back to other pages in the network.

Manipulating the System

Here are a few examples of how users might try to manipulate the system:

1. Creating fake accounts to upvote their own posts and comments: This is a form of vote manipulation that involves creating multiple accounts and using them to upvote their own content. Reddit's algorithms are designed to detect and prevent this kind of behavior, but some users still try to do it.
2. Participating in vote-swapping or karma-trading communities: There are some online communities where users agree to upvote each other's posts or comments in exchange for upvotes on their own content. This behavior is also against Reddit's content policy and can result in accounts being banned.
3. Reposting popular content: Some users will repost popular content that has already been posted elsewhere on Reddit in an attempt to get more upvotes. This behavior is often seen as spammy and can result in downvotes and negative feedback from other users.

To **prevent this kind of behavior**, Reddit has a variety of measures in place to detect and penalize users who try to game the system. For example, the site uses a variety of anti-spam measures, including **rate limiting**, **CAPTCHAs**, and **IP bans**, to prevent users from creating large numbers of fake accounts or posting spammy content.

In addition, Reddit **relies on** a combination of **automated systems and human moderators** to detect and remove content that violates the site's content policy. Users can also report suspicious behavior or content to Reddit's moderators for review.

Ultimately, the best way to prevent users from gaming the system is to promote a culture of integrity and authenticity on the site. By encouraging users to contribute high-quality content and engage in honest, respectful discussion, Reddit can maintain a thriving community that benefits everyone.

Two common techniques that websites use to prevent abuse and protect against spam are:

Rate limiting

Rate limiting is a technique where a website sets limits on how frequently a user can perform certain actions, such as making API calls or submitting forms. The goal of rate limiting is to prevent users from performing these actions too frequently or in large quantities, which can overload the website's servers and cause performance issues. By setting reasonable limits on these actions, a website can ensure that all users have fair access to its resources and that no single user can monopolize them.

IP bans

IP bans, on the other hand, are a more aggressive form of protection that involve blocking access to a website or service based on the user's IP address. An IP address is a unique identifier assigned to each device on the internet, and websites can use this information to identify and block users who engage in abusive or malicious behavior. By blocking a user's IP address, the website can prevent that user from accessing its resources or performing certain actions, even if they try to create new accounts or use different devices.

Note: These mechanisms can have unintended consequences if they're not used carefully. For example, rate limiting that's too strict can make it difficult for legitimate users to perform certain actions on the site, while IP bans can sometimes block entire groups of users, including those who share a network or geographic region with the abusive user. As such, websites need to balance the need for security and protection with the need for accessibility and usability for all users.

Implement Rate limiting

Implementing rate limiting on a website can be done in a variety of ways, depending on the specific needs and resources of the site. Here are a few general steps you can take to implement rate limiting:

1. Determine the actions that need to be rate limited: Start by identifying the specific actions on your website that could be abused or overloaded if performed too frequently. For example, you might want to rate limit user registrations, login attempts, form submissions, or API calls.
2. Set reasonable limits for each action: Once you've identified the actions that need to be rate limited, set reasonable limits on how frequently each action can be performed. This will depend on the resources of your site and the expected usage patterns of your users. For example, you might allow a user to make up to 10 API calls per minute, or submit up to 5 login attempts per minute.
3. Choose a rate limiting mechanism: There are several ways to implement rate limiting, including using web server modules, software libraries, or cloud-based services. Choose the mechanism that best fits your needs and resources.
4. Implement rate limiting in your code: Once you've chosen a mechanism for rate limiting, you'll need to implement it in your code. This may involve adding code to your server-side scripts or APIs to track and enforce rate limits.
5. Monitor and adjust rate limits as needed: As your website grows and usage patterns change, you may need to adjust your rate limits to ensure that they continue to be effective. Monitor your website's performance and usage patterns regularly and adjust your rate limits as needed to balance performance and security.

Overall, implementing rate limiting is an important step in protecting your website against abuse and ensuring fair access to your resources for all users. By setting reasonable limits and using effective rate limiting mechanisms, you can help ensure that your website remains secure, stable, and accessible.