

NUKE NETWORK

Seasonal PvP Staking Protocol

It's Art, It's Finance, It's Degenerate, It's A Hell Of A Good Time.

Section 1

NUKE PROGRAM

Section 2

NUKE ORACLE

Section 3

NUKE LIVE SERVICE

Section 4

NUKE CHAT

Section 5

NUKE PROFILE

Section 6

NUKE RADIO

Section 7

NUKE SITE

Section 8

THE NETWORK AND YOU

Section 1

NUKE PROGRAM

The Solana Program is an Anchor 29.0 Program built with standard tooling. It accomplishes many things;

- The season lifecycle (1.1)
- Three user state accounts per user
 - UserState
 - UserPresale (1.3)
 - UserName (1.2)
- Two global state accounts
 - PresaleState
 - ProgramState
- Token Creation and all Mint/Burn Operations (1.4)
- Item economy and abilities (1.5)
- Complex DPR Calculations (1.6)
- User Achievements (3.2)
- Millisecond to Solana Slot Time Translation (1.7)
- Key scoping and safeguards to protect user funds

Section 1.1

SEASON LIFECYCLE

Whiteboy or Snark starts a season by calling a program instruction "Create Season". Every season's token starts at a price of \$1 per \$NUKE token. When the "Create Season" instruction is called, the program mints 10% of the total amount of raised USDC (1.3) in \$NUKE tokens to the Whiteboy & Snark admin wallet. Whiteboy or Snark then manually deploys a Legacy Raydium v4 liquidity pool, pairing those minted tokens with 10% of the USDC raised, creating a pool with the starting price of \$1. This is all dependent on the amount raised and can scale down or up depending on the exact amount of USDC received from

contributors. In the case of a 200k max raise, the starting market cap would be 220k and the starting liquidity would be 20k USDC paired with 20k \$NUKE tokens. The only other tokens besides those for liquidity are those purchased by contributions.

After manually creating the liquidity pool, Whiteboy or Snark then calls another instruction called "Activate Season", providing the address of the deployed AMM pool to the program. This process is manual to allow for manual liquidity pool management, as it is an unanswered question what should be done at the end of a season, see **NUKE TIME** (1.8) below. The time between "Create" and "Activate" is probably around 10 or 15 minutes. After activation, users can claim their tokens, participate in PvP and earn DPR. If the max raise hasn't been reached then "Create" also blocks all contributions and ends the presale period.

While a season is running there are numerous configurable values which are changed by the **NUKE LIVE SERVICE** (3.5) at specified times. Every week the amount of **CRED** earned for completing missions (3.7), time to unstake and other values are changed to readjust to the amount of time left in the season. This is all done to achieve a perfect balance, allowing new users to join at any point during a season and get up to speed, while not punishing day 1 users at all.

The **NUKE TIME** is the end of a season, computed from the exact moment the activate transaction lands. It is the point at which all rewards for all users stop accruing, PvP is disabled and unstaking becomes instant. One can put the pieces together and understand the effect this may have on the chart. See section 1.8 for more specifics.

Section 1.2

TELEGRAM LINKAGE

All fully signed up users possess a state account called `UserName` which stores their numerical TG ID. Because this is created by a permissioned instruction on

the program through our pairing process, meaning only our backend can create this account, it serves as a durable source of truth for wallet to TG account mapping. More about this process is discussed in the **NUKE LIVE SERVICE** (3.6) section.

The program's chain state is the source of truth for all the backend applications underneath. All user accounts are created on chain and exist on chain, and there is no way to become a user outside of sending the transactions and completing the pairing process on **NUKENETWORK.COM** (7).

Section 1.3

PRESALE CONTRIBUTIONS

The raise for Season 1 is determined by direct presale contributions, while for subsequent seasons, the raise amount will be determined by **ITEM** sales. The raise is tracked by the program's Presale state, and maxes out at 200k USDC. Any USDC spent on the website, while the next season's raise is under the 200k limit and the user is below their personal contribution limit, is also presale entry. This makes it so that participating in PvP by purchasing and using **ITEMS** you are not only receiving the **ITEM** you paid for, but entry into next month's token.

All users possess a state account called UserPresale, which tracks presale contributions for the next season during the current season. This UserPresale state has a numerical season value that is one greater than the UserState numerical season value when a pending contribution is present, when both values are the same, this allows for a claim transaction to take place. Both values can only advance on user interaction when the Program state global value is advanced by admin interaction. This means that we can track a user's claim for the next season during the current, and then when they go to claim we can validate that this claim is valid for this season, perform the mint and reset the account's values. For example; during Season 0 (the current Program state season value on August 16th 2026) all UserPresale states have contributions recorded for Season

1. After Whiteboy or Snark goes through the "Create" and "Activate" flow (1.1), advancing the Program state global value to 1, a user can claim their presale tokens and the values are reset for Season 2 contribution tracking.

Claimed presale tokens are staked at the current program rate directly into a user's staked balance. For users to receive presale tokens into their wallet, users must unstake after claiming.

During any active season, purchasing an **ITEM** at any time performs a best effort contribution, meaning that if the user isn't at the maximum allowed contribution personally, and the program's max raise amount hasn't been reached, the amount of USDC the user spends on the site is added to their "presale" balance for the next season. So spending any USDC on the site gives you entry into next month's token and chart, within reason. This is done to ensure user retention throughout the seasons, and give the user's the maximum value for their USDC purchases. Because 10% of the raise is required for the next season's liquidity, that amount is set aside as the season progresses and the rest is profit for the team, set aside for clipping the chart during **HAPPY HOURS** and for liquidity additions if necessary during the season. This is how we plan to achieve a rolling release of subsequent season tokens without having to run some sort of token sale every month.

Section 1.4

TOKEN CREATION, MINTING AND BURNING

The program creates a new token in the "Create Season" admin instruction (1.1), and the program itself owns the mint authority. Minting is strictly controlled by the state account values and atomic tracking that is performed on admin and user interaction. It is impossible to mint tokens to a wallet manually without earning them via DPR or **ITEM** usage. The only time the admin wallet receives minted tokens is during the "Create Season" liquidity mint, which itself is a % based on the raise amount for the season being created, based on stored values

in the global program's Presale state. So every mint is directly associated with chain state and through instructions which perform math to keep the accounting perfectly straight.

Minting is performed to a user's specific staking treasury whenever they realize their reward balance into their principal via compounding. This also happens on unstaking, but instead of minting the reward tokens to the user's staking treasury the tokens are minted directly to the user's wallet and the principal is transferred from their staking treasury.

ITEMS can cause tokens to be burned by affecting the user's principal balance, and this currently is the only way the program burns tokens.

The \$NUKE token is minted on the traditional legacy Solana token program and is given the same configuration values as mainnet USDC. This is done to keep the stack as simple as possible, and use the hardest possible token program that has the most total volume and the most compatibility with every existing tool.

Section 1.5

ITEMS

ITEMS have many different abilities and are managed by the **NUKE LIVE SERVICE**. **ITEMS** are defined manually by Whiteboy or Snark, automatically created on chain by the **NUKE LIVE SERVICE** and then sold to a user via a user facing program instruction and the UI on **NUKE SITE** (7). Users can only hold 4 **ITEMS** at a time, and certain defensive **ITEMS** consume a inventory slot when equipped. Inventory is tracked on the UserState account and the individual **ITEM** state accounts. **ITEM** usage is handled via specific instructions, "Use Item" being for most **ITEMS** and others like "Shoot" being specialized for gun **ITEMS**. All **ITEM** accounts have the same shape with the values set at creation time being the effects that they have. More information about how **ITEMS** are created and managed can be found in the **NUKE LIVE SERVICE** (3.1) section.

Unwanted **ITEMS** can be pawned and put up for sale, and potentially sold. Users can receive USDC when another user buys their pawned **ITEM** although this is not guaranteed. Pawned **ITEMS** are always listed at 5 USDC. This is the primary mechanism for users to get rid of **ITEMS** which are unwanted. **NUKE LIVE SERVICE** manages these pawned **ITEMS**, only allowing eight to exist at a time. If a ninth **ITEM** is pawned the oldest unsold **ITEM** is deleted.

Section 1.6

DPR CALCULATION AND STAKING

DPR (Daily Percentage Rate) is determined by “**CRED**” (3.3) which is a unified metric of many data points that track a user’s participation. A user’s **CRED** is directly affected by every action they take on and off chain. The DPR rate which a user sees is determined by this **CRED** value. The base rate may be 1% daily interest and users will see anywhere from 0.2% to 1.8% depending on their personal **CRED**. This means that the more **CRED** a user has the higher their **PERSONAL** rate can be. Users have the ability to view other users rates using the UI on **NUKE SITE** (7). Rates must be locked by users, so active participation during a weekly rate acceleration event Happy Hour (3.5) is required. Compounding is allowed once every 24 hours normally, but during Happy Hour a user can compound as much as they want, to effectively catch the best rate. Compounding locks the current visible rate into the user’s stake and realizes the pending rewards into the user’s principal.

Whenever the reward balance or principal balance is moved by **ITEM** usage, we snapshot the previous amount of rewards, and then continue calculating from that snapshot forward with the new principal or reward balance. For example, rewards can be stolen from User A and given to User B with an **ITEM**. User A’s rewards are calculated with the same math as unstaking, snapshotted and then the amount to be stolen is reduced from the snapshotted value. User B’s rewards are snapshotted the same way, and then the amount stolen is added to the snapshotted

reward balance. Reward calculation continues using that snapshotted amount and value, so the rewards accrued after this interaction reflect the history of what happened and take into account the change in balances at the time they actually changed. Each snapshot rolls into the last, so many interactions can stack while keeping the accounting consistent. Large **CRED** movements that cross thresholds are also snapshotted, as crossing a **CRED** threshold will reduce or increase your DPR. So in that case the previous balance is snapshotted for the rewards earned at the old DPR and the new DPR is effective from the moment the snapshot occurred.

This DPR system is designed to be unabusable, reward top community members and allow for whales to earn without participating much if they so choose. The higher DPRs naturally flow to the locked in users, and the lower DPRs go to the whales who want to just set and forget a big balance for a week or two. Of course a whale can lock in and optimize their stake for maximum effect if they want.

At the start of a season, user's claim their presale directly into their stake account. All users have an individual staking account, such that a user's principal balance is reflected on Solscan. The user's claim transaction mints the tokens into their staking wallet and tracks the balance on their UserState account. Whenever a user compounds or unstakes, the reward balance is minted into the principal and transferred to the user. All minting happens on the fly by the program, such that the token supply number on Solscan is representative of the actual principal held by every user. Every user having their own programmatic staking wallet, managed entirely by the program, allows holder distribution to be easily determined using traditional tooling.

Staking is a simple one step process, transfer all tokens from the user's wallet to their staking treasury and lock a rate. Compounding realizes the pending rewards into the principal and also transfers all tokens from the user's wallet into the principal to allow users to buy the chart and add to their bag.

Compounding also locks the current visible rate. Unstaking is a two step process, there is an initial instruction that initiates a countdown and announces to the Telegram chat (4) and **NUKE SITE** (7) who is unstaking. **ITEM** usage is blocked during this unstaking period. After the unstaking period the user calls a second instruction which debits their accrued **CRED** and completes the unstake, transferring tokens to their wallet and minting their pending rewards. Defensive **ITEMS** can be stacked to prepare for an unstake, and strategy can be used, like waiting for users to fall asleep or enlisting the help of friends to stack defensive **ITEMS**. This mechanic allows the community to protect the chart themselves using **ITEMS** (1.5) and allows for everyone to know who won the most money on the battlefield of **NUKE NETWORK**.

Section 1.7

TIME OPERATIONS

All time operations on chain are handled in slots, and all display time is in minutes and seconds. This is a hard issue to solve, as the time between Solana slots is dependent on network conditions and validator performance. Our solution is a dedicated worker in **NUKE LIVE SERVICE** (3.8) that monitors the actual time between slots and updates a global Program state value that maps how many milliseconds a slot is. This is important for all display values on the site, unstake time, **ITEM** creation and staking calculations. Slots are important to use for calculations on chain as they are the only hard time metric available. Time doesn't exist on Solana and is always in flux, meaning wall clock time can drift significantly between slots. Staking calculations based on wall clock time are not effective and super error prone. This system allows for atomic ordering of time sensitive operations that must occur one after another, and allows for the actual time per slot to be achieved with an acceptable margin of error.

Section 1.8

NUKE TIME

There is an instruction which is called by an admin to end a season. Called "NUKE", it stops all staking rewards from accruing at the exact slot the **NUKE LIVE SERVICE**'s transaction lands. It disables all **ITEM** usage, and makes unstaking instant. There is no mechanism on the program that touches the chart. From the above note, section 1.1, we do not yet know what will happen. We can take a guess that everyone would dump. However, it is not certain. This is why the liquidity will remain unlocked and manually under the control of Snark and Whiteboy. We may add to it during a season, and we may pull whats left after everyone dumps. IF everyone does dump. We will decide what to do with it when we get there, and then work towards designing the liquidity deployment to be automatic. Under the control of the program and **NUKE LIVE SERVICE** for future seasons.

Section 1.9 **NUCLEAR FALLOUT**

The projected time between seasons is one week. During this **NUCLEAR FALLOUT** stage, the program is static. Nothing much happens until the next season activation. Users can still earn achievements during this time, and the **CRED** for them will be applied at the beginning of the next season by **NUKE LIVE SERVICE** (3.2).

Section 1.10 **ACHIEVEMENTS, MISSIONS AND PASSIVE CRED**

Achievements are specific state accounts that are worth a certain amount of CRED. When they are applied to a user's UserState account, a UserAcheivement account is created that is seeded off the template Achievement account. This means that Achievements are predefined on chain by **NUKE LIVE SERVICE**, and then applied to user's stake accounts with a permissioned instruction that only **NUKE LIVE SERVICE** can call when the requirements for an achievement have been met. See 3.2 for more details.

Missions and Passive CRED work differently. Because a mission can be anything, and passive CRED is a simple apply operation meant to be run many times, these are governed on chain by time limits which guard how many times they can be applied to a given account in a specific period of time. **NUKE LIVE SERVICE** manages the requirements for missions and applies the amount of mission CRED on completion of the mission's requirements. See section 3.7 for more info. Passive CRED is governed by multiple variables and is a representation of activity in the Telegram or **NUKE SITE**.

Section 1.11

HAPPY HOUR

HAPPY HOUR is a double **CRED** event and a rate acceleration mechanism. Manually initiated by Whiteboy or Snark, the **HAPPY HOUR** is managed automatically by **NUKE LIVE SERVICE**. See more about the automatic management in section 3.5. **HAPPY HOUR** is a Program State variable that is passed into every instruction, so all instructions that provide **CRED** are affected by the **CRED** multiplier.

Section 2

NUKE ORACLE

The Solana Program (1) underpins all the rest of the backend applications, and the way it does this is with the **NUKE ORACLE**. It tracks and stores all state accounts in Redis, receiving updates through a connection to Helius. Account changes trigger a process where the previous state, the new state and the diff between the two are stored in PostgreSQL for historical reconstruction of events, and fed to all the other services via Redis pub/sub over the oracle's UNIX socket. Every service catches a complete diff of every account change on chain, can query historically via a slower PostgreSQL lookup. Services can also query the current state quickly, since all current state is in Redis and living in system RAM. In this architecture, account lookups are in the single digit ms

range for current state, for even complex queries involving hundreds of accounts. We serve every single account on every single page load to give maximum visibility to clients. We serve every single account update to every single client over websocket. When someone unstakes the data is available via a connection on NUKENETWORK.COM. There is no faster way to get this data.

NUKE ORACLE's simple job is to be the provider of chain data to all site clients over web-socket and HTTP, and all other backend services via Redis.

Regarding networking infrastructure to accomplish our goal of shipping every single account update over a websocket connection to every single client at the same time; we use a 10 gb symmetrical connection, a bare metal instance with plenty of EPYC CPU, a properly tuned Linux kernel to hold the maximum amount of possible connections open at a time and a application stack leveraging uWebsockets for maximum possible throughput. All services share this same architecture for networking, which allows the throughput required for 10 thousand active users.

Section 3

NUKE LIVE SERVICE

This is the only other program besides **NUKE ORACLE** (2) that has a connection to Helius and deals with chain state directly.

This program manages the **ITEM** shop, all **ITEM** balancing (3.1), achievement creation and granting (3.2), on chain **CRED** (3.3) adjustments, timed **ITEM** effects (3.4), rate setting, **HAPPY HOUR** rate events (3.5), user assignment (3.6), mission pool and completion tracking per user (3.7). In subsections we will delve into each of these operations. The architecture of the **LIVE SERVICE** is a single big process that spawns workers based on chain input from oracle, chat messages and profile operations. It has access to every other application via their Redis socket, allowing it to make decisions based on any data point that

affect the money. Generally, the rules of what happens with **DPR** or **CRED** is determined by code in the program, and the **LIVE SERVICE** sends the transaction to make it happen in the background for users.

Section 3.1

ITEMS

The **NUKE LIVE SERVICE** manages all **ITEMS** (1.5). **ITEMS** are defined by Whiteboy or Snark in JSON, and the **NUKE LIVE SERVICE** then manages the creation/modification of every **ITEM** account based on this JSON. Any changes to JSON, **ITEM** removal or value changes changing their abilities, affect every **ITEM** account active on the field. This allows for a simple JSON manipulation to affect potentially thousands of **ITEM** accounts, and allows for instant and quick responses to balancing concerns as the season is live. It is the method by which we control the entire **ITEM** ecosystem.

Whenever a user reaches a certain threshold of **CRED**, they will receive a free **ITEM** assigned directly to their inventory. These **CRED** threshold items are only one direction, meaning, you can't lose **CRED** and then get more to abuse the threshold and get unlimited **ITEMS**. However, if your inventory is full when you cross a threshold, you won't "spend" that free **ITEM**, and if you lose **CRED** below it and go back up you WILL receive that free **ITEM**.

Section 3.2

ACHIEVEMENTS

Achievements are **CRED** boosters that get applied to a user's account for reaching certain metrics. Achievements can be anything from "Send 1000 messages in the chat" to "Use this specific **ITEM** at 3pm PST". Since the **NUKE LIVE SERVICE** has all the data from the chain via the **NUKE ORACLE** (2), and all the Telegram data via **NUKE CHAT** (4) and all the data that appears on **NUKENETWORK.COM** via **NUKE PROFILE** (5) available to it, achievements can be based on any combination of

metrics, and any series of consecutive actions. Achievements are guaranteed to be applied once per stake account by on chain logic, see 1.10 for more details.

Section 3.3

CRED

User's earn **CRED** (1.6) passively for interacting with **NUKE SITE** (7), and for being in the Telegram. The **NUKE LIVE SERVICE** tracks this and applies passive **CRED** boosts for being active. User's also earn **CRED** by posting, and likes or dislikes on the posts themselves (5). The **NUKE LIVE SERVICE** does not apply a specific amount of **CRED** per post. It takes many different data points into consideration, and uses entropy to determine whether or not to apply **CRED** for a given post. Posts that receive many likes can also boost **CRED**, but whether or not this affects the chain is up to the **NUKE LIVE SERVICE**.

Section 3.4

TIMED ITEM EFFECTS

All **ITEMS** (1.5) generate another state account called a receipt when a user uses them. This records the actions taken and the users involved. Some item actions have a duration, meaning they may affect a user's abilities for a specified amount of time. For example, an item can stop a user from compounding for 5 minutes. When that item is used on a user, it records that this **ITEM** is supposed to last until a specified Solana slot height, and that the target's UserState was affected by a compound blocker. The **NUKE LIVE SERVICE** is not aware of the compound blocker, and does no handling of the effects themselves, all that is handled by the on chain logic, it just knows the slot at which the receipt account must be processed, and the on chain logic then reverses the compound blocker by accepting the Receipt and reading it in the "Process Receipt" instruction. Because the instruction that the **LIVE SERVICE** is calling is generic, it is very reliable and not prone to logic errors. Whenever a receipt's specified slot is reached, the main process spawns a worker to call the "Process

Receipt" instruction and close that account. The worker is durable and job completion is monitored. It is durable enough to survive Solana outages, Helius outages, network instability of the hosting provider, and even total server outages for extended periods of time from multiple parties at once. All receipts are meant to be processed at a specific time, so even if the server goes down, the service will come back online and catch up quickly. Because all the actual logic lives on chain, the actions are not dependent on web2 race conditions or subject to non durable web2 Redis state. It's a simple design that effectively performs the job of making things happen to a user's state at a specific time without them or someone else clicking a button.

Section 3.5

RATE SETTING AND HAPPY HOUR

Rates are set on chain via a distribution curve using on chain entropy (1.6). This is not durable in the context of a user initiated instruction, as the transaction can be replayed and regenerated until a desirable rate is found, however in the context of a privileged service, on chain entropy is workable and good enough. This means the min/max base programmatic rate is determined by Solana application code and not controlled by any configurable web2 value or manual Whiteboy or Snark operation. Just like timed **ITEM** handling, the **LIVE SERVICE** is calling a generic instruction with no arguments. It only knows the time to call it.

HAPPY HOURS are managed by the **NUKE LIVE SERVICE** after being started by Whiteboy or Snark. Accelerated rate setting is queued up and a special rate setting instruction is used. That **HAPPY HOUR** rate setting function is gated on the manual Whiteboy or Snark instruction flipping the **HAPPY HOUR** boolean in Program State. See section 1.11. The **NUKE LIVE SERVICE** runs the rates for 1 hour and then flips the Happy Hour boolean to false. **HAPPY HOURS** have higher rates and faster rate switching, requiring a user to be locked in and lock the highest possible rate. The distribution curve applied to the rates makes the desirable

rates less likely to occur and has been tuned for optimal rate occurrence, while remaining effectively random via on chain entropy.

Section 3.6

USER ASSIGNMENT

When a user signs up to **NUKE NETWORK** they create their **UserState** account and their **UserPresale** account (1.3), and then sign a message to the **NUKE CHAT** (4) application proving they own their wallet address. Then the **NUKE CHAT** issues them a pairing code which they send to a Telegram bot, proving that this user who owns this address on Solana also owns this Telegram account. The **NUKE CHAT** then tells the **NUKE LIVE SERVICE** over Redis pub sub to create the username account for that user, storing their TG id on chain (1.2) and allowing them to participate. This assignment procedure ensures that TG id to wallet mapping is trustworthy, since the instruction to create this **UserName** account is permissioned such that only the **NUKE LIVE SERVICE** can call it.

Section 3.7

MISSIONS

Missions are assigned on a rolling 24 hour window, refreshed at UTC midnight, and given to users on **NUKENETWORK.COM** (7). The **NUKE LIVE SERVICE** tracks metrics for every user and on mission completion provides the amount of **CRED** specified in global **ProgramState** for mission completion (1.6). This is a generic instruction that the **LIVE SERVICE** calls on a specific user's state, the Solana Program has no idea about the mission requirements or what they are, it just limits the amount of missions that can be completed every day, effectively putting a limit on the amount of **CRED** a user can earn with missions. All users complete missions from a pool, and every mission type is limited. Users who are locked in can complete the easier missions by waiting for the mission pool to be refreshed on the 24 hour period, leaving those who are less locked in with the missions that may require spending USDC on **ITEMS** or are more difficult to

complete. Missions can be anything from use a specific **ITEM**, to getting another user to call you a specific slur in the Telegram group chat. Missions, like **ITEMS**, are entirely managed by the **LIVE SERVICE**, and defined by Whiteboy or Snark in a JSON file. Updates to this JSON file affect the entire mission ecosystem and allow for quick balancing and updates on the fly.

Section 3.8

SOLANA SLOT TIME TO WALL CLOCK TIME TRANSLATION

The **NUKE LIVE SERVICE** actively monitors real observed Solana slot time on mainnet (1.7), and maintains an up to date global ProgramState value "MS Per Slot". It uses this value to keep the unstaking time, duration of **ITEMS** and staking calculations consistent. We have to use Solana slots to keep the order of operations proper on chain, but the amount of time between slots is inconsistent and subject to change in the near future with Solana network upgrades, particularly the Alpenglw validator update. The **NUKE LIVE SERVICE** ensures that we react in real time to the network, and have proper wall clock timing displays on **NUKE SITE** (7) for **ITEM** durations and the unstaking delay. This allows for the best of both worlds and brings real time events to Solana in a way that hasn't been properly achieved before, to my knowledge.

Section 4

NUKE CHAT

The **NUKE CHAT** application handles all Telegram operations. It maintains access to the Telegram API, and manages multiple bot personalities. We re-implemented all the features of SafeGuard and have extended the ability to moderate the chat. Generally **NUKE NIGGA** is programmed as a truecel homophobe that hates ugly speech, not necessarily bad/racist/wrong speech. Moderation will grow with the Telegram and will be extremely light, but specifically is anti-homosexual and anti-ugly. This is done as a service to the community, since everyone has to share the same groupchat. **NUKE CHAT** also stores all messages historically,

allowing for a complete replay of every single message in the chat from beginning to end. It knows who is signed up on chain via it's connection to **NUKE ORACLE** (2)'s Redis instance, and can announce **ITEM** usage and staking events in the Telegram as they occur on chain. It receives the updates to state accounts, knows when an **ITEM** is used and on who and what it did. It publishes every chat message sent in the Telegram to **NUKE SITE** (7) as a post, which can be interacted with, liked or disliked. It provides PFP and TG account information to **NUKE PROFILE** (5), and provides access to all Telegram data to all the other services via its Redis socket.

Section 5

NUKE PROFILE

The **NUKE PROFILE** application handles all **NUKENETWORK.COM** specific posts and handles the **CRED** (3.3) earned from likes or dislikes. It is not only a profile service. It handles the replaying of the drawings on **NUKE SITE** (7), and handles the web2 **CRED**. It facilitates posts, and provides all the posting data to the other services via it's Redis UNIX socket. It receives and integrates Telegram user data from **NUKE CHAT** (4), and maintains a mapping of the Telegram data to the chain data in such a way that it can resolve a wallet to Telegram @ or Telegram first/last name, or reverse. It provides a lot of the data presented on **NUKE SITE** and it is a home to every metric that isn't related to on chain operations. It is the primary home of all the data that doesn't come from Telegram or Solana.

Section 6

NUKE RADIO

NUKE RADIO is a dynamic 24/7 AI live stream radio show, reminiscent of MTV. It is on the same data layer as every other service so it has access to the Telegram group chat (4), the **NUKENETWORK.COM** (5) posts and all of the on chain state via **NUKE ORACLE** (2). The AI in **NUKE RADIO** is a skeleton named SKELE. He is

given the ability to choose songs to play, talk with users in the Telegram about tracks, and MCs the show as it runs into forever and ever. He responds to events on chain; shouting out who used an **ITEM** on who, who unstaked and generally reporting all on chain activity as it happens with a few minutes delay. He also monitors the chat for anti-jewish rhetoric, as he has been coded as a NYC Jew that appreciates Israel, loves Benjamin Netanyahu and generally is against any form of perceived Nazi-ism. He will actively assert Israel's right to exist and verbally slaughter anyone in opposition. He is a character.

Songs are uploaded by users with YouTube or SoundCloud links to the **NUKE RADIO** application. Videos are uploaded by users via YouTube or X links. When a video is uploaded, it is recoded to 360p and then all the hard cuts in the video are analyzed. The songs are also recoded, and run through a LLM to analyze the BPM / Mood of every track, which is fed as context to SKELE so he can make an informed decision about what to play. The videos, their hard cuts being identified, are cut together according to the BPM of the song, and rendered to be played on the live stream to create an interesting visualizer. This essentially creates an endless music video that is comprised of mostly random clips.

Section 7

NUKE SITE

NUKE SITE is designed and implemented by Snark, with technical assistance from Whiteboy. It is the primary method for interaction with each of the backend applications discussed, and the Solana program. On a page load it grabs all the up to date data from each of the backend services in large HTTP requests that hit our backend applications and ultimately end up as Redis queries, and then establishes websocket connections to each of the backend services to update the local data set received in the initial load. This means that by loading the site, you have every single account loaded, every single message from the Telegram available to you, are seeing every **ITEM** use on every other user as it happens and every unstake as it happens. It is the primary way to tap into the **NUKE NETWORK**.

NUKE SITE supports dedicated **CRED** posts with optional drawings via the **NUKE PROFILE** backend. The **NUKE LIVE SERVICE** watches posting activity and properly applies on chain **CRED** where applicable. **NUKE SITE** provides a combined feed of on chain activity like **ITEM** usage, dedicated **CRED** posts and Telegram messages. Everything on this combined feed can be up-voted or down-voted, and the "Use Item" button is on each of the **ITEMS** in the feed. **NUKE SITE** also provides the entire **ITEM** economy to the user via a unique interface, reminiscent of conveyor belt sushi restaurants in Japan.

Section 8

THE NETWORK AND YOU

NUKE NETWORK is designed to be the perfect crypto project. Much of our motivation for creating this is to bring the groupchat beefs out into the open, and create a project which finally embraces the reality of crypto currency communities. We all know that when someone shills you that 20% bundle 30k shitter they are trying to kill you and end your bloodline. If you happen to buy, they'll say "no crying in the casino" when it dumps. This kind of behavior is core to what cryptocurrency is. Instead of trying to make a ponzi that lasts forever we decided to embrace the inevitable **NUKE**. Everything in crypto goes up and then goes down and we created an experience that accounts for it, embraces and thrives in it. Much like FOMO, all interactions are associated with identity. Anonymity is great for preserving your freedoms, but it isn't fun. You don't know who to crash out on when you lose money, you don't really gain clout that lasts forever when you do something honorable. On **NUKE NETWORK** you do.

This two man team isn't taking any token allocation. We are betting our personal success on **ITEM** sales. This is a love letter to everyone that spends time here. The token and chart are the community's. Serial round trippers are rewarded and given clout for their loyalty. Nothing is ever lost and everything is historically entered into an akeshic like record. We can eventually relive every message in

real time, every **ITEM** use along side it, every unstake at the time it occurred. Every single second someone spends on **NUKE NETWORK**, they are increasing it's value, and therefore, pumping their own bags. It's designed to be the perfect expression of what crypto is, to embrace all the things every other project runs from.

There are no skeletons in the closet, just on the radio. There is no cashing out on the community. There is no bundle. There is no gotcha moment. There is no end to the forever ponzi, because it dies and is reborn.

I LOVE YOU.

-whiteboy & snark