Executive Summary

Executive Summary

TCOIN is a regional currency that is introduced in 2024 the Greater Toronto Area (GTA) of Ontario, Canada. It is designed to promote local economic activity and community engagement. Here are the key features and objectives of TCOIN:

1. **Regional Focus:** TCOIN is used primarily in the GTA, encouraging residents to spend their money locally, thus supporting local businesses and reducing economic leakage.
2. **Demurrage System:** One of the unique aspects of TCOIN is its demurrage fee, a concept inspired by the economic theories of Silvio Gesell. The currency loses value over time (typically 1% every month), incentivizing holders to spend it rather than hoard it. This increases the velocity of money, which can stimulate local economic activity.
3. **Universal Basic Income (UBI):** When the TCOIN system generates a surplus, it is given back to participants as a small daily reward to every TCOIN holder who lives in the GTA region.
4. **Conversion and Value:** TCOIN is soft-pegged to the local transit fare at a 1:1 ratio. It can be converted to TTC Full Fare tickets, but at a fee, which also encourages local spending.
5. **Support for Non-Profits:** A portion of the transaction fees is directed towards supporting local non-profit organizations, integrating social benefit into the economic model.
6. **Local Economic Benefits:** By keeping money circulating within the region, the TCOIN aims to strengthen the local economy, support small businesses, and foster a sense of community.
7. **Sustainability and Social Responsibility:** TCOIN promotes sustainable and socially responsible economic practices. It often supports environmentally friendly businesses and practices, aligning with broader sustainability goals.

The price stability mechanisms of the soft-peg to the value of TTC tokens have several components revolving around the concept of reserve ratio, which is [the total value of the reserves in the system divided by the total value of circulating supply of TCOIN].

“Soft-peg” means that there is no guarantee that the value of TCOIN equals that of TTC’s Full Fare. Instead, the economic incentives are designed to bring the value back to the 1:1 peg over time. This is particularly designed to both prepare for a future price increase of the TTC Ful Fare tickets by always building a buffer to be able to handle a price increase, and to start building it back up immediately following a price increase.

Key Price Mechanisms

Here is an overview of the key mechanisms involved:

- Purchase price effects: New TCOINs are purchased and added to the circulating supply (“minted”) when there is demand for them. People buy 1 TCOIN at 101% of the nominal price (currently $3:30) then
    - Effect: STRENGTHENS the reserve ratio up towards 101% (or WEAKENS it down towards 105% if it was larger to start out)
- Demurrage: The programatic demurrage system reduces the supply of TCOI at a typical rate of 1% every month, (which equals 11.4% per year). This supply reduction mechanism can be triggered at any time by anyone, but not more frequent than daily.
    - Effect: STRENGTHENS the reserve ratio.
- Regular users wishing to sell their TCOINs and redeem them for TTC tickets (or equivalent value in CAD) can do so at a fee. Their redemption value is 90% of the face value or 90% the reserve ratio, whichever is lower.
    - Effect: STRENGTHENS the reserve ratio.
- Participating stores can typically redeem TCOIN at 97% of nominal value (or at the reserve ratio, whichever is lower.
    - Effect: STRENGTHEN for the reserve ratio (or NEUTRAL it if it’s below 97%).
- Participating stores sign up to contribute 3% of their sales towards charity. This shows as a transaction fee which is burned and reduces the circulating supply, (while at the same time adds to the charity amount available to the buyers charity of choice).
    - Effect: STRENGTHENS the reserve ratio up to 110% and NEUTRAL above that, since it would be offset by charities spending.
- Whitelisted Charities can buy TCOIN at 97% of its value, up to a max which depends on users activity.
    - Effect: WEAKENS the reserve ratio down to 97% (or strengthens it up to 97% if its low)
- Whenever Metrolinx decides to increase the price of their TTC Full Fare, this will will cause an point-in-time increase in the value of the circulating supply.
    - Effect: WEAKENS the reserve ratio
    - However, we can expect a surge in buying & minting TCOINs prior to such an event, which would STRENGTHEN the reserve ratio.
    - The effect a sell of after the price increase will depend on how far down the price increase drives the reserve ratio.
- Overhead expenses are 1% when the ratio is below 1:1 and 2% when TCOIN is strong.
    - Effect: WEAKENS the reserve ratio

Price vs Collateralization Scenarios

For any stablecoin it is crucially important that the mechanisms for bringing the value back to peg are well designed, both in terms of bringing the value back down to peg if it is overpriced and bringing it back up to peg if it is underpriced. In the case of TCOIN, we go further and design two scenarios within +-10% and two further scenarios beyond 10% from peg

Price is at or above peg

When and why would this happen?

- The underlying bitcoin went up in price
- The constant demurrage reached a point where the collateral ratio reached 110%

How to think about purchases above peg

- Summary
    - Set purchase price at or above peg = peg value (eg $3.35) plus transaction fees.
- Reasoning
    - In a “normal” stablecoin protocol, if the collateralization was at 110% we would have to consider the flash attack of buying in at 100%, voting to increase the price and then sell.
    - But we will have delegated voting by doxxed stewards, and controlled savings & distributions mechanisms for surplus, so the risk of a flash attack is small.
    - There is still the risk of governance takeover for the purpose of draining the treasury, but then again, everyone would still get their share of the surplus and we would hope that the public outcry over destroying a public good would be large enough to prevent this scenario
    - We can allow people to keep buying at 100% because there is no way an attacker could sell for more than 100%

Less than or equal to 10% over peg

Primary stabilization mechanism

- Keep selling new coins at peg (reduces ratio)
- Charge stores 3% fee but don’t mint TCOIN for charity (increases ratio)
- Standard 1% demurrage (increases ratio)

Reasoning

- We are actually NOT targeting peg value. We are targeting a ratio between 110% and 120%as our healthy range. This is to hedge against future TTC price increases.
- So below 110% we have two mechanisms in place to increase the ratio and one that decreases it

Above 10% and at or below 20% over peg

Primary stabilization mechanism

- Keep selling new coins at peg (reduces ratio)
- Charge stores 3% fee and mint as TCOIN for charity (neutral)
- Half demurrage 0.5% (increases ratio somewhat)

Further thoughts

- This is our target range. We want the value to sit between 110% and 120%, so that we are ready for the next price increase from Metrolinx

- Increased sales will unfortunately bring price down, but that's fine because increased adoption is overarchingly important.
- On the other hand, demurrage will contribute to an increasing ratio.
- We're hoping that the net effect of these two forces is somewhat neutral.

More than 20% over peg

Primary stabilization mechanism

- Mint more tokens on an hourly basis into a multisig designated to charity. Mint as many as needed to bring the ratio down to 120%
- Plus the previously mentioned mechanisms (selling at peg and half demurrage) are still active.

Further thoughts

- Distribute the minted tokens to local charities on a weekly or monthly basis
- The default charity is UBI, which means that anyone who bought tokens or used them in a store in the last week will get an allocation
- Use a protocol such as Superfluid to stream the tokens to those recipients

Price is below peg

When and why would this happen?

- The underlying bitcoin went down in price
- Metrolinxs increased their prices
- Standard demurrage (1%) plus store fees (3%)) wasn't significant enough to counteract these two.

How to think about purchases below peg.

- Reasoning
    - Why would anyone buy into an undercollateralized token at peg, when the collateral ratio indicates that it's worth less? Maybe because they believe in the currency beyond just the collateral. Or maybe because they don't care or don't inform themselves.
    - But we can expect to see significant reduction in demand, probably exponentially correlated to the undercollaterization ratio.
    - We shouldn't turn it off though. Any purchases at above current ratio will serve to increase the collateralization ratio back towards peg.
    - So we set purchase price at the halfway point between [peg minus 10%] and the current collateralization ratio. This way people would both get a form of discount and would help increase the collateralization back towards peg.
- Ratio is 96% or higher
    - Purchase price is at par
- Ratio is below 96%
    - Purchase price is at halfway point between [peg minus 4%] and current collateralization ratio (eg if peg is $3.35 the 96% value is 3.216. If the value of collateral is 3.016, then buy at $3.116)

Less than or equal to 3-4% below peg

Primary stabilization mechanism

- Buy TCOIN at peg value less 3% (increases ratio)
- Sell TCOIN at peg value plus 1% fee (increases ratio)

- Increase demurrage to double (2%, increases ratio)
- No charity minting (neutral)

Reasoning

- Collateralization fluctuations within 4% should be expected and no cause for panic.
- Increasing the demurrage to 2% is not nothing, but most people wouldn't notice it significantly. They are already used to seeing the coin count drop on a regular basis.
- We could chose 3% as the cutoff rate, because this is the level at which we are at equilibrium with the store redemptions
- Or we could chose 4% as the level of we also include the 1% minting fee for new tokens

Ratio is more than 3-4% below peg

Primary stabilization mechanism

- Buy TCOIN at RATIO value less 3% (increases ratio)
- Sell TCOIN at RATIO value plus 1% fee (increases ratio)
- Increase demurrage to quadruple (4%, increases ratio)

Scenario

- Price was 3.35 and ratio was 110% before price increase, putting the ratio price at $3.685
- Metrolinx increases price by 20% to $4.00, which drops our ratio to 92%
- Instead of raising the TCOIN price to $4.00 it is raised to $3.685 overnight
- Holders see an increase and are happy about that, but confused as to why the price didn't go up to &4.00
- Byers can buy for $3.72
- Stores redeem for $3.57
- The 4% demurrage, 1% minting fees and 3% store redemption should take caste of increasing the price toward $4 pretty soon

Reasoning

- If we use Bitcoin or USDC as back then this scenario would be triggered by a sharp drop in price
- If we use CAD as back then it probably wouldn't happen, or could happen if Metrolinx increased the price significantly.
- At this ratio people will start getting sceptical about participating
- The best course of action is to drop the price to what truly reflects the asset back, and then slowly inch the price back up with demurrage.
- Set TCOIN price is at halfway point between [peg minus 4%] and current collateralization ratio (eg if peg is $3.35 the 96% value is 3.216. If the value of collateral is 3.016, then buy at $3.116)

T-Coin Whitepaper

ETH Toronto Hackathon Entry

Launching a Toronto Coin

Let's re-democratise the monetary system

Author: Noak Lindqvist

July 24, 2024

Reading Guide:

- *If you're a hacker / blockchain developer with good understanding of ERC20, then you might want to check chapters 4-6 for an overview of the intended tokenomics, and then go to the Appendix for the smart contract structure*
- *If you're a hacker / front end developer then check out Chapter 7*
- *If this is all new to you then try reading chapters 1-3 first.*

Note: This is a part of a family of papers around the same concept.

- *This paper is a shorter and dedicated version of a more complete proposal. This version focuses solely on the electronic version of TCOIN. The full proposal whitepaper includes more thoughts on a physical incarnation of the coin as a "regular" bill (paper money with electronic signature), as well as elaboration of the broader ecosystem implications. See the full proposal here: TDAO Proposal: T-Coin ,*
- *We also have a complementary proposal called "Panhandler and Tipping App", which is intended as one way of kickstarting usage of the TCOIN once it's available. It features specific functional to support donors, panhandlers, waitresses, store managers, and city managers using a UI that's dedicated for each persona and their specific needs. Check out the proposal here: Hackathon idea: Panhandler Donations*

Contents

Chapter 1: Introduction

The emergence of digital currencies has revolutionised the way we think about money and financial transactions. Amidst this wave of innovation, TCOIN stands out as a pioneering local digital currency designed specifically for Toronto. This chapter introduces TCOIN, outlining its purpose, key features, and the unique value it brings to the community.

1.1 Overview of TCOIN

TCOIN is a local digital currency developed by the Toronto DAO. It aims to strengthen the local economy by promoting the circulation of money within Toronto, supporting local businesses, and enhancing community engagement. TCOIN leverages blockchain technology to provide a secure, transparent, and efficient medium of exchange, offering a modern alternative to traditional currencies.

Key Objectives:

1. **Economic Resilience:** TCOIN is designed to bolster the local economy by encouraging residents and visitors to spend locally, thereby keeping more value within the community.
2. **Community Support:** A unique feature of TCOIN is its integration with charitable donations, ensuring that every transaction contributes to local causes and social good.
3. **Technological Innovation:** Utilising blockchain technology, TCOIN offers secure, transparent, and tamper-proof transactions, providing users with confidence in the currency's integrity.

1.2 The Purpose of TCOIN

The primary purpose of TCOIN is to provide a stable and reliable local currency that supports economic and social objectives specific to Toronto. By focusing on the local economy, TCOIN aims to:

1. **Promote Local Spending:** Encourage residents and visitors to use TCOIN at participating local businesses, thereby supporting local entrepreneurs and fostering economic growth within the community.
2. **Enhance Financial Inclusion:** Provide an accessible financial tool for all residents, including those who may be underserved by traditional banking systems, ensuring broader participation in the local economy.
3. **Support Community Initiatives:** Direct a portion of all transactions towards local charities and social causes, reinforcing the connection between economic activity and community well-being.

4. **Encourage Sustainable Practices:** Promote environmentally sustainable and socially responsible economic practices, aligning with broader goals of sustainability and community resilience.

1.3 Key Features of TCOIN

TCOIN is built on several key features that distinguish it from other digital and traditional currencies:

1. **Blockchain-Based Technology:** TCOIN transactions are recorded on a secure blockchain ledger, ensuring transparency and immutability. This technology underpins the currency's reliability and security.
2. **Stable Value Peg:** TCOIN is pegged to the TTC Full Fare, providing a stable reference value for transactions. This pegging mechanism helps to maintain the currency's purchasing power and fosters trust among users.
3. **Charitable Contributions:** Integrated into the currency system is a mechanism for automatic donations to local charities. A portion of each transaction is allocated to a charity of the user's choice, promoting social responsibility and community support.
4. **User-Friendly Application:** The TCOIN app provides a simple and intuitive interface for managing TCOINs, making transactions, and selecting charities. The app is designed to be accessible to users of all technical levels, ensuring wide usability.

1.4 The Vision for TCOIN

The vision for TCOIN extends beyond being just a currency; it is about creating a more connected, resilient, and supportive community. By leveraging technology and innovative economic models, TCOIN aims to:

1. **Build a Strong Local Economy:** By keeping financial transactions local, TCOIN helps to circulate money within Toronto, strengthening the economic fabric of the city.
2. **Foster Community Engagement:** Through its charitable giving component and community focus, TCOIN encourages active participation in local initiatives and creates a sense of shared purpose.
3. **Serve as a Model for Other Communities:** TCOIN aspires to be a model for how local digital currencies can be implemented in other cities, promoting economic resilience, social good, and technological advancement on a broader scale.

1.5 Conclusion

TCOIN represents a bold step towards reimagining local economies in the digital age. By combining economic innovation, community support, and technological reliability, TCOIN offers a unique solution tailored to the needs and aspirations of Toronto. This introduction sets the stage for a deeper exploration of how TCOIN works, its benefits, and the roadmap for its implementation, as detailed in the subsequent chapters of this whitepaper.

Chapter 2: Background and Motivation

The concept of a local currency is not new; communities around the world have experimented with various forms of local money to promote economic resilience and community well-being. TCOIN draws inspiration from these historical and contemporary models, integrating modern blockchain technology to enhance its effectiveness and reach. This chapter explores the background and motivation behind the creation of TCOIN, highlighting its unique approach and the specific needs it addresses within Toronto.

2.1 Historical Context of Local Currencies

Local currencies have been used throughout history as tools for economic stabilisation and community empowerment. They often emerge in response to economic crises or as a means to support local businesses and economies in a globalised market.

Historical Examples:

- **The Wörgl Experiment (1932-1933):** A famous example of a successful local currency, the Wörgl experiment in Austria helped the town overcome the Great Depression by encouraging local spending and employment through the issuance of a demurrage-based currency.
- **The Chiemgauer (2003-present):** In Bavaria, Germany, the Chiemgauer currency has been used to support local businesses and nonprofits, demonstrating the viability of local currencies in modern economies.

Lessons Learned:

- These examples illustrate the potential of local currencies to boost economic activity, reduce reliance on external economies, and foster a sense of community. However, they also highlight the need for careful management and integration with broader economic systems.

2.2 Motivations for TCOIN

The development of TCOIN is driven by specific economic, social, and technological motivations, aligned with the needs and aspirations of the Toronto community.

Economic Resilience and Localism:

- **Support for Local Businesses:** TCOIN is designed to encourage spending within Toronto, helping local businesses thrive in an increasingly competitive environment. By keeping money circulating locally, TCOIN helps to build a more resilient local economy.
- **Reducing Economic Leakage:** In a globalised economy, local businesses often struggle against large multinational corporations. TCOIN aims to reduce economic leakage by incentivizing residents to spend locally, thereby retaining more wealth within the community.

Community Engagement and Social Good:

- **Charitable Integration:** A key feature of TCOIN is its integration with charitable giving, where a portion of transactions supports local non-profits. This feature aligns economic activity with social responsibility, encouraging a culture of giving and community support.
- **Fostering Community Identity:** By providing a unique local currency, TCOIN fosters a stronger sense of community identity and pride. It also empowers residents to take an active role in the economic and social development of their city.

Technological Innovation and Accessibility:

- **Leveraging Blockchain Technology:** The use of blockchain technology ensures that TCOIN transactions are secure, transparent, and efficient. This modern technological approach enhances trust and facilitates broader adoption of the currency.
- **Financial Inclusion:** TCOIN aims to provide accessible financial services to all residents, including those who are underserved by traditional banking systems. The digital nature of TCOIN, coupled with its user-friendly app, ensures that it is accessible to a wide audience, regardless of their financial literacy or access to traditional banking services.

2.3 Unique Approach of TCOIN

TCOIN differentiates itself from other local currencies and digital tokens through its comprehensive approach, which combines economic, social, and technological elements.

1. **Integrated Charitable Component:** Unlike many other currencies, TCOIN integrates charitable giving into its core operation, ensuring that every transaction contributes to social good. This feature not only enhances the community impact of the currency but also builds a strong incentive for participation.
2. **Stability Through Pegging:** TCOIN is pegged to the TTC full fare, providing a stable value reference for users. This stability is crucial for gaining user trust and ensuring that the currency can be reliably used for everyday transactions.
3. **Scalability and Adaptability:** The design of TCOIN allows for scalability and adaptability, making it a potential model for other cities and communities. By setting a precedent in Toronto, TCOIN aims to inspire similar initiatives elsewhere, promoting local economic resilience globally.

2.4 Conclusion

The background and motivation for TCOIN are deeply rooted in historical precedents and the specific economic and social needs of Toronto. By integrating lessons from past local currency experiments with modern technology, TCOIN seeks to create a more resilient, engaged, and inclusive local economy. This foundational understanding sets the stage for exploring the technical aspects, implementation strategies, and broader impacts of TCOIN, as detailed in the subsequent chapters of this whitepaper.

Chapter 3: General TCOIN Token Design

The design of the TCOIN, the digital currency unit within the TCOIN system, is central to its functionality and user experience. This chapter outlines the key components and technological aspects of TCOIN, including its blockchain foundation, transaction capabilities, and security features. The design aims to provide a stable, secure, and user-friendly digital currency tailored to the needs of the Toronto community.

This chapter explores the more generic concepts of the design. The more specific and unique design aspects are discussed thereafter, notably the **Price Pegging** of TCOIN (covered in chapter 4), **Reserve Ratio Maintenance** (chapter 5) and **Charitable Donations** (chapter 6).

3.1 Blockchain Technology

At the heart of the TCOIN system is blockchain technology, which provides the foundation for secure and transparent transactions. The choice of blockchain platform and the underlying architecture are critical for ensuring the reliability and efficiency of the currency.

Platform Choice:

- TCOIN utilises a blockchain platform known for its security, scalability, and transaction speed. The platform supports smart contracts, which are essential for automating various aspects of the TCOIN system, such as minting, burning, and charitable donations.
- The blockchain is designed to be energy-efficient, aligning with the environmental values of the community.

Decentralisation and Transparency:

- Blockchain technology enables decentralisation, reducing the reliance on a central authority and increasing trust among users. All transactions are recorded on a public ledger, ensuring transparency and immutability.
- This transparency is crucial for building trust in the system, as users can independently verify transactions and the overall supply of TCOINs.

Smart Contracts:

- Smart contracts automate key functions within the TCOIN system, such as executing transactions, handling minting and burning processes, and managing the allocation of funds to charities.
- These contracts are programmed to ensure compliance with the system's rules and to execute transactions automatically, without the need for intermediaries.

3.2 Transaction Capabilities

The TCOIN system is designed to facilitate a wide range of transactions, making it versatile for various use cases, from everyday purchases to charitable contributions.

Minting and Burning:

- **Buying & Minting:** Users can mint new TCOINs by purchasing them through the TCOIN app at a price of 105% of the TTC fare. This process not only provides new coins but also contributes to the system's reserves and charitable funds.

- **Redeeming & Burning:** When users wish to redeem their TCOINs, they can sell them back to the system at the TTC fare value. The redeemed coins are then burned, effectively reducing the total supply and maintaining the currency's value stability.

Payment Mechanisms:

The system supports both "push" and "pull" payment mechanisms.

- **Push payments:** Allow users to send TCOINs to others for purchases or transfers.
    - **Settle a debt:** Send to a friend in your list of contacts.
    - **Donations and tipping:** Scan a QR code and send payments to the person who showed it to you (who can also be added as a new contact).
- **Pull payments:** Enables merchants or service providers to request payments from users, similar to invoicing or prompting you to pay a certain amount at checkout.
    - **Pay at checkout:** Scan a QR code on the seller's app, which contains both the amount and the recipient.
    - **Invoice:** Receive a request to pay from someone in your contacts
- These mechanisms are designed to be user-friendly, enabling quick and easy transactions via the TCOIN app.

Integration with Digital Wallets:

- The TCOIN app serves as a digital wallet, allowing users to store, send, and receive TCOINs securely. The wallet features an intuitive interface, real-time balance updates, and transaction history tracking.
- Integration with other digital wallets and financial services is planned for future updates, enhancing the flexibility and utility of TCOIN.

3.3 Security Features

Security is a paramount concern in the design of TCOIN, ensuring that users' funds and data are protected against threats.

1. **Encryption and Data Security:**
2. All transactions and user data are encrypted using advanced cryptographic techniques. This encryption ensures that sensitive information, such as transaction details and user identities, remains secure from unauthorised access.
3. The blockchain’s immutable ledger further safeguards data integrity, preventing tampering or fraud.
4. **Authentication and Access Control:**
5. The TCOIN app employs multi-factor authentication (MFA) and biometric verification to ensure that only authorised users can access and manage their TCOINs.
6. These security measures protect against unauthorised access and enhance overall system security.
7. **Fraud Prevention and Monitoring:**
8. The system includes robust fraud detection and prevention mechanisms, including real-time monitoring of transactions and anomaly detection algorithms.
9. Regular security audits and updates are conducted to address vulnerabilities and enhance the system's resilience against emerging threats.

3.4 User Experience and Accessibility

The design of TCOIN is not just about technology; it also focuses on providing a seamless and accessible user experience.

1. **User-Friendly Interface:**
2. The TCOIN app features a simple and intuitive interface, making it easy for users to navigate and conduct transactions. Features such as quick-send, contact integration, and clear transaction confirmations enhance usability.
3. The app is designed to cater to users with varying levels of technical proficiency, ensuring that it is accessible to a broad audience.
4. **Accessibility Features:**
5. To ensure inclusivity, the app includes features that make it accessible to users with disabilities, such as screen reader compatibility and adjustable text sizes.
6. The app is available on multiple platforms, including iOS, Android, and web browsers, ensuring broad accessibility.

3.5 Future Enhancements

The TCOIN design is built with future scalability and enhancements in mind, allowing for the integration of new features and technologies.

1. **Scalability:**
2. The underlying blockchain infrastructure is scalable, capable of handling increasing transaction volumes as the adoption of TCOIN grows.
3. Future updates will focus on enhancing transaction speeds and reducing fees, further improving the user experience.
4. **Integration with Other Technologies:**
5. As the digital currency landscape evolves, TCOIN plans to integrate with other financial technologies and platforms, such as decentralised finance (DeFi) protocols, to offer users more financial tools and options.
6. These integrations will enhance the utility and flexibility of TCOIN, making it a more versatile and attractive option for users.

3.6 Conclusion

The TCOIN token design is a cornerstone of the TCOIN ecosystem, combining advanced blockchain technology, comprehensive security measures, and a user-centric approach. By focusing on stability, security, and accessibility, TCOIN provides a robust and reliable digital currency solution for Toronto. This design not only meets current needs but is also poised for future growth and adaptation, ensuring that TCOIN can continue to serve and benefit the community in the years to come.

Chapter 4: Price Pegging Mechanism

The TCOIN system employs a robust price pegging mechanism to maintain the stability of TCOINs, which are pegged to the value of a TTC full fare. This mechanism is crucial in ensuring that TCOINs remain a stable and reliable medium of exchange within the Toronto community.

4.1 Modified Bancor Formula

The foundation of the price stability mechanism in the TCOIN system is a modified Bancor formula. This formula helps maintain a stable price for TCOINs, ensuring that their value closely tracks the TTC fare. The Bancor formula adjusts the TCOIN supply in response to market demand and the reserve ratio, balancing the system to keep the TCOIN value consistent with the peg.

Key Features of the Modified Bancor Formula:

- **Elastic Supply:** The supply of TCOINs is dynamically adjusted based on user transactions, such as minting and redemption.
- **Stable Peg:** The primary goal is to maintain the TCOIN value at 1:1 with the TTC fare, supported by a reserve ratio mechanism.

4.2 Price Stability Protocol

To ensure price stability, the TCOIN system sets specific minting and redemption prices:

Minting Price:

- New TCOINs are minted at 105% of the TTC fare value. This slightly higher minting price ensures that the reserve grows over time, creating a buffer to manage price stability and future increases in the TTC fare.
- For example, if the TTC fare is $3.30, TCOINs would be minted at $3.465 per coin.

Redemption Price:

- TCOINs can be redeemed at 100% of the TTC fare value. This redemption rate guarantees that users can exchange their TCOINs for value equivalent to the TTC fare, ensuring reliability and predictability in the currency's value.
- Using the same example, TCOINs would be redeemed at $3.00 per coin.

4.3 Response to Major Price Fluctuations

In the event of significant price fluctuations, particularly those exceeding a 5% change in the TTC fare, the system employs additional mechanisms to maintain stability:

Reserve Ratio Management:

- The reserve ratio, which compares the system's reserves to the outstanding market cap of TCOINs, is maintained between 0.8 and 1.2. This range provides flexibility in managing fluctuations while ensuring adequate reserves for redemption.
- If the reserve ratio falls below 0.8, the system allows market forces to set the TCOIN price, reflecting the reduced reserve backing. This measure protects the system from unsustainable depletion of reserves.

Adjustment Mechanism:

- When the TTC fare increases significantly, the system may adjust the reserve ratio to accommodate the new peg level. For instance, if the TTC fare increases by more than 20%, the reserve ratio allows for a lower buy-back rate to prevent overextension of reserves.
- This mechanism ensures that even in cases of sharp increases in the TTC fare, the TCOIN system can maintain a stable and predictable currency value.

4.4 Maintaining Long-term Stability

The combination of the modified Bancor formula, controlled minting and redemption prices, and a managed reserve ratio allows the TCOIN system to provide a stable currency. This structure not only stabilises the TCOIN value but also prepares the system for long-term sustainability and resilience against market fluctuations, ensuring that TCOIN remains a reliable financial instrument for the Toronto community.

Chapter 5: Reserve Ratio Management and Charitable Donations

The reserve ratio management system in the TCOIN network is essential for maintaining the stability and reliability of TCOINs, especially in times of market fluctuations or changes in the pegged value (TTC full fare). This chapter outlines the reserve ratio framework, its importance, and the integration of charitable donations within the system.

5.1 Target Reserve Ratio

The reserve ratio is a critical metric representing the proportion of reserves held by the system relative to the outstanding market cap of TCOINs. This ratio is fundamental in ensuring that TCOINs maintain their value and can be redeemed reliably.

Nominal Reserve Ratio:

- The nominal target for the reserve ratio is set at 1.0. This means that for every TCOIN in circulation, there is an equivalent amount of reserves backing it.
- The ratio allows the system to redeem TCOINs at their full value, pegged to the TTC fare, providing confidence and stability for users.

Permissible Range:

- The reserve ratio is allowed to fluctuate between 0.8 and 1.2. This range provides a buffer to manage market volatility and unexpected changes in the TTC fare.
- **Lower Boundary (0.8):** Below this threshold, the system's capacity to maintain a stable peg diminishes, and market forces are allowed to set the TCOIN price.
- **Upper Boundary (1.2):** At this level, the system has surplus reserves, beyond which any excess is allocated to charitable causes, preventing indefinite reserve accumulation.

5.2 Impact on Currency Stability

Maintaining the reserve ratio within this specified range is crucial for several reasons:

Stability of Redemption Rates:

- When the reserve ratio is at or above 0.8, TCOINs can be redeemed at 100% of the TTC fare, ensuring that the currency remains stable and predictable for users.
- This stability is vital for user trust and for the practical use of TCOINs in everyday transactions.

Flexibility in Handling Price Increases:

- The system's ability to maintain the reserve ratio allows it to absorb increases in the TTC fare of up to 20% without reducing the buy-back rate. This flexibility helps manage significant price changes without disrupting the currency's stability.

Protection Against Reserve Depletion:

- By allowing the ratio to drop no lower than 0.8, the system prevents the complete depletion of reserves, ensuring that TCOINs retain some value even under stress conditions.

5.3 Charitable Donations

An innovative feature of the TCOIN system is the integration of charitable donations, which are linked to the minting process and the management of excess reserves.

Allocation During Minting:

- For each TCOIN minted at 105% of the TTC fare, 0.05 tokens (5%) are allocated to a charity chosen by the buyer. This mechanism not only supports community engagement but also adds a socially responsible dimension to the currency's use.

Excess Reserve Allocation:

- When the reserve ratio exceeds 1.2, indicating a surplus, the system allocates the excess reserves to local charities. This allocation prevents the unnecessary accumulation of reserves and ensures that the system's growth benefits the broader community.
- This policy reinforces the system's commitment to social good, leveraging financial technology to support local initiatives and causes.

5.4 Long-term Sustainability and Community Impact

The reserve ratio management system and the charitable donation framework collectively contribute to the long-term sustainability and community impact of TCOIN. By ensuring stable currency value, promoting responsible financial practices, and supporting local charities, TCOIN not only offers a reliable medium of exchange but also fosters a positive social and economic environment within Toronto. This integrated approach strengthens the bond between the currency and the community, encouraging widespread adoption and trust in the TCOIN system.

Chapter 6: Charitable Donations and Community Impact

The TCOIN system not only serves as a local digital currency but also incorporates a mechanism to support community engagement and social good through charitable donations. This chapter explores how these charitable contributions are integrated into the TCOIN system and their broader impact on the community.

6.1 Integration of Charitable Donations

One of the unique features of TCOIN is its automatic allocation of funds to local charities through the minting process. This integration ensures that every transaction within the TCOIN system contributes to the community, fostering a culture of giving and social responsibility.

Minting Process and Charitable Allocation:

- When users mint new TCOINs, 105% of the TTC fare is paid, with 100% of this value attributed to the user in TCOINs and the additional 5% directed to a charity of the user's choice.
- This 5% allocation is not just an added cost but a deliberate design to support local causes, ensuring that the growth of TCOIN also benefits the broader community.

Selection of Charities:

- Users can select from a list of pre-approved local charities when minting TCOINs. This list includes a diverse range of organisations focused on various causes, such as education, healthcare, environmental conservation, and social services.
- The selection process ensures transparency and allows users to support causes they are passionate about, increasing engagement and connection to the community.

6.2 Management of Excess Reserves

Beyond the regular charitable contributions made during minting, the TCOIN system has provisions for managing excess reserves, ensuring that additional resources are utilised for community benefit.

Reserve Ratio and Surplus Management:

- The TCOIN system maintains a reserve ratio target range of 0.8 to 1.2. When the reserve ratio exceeds 1.2, indicating a surplus of reserves, the system allocates these excess funds to charitable organisations.
- This mechanism prevents the unnecessary accumulation of reserves and ensures that excess resources are actively contributing to the community rather than being idle.

Community Engagement Through Donations:

- The allocation of surplus reserves to charities helps build stronger community ties and demonstrates the TCOIN system's commitment to social responsibility.
- By supporting local charities, TCOIN fosters goodwill and encourages broader adoption, as users and businesses see tangible social benefits linked to their participation in the system.

6.3 Impact on the Community

The integration of charitable donations into the TCOIN system has several positive impacts on the local community:

Supporting Local Causes:

- The funds allocated to charities through TCOIN transactions support a wide range of local initiatives, helping to address critical social, environmental, and economic issues within the community.
- This direct support enhances the capacity of local organisations to deliver services and programs that benefit the community.

Encouraging Community Participation:

- By involving users in the selection of charities, the TCOIN system promotes a sense of ownership and engagement. Users feel more connected to their community, knowing that their transactions contribute to meaningful causes.
- This engagement is further enhanced by regular updates and reports on how the funds are used, providing transparency and reinforcing trust in the system.

Promoting a Culture of Giving:

- The automatic donation mechanism embedded in TCOIN transactions encourages a culture of giving and social responsibility. This culture extends beyond individual users to businesses and institutions participating in the TCOIN ecosystem.
- As more participants engage with TCOIN, the cumulative impact of these donations grows, amplifying the positive effects on the community.

6.4 Governance Framework

Charity Inventory: The TCOIN smart contract maintains an inventory of approved charities, each represented by an index, name, and address. Only a dedicated list of admins has the authority to add or remove charities from this inventory. This controlled access ensures that only verified and legitimate entities participate in the governance process.

Steward Network: Each approved charity can nominate a steward, also identified by an index, name, and address. These stewards are tasked with governing key parameters of the TCOIN system, including the peg value and the reserve ratio limits. The voting power of a steward is proportional to the sum total of donations to the charities who nominated them, emphasising the collective decision-making process.

Stewards operate as a decentralised oracle network, collectively determining the TCOIN system's parameters. This approach mitigates the need for a centralised oracle, instead relying on the consensus among stewards to establish key economic metrics. For instance, each steward records their valuation of the peg, and the most common value becomes the new system peg. In the case of equal votes for different values, the system maintains the current peg to prevent arbitrary changes. Similarly, stewards vote on the reserve ratio's minimum and maximum bounds, with the system adopting the weighted average of these votes.

Demurrage Function: The contract includes an open function allowing the execution of demurrage, which adjusts token balances according to predefined economic rules. This function is accessible to all users, ensuring transparency and participation in the system's economic adjustments.

Access Control: The contract designates specific roles and permissions: `onlyAdmin` for managing the charity inventory and `onlyCharity` for steward appointments, as well as `onlySteward` for

governance functions. This structure replaces the typical `onlyOwner` model, promoting decentralised control and accountability. Admins, Charities and Stewards are thus empowered to maintain and adjust the system, fostering a community-driven governance model, whilst at the same time not introducing an overly arduous process for example requiring each community member to actively vote on governance parameters.

6.5 Future Directions and Expansion

The TCOIN system's commitment to charitable giving is not static; it is designed to evolve and expand over time, reflecting the needs and values of the community.

Expanding the List of Supported Charities:

- As TCOIN grows, the list of supported charities can be expanded to include more organisations and causes, ensuring diverse representation and addressing a broad spectrum of community needs.
- The community can be involved in nominating and selecting new charities, fostering a participatory approach to philanthropy.

Enhancing Transparency and Reporting:

- Ongoing efforts will be made to improve the transparency of donation flows and their impacts. Regular reporting and updates will be provided to the community, showcasing how the contributions are making a difference.
- This transparency is crucial for maintaining trust and encouraging ongoing participation in the TCOIN system.

Leveraging Technology for Greater Impact:

- The TCOIN system can leverage technology to track the impact of charitable donations more effectively, using data analytics to measure outcomes and optimise resource allocation.
- This data-driven approach can help identify areas where donations have the greatest impact, guiding future decisions and enhancing the overall effectiveness of the charitable component.

Conclusion

The integration of charitable donations into the TCOIN system highlights the currency's commitment to social responsibility and community well-being. By supporting local charities through every transaction, TCOIN not only provides a stable and innovative financial tool but also contributes to building a stronger, more connected community. This dual focus on economic stability and social good positions TCOIN as a transformative force in the Toronto community, setting a precedent for how digital currencies can be leveraged for positive social impact.

Chapter 7: Application and User Experience

The TCOIN ecosystem is designed to be user-friendly and accessible, ensuring that participants can easily engage with the currency through intuitive tools and platforms. This chapter details the functionalities of the TCOIN application, including purchasing, minting, selling, burning TCOINs, and the various transaction mechanisms available to users.

7.1 App Features

The TCOIN application is the central platform for managing TCOINs, providing users with a seamless experience for all their currency needs. The app is designed with a focus on simplicity, security, and accessibility.

Purchasing and Minting TCOINs:

- Users can purchase TCOINs directly through the app. The app supports various payment methods, including credit/debit cards and bank transfers, allowing users to easily acquire TCOINs.
- Minting occurs at 105% of the TTC fare, with the additional 5% supporting reserve accumulation and charitable donations. This minting process ensures that the currency remains backed and that the system's stability is maintained.

Selling and Burning TCOINs:

- Users can sell their TCOINs back to the system through the app, typically at 100% of the TTC fare value. This feature ensures liquidity and allows users to redeem their TCOINs for value.
- The app facilitates the burning of TCOINs during redemption, reducing the total supply and adjusting the system's reserves accordingly.

7.2 Transaction Mechanisms

The TCOIN system supports a variety of transaction types, catering to different user needs and scenarios.

Push (Payment) System:

- The app includes a "push" payment mechanism, allowing users to send TCOINs to others, such as for purchasing goods and services. This feature is similar to traditional payment systems and is designed to be quick and straightforward.
- Users can select the recipient from their contacts, enter the amount, and confirm the transaction, which is processed instantly.

Pull (Invoice) System:

- The "pull" mechanism allows businesses and service providers to request payments from users. This is particularly useful for invoicing, where the payee can generate a payment request that the user can review and approve.
- The app notifies users of incoming invoices, which they can pay directly through the app interface.

Transaction Security and Privacy:

- The app incorporates robust security features, including encryption, two-factor authentication (2FA), and biometric verification, ensuring that all transactions are secure.
- User data privacy is a priority, with the app complying with relevant data protection regulations and employing state-of-the-art security protocols to safeguard personal information.

7.3 User Interface and Experience

The design of the TCOIN app focuses on providing a user-friendly interface that simplifies the complex processes of managing digital currency.

Intuitive Design:

- The app features an intuitive design with clear navigation, making it accessible to users of all technical levels. The dashboard provides a comprehensive overview of the user's TCOIN balance, recent transactions, and options for managing their currency.
- Real-time updates and notifications keep users informed about their transactions and the status of their TCOINs, enhancing transparency and confidence in the system.

Support and Resources:

- The app includes a help section with FAQs, tutorials, and customer support options to assist users in navigating the platform and resolving any issues.
- Educational resources on digital currency, blockchain technology, and the benefits of TCOIN are also available, promoting informed participation in the TCOIN ecosystem.

7.4 Integration and Compatibility

The TCOIN app is designed to be compatible with a wide range of devices and systems, ensuring broad accessibility.

Cross-Platform Availability:

- The app should be available on major mobile platforms (iOS and Android) and as a web application, allowing users to access their TCOINs from various devices.
- Cross-platform functionality ensures that users can manage their TCOINs conveniently, whether at home, work, or on the go.

Integration with Other Financial Services:

- The app can be designed to integrate seamlessly with other financial services and digital wallets, enabling users to manage their financial assets holistically.
- Future updates may include integrations with other cryptocurrencies, payment platforms, and financial tools, expanding the app's utility and user base.

7.5 Conclusion

The TCOIN Application is a critical component of the TCOIN ecosystem, providing users with a reliable, secure, and easy-to-use platform for managing their TCOINs. By focusing on user experience, security, and broad accessibility, the TCOIN app ensures that the currency is both practical and appealing, supporting its adoption and use in everyday transactions.

Chapter 8: Implementation Roadmap

The successful deployment and adoption of T-Coin requires a comprehensive implementation roadmap, detailing the key phases and strategies necessary to bring T-Coin to life. This roadmap encompasses the development of the core technology, community engagement, and future expansion. Additionally, this chapter outlines specific goals for the ETH Toronto hackathon, focusing on the initial development efforts.

Simplified summary

MVP / Pilot:

- Get grants
- Design & build the app
- Testing, peer reviews & audit (of both contract and app)
- Community management & marketing
- First on-ramp? Cash, P2P, credit cards and/or e-transfer?
- Bank and transparency of reserves? Plaid and Oracle?
- Recruit a few shops that will accept TCOIN

Then after the initial pilot:

- Partnerships where ppl can buy TCOIN. Credit unions? Western Union?
- Partnership with liquidity provider / investors / sponsors.
- Sign up charities and their stewards
- Automated off-ramp

8.1 Building the Foundations

The ETH Toronto hackathon presents a unique opportunity to lay the groundwork for T-Coin by developing key components of the system. The focus during the hackathon will be on creating essential smart contracts, a pilot version of the T-Coin app, and exploring specialized use cases like charitable donations and tipping.

Hackathon Goal: Development of Smart Contracts:

- **Core Contracts:** Building the foundational smart contracts that manage the minting, burning, and transfer of eCoins. These contracts will implement the modified Bancor formula and reserve ratio mechanisms to ensure price stability and reserve management.
- **Charitable Donation Contracts:** Smart contracts specifically designed to automate the allocation of a portion of the minted eCoins to chosen charities. This functionality will be crucial in integrating the charitable giving component seamlessly into the T-Coin ecosystem.
- **Security and Compliance:** Ensuring that the smart contracts include robust security features to prevent fraud and ensure compliance with legal and regulatory standards. Peer-reviews and audit of the contracts.

Pilot Version of the T-Coin App:

- **Basic Features:** Developing a pilot version of the T-Coin app that includes basic functionalities such as eCoin wallet management, transaction history, and simple transaction capabilities (buying & minting, redeeming & burning, and transferring eCoins).
- **User Interface Design:** Creating an intuitive and user-friendly interface that allows users to easily navigate the app and perform transactions. The pilot version will focus on core usability, preparing for further feature enhancements post-hackathon.
- **Testing and Feedback Collection:** Conducting initial testing of the app's functionalities and gathering feedback from participants to refine and improve the user experience.

Optional: Specialised App for Donations and Tipping

- **Panhandler Donations:** Exploring the development of a lighter version of the T-Coin app specifically designed to facilitate donations to panhandlers or other charitable causes. This app would provide a simple interface for users to donate eCoins directly, promoting social good within the community.
- **Tipping in Restaurants:** A similar lightweight app could be developed to streamline tipping in restaurants and other service industries, making it easier for users to give tips using eCoins. This specialized app would highlight the flexibility and utility of T-Coin in various everyday scenarios.
- Note: We have a dedicated document to this “tipping app”. C*heck out that proposal here:* *Hackathon idea: Panhandler Donations*

8.2 Development Phases

The implementation of TCOIN will be carried out in distinct phases, each focusing on crucial aspects of the project's launch and sustainability.

Phase 1: Research and Feasibility Study

- **Objectives:** To conduct comprehensive research on the economic, technological, and regulatory aspects of implementing a local digital currency in Toronto.
- **Actions:**
    - **Economic Analysis:** Assess the potential impact of TCOIN on local businesses and the broader economy.
    - **Technical Evaluation:** Identify suitable blockchain technologies and security protocols.
    - **Evaluate on-ramp options:**.Most e-transfer and credit card providers will not accept crypto currencies. We need to find which ones do, and start the journey of setting up partnerships with them. Or alternatively list TCOIN on a peer to peer marketplace.
    - **Evaluate partnerships with local credit unions:** In the best scenario we would partner with a local credit union or similar outlets, which would handle the on-ramp and potentially also the off-ramp scenario.
    - **Design solution for transparency of reserves:** Users need confidence that the reserves are managed well. Design a system for transparency and easy access to audit the reserves.
    - **Regulatory Review:** Consult with legal experts to understand the regulatory landscape and ensure compliance.

Phase 2: Technical Design and Development

- **Objectives:** To develop the technological infrastructure and applications necessary for the operation of TCOIN.
- **Actions:**

    - **Blockchain Development:** Establish a secure and transparent blockchain ledger for tracking transactions.
    - **App Development:** Design and build the TCOIN app, ensuring it is user-friendly and secure.
    - **System Testing:** Conduct rigorous testing of the blockchain and app to identify and resolve any issues before launch.
    - **Financial Setup:** Finalise partnerships with at least one of each of: On-ramp, off-ramp, bank account to hold the reserves, Plaid to report on the reserves, etc. Set up accounts with them and integrate APIs as needed.
    - **Develop Processes and Procedures:** Ensure we have roles and responsibilities clear for how to take payments, mint and burn TCOIN, pay merchants, evaluate and approve charities, assign and train stewards, run the DAO, add or remove approved assets, etc.

Phase 3: Regulatory Compliance and Partnership Building

- **Objectives:** To secure regulatory approvals and build partnerships with local stakeholders, including businesses and community organisations.
- **Actions:**
    - **Regulatory Approvals:** Work with regulatory bodies to ensure TCOIN meets all legal requirements.
    - **Stakeholder Engagement:** Engage with local businesses, financial institutions, and community leaders to build support for TCOIN.
    - **Partnership Agreements:** Formalise partnerships with entities that will accept or support TCOIN, including local businesses and public services.
    - **Political Lobbying:** Talk to elected representatives about not only supporting TCOIN, but also adding it to the Metrolinx mandate to sell us TTC Full Fares for the reserve, and accepting TCOIN as payment on public transit. And to accept TCOIN as payment for business licensing fees and other taxes.

Phase 4: Pilot Program

- **Objectives:** To test the TCOIN system in a controlled environment, gather user feedback, and refine the system based on real-world usage.
- **Actions:**
    - **Pilot Launch:** Implement a limited rollout of TCOIN in a specific district or among a select group of users.
    - **User Feedback Collection:** Gather feedback from pilot participants to identify strengths and areas for improvement.
    - **System Refinement:** Make necessary adjustments to the system and app based on pilot feedback, ensuring readiness for full-scale launch.

Phase 5: Full-Scale Launch and Marketing

- **Objectives:** To officially launch TCOIN to the wider Toronto community and promote its adoption.
- **Actions:**
    - **Marketing Campaign:** Launch a comprehensive marketing campaign to raise awareness about TCOIN, including digital marketing, public relations, and community events.
    - **Incentive Programs:** Offer incentives for early adopters, such as discounts, bonuses, or rewards for using TCOIN.
    - **Expansion of Acceptance:** Work to expand the network of businesses and services accepting TCOIN, enhancing its utility and accessibility.

Phase 6: Post-Launch Evaluation and Expansion

- **Objectives:** To evaluate the impact of TCOIN, ensure system stability, and explore opportunities for expansion.
- **Actions:**
    - **Impact Assessment:** Conduct a thorough evaluation of TCOIN's economic, social, and operational impacts.
    - **Continuous Improvement:** Implement ongoing improvements based on evaluation findings and user feedback.
    - **Expansion Planning:** Explore opportunities to expand TCOIN to new areas, services, or partnerships, potentially scaling the model to other regions or communities.

8.3 Community and Regulatory Engagement

Engaging the community and regulatory bodies is crucial for the successful adoption and long-term sustainability of TCOIN.

Community Engagement:

- **Educational Workshops:** Host workshops and informational sessions to educate the public about TCOIN, its benefits, and how to use it.
- **Feedback Mechanisms:** Establish channels for continuous feedback from the community, ensuring that the system evolves based on user needs and preferences.
- **Local Media Partnerships:** Collaborate with local media to share success stories and updates, fostering a positive public perception of TCOIN.

Regulatory Compliance:

- **Ongoing Dialogue:** Maintain an open dialogue with regulatory authorities to stay compliant with financial regulations and adapt to any changes in the legal landscape.
- **Transparent Operations:** Ensure that all aspects of the TCOIN system, from transaction processes to reserve management, are transparent and accountable to regulatory standards.
- **Legal Support:** Engage legal experts to navigate complex regulatory environments and ensure that TCOIN remains a lawful and secure currency option.

8.4 Future Vision and Adaptation

The implementation roadmap not only guides the initial launch of TCOIN but also sets the stage for its future development and expansion.

Technological Advancements:

- **Blockchain Innovations:** Stay at the forefront of blockchain technology, integrating new features and security enhancements as they become available.
- **Integration with Other Systems:** Explore opportunities to integrate TCOIN with other digital currencies, payment systems, or financial technologies, enhancing its versatility and reach.

Market and Geographic Expansion:

- **Expanding Use Cases:** Identify and develop new use cases for TCOIN, including partnerships with more sectors and businesses.

- **Regional Expansion:** Consider the potential for expanding TCOIN to other regions, either through direct deployment or by sharing the TCOIN model as a blueprint for other communities.

Sustainability and Social Impact:

- **Community Investment:** Continue to invest in community initiatives and charitable causes, reinforcing the social responsibility aspect of TCOIN.
- **Sustainability Focus:** Ensure that TCOIN contributes to a sustainable local economy by promoting environmentally friendly practices and supporting local businesses.

8.4 Conclusion

The implementation roadmap for TCOIN provides a clear and structured approach to launching and scaling the currency, ensuring that it meets the needs of the Toronto community while remaining adaptable to future changes and opportunities. Through careful planning, community engagement, and ongoing innovation, TCOIN aims to establish itself as a cornerstone of local economic resilience and a model for community-based digital currencies worldwide.

Chapter 9: A New Circular Economy Ecosystem

The TCOIN initiative represents a pioneering effort to create a local digital currency for Toronto, aimed at fostering economic resilience, supporting community engagement, and leveraging innovative blockchain technology. As we conclude this whitepaper, it is essential to reflect on the broader vision, the expected impacts, and the path forward for TCOIN.

9.1 Recap of TCOIN's Benefits

Economic Resilience and Local Support: TCOIN is designed to bolster the local economy by encouraging spending within Toronto, thus supporting local businesses and services. This localised economic focus can help insulate the community from broader economic fluctuations and create a more self-sustaining economic ecosystem.

Community Engagement and Charitable Contributions: A unique feature of TCOIN is its integration of charitable donations into the currency system. By allocating a portion of newly minted TCOINs to charities, TCOIN not only fosters community goodwill but also directly contributes to social causes, enhancing the overall impact of each transaction.

Innovation and Security: Utilising blockchain technology ensures that TCOIN transactions are secure, transparent, and tamper-proof. This technological foundation supports trust in the system and provides a modern, efficient method for currency transactions.

Stability and Predictability: The modified Bancor formula and reserve ratio management system provide a stable and predictable currency value, pegged to the TTC fare. This stability is crucial for user trust and practical everyday use, distinguishing TCOIN from more volatile cryptocurrencies.

9.2 Vision for the Future

The implementation of TCOIN is not merely about introducing a new currency but is a part of a broader vision to reimagine local economies and community engagement in the digital age.

Model for Other Communities: TCOIN aims to be a model that other cities and regions can adopt, showcasing how local currencies can be effectively integrated with modern technology to support local economies and communities.

Continuous Improvement and Adaptation: The TCOIN system is designed to evolve, with continuous improvements and updates to meet the changing needs of users and the community. This adaptability is crucial for maintaining relevance and ensuring that TCOIN can respond to new economic, technological, and social challenges.

Global Network of Local Currencies: Looking ahead, there is potential for TCOIN to be part of a broader network of local currencies, each tailored to the specific needs and characteristics of different communities but interoperable to enhance economic collaboration and resilience.

9.3 Call to Action

For TCOIN to succeed, it requires the active participation and support of the Toronto community, businesses, and institutions.

Engage and Participate: Community members are encouraged to engage with TCOIN, using it in daily transactions, supporting local businesses that accept it, and participating in discussions and feedback to help improve the system.

Support from Businesses and Institutions: Local businesses and institutions can play a vital role by accepting TCOIN and integrating it into their payment systems. This support not only aids in the currency's adoption but also contributes to the broader goal of local economic resilience.

Collaborate and Innovate: TCOIN is an ongoing project, and there is ample opportunity for collaboration and innovation. Whether through developing new features, expanding into new sectors, or exploring integration with other digital currencies, stakeholders are encouraged to contribute to the growth and evolution of TCOIN.

9.4 Conclusion

TCOIN is more than just a currency; it is a tool for community empowerment, economic stability, and social good. By leveraging blockchain technology and innovative economic models, TCOIN seeks to create a vibrant, inclusive, and resilient local economy in Toronto. As we move forward, the continued support, participation, and innovation from all community members will be essential in realising the full potential of this groundbreaking initiative. Together, we can build a stronger, more connected, and prosperous future for Toronto and beyond.

The TCOIN system incorporates a comprehensive governance and asset management framework, leveraging smart contracts compatible with the ERC20 standard and built using OpenZeppelin libraries for security and standardization. Key components of this system include an inventory of approved charities and a network of stewards, both critical to the decentralized governance model.

End of Document

ETH TO Developments

Hackathon Developments

Summary of Smart Contract Features

The comprehensive smart contract for the TCOIN system integrates multiple functionalities, including ERC20 token management, asset-backed token issuance, governance through stewards and charities, and economic parameter adjustments. Below are the key features and functions included in the contract:

1. ERC20 Token Management

- **Standard ERC20 Functions:** The contract includes the standard ERC20 functions (e.g., `transfer`, `approve`, `transferFrom`, `balanceOf`, `totalSupply`) to manage token transfers and balances.
- **Minting and Burning:** Functions to mint new tokens when assets are deposited and burn tokens when assets are withdrawn, maintaining a stable asset-to-token ratio.

2. Asset Management

- **Deposit and Withdrawal:** Users can deposit assets into the contract, which mints equivalent tokens, and withdraw assets by burning tokens.
- **Asset Balances:** The contract tracks individual asset balances, ensuring users' tokens are backed by actual assets held in reserve.

3. Governance and Administration

- **Charity Management:**
    - **Add/Remove Charities:** Admins can add or remove charities from the system. Removing a charity automatically unappoints their steward.
    - **Appoint/Remove Stewards:** Charities can appoint or remove stewards, who have governance powers based on the number of appointing charities.
- **Steward Voting:** Stewards vote on critical parameters such as the peg value, reserve ratios, and demurrage rate, influencing the economic model of the token.

4. Economic Parameter Management

- **Peg Value Adjustment:** Stewards can vote to update the peg value, which is crucial for maintaining the token's stability and reflecting changes in the underlying asset value.
- **Reserve Ratio Adjustment:** Stewards vote on the minimum and maximum reserve ratios, which help ensure the system's liquidity and solvency.
- **Demurrage Rate Management:**
    - **Set Demurrage Rate:** A function allows stewards to set the rate at which tokens are periodically deducted (demurrage), encouraging token circulation.
    - **Apply Demurrage:** An open function that redistributes or deducts token values based on the set demurrage rate, though implementation must be efficient to manage gas costs.

5. Security and Emergency Controls

- **Admin Roles:** Admins manage critical functions like adding/removing charities and handling emergencies, ensuring secure and controlled access to sensitive operations.
- **Emergency Functions:**
    - **Pause/Resume Contract:** Admins can pause the contract's operations in emergencies, providing a mechanism to halt all token activities to prevent or mitigate issues.
    - **Emergency Withdraw:** Admins have the authority to perform emergency withdrawals, ensuring assets can be protected in extreme situations.

Key Considerations

1. **Decentralized Governance:** The use of stewards appointed by charities ensures that governance decisions reflect a decentralized consensus, enhancing transparency and community trust.
2. **Security:** Ensuring the contract's security against common vulnerabilities (e.g., reentrancy, overflow/underflow, access controls, and thorough input validation) is essential.
3. **Gas Efficiency:** Functions involving broad changes, such as demurrage, need careful consideration for gas efficiency. Optimizations or alternative approaches (e.g., periodic snapshots) may be necessary to manage costs.
4. **Modularity and Upgradability:** While this contract outline combines several features, consider modular designs or upgradability patterns if future expansions are anticipated.
5. **Compliance and Auditing:** Regular audits and compliance checks are essential, particularly given the financial and governance aspects of the contract. Focus audit on functions that manage assets and adjust economic parameters.

The SmartContract design should integrate robust governance, secure asset management, and flexible economic parameter controls, making it suitable for a stable and transparent asset-backed token system.

Orchestration

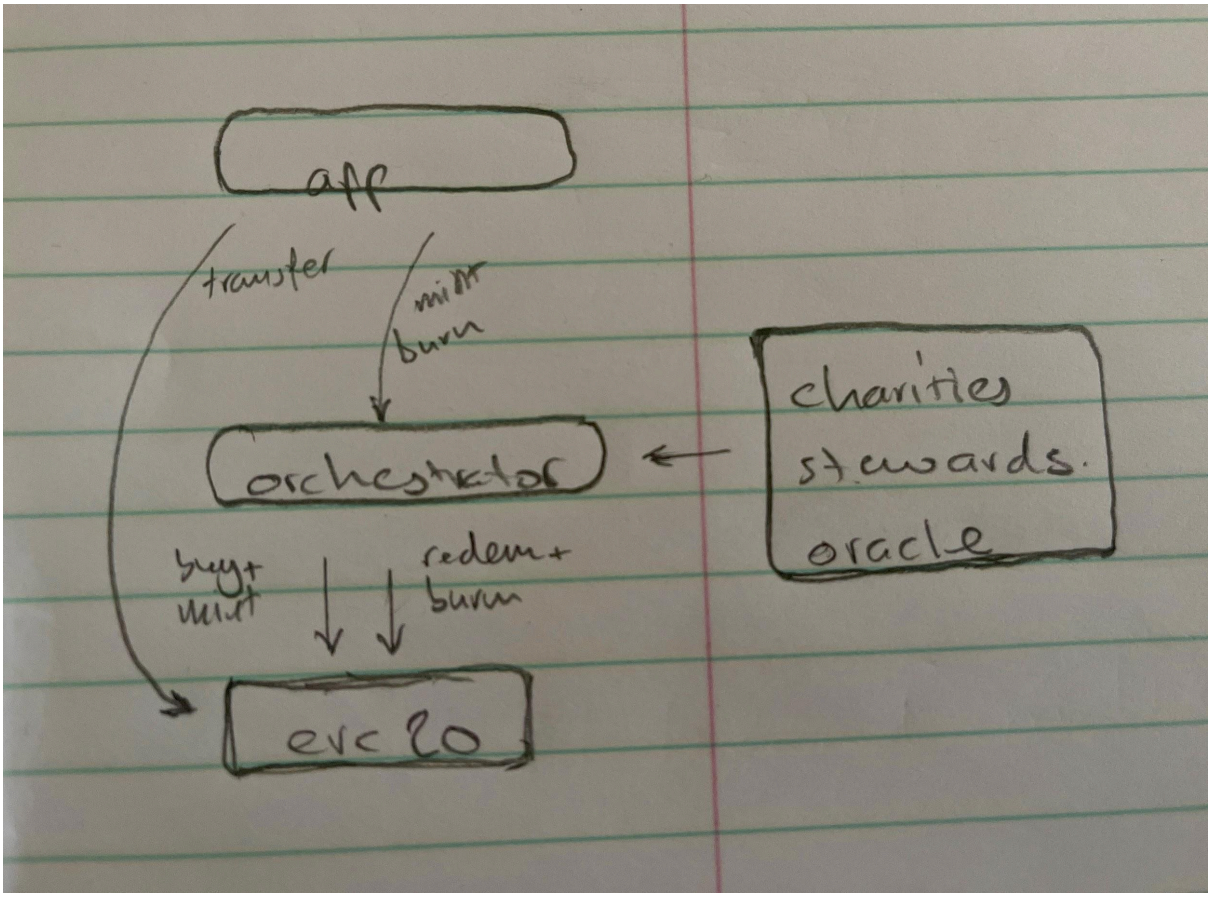
We are planning to orchestrate the flow between one admin multisig wallet, an admin app and several smart contracts:

- **Multisig Wallet**: For admin tasks
    - Assign Charities in the Oracle contract
    - Whitelist asset tokens in the Orchestrator contract
    - Assign release priority and release action per asset in the Orchestration Contract
- **Asset Tokens** (ERC20 Contracts, multiple possible)
    - Only Multisig can mint Asset Tokens which represent off-chain treasury of some corresponding asset (e.g. TTC Tickets, or CAD in a bank account)
    - One smart contract for each approved asset token
    - Note: Does not mint to caller. Only mints to the Orchestration contract
- **Oracle** Smart Contract
    - Handle charities and their assignment of Stewards
    - Handle Stewards and their "oracle-voting"
    - Calculate voting results and updating governance variables in the Orchestrator accordingly
- **Orchestration TTC** smart contract - for TTC-denominated tokens, no price oracle needed
- **Orchestration CAD** smart contract - for CAD-denominated tokens, with price oracle
    - Ensure governance variables are enacted

    - Anyone can trigger batch pre-mint TCOIN to treasury by depositing asset tokens to the orchestrator (although the Multisig will typically be the only one to do this)
    - Calculate how much to mint for each asset deposited
    - Trigger minting of TCOIN to treasury in the TCOIN contract accordingly
    - Calculate how much of which asset to release for each TCOIN burn
- **TCOIN** ERC20 Token smart contract
    - Anyone can call the transfer of a token from their own public key to another
    - Anyone can delegate access to tokens from their own public key to another
    - Only orchestrator contract can call whitelisting of stores
    - Only whitelisted Stores can call deposit & burn function (which also always reduces max supply and triggers release of assets back to the Asset Token Contracts
- **Our App**
    - Holds the private keys for majority of our users
    - Triggers transfers, delegated access, purchase2mint, and burn2redeem transactions
    - Shows users their balances and transaction history
    - Stores and retrieves added metadata per transaction from our backend database

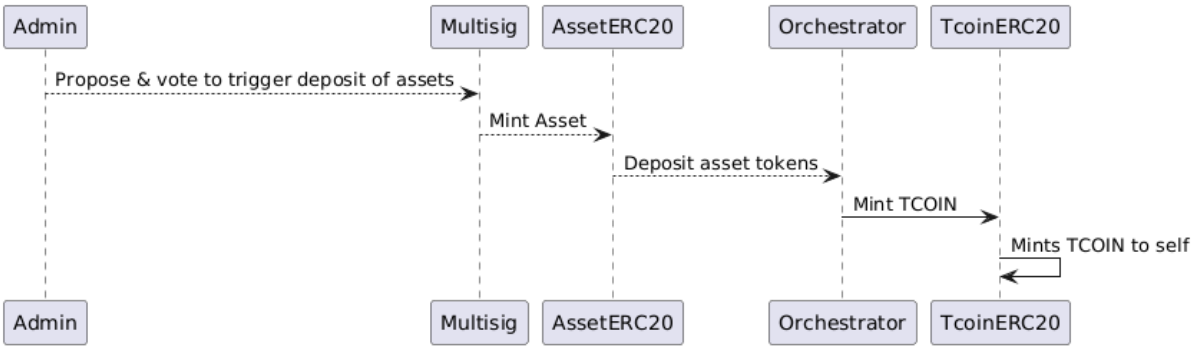
Tentative asset release logic

- If Asset released is a CAD denominated stablecoins
    - Then release it to the caller of the “deposit & burn” function
- Else if the Asset released is a CAD denominated representation of off-chain assets which are auto-deposited to the original caller
    - Then release it back to the Asset Contract and burn it there
- Else if the Asset released is a TTC denominated token representing a TTC token which needs to be mailed to the user
    - Then release it to an account representing TTC Tokens pending distribution to customer

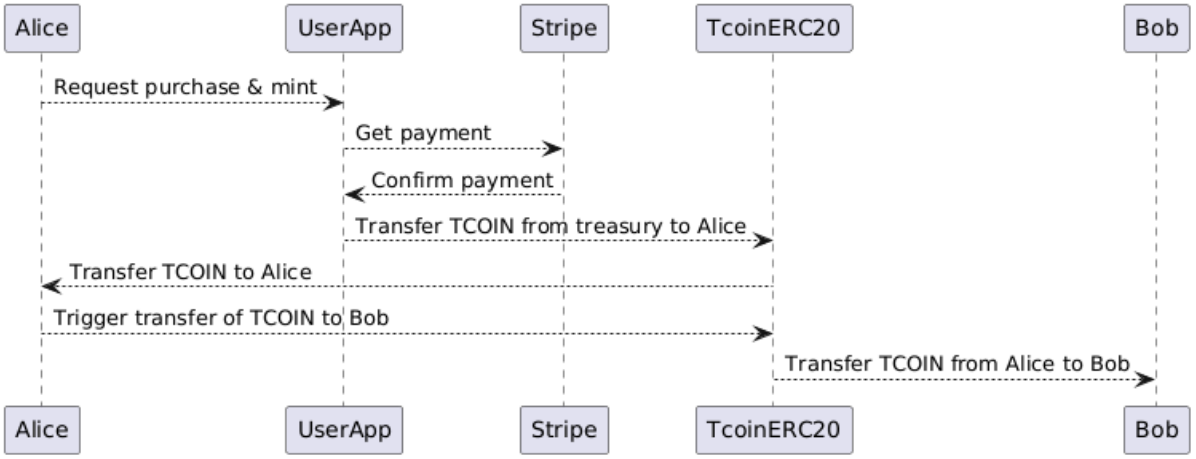


Sequence Diagrams

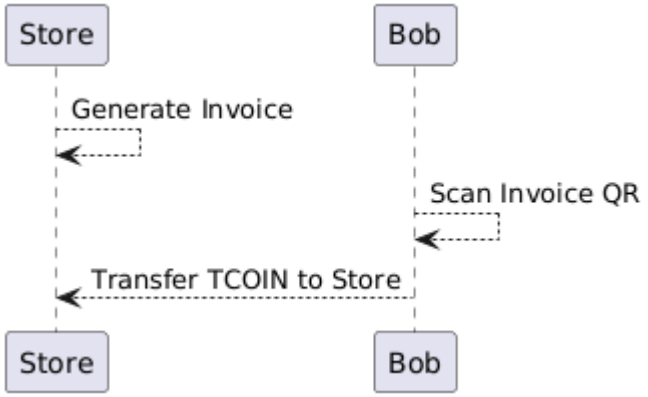
Batch Pre-mint TCOIN Tokens to Treasury



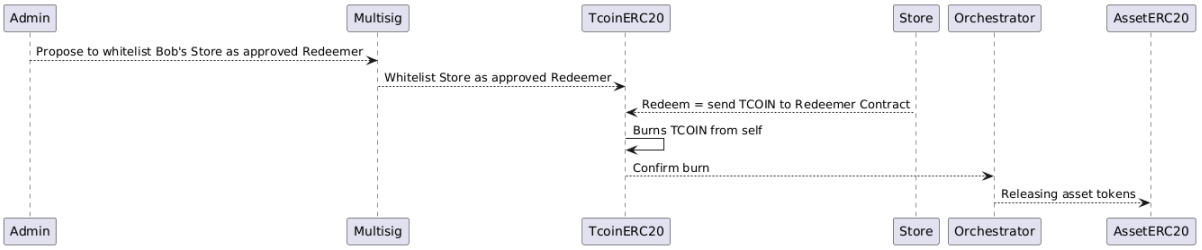
Alice Buys TCOIN and donates to Bob



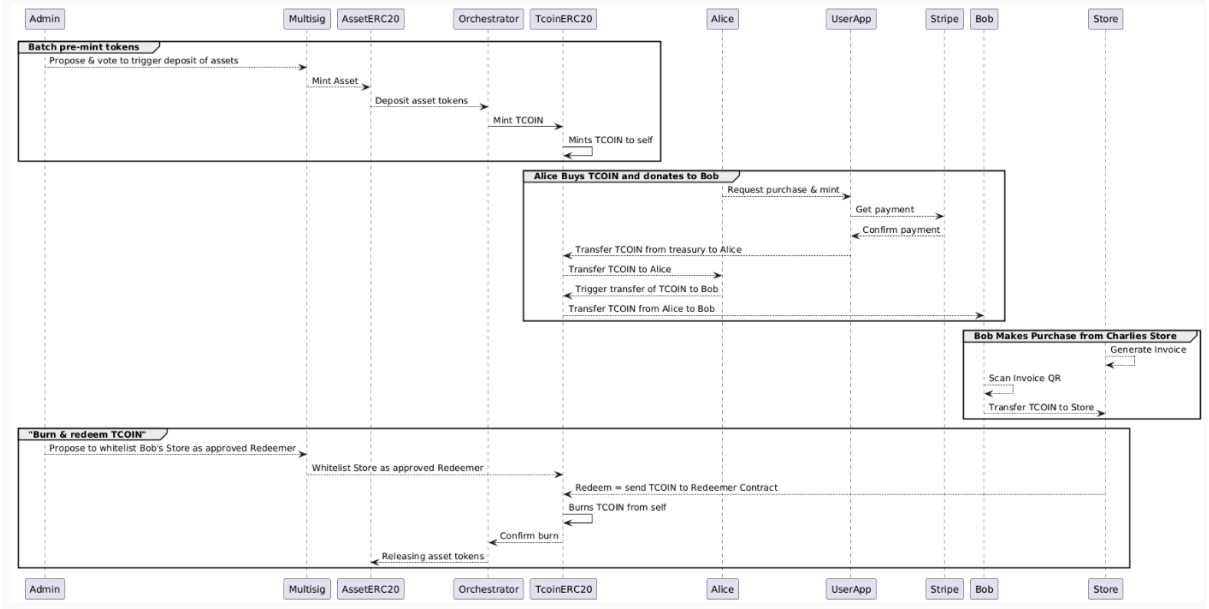
Bob Makes Purchase from Charlie's Store



Charlie Burns TCOIN & Redeems for Assets



Full Flow



To regenerate and edit the image above, paste the code below into https://www.plantuml.com/plantuml/uml/

```
@startuml
group Batch pre-mint tokens
Admin --> Multisig: Propose & vote to trigger deposit of assets
Multisig --> AssetERC20: Mint Asset
AssetERC20 --> Orchestrator: Deposit asset tokens
Orchestrator -> TcoinERC20: Mint TCOIN
TcoinERC20 -> TcoinERC20: Mints TCOIN to self
end

group Alice Buys TCOIN and donates to Bob
Alice --> UserApp: Request purchase & mint
UserApp --> Stripe: Get payment
Stripe --> UserApp: Confirm payment
UserApp --> TcoinERC20: Transfer TCOIN from treasury to Alice
TcoinERC20 --> Alice: Transfer TCOIN to Alice
Alice --> TcoinERC20: Trigger transfer of TCOIN to Bob
TcoinERC20 --> Bob: Transfer TCOIN from Alice to Bob
end

group Bob Makes Purchase from Charlies Store
Store --> Store: Generate Invoice
Bob --> Bob: Scan Invoice QR
Bob --> Store: Transfer TCOIN to Store
end

group "Burn & redeem TCOIN"
Admin --> Multisig: Propose to whitelist Bob's Store as approved Redeemer
Multisig --> TcoinERC20: Whitelist Store as approved Redeemer
Store --> TcoinERC20: Redeem = send TCOIN to Redeemer Contract
```

```
TcoinERC20 -> TcoinERC20: Burns TCOIN from self
TcoinERC20 --> Orchestrator: Confirm burn
Orchestrator --> AssetERC20: Releasing asset tokens
end
@enduml
```

Outstanding Decision to Take

Where to store transaction Metadata?

Options

- Option A: Fully leverage blockchain and smart contract
    - Issue events for each transaction which include metadata such as from, to, transaction type, amount, date
    - Build a custom Indexer which reports all transaction data to the app
- Option B: Fully leverage database
    - Build a back end database which holds a full shadow copy of the blockchain transactions, which more datapoints for each transaction
    - Blockchain events include a minimum of metadata
    - Use any standard indexer
    - Indexer not leveraged by the app
- Option C: Hybrid
    - Build a back end database which holds only complementary information that's not on chain.
    - Blockchain events include a minimum of metadata
    - Use any standard indexer
    - Our app pulls data first from the indexer, and then adds metadata from our database of complementary metadata.

Option A: Fully Leverage Blockchain and Smart Contracts

- t Data Storage: On-chain with smart contract events
- Data Completeness: Comprehensive, all transaction details included
- Scalability: Limited by blockchain capacity
- Cost: Higher, due to on-chain storage costs
- Data Security: High, secured by blockchain
- Latency: Potentially higher, due to blockchain transactions
- Ease of Development: Complex, requires smart contract development
- Data Accessibility: High for public data, limited for private data
- Transparency and Trust: High, blockchain is transparent
- Flexibility in Data Structure: Low, constrained by smart contract design
- Regulatory Compliance: Depends on blockchain and data handling policies
- Operational Complexity: High, involves blockchain and smart contracts

Option B: Fully Leverage Database

- Data Storage: Off-chain in a centralized database
- Data Completeness: Extensive, includes additional metadata
- Scalability: Scalable, depends on database capacity
- Cost: Lower, database storage is cheaper
- Data Security: Moderate, dependent on database security
- Latency: Lower, faster database queries
- Ease of Development: Easier, standard database management
- Data Accessibility: High, controlled by database permissions
- Transparency and Trust: Lower, depends on database integrity
- Flexibility in Data Structure: High, flexible database schemas
- Regulatory Compliance: Easier to manage compliance centrally
- Operational Complexity: Moderate, standard database operations

Option C: Hybrid

- Data Storage: Combination of on-chain and off-chain data
- Data Completeness: Basic on-chain data plus complementary metadata
- Scalability: Scalable with focused database use
- Cost: Moderate, mix of on-chain and off-chain costs
- Data Security: Moderate to high, depending on implementation
- Latency: Moderate, database queries plus blockchain indexing
- Ease of Development: Moderate, requires both smart contract and database integration
- Data Accessibility: Balanced, private data off-chain, public on-chain
- Transparency and Trust: Balanced, on-chain transparency for public data
- Flexibility in Data Structure: High, flexible off-chain storage
- Regulatory Compliance: Easier to manage off-chain compliance, with on-chain transparency
- Operational Complexity: Moderate to high, integration required for hybrid model

Which Reserve Token to Choose?

Reserve token options to evaluate:

- **Actual Old School TTC Tokens**
  - Token:
    - Build our own crypto token to represent the collateral (customizable)
  - Underlying asset:
    - Extremely limited supply
  - Our pre-purchase:
    - We would have to scour Kijiji and Facebook marketplace for any sellers of the token
  - On-ramp implication:
    - Less likely that they kick us out, as underlying is not a “crypto”
- **Bulk purchase Full Fare TTC Paper Tickets**
  - Token:
    - Build our own crypto token to represent the collateral (customizable)
  - Underlying asset:
    - Expires after 5 years, so requires us to build a method to circulate the inventory
    - “Resale of tickets not permitted” - we may be prohibited from buying in the future, and stores are certainly prohibited from selling the tickets

        - https://www.prestocard.ca/en/commercial/bulk-presto-tickets
    - Our pre-purchase
        - Minimum order quantity: 100 tickets
        - $20 shipping fee per order
    - On-ramp implication:
        - Less likely that they kick us out, as underlying is not a “crypto”
    -
- **USDC**
    - Token = The Asset.
        - Does not require our own crypto token (simple)
    - Underlying:
        - Stablecoin, already has built-in value
        - More prone to hacker attacks
    - Our pre-purchase:
        - Buy on any crypto exchange
        - Send to our multisig wallet
        - Then multi-sign a transaction to send it to the smart contract and mint
    - On-ramp implication:
        - More likely that they kick us out, since underlying is a “crypto”
- **CAD Dollars**
    - Token:
        - Build our own crypto token to represent the collateral (customizable)
    - Underlying:
        - Requires incorporation and dedicated bank account
        - Fiat is subject to possible account lockups by banks
        - Fiat is subject to inflation
    - Our pre-purchase = the asset
        - No need for us to pre-purchase
    - On-ramp implication:
        - Less likely that they kick us out, as underlying is not a “crypto”

When determining the appropriate reserve asset for TCOIN, several factors should be considered to ensure that the chosen asset supports the stability, accessibility, and regulatory compliance of the currency. Here are the key areas to consider:

1. Stability and Value Preservation
    - **Pegging Mechanism**: How stable is the asset's value over time? Is it susceptible to significant market fluctuations or inflation?
    - **Over-collateralization**: Can the asset support an over-collateralization strategy to absorb market volatility?
    - **Longevity, risk of expiry**: What are the chances that the asset will experience major disruptive events and lose its value
2. Liquidity and Accessibility
    - **Market Liquidity**: How liquid is the asset? Can it be easily converted into other currencies or assets?
    - **Ease of Access**: How accessible is the asset for the TCOIN treasury and users? Are there sufficient exchanges and platforms supporting it?
3. Ease of On-Ramping
    - **Integration with Payment Systems**: Can users easily acquire and redeem the asset using traditional or digital payment systems?

    - **User Experience**: How straightforward is the process for users to exchange fiat or other cryptocurrencies for the reserve asset?
4. Oracle and Technological Considerations
    - **Oracle Integration**: How reliable and secure are the oracles used to feed price and collateral data into the system? Are they decentralised to avoid single points of failure?
    - **Blockchain Compatibility**: Does the asset integrate well with the chosen blockchain platform for TCOIN? Does it support the necessary smart contracts and infrastructure?
5. Regulatory and Legal Considerations
    - **Regulatory Compliance**: Is the asset compliant with local and international financial regulations? Are there clear guidelines on its use and reporting requirements?
    - **Legal Risks**: What are the potential legal risks associated with using the asset, especially concerning money laundering, securities laws, and tax obligations?
6. Community and Economic Impact
    - **Local Economic Focus**: Does the asset align with the local economic goals of TCOIN, such as supporting local businesses and services?
    - **Community Trust and Acceptance**: How likely are local businesses and consumers to trust and accept the asset as part of the TCOIN system?
7. Security and Counterparty Risk
    - **Custodial Security**: How secure are the storage and management options for the asset?
    - **Counterparty Risk**: If the asset involves a centralised issuer (e.g., USDC), what are the risks associated with the issuer's solvency and operational practices?
8. Scalability and Future Adaptability
    - **Scalability**: Can the asset scale with the growth of the TCOIN ecosystem? Are there limits to how much of the asset can be acquired and managed?
    - **Flexibility for Future Changes**: How easy is it to change or update the reserve asset strategy if needed in the future?

Evaluation matrix:

Evaluation Factors	Actual TTC Tokens (limited supply)	Pre-purchased Full Fare Paper Tickets (expiring)	USDC (Stablecoin)	QCAD (Stablecoin)	CAD Canadian Dollars
Value Preservation: Peg Stability	Stable, tied 1:1 to local transit fares		Very unstable, twice removed, pegged to USD, not TTC	Unstable, subject to inflation (=deflation of the money)	Unstable, subject to inflation (=deflation of the money)
Value Preservation: Longevity, risk of expiry	High. TTC has not announced any phase out of tokens.	Low, pre-purchased full fare token equivalents expire after 2 years	High, accepted by most Canadian exchanges, one of the most trusted Crypto projects	New Crypto project, soon trusted by Coinbase, potentially good value preservation	Very high longevity of Canadian Dollars
Liquidity and Accessibility	Extremely low liquidity, very few remaining in circulation, less than $5k	OK liquidity, specific to Toronto, must be an approved local community	High liquidity, globally accepted	Medium liquidity, specific to Canada, market cap approx $200k	Very high liquidity, widely accepted

		to buy it			
Pre-purchase Options For Underlying Asset	Very hard, need to scour Kijiji and Facebook Marketplace	Hard: Requires centralised partnership with Metrolinx	Easy, available on major exchanges.	Medium, available on some major exchanges	Very easy, no need to pre-purchase. Payment = underlying asset.
Ease of On-Ramping	Our integration: Medium: Requires Stripe integration for credit cards and optionally also a PAD integration For users: Very Easy: Pay with credit cards (or using PAD if we enable it)				
Oracle and Technological Considerations	Good. No need for price oracles. Value doesn't change.		Not good. Requires reliable centralised oracles for price data, as USD pricing changes constantly.	Medium. Only limited need for external oracles since prices change very infrequently.	Medium. Only limited need for external oracles since prices change very infrequently.
Regulatory and Legal Considerations	Localised regulations		Evolving global and local regulations	Evolving local regulations	Well-established Canadian regulations
Risk of Regulatory / Bank / Stripe Interference (e.g. asset seizure)	Medium: Stripe doesn't like crypto, but allows it if both tokens are within our control.	Medium: Stripe doesn't like crypto, but allows it if both tokens are within our control.	Increased risk for Stripe to kick us out, since it increases our "cryptoness"		Medium: Stripe doesn't like crypto, but allows it if both tokens are within our control.
Community and Economic Impact	Strong local economic focus		Less local focus, broader scope.	Medium, traditional currency value, new crypto concept	Neutral, traditional currency
Security: Counterparty Risk, issuer	Secure, backed by a local entity		Relatively secure, but counterparty risk exists		Secure, backed by government
Security: Counterparty Risk, holder	Toronto DAO represents a new, unknown entity, high centralised counterparty risk		If Smart Contract non-upgradeable: No counterparty risk If upgradeable: High counterparty risk		Toronto DAO represents a new, unknown entity, high centralised counterparty risk
Security: Risk of Attack	No-value utility tokens are not attractive to hack and steal		Our smart contract would be a high value target for hackers to attack and steal the underlying asset		No-value utility tokens are not attractive to hack and steal
Scalability & Future Adaptability	Limited by local infrastructure		Highly scalable, adaptable	Scalable nationally, but subject to national policies	

Customizable for Global Expansion	Reserve token concept is scalable with local customizations	No customizations, stuck with USDC (unless they choose different stablecoin)	No customizations, stuck with QCAD (unless they choose different stablecoin)	Reserve token concept is scalable with local customizations
Flexibility for Future Changes	Moderate, requires local agreements	High, easily tradeable or replaceable	High, easily tradeable or replaceable	Moderate, stable but policy-dependent

Recommendation:

- Don't use USD-denominated tokens
    - Volatility and risk of de-peg with foreign exchange rates,
    - Complexity with the need for centralized oracles.
- Build a multi-token model with a number of separate reserve tokens, all of them with zero inherent value
    - TTC-denominated Tokens
        - Token 1: Represents our centralised ownership of old school TTC Tokens
        - Token 2: Represents our centralised ownership of expiring TTC Tickets
        - Token 3: Future placeholder for a token that represents our prepurchased non-expiring Full Fares Credits with Metrolinx
    - CAD-denominated Tokens
        - Token 4: Represents our CAD-denominated bank assets
        - Token 5: Locked QCAD, a stablecoin pegged to the Canadian dollar, issued by Stablecorp, a partnership between 3iQ and Mavennet
        - Token 6: Locked CADT, a stablecoin pegged to the Canadian dollar. It is part of the ecosystem developed by TrustToken
        - Token 7: Locked CUSD (Canadian Dollar Coin), a stablecoin developed by the Canadian firm Catalyst Network Foundation
- Locked QCAD, CADT and CUSD are locked up in a separate smart contract of ours, with strict only-owner controls. This removes the inherent value from the main contract and reduces the risk of hacking and theft of reserve assets.
- Build one Stripe integration to buy the customer facing token (no need for three different Stripe integrations)
- Pros
    - Flexibility to use a combination of tokens as circumstances change
    - Can balance and optimise the asset mix
    - Less dependent on one solution, robustness for unforeseen circumstances
    - No need for TTC:CAD:USD pricing oracles
    - Customers don't need to learn about CEX, Metamask, etc crypto concepts
    - Strong local & Canadian economic focus
    - Reserve tokens are utility tokens only, hold no value and are much less prone to hacker attacks
    - Low counterparty risk with Metrolinx & Central Bank of Canada
- Cons
    - Still requires oracle to bring in the TTC:CAD exchange rate
    - More exposed to Stripe's whims. They can now object against any one of our three assets and pull their services
    - Some exposure to the expiring Full Fare tickets, so we still need to build a way to offload these

- More complicated solidity code with a multi-token smart contract, especially when it comes to calculating and managing the reserve ratio
- High counterparty risk with Toronto DAO
- Missing out on opportunity to eliminate counterparty risk with blockchain tech

Idea: My Accounts Mapping

Allow users a way to transfer between their own accounts without incurring transaction fees.

The smart contract could be built to provide a secure way to avoid transaction fees between “my own accounts”. We could accomplish this with account linking, if a user first signs a transaction identifying their intent to link accounts and verifying a unique nonce.

Key Features:

1. **Account Linking:** Users can link multiple accounts (up to a specified maximum) using signature verification.
2. **Blacklisting**: If an account is linked to multiple primary accounts, it and all related accounts are blacklisted and will no longer benefit from fee exemptions.
3. **Fee Management:** The contract checks whether a fee should be applied based on the linking and blacklisting status.
4. **Owner Controls:** The contract owner can adjust the maximum number of linked accounts and the transaction fee.

Abuse prevention

This feature could be abused though, by trying to include the same account in several such mappings, or by mapping a large amount of accounts together. We would have to ensure that any attempt to circumvent fees is handled

- Require verification of a time bound nonce when initializing the mapping.
- Set a max limit of how many accounts one user can reasonably have. We’re starting with a max of 5
- Allow including the same account in more than one mappings. But also detect this behaviour and penalize all accounts involved in this “multi-mapping”. The penalty is that all of these accounts will be charged the full transaction fee.

Sample implementation

Here is a complete smart contract sample implementation, including the logic for account linking, blacklisting, and fee management, with a cap on the number of related accounts that can be linked:

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/access/Ownable.sol";

contract FeeManagedContract is Ownable {
    uint256 public transactionFee = 100; // Fee in basis points (1% = 100)
    uint256 public maxLinkedAccounts = 5;
```

```
    mapping(address => address[]) private linkedAccounts;
    mapping(address => bool) private blacklisted;

    event AccountLinked(address indexed primaryAccount, address indexed linkedAccount);
    event AccountBlacklisted(address indexed account);
    event Transfer(address indexed from, address indexed to, uint256 amount, bool feeApplied);

    // Function to link another account to the caller's primary account
    function linkAccount(address _otherAccount, bytes memory signature1, bytes memory signature2)
public {
        require(!blacklisted[msg.sender], "Your account is blacklisted.");
        require(!blacklisted[_otherAccount], "The account you are trying to link is blacklisted.");
        require(linkedAccounts[msg.sender].length < maxLinkedAccounts, "Maximum linked accounts
reached.");

        // Check if _otherAccount is already linked elsewhere
        for (uint i = 0; i < linkedAccounts[_otherAccount].length; i++) {
            if (linkedAccounts[_otherAccount][i] != msg.sender) {
                _blacklistAccountAndRelated(_otherAccount);
                return;
            }
        }

        // Verify the signatures
        require(verifySignature(msg.sender, signature1), "Invalid signature from primary account.");
        require(verifySignature(_otherAccount, signature2), "Invalid signature from secondary account.");

        // Link the account
        linkedAccounts[msg.sender].push(_otherAccount);
        emit AccountLinked(msg.sender, _otherAccount);
    }

    // Internal function to blacklist an account and all related accounts
    function _blacklistAccountAndRelated(address _account) internal {
        blacklisted[_account] = true;
        emit AccountBlacklisted(_account);

        for (uint i = 0; i < linkedAccounts[_account].length; i++) {
            address linkedAccount = linkedAccounts[_account][i];
            if (!blacklisted[linkedAccount]) {
                _blacklistAccountAndRelated(linkedAccount);
            }
        }
    }

    // Function to check if two addresses are linked
    function isLinkedAccount(address _from, address _to) internal view returns (bool) {
        if (_from == _to) return true;

        for (uint i = 0; i < linkedAccounts[_from].length; i++) {
            if (linkedAccounts[_from][i] == _to) return true;
```

```
    }

    for (uint i = 0; i < linkedAccounts[_to].length; i++) {
        if (linkedAccounts[_to][i] == _from) return true;
    }

    return false;
}

// Transfer function that checks if the fee should be applied
function transfer(address _to, uint256 _amount) public {
    bool feeApplied = false;

    if (blacklisted[msg.sender] || blacklisted[_to]) {
        // Apply fee regardless of linked status
        uint256 fee = calculateFee(_amount);
        _transferWithFee(msg.sender, _to, _amount, fee);
        feeApplied = true;
    } else if (isLinkedAccount(msg.sender, _to)) {
        // Transfer without fee
        _transferWithoutFee(msg.sender, _to, _amount);
    } else {
        // Transfer with fee
        uint256 fee = calculateFee(_amount);
        _transferWithFee(msg.sender, _to, _amount, fee);
        feeApplied = true;
    }

    emit Transfer(msg.sender, _to, _amount, feeApplied);
}

// Example fee calculation function
function calculateFee(uint256 _amount) internal view returns (uint256) {
    return (_amount * transactionFee) / 10000; // Fee calculated based on basis points
}

// Dummy internal function to handle transfers without fee
function _transferWithoutFee(address _from, address _to, uint256 _amount) internal {
    // Implement your token transfer logic here without fee
}

// Dummy internal function to handle transfers with fee
function _transferWithFee(address _from, address _to, uint256 _amount, uint256 _fee) internal {
    // Implement your token transfer logic here with fee
}

// Function to verify the signature of an account
function verifySignature(address account, bytes memory signature) internal pure returns (bool) {
    // Implement your signature verification logic here
    return true; // Placeholder, should return true if the signature is valid
}
```

```
    // Function to change the maximum number of linked accounts, only callable by the owner
    function setMaxLinkedAccounts(uint256 _maxLinkedAccounts) public onlyOwner {
        require(_maxLinkedAccounts > 0, "Maximum linked accounts must be greater than 0.");
        maxLinkedAccounts = _maxLinkedAccounts;
    }

    // Function to change the transaction fee, only callable by the owner
    function setTransactionFee(uint256 _transactionFee) public onlyOwner {
        require(_transactionFee >= 0, "Transaction fee must be non-negative.");
        transactionFee = _transactionFee;
    }
}
```

Considerations:

- **Security**: The `verifySignature` function should be properly implemented to prevent unauthorized linking of accounts.
- **Gas Costs**: Depending on the number of accounts and complexity, consider optimizing gas usage, especially in the `_blacklistAccountAndRelated` function.
- **Integration with main implementation**: The above contract was AI generated to provide an idea of how to build this feature. Care should be taken to consider interaction with all other functions and features of our main set of contracts, if and when we implement this feature.

Smart Contract

TCOIN Smart Contract Outline

Note: This section is an outdated copy from when we were thinking of it as one smart contract. We have since separated the logic into several related smart contracts. The latest implementation can be found at https://github.com/GreenPill-TO/TorontoCoin

1. Basic Contract Information

- **Contract Name:** TCOINAssetBackedToken
- **Inherits From:** ERC20, Ownable (optional, if using OpenZeppelin's Ownable for admin control)

2. State Variables

ERC20 Token Variables: Standard ERC20 variables (e.g., total supply, balances).

- `name`: A string variable that stores the name of the token (e.g., "TorontoCoin").
- `symbol`: A string variable that stores the symbol of the token (e.g., "TCOIN").
- `decimals`: A uint8 variable that specifies the number of decimal places the token uses. This determines how divisible the token is. For example, if `decimals` is set to 18, the token can be divided down to 18 decimal places, similar to how Ether works in Ethereum.
- `totalRawSupply`: A uint256 variable that keeps track of the total supply of rawTokens currently in circulation. Holds the value that would normally be held in `totalSupply` in a normal ERC20 implementation. This value is increased during a mint event and decreased during a burn event.
- `totalSupply`: A uint256 variable that keeps track of the total supply of tokens currently in circulation, after demurrage. When `totalRawSupply` is updated during a mint or a burn event, then this variable is also updated to be equal to ***totalSupply() * rebaseFactor / 10^18***
- `balances`: A mapping from addresses to uint256 that stores the balance of each account. This is used to keep track of how many tokens each account holds.
- `allowances`: A mapping from address to another mapping of address to uint256. This mapping is used to store allowances. The `allowances` mapping records the number of tokens that an owner has allowed a spender to use on their behalf, facilitating functions like `transferFrom`.

Governance Variables:

- **Structs:** Admin, Charity, Steward and Redeemer structs to store relevant information about each (index, name, address).
- **Mappings:**
    - `admins`: Mapping of admin indices to Admin structs.
    - `charities`: Mapping of charity indices to Charity structs
    - `stewards`: Mapping of steward indices to Steward structs.
    - `redeemers`: Mapping of redeemer indices to Redeemer structs.
    - `approvedCharities`: Mapping of charity addresses to bool indicating approval as a charity.
    - `stewardVotes`: Mapping of steward addresses to their voting power.

    - `approvedRedeemers`: Mapping of redeemer addresses to bool indicating approval as a redeemer
    -
- **Counters:** `adminCount`, `charityCount`, `stewardCount`, and `redeemerCount` to track the number of active admins, charities, stewards and redeemers.

Asset Management Variables:

- `assetBalances`: Mapping to track the amount of assets deposited by each user.

Economic Variables:

- `pegValue`: Current value of the asset peg (e.g. $3.30)
- `minReserveRatio` and `maxReserveRatio`: Boundaries for the reserve ratio.
- `demurrageRate`: Rate at which demurrage is applied.
- `redemptionFee`: Fixed fee to be charged each time TCOIN is redeemed for the reserve asset.

3. Modifiers

- **onlyAdmin:** Restricts functions to admin addresses (can assign Charities, can call emergency functions)
- **onlyCharity:** Restricts functions to approved charity addresses (can assign Stewards)
- **onlySteward:** Restricts functions to approved steward addresses (can vote on variables)
- **onlyRedeemer:** Restricts functions to approved redeemer addresses (can redeem TCOINS for reserve assets with only fixed fee, without variable fee)

4. Events

- `AdminAdded`, `AdminRemoved`
- `CharityAdded`, `CharityRemoved`
- `StewardAppointed`, `StewardRemoved`
- `RedeemerAppointed`, `RedeemerRemoved`
- `AssetListed`, `AssetDelisted`
- `AssetDeposited`, `AssetWithdrawn`
- `PegUpdated`, `ReserveRatioUpdated`, `DemurrageRateUpdated`
- `Transfer`: Emitted when tokens are transferred, including zero value transfers.
- `Approval`: Emitted when the allowance of a spender is set by a call to `approve`.

5. Initial Setup

- **`constructor()`**: Set up contract with initial peg value, add the contract creator as an admin, etc.

6. Admin Functions

- **`addAdmin`**`(address _admin) onlyAdmin`: Adds a new admin.
- **`removeAdmin`**`(address _admin) onlyAdmin`: Removes an admin.

7. Charity and Redeemer Management Functions

- **addCharity**(uint256 index, string memory name, address charityAddress) external onlyAdmin: Adds a new charity. Returns the assigned index value.
- **removeCharity**(uint256 index) external onlyAdmin: Removes a charity and removes one voting power from their steward.
- **addRedeemer**(uint256 index, string memory name, address redeemerAddress) external onlyAdmin: Adds a new redeemer. Returns the assigned index value.
- **removeRedeemer**(uint256 index) external onlyAdmin: Removes a redeemer and removes one voting power from their steward.

8. Steward Management Functions

- **addSteward**(uint256 index, string memory name, address stewardAddress) external onlyCharity: Adds a steward to the list of Stewards, and also adds voting power for them to represent the charity who added them by calling `_appointSteward.` This function should also ensure that each charity cannot appoint more than one Steward at a time.
- **changeSteward**(uint256 index, uint256 index) external onlyCharity: Changes appointed steward for a charity, removing the full voting power of the Charity for the previous Steward by calling `_appointSteward` and adding that same voting power for the new Steward by calling `_removeSteward`.
- **_appointSteward**(uint256 index) internal: Adds a steward vote when appointed by a charity.
- **_removeSteward**(uint256 index) internal: Removes a steward vote when un-appointed by a charity.
- **updatePowers**() external: Can be called by anyone in the community to update the voting power of all Stewards based on any changes made by calling `_appointSteward` and/or `_removeSteward`, as well as refresh the voting power based on the most recent minting of donations to Charities.

9. Asset Management Functions + Mint and Burn

- **depositAsset**(uint256 amount) external: Our main function for purchasing TCOIN. Deposits assets into the contract reserves and mints corresponding tokens.
    - The amount to be minted needs to factor in both the `reBaseFactor` to adjust for demurrage and the `reserveRatio` including the `maxRatio` and `minRatio` to adjust for the circulating market cap vs the reserve.
    - A `mintingRatio` is also applied, typically at 1 / 105%
- **withdrawAsset**(uint256 amount) external onlyRedeemer: Our main redemption function. Withdraws assets from reserve by burning tokens.
    - The amount to be burned needs to factor in both the `reBaseFactor` to adjust for demurrage and the `reserveRatio` including the `maxRatio` and `minRatio` to adjust for the circulating market cap vs the reserve.
    - The `redemptionFee` is also applied

- **`withdrawAssetWithFee`**`(uint256 amount) external`: A backup redemption function for people who are not assigned redeemer role. Same as the previous function, but also applies a variable redemption fee = the minting fee applied again.
    - Withdraws assets from reserve by burning tokens.
    - The amount to be burned needs to factor in both the `reBaseFactor` to adjust for demurrage and the `reserveRatio` including the `maxRatio` and `minRatio` to adjust for the circulating market cap vs the reserve.
    - The `redemptionFee` is also applied
    - A `mintingRatio` is also applied, typically at 105%

10. Governance and Voting Functions

- **`votePeg`**`(uint256 newPegValue) external onlySteward`: Allows stewards to vote on a new peg value. Evaluates if the peg should be changed, and executes if the change is needed.
- **`voteReserveRatio`** `external onlySteward(uint256 newMinRatio, uint256 newMaxRatio) external onlySteward`: Allows stewards to vote on new reserve ratio boundaries. Evaluates if the ratio should be changed, and executes if the change is needed.
- **`voteDemurrageRate`**`(uint256 newDemurrageRate) external onlySteward`: Allows stewards to set the demurrage rate. Evaluates if the rate should be changed, and executes if the change is needed.
- **`applyDemurrage`**`() external`: Anyone can apply the demurrage to deduct token values from all holders based on the set rate, based on time difference between the last time it was applied and current. Use formula [rate / blocks per year * (current block - block of last applyDemurrate)] Cannot be executed too often, to minimise the effect of rounding errors. Suggested frequency is max weekly.

11. Emergency Functions (CAUTION: Consider NOT implementing these as they add to both centralization and counterparty risk)

- **`pauseContract`**`() external onlyAdmin`: Pauses the contract's operations.
- **`resumeContract`**`() external onlyAdmin`: Resumes the contract's operations.
- **`emergencyWithdraw`**`(address to, uint256 amount) external onlyAdmin`: Allows for emergency withdrawal of funds/assets.

12. Modified ERC20 Functions

- **`_rawTotalSupply`**`() internal`: Returns the total supply of minted rawTokens in circulation. Implements the standard code typically found in `totalSupply()` in normal ERC20 implementations
- **`totalSupply`**`() external`: Returns the total supply of minted tokens in circulation. The formula for this is ***= round (rawTotalSupply * rebaseFactor / 10^18)***
- **`_rawBalanceOf`**`(address account) internal`: Returns the token balance of a specified address. Implements the standard code typically found in `balanceOf()` in normal ERC20 implementations.
- **`balanceOf`**`(address account) external`: Returns the token balance of a specified address after demurrage has been applied. The formula for this is ***balanceOf(account) = rawBalanceOf(account) * rebaseFactor / 10^18***

- `_rawTransfer(address recipient, uint256 amount) internal`: Transfers a specified amount of rawTokens from the caller's account to another account. Implements the standard code typically found in `transfer()` in normal ERC20 implementations.
- `transfer(address recipient, uint256 amount) external`: Transfers a specified amount of TCOIN tokens from the caller's account to another account. Calls `rawTransfer` with the amount set to ***amount * rebaseFactor / 10^18***

13. Standard ERC20 Allowance Functions

- `approve(address spender, uint256 amount) external`: Sets the amount of tokens that an owner allows a spender to use.
- `allowance(address owner, address spender) external`: Returns the remaining number of tokens that an owner allowed to a spender.
- `transferFrom(address sender, address recipient, uint256 amount) external`: Transfers tokens from one address to another, using an allowance.
- `increaseAllowance(address spender, uint256 addedValue) external`: Increases the amount of tokens that an owner allows a spender to use.
- `decreaseAllowance(address spender, uint256 subtractedValue) external`: Decreases the amount of tokens that an owner allows a spender to use.

Comments on CAD.sol

Recommendations for the CAD Contract:

Rename totalTTCSupply:
Change to netTotalSupply: Update the variable name from totalTTCSupply to netTotalSupply for better clarity and consistency with standard naming practices. This change will make it clear that the variable represents the net circulating supply after considering minted and burned tokens.

Modify the burn Function:
Remove onlyOwner Modifier: Adjust the current burn function to remove the onlyOwner modifier. This will allow users to burn tokens from their own wallet without requiring owner privileges.
Remove the from Parameter: Simplify the burn function by removing the from parameter, ensuring that the function can only be used to burn tokens from msg.sender's wallet.

Create a New burnFrom Function:
Owner-Only with Permissions: Implement a new burnFrom function that includes the onlyOwner modifier, allowing the owner to burn tokens from other addresses.
Permission Check: Ensure that this function checks that msg.sender is the contract owner and has been explicitly granted permission to burn tokens from the specified from address. This added security step will prevent unauthorized burns and align with best practices for token management.

Check for valid inputs
Add checks with require statements to ensure that inputs are valid. For example, in the mint and burn functions, ensure that the amount is greater than zero. And in the mint function, also ensure that you're not minting to the zero address.

Other
Use SafeMath library from OpenZeppelin to prevent overflow and underflow issues in arithmetic operations.

Example

```
function mint(address to, uint256 amount) external onlyOwner {
    require(to != address(0), "Mint to the zero address");
    require(amount > 0, "Amount must be greater than zero");
    _mint(to, amount);
    totalMinted += amount;
```

```
    // Update Total Supply
    unchecked {
        totalTTCSupply = totalMinted - totalBurned;
    }
}
```

BlockHack: Key Mgmt

Hackathon Challenge #1

Build a TCOIN Web App that is also a Wallet w/ User Friendly Key Mgmt.

Challenge Overview

Develop a new, responsive single-token web-app wallet for TCOIN that prioritises user-friendliness and seamless **transaction signing without the need for third-party wallets**. We want to enhance user experience by reducing dependency on external platforms. This wallet should be a mobile-first and highly intuitive "web 2.5" experience. You should leverage Supabase for authentication and backend storage. The solution should integrate CUBID's create-user API for generating a UUID-format salt and also utilise CUBID's API for securely storing a backup secret. The app should be built using Next.js (preferred) or another suitable framework.

Core Requirements

1. **Responsive, Mobile-First Design**

    - Design the wallet with mobile users as the primary focus, ensuring an excellent user experience across various screen sizes.

2. **Seamless User Authentication**

    - Use Supabase for OTP-based email authentication.
    - Upon login, retrieve two encrypted versions of the user's private key from the backend.

3. **Encryption Strategy**

    - Multi-sourced encryption salt
        - Utilize CUBID's "create-user" API to generate a user-specific salt
        - Generate another UUID salt from the supabase backend
        - Also implement a versioned third salt from the environment variables.
        - Together, these adds a high level of entropy and security by ensuring that each user's key derivation process is tied to three distinct and separate UUID-formated identifiers from separate sources
    - Implement two encrypted versions of the user's private key:

- **Device Key Encryption**: Encrypt the private key with a device key stored in and Indexed DB locally.
- **User PIN/Password Encryption**: Encrypt the private key with a user-provided PIN (though consider the negative tradeoff on user experience…)

4. **Login and Key Retrieval Flow**

    - After OTP authentication, the server should send both encrypted versions of the private key.
    - If the device key is present in local storage, allow the user to sign transactions seamlessly.
    - On a new device, prompt the user to enter their PIN or password to decrypt their private key.

5. **Transaction Signing**

    - Enable users to sign transactions with a single click without further authentication after the initial session login.
    - For cumulative transactions exceeding e.g. $100 per day: For passkey-enabled devices require passkey authentication, for all other devices require the pin code. Let the user change the dollar threshold. Do not store the PIN code. Instead, verify the PIN by decrypting the second encrypted copy of the private key.

6. **Multi-Device Support**

    - Implement a mechanism to create a new device-encrypted key for each new device. Store this in the backend, ensuring each device is uniquely paired with an encrypted key.

7. **User Experience Considerations**

    - Minimize the need for frequent password/PIN input.
    - Ensure compatibility with password managers such that users who use them are prompted to store or update their PIN or password. Examples of commonly supported password managers include LastPass, 1Password, and built-in browser password managers like those in Chrome and Safari.
    - Provide a user-friendly option for password changes, ensuring secure replacement of the password-encrypted private key in the backend.

Forking?

It's entirely possible that you would be able to fork an existing solution to be able to achieve this end result. Some possible options **might** include:

- **Authereum**

    - **Description:** While Authereum's official service is no longer maintained, its open-source in nature and can be adapted. You can use its architecture as a base and host the solution yourself to maintain control.
    - **Benefits:** Provides a foundation for integrating email-based or passwordless authentication and semi-custodial key management, which you can adapt for complete ownership of the infrastructure.
    - **Self-Hosted:** Ensure all components, including key management, are hosted on your servers.
- Forking **Web3Modal** or use the newer version called **Reown.com AppKit,** or use **RainbowKit** - each requiring custom key handling.
    - **Description:** Both Web3Modal and RainbowKit are popular frameworks for wallet connections. Although typically used to integrate third-party wallets, they can be forked to build an internal wallet system that manages keys directly in the app.
    - **Benefits:** Customizable and open-source, allowing you to integrate your own authentication and storage logic without reliance on a third-party wallet provider.
    - **Customization:**
        - Implement internal key generation and encryption mechanisms.
        - Integrate Supabase for user authentication and backend storage.
    - **Considerations:** Ensure the signing and key storage logic is managed within your infrastructure.
- Not suitable
    - Torus/Web3Auth: Even though they provide key management and social login integration, they introduce lock in and dependency on their platform and pricing.
    - Magic.link: A paid service that manages authentication and key storage, creating reliance on an external provider.

Please note that the above are ChatGPT-provided examples, and they **might be entirely unusable for this purpose**. **Do your own research!**

Bonus Features

- Integrate passkey support for devices that support this authentication method to enhance security and user experience.
- Address book feature, suggesting to add new recipients/senders to the address book, and ability to send money to or request money from a contact.
- QR based transaction addressing. Scan a QR code to populate the recipient. Generate a QR code fro someone else to scan.

Technical Requirements

- hackathon: Wallet App Add the solution to the **GreenPill-TO/Genero** monorepo on GitHub as a feature branch
- Store your app (page.tsx, styling and any sub-pages) under `//app/tcoin/wallet-x`, where `X` is the project name. (You can start by copying the app that's available in the same mono-repo at `//app/tcoin/wallet`)
- Name your feature branch blockhack-x
- Store any reusable components in `//shared/components`
- Use the app router (not the page router)
- Follow best practices for code organisation and documentation.

Guidelines for Success

- Ensure that your app does not require users to store or manage a seed phrase.
- Build with future scalability in mind, supporting a multi-device user experience.
- Emphasize a seamless and secure user journey, maintaining usability while safeguarding private key management.
- Ensure private keys and pin numbers are only handled client-side, never leave the device, are not stored in local storage, and are securely deleted from memory and over-written at the end of each session. Ensure proper garbage collection.
- Simpler and more streamlined is better. Don't support more than one currency. Make it look like a normal web2 bank app from a niche bank.

Footnote *Note: This project is semi-custodial regarding key storage, since the central app and its admins does not have access to the private keys. On the other hand, the project follows a centralised approach for authentication and key backup. However, the project remains decentralised at its core, as anyone could fork or develop and launch a similar front end to interact with the TCOIN ecosystem. The centralised authentication is essential to ensure the app's registration and legal compliance for use in Canada.*

We look forward to seeing creative and robust solutions that push the boundaries of user-friendly wallet design with innovative key management!

BlockHack: QR Engine

Hackathon Challenge #2

Developing a QR Code-Based Transaction Engine for TCOIN Wallet

Challenge Overview

Expand on the existing TCOIN wallet UI located at `//app/tcoin/wallet` in this repo https://github.com/GreenPill-TO/Genero to develop a fully functional QR code-based transaction engine. This engine should provide seamless payment and request capabilities while leveraging pre-existing wallet infrastructure. For this challenge, you can hardcode the private key provided for development purposes, as key management is a separate challenge.

Core Requirements

1. **Responsive, Mobile-First Design**
    - Design the wallet with mobile users as the primary focus, ensuring an excellent user experience across various screen sizes.
2. **Primitives**
    - **QR Generation:** Create a feature for generating a QR code that can be scanned by others to initiate a payment or request.
    - **QR Scanning:** Implement functionality to scan a QR code to populate recipient information for a transaction.
    - **Signup vs transact:** Users who scan a QR code or enter the link should be taken to a page where it is determined if they are already a user or not. Prior users will be taken to a deep link. New users will first be redirected to a signup flow, before being taken to the deep link.
    - **Transaction confirmation:** A key to acceptance in stores is speed and reliability of confirmation that the transaction went through. Build confirmation two ways. Build a confirmation screen on the sender's device that incorporates dynamic and animated content that includes amount and recipient, to prove that it's not a static image which can be faked. Also build confirmation on the recipient device to show that the amount and the sender, easily accessed through a notification.
3. **QR Code-Based Transaction Engine**
    - **Push Payments / Send**: Implement a traditional transaction feature that allows users to send payments to another user. But instead of pasting their wallet address, obtain it by either scanning a QR code that populates the recipient's address, or get it from the address list of past senders & receivers.
    - **Payment Links:** Build a feature that allows users to generate a payment link. The user should enter an amount, add a message, and generate a shareable link / QR code which is copied to their clipboard. They can send this link to

anyone. If the recipient is a prior user, they will receive the money when they click the link. If the recipient is not a user, then they will first be directed to the signup flow before receiving the money.
    - **Pull Payments / Request Links**: Create a feature that enables users to generate a QR code or link that others can scan or use to send them payments. There should be an unpriced link by default, and the user should also be able to generate a priced link (e.g. scan this link to send me 10 coins)
    - **Itemized Invoices**: Build support for generating itemized invoices that can be sent with a payment request. This is similar to the prized QR code, but with access to an underlying data object with quantities and itemized pricing. Invoices should have a time-to-live determined by the user, after which they are purged if unpaid. Paid invoices should remain in the system always.
    - **In-App Notifications**: Implement notifications within the app that inform users when a payment has been completed, and when an invoice has been received. Users should be able to control who in their contact list is allowed to send them invoices.
4. **Address Book Feature**
    - Create an address book within the app to store recipient and sender contact details.
    - Add logic to suggest saving new recipients or senders to the address book after a transaction.
    - Implement the ability to send money to or request money from a contact in the address book.
    - Implement the ability to set permissions per contact, controlling if they are allowed to send invoices, etc.

Bonus Features

- **Role-based settings:** Allow waitresses to set default percentages (e.g. 15% / 18% / 20%). Allow panhandlers to set custom amounts requested (e.g. $2 / $5 / $10). Allow regular users to set a standard default tip for panhandlers, and a standard default percentage for waitresses.
- **SKUs for Stores:** Build a feature for store managers to define SKUs with names and prices. Make a Point-of-Sale-like feature for them to add up these SKUs into one invoice.
- **Integrate Passkey Support**: Enhance security and user experience by integrating passkey support for devices that are compatible with this authentication method. Require a passkey signature before authorizing a transaction valued over $100 (or if the cumulative amount for the day is above $100). Allow the user to change this threshold.

Technical Requirements

- Add the solution to the **GreenPill-TO/Genero** monorepo on GitHub as a feature branch
- There is a UI shell available as a starting point at `//app/tcoin/wallet`
- Name your feature branch “blockhack-x”, where x is your team name.

- Store any reusable components in `//shared/components`
- Use the app router (not the page router)
- Follow best practices for code organization, modular design, and documentation.
- Ensure that all features are implemented with a mobile-first design approach for optimal usability on smartphones and tablets.
- Ensure data security and user privacy by following secure data handling practices.

Guidelines for Success

- Prioritize user experience with a streamlined and intuitive interface.
- Implement QR code scanning and generation with attention to reliability and speed.
- Ensure secure handling of the private key (even though hardcoded for now, plan for future security enhancements).
- Maintain consistency with the overall design and functionality of the TCOIN wallet.

We look forward to seeing innovative and user-friendly solutions that enhance the capabilities of the TCOIN wallet through this QR code-based transaction engine!

BlockHack: Indexer

Hackathon Challenge #3

Building an Indexer, Table, and Graphing System for TCOIN Wallet

Challenge Overview

Enhance the TCOIN wallet by developing an indexer system that tracks transaction history, along with features to display a detailed table of past transactions and a graph showcasing cumulative contributions. This challenge will add significant value to the user experience by providing insights into transaction data directly within the app.

Core Requirements

1. **Transaction Indexer**

    - Develop an indexer that tracks and records all user transactions within the TCOIN wallet.
    - Ensure the indexer updates in real-time or near real-time, reflecting the latest transactions as they occur.
    - The indexer should categorize transactions (e.g., sent, received, pending) for easier data management and display.

2. **Table of Past Transactions**

    - Implement a feature to display a table within the app that lists all past transactions.
    - The table should include details such as transaction ID, date and time, amount, sender/recipient, status, and transaction type.
    - Include filtering and sorting options, allowing users to view transactions by date, type, and amount.
    - Ensure the table is paginated or has infinite scrolling for better performance with large datasets.

3. **Graph of Cumulative Contributions**

    - Create a visual graph within the app that displays cumulative contributions over time.
    - Allow users to switch between different time frames (e.g., last 7 days, last month, all time) to view their contribution trends.

    - Implement an interactive graph that highlights data points on hover and provides tooltips with specific details.
    - Ensure the graph is responsive and displays well on both desktop and mobile devices.

Additional Features to Include

- **User Insights and Analytics**:
    - Display summary statistics such as total contributions, average transaction amount, and highest single transaction.
    - Provide insights into transaction patterns, like most active days or weeks.
- **Export Functionality**:
    - Implement a feature that allows users to export their transaction history as a CSV or PDF file.
- **User Preferences**:
    - Add user settings to customize how the table and graph display data (e.g., currency format, light/dark mode).

Bonus Feature

- **Real-Time Notifications**: Integrate notifications that inform users of significant transaction milestones (e.g., when cumulative contributions surpass a certain threshold).
- **Passkey Authentication Integration**: For users with compatible devices, require passkey authentication to access detailed transaction history or export data, adding an extra layer of security.

Technical Requirements

- Add the solution to the **GreenPill-TO/Genero** monorepo on GitHub as a feature branch
- There is a UI shell available as a starting point at `//app/tcoin/wallet`
- Name your feature branch “blockhack-x”, where x is your team name.
- Store any reusable components in `//shared/components`
- Use the app router (not the page router)
- Follow best practices for code organization, modular design, and documentation.

- Ensure that all features are implemented with a mobile-first design approach for optimal usability on smartphones and tablets.
- Use appropriate libraries for data visualization (e.g., Chart.js, D3.js) and ensure charts are optimized for performance.
- Ensure data security and user privacy by following secure data handling practices.

Guidelines for Success

- Prioritize creating an intuitive user experience with clear and concise data presentation.
- Make sure that the indexer is efficient and doesn't impact the overall performance of the app.
- Test the graph and table for usability, ensuring they remain interactive and informative on both desktop and mobile devices.
- Implement clear documentation on how developers can contribute to or expand the indexer and visualization features.

We look forward to your creative solutions that enhance the analytical and transactional capabilities of the TCOIN wallet. This challenge aims to provide users with greater insight into their transaction history and improve overall financial tracking.

BlockHack: UI/UX

Hackathon Challenge #4

Designing a Comprehensive TCOIN Ecosystem UI

Challenge Overview

Develop a high-fidelity graphic design and Figma model for a comprehensive TCOIN ecosystem that includes a full website redesign, a wallet app, a governance app for stewards, and specialised management applications. This challenge aims to create a visually appealing and user-friendly interface that integrates all major aspects of the TCOIN system, enhancing both user and administrative experiences.

Core Components to Develop

1. **Landing Page**

    - Design an engaging and informative landing page that clearly communicates the purpose of TCOIN, its benefits, and how users can get started.
    - Ensure the design is responsive and adapts well to both desktop and mobile devices.
    - Include clear call-to-action elements, such as “Sign Up” and “Learn More” buttons.

2. **Authentication**

    - Streamline the OTP-based authentication flow that ensures easy user registration with options for either email or phone number.

3. **Sign-Up Flow**

    - Create a streamlined sign-up and welcome process for new users
    - Options to enhance their profile with email, socials, and phone number to gain a higher “human score”
    - Option to select preferred notifications for incoming payments and invoices
    - Include user onboarding steps with a brief introduction to TCOIN and its features.

4. **Wallet App**

- Design a mobile-first wallet app that allows users to manage their TCOIN balance, view transaction history, and send/receive TCOIN.
- Integrate QR code functionality for making and receiving payments.
- Integrate functionality to create itemised invoices, and to create payment links.
- Ensure security measures such as passkey or biometric authentication for enhanced protection.

5. **Governance App for Stewards**

    - Develop a governance app (or role-specific page within the main app) where stewards can vote on key governance parameters.
    - Include a dashboard displaying current proposals, voting statuses, and the outcome of previous votes.
    - Design voting interfaces that are intuitive and accessible.

6. **Store Management App for Store Owners**

    - Create an app (or role-specific page within the main app) where store owners can manage their stores, including assigning clerks, setting SKUs, and building checkout pages.
    - Include functionality for generating QR codes that clerks can use for accepting TCOIN payments.
    - Design features for redeeming TCOIN balances and viewing transaction analytics.

7. **Governance Page for Charities**

    - Design a governance page tailored for charities to assign stewards and participate in community decision-making.
    - Ensure the interface provides an overview of assigned stewards and their responsibilities.

8. **Governance Page for City Admins**

    - Develop a governance page that allows city admins to assign charities, manage redemption requests/burns, and handle purchase requests/minting.
    - Include dashboards to monitor activity, review and approve requests, and oversee the distribution and flow of TCOIN.

Design Requirements

- Create high-fidelity designs that prioritise clarity, usability, and an engaging user experience.
- Ensure all designs are cohesive and maintain a consistent visual theme across the entire ecosystem.

- Use Figma or similar tool to build interactive prototypes for each component, demonstrating how users will navigate through the apps and pages.
- Design with scalability and future feature expansion in mind.

Bonus Features

- **Integrate Passkey Support**: Include passkey or biometric support for all apps to improve security and user experience.
- **Advanced Analytics Dashboards**: Add interactive analytics dashboards for store owners and city admins to track TCOIN usage trends and performance.

Guidelines for Success

- Focus on creating a seamless user journey that reduces friction and enhances accessibility.
- Ensure that each app and page has a logical flow and intuitive navigation.
- Test the designs with potential users to gather feedback and iterate based on their experiences.
- Follow best practices for graphic design, including alignment, spacing, and readability.

We look forward to seeing innovative and high-fidelity designs that showcase your creativity and attention to detail, contributing to the development of a comprehensive and user-friendly TCOIN ecosystem.