3. Programmable Infrastructure: The New Asset Class

For centuries, infrastructure has been treated either as a public liability financed through taxes or a long-term private investment recouped slowly over decades. It was built, maintained, and depreciated on balance sheets without generating real-time income or enabling dynamic engagement. This model has proven increasingly unsustainable in the face of rapid urban growth, climate demands, and citizen expectations for transparency and performance.

RMBT introduces a revolutionary shift by defining infrastructure as programmable, revenue-generating, and tokenized assets. These assets are governed by smart contracts and operated within decentralized economies. The result is a new asset class that is liquid, transparent, and inherently participatory.

- **Roads become smart micro-economies**: By embedding toll logic into smart contracts and connecting road segments to blockchain addresses, RMBT turns streets into monetizable networks. Each vehicle that uses a mapped road pays a small fee, which is distributed in real time to municipalities, road investors, and DAO treasuries. The toll rate can be dynamic, factoring in time of day, road condition, or congestion.
- **Energy grids evolve into decentralized, peer-audited power markets**: Solar panels, kinetic tiles, and other smart energy systems are registered on-chain. Producers are rewarded based on verified output, while consumers pay using stable tokens or RMBT. The system supports decentralized market pricing, SDG tracking, and tokenized rewards for sustainable behavior.
- **Buildings, bridges, and monuments become utility-bearing NFTs**: Public and private structures are minted as NFTs with attached metadata, maintenance requirements, staking logic, and voting rights. Ownership can be fractionalized, staked, or leased, introducing a tradable layer to previously illiquid civic assets.

This evolution creates a programmable financial layer over physical infrastructure, one that introduces modular rights and revenue streams such as:

- Usage-based tolling and dynamic pricing
- Maintenance staking with performance incentives
- SDG-linked yield bonuses for achieving impact metrics
- DAO-issued grants for upgrades, retrofitting, or expansion
- NFT-based access, governance, or leasing mechanisms

By decoupling infrastructure value from static ownership and coupling it with blockchain-based financial primitives, RMBT unlocks a global class of public-private digital infrastructure investments. It enables every street, every pipe, and every solar panel to become a micro-economy, governed and monetized in real time.

4. RMBT Toolkit Architecture

The RMBT Toolkit is a comprehensive infrastructure protocol designed for flexibility, speed, and composability. It enables developers, municipalities, investors, and engineers to deploy real-world infrastructure logic in a decentralized, monetized, and community-governed manner. This section elaborates on the core architecture across multiple dimensions, from smart contract modules to real-time data pipelines, all engineered to support global scalability and real-world interoperability.

Overview of Core Modules:

- Toll smart contracts with multi-party revenue splitting
- NFT registry for streets and infrastructure segments
- Local token generator (e.g., $CITYroad) linked to RMBT
- Staking engine with SDG-linked performance rewards
- DAO module with quadratic voting and budget disbursement
- Pyro-energy and kilowatt verification contracts
- REST/WebSocket APIs for developers

Overview of Technology Stack:

- Smart Contracts: Solidity (TVM/TRON)
- Backend APIs: Node.js, Redis, Postgres
- Dashboard SaaS Portal: React with Web3 login (TronLink)
- Data Storage: BTFS/IPFS for files, Redis for caching

4.1 Details of Core Functional Modules

Toll Smart Contracts with Multi-Party Revenue Splitting
Toll contracts allow for programmable monetization of roads, rail platforms, and utility networks. These contracts receive payments in RMBT or local tokens and automatically distribute funds across designated recipients. For example, a single micro-toll payment may be split among the municipal authority, local investors, the DAO treasury, and the infrastructure contractor. Revenue distribution logic is transparent, auditable, and enforced at the code level.

Street and Infrastructure NFT Registry
RMBT introduces a token registry system where roads, bridges, utility segments, or buildings are represented as NFTs. These NFTs contain metadata such as geographic coordinates, usage rules, traffic volume, staking APYs, and ownership

status. NFTs serve not just as digital certificates but as operational anchors for real-world usage. They can be transferred, leased, or co-owned, offering fractional access to infrastructure value.

Local Token Generator ($CITYroad Tokens)
Each city or district may generate its own governance or transactional token, such as $DLAroad (Douala), $MORroad (Moroni), or $YDEroad (Yaoundé). These local tokens are pegged to RMBT through wrapper and vault contracts, enabling localization of the economy while maintaining reserve backing. The system is designed for seamless swap, staking, and utility across both local and parent-token ecosystems.

Staking Engine with SDG-Linked Rewards
The staking engine enables RMBT holders to allocate tokens to specific infrastructure assets. Yield is generated from verified usage such as tolls paid or energy sold and is augmented by SDG-linked performance. For instance, if a road achieves a predefined foot traffic score or carbon savings threshold, the yield to stakers increases. Withdrawals can be gated by time, performance benchmarks, or governance votes.

DAO Module with Quadratic Voting and Budget Disbursement
DAO functionality is embedded into every phase of the asset lifecycle. From project approval and contractor payouts to city token issuance and smart contract upgrades, all key decisions flow through community-driven proposals. Voting utilizes quadratic weighting to balance small and large stakeholders, and budget allocation occurs via on-chain treasury contracts.

Pyro-Energy and Kilowatt Verification Contracts
These modules interface with hardware such as pressure tiles, solar panels, and energy nodes. Verified outputs are recorded on-chain using oracle submissions. Power producers earn RMBT tokens based on kilowatt generation, and institutions or municipalities may purchase energy units in bulk. The design supports a distributed, accountable green energy grid with automatic reward logic.

REST and WebSocket APIs for Developers
External systems including city apps, payment platforms, or mobility networks can interact with RMBT infrastructure using RESTful and WebSocket endpoints. API access is permissioned via key provisioning with granular rate limits and usage logs. Data streams include toll payments, infrastructure status, energy metrics, and DAO events. Developers can also trigger contract actions through POST and PATCH methods.

4.2 Details of Technology Stack

Smart Contract Layer (Solidity on TVM)
All core contracts are deployed on the TRON Virtual Machine (TVM), ensuring full compatibility with TRC-20 standards. These include the RMBT token contract, NFT logic, staking mechanisms, DAO protocols, toll processing contracts, and SDG performance logic.

Backend APIs (Node.js, Redis, Postgres)
A high-throughput backend supports real-time interaction between smart contracts and front-end applications. Redis is used for caching active metrics, while Postgres handles persistent off-chain logs, user metadata, and analytical summaries. The backend architecture is built on Node.js and optionally NestJS for modular, scalable service logic.

Dashboard SaaS Portal (React with Web3 Login)
The RMBT Toolkit includes a robust dashboard environment hosted as a Software-as-a-Service (SaaS) platform. The frontend is developed in React and supports Web3 wallet-based authentication using TronLink. Stakeholders such as municipalities, developers, and investors can view and manage roads, staking pools, earnings, governance participation, and city token deployment through an intuitive user interface.

Data Storage (BTFS/IPFS and Redis)
Critical documents, blueprints, staking terms, and asset metadata are stored on decentralized file systems such as BTFS and IPFS. Live polling and real-time analytics are handled via Redis for speed and responsiveness. Large-scale historical and SDG-tracking datasets are stored in structured relational databases.

The RMBT Toolkit is intentionally designed for extensibility. Future versions will introduce support for zk-SNARKs to enable anonymous staking, rollup-based Layer 2 scalability enhancements, and multi-chain interoperability with Ethereum Layer 2 networks, Polygon, and BNB Smart Chain. This positions RMBT not only as a toolkit, but as a fully programmable operating system for the decentralized infrastructure economy of the future.

4.3 Revenue Split (Micro-Level Transaction Flow)

The RMBT Toolkit enables modular infrastructure monetization through programmable smart contracts that automatically distribute incoming value—such as tolls, energy payments, or staking fees—across predefined stakeholder classes. This

revenue split occurs in real time, ensuring every micro-transaction is transparently accounted for, and aligned with ecosystem priorities.

Overview

Each asset onboarded to the RMBT system (e.g. roads, pyro-energy tiles, water pipes) has a smart contract attached to it that defines how revenues are split among contributors, maintainers, and stakeholders. This logic is embedded into the `splitRevenue()` function and is called upon each transaction such as `payToll()` or `recordKWh()`.

The revenue distribution is based on pre-approved parameters, which can be dynamically updated by the DAO based on asset performance, traffic volume, or SDG achievement scores.

Sample Toll Payment Distribution

Let's consider a 1 RMBT payment on a tokenized smart road:

Recipient Class	Allocation	Description
DAO Treasury (You the token holders)	50%	Used for reinvestment, grants, and network operations
City Operator (Local Government)	20%	Local authority or validator operating the infrastructure
Oracle or Metric Reporter (Neighbour & Store Owner)	15%	For providing verified usage data (e.g. traffic, pollution, footfall)
Maintenance Contractor (Local Vendors)	10%	Allocated to those tasked with physical upkeep of the infrastructure
Burn Reserve (Carbon Credits / Carbon Deeds)	5%	Routed to the burn vault for deflationary control

These values can be different for energy modules, waste collection, or water systems. For instance, a `recordKWh()` transaction for a solar tile may prioritize the energy producer over the contractor.

Smart Contract Execution

Each NFT-based asset (e.g. StreetNFT) has a metadata registry where the following is defined:

```
recipientAddresses[]
recipientWeights[]
```

The `splitRevenue()` function is triggered by payment calls and executes proportional transfers based on those arrays:

```
function splitRevenue(uint256 amount) external {
    for (uint256 i = 0; i < recipientAddresses.length; i++) {
        uint256 share = (amount * recipientWeights[i]) / 100;
        RMBT.transfer(recipientAddresses[i], share);
    }
}
```

Weights are locked during deployment and can only be updated via DAO-approved governance.

Dynamic Adjustments

Revenue splits are not static. DAO governance allows real-time rebalancing of these splits through proposals that can:

- Increase allocation to validators in high-performance zones
- Reward infrastructure outperforming its SDG targets
- Reduce shares to non-performing or inactive stakeholders
- Trigger split resets post maintenance cycles

Proposals must include:

- Justification (e.g. KPI exceeded)
- Target NFT or zone
- Proposed allocation structure
- DAO voting window

Once passed, the `updateRevenueModel()` contract function updates the underlying registry.

On-Chain Transparency

Each revenue-splitting transaction is visible on-chain and indexed in real time:

- Beneficiary address and amount
- Source contract and asset
- Timestamp and transaction ID

The RMBT Dashboard offers graphical breakdowns per asset, per recipient, and per category. This ensures that citizens, investors, city operators, and even auditors have full visibility into the network's economic flows.

Future Enhancements

To improve the flexibility and scalability of the system, future versions of the revenue engine may include:

- Time-weighted rewards (based on contract uptime or seasonal traffic)
- Oracle score multipliers (accurate reporters receive more)
- Role-based boosts (e.g. women-owned maintenance teams)
- Plugin-based fee logic (custom modules per city or SDG zone)

RMBT transforms infrastructure payments into dynamic, real-time economic flows that benefit the entire ecosystem. Through programmable revenue distribution logic, the protocol ensures every payment—no matter how small—is fairly routed, transparently executed, and evolution-ready.

5. Smart Contract & Technical Layer

At the core of RMBT lies a robust suite of smart contracts engineered for efficiency, modularity, and seamless interaction with the broader TRON ecosystem. All smart contracts are written in Solidity and deployed on the TRON Virtual Machine (TVM), ensuring compatibility with TRC-20 standards and integration with wallets, exchanges, and developer tools throughout the TRON network.

While RMBT itself operates as a TRC-20 compliant token, the broader system relies on independent but interoperable contracts that govern toll payments, infrastructure ownership, staking systems, DAO governance, and energy-based rewards. These contracts form the programmable backbone of the RMBT infrastructure layer,

separating token logic from utility layers to preserve upgradability, compliance, and platform neutrality.

5.1 TRC-20 Integration and Separation of Concerns

The RMBT token contract adheres to the TRC-20 standard and includes all essential methods such as `transfer()`, `approve()`, `transferFrom()`, and `allowance()`. This ensures RMBT functions like any other TRON asset across wallets such as TronLink, decentralized exchanges like SunSwap, and developer environments.

To maintain flexibility, RMBT's infrastructure logic is modularized. This means the RMBT token is not embedded within each application logic but instead interacts through clearly defined interfaces, such as via staking pools or toll contracts. This design pattern enables independent upgrades, secure integration with external applications, and clear separation between financial and infrastructure codebases.

5.2 Core Contracts and Their Functions

The following are the foundational smart contracts that comprise the RMBT Toolkit:

TollContract

Function: Implements metered pricing logic for tokenized roads and transportation assets.

Key Capabilities:

- Calculates toll fees based on distance, time, or dynamic demand
- Accepts payments in RMBT and local city tokens
- Splits received funds to designated parties using pre-defined allocation ratios

Typical Methods:

```
payToll(address road, uint256 meters)
setRate(address road, uint256 rate)
splitRevenue(address road)
```

This contract is optimized for high-throughput and is gas-efficient due to TRON's TVM execution model. It enables infrastructure assets to become continuous sources of micro-revenue.

StreetNFT

Function: Registers and manages tokenized infrastructure assets such as roads, bridges, or transport stations.

Key Capabilities:

- Mints unique NFTs representing physical infrastructure
- Stores metadata such as location, usage class, and governance rights
- Allows for transfer of ownership or staking rights

Typical Methods:

```
mintNFT(uint256 id, string metadataURI)
transferNFT(address to, uint256 id)
ownerOf(uint256 id)
```

Each NFT acts as an operational shell within which tolls, staking, and governance can occur. NFTs are indexable and linkable to real-world GIS systems for visual infrastructure mapping.

Example:

AMBANIroadNFT - This tokenized street segment represents Ambani Street, a major thoroughfare funded and maintained under a public-private partnership. The street's NFT contains GPS-coordinates, zoning metadata, toll logic, and DAO staking rules. Streets named after prominent families can become high-value governance zones, attracting private infrastructure investors and enabling localized economic layers governed through smart contract modules.

StakingPool

Function: Enables users to stake RMBT into specific infrastructure assets and earn proportional yield.

Key Capabilities:

- Allows staking into roads, city tokens, or pooled infrastructure funds
- Tracks user shares, yield earned, and SDG performance multipliers
- Governs withdrawal rules based on contract milestones or DAO votes

Typical Methods:

```
stake(uint256 amount, uint256 assetId)
claimYield(uint256 assetId)
unstake(uint256 amount, uint256 assetId)
```

This contract dynamically calculates rewards based on usage and SDG metrics submitted by oracles. Withdrawals may be limited during DAO-initiated lock periods or maintenance windows.

CityTokenMegaFactory

Function: Deploys local governance or payment tokens pegged to RMBT.
Regulation: Only assisting city launch tokens.

Key Capabilities:

- Creates ERC-20 compliant city tokens with pegged backing (e.g., $KHIroad)
- Configures tokenomics, backing ratio, mint limits, and DAO parameters
- Integrates these tokens into toll, staking, and governance contracts

Typical Methods:

```
createCityToken(string name, string symbol, uint256 ratio)
linkToInfrastructure(uint256 tokenId, address nftAddress)
swapWithRMBT(address user, uint256 amount)
```

This contract facilitates localization of infrastructure economies, enabling smoother UX for citizens while maintaining reserve control.

TreasuryDAO

Function: Controls budget approvals, fund disbursement, and infrastructure grants through community voting.

Key Capabilities:

- Allows proposals for contract upgrades, grants, and operational spending
- Implements quadratic voting to ensure equitable decision-making
- Allocates funds from DAO reserves based on passed proposals

Typical Methods:

```
propose(uint8 type, uint256 amount, bytes calldata data)
vote(uint256 proposalId, uint256 weight)
execute(uint256 proposalId)
```

The TreasuryDAO contract is also responsible for burn votes, staking incentives, and strategic expansion decisions such as launching new city tokens.

EnergyYield

Function: Tracks verified kilowatt-hour output from pyro-energy systems and solar panels and distributes RMBT rewards accordingly.

Key Capabilities:

- Records energy production submitted by registered oracles
- Assigns energy credits to producers and distributes RMBT in return
- Supports institutional buyers to pre-purchase or subsidize clean energy

Typical Methods:

```
recordKWh(address producer, uint256 amount)
disburseReward(address producer)
setOracle(address validator)
```

This contract underpins the green energy economy of RMBT-powered cities, linking smart infrastructure directly to token-based incentive flows.

5.3 Technical Design Principles

- **Security-first:** All contracts are audited and follow strict role-based permissions using OpenZeppelin's `AccessControl.sol`
- **Upgradeable architecture:** Implemented using `TransparentUpgradeableProxy` patterns to allow modular upgrades
- **Modular logic layers:** Each functional contract operates independently, allowing plug-and-play deployment and future composability with third-party DApps
- **Event indexing:** All contracts emit standardized events for off-chain indexing and API integration

This smart contract layer serves as the programmable backbone of the RMBT ecosystem. It provides a flexible, battle-tested foundation on which cities, companies, and citizens can build decentralized infrastructure economies at any scale.

6. Role-Based Access Control (RBAC)

To ensure operational security and controlled delegation of infrastructure responsibilities, RMBT employs a Role-Based Access Control (RBAC) model across all smart contracts. This structure enables decentralized yet secure participation from multiple stakeholders, including municipalities, developers, investors, contractors, and automated sensors.

RBAC is enforced through OpenZeppelin's `AccessControl` library, a widely adopted Solidity standard that allows dynamic assignment, revocation, and verification of roles. All permissions within the protocol are linked to specific smart contract roles that map to real-world governance and operational responsibilities.

The following roles are currently defined within the RMBT protocol:

ADMIN_ROLE

Permissions:

- Global configuration of the ecosystem
- Deployment and upgrading of modules
- Assignment and revocation of roles
- Oversight of integration and linking contracts

Examples:

- Deploying a new staking pool
- Assigning `CITY_OPERATOR` rights to a municipality
- Modifying SDG weighting parameters

This role is reserved for DAO-elected multisig accounts or system administrators during pilot phases.

CITY_OPERATOR

Permissions:

- Launch toll contracts for approved roads
- Mint and manage StreetNFTs
- Configure and maintain staking pools
- Update infrastructure metadata

Examples:

- Registering a new toll-enabled street
- Adjusting toll rates on high-traffic days
- Reassigning staking pools after road expansion

`CITY_OPERATOR` is typically assigned to municipal agencies or contracted urban infrastructure platforms.

INVESTOR

Permissions:

- Stake RMBT into approved infrastructure assets
- Claim yield based on verified asset performance
- Participate in DAO voting and governance proposals

Examples:

- Staking RMBT into a smart bridge in São Paulo
- Earning yield bonuses after energy SDG goals are met
- Voting on proposals for token burns or grant disbursements

`INVESTOR` is the most widely accessible role and forms the foundation of RMBT's citizen-stakeholder model.

CONTRACTOR

Permissions:

- Receive milestone-based payouts in USDT or RMBT
- Submit proof-of-completion for infrastructure projects
- Interact with DAO task tracking and funding modules

Examples:

- Triggering payment after completing a toll booth
- Uploading blueprints or sensor data as audit evidence
- Receiving bonuses for ahead-of-schedule delivery

`CONTRACTOR` accounts are time-bound and limited to the scope of verified smart Build-Operate-Transfer (BOT) agreements.

ORACLE

Permissions:

- Submit off-chain metrics related to SDG performance
- Validate energy output, foot traffic, or environmental data
- Influence contract logic through verified real-world events

Examples:

- Sending live air quality readings for SDG 13 metrics
- Verifying solar output from a decentralized energy grid
- Submitting foot traffic for walk-to-earn modules

Oracles must be pre-approved and may include IoT devices, civic institutions, or third-party validators. Their data is cryptographically signed and submitted on-chain for transparency and trust.

On-Chain Role Enforcement

Access to role-restricted functions is programmatically enforced in Solidity using statements such as:

```
require(hasRole(CITY_OPERATOR, msg.sender), "Not authorized");
```

Role assignments and removals are managed using:

```
grantRole(bytes32 role, address account)
revokeRole(bytes32 role, address account)
```

Administrators and DAO proposals may invoke these methods to ensure responsive, adaptive governance.

RBAC is a critical safeguard that ensures the RMBT protocol remains secure, scalable, and compliant with institutional-grade role management, without compromising on decentralization or modularity.