Technical Specifications

Note: This technical specification document contains placeholders for diagrams and certain technical details that will be refined as development progresses. Contributors interested in helping with these areas are encouraged to open an issue or discussion. This file merges v0.2 with all approved copy-edits to complete the migration from Chrysalis/Tangle to the Rebased object-DAG.

A.1. System Architecture Overview

A.1.1. High-Level Architecture

The OmeoneChain platform is organized in five layers:

Layer	Purpose	Key Technology
Base	Settlement & metadata	**IOTA Rebased object-DAG (MoveVM)** -- low-fee, DAG-based ledger with Mysticeti DPoS & fee-burn mechanism
Storage	Persistent content	Hybrid on-chain/off-chain model: IPFS + optional Aleph.im pin-service
Protocol	Core business logic	Recommendations, token rewards, reputation, governance
API / Adapter	Chain abstraction	RebasedAdapter, EVMAdapter, MockAdapter -- provide identical trait: (submitTx, queryState, watchEvents)
Application	End-user & dApp UX	Web, mobile, Move-script entry points, third-party dApps

![System Architecture Diagram] **[PLACEHOLDER: Architecture diagram showing component interactions]**

A.1.2. IOTA Rebased Integration

- **Network Type:** IOTA Rebased test-net → main-net
- **Node Roles:** iotacore **validators** / **full-nodes**; optional legacy-snapshot read-only pool for archival Chrysalis data.
- **Fee Model:** Micro-fee + burn; dApps may use a **sponsor wallet** (gas-station) so end-users experience a feeless interface.

- **MoveVM:** Level-1 Move smart-contracts; on-chain objects reference IPFS CIDs and governance hashes.

A.1.3. Modular Design

- **Blockchain Abstraction Layer (BAL)**

Adapter	Language	RPC	Status
RebasedAdapter	Move	gRPC / WS	primary
EVMAdapter	Solidity	JSON-RPC	parity build
MockAdapter	SQLite	local	unit tests

Other modular facets:

- **Storage Abstraction:** Filecoin/Arweave/S3 pluggable behind common interface.
- **Pluggable Consensus:** protocol layer is chain-agnostic.

A.2. Data Structures and Models (delta-friendly)

Below are the canonical objects; only updated fields are shown---unchanged fields remain as in v0.1.

A.2.1. Recommendation Data Structure

Each recommendation in the system consists of:

```
{
 "id": "string", // Unique identifier
 "author": "string", // User's public key or identifier
 "timestamp": "ISO8601 string",
 "serviceId": "string", // Identifier of the service being recommended
 "category": "string", // Service category (e.g., "restaurant", "hotel")
 "location": {
  "latitude": "float",
  "longitude": "float",
  "address": "string" // Optional human-readable address
```

```
 },
 "rating": "integer", // 1-5 scale
 "content": {
  "title": "string",
  "body": "string",
  "media": [
   {
    "type": "string", // "image", "video", etc.
    "ipfsHash": "string", // Reference to stored media
    "caption": "string" // Optional
   }
  ]
 },
 "tags": ["string"], // Array of relevant tags
 "verificationStatus": "string", // "verified", "unverified"
 "contentHash": "string", // Hash of the content for verification
 "tangle": {
  "objectId": "string", // on-chain object ID
  "commitNumber": "integer" // ledger commit height
 },
 "chainID": "string" // Indicates originating chain
}
```

A.2.2. User Reputation Model

The reputation system tracks user contributions and overall trustworthiness:

```
{
 "chainID": "string", // Rebased-testnet-001
```

```
  "userId": "string", // Public key or identifier
  "totalRecommendations": "integer",
  "upvotesReceived": "integer",
  "downvotesReceived": "integer",
  "reputationScore": "float", // Calculated score
  "verificationLevel": "string", // "basic", "verified", "expert"
  "specializations": ["string"], // Categories of expertise
  "activeSince": "ISO8601 string",
  "tokenRewardsEarned": "integer",
  "followers": "integer",
  "following": "integer",
  "ledger": { "objectID": "string", "commitNumber": "integer" }
}
```

A.2.3. Token Transaction Model

Token transfers and rewards are structured as:

```
{
  "transactionId": "string", // Unique identifier
  "sender": "string", // Public key or "SYSTEM" for rewards
  "recipient": "string", // Public key
  "amount": "integer",
  "timestamp": "ISO8601 string",
  "type": "string", // "reward", "transfer", "fee", etc.
  "actionReference": "string", // Related recommendation/action ID
  "tangle": {
    "objectId": "string", // Rebased object ID
    "commitNumber": "integer" // Reference commit number
```

```
 }
}
```

A.2.4. Service Entity Model

Services that are recommended are structured as:

```
{
 "serviceId": "string", // Unique identifier
 "name": "string",
 "category": "string",
 "subcategories": ["string"],
 "location": {
  "latitude": "float",
  "longitude": "float",
  "address": "string",
  "city": "string",
  "country": "string"
 },
 "website": "string", // Optional
 "contact": "string", // Optional
 "verificationStatus": "string", // "claimed", "verified", "unclaimed"
 "averageRating": "float",
 "totalRecommendations": "integer",
 "totalUpvotes": "integer",
 "createdAt": "ISO8601 string",
 "updatedAt": "ISO8601 string"
}
```

A.3. Core System Components

A.3.1. Recommendation Engine

The recommendation engine is responsible for processing, storing, and retrieving user recommendations:

1. **Submission Processing**:
    - Content validation (spam detection, content policy checking)
    - Metadata extraction and categorization
    - IPFS storage of full content
    - Rebased DAG transaction creation for metadata and verification
2. **Retrieval and Search**:
    - Multi-faceted search capabilities (location, category, rating)
    - Personalization based on user preferences and history
    - Ranking algorithms based on reputation, recency, and quality metrics
3. **Voting and Engagement**:
    - Upvote/downvote processing
    - Comment and discussion functionality
    - Signal processing for quality assessment

[PLACEHOLDER: Specific algorithms for recommendation ranking and personalization]

A.3.2. Token Reward System

The token system manages incentives for platform participation:

1. **Reward Calculation Logic**:
    - Creating a recommendation: Tokens mint only after the tip's Trust Score reaches 0.25 (Formula: Reward = 1 TOK × Σ Trust-weights (cap 3 ×).
    - Trusted Up-Votes
    - Reputation factors for reward adjustment
    - Halving implementation based on distribution milestones
2. **Distribution Mechanisms**:

- Automatic reward issuance for qualifying actions
- Periodic batch processing for optimized transaction volume
- Special event rewards (leaderboards, challenges)

3. **Token Utility Functions**:
   - NFT purchase integration
   - Sponsor-wallet flow: On fee-based DAGs, the sponsor wallet prepays baseFeeμIOTA; the reward contract refunds in batched settlement.
   - Reward formula: baseReward × qualityMultiplier × reputationFactor -- baseFeeμIOTA
   - Governance staking
   - Service provider revenue share processing

[PLACEHOLDER: Specific mathematical formulas for reward calculations and anti-spam measures]

A.3.3. Reputation System

The reputation system assigns trust levels to users based on history and contribution quality:

1. **Score Calculation Factors**:
   - Recommendation quality (measured by upvotes)
   - Consistency of contributions
   - Verification status
   - Longevity on platform
   - Diversity of contributions
2. **Trust Mechanisms**:
   - Sybil resistance through progressive reputation building
   - Specialized expertise recognition in specific categories
   - Verification tiers with escalating requirements
3. **Governance Integration**:

- Reputation-weighted voting influence
- Special proposal privileges for highly reputed users

[PLACEHOLDER: Detailed reputation scoring algorithms and Sybil resistance mechanisms]

A.3.4. Hybrid Storage System

The platform implements a two-tiered storage approach:

1. **On-Chain Storage (Base Ledger -- Rebased DAG)**:
    - Transaction metadata
    - Verification hashes
    - User reputation updates
    - Token transfers
    - Governance actions
2. **Off-Chain Storage: IPFS (+ optional Aleph.im auto-pin).**
    - Full recommendation content
    - Media attachments (images, videos)
    - Historical data archives
    - Service details and extended information
    - Funding: storage_pool = 2 % of Service-Provider share for pin-cost governance.
3. **Data Synchronization**:
    - Content addressing via IPFS hashes stored on the base ledger
    - Verification mechanisms to ensure data consistency
    - Redundancy protocols for critical data

[PLACEHOLDER: Specific IPFS configuration details and pinning strategy]

A.4. API Specifications

A.4.1. Core API Endpoints

The platform exposes the following primary API endpoints:

Recommendation Management

GET /api/v1/recommendations # List recommendations with filtering

POST /api/v1/recommendations # Create recommendation

GET /api/v1/recommendations/{id} # Get single recommendation

PUT /api/v1/recommendations/{id} # Update recommendation (author only)

DELETE /api/v1/recommendations/{id} # Mark recommendation as deleted

POST /api/v1/recommendations/{id}/upvote # Upvote recommendation

POST /api/v1/recommendations/{id}/downvote # Downvote recommendation

User and Account Management

POST /api/v1/users # Create user account

GET /api/v1/users/{id} # Get user profile

PUT /api/v1/users/{id} # Update user profile

GET /api/v1/users/{id}/recommendations # Get user's recommendations

GET /api/v1/users/{id}/reputation # Get reputation details

POST /api/v1/users/{id}/follow # Follow user

POST /api/v1/users/{id}/unfollow # Unfollow user

Token and Wallet Operations

GET /api/v1/wallet # Get wallet balance

GET /api/v1/wallet/transactions # List transactions

POST /api/v1/wallet/transfer # Transfer tokens

GET /api/v1/rewards # Get reward statistics

Service and Discovery

GET /api/v1/services # List services with filtering

GET /api/v1/services/{id} # Get service details

POST /api/v1/services # Register new service

GET /api/v1/services/{id}/recommendations # Get recommendations for service

NFT and Experience Management

GET /api/v1/nfts # List available NFTs

POST /api/v1/nfts # Create NFT (for service providers)

GET /api/v1/nfts/{id} # Get NFT details

POST /api/v1/nfts/{id}/purchase # Purchase NFT

Move-script entrypoints (gas-optimized)

- post_recommendation(author, cid)
- vote(recommendationId, up)
- claim_reward(txId)

A.4.2. Developer API for dApps

The platform provides additional endpoints for third-party developers:

GET /api/v1/developer/recommendation-stats # Get aggregated recommendation statistics

GET /api/v1/developer/trending # Get trending recommendations/services

POST /api/v1/developer/webhook # Register webhook for events

GET /api/v1/developer/categories # Get category taxonomy

[PLACEHOLDER: Detailed API schema with request/response formats and authentication requirements]

A.4.3. Integration Patterns

For third-party integrations, the platform supports:

1. **REST API Access**: Standard authenticated REST endpoints for web and mobile applications
2. **Event Streams**: Webhook notifications for real-time updates
3. **Ledger Direct Access (Rebased DAG object streaming)**: For advanced applications with direct Rebased interaction needs
4. **IPFS Gateway**: Direct content access for distributed applications

A.5. Security Specifications

A.5.1. Authentication and Authorization

1. **User Authentication Methods**:
    - Ed5519/Ed448 seed-based cryptographic authentication (ledger compatible)
    - Optional email/password with strong encryption
    - Multi-factor authentication for sensitive operations
    - Social authentication providers (optional, for mainstream adoption)
2. **Authorization Model**:
    - Role-based access control for administrators and moderators
    - Capability-based security for specific platform operations
    - Ownership-based permissions for user content

A.5.2. Data Protection

1. **Encryption Standards**:
    - Data at rest: AES-256 encryption
    - Data in transit: TLS 1.3
    - End-to-end encryption for private messages
2. **Privacy Measures**:
    - Pseudonymous identity options for users
    - Configurable privacy settings
    - Compliance with GDPR and similar regulations
    - Data minimization principles

A.5.3. Smart Contract Security

For token operations and governance functions:

1. **Formal Verification**: Critical smart contracts undergo formal verification
2. **Audit Requirements**: Third-party security audits before mainnet deployment
3. **Upgrade Mechanisms**: Secure patterns for contract upgradeability
4. **Rate Limiting**: Protection against transaction spam and DoS attacks

[PLACEHOLDER: Specific security audit procedures and formal verification methods]

A.6. Infrastructure Requirements

A.6.1. Node Infrastructure

The minimum infrastructure requirements for running platform nodes:

1. **IOTA Nodes**:
    - *iotacore* validadator / full-node
    - Minimum 24 CPU cores, 128 GB RAM, 1 TB NVMe (production)
    - High-bandwidth network connection (100+ Mbps)
    - Geographic distribution for network resilience
2. **IPFS Nodes**:
    - Kubo implementation with pinning services
    - Minimum 4 CPU cores, 8GB RAM, 2TB storage
    - Content replication across multiple nodes
    - Dedicated gateway services for high-availability content access

[PLACEHOLDER: Detailed node setup and configuration instructions]

A.6.2. Scaling Considerations

The infrastructure is designed to scale with adoption:

1. **Horizontal Scaling**: Adding nodes to handle increased load
2. **Sharding Strategy**: Future implementation of data sharding for performance
3. **Caching Layer**: Distributed caching for frequently accessed content
4. **Load Balancing**: Geographic distribution based on user concentration

A.7. Implementation Roadmap

A.7.1. Development Phases

The technical implementation will follow this phased approach:

Phase 1: Core Protocol Development (Q1-Q2 2025)

- RebasedAdapter integration and basic transaction handling
- Core data structures and validation logic

- Basic recommendation submission and retrieval

Phase 2: MVP and Testing (Q3-Q4 2025)

- Token reward system implementation
- Reputation system core functionality
- Basic web interface for testing
- Private testnet deployment

Phase 3: Beta Release (Q1-Q2 2026)

- Complete API implementation
- Mobile app development
- Security audits and optimizations
- Public testnet launch

Phase 4: Mainnet and Ecosystem (Q3 2026+)

- Production deployment on Rebased mainnet
- Developer tools and documentation
- Third-party integration support
- dApp ecosystem development

A.7.2. Testing Strategy

The platform will undergo multiple testing phases:

1. **Unit Testing**: All components with >90% code coverage
2. **Integration Testing**: Cross-component functionality validation
3. **Security Testing**: Penetration testing and vulnerability assessment
4. **Performance Testing**: Load testing under various usage scenarios
5. **User Testing**: Controlled beta testing with real users

[PLACEHOLDER: Specific testing frameworks and methodologies to be used]

A.8. Governance Technical Implementation

A.8.1. On-Chain Governance Mechanisms

1. **Proposal System**:
    - JSON schema for governance proposals
    - Validation rules for proposal submission
    - Lifecycle management (draft, active, voting, implemented)
2. **Voting System**:
    - Reputation-weighted voting mechanism
    - Token-based staking for proposal creation
    - Vote counting and threshold determination
    - Result finalization and execution
3. **Execution Mechanisms**:
    - Automated parameter changes based on approved proposals
    - Manual implementation process for complex changes
    - Transparency logging for all governance actions

[PLACEHOLDER: Detailed governance proposal structure and voting algorithms]

A.9. Third-Party Development Guidelines

A.9.1. dApp Integration

Developers building on the platform should adhere to these integration patterns:

1. **Authentication**: OAuth 2.0 flow for user authorization
2. **Data Access**: REST API with appropriate rate limiting
3. **Event Handling**: Webhook registration for real-time updates
4. **Token Integration**: Standard token transfer mechanisms

A.9.2. Extension Points

The platform provides specific extension points for developers:

1. **Custom Recommendation Filters**: Pluggable algorithms for personalization
2. **Visualization Widgets**: Embeddable components for recommendations
3. **Reputation Systems**: Extended reputation metrics and badges

4. **Specialized dApps**: Vertical-specific recommendation applications

[PLACEHOLDER: SDK documentation and example dApp implementations]

A.10. Compliance and Standards

A.10.1. Regulatory Considerations

The technical implementation will accommodate:

1. **KYC/AML**: Optional integration points for third-party KYC providers
2. **Data Protection**: GDPR, CCPA, and LGPD compliance mechanisms
3. **Content Moderation**: Community-based flagging and review system

A.10.2. Technical Standards

The platform will adhere to relevant technical standards:

1. **W3C Standards**: For web interfaces and accessibility
2. **Rebased object-DAG Standards**: IOTA Rebased specification set
3. **IPFS Standards**: For distributed content storage
4. **JSON Schema**: For API contracts and data validation
5. **OAuth 2.0**: For authentication flows

[PLACEHOLDER: Specific compliance testing procedures]