# WeSolar: Detailed Technical Development Specifications

**Table of Contents:**

# 1. User Interface (UI) Layer

### 1.1 Web and Mobile Apps

- **Technology Stack:**
  - **Frontend Framework:** React.js (for web), React Native (for mobile)
  - **State Management:** Redux for managing application state across different components.
  - **UI/UX Design:** Use Material-UI for React to ensure a consistent, modern design across all devices.
  - **Responsive Design:** Ensure compatibility with various screen sizes, from desktop to mobile.
  - **API Integration:** Axios or Fetch API for asynchronous communication with backend services.
- **Features:**
  - **Dashboard:** Real-time data visualization for solar energy production, consumption, and financial transactions.
  - **Profile Management:** User profile setup, including property details, lender information, and installer qualifications.
  - **Bidding Interface:** Interactive platform where users can post projects, receive bids, and select providers.
  - **Rating System:** Users can rate each other post-project, affecting future bid placements.

### 1.2 Dashboard

- **Real-time Monitoring:** Integration with WebSocket or similar technologies to push updates to the dashboard in real-time.
- **Data Visualization:** D3.js or Chart.js for dynamic data visualizations, including graphs for energy production and financial summaries.
- **Custom Widgets:** Users can add/remove widgets to track specific metrics relevant to their interests.

# 2. Application Layer

### 2.1 Smart Contracts Module

- **Language:** Solidity for VeChainThor, tailored to support VeChain's specific features like fee delegation and multi-task transaction (MTT).
- **Contract Types:**
  - **Property Listing Contract:** Handles property details, available roof specs, and user-defined requirements.
  - **Bidding Contract:** Manages the bidding process, including bid submission, selection, and contract finalization.
  - **Installation Contract:** Governs the relationship between the property owner and the installer, including milestones, payments, and quality assurance.
  - **Lending Contract:** Defines terms between lenders and property owners, including repayment schedules, interest rates, and collateral management.
  - **Energy Trading Contract:** Facilitates the sale of excess energy, either back to the grid or to peers within the WeSolar network.
- **Deployment and Testing:**
  - **Truffle Suite:** For smart contract development, deployment, and testing.
  - **Ganache:** For creating a local blockchain environment to test smart contracts.
  - **MythX or ConsenSys Diligence:** Tools for security audits of smart contracts.

### 2.2 AI Algorithms

#### Contract Management AI

- **Model Type:** Machine Learning model, possibly a Reinforcement Learning model for dynamic contract adjustments.
- **Data Inputs:** Historical contract data, success rates, default rates, user ratings, and real-time bidding metrics.
- **Training Dataset:** Use synthetic and real-world data from similar platforms (e.g., lending platforms) for model training.
- **Integration:** Python-based API using TensorFlow or PyTorch, integrated with the backend to trigger smart contract adjustments.

#### Grant Management AI

- **Model Type:** Natural Language Processing (NLP) model for scanning and drafting proposals.
- **APIs:** Integration with public and private grant databases through RESTful APIs.
- **Automation Framework:** Use Selenium or similar tools for automating the grant submission process.
- **Proposal Writing:** GPT-based model fine-tuned for grant writing to customize proposals based on project specifics.

**Energy Management AI**

- **Model Type:** Time Series Analysis for predicting energy production and consumption.
- **Data Inputs:** Real-time data from smart solar meters, weather conditions, historical energy production data.
- **Tokenization Integration:** Links directly to the tokenization module, triggering WeSol token creation and distribution based on energy metrics.

**2.3 Tokenization and DeFi Module**

**WeSol Tokens**

- **Token Standard:** VeChain's VIP180 token standard for fungible tokens.
- **Tokenomics:** Define rules for token minting (based on energy production), distribution (based on proof of participation and proof of production), and burning mechanisms.
- **Smart Contract Logic:** Create and manage the lifecycle of WeSol tokens, including issuance, transfers, and redemptions.

**Distribution Algorithm**

- **Consensus Mechanisms:**
    - **Proof of Participation (PoP):** Rewards users based on active participation in the network, such as energy production, investments, and installations.
    - **Proof of Production (PoProd):** Rewards based on the amount of energy generated and contributed to the grid or network.
    - **Proof of Consumption (PoC):** Incentivizes users to consume energy in an eco-friendly way, rewarding efficient usage.

**Wallet Integration**

- **Technology Stack:**
    - **Backend:** Node.js for server-side processing of wallet functions.
    - **Blockchain Interaction:** VeChain's Thorify library for interacting with the VeChainThor blockchain.
    - **Wallet Security:** Use of hardware security modules (HSMs) for secure key management.
    - **Virtual Debit Card Integration:** Partner with a service like Visa's crypto debit cards to enable spending WeSol tokens directly.

# 3. Blockchain Layer

### 3.1 VeChainThor Network

#### Smart Contracts Execution

- **MTT (Multi-Task Transaction):** Utilize VeChain's unique MTT feature to batch multiple operations within a single transaction, reducing transaction costs and improving efficiency.
- **Transaction Confirmation:** Integrate VeChainThor's Proof of Authority (PoA) consensus mechanism for fast and reliable transaction confirmations.

#### Tokenization Engine

- **Custom Tokens:** Develop and manage custom WeSol tokens through VeChain's VIP180 standard.
- **Token Issuance:** Create smart contracts that automatically issue tokens based on predefined criteria (e.g., energy production metrics).
- **Immutable Records:** Store all transactions and energy production data on the blockchain for transparency and traceability.

#### Data Recording

- **On-Chain Data:** All financial transactions, token distributions, and energy metrics will be recorded on-chain.
- **Data Query:** Use VeChain's Thor APIs to query data for reporting and real-time dashboards.

### 3.2 VeBetterDAO Integration

#### Governance

- **DAO Contracts:** Develop DAO smart contracts to manage voting and dispute resolution processes.
- **B3TR Token Integration:** Implement staking and voting mechanisms using the B3TR token, ensuring that stakeholders have a say in platform governance.
- **Dispute Resolution:** Automate minor disputes with predefined logic, while escalating major issues to a DAO vote.

#### Staking and Rewards

- **Staking Mechanism:** Enable users to stake B3TR tokens in return for governance privileges or additional rewards.
- **Rewards Distribution:** Smart contracts that automatically distribute rewards based on participation metrics, funded through transaction fees or other revenue streams.

### 3.3 B3TR Token Integration

- **Rewards Program:** Integrate B3TR tokens into WeSolar's rewards system, allowing users to earn and stake tokens for additional benefits.
- **Staking Mechanism:** Allow users to stake B3TR tokens to participate in platform governance, with the possibility of earning additional WeSol tokens as rewards.
- **Voting Rights:** Use B3TR tokens to grant voting rights on key platform decisions, particularly in dispute resolutions or policy changes.

# 4. Storage Layer

### 4.1 On-chain Data

- **Transaction Data:** Store all contract-related transactions, token transfers, and energy production metrics on the VeChainThor blockchain.
- **Contract States:** Persist smart contract states on-chain to ensure transparency and immutability.
- **Security:** Utilize VeChain's inherent security features to protect on-chain data against tampering and unauthorized access.

### 4.2 Off-chain Storage

- **User Data:** Store user profiles, property details, and non-sensitive data off-chain in a secure cloud storage solution.
- **Database:** Use a scalable NoSQL database like MongoDB for storing large datasets related to user interactions, project history, and AI models.
- **AI Models:** Store trained AI models off-chain, with periodic updates to the models based on real-time data.
- **Data Encryption:** Encrypt all off-chain data using AES-256 to ensure privacy and security.

# 5. Security Layer

### 5.1 Smart Contract Security

- **Multi-signature Wallets:** Implement multi-signature wallets for critical transactions to add an extra layer of security.
- **Audit:** Regular smart contract audits using tools like MythX, OpenZeppelin, and third-party auditors to identify and fix vulnerabilities.
- **Bug Bounty Program:** Establish a bug bounty program to encourage external security experts to identify and report vulnerabilities.

### 5.2 User Authentication

- **OAuth 2.0:** Implement OAuth 2.0 for secure user authentication across web and mobile platforms.
- **Biometric Security:** Integrate fingerprint or facial recognition for mobile app users to enhance security.
- **Two-Factor Authentication (2FA):** Offer optional 2FA to provide additional security for user accounts.

### 5.3 Data Encryption

- **On-Chain Encryption:** Leverage VeChain's built-in encryption mechanisms for securing on-chain data.
- **Off-Chain Encryption:** Encrypt all off-chain user data and sensitive information with AES-256 encryption.
- **Secure Key Management:** Use HSMs and secure enclaves for managing cryptographic keys.

# 6. Integration Layer

### 6.1 External APIs

- **Grant Databases:** Integrate with grant databases like Grants.gov or private funding APIs to pull relevant opportunities.
- **Energy Grid Integration:** Use APIs provided by local energy companies to manage the buying/selling of energy from the grid.
- **Payment Gateways:** Integrate with payment processors to facilitate fiat-to-crypto conversions and vice versa.

### 6.2 Interoperability Framework

- **Cross-chain Compatibility:** Develop bridges or APIs to interact with other blockchains if WeSolar expands beyond VeChainThor.
- **Oracle Integration:** Use oracles like Chainlink to fetch external data (e.g., energy prices, weather conditions) necessary for AI algorithms and smart contracts.

# 7. Deployment Strategy

### 7.1 Development Phases

- **Phase 1: Core Development**
  - Build the core platform, including the smart contract module, bidding system, and basic tokenization features.
  - Initial development of the UI layer, focusing on user profiles, project listings, and bidding interface.
- **Phase 2: AI and Advanced Features**
  - Develop and integrate AI algorithms for contract management and grant proposal automation.
  - Complete the tokenization module and integrate it with the smart contract system.
- **Phase 3: DeFi Wallet and Finalization**
  - Build and integrate the virtual DeFi wallet, ensuring it supports both WeSol and B3TR tokens.
  - Conduct comprehensive testing and security audits across all components.
- **Phase 4: Beta Testing and Launch**
  - Deploy the platform on VeChainThor's mainnet.
  - Run a beta testing phase with a select group of users to gather feedback and make final adjustments.
  - Launch the WeSolar platform to the public.

## Conclusion

This detailed technical specification outlines the architecture and development plan for WeSolar, ensuring that all components are well-defined and ready for implementation. The next steps involve the actual coding and development, starting with the core platform and progressing through the various phases until full deployment.

Presented by:

Cornelius Lawrence DeFalco

Co-Founder, WeSolar

cornelius@wesolar.io

+1.804.404.5059