$KIN WHITE PAPER

The $KIN Token Ecosystem White Paper

Introduction

The $KIN Token is a Real World Asset (RWA) that bridges digital assets linked to physical activities in the beauty industry. It integrates real-world elements of the Beauty Industry into an innovative digital ecosystem, offering holders a tangible connection to physical beauty products. The $KIN Token ecosystem is designed to address critical problems in the beauty industry by leveraging AI and blockchain technology to provide solutions.

The objective is to synergize AI, blockchain technology, and tangible beauty products with the $KIN Token.

As a holder of the $KIN Token, you are not just keeping pace with blockchain and crypto trends; you are acquiring a digital asset that represents a tangible connection to real-world activities in the beauty industry.

This unique token provides considerable value by integrating real elements of the beauty world into an innovative digital asset.

In the near future, the physical operations with the $KIN will be utilized by HAUM, a reputable skincare brand established in 2018 in Indonesia. HAUM will leverage the $KIN Token to provide exclusive rewards to its loyal customers, thereby fostering a unique relationship between the digital asset and the physical business in the beauty industry, as well opportunity for other brands.

Led by the same creator as the $KIN Token, HAUM has laid a strong foundation for future growth.

Addressing Key Issues in the Beauty Industry with AI

The $KIN Token ecosystem is designed to address critical problems in the beauty industry using AI technology. Here's how AI & our project can solve these issues for both brands and consumers:

WEB3 DEVELOPMENT	PROJECTS
DAPP	→ $KIN Wallet : A secure wallet for storing and facilitating $KIN Token transactions within the ecosystem. → REWARDS : A rewards program where customers earn $KIN Tokens through activities like purchasing products, engaging with the brand, and participating in community events. → Web3 Ecommerce : An e-commerce platform leveraging blockchain technology for transparent and secure transactions, integrated with AI-based recommendations for a seamless shopping experience. → $KIN ReviewHub : A platform for user reviews, brand engagement, and product ranking based on feedback, with verified reviews and dynamic ranking → $KIN TALK : Direct Chat Consultation Platform with Experts: $kinTalk allows users to chat directly with dermatologists, skincare specialists, and lab professionals
DEFI	→ $KIN GrowthFund : An investment platform that provides funding for beauty industry brands, with community-driven selection and performance tracking. → Staking : Users can stake their $KIN Tokens to earn rewards or participate in governance. Staking provides liquidity and stability to the ecosystem.
DAO	→ For $kin Ecosystem sustainable. → Everyone can be part of next project decisions.
AI	→ $KIN AI : Is an advanced artificial intelligence system specifically designed for the beauty industry. $KIN AI continuously evolves and learns over time, improving its accuracy and effectiveness with each interaction (Project 24/7 CS, Food & Drug Regulatory Platform, $KIN Market Monitoring).

PROJECT 00

Project	Description	Problem	Solution	Benefit
Wallet	A secure digital wallet for storing and managing $KIN tokens.	Users need a safe and efficient way to store and manage their $KIN tokens.	Provide a user-friendly, secure digital wallet integrated into the DAPP.	Secure storage, easy management of $KIN tokens, seamless transactions.
Rewards	A system to reward users for various activities within the $KIN ecosystem.	Users need incentives to engage with the platform and contribute valuable data and feedback.	Implement a rewards system that issues $KIN tokens for activities like reviews, feedback, and usage.	Increased user engagement, valuable data collection, enhanced community involvement.
Staking	A mechanism for users to stake their $KIN tokens to earn rewards and support the network's security.	Users need a way to earn passive income and support the network's security.	Introduce a staking mechanism where users can lock up their $KIN tokens to earn rewards.	Passive income for users, increased network security, enhanced token value stability.
Web3 Ecommerce	A decentralized marketplace for beauty products using $KIN tokens.	Users need a transparent, secure, and efficient	Develop a Web3 ecommerce platform that uses blockchain technology for transactions.	Transparent transactions, secure payments, enhanced user trust, and efficient commerce.

PROJECT $KIN GrowthFund: Empowering Brand Expansion

Integration	Mechanism	Benefits
→ Investment Platform: $KIN GrowthFund is an investment platform that provides funding to beauty industry brands needing capital to scale up their business and sales. → Selection Process: Brands can apply for funding through the platform, presenting their business plans, growth strategies, and funding requirements. → Investment Mechanism: Selected brands receive investment in the form of $KIN Tokens, which they can use for various growth initiatives such as marketing, product development, and expanding distribution channels.	1. Application Submission: Brands submit their funding applications, detailing their business plans and growth strategies. 2. Funding Allocation: Based on the votes, funds are allocated to the selected brands. 3. Performance Tracking: The platform tracks the performance of the funded brands, providing regular updates to the community on their progress and impact. *Further plan Investor Voting: $KIN Token holders review the applications and vote for the brands they want to support.	→ Brand Growth: Provides essential funding to beauty industry brands, enabling them to scale their operations and increase sales. → Community Engagement: Involves the $KIN community in the investment process, fostering a sense of ownership and engagement. → Market Expansion: Helps brands expand their market reach and improve their product offerings, benefiting both the brands and consumers.

PROJECT 24/7 Customer Support and Product Explanation

Integration :

→ Implement AI that can provide instant and accurate responses to customer inquiries 24/7. These AI can be trained on the brand's product portfolio and skincare or beauty product knowledge base to offer personalized recommendations and detailed product explanations.

Problem	AI Solution	Benefits
→ Brands often struggle to provide consistent, round-the-clock support to answer customer queries about their skincare or beauty products.	→ AI-Powered: Implement AI that can provide instant and accurate responses to customer inquiries 24/7. These AI can be trained on the brand's product knowledge and skincare or beauty knowledge base to offer personalized recommendations and detailed product explanations.	→ Continuous Support: Ensures customers always have access to support, enhancing customer satisfaction and trust.
→ Customers frequently seek detailed explanations about product benefits, ingredients, and suitability for their skin concerns.	→ Virtual Assistants: AI virtual assistants can guide customers through product features, usage instructions, and benefits, ensuring they have all the information they need to make informed decisions.	→ Continuous Support: Ensures customers always have access to support, enhancing customer satisfaction and trust.
→ Customers often feel overwhelmed by the vast array of skincare or beauty products available and are unsure which products are best suited for their specific skin needs.	→ Personalized Product Recommendations: AI algorithms analyze consumer data, including skin type, concerns, and preferences, to offer personalized skincare or beauty product recommendations	→ Enhanced Decision-Making: Simplifies the process of choosing skincare or beauty products, reducing confusion and improving satisfaction.

PROJECT Food & Drug Regulatory Registration Consultancy: Ensuring Compliance and Facilitating Market Access

Integration :

→ Comprehensive guidance on Food & Drug Regulatory approval, export and import assistance, and halal certification support.

→ AI-driven solutions streamline documentation and compliance processes.

Problem	AI Solution	Benefits
→ Regulatory Complexity: Individuals or new businesses looking to create skincare or beauty products often lack the knowledge and expertise to navigate the complex Food & Drug Regulatory product registration process.	→ Comprehensive Guidance: The Food & Drug Regulatory registration consultancy helps individuals or businesses looking to create skincare or beauty products navigate the complex process of obtaining Food & Drug Regulatory approval. This includes information related to regulatory standards, documentation, and submission processes.	→ Regulatory Compliance: Ensures that all products meet Food & Drug Regulatory regulatory standards, building trust and credibility with consumers and industry partners.
	→ AI-Driven Solutions: AI assists in preparing and submitting required documentation, streamlining the compliance process, and providing real-time updates on regulatory changes.	→ Efficiency and Accuracy: AI-driven solutions reduce errors and streamline the documentation process, saving time and resources for businesses.
→ Export and Import Challenges: Existing skincare or beauty brands may struggle with exporting their products internationally or importing foreign products into Indonesia.	→ Export and Import Assistance: Provides guidance for brands looking to export their products internationally or import foreign skincare or beauty products into Indonesia.	→ Market Expansion: Facilitates the entry of skincare or beauty products into new markets through export and import guidance, expanding business opportunities.
→ Halal Certification: Ensuring products meet halal standards can be challenging without proper guidance.	→ Halal Certification Support: Assists in the process of obtaining halal certification, ensuring products meet the required standards for halal markets.	→ Halal Certification: Opens access to halal markets by ensuring products meet halal certification requirements.

PROJECT $KIN ReviewHub: Enhancing Transparency and Value

Integration :

→ A platform for user reviews, brand engagement, and product ranking based on feedback.

→ Verified reviews and a dynamic ranking system increase transparency and trust.

Problem	Solution	Features	Benefits
→ Consumer Confusion: Consumers find it difficult to choose the right products or keep up with trends due to the overwhelming number of options available and the lack of reliable comparative information.	→ User Reviews: $KIN Token holders and users can leave reviews and comments on skincare or beauty products, providing valuable feedback to other consumers and brands.	→ Verified Reviews: Only verified users or $KIN Token holders can leave reviews, ensuring authenticity and reducing spam. → Incentives: Users can earn $KIN Tokens for leaving detailed and helpful reviews, encouraging quality contributions.	→ Transparency: Provides a transparent platform where users can share honest feedback and experiences, helping others make informed decisions.
	→ Brand Engagement: Brands can engage with reviewers, respond to feedback, and use insights to improve their products.	→ Community Interaction: Users can upvote or respond to reviews, fostering a community of engaged skincare or beauty enthusiasts.	→ Brand Value: Helps brands increase their product value by showcasing positive reviews and responding to customer feedback.
	→ Product Ranking: The platform ranks products based on the number and quality of reviews, providing a transparent and dynamic view of product popularity and effectiveness.	→ Rating System: Products are rated based on user feedback, with detailed comments and ratings for various aspects such as effectiveness, ingredients, and value for money.	→ Market Insights: Brands gain valuable insights into consumer preferences and product performance, aiding in product development and marketing strategies.
	→ Guidelines and Trends: The platform offers guidelines to help consumers choose products based on reviews and trends, making it easier to compare products and stay informed.	→ Guideline Tools: Tools and filters to help users compare products easily and stay updated on the latest trends.	→ Consumer Empowerment: Empowers consumers with reliable information, making it easier to choose the right products and stay informed about trends.

PROJECT $KIN Market Monitoring

Integration :

→ AI aggregates and analyzes data contributed for beauty Industry Market Monitoring.

Problem	AI Solution	Benefits
→ Limited Sourcing: Traditional product development processes in the beauty industry can be slow in regards of trend search, which a lot of time of desktop research.	→ AI can provide Skincare/beauty trend update such as information of any new and exist skincare or beauty product, to update brand what's happening without manual research.	→ Stay ahead of Market Trends with more faster.
→ Ineffective Market Review Monitoring: Brand need to do desktop research due to specific product opportunity by ingredients which take more time to sort which are similar product with specific details like price, usp, etc.	→ AI can provide list of competitor product based on hero ingredient, details ingredients, USP information, price and product size variations details.	→ Effective to explore an opportunity, define red/blue ocean product, or learn from competitor. Brand able to define the opportunity of new product or market review with details information such as ingredient, price, USP, etc.
	→ Sourcing data update of beauty industry update by real skincare or beauty enthusiast, they are able to contribute every single skincare or beauty product to input and the details	→ Real data to use for analyzing market.

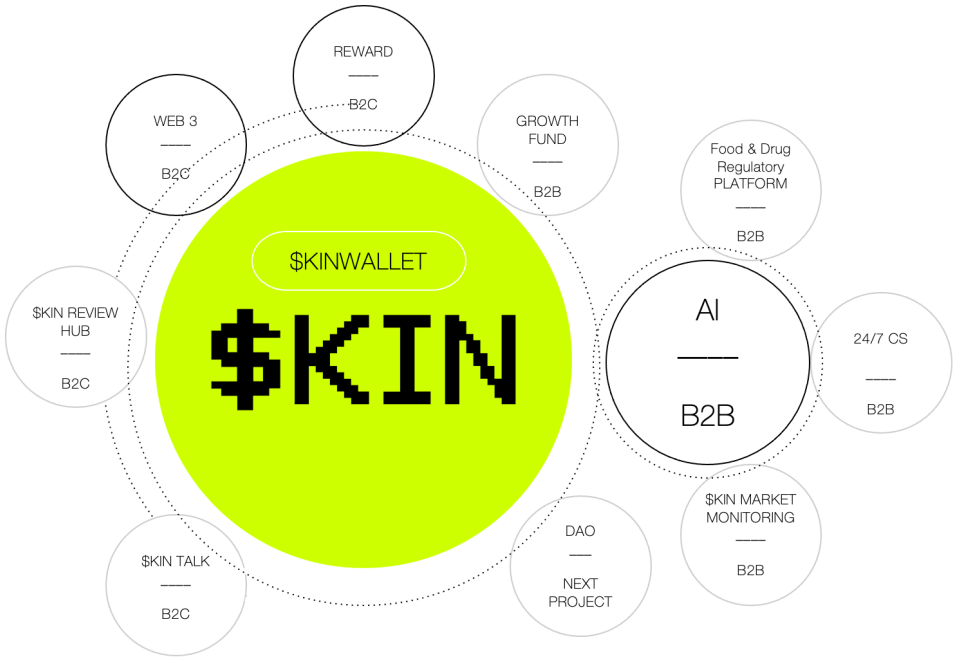
PROJECT $kinTalk : Direct Skincare Consultation Platform

Integration :

→ Direct Chat Consultation Platform with Experts: $kinTalk allows users to chat directly with dermatologists, skincare specialists, and lab professionals. This ensures users receive personalized, accurate, and reliable information tailored to their specific needs.

Problem	AI Solution	Benefits
→ Lack of Knowledge: Many consumers lack the knowledge to choose the right skincare products and routines for their specific needs.	→ Direct Chat Consultation Platform with Experts: $kinTalk allows users to chat directly with dermatologists, skincare specialists, and lab professionals. This ensures users receive personalized, accurate, and reliable information tailored to their specific needs.	→ Immediate Assistance: Users can get real-time answers to their skincare questions and concerns.
→ Misinformation: Consumers often encounter conflicting advice about skincare, making it difficult to find reliable information.		→ Personalized Advice: Customized recommendations and solutions based on individual skin types and conditions.
→ Access to Experts: There is currently no dedicated platform that connects consumers directly with dermatologists, skincare specialists, and lab professionals.		→ Accessibility: Makes expert advice more affordable and accessible to a wider audience.

The $KIN Token Ecosystem

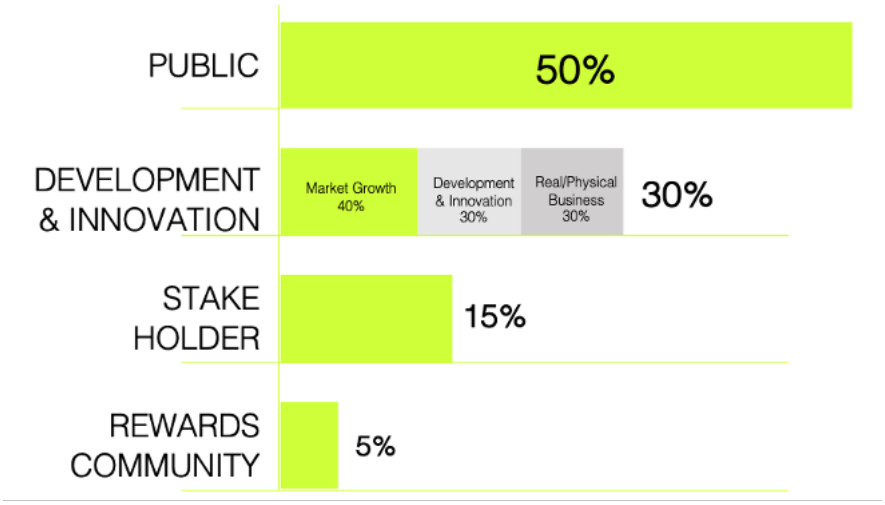


Fee Generation within the Ecosystem

The $KIN Token ecosystem is designed not only to provide value but also to generate sustainable revenue through multiple channels:

Channel	Description	Fee
Subscription Model for $KIN AI	Is an advanced artificial intelligence system specifically designed for the beauty industry. $KIN AI continuously evolves and learns over time, improving its accuracy and effectiveness with each interaction. (24/7, Food & Drug Regulatory CONSULTANT, $KIN MARKET MONITORING)	→ Basic Free Tier: Limited recommendations and features. → Premium Subscription: In-depth analysis, personalized routines, exclusive content for a monthly or annual fee.
Affiliate Fees from Web3 Ecommerce Platform	Generates revenue from product purchases and brand listings.	→ Commission: Earns a commission on products purchased.
$KIN WALLET	Applies small fees for transactions and withdrawals to ensure secure handling of $KIN Tokens.	→ Operational Fees: A small fee applied for transactions and withdrawals made through the $KIN Wallet.
Platform Fees for $KIN ReviewHub	Brand engagement	→ Promoted Advertising Listing: Brands can pay for promoted listings to highlight their products
$KIN GrowthFund	Provides funding for beauty industry brands	Investors purchase $KIN Tokens to participate in funding opportunities through the $KIN GrowthFund. Brands that receive funding are required to commit to a buyback agreement, where they allocate a portion of their profits or revenue to repurchase $KIN Tokens from the market. This buyback process, facilitated by transparent financial reporting and smart contracts, ensures continuous engagement and value creation within the ecosystem. The repurchased tokens are then reintegrated into the $KIN treasury or used for staking rewards, enhancing the token's value and supporting the ecosystem's sustainability.

Tokenomic



Basic Information

Project Name:	$KINSPACE	
Token Name:	$KIN	
Symbol:	$KIN	
Token Contract Address:	0x176f911705ef58DC54a47969656baeE22275c110	
Decimals:	18	
Contract Ownership:	Renounced YES	
Launch Date:	02/02/24	
Total Supply:	1,000,000,000	
50%	PUBLIC	
30%	DEVELOPMENT & INNOVATION	Market Growth : 40% This allocation aims to build a strong and engaged community, marketing campaigns, staking rewards, and governance incentives.
		Development & Innovation: 30% Token will be allocated for developers and innovations. This portion incentivizes those who contribute to the project's development, innovation, and feature enhancement.
		Real / Physical Business: 30%
15%	STAKEHOLDER	
5%	REWARDS COMMUNITY	

Token Security and Trust Measures

To ensure the security, integrity, and trustworthiness of the SKINSPACE ecosystem, we have implemented several robust security measures. These measures are designed to protect the interests of our token holders, stakeholders, and the community, while also addressing concerns about potential rug pulls and ensuring the long-term sustainability of the project.

Multi Signature Wallets

Multi Signature Wallets for Enhanced Security
Purpose: Multi Signature wallets are used to manage staked tokens and project funds, providing an additional layer of security and transparency. This mechanism requires multiple approvals for any transactions, significantly reducing the risk of unauthorized access and misuse of funds.

Key Features of Multi Signature Wallets:	
Multiple Approvals:	Transactions from multi signature wallets require the approval of multiple trusted parties. This means that no single individual has control over the funds, enhancing security and trust.
Decentralized Control:	By distributing control among several parties, multisignature wallets prevent unilateral actions and reduce the risk of fraud.
Transparent Transactions:	All transactions from multi signature wallets are publicly recorded on the blockchain, providing transparency and accountability.
Reduced Risk of Hacking:	The need for multiple approvals makes it significantly more difficult for hackers to compromise the wallet and access funds.
Implementation in SKINSPACE:	
Stakeholder Tokens:	All staked tokens from stakeholders are managed through multi signature wallets. This ensures that any transaction involving these tokens requires multiple approvals, thereby safeguarding the interests of stakeholders.
Development & Innovation Funds:	Funds allocated for development, innovation, and business expansion are also managed using multi signature wallets. This guarantees that these critical resources are protected and used appropriately.

$KIN Token Ecosystem Roadmap 2023 - 2027

Phase 1: Preparation and Foundation (Q3 2023 - Q1 2024)

2023 - 2024 I Q3 - Q4:

Preparation and Smart Contract Development

- → Conduct extensive market research to identify target demographics, competitor analysis, and market opportunities.
- → Begin the development of smart contracts for the $KIN Token, ensuring security and efficiency in transactions.

2024 I Q1:

ICO Planning and Marketing Strategy

- → Develop a comprehensive plan for the ICO, including Tokenomics, pricing, and distribution strategy.
- → Formulate a robust marketing strategy to raise awareness about the project and attract potential investors.
- → Establish initial partnerships with key stakeholders and influencers in the beauty industry.

Phase 2: Initial Launch and Community Building (Q2 2024 - Q3 2024)

2024 I Q2 :

ICO Launch and Community Building

- → Launch the ICO campaign, leveraging marketing efforts to reach a wide audience and encourage participation.
- → Build an active and engaged community around the $KIN Token project through social media, forums, and community events.
- → Conduct regular updates and communication with the community to keep them informed about project developments.
- → Initiate a presale ICO campaign to attract early investors and supporters, offering exclusive bonuses and discounts.
- → Conduct targeted marketing campaigns to promote the presale ICO and generate interest among potential investors.
- → Ensure compliance with relevant regulations and legal requirements for conducting a presale ICO, including KYC/AML procedures.

2024 I Q3 :

Career Hiring and Expansion

- → Expand the team by hiring skilled professionals in key areas such as development, marketing, and operations.
- → Develop strategic partnerships with additional beauty brands and industry players to expand the project's reach.
- → Continue community engagement efforts to foster a strong sense of belonging and loyalty among supporters.

Phase 3: Building the Core Ecosystem (Q4 2024 - Q1 2025)

2024 I Q4 :

$KIN Wallet

- → Launch the secure $KIN Wallet for storing and transacting $KIN Tokens.
- → Ensure user-friendly features and robust security measures for safe transactions.

Rewards

- → Apply initiate rewards with HAUM brand.

Phase 4: Launching Key Projects (2025 - 2026)

2025 I Q2:

AI 24/7

- → Launch the AI-based $KIN analysis tool to provide personalized skincare or beauty recommendations to users.
- → Integrate AI capabilities to analyze user data and offer tailored product suggestions.

2025 I Q3:

Web3 Commerce Platform

- → Official launch of the Web3 commerce platform, allowing users to purchase skincare or beauty products using $KIN Tokens.
- → Implement blockchain technology to ensure transparent and secure transactions.

$KIN GrowthFund

- → Launch the $KIN GrowthFund to provide funding for beauty brands needing capital to scale their business.
- → Implement community-driven selection and performance tracking.

2025 I Q4:

Food & Drug Regulatory Registration Consultancy

- → Expand the Food & Drug Regulatory registration consultancy to provide guidance for creating skincare or beauty products, exporting, importing, and obtaining halal certification.
- → Utilize AI to streamline documentation and compliance processes.

Exchange Listing and Project Building

- → Secure listings on major cryptocurrency exchanges to increase liquidity and accessibility of the $KIN Token.
- → Focus on building the foundation for project expansion and development, including further partnerships, platform enhancements, and ecosystem growth.
- → Prepare for the official launch of the Web3 commerce platform, AI-based $KIN analysis tool, and $KIN Wallet in the upcoming phases.

Phase 5: Expanding the Ecosystem (2026)

2026 I Q1:

$KIN ReviewHub

- → Launch the $KIN ReviewHub for product reviews, brand engagement, and product ranking based on feedback.
- → Implement features for verified reviews, rating systems, and community interaction.

2026 I Q3:

$KIN Market Monitoring

- → Launch the $KIN market monitoring for provide data in beauty industry.

2026 I Q4:

SkinTalk: Direct Skincare Consultation Platform

- → Launch the SkinTalk platform, enabling users to chat directly with dermatologists, skincare specialists, and lab professionals.
- → Integrate AI-based preliminary skincare advice tools to assist users before connecting them to experts.
- → Develop a subscription and fee-based model for consultations and expert interactions.

Phase 6: Scaling and Innovation (2026 - 2027)

2027 I Q1:

Future Developments and Scaling

- → Focus on continuous improvement and scaling of existing projects.
- → Explore new opportunities and innovations within the $KIN Token ecosystem.

SMART CONTRACT

```
        // SPDX-License-Identifier: MIT

pragma solidity ^0.8.2;

abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }

    function _msgData() internal view virtual returns (bytes calldata) {
        return msg.data;
    }
}
interface IERC20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}
library Address {
    function isContract(address account) internal view returns (bool) {
        uint256 size;
        assembly {
            size := extcodesize(account)
        }
        return size > 0;
    }
    function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance");

        (bool success, ) = recipient.call{value: amount}("");
        require(success, "Address: unable to send value, recipient may have reverted");
    }
    function functionCall(address target, bytes memory data) internal returns (bytes memory) {
        return functionCall(target, data, "Address: low-level call failed");
    }
    function functionCall(
        address target,
        bytes memory data,
        string memory errorMessage
    ) internal returns (bytes memory) {
        return functionCallWithValue(target, data, 0, errorMessage);
    }
    function functionCallWithValue(
        address target,
        bytes memory data,
        uint256 value
    ) internal returns (bytes memory) {
        return functionCallWithValue(target, data, value, "Address: low-level call with value failed");
    }
    function functionCallWithValue(
        address target,
```

```
        bytes memory data,
        uint256 value,
        string memory errorMessage
    ) internal returns (bytes memory) {
        require(address(this).balance >= value, "Address: insufficient balance for call");
        require(isContract(target), "Address: call to non-contract");

        (bool success, bytes memory returndata) = target.call{value: value}(data);
        return verifyCallResult(success, returndata, errorMessage);
    }
    function functionStaticCall(address target, bytes memory data) internal view returns (bytes memory) {
        return functionStaticCall(target, data, "Address: low-level static call failed");
    }
    function functionStaticCall(
        address target,
        bytes memory data,
        string memory errorMessage
    ) internal view returns (bytes memory) {
        require(isContract(target), "Address: static call to non-contract");

        (bool success, bytes memory returndata) = target.staticcall(data);
        return verifyCallResult(success, returndata, errorMessage);
    }
    function functionDelegateCall(address target, bytes memory data) internal returns (bytes memory) {
        return functionDelegateCall(target, data, "Address: low-level delegate call failed");
    }
    function functionDelegateCall(
        address target,
        bytes memory data,
        string memory errorMessage
    ) internal returns (bytes memory) {
        require(isContract(target), "Address: delegate call to non-contract");

        (bool success, bytes memory returndata) = target.delegatecall(data);
        return verifyCallResult(success, returndata, errorMessage);
    }
    function verifyCallResult(
        bool success,
        bytes memory returndata,
        string memory errorMessage
    ) internal pure returns (bytes memory) {
        if (success) {
            return returndata;
        } else {
            if (returndata.length > 0) {
                assembly {
                    let returndata_size := mload(returndata)
                    revert(add(32, returndata), returndata_size)
                }
            } else {
                revert(errorMessage);
            }
        }
    }
}
abstract contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
    constructor() {
        _setOwner(_msgSender());
    }
```

```
    function owner() public view virtual returns (address) {
        return _owner;
    }
    modifier onlyOwner() {
        require(owner() == _msgSender(), "Ownable: caller is not the owner");
        _;
    }
    function renounceOwnership() public virtual onlyOwner {
        _setOwner(address(0));
    }
    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        _setOwner(newOwner);
    }

    function _setOwner(address newOwner) private {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);
    }
}
contract ERC20$kin is IERC20, Ownable {

    using Address for address;

    mapping (address => uint256) private _rOwned;
    mapping (address => uint256) private _tOwned;
    mapping (address => mapping (address => uint256)) private _allowances;

    mapping (address => bool) private _isExcluded;
    address[] private _excluded;

    uint256 private constant MAX = ~uint256(0);
    uint256 private _tTotal;
    uint256 private _rTotal;
    uint256 private _tFeeTotal;
    uint8 public _feeDivider;

    string private _name;
    string private _symbol;
    uint8 private _decimals;

    constructor (address owner_, string memory name_, string memory symbol_, uint8 decimals_, uint256 supply_, uint8 feeDivider_) Ownable() {

        require(feeDivider_ > 0, "ERC20 Macadamia: divider has to be grater than zero");
        require(decimals_ < 19, "ERC20 Macadamia: decimals has to be between 0 and 18");
        require(supply_ > 0, "ERC20 Macadamia: supply has to be grater than zero");

        _name = name_;
        _symbol = symbol_;
        _decimals = decimals_;
        _feeDivider = feeDivider_;
        _tTotal = supply_;
        _rTotal = (MAX - (MAX % _tTotal));
        _rOwned[owner_] = _rTotal;
        emit Transfer(address(0), owner_, _tTotal);

        transferOwnership(owner_);

    }

    function name() external view returns (string memory) {
```

```
    return _name;
}

function symbol() external view returns (string memory) {
    return _symbol;
}

function decimals() external view returns (uint8) {
    return _decimals;
}

function totalSupply() external view override returns (uint256) {
    return _tTotal;
}

function balanceOf(address account) external view override returns (uint256) {
    if (_isExcluded[account]) return _tOwned[account];
    return tokenFromReflection(_rOwned[account]);
}

function transfer(address recipient, uint256 amount) external override returns (bool) {
    _transfer(_msgSender(), recipient, amount);
    return true;
}

function allowance(address owner, address spender) external view override returns (uint256) {
    return _allowances[owner][spender];
}

function approve(address spender, uint256 amount) external override returns (bool) {
    _approve(_msgSender(), spender, amount);
    return true;
}

function transferFrom(address sender, address recipient, uint256 amount) external override returns (bool) {
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(), _allowances[sender][_msgSender()] - amount);
    return true;
}

function increaseAllowance(address spender, uint256 addedValue) external virtual returns (bool) {
    _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
    return true;
}

function decreaseAllowance(address spender, uint256 subtractedValue) external virtual returns (bool) {
    _approve(_msgSender(), spender, _allowances[_msgSender()][spender] - subtractedValue);
    return true;
}

function isExcluded(address account) external view returns (bool) {
    return _isExcluded[account];
}

function totalFees() external view returns (uint256) {
    return _tFeeTotal;
}

function reflect(uint256 tAmount) external {
    address sender = _msgSender();
    require(!_isExcluded[sender], "Excluded addresses cannot call this function");
    (uint256 rAmount,,,,) = _getValues(tAmount);
```

```
    _rOwned[sender] = _rOwned[sender] - rAmount;
    _rTotal = _rTotal - rAmount;
    _tFeeTotal = _tFeeTotal + tAmount;
}

function reflectionFromToken(uint256 tAmount, bool deductTransferFee) external view returns(uint256) {
    require(tAmount <= _tTotal, "Amount must be less than supply");
    if (!deductTransferFee) {
        (uint256 rAmount,,,,) = _getValues(tAmount);
        return rAmount;
    } else {
        (,uint256 rTransferAmount,,,) = _getValues(tAmount);
        return rTransferAmount;
    }
}

function tokenFromReflection(uint256 rAmount) public view returns(uint256) {
    require(rAmount <= _rTotal, "Amount must be less than total reflections");
    uint256 currentRate =  _getRate();
    return rAmount / currentRate;
}

function excludeAccount(address account) external onlyOwner() {
    require(!_isExcluded[account], "Account is already excluded");
    if(_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}

function includeAccount(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}

function _approve(address owner, address spender, uint256 amount) private {
    require(owner != address(0), "ERC20: approve from the zero address");
    require(spender != address(0), "ERC20: approve to the zero address");

    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
}

function _transfer(address sender, address recipient, uint256 amount) private {
    require(sender != address(0), "ERC20: transfer from the zero address");
    require(recipient != address(0), "ERC20: transfer to the zero address");
    require(amount > 0, "Transfer amount must be greater than zero");
    if (_isExcluded[sender] && !_isExcluded[recipient]) {
        _transferFromExcluded(sender, recipient, amount);
    } else if (!_isExcluded[sender] && _isExcluded[recipient]) {
        _transferToExcluded(sender, recipient, amount);
    } else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
        _transferStandard(sender, recipient, amount);
```

```
        } else if (_isExcluded[sender] && _isExcluded[recipient]) {
            _transferBothExcluded(sender, recipient, amount);
        } else {
            _transferStandard(sender, recipient, amount);
        }
    }

    function _transferStandard(address sender, address recipient, uint256 tAmount) private {
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount, uint256 tFee) = _getValues(tAmount);
        _rOwned[sender] = _rOwned[sender] - rAmount;
        _rOwned[recipient] = _rOwned[recipient] + rTransferAmount;
        _reflectFee(rFee, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function _transferToExcluded(address sender, address recipient, uint256 tAmount) private {
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount, uint256 tFee) = _getValues(tAmount);
        _rOwned[sender] = _rOwned[sender] - rAmount;
        _tOwned[recipient] = _tOwned[recipient] + tTransferAmount;
        _rOwned[recipient] = _rOwned[recipient] + rTransferAmount;
        _reflectFee(rFee, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function _transferFromExcluded(address sender, address recipient, uint256 tAmount) private {
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount, uint256 tFee) = _getValues(tAmount);
        _tOwned[sender] = _tOwned[sender] - tAmount;
        _rOwned[sender] = _rOwned[sender] - rAmount;
        _rOwned[recipient] = _rOwned[recipient] + rTransferAmount;
        _reflectFee(rFee, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function _transferBothExcluded(address sender, address recipient, uint256 tAmount) private {
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount, uint256 tFee) = _getValues(tAmount);
        _tOwned[sender] = _tOwned[sender] - tAmount;
        _rOwned[sender] = _rOwned[sender] - rAmount;
        _tOwned[recipient] = _tOwned[recipient] + tTransferAmount;
        _rOwned[recipient] = _rOwned[recipient] + rTransferAmount;
        _reflectFee(rFee, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function _reflectFee(uint256 rFee, uint256 tFee) private {
        _rTotal = _rTotal - rFee;
        _tFeeTotal = _tFeeTotal + tFee;
    }

    function _getValues(uint256 tAmount) private view returns (uint256, uint256, uint256, uint256, uint256) {
        (uint256 tTransferAmount, uint256 tFee) = _getTValues(tAmount);
        uint256 currentRate =  _getRate();
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee) = _getRValues(tAmount, tFee, currentRate);
        return (rAmount, rTransferAmount, rFee, tTransferAmount, tFee);
    }

    function _getTValues(uint256 tAmount) private view returns (uint256, uint256) {
        uint256 tFee = tAmount / _feeDivider;
        uint256 tTransferAmount = tAmount - tFee;
        return (tTransferAmount, tFee);
    }

    function _getRValues(uint256 tAmount, uint256 tFee, uint256 currentRate) private pure returns (uint256, uint256, uint256) {
}
```

```
        uint256 rAmount = tAmount * currentRate;
        uint256 rFee = tFee * currentRate;
        uint256 rTransferAmount = rAmount - rFee;
        return (rAmount, rTransferAmount, rFee);
    }

    function _getRate() private view returns(uint256) {
        (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
        return rSupply / tSupply;
    }

    function _getCurrentSupply() private view returns(uint256, uint256) {
        uint256 rSupply = _rTotal;
        uint256 tSupply = _tTotal;
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
            rSupply = rSupply - _rOwned[_excluded[i]];
            tSupply = tSupply - _tOwned[_excluded[i]];
        }
        if (rSupply < _rTotal / _tTotal) return (_rTotal, _tTotal);
        return (rSupply, tSupply);
    }
}
```