Magnet Milestone 3 Proposal: An Appchain Tech Stack leverages the Coretime Model to enhance DOT use cases and facilitate Polkadot's growth

Proponent: 15mAbCUcBa7jT8Rak8j7hC5w9jUM8b7LFUuhp9nENqmtehLn
Date: 7.4.2024
Requested DOT: 33,956.17 DOT | USD $294,400 (The amount of DOT is converted based on the EMA-7 price(Subscan) on the day of the official submission, 2024-4-7, at the rate of $8.67 per DOT.)
Short description: Leveraging the Coretime Model, Magnet Stack is an Appchain Stack Solution Tailored to Amplify the use cases of DOT and facilitate Polkadot Ecosystem Growth.

---

1. Context

As the blockchain industry continues to evolve, the trend towards diversified forms of DApps becomes increasingly apparent. In addition to the currently prevalent smart contract DApps, more and more applications are opting for the form of Appchains to support their business scenarios. Furthermore, Rollup technology is gradually becoming mainstream for scaling blockchain performance.

At the recent Sub0 conference, Dr. Gavin proposed a new network concept called the JAM chain. JAM chain is not a smart contract chain, but its emergence will enable Polkadot to have the capability to handle Rollup compatibility. With the advent of JAM chain, Polkadot will have comprehensive capabilities to support both conventional

Appchains and Rollup. This will ultimately position Polkadot as a leader in the future blockchain world, becoming the world computer of Web3.

As the first common good smart contract platform developed on coretime with DOT as the Gas fee, Magnet has successfully completed the previously planned development tasks. Furthermore, the alignment between the Rollup module scheduled for milestone 3 and the emerging concept of JAM chain is a testament to the high level of synergy between Magnet and the Polkadot ecosystem. However, the launch of the JAM chain is expected to occur 18-20 months later. To prevent any mismatch between Magnet's development and the JAM chain, Magnet has decided to postpone the development of the Rollup module until the JAM chain interface is confirmed. This decision aims to enhance development efficiency and mitigate development risks.

Meanwhile, as the influence of Polkadot continues to expand and technology undergoes continuous upgrades, there will be an increasing number of development teams eager to participate in the Polkadot ecosystem. Moreover, within the community, there are high-quality ecosystem projects like Tanssi and Lastic that provide developers with tools to facilitate the construction of their applications. However, for new developers entering the ecosystem, they need to separately understand Polkadot-SDK, Tanssi, and Lastic, and then integrate them based on their actual needs for development. This process incurs intangible costs in terms of both technical and business learning.

In order to reduce the development learning costs for development teams, encourage the emergence of a large number of common good applications, and ultimately promote the prosperity of the Polkadot ecosystem and the application scenarios of DOT, Magnet will integrate our development experience and economic model experience on Magnet. We will abstract the operational modules based on coretime and integrate the ecosystem functionalities provided by Tanssi and Lastic to create a Common Good Stack for the community. With Magnet Stack, developers will be able to integrate the Polkadot ecosystem in a one-stop manner and acquire the ability to build their own business models based on DOT and coretime, allowing them to focus more on business

development. With Magnet Stack, whether deploying a parathread, container chain, using Bulk coretime, or On-demand coretime mode, developers will be able to quickly implement these through our services, ultimately facilitating the widespread application of Polkadot technology to the greatest extent possible.

Furthermore, Magnet is currently undergoing continuous testing on the Rococo network. We will continue to keep the Magnet Network updated in accordance with the progress of Polkadot-SDK to ensure that Magnet can be synchronized online immediately after Coretime is launched on Polkadot.

As of Milestone 2, the project has successfully improved optimization and updated its overall economic system and operational logic on the public GitHub repository, in line with ongoing progress. All features outlined in Milestone 2 have been developed, and we are currently in the phase of continuous testing and code optimization. And for Milestone 3, the main objective is the development of Magnet Stack, which encompasses coretime and parachain functionalities. This Stack serves as a comprehensive toolkit for developers, providing them with the necessary tools and resources to integrate coretime and parachain features into their projects effectively. Through continuous updates and improvements, Magnet ensures that the Stack remains robust, reliable, and aligned with the evolving needs of the ecosystem. This proposal is submitted with the intention of securing funding for our Milestone 3, totaling USD $294,400.00.

2. The Achievement of Milestone 2

All development achievements of milestone 2 have been posted on Github, you can find this in this link: GitHub
You can also follow the instructions to study how to interact with Magnet: https://doc.magport.io/
Magnet Bulk Coretime Testnet has been online on Rococo, you can find it with this link: Magnet Testnet on Rococo

To facilitate use by the average user, we have additionally prepared a web-based application, making it easier for users to experience all of Magnet's features. More functionalities are currently in development, and you can try them out through this link: Rococo Testnet Application

Development of DOT's WASM Platform

Enable Magnet to become a WASM smart contract platform with DOT as the native GAS token and complete the GAS model mechanism. Development of the platform's core functionalities, with updates on the development progress being synchronized and posted on the public GitHub repository. You can find the instruction of deployment Wasm Smart Contract with this Link.

Development of on-chain governance functionality

Completed on-chain governance functionality using DOT as the governance token.

As Magnet operates as a tokenless blockchain, all governance requirements will be delegated to DOT holders. All development progress is synchronized with the public GitHub repository. You can find the development detail with this GitHub link.

Optimizing Coretime Order Pallet

Following the development progress of Parity, upgraded the Coretime Order Pallet logic to maintain optimal compatibility. Completed the compatibility work for Gas processing logic between WASM and EVM. And developed and updated core functionalities to Polkadot-SDK 1.7.0, all progress is synchronized to GitHub. This part is divided to three part, locate respectively at Node, Order Pallet, and primitives**.**

Optimizing Profit Distribution Pallet

Following Parity's development progress, updated and adjusted the logic of the Profit Distribution Pallet (e.g. Coretime credit account), ensuring this feature remains in sync

with the latest coretime ordering logic at all times. After launching on-demand coretime on Rococo, Magnet will active **cross-chain(XCM)** profit distribution on Rococo. Ensured the accuracy and efficiency of profit distribution, with all progress being synchronized to GitHub. You can find the development detail with this GitHub link.

Optimizing Blocks Assurance Pallet

Optimized the logic of the Block Assurance Pallet, refined the code for this feature, and adjusted the configuration parameters. Simultaneously, in line with Parity's latest development progress, update and optimize the code, preparing for future optimization of the code logic for direct ordering by sovereign accounts. Ensure the stable block production of the platform, with development progress being updated on the public GitHub repository. You can find the development detail with this GitHub link.

Optimizing EVM and Bridge Functionality

Maintained EVM functionality, ensured synchronization with other EVM chains, and enhanced the contract compatibility of Magnet EVM. And optimized the bridge functionality and achieved a seamless user experience for the binding module. Maintained the optimization and updating of EVM, with all progress being synchronized to GitHub. You can find the development detail with this Pot and Assets Bridge.

Integrate the Latest XCM pallet for parachain communication

According to Parity's development progress, tested and optimized the XCM code to ensure the reliability of this feature. Completed the HRMP channel testing, enabling other parachains to transfer tokens to Magnet via XCM, and ultimately enter into the EVM. And maintained communication with the Polkadot ecosystem, enabling the seamless transfer of DOT to the platform. Development progress is updated on the public GitHub repository. You can find the development detail with this GitHub link.

Upgrade Magnet to Polkadot-SDK 1.7.0 and optimize the code architecture

To keep pace with Parity's development progress and adapt Magnet to the latest Rococo Coretime module, Magnet has been upgraded to Polkadot SDK 1.7.0. Furthermore, a comprehensive restructuring of all features and the code architecture has been undertaken to accommodate the content changes in version 1.7.0, facilitating future continuous upgrades and optimization of various functions. You can find this from the overview link.

3. The Implementation Plan of Milestone 3

3.1 Maintenance of Magnet Network

The maintenance of the Magnet Network is critical for ensuring its reliability, security, and performance. By continuously upgrading functional modules, progressing in testing network improvements, and conducting ongoing monitoring, the network can provide users with a seamless and dependable experience. This commitment to maintenance helps to safeguard the integrity and usability of the network, ultimately enhancing user satisfaction and trust in the Magnet ecosystem.

As Parity continuously updates its systems, our team must remain vigilant in monitoring and managing the Magnet network to align with these changes. This involves regular updates to address any identified bugs, vulnerabilities, or performance issues, as well as implementing enhancements to keep the network competitive and aligned with industry standards. Additionally, ongoing communication with the community and stakeholders is crucial to gather feedback and prioritize development efforts effectively, ensuring that the network evolves in line with user needs and expectations.

The Coretime module plays a pivotal role in the Magnet network, governing the allocation and management of coretime resources. Maintenance of this module involves constant monitoring for anomalies or issues such as performance bottlenecks or security vulnerabilities, while upgrades aim to improve its efficiency, scalability, and

robustness. Thorough testing of new features and improvements is essential to validate their effectiveness and minimize the risk of disrupting network operations.

The profit distribution module is responsible for fairly distributing profits generated within the Magnet network to relevant stakeholders. Upgrades to this module may involve optimizing algorithms, improving accuracy, and introducing new mechanisms to enhance transparency and fairness in profit distribution. Careful testing and validation are necessary to ensure that upgrades do not introduce unintended consequences or biases in profit distribution.

Similarly, the profit calculation module requires upgrades to enhance accuracy and reliability in calculating profits generated within the Magnet network. These upgrades may focus on improving calculation algorithms, enhancing precision, and incorporating additional factors for more comprehensive profit management. Rigorous testing and validation are crucial to ensure accurate and reliable profit calculations without introducing errors or inaccuracies.

The block assurance module ensures the integrity and reliability of block production and verification processes within the Magnet network. Maintenance involves monitoring for issues or anomalies that may compromise the security or stability of block operations, while upgrades aim to enhance the efficiency, resilience, and security of block assurance mechanisms. Thorough testing and validation are essential to verify the effectiveness and robustness of upgraded block assurance mechanisms.

Monitoring and upgrades of the Polkadot SDK are essential as it serves as the foundation for building and maintaining the Magnet Network. Monitoring involves tracking updates and releases to identify new features, improvements, and bug fixes relevant to the Magnet network. Upgrades aim to incorporate the latest SDK enhancements, ensuring compatibility, performance optimization, and access to new functionalities. Testing and validation of SDK upgrades are critical to ensure seamless integration with existing Magnet components and mitigate potential compatibility issues.

3.2 Magnet Stack Functionality Development

In the process of developing Magnet Stack, several key steps are necessary to ensure a smooth integration and optimal functionality. Firstly, a thorough analysis of the existing functions must be conducted, encompassing support for multiple virtual machines (EVM, WASM, MoveVM), Coretime (bulk and on-demand), advanced features like profit calculation and distribution, as well as eco functions such as Lastic Integration and the Appchain model. These functions will then be systematically defined as independent components, each assigned a specific set of tasks to enhance modularity and efficiency.

Following the definition of functions and components, it's essential to determine inter-component dependencies and interactions. By identifying these dependencies, the components can be integrated seamlessly within the technology stack, ensuring cohesive operation. Additionally, clear and consistent interfaces must be designed for each component, facilitating smooth interaction and accessibility for external developers.

Documentation and development guides play a crucial role in ensuring effective utilization of the technology stack. Detailed technical documentation should be provided for each component and the entire stack, including comprehensive API references, feature descriptions, setup guides, and practical use cases. Furthermore, a comprehensive development guide is essential, offering step-by-step instructions on using the technology stack to build applications, covering everything from environment setup to workflow implementation.

Practical application examples and tutorials are indispensable for developers looking to leverage the technology stack effectively. Developing a series of example applications that showcase the diverse functionalities of the stack will provide valuable insights. Additionally, offering tutorials and case studies will help developers understand how to apply the technology stack in various real-world scenarios and projects.

Finally, ongoing maintenance and updates are essential to keep the technology stack relevant and efficient. Regular updates, based on technological advancements and community feedback, should be implemented to introduce new features, optimize existing components, and address any bugs or issues. A robust version control system, utilizing semantic versioning, helps developers understand the nature and scope of changes with each update.

Multiple Virtual Machines (VM): Magnet Stack supports multiple virtual machines to accommodate different smart contract execution environments. This includes compatibility with EVM, WASM, and MoveVM. By supporting multiple VMs, developers have the flexibility to choose the most suitable environment for their specific use case, thereby enhancing compatibility and interoperability within the ecosystem.

In this milestone, we will integrate MoveVM into our stack, to meet more varied demands for VM.

Fundamental Function: The Magnet Stack encompasses fundamental functions essential for its operation within the Polkadot ecosystem. One such function is OpenGov, facilitating transparent governance processes by enabling proposal submission and on-chain upgrades, and ensuring democratic decision-making. Additionally, the XCM (Cross-Chain Messaging) functionality allows seamless interaction with other chains within Polkadot, facilitating secure communication and asset transfer. Furthermore, the MultiSig feature enhances security and trust by providing multi-signature transaction capabilities, allowing multiple authorized parties to collectively approve transactions, thereby ensuring secure and reliable smart contract deployment and execution. Together, these fundamental functions form the backbone of Magnet Stack, enabling efficient and secure operation within the Polkadot network.

Coretime: Coretime functionality is essential for managing block production and resource allocation within the Magnet network. The Magnet Stack offers two coretime models: Bulk Coretime and On-demand Coretime. Bulk Coretime allows for the

pre-allocation of resources, while On-demand Coretime dynamically allocates resources based on transaction demand. This flexibility enables efficient resource management and ensures optimal performance under varying workload conditions.

This development will abstract two modes of operation, namely On-demand model and Bulk Coretime model. Depending on the chosen setting, the corresponding functionalities of other collaborative modules will automatically adjust accordingly.

Advanced Functions: The Magnet Stack incorporates advanced functions to enhance operational control and management. This includes features such as Profit Calculation, Profit Controller, Profit Distribution, and Block Assurance. Profit Calculation enables the calculation of transaction fees and rewards, while Profit Controller allows for the management and optimization of profit distribution. Profit Distribution ensures fair and transparent distribution of profits among network participants, while Block Assurance enhances the security and reliability of block confirmation processes.

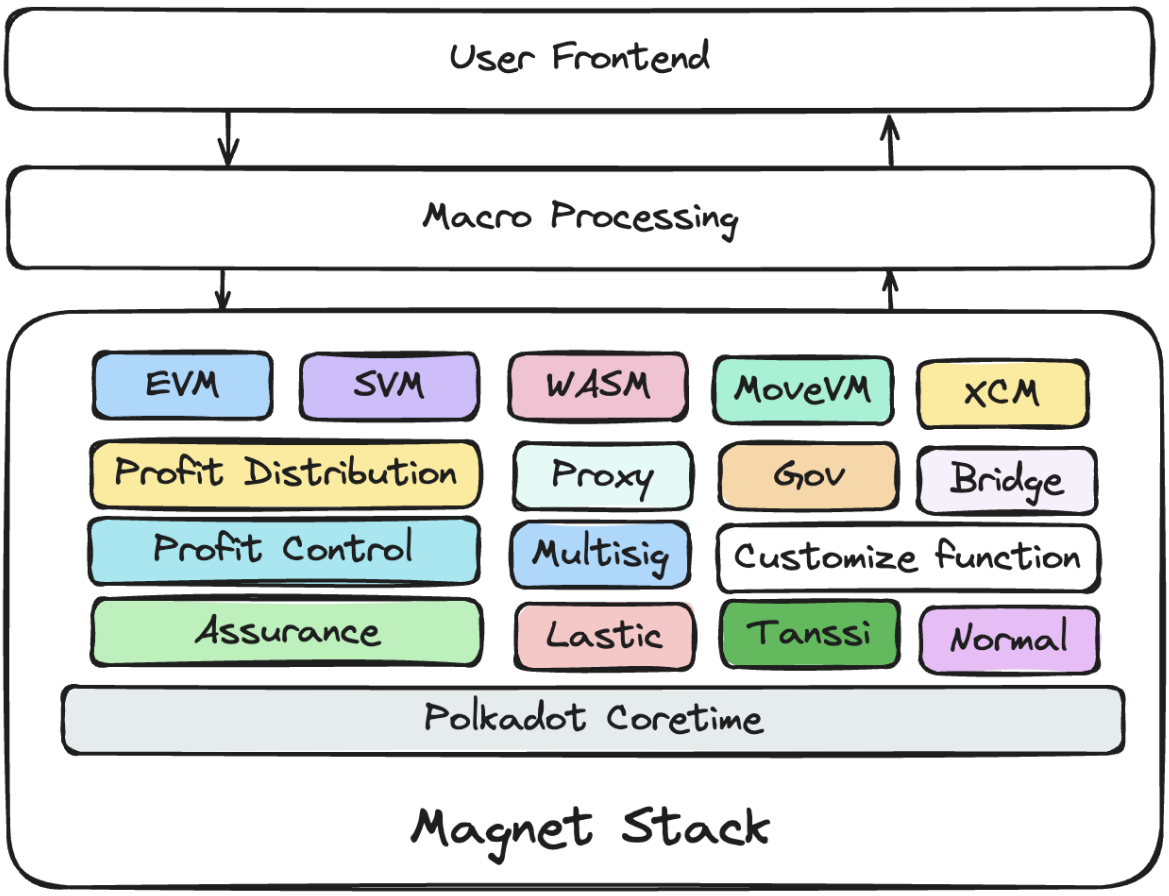
In this development, a new parameter feature will be added to profit control, allowing users to customize the minimum profit level they wish to achieve, thereby further increasing the flexibility of profit control for users. Similarly, Profit distribution will also introduce a set of parameters that allow users to customize the profit distribution target addresses. Regardless of which Parachain the target addresses are on, profits can be distributed via XCM. Lastly, for the Assurance function, the customizable waiting time will dynamically adjust based on the Gas fees and the target difference, to further enhance the user experience.

Eco Functions: Integration with Lastic, is an important infrastructure of coretime, which will provide coretime market to the polkadot ecosystem. so we want to integrate Lastic into our stack. In this way, developers can select the Lastic service when developing projects with our stack, thereby obtaining coretime resources more stably and efficiently.

Appchain Model: The Magnet Stack supports different Appchain models, including parathread and Tanssi-container chain. These models offer diverse options for deploying and managing blockchain applications within the Magnet ecosystem. Para threads provide parallel processing capabilities, enabling scalability and performance optimization, while Tanssi-container chains offer containerized environments for deploying and managing application-specific chains. This flexibility allows developers to choose the most suitable Appchain model based on their application requirements and deployment preferences.

Overall, Magnet Stack provides a comprehensive suite of functionalities designed to empower developers and organizations to build and deploy innovative blockchain applications with ease. By incorporating advanced features, supporting multiple VMs, and offering diverse deployment options, Magnet Stack facilitates the development of robust and scalable blockchain solutions, driving growth and adoption within the ecosystem.

3.3 Architecture of Magnet Stack



Magnet Template Layer

This layer forms the foundation of the entire tool, containing all the templates for Substrate modules that can be instantiated with specific functionalities. These templates should be designed for easy parameterization to allow customization based on user configurations.

Dynamic Generator Layer

This layer is responsible for dynamically generating Substrate code based on the user's provided configurations. This layer can not only rely on normal script measure but also leverage Rust's macro system, especially procedural macros, to facilitate the dynamic code generation process.

User frontend Layer

The user frontend layer is the interface that interacts with users, allowing them to select desired module functionalities, configure parameters, and ultimately generate Substrate code. This layer could be a web application, desktop application, or a command-line tool, depending on your target user base and usage scenarios.

3.4 Magnet Stack Feature Overview

	Category	Functionality	Notes
Magnet Stack	VM	EVM	For EVM smart contract
		WASM	For WASM smart contract
		MoveVM	For Move contract
		SVM	For Solana smart contract (Future Plan)
	Fundamental function	OpenGov	Proposal and on-chain upgrade
		XCM	Interact with Polkadot ecosystem
		MultiSig/Proxy	Multisig traction and Smart contract deployment

	Advanced Function	Profit Control	Customize the desired profit margin to control the timing of coretime ordering
		Profit Calculation	In the on-demand mode, calculations are based on the cost per single coretime, while in the Bulk mode, calculations are based on the average cost.
		Profit Distribution	Allows for the customization of the number and ratio of Profit distribution addresses, while also supporting cross-chain profit distribution via XCM.
		Block Assurance	In the On-demand mode, the waiting time can be dynamically reduced based on the current accumulation of Gas fees. Simultaneously, the trigger time for forced ordering can be adjusted.
	Eco Function	Lastic	Coretime secondary market
		Signet	a platform that simplifies multisig usage for teams and provides a base upon which developers can extend and integrate to create richer Multisig experiences.
	Appchain model	Parathread	Standard Parathread
		Tanssi	Tanssi container chain
	Coretime model	On-demand	On-demand coretime
		Bulk	Bulk Coretime

3.5 Workflow of Magnet Stack

i. Project Initialization: Developers create a new blockchain project using Magnet Stack, they can quickly start the project by choosing from the preset network configurations and templates provided by Magnet Stack.

ii. Module Selection & Configuration:

(1) **Virtual Machines and Features Selection:** developers select the desired virtual machines and features from various Magnet Templates to tailor the project to their specific needs.

(2) **Coretime Model Configuration:** Developers choose between On-demand and Bulk modes for the Coretime model. The selection automatically adjusts other functionalities to align with the chosen mode.

(3) **Fundamental Functionalities Configuration:** Developers configure the core functionalities, with OpenGov for governance and XCM for cross-chain

communication respectively. MultiSig, an optional feature, can be chosen for enhanced security through multi-signature transactions.

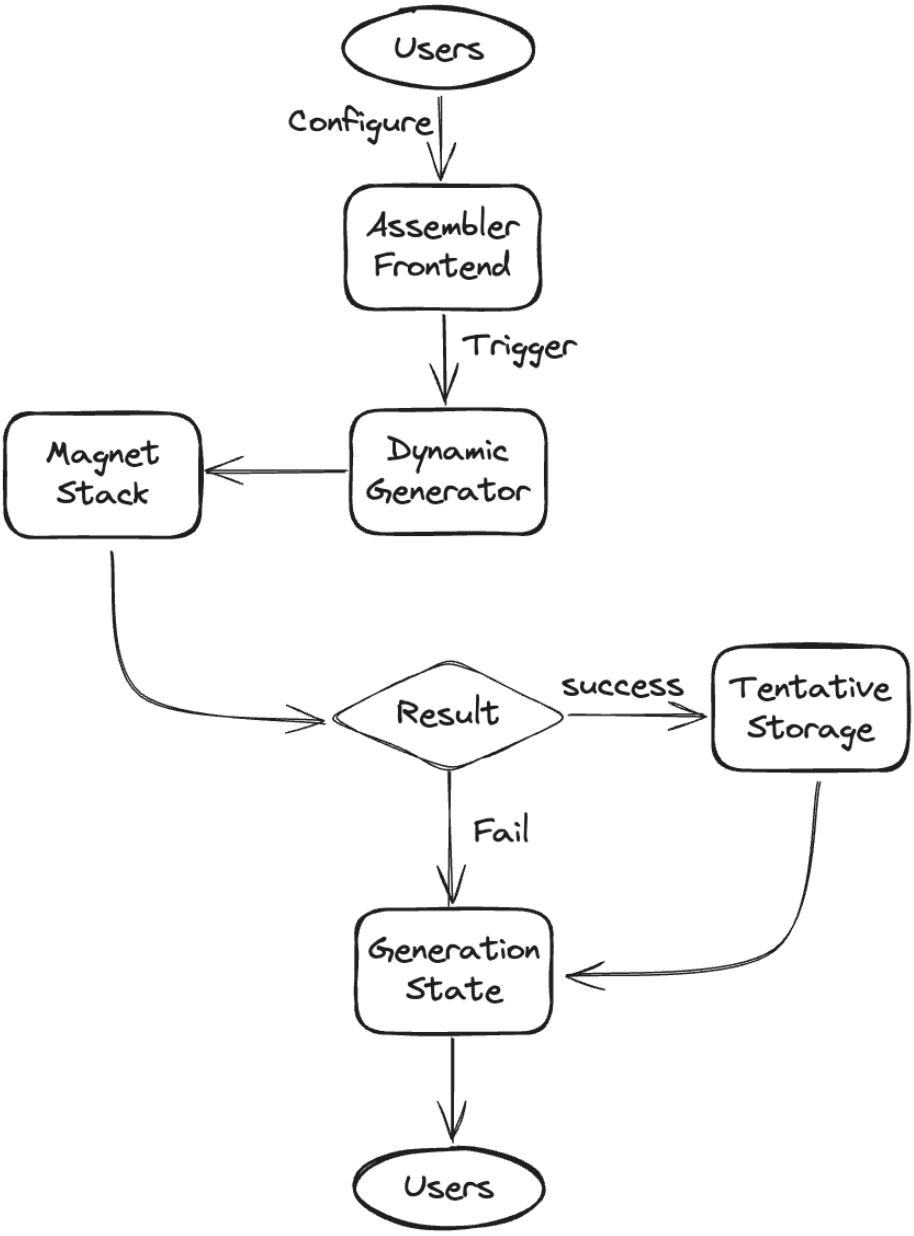
(4) **Advanced Functionalities Configuration:** Developers can adjust the parameters of Profit Control, Calculation, Distribution, and Block Assurance to tailor the configuration to their business needs.

iii. Code Generation: Using the dynamic generator layer of Magnet Stack, the corresponding Substrate code is automatically generated based on the selected modules and configurations. This step reduces the need for manual coding and improves development efficiency. When the code generation has finished, the developer will have all necessary codes to deploy nodes for their independent network with per-selected features.

iv. Business Logic Development: Developers have the flexibility to build their projects using Pallet modules or Smart contracts tailored to their specific business requirements. For Pallet modules, developers can create custom logic and corresponding APIs, and then seamlessly integrate the pallet into the pre-existing Substrate codebase, adhering to Substrate's development standards. In the case of Smart contracts, developers can establish the rules and operations of the blockchain application, creating transaction logic to manage data and interactions effectively. Furthermore, developers can design user interfaces to facilitate end-user interaction and develop off-chain functions for tasks that are more efficiently handled outside the blockchain. Such capabilities grant Magnet Stack the versatility to cater to the varied needs of users across different use cases.

v. Deployment & Release: After completing testing and optimization, the project is ready for deployment. Developers deploy the nodes using the Substrate-based blockchain network code generated by Magnet Stack, which includes the pre-defined features and functionalities. This process involves setting up the network infrastructure, starting the nodes, and registering the parathread. and ensuring they are properly interconnected. Once the network is up and running, developers can deploy their smart contracts to the network or directly initial the business module. Final testing and verification are conducted to ensure that all components of the project are functioning correctly and efficiently. After confirming that everything is operating smoothly, the project is officially released and made available for use.

And here is the flowchart for the workflow:



The workflow commences as users furnish parameters essential for the task. These parameters are subsequently analyzed during the assembly phase to determine the specific requirements. Following this analysis, the requisite modules are fetched and compiled into suitable code structures. Once the assembly process is complete, the finalized code is delivered to the users, enabling them to utilize the tailored solutions effectively.

4. Benefits

For the Polkadot Ecosystem

I. **Lowering Development Difficulty:** By reducing the complexity of developing on Polkadot, Magnet Stack encourages more developers to choose the Polkadot ecosystem, further promoting the creation of infrastructure and enriching the business environment within the ecosystem.

II. **Attracting Users:** Magnet Stack bolsters the Polkadot ecosystem by drawing in new projects and promoting their success, thereby increasing user activity and ecosystem vitality.

III. **Expanding DOT Demand:** Magnet Stack enhances the utility of DOT by broadening its applications, thereby contributing to a more balanced supply and demand dynamic for the token.

For Developers:

I. **Business Models around Coretime:** Magnet Stack offers developers business models focused on coretime, enabling them to effortlessly configure their coretime requirements and derive revenue from operating their appchains.

II. **Reducing Legal and Financial Risks:** By eliminating the necessity of issuing a native token, developers can reduce legal and financial risks and integrate more closely with the Polkadot ecosystem.

III. **Focusing on Business Development:** By lowering development difficulty and integrating important functionalities needed for appchains, Magnet Stack allows developers to concentrate on business module development.

For DOT Holders:

I. **Diverse Service Options:** The capability of Magnet Stack to rapidly onboard numerous appchains offers DOT holders more scenarios to utilize their tokens, thereby broadening the range of services available to them.

II. **Enhancing DOT Demand:** The increased number of appchains further boosts the demand for DOT, contributing to the appreciation of its value.

III. **Bringing Fresh Blood to the Ecosystem:** By attracting more projects and developers, Magnet Stack injects fresh blood into the Polkadot ecosystem, enhancing market liquidity and depth.

5. Magnet Stack Business Model

The business model of Magnet Stack is structured to cater to the following aspects: **Customized Requirements:** Magnet Stack aims to address the specific needs and preferences of its clients by offering tailor-made solutions. This involves closely collaborating with clients to understand their unique requirements and developing customized features, functionalities, and solutions that align with their business objectives.

Subsequent Maintenance and Upgrade Demands: In addition to providing initial customizations, Magnet Stack also commits to offering continuous support and maintenance services. This involves addressing any issues that arise post-implementation, providing technical assistance, and ensuring that the platform remains up-to-date with the latest technology trends. Furthermore, as clients' needs evolve over time, Magnet Stack remains flexible to accommodate future upgrades and enhancements to the platform.

Value-Added Services: Overall, Magnet Stack's business model is centered around delivering value-added services that go beyond basic product offerings. These services may include consulting, training, ongoing support, and assistance with implementation and integration. By providing comprehensive solutions and support, Magnet Stack aims to build long-term relationships with clients and generate revenue through the delivery of high-quality services.

6. Budget

The current application is for the development costs of Milestone 3, with a total development duration of 3660 person-hours at an hourly rate of $80, resulting in a total cost of $294,400 (the amount of DOT will be converted based on the EMA-7 price on the day of the official submission). The development content for this phase is expected to be completed within 4 months. The specific development breakdown of each milestone is as follows:

Milestone	Task	Notion	Hours	Cost
Milestone3	Maintenance	Maintain the continuous update and network stability of Magnet, update the coretime model with Rococo, and handle the emergency	600	48000
	Logic design and function analysis	Design business logic and detailed functionalities.	80	6400
	Component Definition for Modularity	For the detailed design of added functionalities, ensure they better conform to modularization requirements.	240	19200
	Integrate MoveVM	For Move contract	160	12800
	Integrate Lastic	Coretime secondary market	200	16000
	Integrate MultiSign and Proxy	Multisign Txs and Smart contract deployment	100	8000
	Upgrade Profit Control	Customize the desired profit margin to control the timing of coretime ordering, this function can't be active in the Bulk coretime model	360	28800
	Upgrade Profit Calculation	In the on-demand mode, calculations are based on the cost per single coretime, while in the Bulk mode, calculations are based on the average cost.	400	32000
	Upgrade Profit Distribution	Allows for the customization of the number and ratio of Profit distribution addresses, while also supporting cross-chain profit distribution via XCM.	380	30400

	Upgrade Block Assurance	In the On-demand mode, the waiting time can be dynamically reduced based on the current accumulation of Gas fees. Simultaneously, the trigger time for forced ordering can be adjusted.	300	24000
	Develop the user interface and the generator	Develop an intermediary business layer for Magnet Stack, enabling it to fetch specific codes based on user needs. Moreover, to facilitate user access, a frontend will be developed that encompasses all operational features of the Template.	420	33600
	Integration and Testing	Integrate all functionalities and test for optimization.	320	25600
	Technical Documentation Writing	Guidance Documentation	120	9600
Milestone 3 Total			**3660**	**294400**

7. Team Member

Acai (Xianzhe Liang) PBA alumni. holds a Bachelor's degree in Computer Science and a Master's degree in Management. He is the project leader responsible for project development planning, solution design, and development progress management. He previously worked at ASUS. https://www.linkedin.com/in/xianzhe-liang-849446282/

Kyle with a Master's degree in Applied Actuarial Science and a Postgraduate Diploma in Computer Science from the University of Kent. His experience working at Bangkok Bank provided him with a comprehensive understanding of crypto from both financial and technical perspectives, which became a cornerstone for his venture into blockchain project development in 2017. Since then, Kyle has played pivotal roles in product design and project management across various projects,

Daniel PBA alumni. Previously worked at SenseTime, Amber Group as a senior software engineer, with experience in backend development for AI model and crypto wallet design. Well-versed in Substrate, Golang, Rust, and Solidity.

Toints
Dev, serves as the Technical Lead and Core Engineer for the project, managing and executing its technical development. Previously, he held the role of Technology Lead at Mytoken and was also involved in maintaining a fork of Monero for Bitmain.

Vincent Yu holds a Master's degree in Electronic Information and works as a Development Engineer, responsible for the development of project functionality modules.

Zachary holds a Master's degree in Computer Science and works as a Development Engineer, responsible for the development of project functionality modules.

Alex formerly provided security software services for the traditional banking industry. Entered the blockchain industry in 2018. Proficient in blockchain application development and familiar with mainstream blockchain technologies, including Substrate. Skilled in development languages such as Rust.

LeeJ officially entered the blockchain industry in 2018. As a core developer on our team, he contributed to the development of the project's central functions. He is well-versed in Substrate, Cosmos, and Rust.