

Subscan - Data Dashboard Customization Toolset



Proposer	Subscan
Beneficiary	14RYaXRSqb9rPqMaAVp1UZW2czQ6dMNGMbvkwfifi6m8ZgZ
Date	28th November, 2022
Requested DOT	201,600 USD (EMA7 Price: 37,183.847 DOT)
Contact	yakio.liu@subscan.io

Table of Content

Summary	3
Context of the Proposal	4
Problem Statement	4
Proposed Solutions	6
Main Features	7
Editor	7
Parser	16
Management Tool	19
Future Plan	20
Payment Conditions & Budget	23
Editor	23
Parser	24
Management Tool	25
Developer Documentation & Tutorial	26
Total	27

Summary

This proposal aims to provide ecological users with an open-source toolset, which covers all processes of custom data dashboard and is suitable for mainstream front-end frameworks. Users can use it to conveniently generate, configure, and manage data widgets and inject them into any project that needs to display blockchain data.

The toolset contains three parts: **Editor**, **Parser**, and **Management Tools**. The editor is used to generate and configure data widgets. The parser is used to parse and display widgets. Management Tool is used for widgets management. They all follow the [License apache 2.0](#).

We provide it with a wealth of widgets (covering more than 90% of the [Subscan explorer](#) component library) and support commonly used data sources in the blockchain, such as HTTP API, GraphQL, State, LocalStorage, etc. Any developer (even users without a development background) can use it to customize blockchain data views easily.

1) Using this toolset, users can easily create a user-friendly explorer UI with any open-source explorer solution (such as [Subscan Essentials](#)).

2) Using this toolset, you can customize widgets with custom data sources, such as API / GraphQL([Subquery](#)), that can be integrated into your websites, Dapps, or any other web views.

In the future, Subscan will develop and deploy free online services based on this toolset. It will enable users to create and submit widgets to Subscan explorer. Our ideal goal is to create a Dune-like data visualization publishing and collaboration platform.

Context of the Proposal

[Subscan Essentials](#) has awarded the W3F grant in 2020, which is aimed at Substrate developers and provides solutions for Substrate based blockchain data indexing and parsing.

As the core code of subscan, it has been used by many explorers of the Substrate ecosystem and welcomed by the community.

During these two years, we have kept close contact with the developers of the Substrate ecosystem. While receiving applause, we also found many shortcomings. The community urgently needs a **developer-friendly, extensible** and **economical** explorer solution to help them grow better. Subscan hopes to solve this problem and provide better public products and infrastructure for the Substrate ecosystem.

Problem Statement

The blockchain explorers in the current Substrate ecosystem usually have the following problems:

1) Unfriendly to developers

In consideration of generality, the UI/UX design of many open source explorers (including Subscan Essentials) is kind of simple. To achieve a high-quality user experience, developers always have to invest more manpower in design and front-end, which means that for a network to obtain a user-friendly explorer, it will take 2-3 employees weeks, if not months, which is costly for a small team.

It is more meaningful for them to focus on the project rather than the user experience of the infrastructure. Therefore, a developer-friendly, efficient, and Low-Code explorer editing tool is necessary.

2) Lack of extensibility

As the explorer that integrates the most Substrate networks, Subscan has witnessed the development of the Substrate ecosystem. More and more projects are built based on

Substrate. They have made outstanding achievements in different professional directions, such as Defi, contract, privacy, storage, cross-chain bridges, etc. The core functions of these projects are not substrate-native modules, they are usually customized according to specific business requirements. The project team hopes that these functions can be integrated into explorer.

Unfortunately, there are many open source explorers in the Substrate ecosystem (including Subscan Essentials), which only provide basic functions, such as viewing blocks, accounts, extrinsics, events, etc., and lack in-depth support for custom modules. Moreover, there is a lack of flexible and configurable explorer frameworks in the ecosystem, so it will be difficult for developers to customize some functions for their own networks.

3) The cumbersome development process brings cost waste

To cover the growing development and operation costs of Subscan, we had to charge new networks for integration and feature customization. Compared with other ecological explorers' prices, although the price of Subscan is very cost-effective, it is still a problem for some start-up projects.

Especially for the development of custom functions, we need to communicate requirements, reorganize and learn the on-chain logic, back-end data indexing and parsing, front-end display, etc., which is a very long process. Therefore, every time we support a new module, we have to pay a lot of labor costs and time. At the same time, it also brings a lot of economic burden to consumers. We realize that in the long run, this is not a healthy development path - even if we increase employees, we cannot customize functions for 100 networks simultaneously. And as the project team(consumer) is most familiar with functions, they deserve to have a more economical way to get custom features.

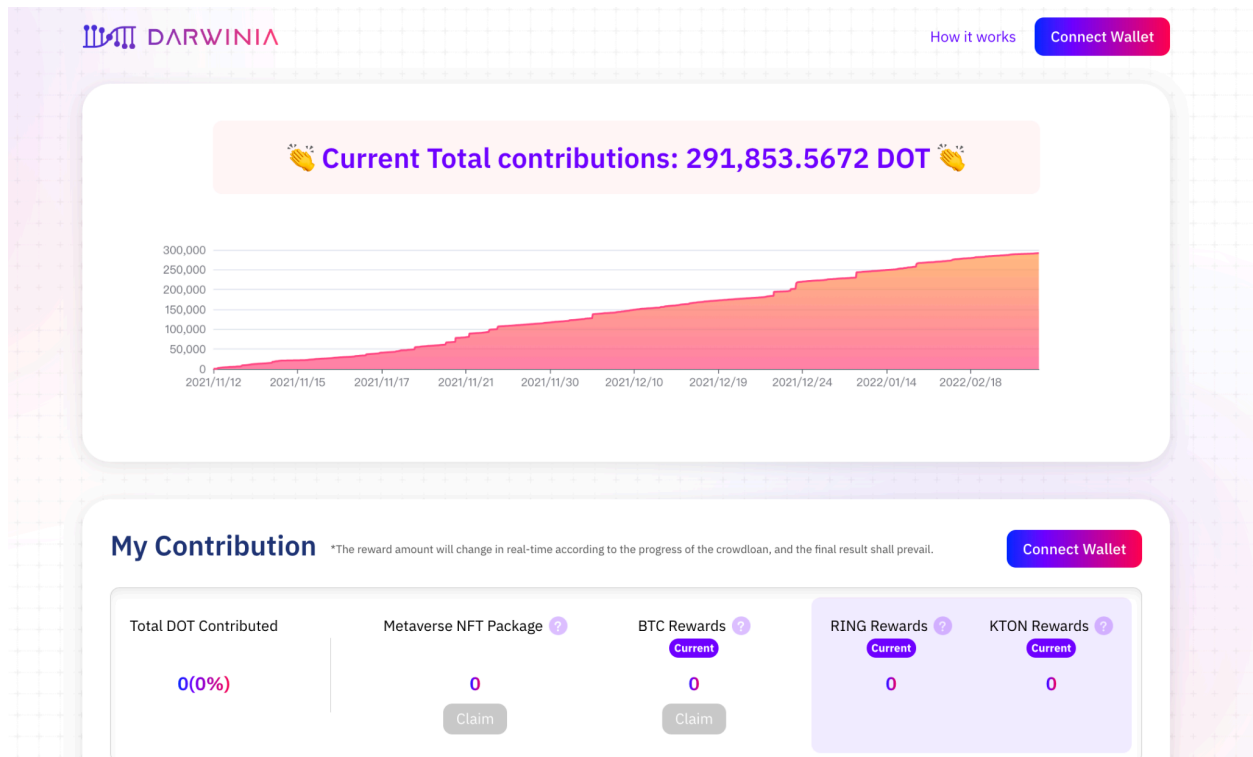
Proposed Solutions

There is an old saying in China that *“Give a man a fish, and you feed him for a day; teach a man to fish, and you feed him for a lifetime.”* (授人以鱼不如授人以渔). Since no explorer can meet the above requirements at the same time. To solve these problems, we propose an innovative solution, that is, to develop an open, extensible, developer-friendly, and economical tool - **Subscan Explorer Widget Low Code Editor & Parser**, and provide **online platform services**. By using this tool, users can build explorers and develop custom views more conveniently and efficiently. Around the management platform, all substrate developers can participate in the construction of a shared and collaborated blockchain data ecology.

Use Case:

- 1) Using this Toolset, users can easily create a user-friendly explorer UI with any open-source explorer solution (such as [Subscan Essentials](#)).
- 2) Using this Toolset, you can create custom data widgets with custom data sources such as API / GraphQL([Subquery](#)), which can be integrated into your websites, Dapps, or any other web views that need to display blockchain data.

E.g., This is a [Crowdloan Dashboard](#) on Darwinia's official website, and it took typically 2-3 weeks to develop. Using this toolset, it may take only 15 minutes to configure and generate the core data view widget.



Main Features

Toolset contains three parts: Editor, Parser, and Management Tools. The editor is used to generate and configure data views. The parser is used to parse and display views. Management Tool is used for view management. They all follow the [License apache 2.0](#).

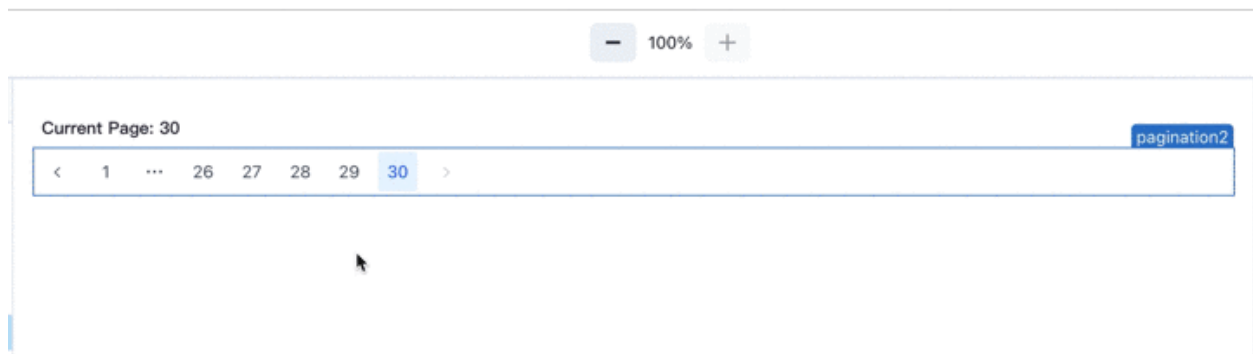
The Editor

The Editor is the interface to create, configure, and export Widgets. We have rich built-in widgets and support commonly used data sources, such as HTTP API, GraphQL, State, LocalStorage, etc. Any developer (even users without coding background) can use it through simple learning and easily customize the blockchain data view.

1) Data Binding

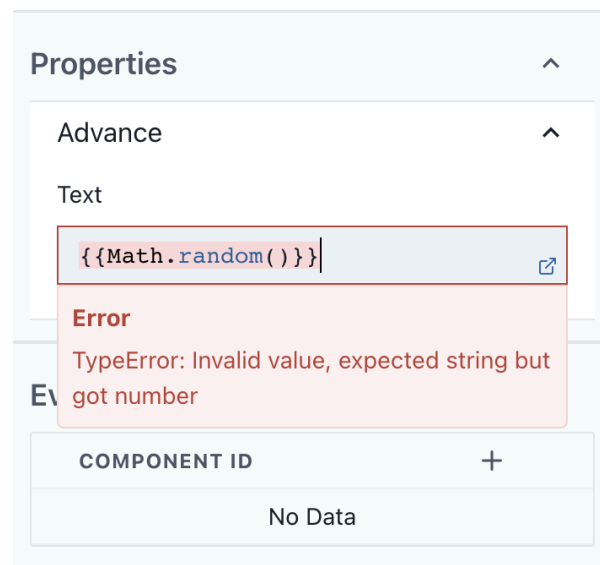
The Editor has a built-in responsive state structure to fulfill the interactive mode of "data-view" binding. When the state of a component changes, all components referencing the state will be updated automatically.

For example, if we want the label to render the currently selected page number by the pagination component, we only need to fill in `{{ pagination.currentPage.toString() }}` in the page number label display value. When clicking on a different page number, the page number label will display the value in real-time.



2) Type safety

Typescript has become the mainstream language that supports type verification for front-end developers. Type safety can improve the development experience and code quality. Typescript and JSON schema is widely applied in the Editor to achieve an excellent Type system. For example, if a component requires a Number type, but the provided value is a String, then the editor will throw an exception in real time to remind developers and users.

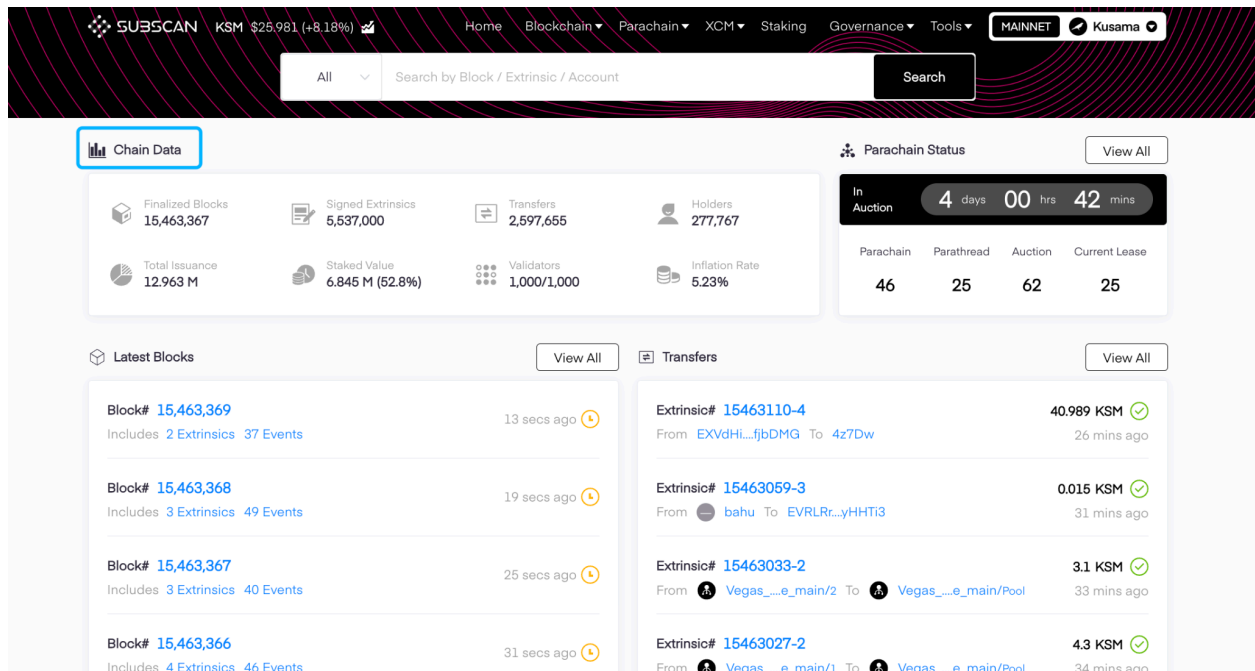


3) Excellent scalability



After sufficient market research, Subscan analyzed common blockchain data display scenarios and business requirements and selected and crafted a batch of built-in Widgets. These Widgets not only simply display data, but also offer excellent user experience. They have a wide range of usage scenarios. They can be used to build explorers conveniently and quickly, and are also suitable for any scenario where data needs to be displayed.

In this proposal, we will provide the following 9 components. More components will be released when they are ready. Good community-created widgets will be promoted and included in the built-in widget library.


- Text: Title, Heading1, Heading2, , Normal text (font, size, color), Links.











- Table

Validator	Self-Bonded	Total Bonded	Nominator	Commission	Grandpa Vote	Reward Point	Latest Mining
 Figment	0 DOT	3,838,706 DOT	28	10.00%	808	36,080	12214024
16eyBS...yCoieB	0 DOT	3,623,485 DOT	2	100.00%	116	49,560	12213891
12pgR5...jGwNHL	0 DOT	3,521,897 DOT	1	100.00%	1,680	24,320	12213972
1jZqir...2ufbyN	0 DOT	3,354,569 DOT	3	100.00%	15	13,280	12213933
14zy72...SiS3ti	0 DOT	3,311,822 DOT	1	100.00%	1,744	37,060	12214105
 Polkadot...ea lgar	660,716 DOT	3,305,019 DOT	587	1.00%	1,691	26,220	12213907

- Status data (Icon+Title+Data)



 Chain Data

 Finalized Blocks 12,214,175	 Signed Extrinsics 10,436,601	 Transfers 8,787,926	 Holders 1,064,126
 Total Issuance 1.235 B	 Staked Value 701.657 M (56.7%)	 Validators 297/297	 Inflation Rate 7.68%

- Detail Panel

Timestamp	2022-09-26 06:47:36 (+UTC)
Block Time	1 min ago
Block	 12214188
Life Time	immortal
Extrinsic Hash	0xc226ce86ba2737d31dcaa7b9de11590ae851580bc6a51d2b95b6fad153937eab 
Module	Balances
Call	Transfer 
Sender	 157yj3XhXKLia6nqqmk1mVk6RRWHCSUFHAtmxxgh2MVaDHz 
Destination	 15GADXLmZpfCDgVcPuLGCwLAWw3hV9UpwPHw9BJuZEKQREqB 
Value	 277.6930479527 DOT (\$1,738.44) 

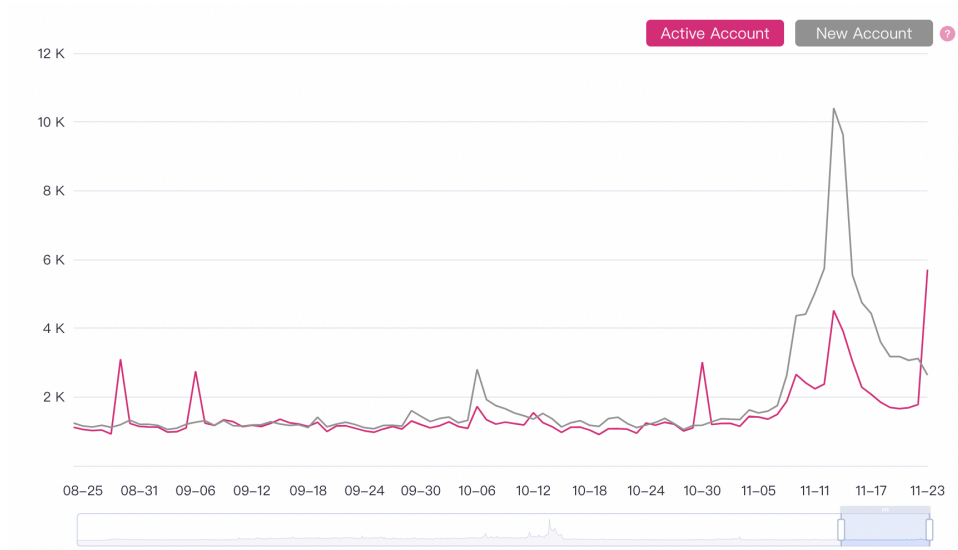
- Tab

Parachain (28) Parathread (13) Registered (19)			
Para ID	Fund ID	Lease Period	Owner
2086 ? Unknown	-	9-16	1nMg9m...S2HhuU
2058 ? Unknown	-	9-16	13DgtS...jfwcUH
2056 ? Unknown	2056-45	9-16	16Hqym...QHITgU
2055 ? Unknown	-	9-16	1jToXr...2CAxEJ
2052 ? Unknown	2052-43	9-16	15gPUA...8kXcZT
2046  Darwinia Parachain	-	9-16	 Darwinia Dev

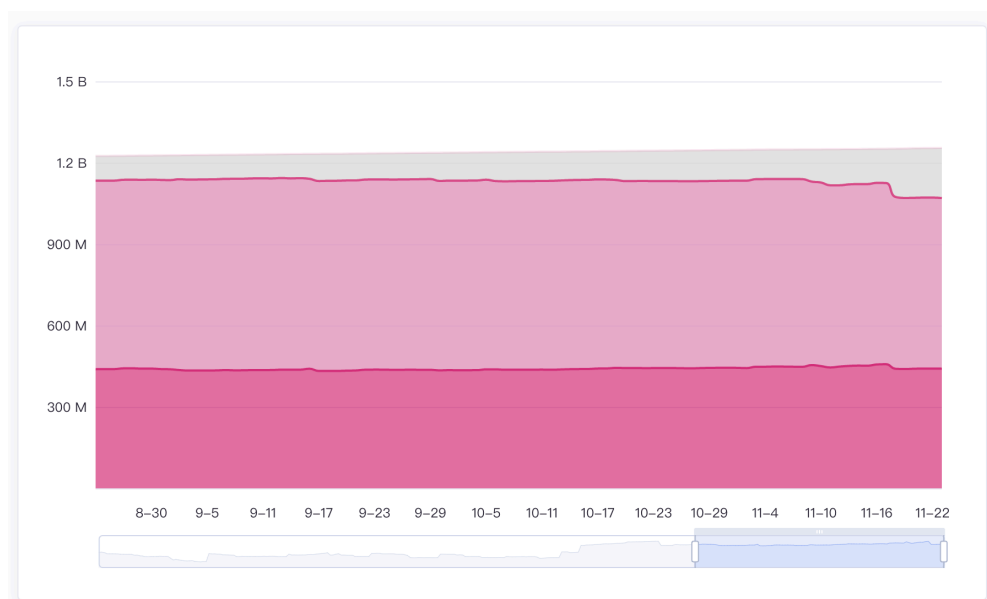
- Timeline

Status	Time	Extrinsic Index
● Reserve Para ID#2013	325 days 11 hrs ago	7560773-3
● Register Para ID#2013	324 days 15 hrs ago	7572269-35
● Create Crowdloan#2013-4	324 days 11 hrs ago	7574790-8
● Participate in Auction#1	318 days 13 hrs ago	7658910-0
● Participate in Auction#2	311 days 13 hrs ago	7759710-0
● Participate in Auction#3	304 days 12 hrs ago	7860510-0
● Participate in Auction#4	297 days 12 hrs ago	7961310-0

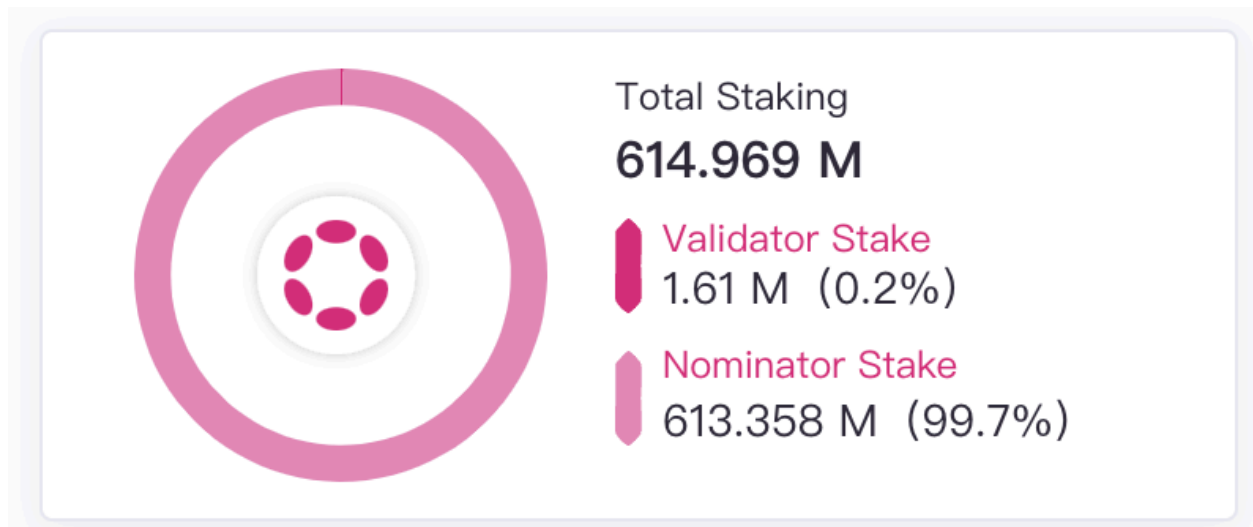
- Line chart (single & multiple)



- Stacked chart



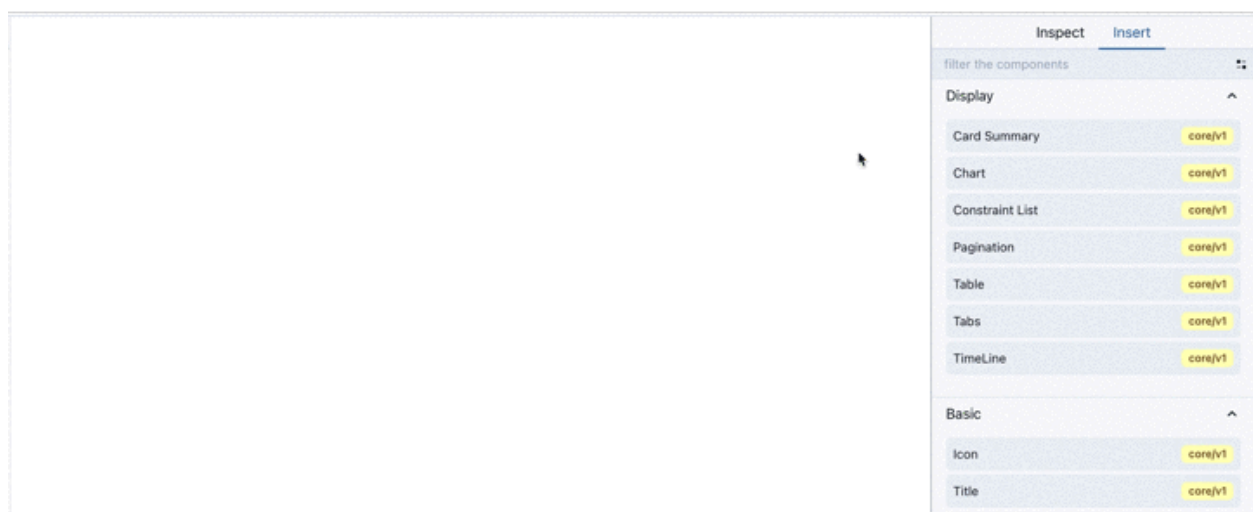
- Circular ratio chart



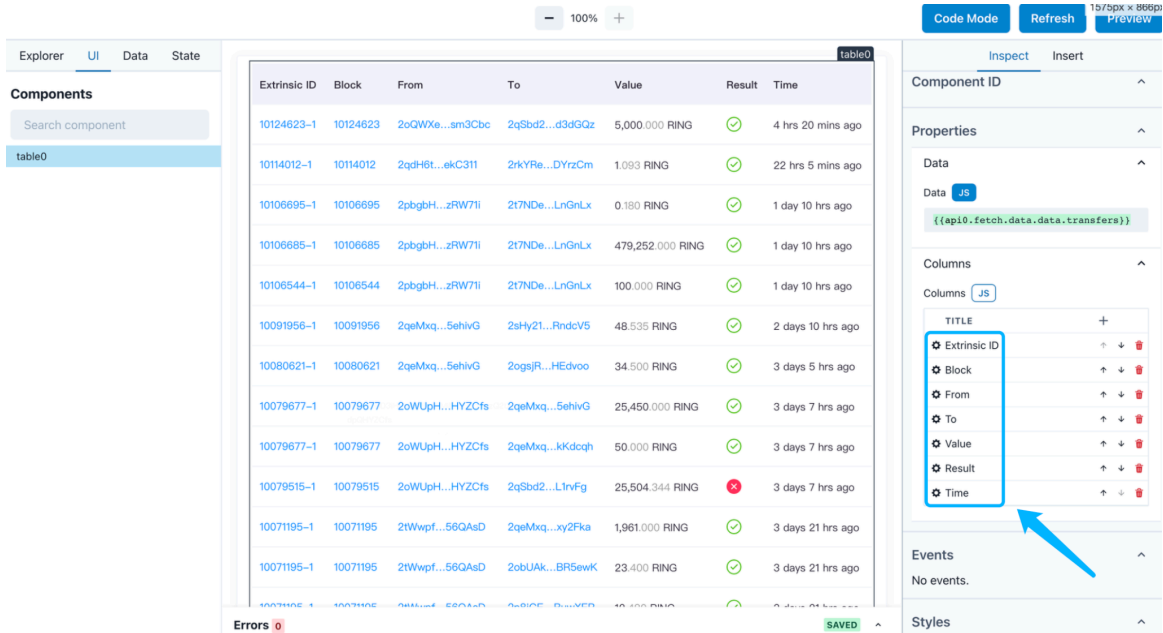
Use Case :

The following takes the Table as an example to show the configuration process in a real-world environment:

Step1: Drag the Table component into the editing canvas:



Step2: Configure the Table in inspect tab



100% +

Code Mode Refresh 15/3px x 888px PREVIEW

Explorer UI Data State

Components

Search component

table0

Extrinsic ID	Block	From	To	Value	Result	Time
10124623-1	10124623	2oQWxe...sm3Cbc	2qSbd2...d3dGQz	5,000.000 RING	✓	4 hrs 20 mins ago
10114012-1	10114012	2qstH6t...ekC3t1	2rkYRe...DYrzCm	1.093 RING	✓	22 hrs 5 mins ago
10106695-1	10106695	2pbgbH...zRW7t1	2t7NDe...LnGnLx	0.180 RING	✓	1 day 10 hrs ago
10106685-1	10106685	2pbgbH...zRW7t1	2t7NDe...LnGnLx	479,252.000 RING	✓	1 day 10 hrs ago
10106544-1	10106544	2pbgbH...zRW7t1	2t7NDe...LnGnLx	100.000 RING	✓	1 day 10 hrs ago
10091956-1	10091956	2qeMxq...5ehivG	2sHy21...RndcV5	48.535 RING	✓	2 days 10 hrs ago
10080621-1	10080621	2qeMxq...5ehivG	2ogsjR...HEdvo	34.500 RING	✓	3 days 5 hrs ago
10079677-1	10079677	2oWUpH...HYZCfs	2qeMxq...5ehivG	25,450.000 RING	✓	3 days 7 hrs ago
10079677-1	10079677	2oWUpH...HYZCfs	2qeMxq...kKdcqh	50.000 RING	✓	3 days 7 hrs ago
10079515-1	10079515	2oWUpH...HYZCfs	2qSbd2...L1rvFg	25,504.344 RING	✗	3 days 7 hrs ago
10071195-1	10071195	2tWwpl...56QAsD	2qeMxq...xy2Fka	1,961.000 RING	✓	3 days 21 hrs ago
10071195-1	10071195	2tWwpl...56QAsD	2obUAK...BR5ewK	23.400 RING	✓	3 days 21 hrs ago

Errors 0

Inspect Insert

Component ID

Properties

Data

Data JS

```

{{api0.fetch.data.data.transfers}}

```

Columns

Columns JS

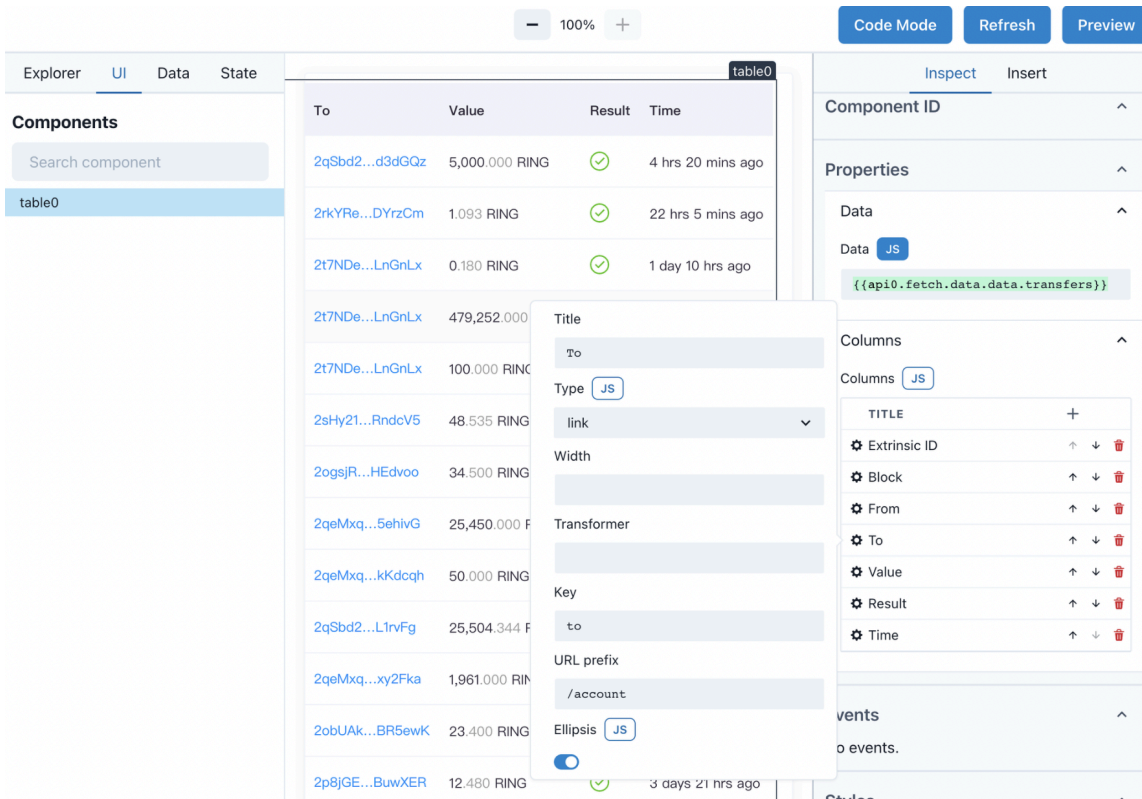
TITLE

- Extrinsic ID
- Block
- From
- To
- Value
- Result
- Time

Events

No events.

Styles



100% +

Code Mode Refresh Preview

Explorer UI Data State

Components

Search component

table0

To	Value	Result	Time
2qSbd2...d3dGQz	5,000.000 RING	✓	4 hrs 20 mins ago
2rkYRe...DYrzCm	1.093 RING	✓	22 hrs 5 mins ago
2t7NDe...LnGnLx	0.180 RING	✓	1 day 10 hrs ago
2t7NDe...LnGnLx	479,252.000 RING	✓	1 day 10 hrs ago
2t7NDe...LnGnLx	100.000 RING	✓	1 day 10 hrs ago
2sHy21...RndcV5	48.535 RING	✓	2 days 10 hrs ago
2ogsjR...HEdvo	34.500 RING	✓	3 days 5 hrs ago
2qeMxq...5ehivG	25,450.000 RING	✓	3 days 7 hrs ago
2qeMxq...kKdcqh	50.000 RING	✓	3 days 7 hrs ago
2qSbd2...L1rvFg	25,504.344 RING	✗	3 days 7 hrs ago
2qeMxq...xy2Fka	1,961.000 RING	✓	3 days 21 hrs ago
2obUAK...BR5ewK	23.400 RING	✓	3 days 21 hrs ago
2p8JGE...BuwXER	12.480 RING	✓	3 days 21 hrs ago

Errors 0

Inspect Insert

Component ID

Properties

Data

Data JS

```

{{api0.fetch.data.data.transfers}}

```

Columns

Columns JS

TITLE

- Extrinsic ID
- Block
- From
- To
- Value
- Result
- Time

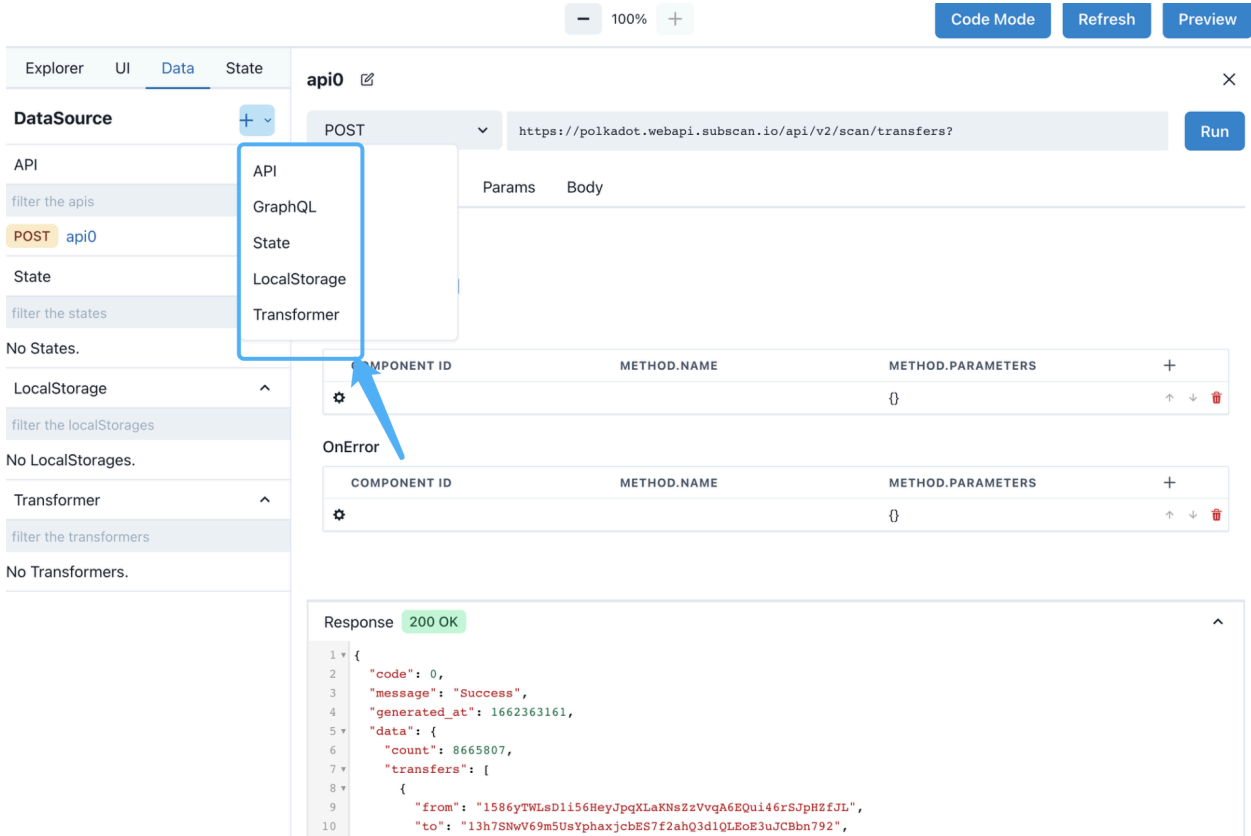
Events

No events.

Styles

4) Support multiple data sources support

Editor supports common data sources, such as HTTP API, GraphQL, State, LocalStorage, etc.



The screenshot shows the Subscan Editor interface. The 'Data' tab is selected, and a dropdown menu is open, listing the following data sources: API, GraphQL, State, LocalStorage, and Transformer. The 'API' option is highlighted. The main editor area shows a POST request to the URL `https://polkadot.webapi.subscan.io/api/v2/scan/transfers?`. The response is a 200 OK status with the following JSON data:

```

1 {
2   "code": 0,
3   "message": "Success",
4   "generated_at": 1662363161,
5   "data": {
6     "count": 8665807,
7     "transfers": [
8       {
9         "from": "1586yTWLeD1i56HeyJpqXLakNsZzVvqA6EQui46rSjpHzfJL",
10        "to": "13h7SNwV69m5UsYphaxjcbES7f2ahQ3d1QLEoE3uJCBbn792",

```

The Parser

The Parser is like a video player, and JSON data is the video source file. Subscan widget data is in flat JSON format to avoid deep nesting. You can use the Editor to generate and export JSON data for the Parser to quickly render Widgets on your own site.

1) Adapt to mainstream front-end frameworks

If your site is based on react, we provide a React version SDK, you can use it like a normal

component. If your app is developed based on Vue, there is also an almost perfect alternative solution.

React SDK:

```
import { Preview } from "testlowcode";

export const ParserDemo = ({ widgetJsonData }: { widgetJsonData: string }) => {
  | return <Preview options={JSON.parse(widgetJsonData)} />;
};
```

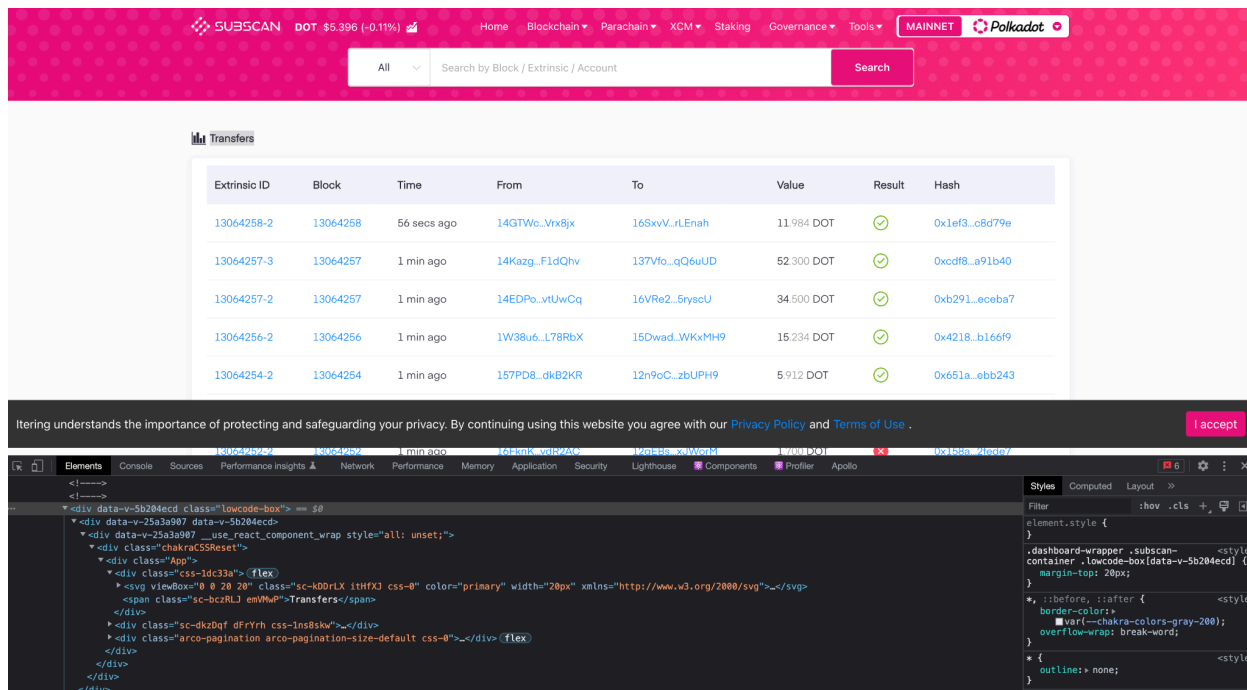
Vue:

```
<script>
import { mapState } from 'vuex';
import { Preview } from 'testlowcode';
import { applyReactInVue } from 'vuereact-combined';

export default {
  | computed: {
  |   | ...mapState({
  |     | darkmode: (state) => state.global.darkmode === 'on',
  |   }),
  | },
  | props: {
  |   | widgetJsonData: String,
  | },
  | data() {
  |   | return {
  |     | json: JSON.parse(this.widgetJsonData),
  |   };
  | },
  | components: {
  |   | Preview: applyReactInVue(Preview),
  | },
};
</script>

<template>
  | <div v-if="json">
  |   | <Preview :options="json" :isDark="darkmode" />
  | </div>
</template>
```

Example of integration in a Vue application:



The screenshot displays the SUBSCAN website interface. At the top, there's a navigation bar with the SUBSCAN logo, DOT price (\$5.396), and various menu items like Home, Blockchain, Parachain, XCM, Staking, Governance, and Tools. A search bar is also present. Below the navigation bar, a table titled "Transfers" lists recent transactions. The table has columns for Extrinsic ID, Block, Time, From, To, Value, Result, and Hash. Below the table, there's a dark banner with a privacy notice and an "I accept" button. At the bottom, a browser developer console is open, showing the HTML structure of the page, including a `<div data-v-5b204ecd class="lowcode-box">` element.

Extrinsic ID	Block	Time	From	To	Value	Result	Hash
13064258-2	13064258	56 secs ago	14GTWc_Vrx8jx	16SxvV_rLEnah	11.984 DOT	✓	0x1ef3_c8d79e
13064257-3	13064257	1 min ago	14Kazg_F1dQhv	137Vfo_qQ6uUD	52.300 DOT	✓	0xcdf8_a91b40
13064257-2	13064257	1 min ago	14EDPo_vtUwCq	16VRe2_5ryscU	34.500 DOT	✓	0xb291_eceba7
13064256-2	13064256	1 min ago	1W38u6_L78RbX	16Dwad_WKxMH9	15.234 DOT	✓	0x4218_b166f9
13064254-2	13064254	1 min ago	157PD8_dkB2KR	12n9oC_zbUPH9	5.912 DOT	✓	0x651a_ebb243

2) Backward compatibility

Each Widget JSON data contains metadata, which is used to record the version number, name, description, etc. Through these metadata, the Parser can accurately obtain the parsing rules of the Widget and display the Widget of the corresponding version. In the latest version of the Parser, it can perfectly adapt to the widget of the historical version.

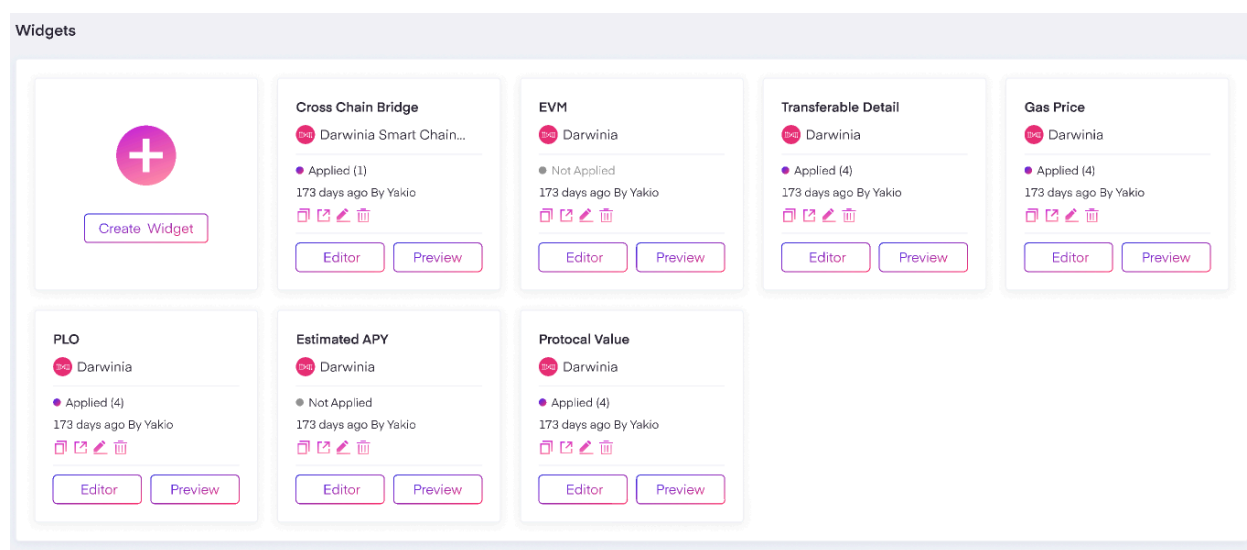
3) Customize

Dark Mode: Dark mode is a display mode with high contrast or inverse color mode. This mode is becoming more and more popular among users. It is believed to reduce eye strain and make reading easier than traditional black-on-white text.

Theme style: The tool provides a set of outstanding default styles. On this basis, the Parser provides an interface so the site can personalize the color theme of the widget, making it more colorful and better reflecting the brand style.

Management Tool

A local management tool is necessary for each user. We think downloading every widget and editing original files is not user-friendly.



1) Widget Storage

Users can view all widgets created or edited in their local history in the Widget Management tool. It will greatly reduce the obstacles caused by repeated import and export and file management. Here you can also quickly create new Widgets or delete unnecessary Widgets.

2) Widget Management

Users can perform operations such as duplicating, exporting, editing information and content, deleting, and previewing their widgets.

Duplicate function: create a copy of Widget

Export function: download the source code file of Widget, which is convenient for Widget to communicate and share among different users, or apply the code to your own website.

Edit information: used to mark your own Widget(Network, function name)

Editor: enter the editor state to edit the Widget

Preview: Create a temporary link to view the display of the Widget on the web page and adjust the page size to observe the adaptation of various screen sizes.

Future Plan

We are very optimistic about the future of this toolset and have many plans that we would like to share with the community, although these are not included in this proposal.

1) Open Platform Integration

A hosted version of the management tools will be integrated into the Subscan Open Platform. Web3 users can access it without deployment headaches.

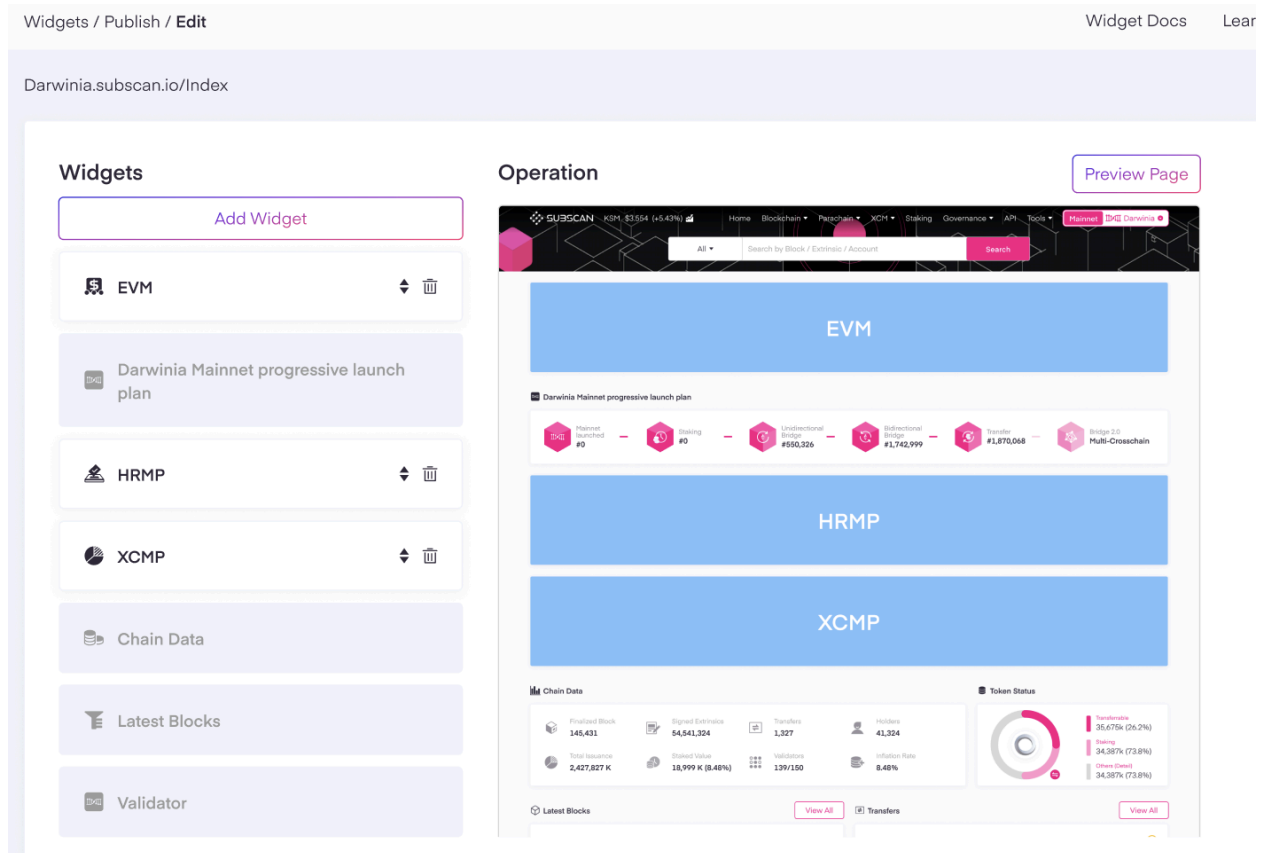
2) Widget Management

Users can create, delete, duplicate, share, import, export, preview, etc., on the platform.

Since different users in the same organization have different data research focuses, Widget Management provides a collaboration platform. Users can share Widgets with others in the same organization, and they can view or duplicate Widgets from other users in the same organization. It will significantly enhance the user's ability to create visual data.

3) Subscan Integration

We strongly encourage users to share their data visualization with other partners in the ecosystem. Therefore, Subscan explorer will integrate widgets provided by users.



After the user publishes widgets to the Subscan or other websites, there is also a version control system to facilitate the user to view the historical version, version backtracking, update, etc.

4) Security

To ensure the safety and reliability of Widgets published by users, we will also provide a backstage management platform for auditing the security of Widgets.

It can avoid the possibility of some malicious attacks and reduce the risk to the entire system. We will continue to conduct security audits on the Widgets submitted by users and feedback suggestions to users.

5) More Widgets

We will continue to add more Widget components to enrich the way user data is presented.

In the future, we will allow users to customize their own Widgets vision to enhance the adaptability of Widgets.

6) Data Visualization Platform

In the future when more users are familiar with the Widget editor, we will develop a Dune-like data visualization publishing and collaboration platform. On this platform, all community users can publish their own analysis reports with widgets. The platform will provide various points of view for all ecological participants.

Payment Conditions & Budget

Editor

1) Task list:

- ☒ Market research, user demand research
- ☒ Feasibility Analysis / Prototyping
- ☒ UI/UX Design
- ☒ Data and protocol standard formulation
- ☒ Data component standard formulation
- ☒ Framework design and development scaffolding construction: typescript, eslint, store, and common hooks required for front-end engineering
- ☒ Adapt to common front-end frameworks such as React and Vue
- ☒ Common components development: Text, Table, Status data, Tab, Timeline, Charts
- ☒ Import/export widgets configuration
- ☒ Support data source GraphQL API
- ☒ Support data source Subscan API
- ☒ Type safety validation and exception reminder
- ☒ Safety check for user input in editor
- ☒ Testing and bug fix

2) Timeline:

9.5 Weeks: November 1, 2022 - January 6, 2023

3) Detail:

Position	Workload (Day)	Daily salary (USD)	Amount	Budget (USD)
Front-end developer	45	600	3	81,000
UI/UX designer	18	500	1	9,000
Product manage	22	500	1	11,000
Test Engineer	19	450	1	8,550
Project manager	18	500	1	9,000

Subtotal: \$118,550

Parser

1) Task List :

- ☒ Widget JSON data format validation
- ☒ Adapt to common front-end frameworks such as React and Vue
- ☒ Build widget viewer
- ☒ UI/UX Design
- ☒ Support dark mode
- ☒ Support custom theme style
- ☒ Testing and bug fix

2) Time Line:

5 Weeks: January 2, 2023 - February 3, 2023

3)Detail:

Position	Workload (Day)	Daily salary (USD)	Amount	Budget (USD)
Front-end developer	18	600	3	32,400
Product manager	8	500	1	4,000
UI/UX designer	6	500	1	3,000
Test Engineer	10	450	1	4,500
Project manager	12	500	1	6,000

Subtotal: \$49,900

Management Tool

1) Task list:

- ☒ Management platform business research
- ☒ Feasibility Analysis / Prototyping
- ☒ Framework design and development scaffolding construction: typescript, eslint, store, and common hooks required for front-end engineering
- ☒ Design file for management tool
- ☒ Widgets management
- ☒ Adaptation with Editor
- ☒ Testing and bug fix

2) Timeline:

3 Weeks: January 25, 2023 - February 20, 2023

3)Detail:

Position	Workload (Day)	Daily salary (USD)	Amount	Budget (USD)
Front-end developer	14	600	2	16,800
UI/UX designer	5	500	1	2,500
Product manager	4	500	1	2,000
Test Engineer	5	450	1	2,250

Subtotal: \$23,550

Developer Documentation & Tutorial

1) Task List:

- ☒ Editor developer documentation
- ☒ Parser developer documentation
- ☒ Management tool developer documentation
- ☒ Editor user tutorial
- ☒ Management platform user tutorial
- ☒ Editor video tutorial
- ☒ Management tool video tutorial

2) Timeline:

2 Weeks: February 15, 2023 - February 28, 2023

3)Detail:

Position	Workload (Day)	Daily salary (USD)	Amount	Budget (USD)
Front-end developer	6	600	1	3,600
UI/UX designer	4	500	1	2,000
Product manager	8	500	1	4,000

Subtotal: \$9,600

Total

The total amount above is **\$201,600**, and the amount of DOT will be converted based on the **EMA7** price on the day of the official submission.

2022-12-08 01:55:42 (+UTC), Block #13260191

DOT EMA7 Price (USD): 5.421;

Number of DOT: 37,183.847

[https://polkadot.subscan.io/tools/price_converter?value=201600&type=time&from=USD
&to=DOT&time=1670464800](https://polkadot.subscan.io/tools/price_converter?value=201600&type=time&from=USD&to=DOT&time=1670464800)