Snapshot Week 10 of Group InfluxUI-PG01

No-Code Solution for InfluxDB

by

Xiaoyue Rao a1819070

Jianghao Jin a1880849

Tiantian Wang a1894037

Manhong Chen a1904387

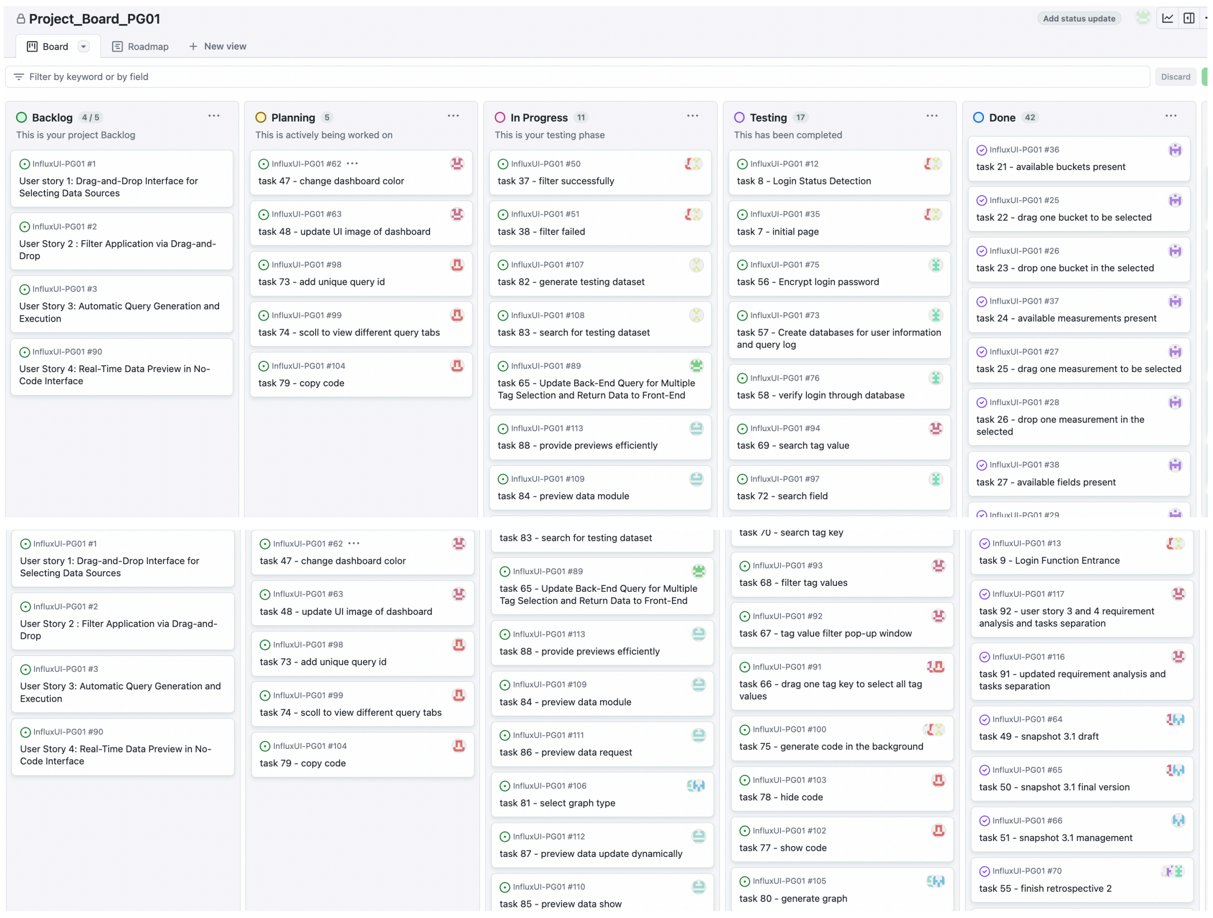
Ziyan Zhao a1883303

Ling Luo a1847162

Yufei Wang a1897360

Yinkai Yuan a1909218

1. Product Backlog and Task Board



Project_Board_PG01

Add status update

Board · Roadmap · + New view

Filter by keyword or by field

Discard

Backlog 4 / 5
This is your project Backlog

InfluxUI-PG01 #1
User story 1: Drag-and-Drop Interface for Selecting Data Sources

InfluxUI-PG01 #2
User Story 2 : Filter Application via Drag-and-Drop

InfluxUI-PG01 #3
User Story 3: Automatic Query Generation and Execution

InfluxUI-PG01 #90
User Story 4: Real-Time Data Preview in No-Code Interface

Planning 5
This is actively being worked on

InfluxUI-PG01 #62
task 47 - change dashboard color

InfluxUI-PG01 #63
task 48 - update UI image of dashboard

InfluxUI-PG01 #98
task 73 - add unique query id

InfluxUI-PG01 #99
task 74 - scoll to view different query tabs

InfluxUI-PG01 #104
task 79 - copy code

In Progress 11
This is your testing phase

InfluxUI-PG01 #50
task 37 - filter successfully

InfluxUI-PG01 #51
task 38 - filter failed

InfluxUI-PG01 #107
task 82 - generate testing dataset

InfluxUI-PG01 #108
task 83 - search for testing dataset

InfluxUI-PG01 #89
task 65 - Update Back-End Query for Multiple Tag Selection and Return Data to Front-End

InfluxUI-PG01 #113
task 88 - provide previews efficiently

InfluxUI-PG01 #109
task 84 - preview data module

Testing 17
This has been completed

InfluxUI-PG01 #12
task 8 - Login Status Detection

InfluxUI-PG01 #35
task 7 - initial page

InfluxUI-PG01 #75
task 56 - Encrypt login password

InfluxUI-PG01 #73
task 57 - Create databases for user information and query log

InfluxUI-PG01 #76
task 58 - verify login through database

InfluxUI-PG01 #94
task 69 - search tag value

InfluxUI-PG01 #97
task 72 - search field

Done 42

InfluxUI-PG01 #36
task 21 - available buckets present

InfluxUI-PG01 #25
task 22 - drag one bucket to be selected

InfluxUI-PG01 #26
task 23 - drop one bucket in the selected

InfluxUI-PG01 #37
task 24 - available measurements present

InfluxUI-PG01 #27
task 25 - drag one measurement to be selected

InfluxUI-PG01 #28
task 26 - drop one measurement in the selected

InfluxUI-PG01 #38
task 27 - available fields present

InfluxUI-PG01 #29

InfluxUI-PG01 #1
User story 1: Drag-and-Drop Interface for Selecting Data Sources

InfluxUI-PG01 #2
User Story 2 : Filter Application via Drag-and-Drop

InfluxUI-PG01 #3
User Story 3: Automatic Query Generation and Execution

InfluxUI-PG01 #90
User Story 4: Real-Time Data Preview in No-Code Interface

InfluxUI-PG01 #62
task 47 - change dashboard color

InfluxUI-PG01 #63
task 48 - update UI image of dashboard

InfluxUI-PG01 #98
task 73 - add unique query id

InfluxUI-PG01 #99
task 74 - scoll to view different query tabs

InfluxUI-PG01 #104
task 79 - copy code

task 83 - search for testing dataset

InfluxUI-PG01 #89
task 65 - Update Back-End Query for Multiple Tag Selection and Return Data to Front-End

InfluxUI-PG01 #113
task 88 - provide previews efficiently

InfluxUI-PG01 #109
task 84 - preview data module

InfluxUI-PG01 #111
task 86 - preview data request

InfluxUI-PG01 #106
task 81 - select graph type

InfluxUI-PG01 #112
task 87 - preview data update dynamically

InfluxUI-PG01 #110
task 85 - preview data show

task 70 - search tag key

InfluxUI-PG01 #93
task 68 - filter tag values

InfluxUI-PG01 #92
task 67 - tag value filter pop-up window

InfluxUI-PG01 #91
task 66 - drag one tag key to select all tag values

InfluxUI-PG01 #100
task 75 - generate code in the background

InfluxUI-PG01 #103
task 78 - hide code

InfluxUI-PG01 #102
task 77 - show code

InfluxUI-PG01 #105
task 80 - generate graph

InfluxUI-PG01 #13
task 9 - Login Function Entrance

InfluxUI-PG01 #117
task 92 - user story 3 and 4 requirement analysis and tasks separation

InfluxUI-PG01 #116
task 91 - updated requirement analysis and tasks separation

InfluxUI-PG01 #64
task 49 - snapshot 3.1 draft

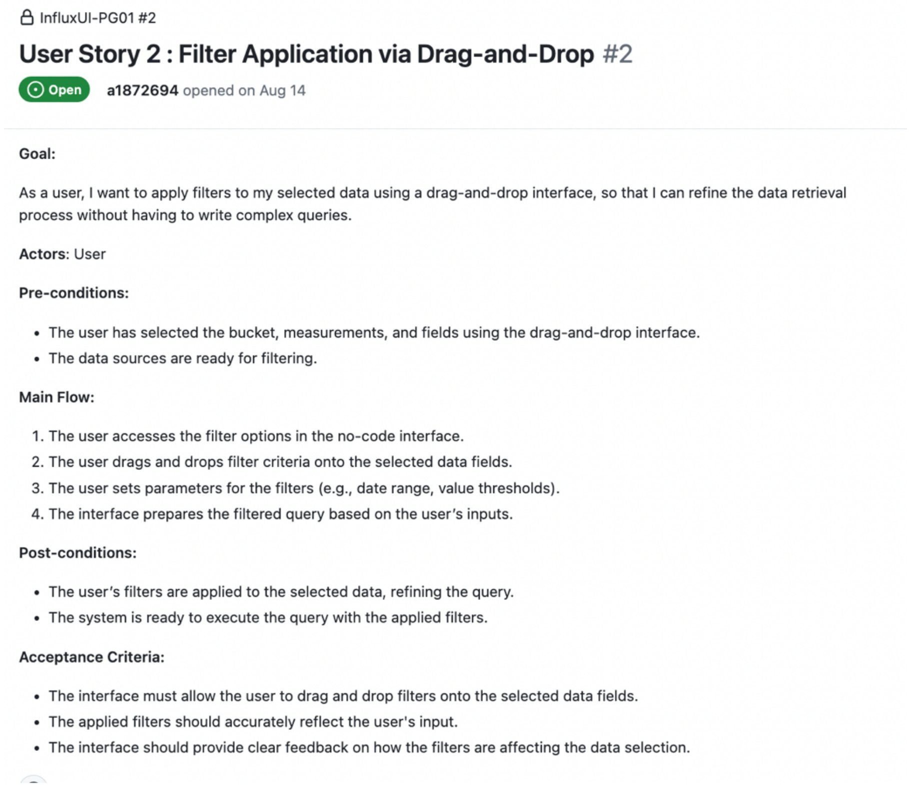
InfluxUI-PG01 #65
task 50 - snapshot 3.1 final version

InfluxUI-PG01 #66
task 51 - snapshot 3.1 management

InfluxUI-PG01 #70
task 55 - finish retrospective 2

Fig.1 The Backlog of Project

2. Sprint Backlog and User Stories



InfluxUI-PG01 #2

User Story 2 : Filter Application via Drag-and-Drop #2

Open **a1872694** opened on Aug 14

Goal:

As a user, I want to apply filters to my selected data using a drag-and-drop interface, so that I can refine the data retrieval process without having to write complex queries.

Actors: User

Pre-conditions:

- The user has selected the bucket, measurements, and fields using the drag-and-drop interface.
- The data sources are ready for filtering.

Main Flow:

1. The user accesses the filter options in the no-code interface.
2. The user drags and drops filter criteria onto the selected data fields.
3. The user sets parameters for the filters (e.g., date range, value thresholds).
4. The interface prepares the filtered query based on the user's inputs.

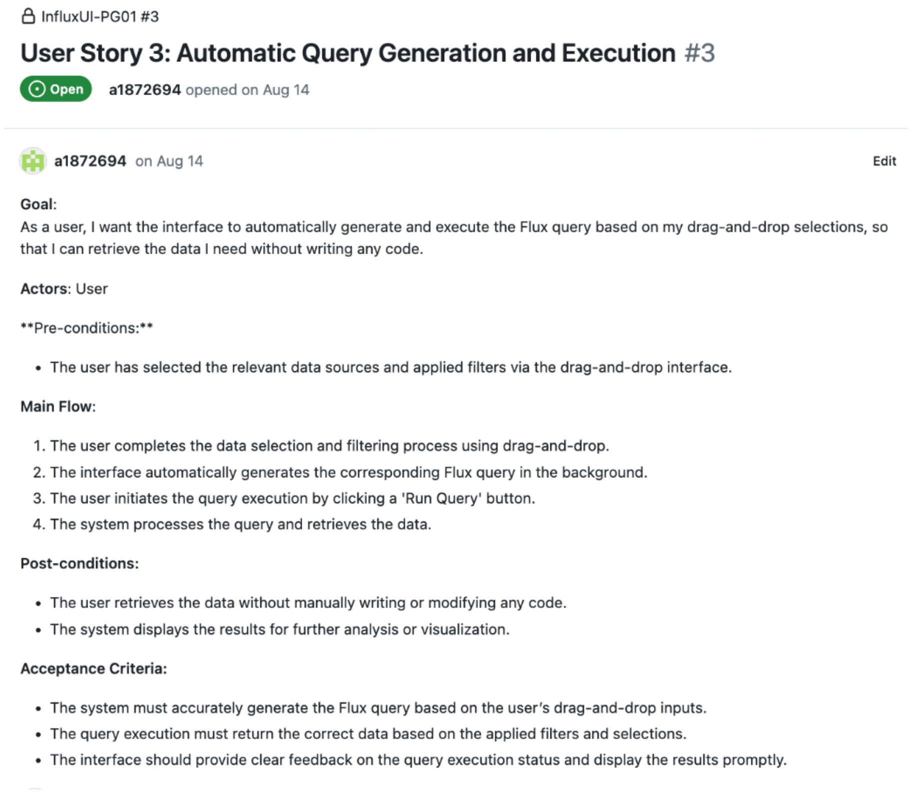
Post-conditions:

- The user's filters are applied to the selected data, refining the query.
- The system is ready to execute the query with the applied filters.

Acceptance Criteria:

- The interface must allow the user to drag and drop filters onto the selected data fields.
- The applied filters should accurately reflect the user's input.
- The interface should provide clear feedback on how the filters are affecting the data selection.

Fig. 2 User Story 2



InfluxUI-PG01 #3

User Story 3: Automatic Query Generation and Execution #3

Open **a1872694** opened on Aug 14

a1872694 on Aug 14 Edit

Goal:
As a user, I want the interface to automatically generate and execute the Flux query based on my drag-and-drop selections, so that I can retrieve the data I need without writing any code.

Actors: User

Pre-conditions:

- The user has selected the relevant data sources and applied filters via the drag-and-drop interface.

Main Flow:

1. The user completes the data selection and filtering process using drag-and-drop.
2. The interface automatically generates the corresponding Flux query in the background.
3. The user initiates the query execution by clicking a 'Run Query' button.
4. The system processes the query and retrieves the data.

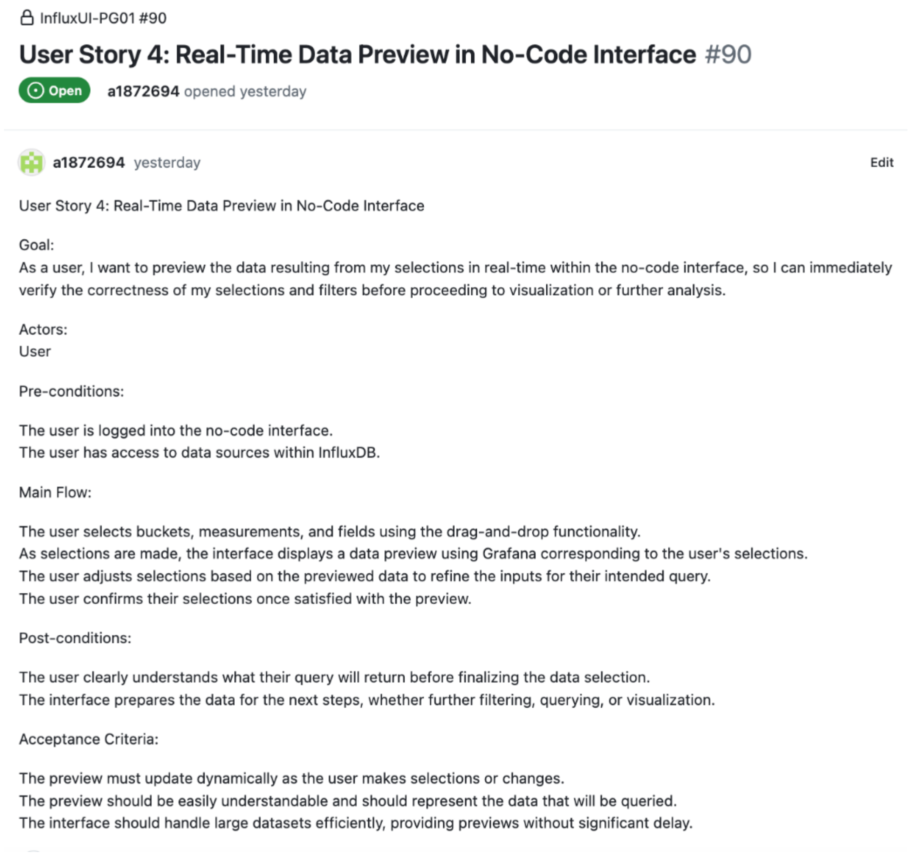
Post-conditions:

- The user retrieves the data without manually writing or modifying any code.
- The system displays the results for further analysis or visualization.

Acceptance Criteria:

- The system must accurately generate the Flux query based on the user's drag-and-drop inputs.
- The query execution must return the correct data based on the applied filters and selections.
- The interface should provide clear feedback on the query execution status and display the results promptly.

Fig. 3 User Story 3


InfluxUI-PG01 #90

User Story 4: Real-Time Data Preview in No-Code Interface #90

Open a1872694 opened yesterday

a1872694 yesterday Edit

User Story 4: Real-Time Data Preview in No-Code Interface

Goal:
As a user, I want to preview the data resulting from my selections in real-time within the no-code interface, so I can immediately verify the correctness of my selections and filters before proceeding to visualization or further analysis.

Actors:
User

Pre-conditions:

The user is logged into the no-code interface.
The user has access to data sources within InfluxDB.

Main Flow:

The user selects buckets, measurements, and fields using the drag-and-drop functionality.
As selections are made, the interface displays a data preview using Grafana corresponding to the user's selections.
The user adjusts selections based on the previewed data to refine the inputs for their intended query.
The user confirms their selections once satisfied with the preview.

Post-conditions:

The user clearly understands what their query will return before finalizing the data selection.
The interface prepares the data for the next steps, whether further filtering, querying, or visualization.

Acceptance Criteria:

The preview must update dynamically as the user makes selections or changes.
The preview should be easily understandable and should represent the data that will be queried.
The interface should handle large datasets efficiently, providing previews without significant delay.

Fig. 4 User Story 4

2.1 Brief description of the user stories selected

In this sprint, our group focuses on user story 2, user story 3 and user story 4. As a user in user story 2, they want to apply filters to their selected data using a drag-and-drop interface, so that they can refine the data retrieval process without having to write complex queries as a user. As a user in user story 3, they want the interface to automatically generate and execute the Flux query based on their drag-and-drop selections, so that they can retrieve the data they need without writing any code. Finally, in user story 4, as a user, they want to preview the data resulting from their selections in real-time within the no-code interface, so they can immediately verify the correctness of their selections and filters before proceeding to visualization or further analysis.

With the discussion in sprint 3 review meeting and sprint 4 planning meeting, we found that the understanding between client and our group is a little bit different, and there are more details about the filtering function shown to us, so we decide to update what we have developed for filtering to meet the client's requirements. In addition, to keep up with our development schedule, we would like to work on user story 3 and 4.

3. Definition of Done

Definition of Done		
Items No.	Modules	checklist
1	Updated Filtering	For tags, user can drag one tag key to select all tag values for that key
2		For tags, user can use filter to select one or more tag values at the same time
3		For tags, user can search the key in the drag-box
4		For fields, user can search the field name in the drag-box
5		As filtering tag values, user can search the values in a pop-up window
6		For measurements, user can search the measurement name in the drag-box
7	Updated Query	For each query tab, it will have its unique query id
8		When there are too many query tabs, user can scroll the mouse to view all of them
9	Code Generation	The interface automatically generates the corresponding Flux query in the background
10		User can click the button "run query" to create a query and generate code
11		The system processes the query and retrieves the data.
12		User can click the button "show code" to view the code
13		User can click the button "hide code" to not show the code
14		User can click the button "copy" to copy the code
15	Graph generation	When user click "run query", there is a figure shown in the dashboard
16		Add a graph module in the dashboard for future to show the figure
17		User can select the type of graph, such as line, bar, and so on

18	Testing dataset	Create some dataset with more than 10000 data
19		Find some existing dataset to test all functions
20	Real-time data preview	Add a preview data module in the dashboard
21		Based on user's selection in drag-and-drop, the preview data will be shown in 3 seconds if the user doesn’t have any other movements.
22		Once user finishes their selection (in 3 seconds without any other movements), the front-end send the request to the backend successfully
23		The preview must update dynamically as the user makes selections or changes.
24		The interface should handle large datasets efficiently, providing previews without significant delay.
25	Documentation	Finish and submit snapshot 4.1 on time
26		Finish and submit snapshot 4.2 on time
27		Analysis the updated requirements demonstrated in sprint meeting, separate it into tasks, and write the tasks on the project board
28		After the meeting, summarize meeting notes for both the team and Sprint meetings and upload for future reference.

4. Summary of Changes

- **Planning:**

  - **Copy Code and Query ID**

    After displaying the generated Flux code with the "Show Code" feature, a "Copy Code" option will enable users to easily copy the code to their clipboard. Additionally, each query will have a unique query ID generated and attached for easy identification and tracking.

  - **UI Improvements**

    UI improvements are planned for the dashboard, with updates to colors and images to ensure a uniform design once all features have been fully implemented.

- **In Progress:**

  - **Large Dataset Test Sample**

    The datasets provided in the InfluxDB documentation are relatively simple and do not adequately capture the complexity required to test the features outlined in the user stories and client requirements. We are seeking larger datasets to ensure comprehensive testing of all available features.

  - **Preview Data after Selection**

    When “preview data” button is clicked, users will be able to view selected data in a table that shows the summary of lines and entries of all the selected data with header information so that users can determine if they would like to proceed with the query and graph generation.

  - **Graph Generation**

    Users will be able to select the graph type and click a button to generate the graph based on user selections which will be displayed in the display area.

- **Testing/Done:**
  - **Login Logic**

    Test the login feature using existing InfluxDB accounts to verify that all buckets associated with these accounts are properly reflected in the interface.
  - **Filter Search Logic**

    Users can now search for specific filter key/value in each filter area to deal with large dataset scenarios.
  - **Code Generation**

    When users select the bucket, a Flux code line will be generated automatically, and then users proceed to select other filters, Flux code lines will be generated corresponding to the selections. Users can click “Show Code” button to display the Flux code for their selections, the code area will be displayed under the graph display area. Users can then click “Hide Code” to hide the code.