PANDAS OVERFLOW

CSCD01 Group Project

Deliverable 4

Zhiqi Chen
Christina Ma
Gabriela Esquivel Gaghi
Hyun Woo (Eddie) Shin
Arthur Lu
Man Hei Ho

Issue 1: [ENH] Support reduction methods on Index #50021

Assigned Member Tasks

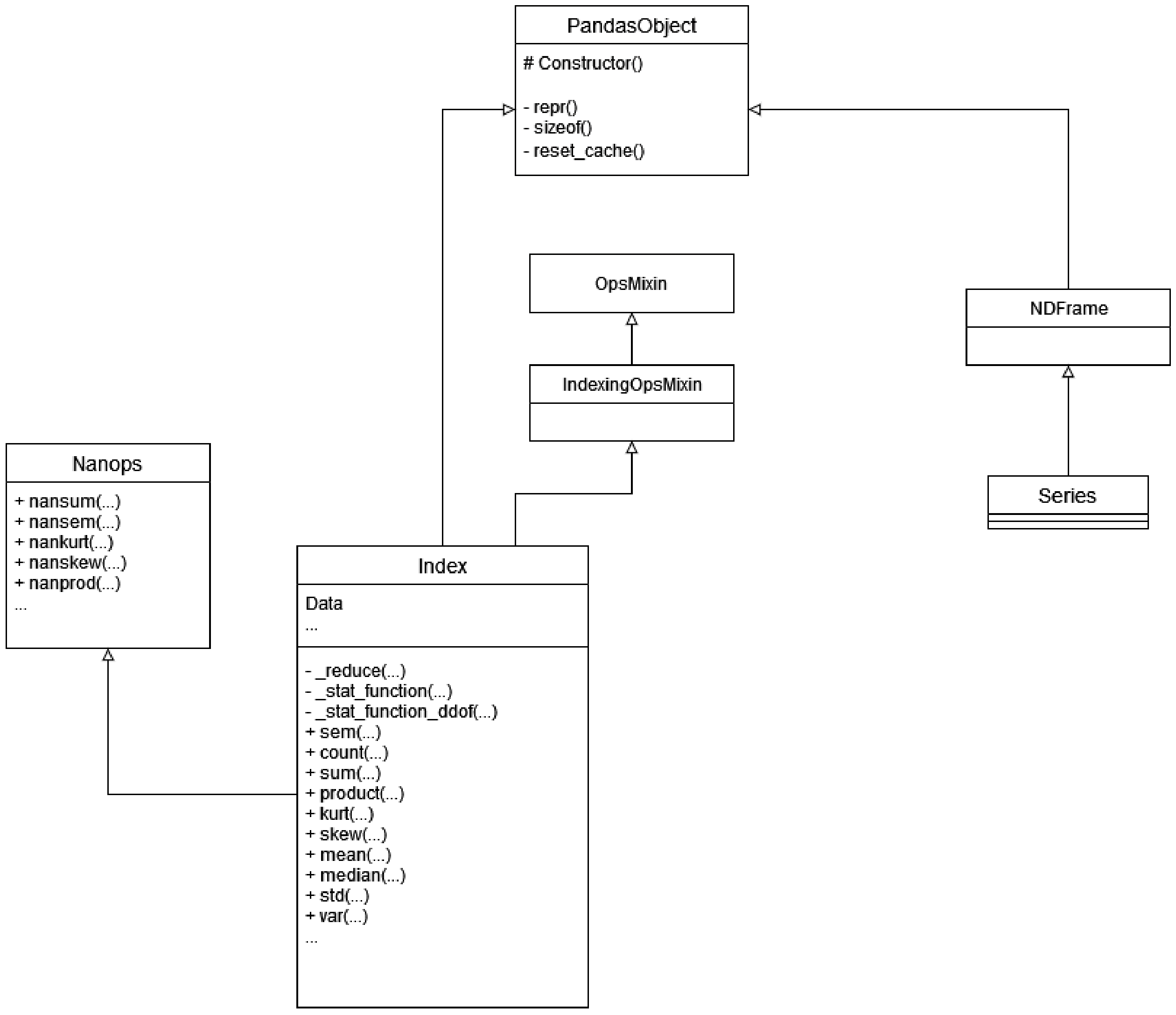
- Gabriela Esquivel Gaghi: Implemented the `mean()`, `median()`, `std()`, and `var()` Index methods, as well as the related tests and documentations.
- Zhiqi Chen: Implemented the `sum()`, `product()`, `count()` Index methods, as well as the related tests and documentations. Documentation of the design and implementation
- Arthur Lu: Implemented the `kurt()`, `skew()`, and `sem()` Index methods, as well as the related tests and documentations.

Issue Description

This issue requests an enhancement to the pandas Index to support all numeric reduction methods that are also supported by Series: count, sum, max, min, mean, prod, std, var, median, kurt, skew, and sem. Making Index more closely aligned and behave more similarly with Series.

While `max()` and `min()` are both already supported within index as reduction functions, the rest will need to be implemented as new reduction functions within index.

Design Documentation



Changes to the code base of Pandas are minimal in terms of impact to the original pandas design. Considering the fact that reduction functions already existed in Index – such as `min()` and `max()` – the changes to Index are mostly in expanding the number of reduction functions from 2 to 12. Furthermore, Nanops provided very useful helper functions that made creating the reduction functions easier and more decoupled. Errors are more unlikely as Nanops has been in use in pandas for a significant amount of time, and it is well tested.

An additional test file for testing these reduction functions and a file for acceptance testing were made. No other significant changes were made.

Implementation Documentation

For further compatibility with Series, as requested in the issue, the Nanops methods were used to perform each reduction operation. Each reduction function calls either `_stat_function` or `_stat_function_ddof`, passing (among other parameters) the specific Nanops method that will be used for calculations. Then the `_stat_function` or `_stat_function_ddof` will do some validation checking before calling `_reduce` which will invoke the proper reduction operation. This returns the value that the reduction operation performed over the index.

By creating these reduction methods for Index, we have satisfied the request of implementing all numeric reduction methods that Series has. This, in turn, makes Index more closely aligned to Series.

Testing

Unit Testing

Unit testing for the issue is provided at `pandas-copy/pandas/tests/indexes/test_reductions.py`. There are 19 test cases to test covering all 10 additional reduction methods as well as error cases when the user attempts to use these methods on non-numeric Index values. These test cases use randomized Index values generated from `pandas._testing.makeIndex()` methods in order to cover a wide variety of cases.

Acceptance Testing

The `pandas-copy/acceptance_tests_D4/issue1.py` file provides example acceptance tests that follow the steps outlined below under Acceptance Test for each scenario.

Test Objective: To ensure that users can perform reduction methods on a numeric Index.

Test Scenario 1: The user is working with an Index of integers.
Test Steps:
1. The user imports pandas into their file
2. The user creates a pandas Index with their desired integer values
3. The user uses the new reduction methods on the Index: kurt, skew, sem, count, sum, product, mean, median, std, and var.

Test Result: If the user can complete the steps without encountering any errors and the output matches their expected result then this has passed the user acceptance test.

Alternative Scenario 2: The user is working with an Index of integers with extreme values.

Alternative Scenario 3: The user is working with an Index of floats.

File Changes Made

`pandas-copy/pandas/core/indexes/base.py`
- Lines 6802–6835, 6967–7113, 7165–7432: Added reduction methods and necessary helper functions.

`pandas-copy/pandas/tests/indexes/test_reductions.py`
- New file: Contains unit tests for the reduction methods.

`pandas-copy/acceptance_tests_D4/issue1.py`
- New file: Contains acceptance tests for the reduction methods.

Issue 2: ENH: Allow easy selection of ordered/unordered categorical columns #46941

Assigned Member Tasks

- Hyun Woo (Eddie) Shin: participated in the designing of the solution and implemented the solution.
- Man Hei Ho: participated in the designing of the solution and implemented the solution.
- Christina Ma: participated in the designing of the solution and implemented unit tests for the solution.

Issue Description

This issue requests an enhancement to the pandas DataFrame to allow for a simpler way to select categorical columns that are either ordered or unordered. The specific requirement is to have an easy way (eg. a method) to get a new DataFrame with only ordered/unordered categorical columns.

Currently, the pandas DataFrame has a `select_dtypes()` function that enables users to get a new DataFrame with a subset of columns by specifying a list of column's data types to include or exclude. `CategoricalDtype` is a pandas data type for categorical data that includes categories and orderedness. This means that a categorical column could be either ordered or unordered. However, the current `select_dtypes()` function only supports including or excluding all categorical columns. The pandas DataFrame should support selecting only ordered or only unordered categorical columns.

Design Documentation

We decided to implement two solutions:

1. Modify the existing `select_dtypes()` function by adding an optional `ordered` parameter to allow including/excluding only ordered/unordered categorical columns.
2. Add a new Accessor object , `DataFrameCategoricalAccessor`, to DataFrame to add functionalities relating to the categorical columns of a DataFrame.

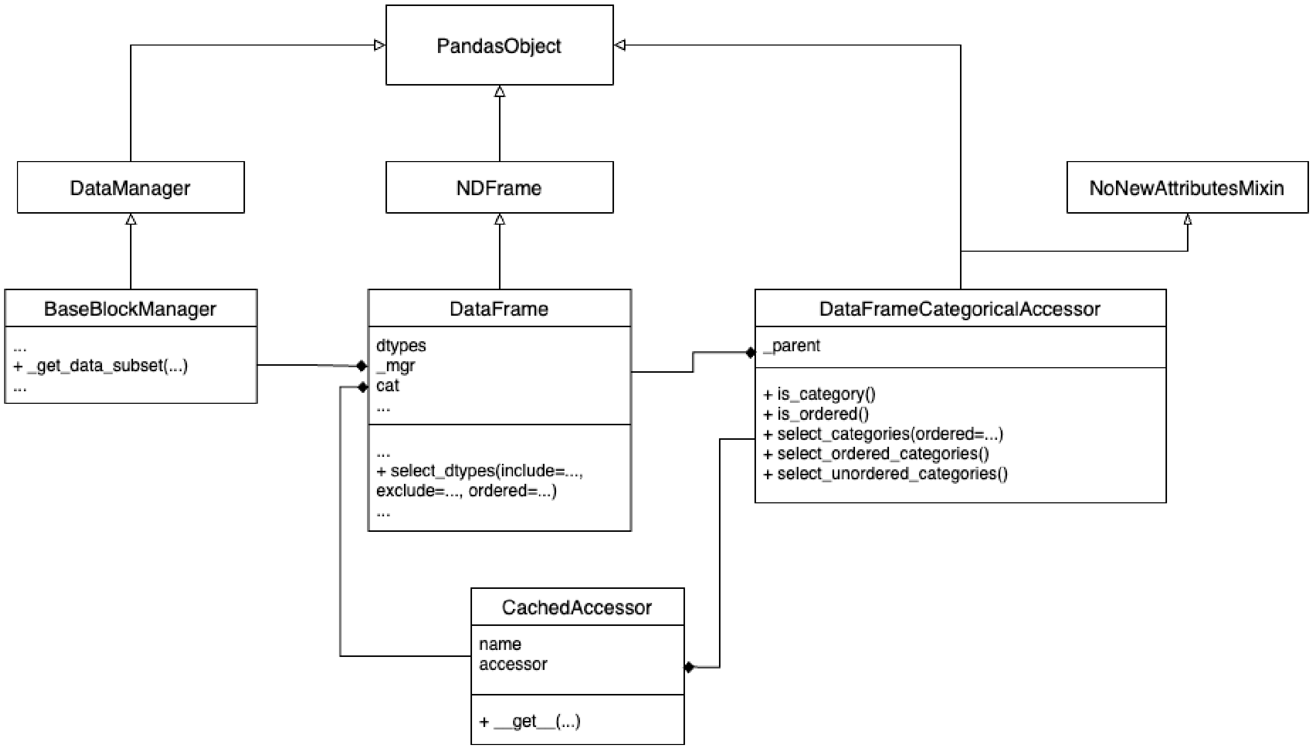
The above two solutions solves the same issue (ie. an easy way to select only ordered/unordered categorical columns) but they have different purposes and functions.

Solution 1 extends the existing `select_dtypes()` function directly, enabling users to select only ordered or unordered categorical columns. Note that with this function, users can choose to include/exclude a list of data types along with (ordered/unordered) categorical dtype.

Solution 2 involves implementing an Accessor class for the categorical columns of a DataFrame, similar to the `CategoricalAccessor` class for Series. The purpose of this class is to allow users to easily access the (ordered/unordered) categorical columns of a DataFrame and provide other functions relating to its categorical columns. The `DataFrameCategoricalAccessor` currently implements five functions, but it can be extended to support new requirements for a DataFrame's categorical columns. The five functions are:

1. `is_category()` returns a list of booleans which indicates whether a column is categorical or not.
2. `is_ordered()` returns a list of booleans and pandas.NA; boolean indicates whether a categorical column is ordered or not and pandas.NA indicates the column is not categorical.
3. `select_categories(ordered=None)`: returns a DataFrame with only categorical columns.
4. `select_ordered_categories()`: returns a DataFrame with only ordered categorical columns.
5. `select_unordered_categories()`: returns a DataFrame with only unordered categorical columns.

Implementation Documentation



For solution 1, the main change is in the `predicate` function that is passed into BaseBlockManager's `_get_data_subset()` function to determine which `dtypes` to include in the subset.

For solution 2, we utilized pandas's `CachedAccessor` class to create a new attribute `cat` in DataFrame. When user tries to access the `cat` attribute (ie. dataframe.cat), the `CachedAccessor` class would initialize a `DataFrameCategoricalAccessor` object (on first time call) with the DataFrame and caches it as an attribute of the DataFrame with the given name (`cat` in this case). The `DataFrameCategoricalAccessor` receives a DataFrame object on initialization, set it as an attribute `_parent`, and all the functions are performed on that DataFrame. The `DataFrameCategoricalAccessor` also extends the `NoNewAttributesMixin` class to ensure that no new attributes could be added to this class.

Testing

Unit Testing

Unit test for solution 1 implementation is located at `pandas-copy/pandas/tests/frame/methods/test_select_dtypes.py`. There are 8 test cases to test for the optional `ordered` parameter in `select_dtypes` function. These test using the `select_dtypes` function with different values for the two other pre-existing parameters, `include` and `exclude`, alongside the new `ordered` parameter. Each test case is commented to indicate the purpose of each test case.

Unit tests for solution 2 implementation are located at `pandas-copy/pandas/tests/frame/accessors/test_cat_accessor.py`. This is a newly created file since a new class was created. The test file location for the DataFrameCategoricalAccessor class mimics the test file location for the CategoricalAccessor class (an existing class that DataFrameCategoricalAccessor was modeled after). This test file provides individual unit tests for each of the five functions created.

Acceptance Testing

The `pandas-copy/acceptance_tests_D4/issue2.py` file provides example acceptance tests that follow the steps outlined below under Acceptance Test for each scenario.

Solution 1 Acceptance Test

Test Objective: To ensure that users can select ordered or unordered categories in a dataframe.

Test Scenario 1: A user wants to only select ordered categories in a DataFrame.

Test Steps:

4. The user imports pandas into their file
5. The user creates a pandas DataFrame with their desired data
6. The user uses this method `select_dtypes(include=["category"], ordered=True)` on their DataFrame object to select ordered categories.

Test Result: If the user can complete the steps without encountering any errors and the output matches their expected result then this has passed the user acceptance test.

Alternative Scenario 2: A user wants to select unordered categories in a DataFrame
At step 3, set `ordered=False` instead to select unordered categories.

Solution 2 Acceptance Test

Test Objective: To ensure that users can select ordered or unordered categories in a dataframe.

Test Scenario 1: A user wants to only select ordered categories in a DataFrame.

Test Steps:

1. The user imports pandas into their file
2. The user creates a pandas DataFrame with their desired data called `df`
3. The user uses these methods `.cat.select_categories(ordered=True)` on their DataFrame object to select ordered categories.

Test Result: If the user can complete the steps without encountering any errors and the output matches their expected result then this has passed the user acceptance test.

Alternative Scenario 2: A user wants to select unordered categories in a DataFrame.
At step 3, set `ordered=False` instead to select unordered categories.

Alternative Scenario 3: A user wants to select ordered categories in a DataFrame.
At step 3, use the method, `select_ordered_categories()`, instead of `select_categories(...)`.

Alternative Scenario 4: A user wants to select unordered categories in a DataFrame.
At step 3, use the method, `select_unordered_categories()`, instead of `select_categories(...)`.

Alternative Scenario 5: A user wants to select ordered categories in a DataFrame.
At step 3, use the method, `.loc[:, df.cat.is_ordered() & pd.notnull(df.cat.is_ordered())]` instead of `.cat.select_categories(...)`.

File Changes Made

`pandas-copy/pandas/core/arrays/categorical.py`

- Lines 2485-2701: Created DataFrameCategoricalAccessor which is an accessor object for categorical columns of a DataFrame.

`pandas-copy/pandas/core/frame.py`

- Lines 4714-4716, 4738-4741, 4777-4789: Modified `select_dtypes()` by adding extra `ordered` parameter to sort out ordered & unordered categories from a DataFrame.

`pandas-copy/pandas/tests/frame/methods/test_select_dtypes.py`

- Line 10-11, 99-149: Contains unit tests for the solution 1 of the issue.

`pandas-copy/pandas/tests/frame/accessors/test_cat_accessor.py`

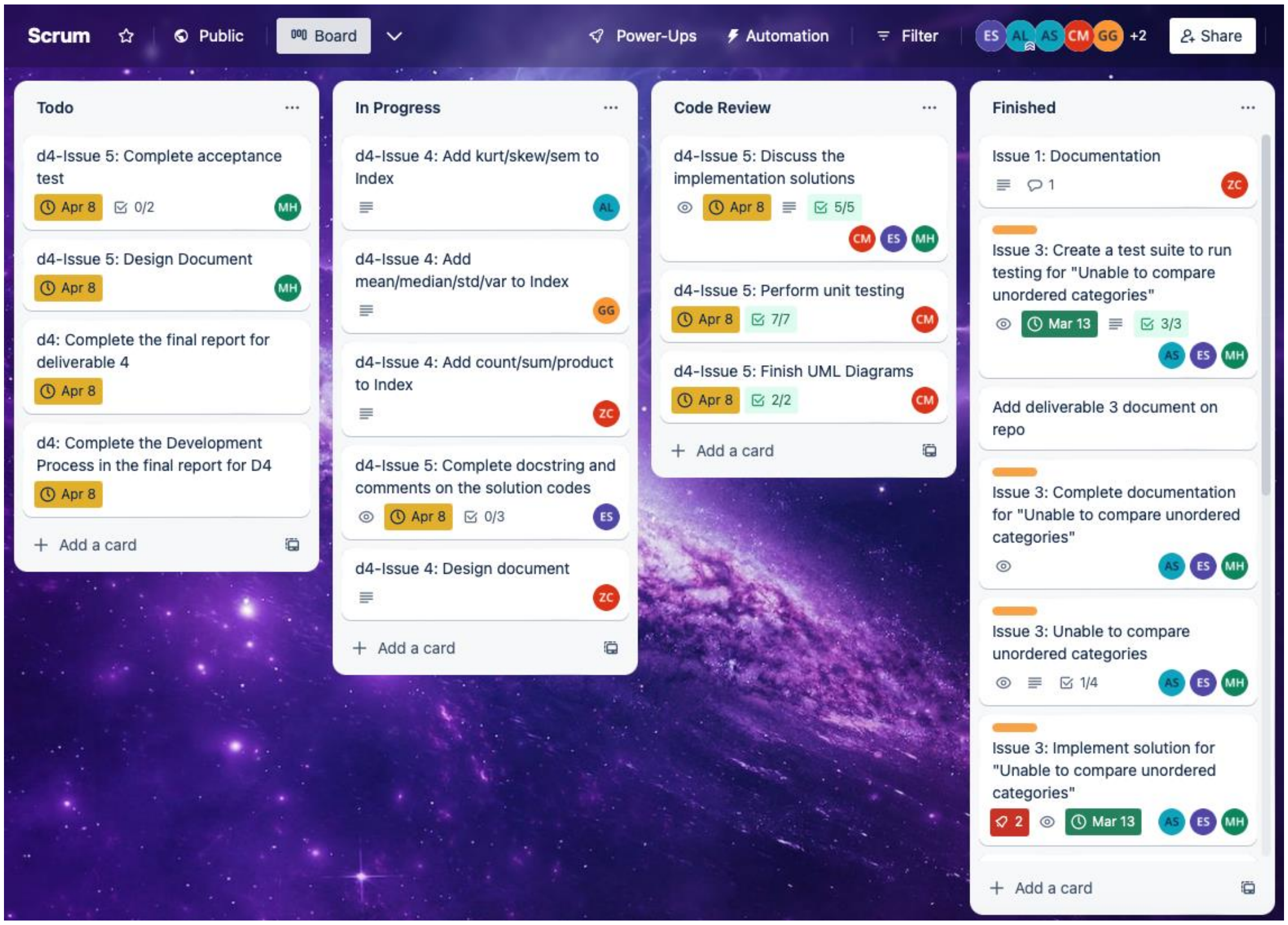
- New file: Contains unit tests for the solution 2 of the issue.

`pandas-copy/acceptance_tests_D4/issue2.py`

- New file: Contains acceptance tests for modified version of `select_dtypes()`and the newly implemented methods inside `DataFrameCategoricalAccessor` class.

Software Development Process: Kanban

Trello Board Link



Todo

- d4-Issue 5: Complete acceptance test — Apr 8, 0/2, MH
- d4-Issue 5: Design Document — Apr 8, MH
- d4: Complete the final report for deliverable 4 — Apr 8
- d4: Complete the Development Process in the final report for D4 — Apr 8
- Add a card

In Progress

- d4-Issue 4: Add kurt/skew/sem to Index — AL
- d4-Issue 4: Add mean/median/std/var to Index — GG
- d4-Issue 4: Add count/sum/product to Index — ZC
- d4-Issue 5: Complete docstring and comments on the solution codes — Apr 8, 0/3, ES
- d4-Issue 4: Design document — ZC
- Add a card

Code Review

- d4-Issue 5: Discuss the implementation solutions — Apr 8, 5/5, CM ES MH
- d4-Issue 5: Perform unit testing — Apr 8, 7/7, CM
- d4-Issue 5: Finish UML Diagrams — Apr 8, 2/2, CM
- Add a card

Finished

- Issue 1: Documentation — 1, ZC
- Issue 3: Create a test suite to run testing for "Unable to compare unordered categories" — Mar 13, 3/3, AS ES MH
- Add deliverable 3 document on repo
- Issue 3: Complete documentation for "Unable to compare unordered categories" — AS ES MH
- Issue 3: Unable to compare unordered categories — 1/4, AS ES MH
- Issue 3: Implement solution for "Unable to compare unordered categories" — 2, Mar 13, AS ES MH
- Add a card

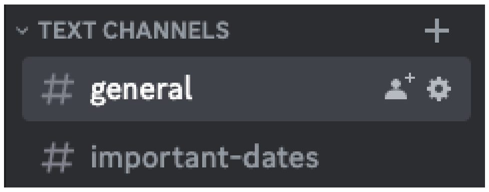
Trello was used to visually represent the team’s progress on their respective issues and subtasks. Each ticket on the Trello board contained a link to the issue on the Pandas repository as well as a short description of what needed to be implemented to resolve the ticket.
You can find the detailed overview of our SCRUM Board here. (You may need to sign up Trello and join the group to view the activities.)

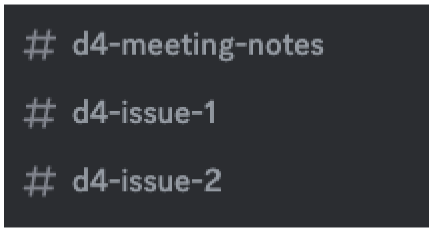
Team Communication

We met every Wednesday on Discord to update the progress on our assigned tasks, plan out next steps, and discuss any challenges that need to be addressed during the meetings. We also had some additional meetings beyond our scheduled Wednesday meetings for weeks that required additional team support due to challenges faced. These additional meetings were typically held amongst members who were working on the same issue.

All other forms of communications were through our Discord chat server in our various text channels.

- general: Any communication meant for all group members.
- Important-date: Posted future meeting dates
- d4-meeting notes: All meeting minutes
- d4-issue-1: Any communication regarding issue 1
- d4-issue-2: Any communication regarding issue 2


TEXT CHANNELS
# general
# important-dates


d4-meeting-notes
d4-issue-1
d4-issue-2

All members periodically checked the chat and replied to any questions in a timely manner. The chat server was effective in asking broad questions and collaborating to resolve challenges faced. This kept our meetings short as most challenges were resolved before we had our meeting.

Meeting Notes

Meeting #	Date & Time	Duration	Summary
1	Sun, March 19 6:00 pm	36 mins	• Read through deliverable 4 document • Selected potential issues for Deliverable 4
2	Wed, March 22 7:30 pm	25 mins	• Finalized issues selection • Assigned each member to an issue
3	Fri, March 25 2:00 pm	31 mins	• Divided issue 1 into sub-tasks • Divided sub-tasks between members of issue 1 • Issue 2 switched issues
4	Wed, April 5 9:30 pm	55 mins	• Issue 2 Meeting • Discussed the issue together • Designed two solutions • Assigned work to each member
5	Fri, April 7 10:30 pm	54 mins	• Issue 2 Meeting • Reviewed both solution implementations • Discussed remaining tasks and assigned them between members
6	Fri, April 7 4:00 pm	30 mins	• Issue 1 Meeting • Discussed documentation and ensured consistency between branches
7	Sat, April 8 5:00 pm	25 mins	• Reviewed the progress for both issues • Discussed finishing touches of report documentation