Astra

Team 34, for QEYnet

About Astra

QEYnet

- Startup company aiming to facilitate quantum key distribution with satellites
- Needs to monitor status of satellites and other devices

About Astra

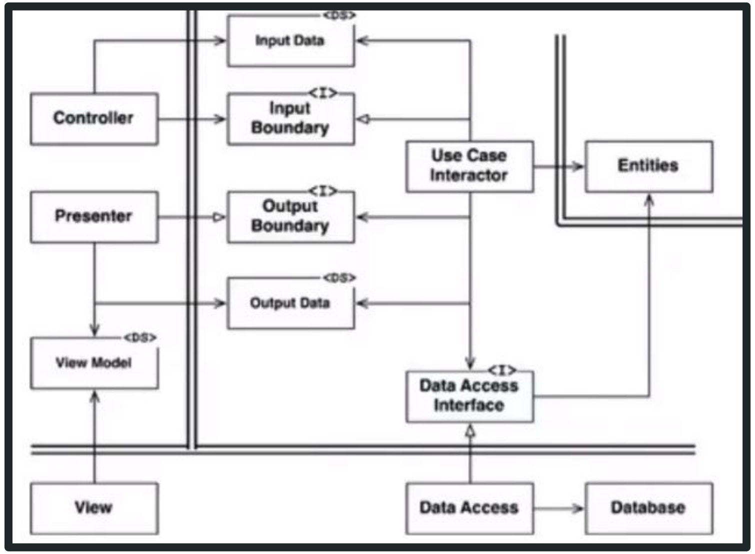
- Purpose: Monitoring telemetry data for devices
    - Track status of devices
    - Indicate potential problems with devices
    - Visualize device telemetry data
- Target users:
    - QEYnet employees
    - QEYnet customers
- Local application developed from scratch

Demo

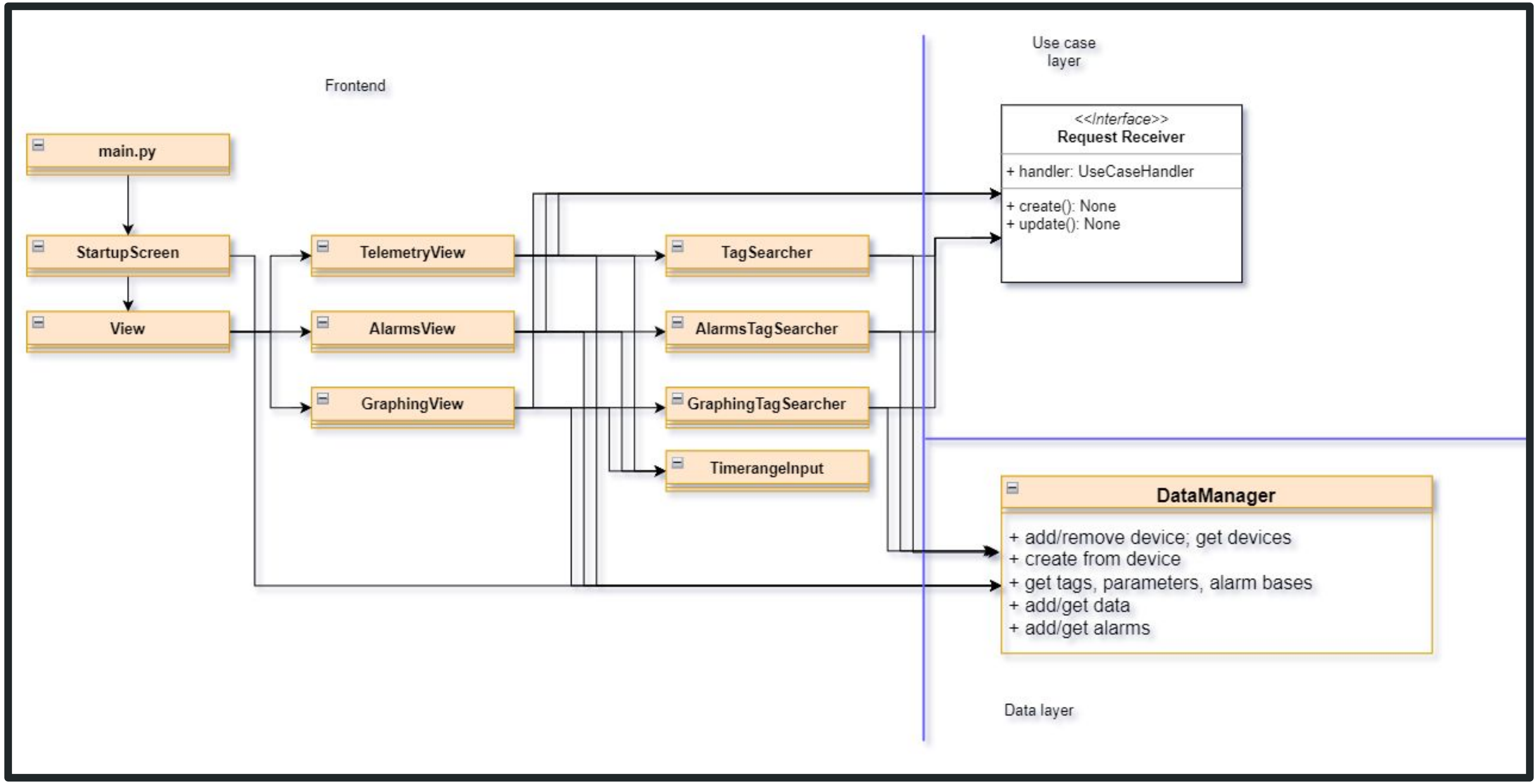
Architecture & Technical Discussion

Architecture

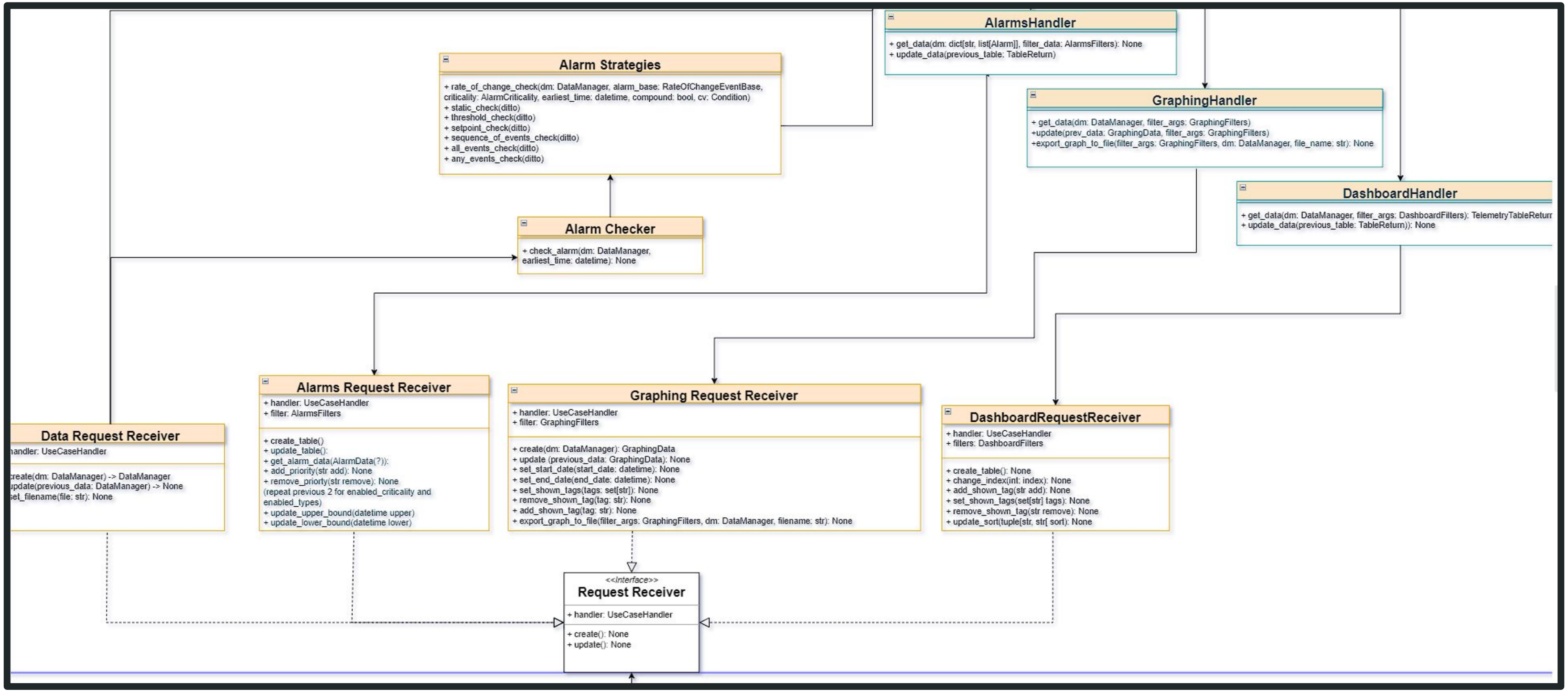
- Divided into three major sections: data, use case, frontend
  - Sections exist for logical organization – clean architecture is still respected
  - Data: file I/O and persistence
  - Use case: data retrieval and formatting
  - Frontend: GUI for the program



Frontend Subteam



Use Case Subteam



```
AlarmsHandler
+ get_data(dm: dict[str, list[Alarm]], filter_data: AlarmsFilters): None
+ update_data(previous_table: TableReturn)
```

```
Alarm Strategies
+ rate_of_change_check(dm: DataManager, alarm_base: RateOfChangeEventBase,
criticality: AlarmCriticality, earliest_time: datetime, compound: bool, cv: Condition)
+ static_check(ditto)
+ threshold_check(ditto)
+ setpoint_check(ditto)
+ sequence_of_events_check(ditto)
+ all_events_check(ditto)
+ any_events_check(ditto)
```

```
GraphingHandler
+ get_data(dm: DataManager, filter_args: GraphingFilters)
+update(prev_data: GraphingData, filter_args: GraphingFilters)
+export_graph_to_file(filter_args: GraphingFilters, dm: DataManager, file_name: str): None
```

```
DashboardHandler
+ get_data(dm: DataManager, filter_args: DashboardFilters): TelemetryTableReturr
+ update_data(previous_table: TableReturn)): None
```

```
Alarm Checker
+ check_alarm(dm: DataManager,
earliest_time: datetime): None
```

```
Data Request Receiver
handler: UseCaseHandler

create(dm: DataManager) -> DataManager
update(previous_data: DataManager) -> None
set_filename(file: str): None
```

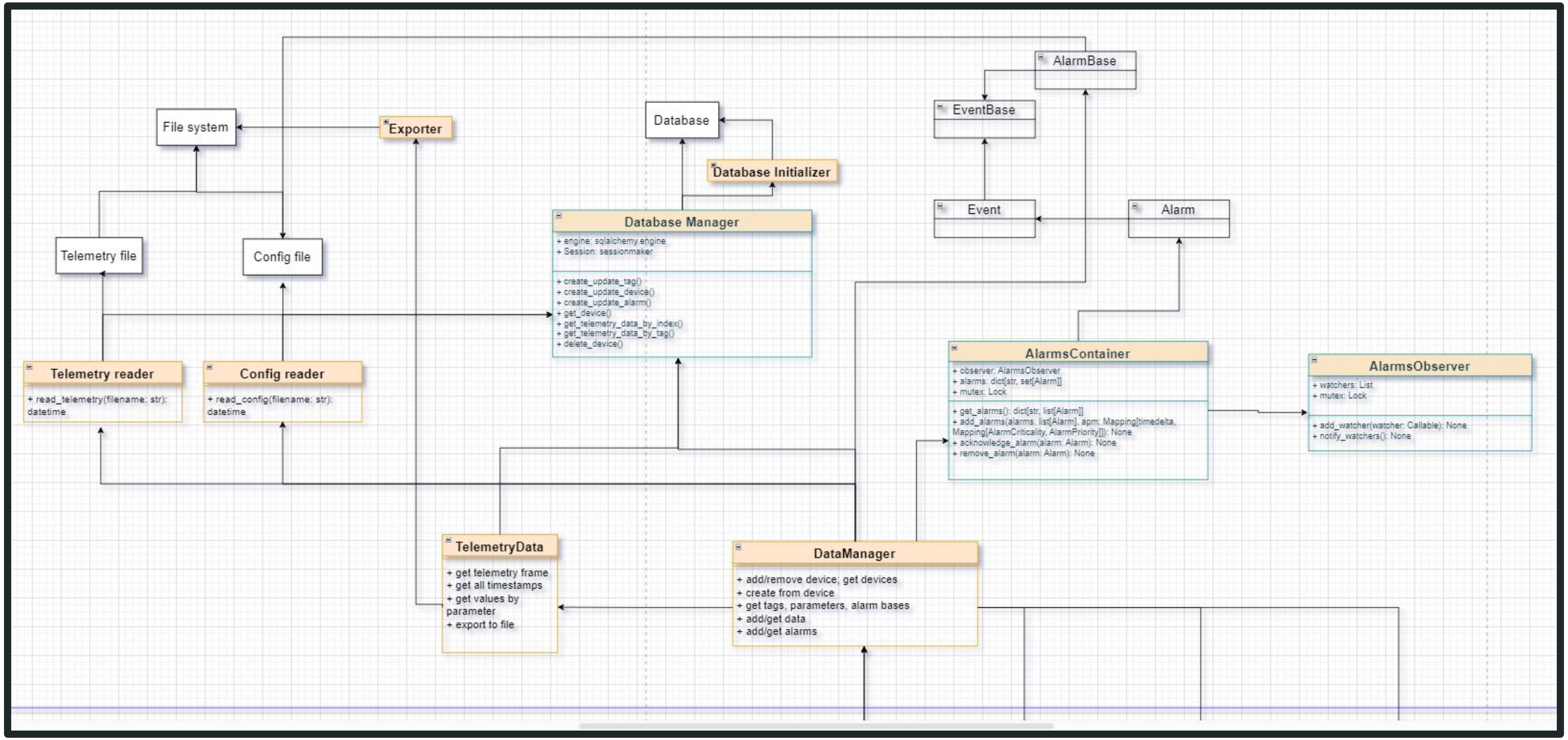
```
Alarms Request Receiver
+ handler: UseCaseHandler
+ filter: AlarmsFilters

+ create_table()
+ update_table():
+ get_alarm_data(AlarmData(?)):
+ add_priority(str add): None
+ remove_priorty(str remove): None
(repeat previous 2 for enabled_criticality and
enabled_types)
+ update_upper_bound(datetime upper)
+ update_lower_bound(datetime lower)
```

```
Graphing Request Receiver
+ handler: UseCaseHandler
+ filter: GraphingFilters

+ create(dm: DataManager): GraphingData
+ update (previous_data: GraphingData): None
+ set_start_date(start_date: datetime): None
+ set_end_date(end_date: datetime): None
+ set_shown_tags(tags: set[str]): None
+ remove_shown_tag(tag: str): None
+ add_shown_tag(tag: str): None
+ export_graph_to_file(filter_args: GraphingFilters, dm: DataManager, filename: str): None
```

```
DashboardRequestReceiver
+ handler: UseCaseHandler
+ filters: DashboardFilters

+ create_table(): None
+ change_index(int: index): None
+ add_shown_tag(str add): None
+ set_shown_tags(set[str] tags): None
+ remove_shown_tag(str remove): None
+ update_sort(tuple[str, str[ sort): None
```

```
<<Interface>>
Request Receiver
+ handler: UseCaseHandler

+ create(): None
+ update(): None
```

Data Subteam

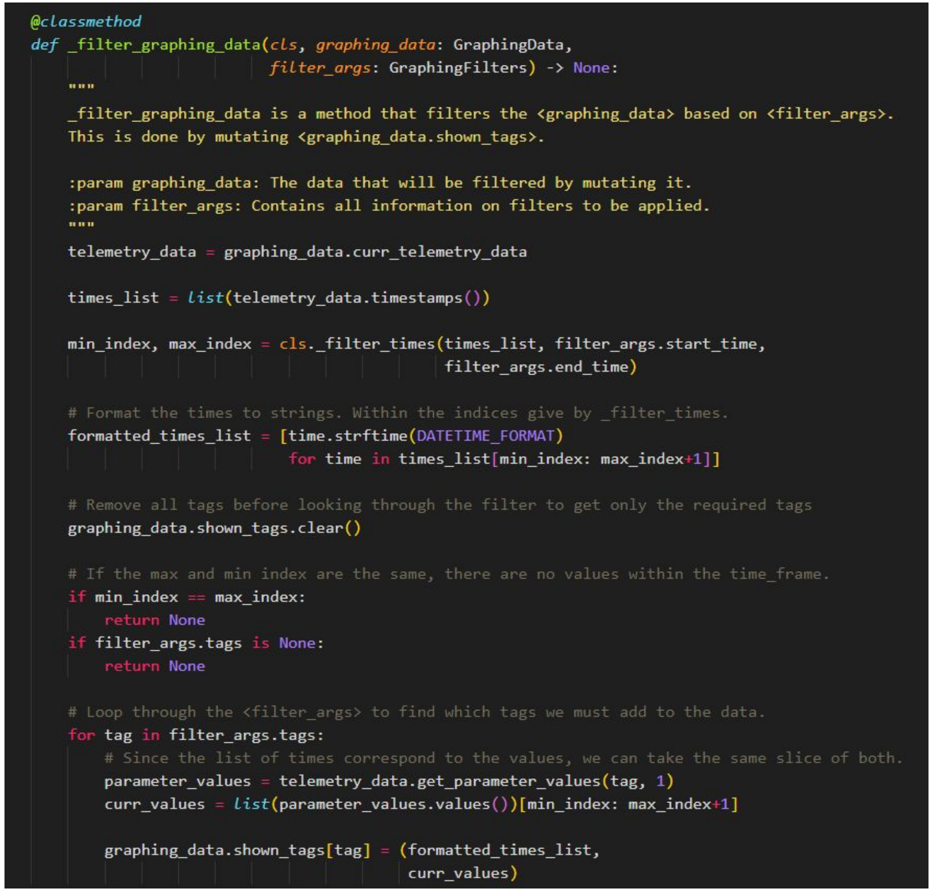

AlarmBase
File system
Exporter
Database
EventBase
Database Initializer
Event
Alarm
Database Manager
+ engine: sqlalchemy.engine
+ Session: sessionmaker
+ create_update_tag()
+ create_update_device()
+ create_update_alarm()
+ get_device()
+ get_telemetry_data_by_index()
+ get_telemetry_data_by_tag()
+ delete_device()
Telemetry file
Config file
AlarmsContainer
+ observer: AlarmsObserver
+ alarms: dict[str, set[Alarm]]
+ mutex: Lock
+ get_alarms(): dict[str, list[Alarm]]
+ add_alarms(alarms: list[Alarm], apm: Mapping[timedelta, Mapping[AlarmCriticality, AlarmPriority]]): None
+ acknowledge_alarm(alarm: Alarm): None
+ remove_alarm(alarm: Alarm): None
AlarmsObserver
+ watchers: List
+ mutex: Lock
+ add_watcher(watcher: Callable): None
+ notify_watchers(): None
Telemetry reader
+ read_telemetry(filename: str): datetime
Config reader
+ read_config(filename: str): datetime
TelemetryData
+ get telemetry frame
+ get all timestamps
+ get values by parameter
+ export to file
DataManager
+ add/remove device; get devices
+ create from device
+ get tags, parameters, alarm bases
+ add/get data
+ add/get alarms

Tech Stacks

- Programming language: Python
  - Used by QEYnet already
  - Familiar to the team
  - Has libraries like Tkinter, Matplotlib, pandas
- Coding in Python 3.12
  - Newest version
  - Has helpful static typing features

Coding Style

- Follow PEP 8
  - Line length 100
- Flake8 for style enforcement
- Decided-upon conventions:
  - Single quotes for all strings
  - Docstring format: reST

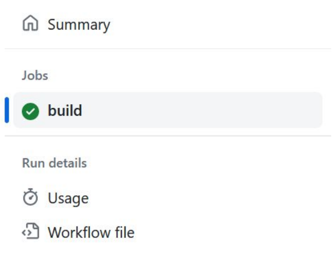


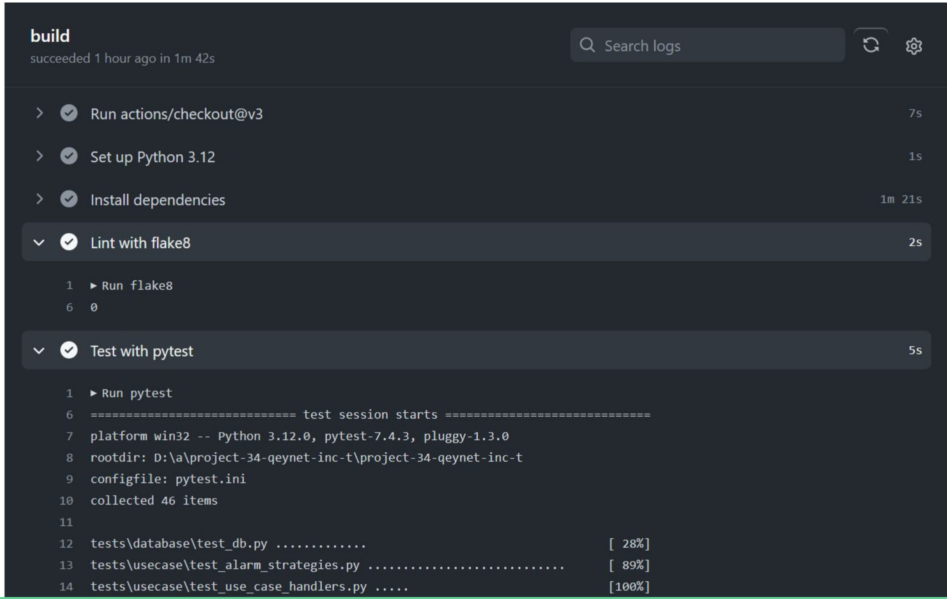
```python
@classmethod
def _filter_graphing_data(cls, graphing_data: GraphingData,
                          filter_args: GraphingFilters) -> None:
    """
    _filter_graphing_data is a method that filters the <graphing_data> based on <filter_args>.
    This is done by mutating <graphing_data.shown_tags>.

    :param graphing_data: The data that will be filtered by mutating it.
    :param filter_args: Contains all information on filters to be applied.
    """
    telemetry_data = graphing_data.curr_telemetry_data

    times_list = list(telemetry_data.timestamps())

    min_index, max_index = cls._filter_times(times_list, filter_args.start_time,
                                             filter_args.end_time)

    # Format the times to strings. Within the indices give by _filter_times.
    formatted_times_list = [time.strftime(DATETIME_FORMAT)
                            for time in times_list[min_index: max_index+1]]

    # Remove all tags before looking through the filter to get only the required tags
    graphing_data.shown_tags.clear()

    # If the max and min index are the same, there are no values within the time_frame.
    if min_index == max_index:
        return None
    if filter_args.tags is None:
        return None

    # Loop through the <filter_args> to find which tags we must add to the data.
    for tag in filter_args.tags:
        # Since the list of times correspond to the values, we can take the same slice of both.
        parameter_values = telemetry_data.get_parameter_values(tag, 1)
        curr_values = list(parameter_values.values())[min_index: max_index+1]

        graphing_data.shown_tags[tag] = (formatted_times_list,
                                         curr_values)
```

Process & Deployment

Automation

- Github workflows
  - Linting
  - Running tests





```
Run actions/checkout@v3                                   7s
Set up Python 3.12                                        1s
Install dependencies                                 1m 21s
Lint with flake8                                          2s
▸ Run flake8
0
Test with pytest                                          5s
▸ Run pytest
============================ test session starts ============================
platform win32 -- Python 3.12.0, pytest-7.4.3, pluggy-1.3.0
rootdir: D:\a\project-34-qeynet-inc-t\project-34-qeynet-inc-t
configfile: pytest.ini
collected 46 items

tests\database\test_db.py ............                                 [ 28%]
tests\usecase\test_alarm_strategies.py ...........................     [ 89%]
tests\usecase\test_use_case_handlers.py .....                          [100%]
```

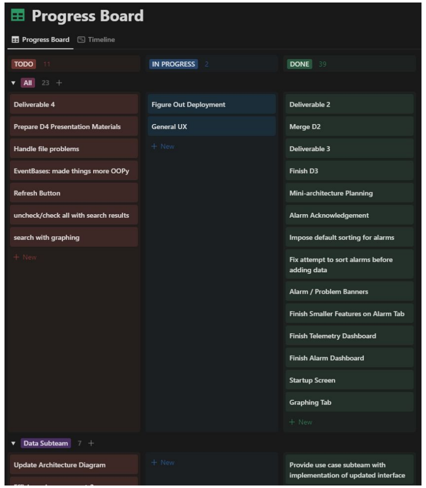
Working Together

Asynchronous work

- Discord for communication
- Used a Notion progress board

Synchronous work

- Meeting regularly on Thursdays
- Weekly meetings with partner
  - Asynchronous communication through Discord



Progress Board

Progress Board | Timeline

TODO 11	IN PROGRESS 2	DONE 39
All 23		
Deliverable 4	Figure Out Deployment	Deliverable 2
Prepare D4 Presentation Materials	General UX	Merge D2
Handle file problems	+ New	Deliverable 3
EventBases: made things more OOPy		Finish D3
Refresh Button		Mini-architecture Planning
uncheck/check all with search results		Alarm Acknowledgement
search with graphing		Impose default sorting for alarms
+ New		Fix attempt to sort alarms before adding data
		Alarm / Problem Banners
		Finish Smaller Features on Alarm Tab
		Finish Telemetry Dashboard
		Finish Alarm Dashboard
		Startup Screen
		Graphing Tab
		+ New
Data Subteam 7		
Update Architecture Diagram	+ New	Provide use case subteam with implementation of updated interface

Deployment & Accessing the application

Accessing the app

- Local application for Windows
- Download executable from repo

Reflection & Key Learnings

Places for improvement

- Taking initiative for next steps in project
- Code review and branching process

Next steps for D5

- Clean up the code and architecture
- More comprehensive automated testing
- Better error handling for file reading
- Optimization:
    - Speed up alarm checking
    - Cache results from database

Handing off the code

- Clean everything up
- Ensure code, architecture, and issues are documented
- Partner will fork the repo and develop from there

Individual Contributions

Andy Guo

- Frontend subteam - help design in initial architecture of frontend
- Telemetry tab - displaying data table and column filtering
- Normalization of graphing data
- Frontend tag searching and selection

Lingyun Jiang

- Data subteam - initial datateam architecture
- Database setup and initialization
- All Database interactions
- File parser for both the telemetry and configuration files
- Unit tests for Database manager.

Bonan Luan

- Data subteam - initial general architecture and DataManager
- Frontend - time range selection and startup screen
- Communication with partner
- General team coordination

Alexander Mathioudakis

- Helped with high level architecture design
  - Participated in creating our architecture diagram
- Use case subteam -
  - Wrote use-case classes for graphing tab
  - Wrote half of the alarm checking algorithms

Liam Odero

- Use case subteam - Helped design general architecture
  - Logic for dashboard and alarms tabs
  - Logic for half of alarm checking algorithms
- Frontend - backend integration
- Data - Setup alarm storage

Albert Yan

- Frontend subteam
  - Implemented initial frontend for view
  - Initial setup and prototyping
- Alarms tab
  - Created the alarms view
  - Displaying alarms, button filtering

Thank you! Any questions?