# Project Documentation

*Team Number 42*

This documentation provides an overview of the purpose, usage, and functionality of each module, class, and method in the project. It includes examples and usage scenarios to guide users effectively. Additionally, it incorporates implementation documentation with UML class and swimlane diagrams to visualize the system architecture and interaction flow.

## 1. main.py

This script serves as the entry point for the application. It demonstrates the functionalities of querying and processing geospatial data using the classes defined in classes.py.

- Python 3
- Requests library

Usage:

```
from classes import DatabaseConnection, Datacube


if __name__ == "__main__":
    # Creating a DatabaseConnection object with WCPS processing server
    dbc = DatabaseConnection("https://ows.rasdaman.org/rasdaman/ows")


    # Creating a Datacube object with the DatabaseConnection object
    dco = Datacube(dbc)
```

```
# Example - 1: finding the minimum of the AvgLandTemp datacube within a specified range
datacube_name = "AvgLandTemp"
subset_params = 'Lat(53.08), Long(8.80), ansi("2014-01":"2014-12")'
result1 = float(dco.min_datacube(datacube_name, subset_params))
print("Example 1 Result:", result1)
assert result1 == 2.2834647, "Example 1 test failed!"

# Similar Examples

# Confirmation of successful execution of all tests
print("All tests passed successfully!")
```

## 2. classes.py

This script contains the definitions of the DatabaseConnection and Datacube classes, responsible for interacting with the WCPS server and performing data queries and processing.

- Python 3
- Requests library

### DatabaseConnection class:

- Responsible for establishing a connection to the WCPS server and executing WCPS queries.
- Methods:
- __init__(self, server_url): Constructor to initialize the server URL.

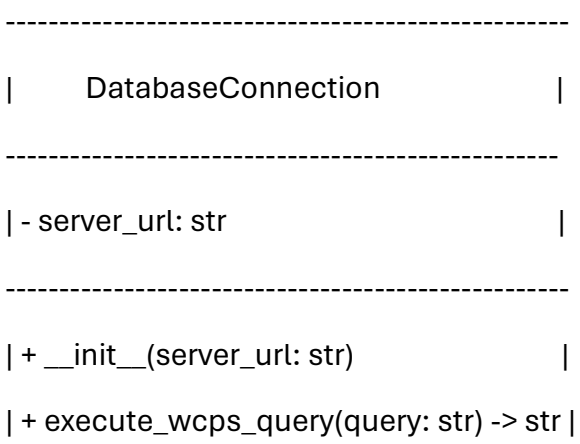
- execute_wcps_query(self, query): Executes a WCPS query and returns the response.

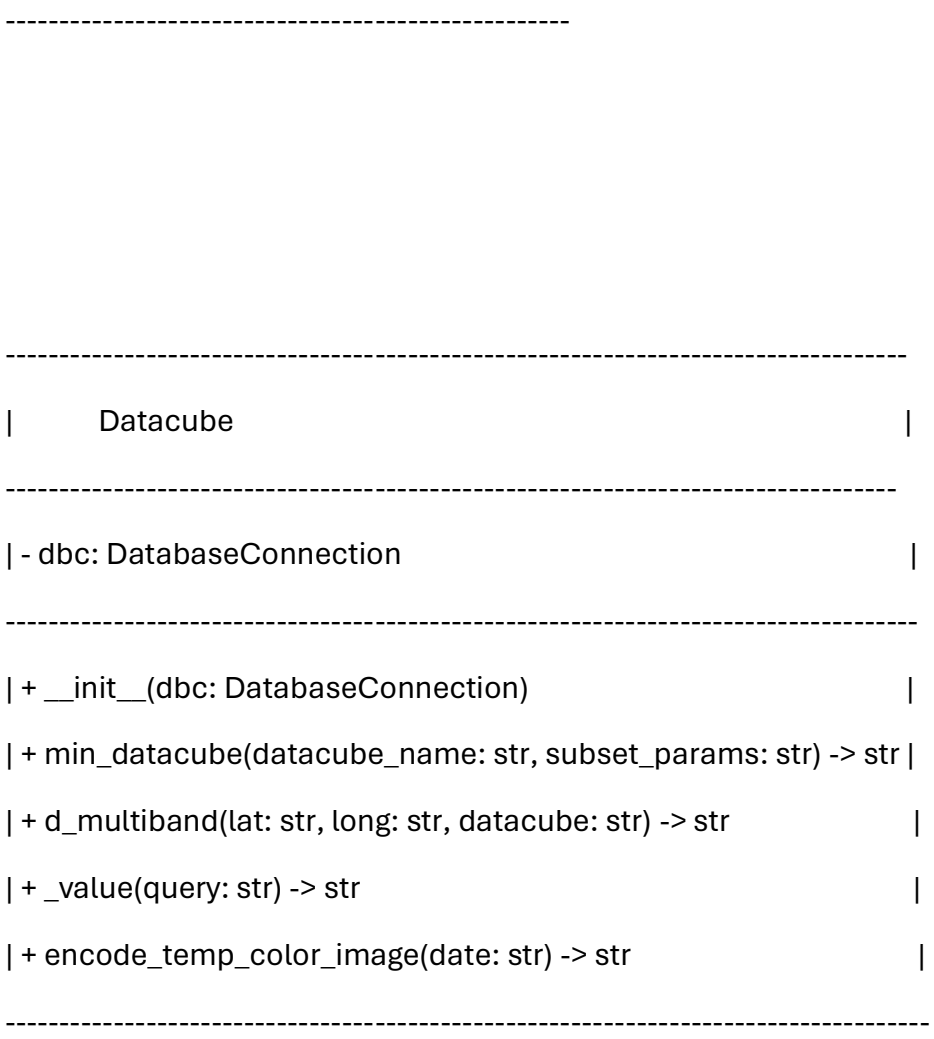
## Datacube class:

- Utilizes the DatabaseConnection object to perform various data processing operations.
- Methods:
- __init__(self, dbc): Constructor to initialize the DatabaseConnection object.
- min_datacube(self, datacube_name, subset_params): Finds the minimum value within a specified subset of a datacube.
- d_multiband(self, lat, long, datacube): Retrieves multiband data for a specific location.
- _value(self, query): Executes a custom WCPS query.
- encode_temp_color_image(self, date): Encodes temperature color image data for a specific date.

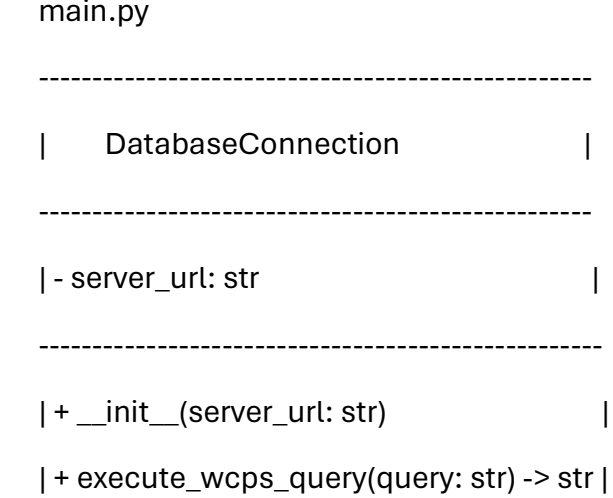
## Implementation Documentation:

### UML Class Diagram:






**Swimlane Diagram:**



---

Classes.py

```
-------------------------------------------------------------------------------
|             Datacube                                                        |
-------------------------------------------------------------------------------
| - dbc: DatabaseConnection                                                   |
-------------------------------------------------------------------------------
| + __init__(dbc: DatabaseConnection)                                         |
| + min_datacube(datacube_name: str, subset_params: str) -> str |
| + d_multiband(lat: str, long: str, datacube: str) -> str                    |
| + _value(query: str) -> str                                                 |
| + encode_temp_color_image(date: str) -> str                                 |
-------------------------------------------------------------------------------
```

## Conclusion:

This project demonstrates how to interact with a WCPS server for querying and processing geospatial data. It provides a foundation for building applications that require geospatial data analysis and visualization capabilities.