INFRASTRUCTURE

CSP-586: SOFTWARE MODELING DEVELOPMENT WITH UML

TEAM # 14:

ANSHIKA TRIVEDI [atrivedi1@hawk.iit.edu]

CHITRARTH SINGH [csingh7@hawk.iit.edu]

JOELDEEP KAUR [fjoeldeepkaur@hawk.iit.edu]

NISHTHA GARG [ngarg5@hawk.iit.edu]

SAURABH KHETAN [skhetan1@hawk.iit.edu]

PROJECT OVERVIEW

Infrastructure dashboard is a locally hosted resource that provides simple and quick access to collections of time series data. One of the most effective ways to display development indicators is through graphs and charts. A visual display of data makes comparisons easier and promotes a better understanding of trends. They provide data dashboards on various topics as well as access to all the underlying data through our latest data visualization. It has advanced functions for selecting and displaying data, performing customized queries, and creating charts. Users can create dynamic custom tables based on their selection of countries, indicators and years. The datasets have been downloaded from **World Bank website (**databank.worldbank.org).

Featured highlights include:

- Simple and quick access to data from 15 databases.
- Enhanced visualizations capabilities with Excel-like functions.
- Features to view tables i.e. datasets, charts.
- Features to filter on basis of sub-regional country.

REQUIREMENTS / FEATURES LIST

REQUIREMENTS:

1. ECMA6
2. JavaScript
3. HTML
4. CSS
5. Chart.js
6. Dataframe.js
7. Bootstrap

FEATURES:

1. Choosing the desired regional dataset out of the 3 main infrastructure attributes.
2. Option of choosing the labels to filter dataset and see a concise dataset.
3. Option of plotting charts type and visualizing data through charts.
4. Selecting the labels[column] for querying and fetching values from dataset and consequently plotting charts from those values.
5. Selecting the rows and querying based on the rows values. For e.g.: country_name

USE CASES & ACTORS

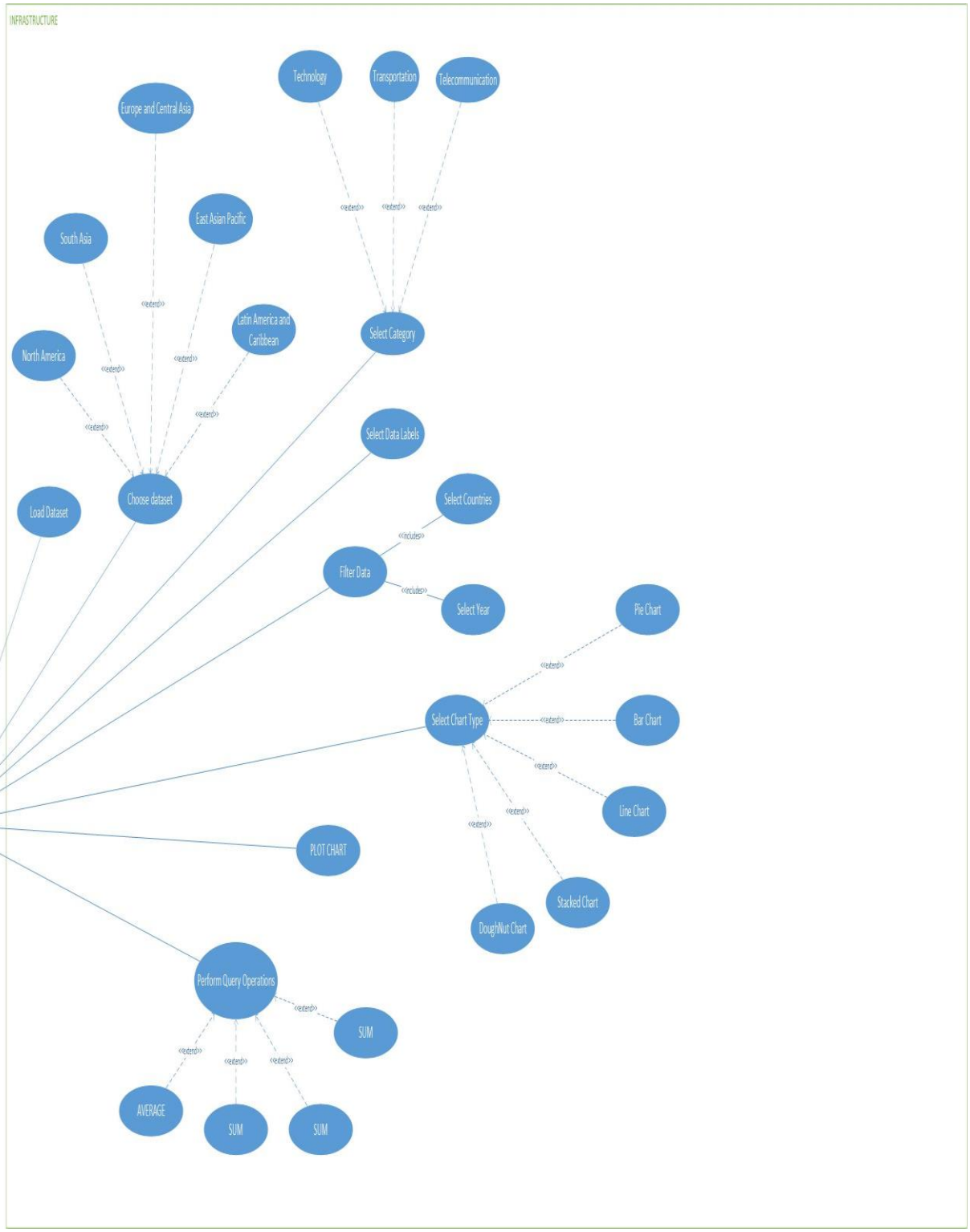
USE CASES LIST:

- LOAD DATASET
- CHOOSE DATASET
- SELECT CATEGORY
- SELECT DATA LABELS
- SELECT CHART TYPE
- FILTER DATA
- PERFORM QUERY OPERATIONS
- PLOT CHART

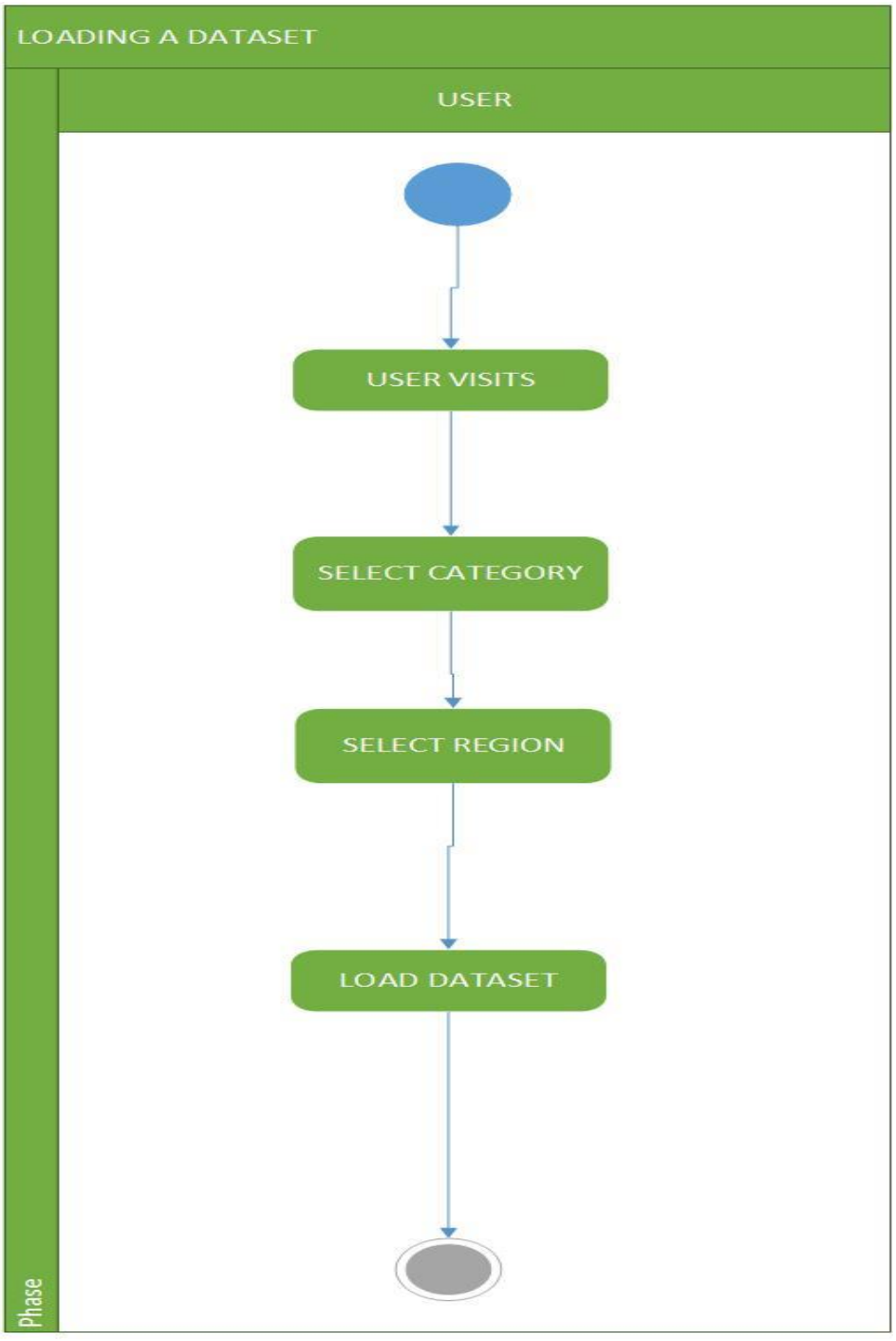
ACTORS:

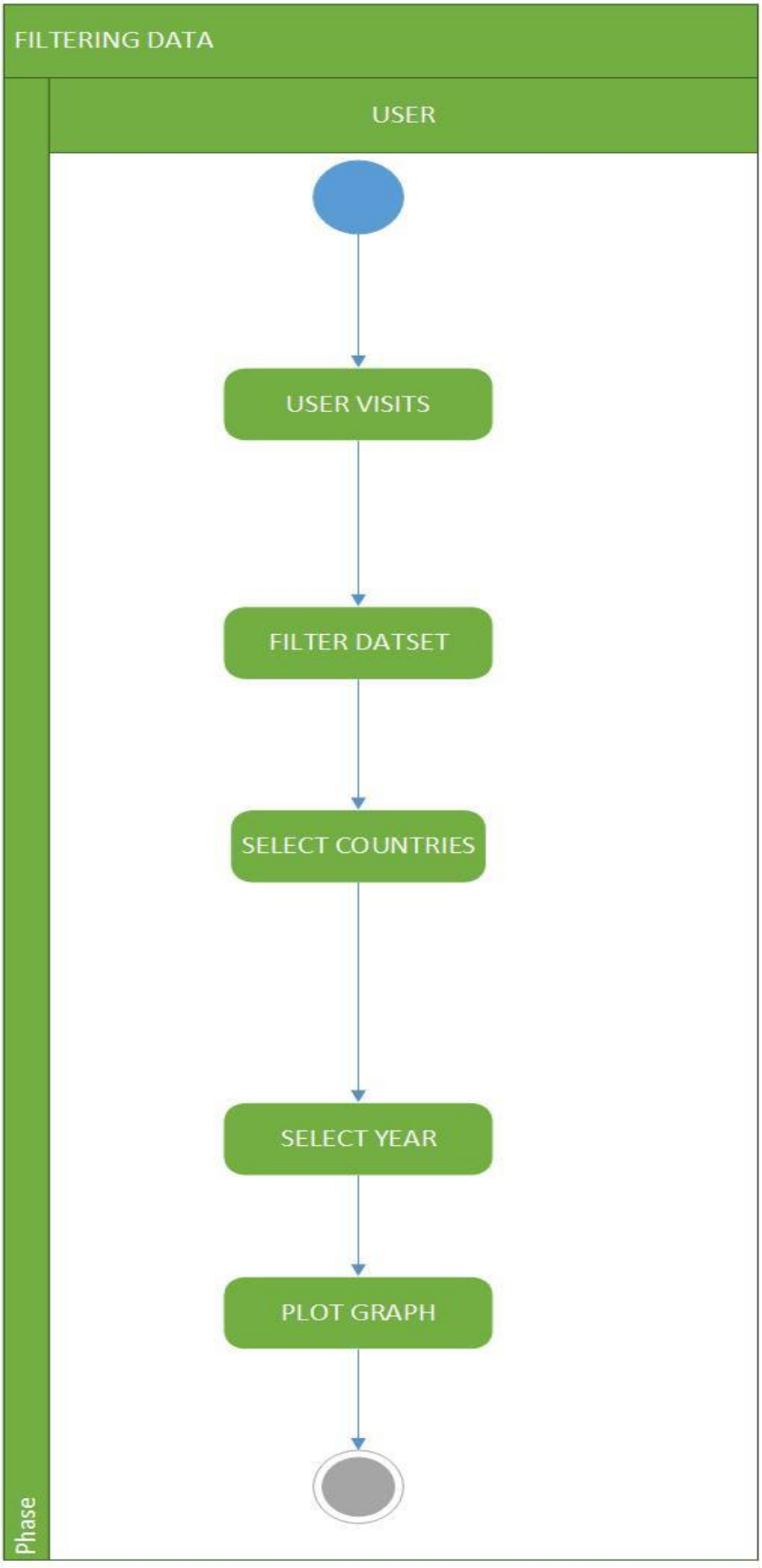
- USER

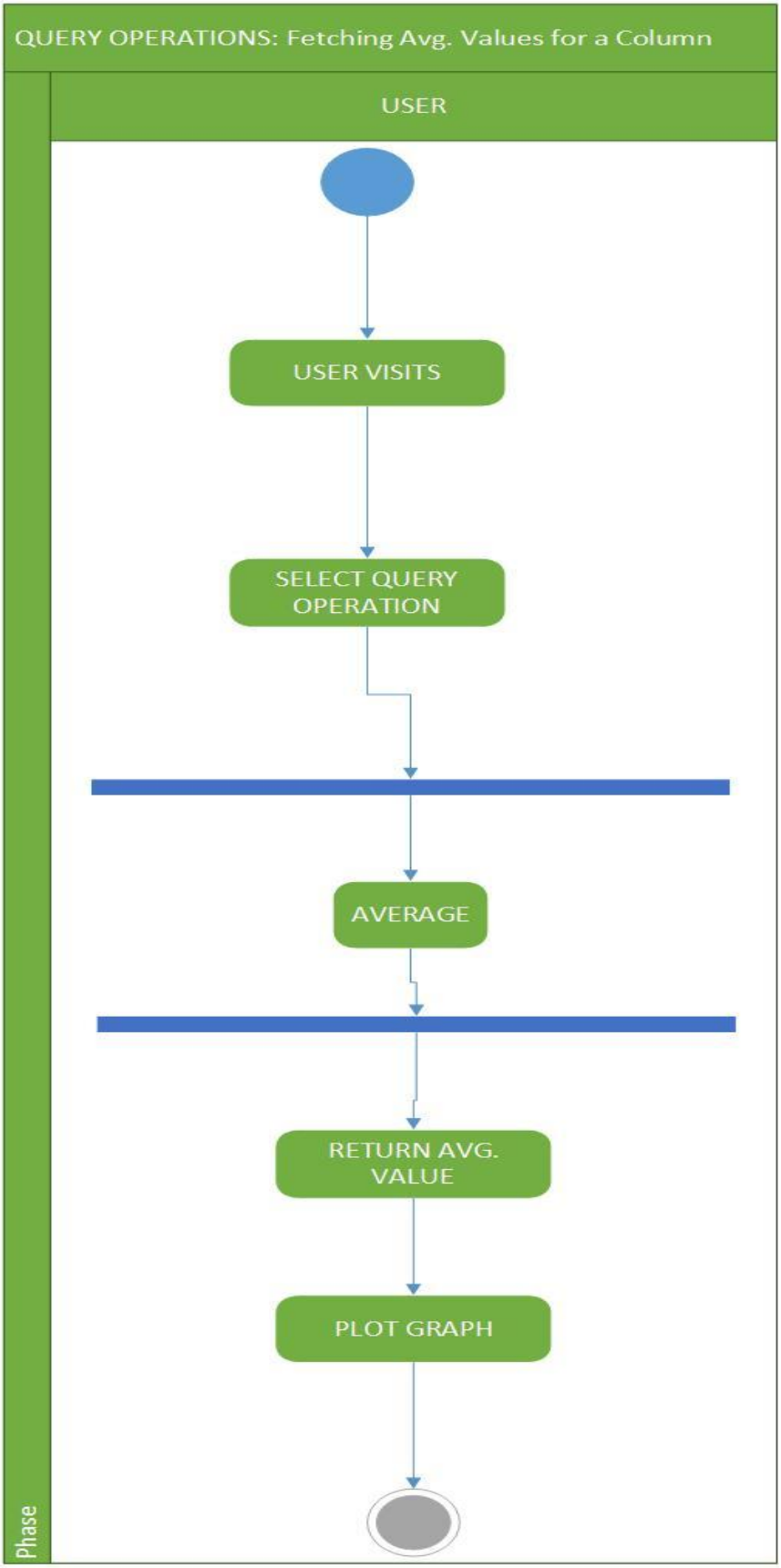
USE CASE DIAGRAM

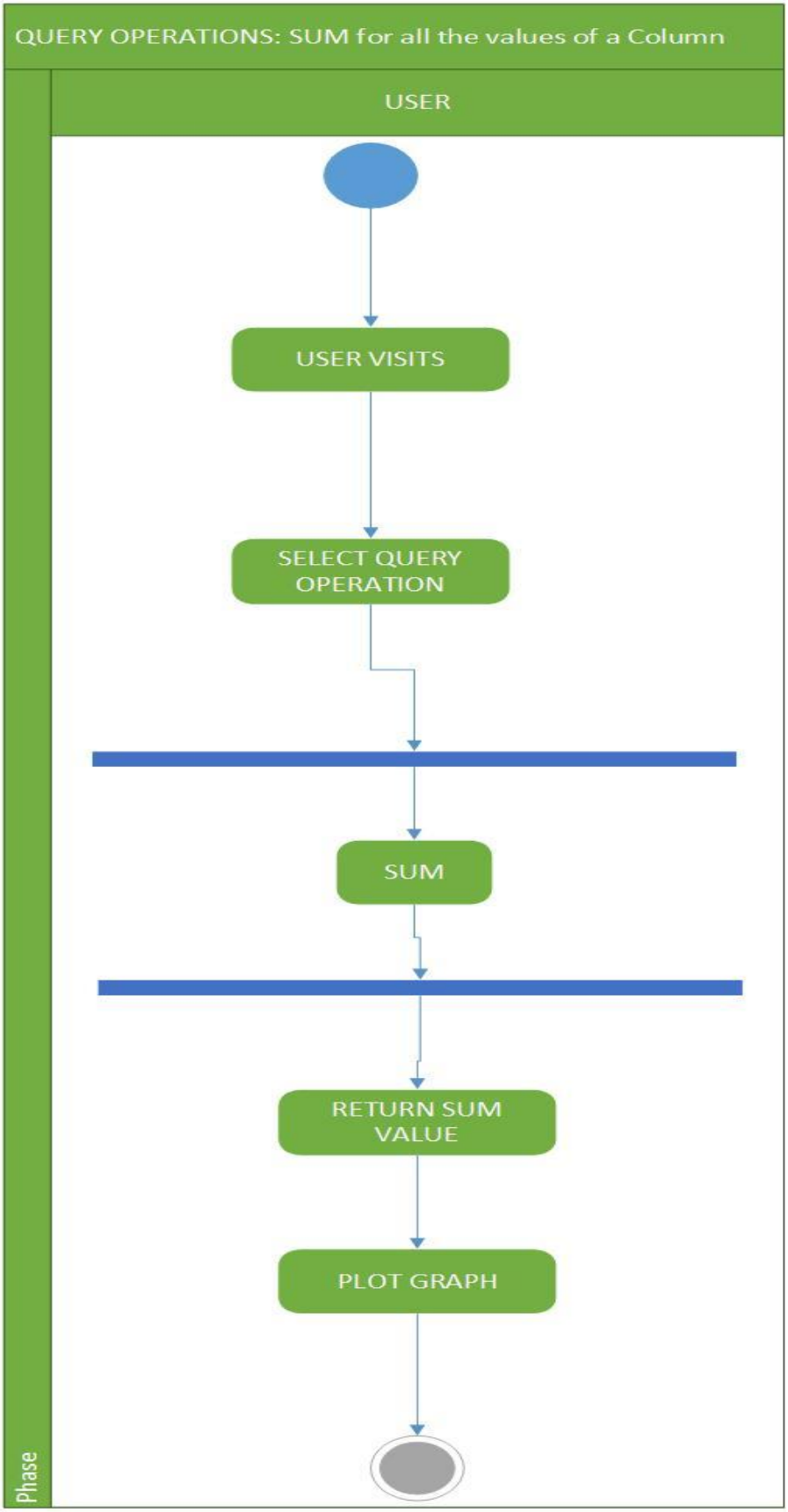


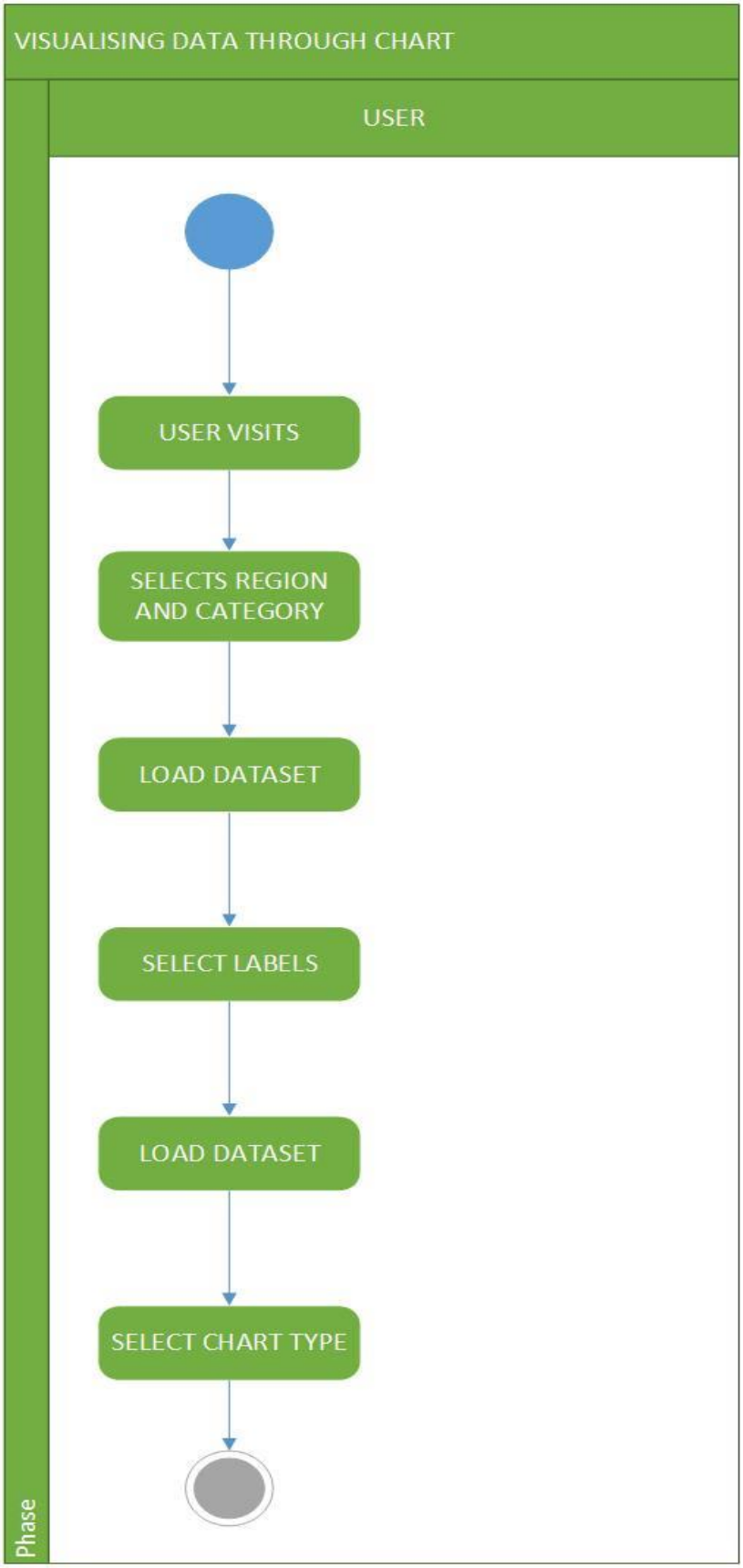
ACTIVITY DIAGRAMS

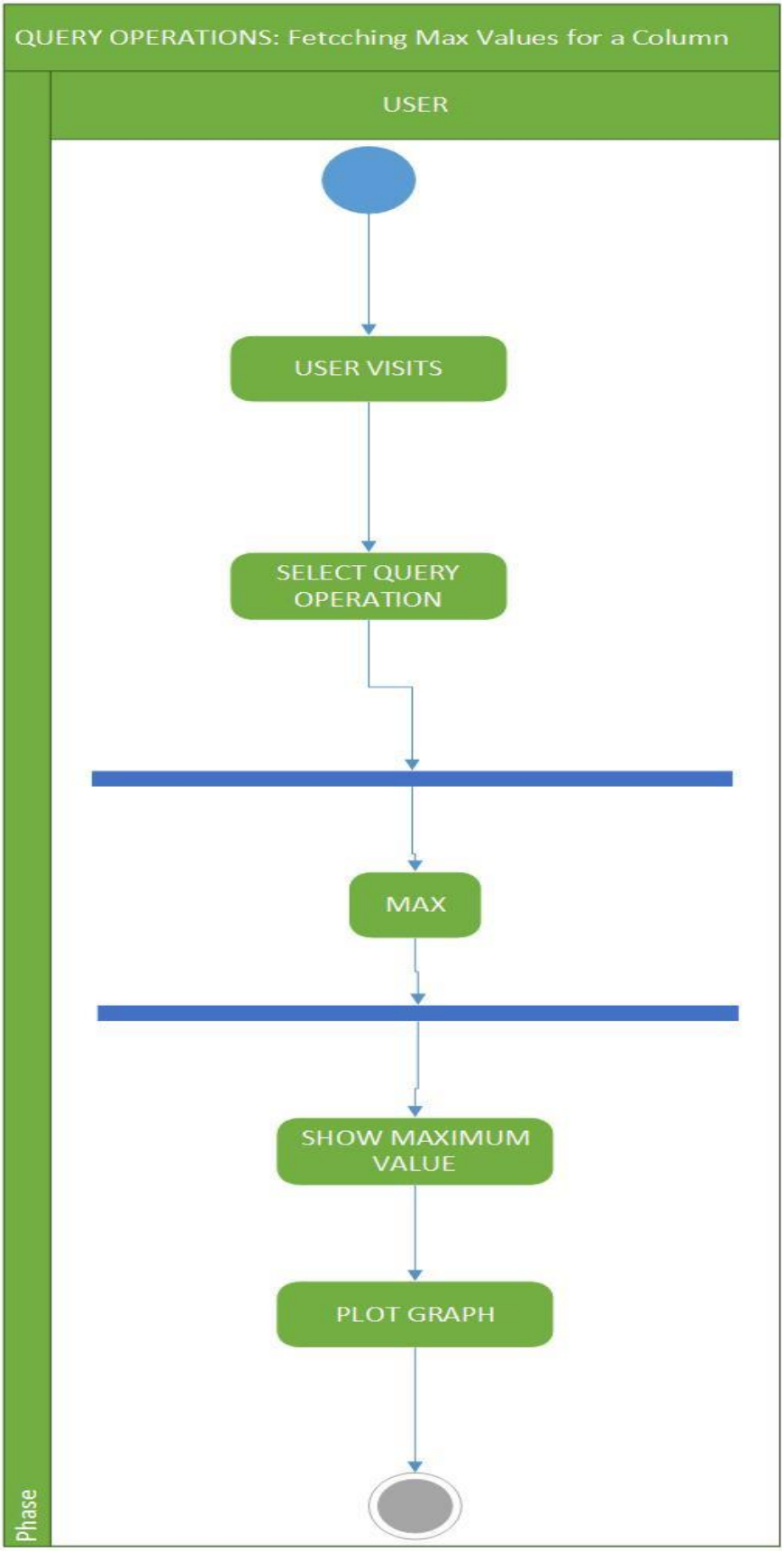


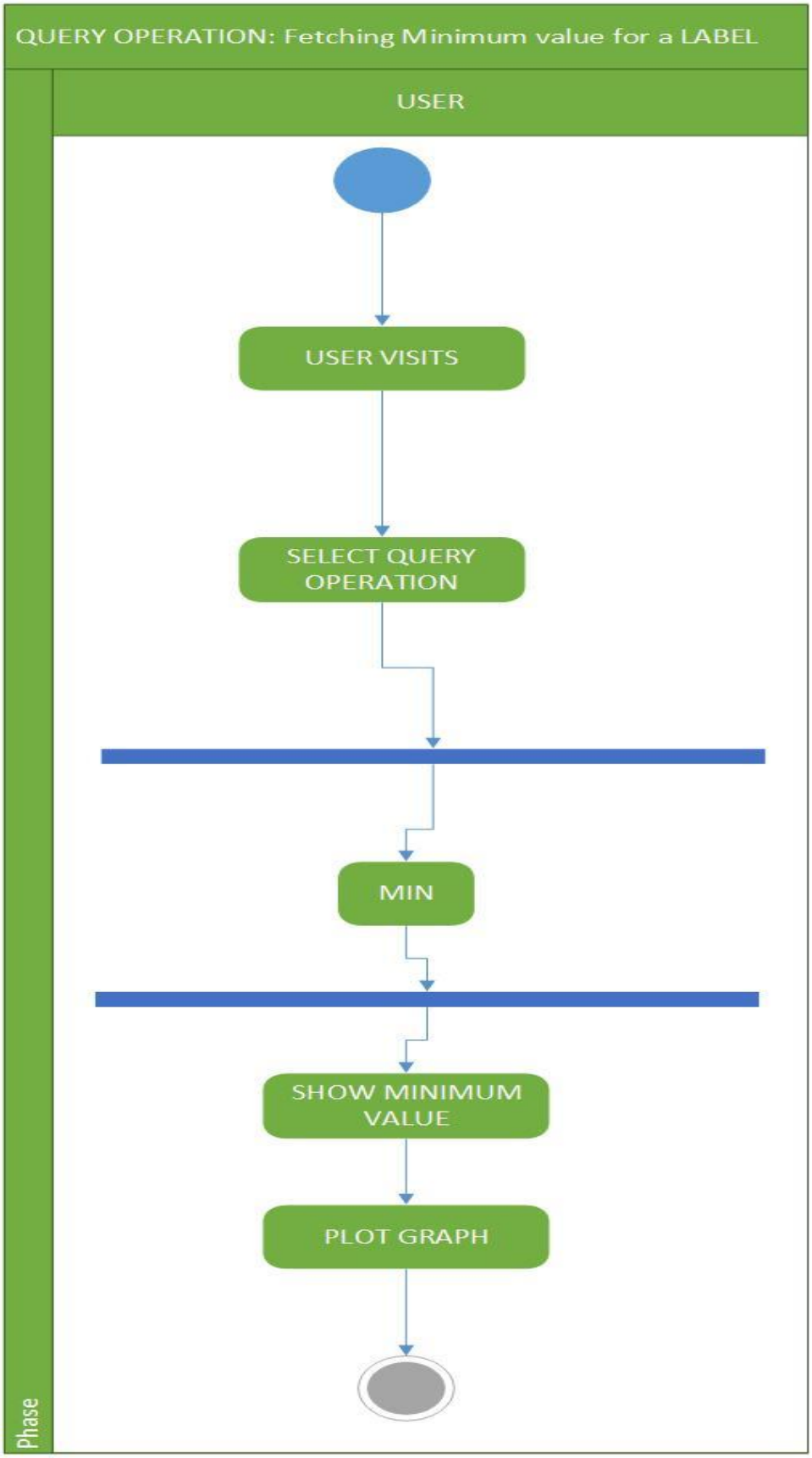

FILTERING DATA
USER
USER VISITS
FILTER DATSET
SELECT COUNTRIES
SELECT YEAR
PLOT GRAPH
Phase


QUERY OPERATIONS: Fetching Avg. Values for a Column
USER
USER VISITS
SELECT QUERY OPERATION
AVERAGE
RETURN AVG. VALUE
PLOT GRAPH
Phase

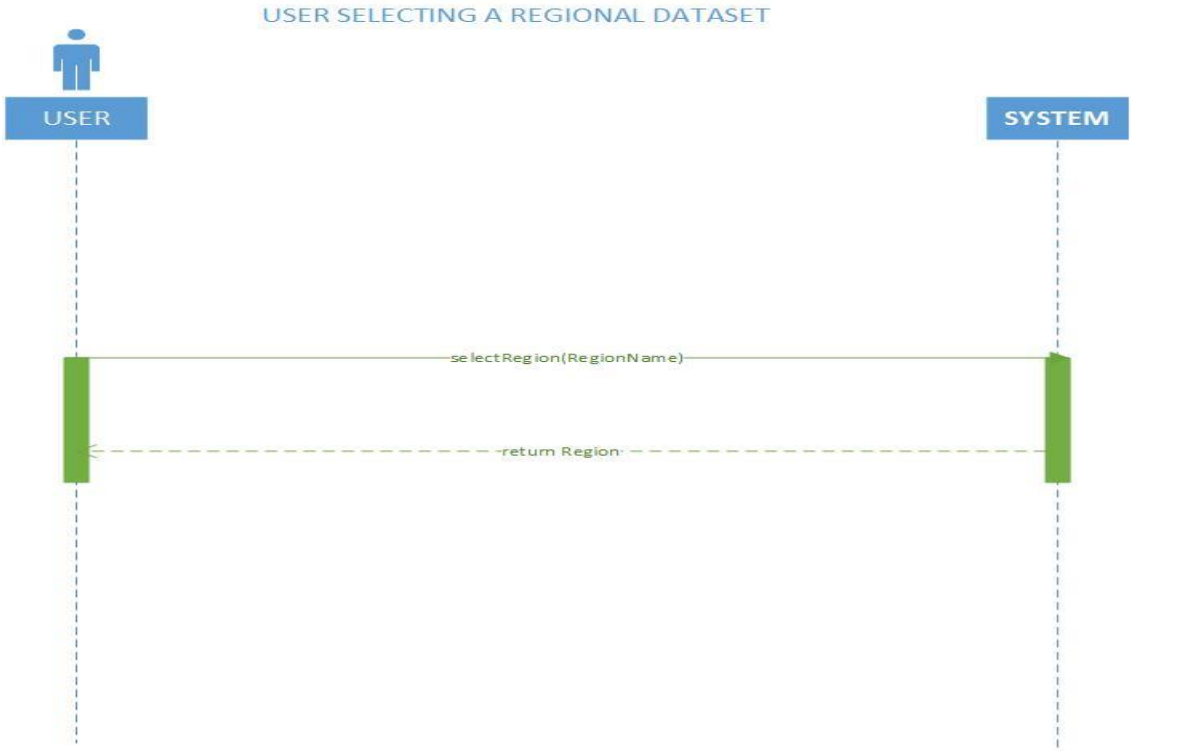

QUERY OPERATIONS: SUM for all the values of a Column
USER
USER VISITS
SELECT QUERY OPERATION
SUM
RETURN SUM VALUE
PLOT GRAPH
Phase

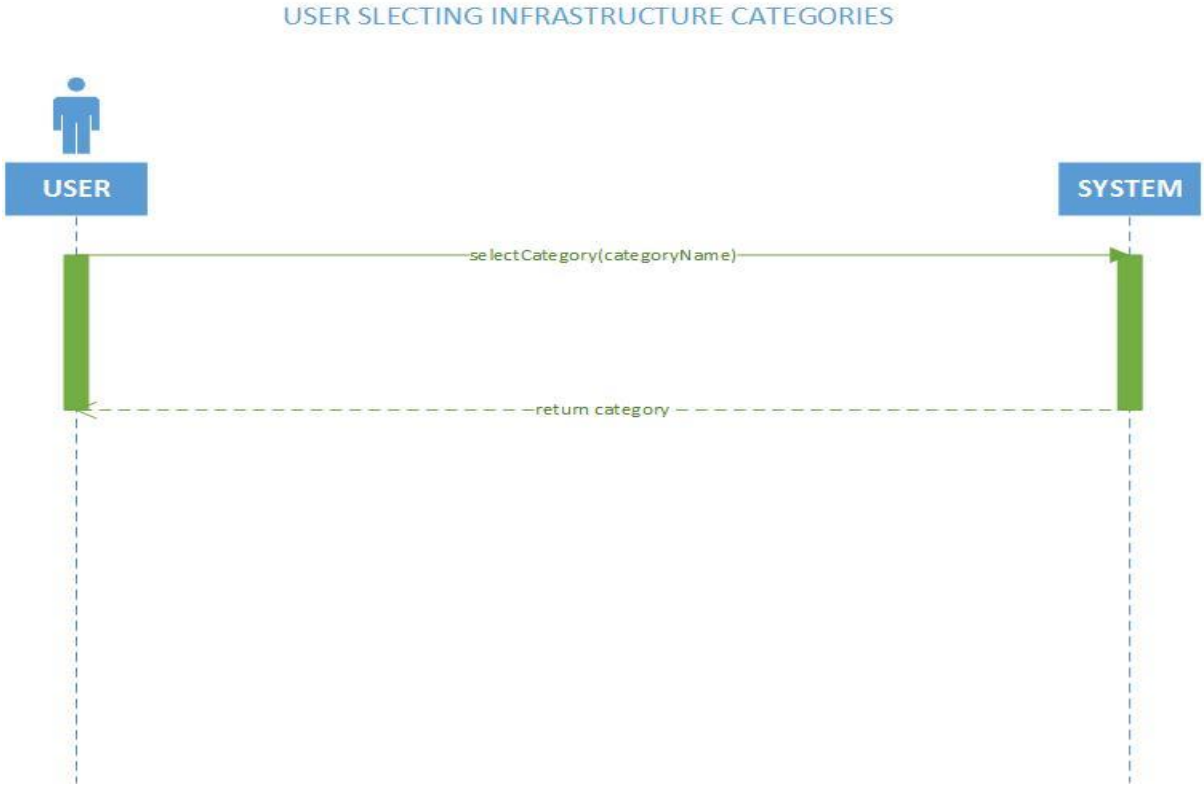

VISUALISING DATA THROUGH CHART
USER
USER VISITS
SELECTS REGION AND CATEGORY
LOAD DATASET
SELECT LABELS
LOAD DATASET
SELECT CHART TYPE
Phase


QUERY OPERATIONS: Fetcching Max Values for a Column
USER
USER VISITS
SELECT QUERY OPERATION
MAX
SHOW MAXIMUM VALUE
PLOT GRAPH
Phase

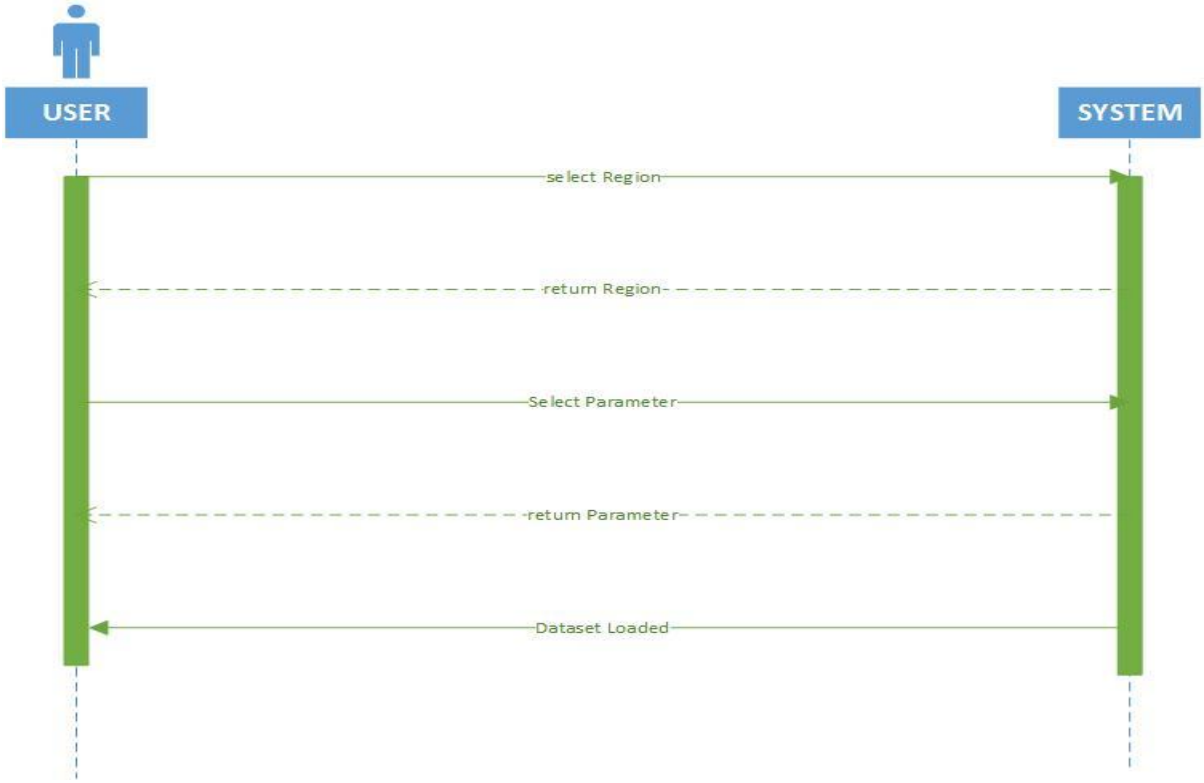

QUERY OPERATION: Fetching Minimum value for a LABEL
USER
USER VISITS
SELECT QUERY OPERATION
MIN
SHOW MINIMUM VALUE
PLOT GRAPH
Phase

SYSTEMS SEQUENCE DIAGRAM

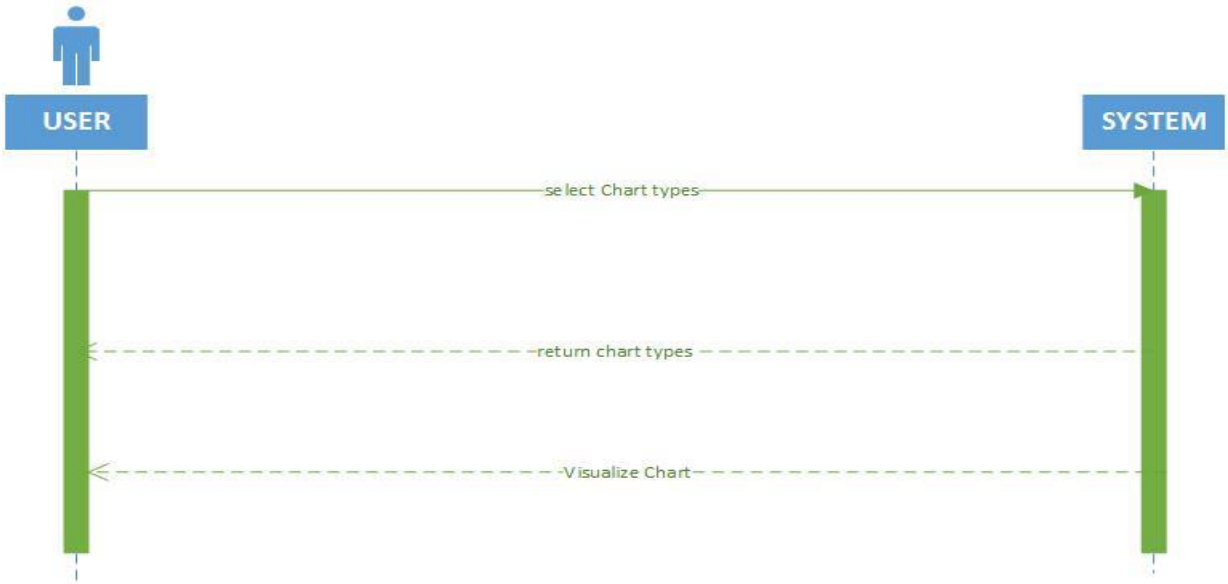




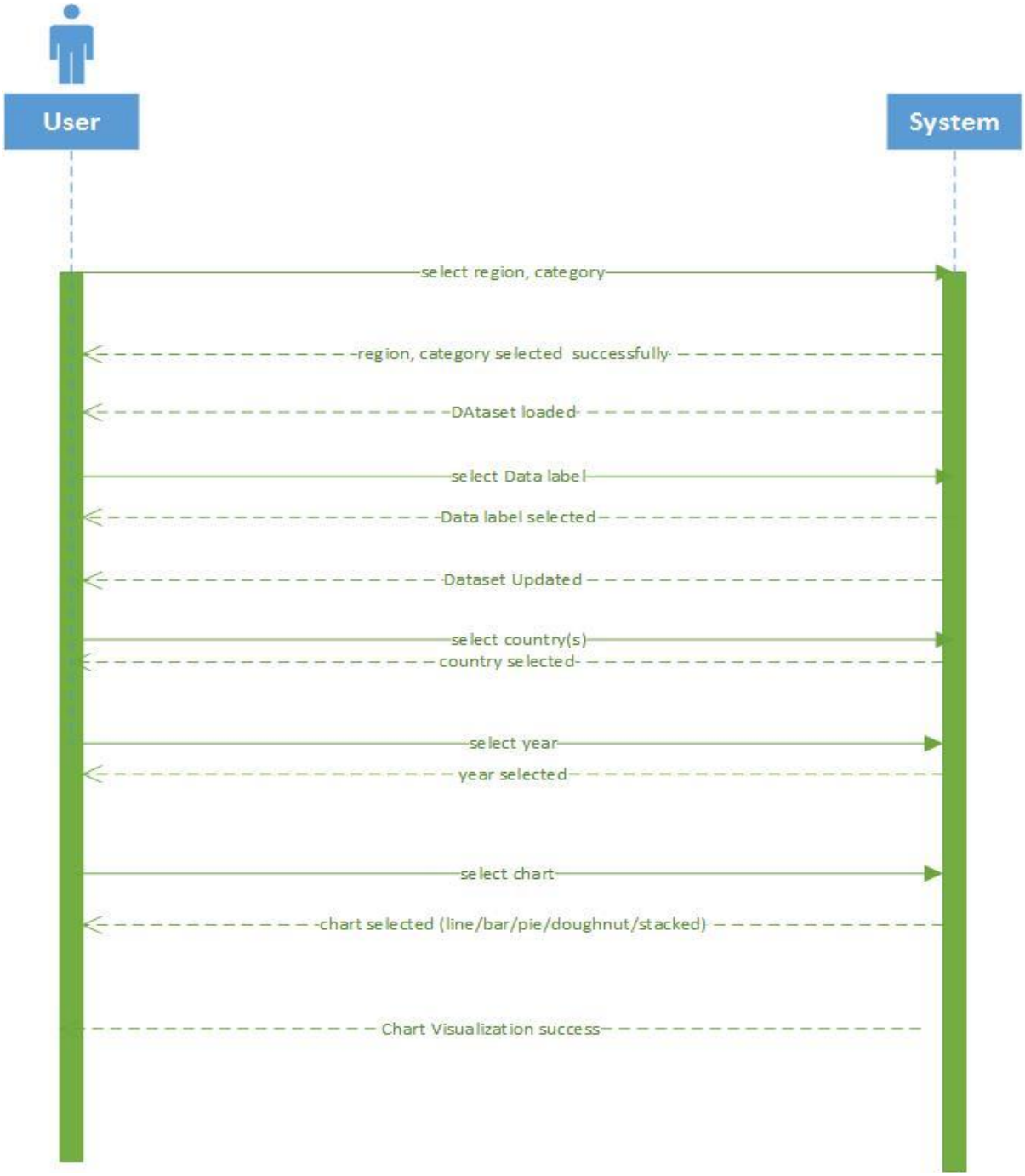
USER LOADING THE DATASET



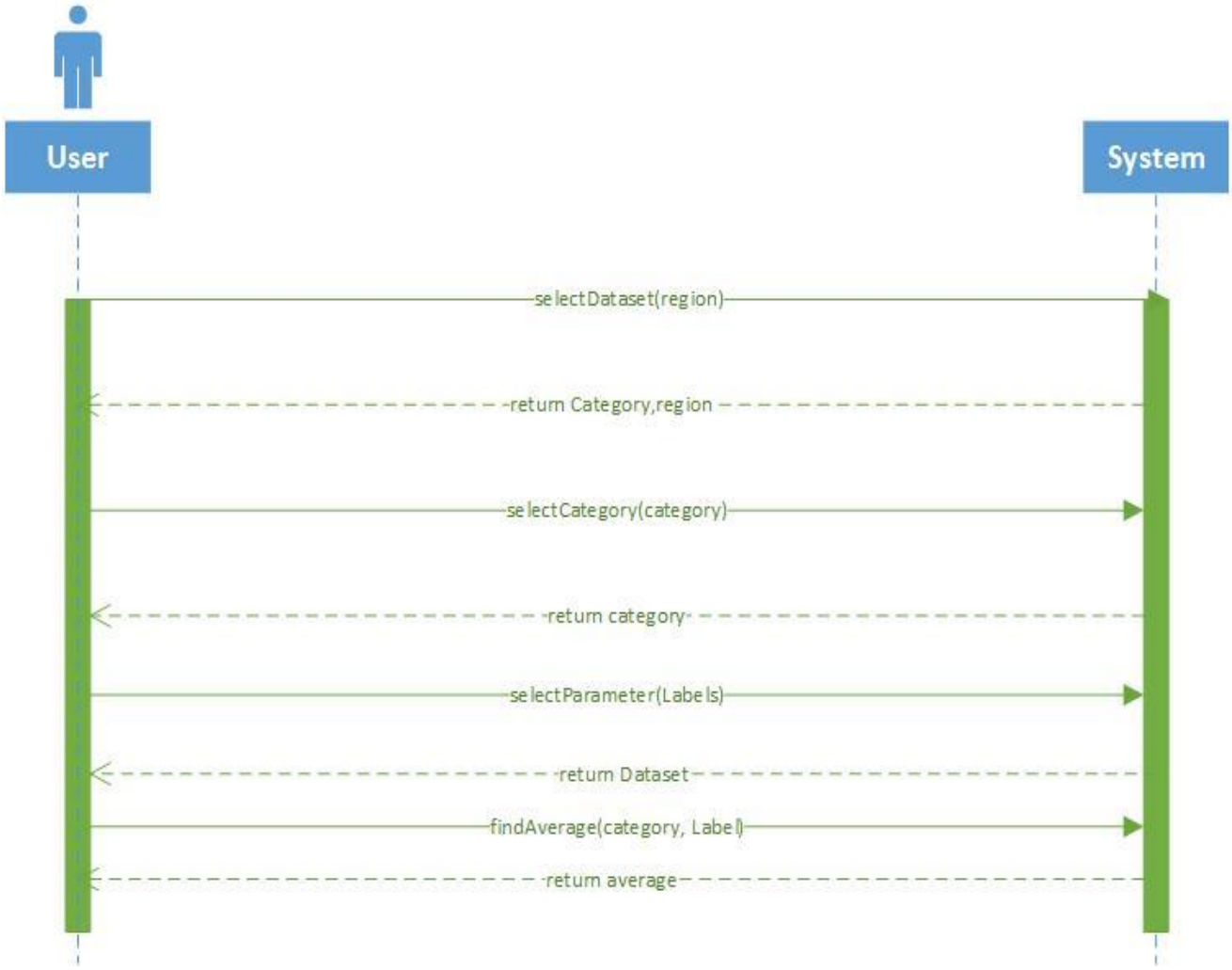
USER SELECTING THE CHART TYPE



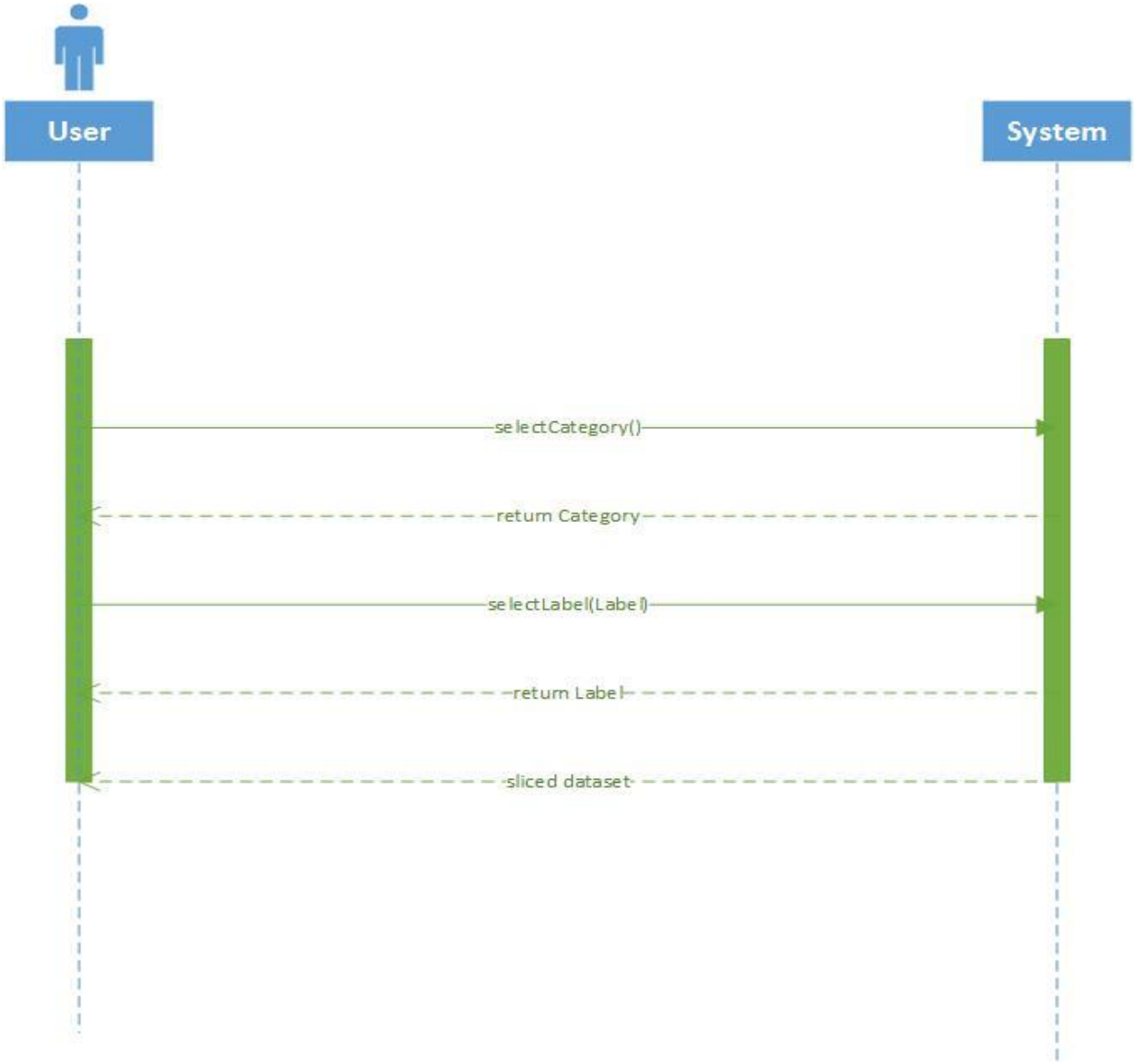
2. Dataset Visualization using different Charts



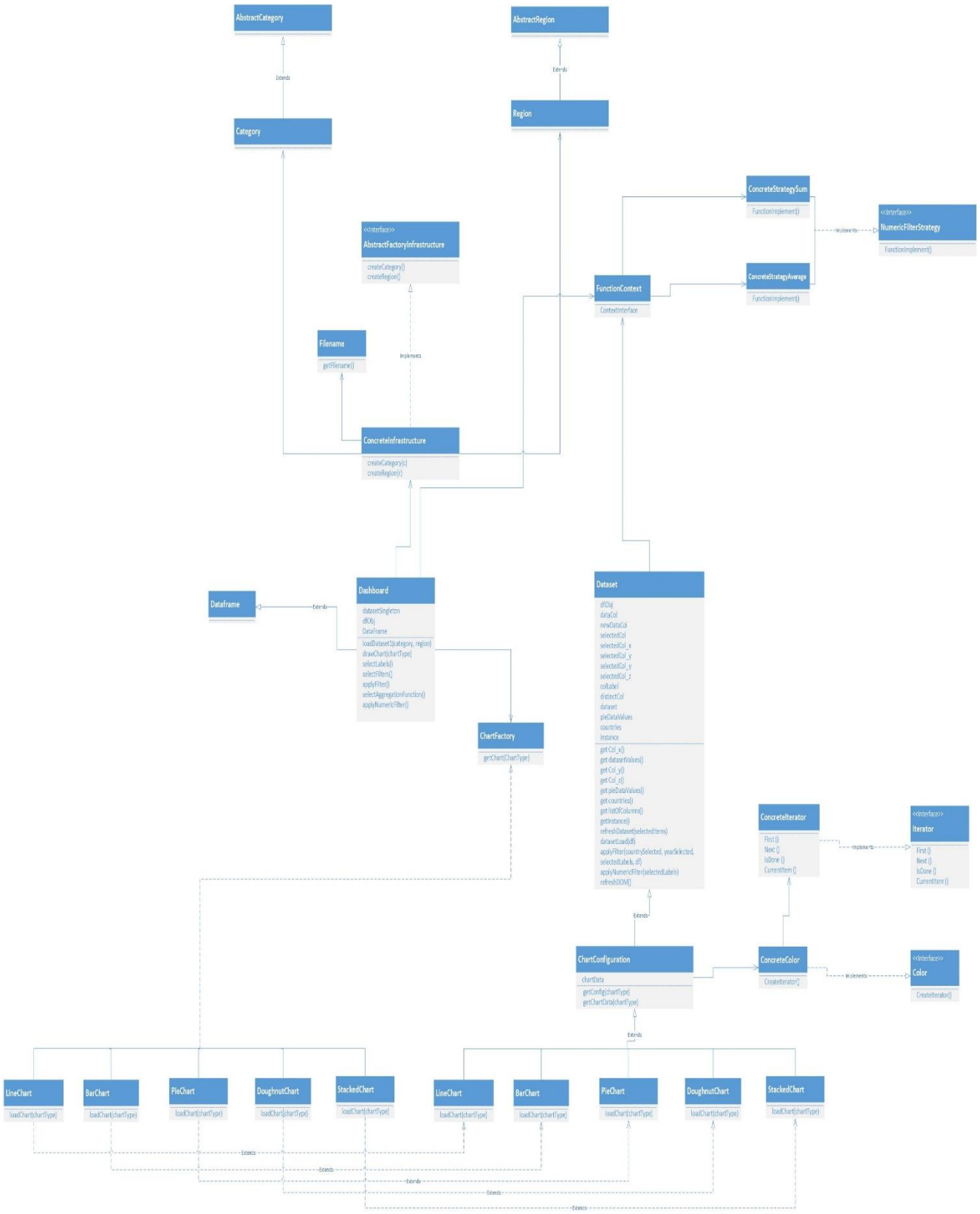
4. Find average of particular country on the basis of particular parameter



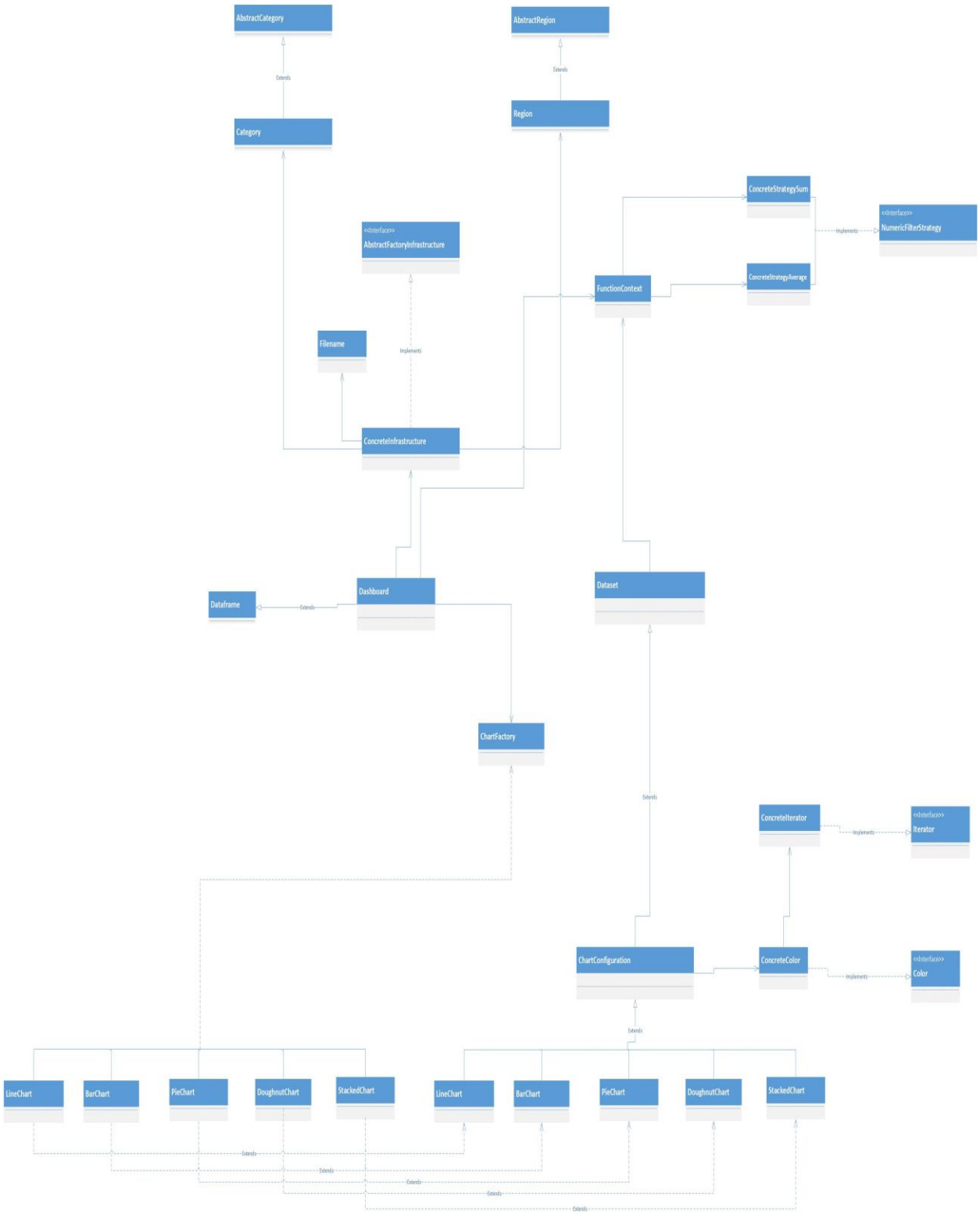
USER SLICING THE DATASET



DOMAIN MODEL CLASS DIAGRAM



DESIGN MODEL CLASS DIAGRAM

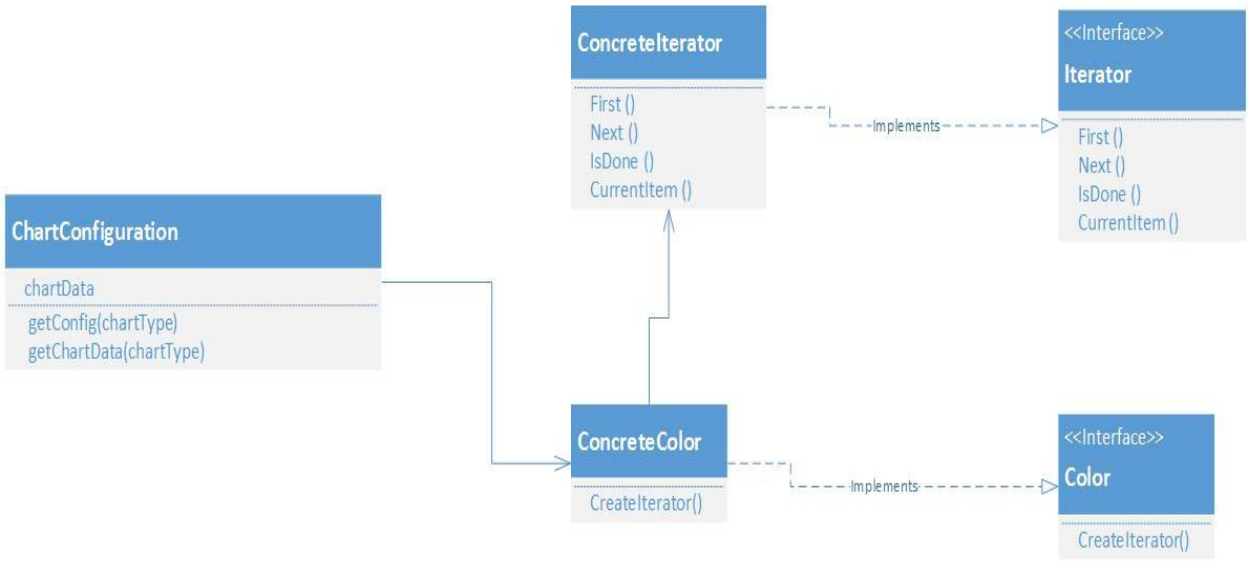


DESIGN PATTERNS

1. ITERATOR Design PATTERN

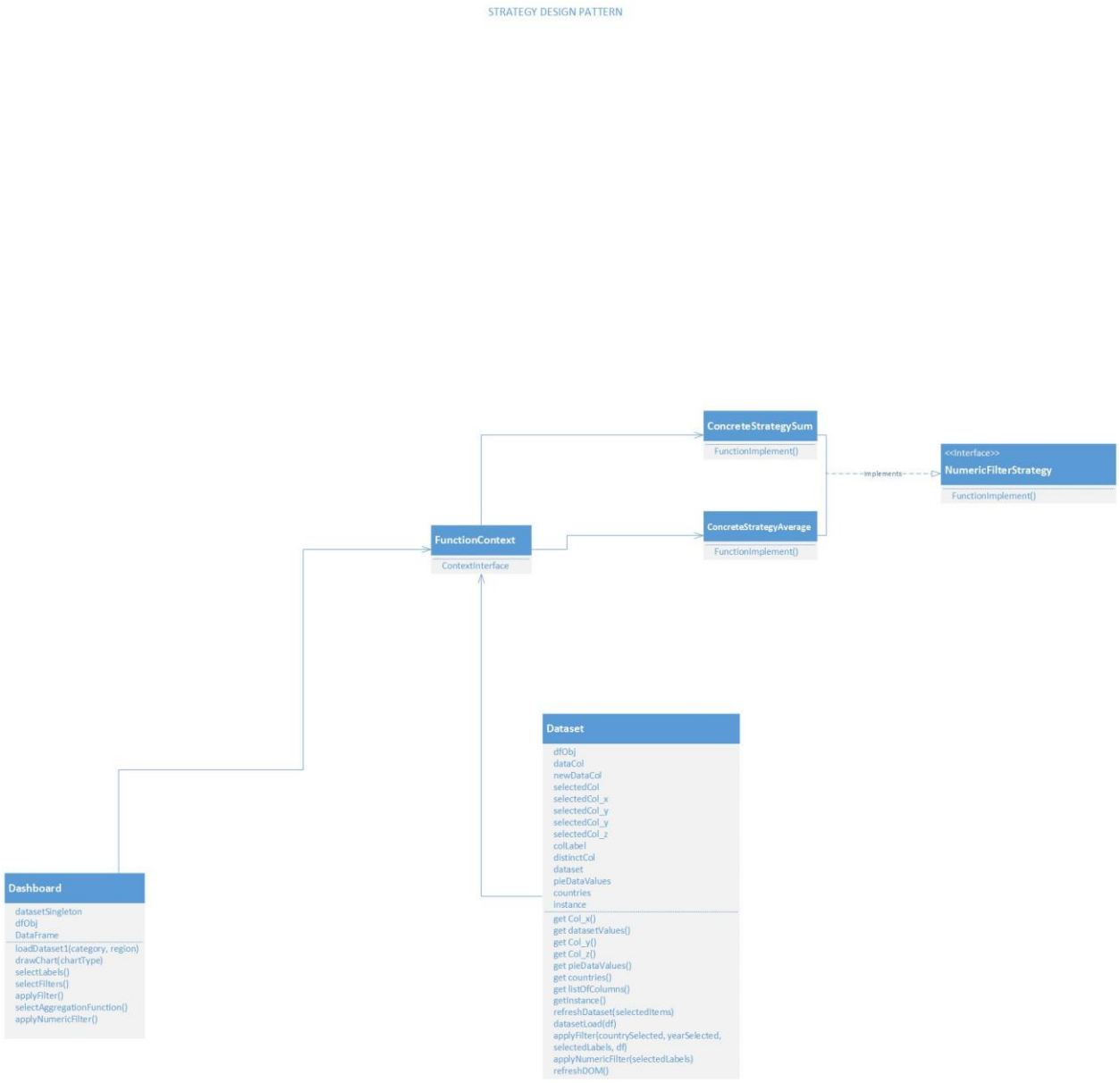
- The iterator design pattern is used when different types of collections are used and you need to traverse through all them. This helps in accessing collection object without exposing the underlying implementation.
- Here Iterator is like an interface and all the concrete classes of it implements that.

ITERATOR DESIGN PATTERN



2. STRATEGY DESIGN PATTERN

- In Strategy pattern, objects are created to represent various strategies and a context object whose behavior varies as per its strategy object.
- The NumericFilterStrategy class is an interface. If we introduce a new instance Dataset then it can be added as the subclass to NumericFilterStrategy interface without impacting the system.

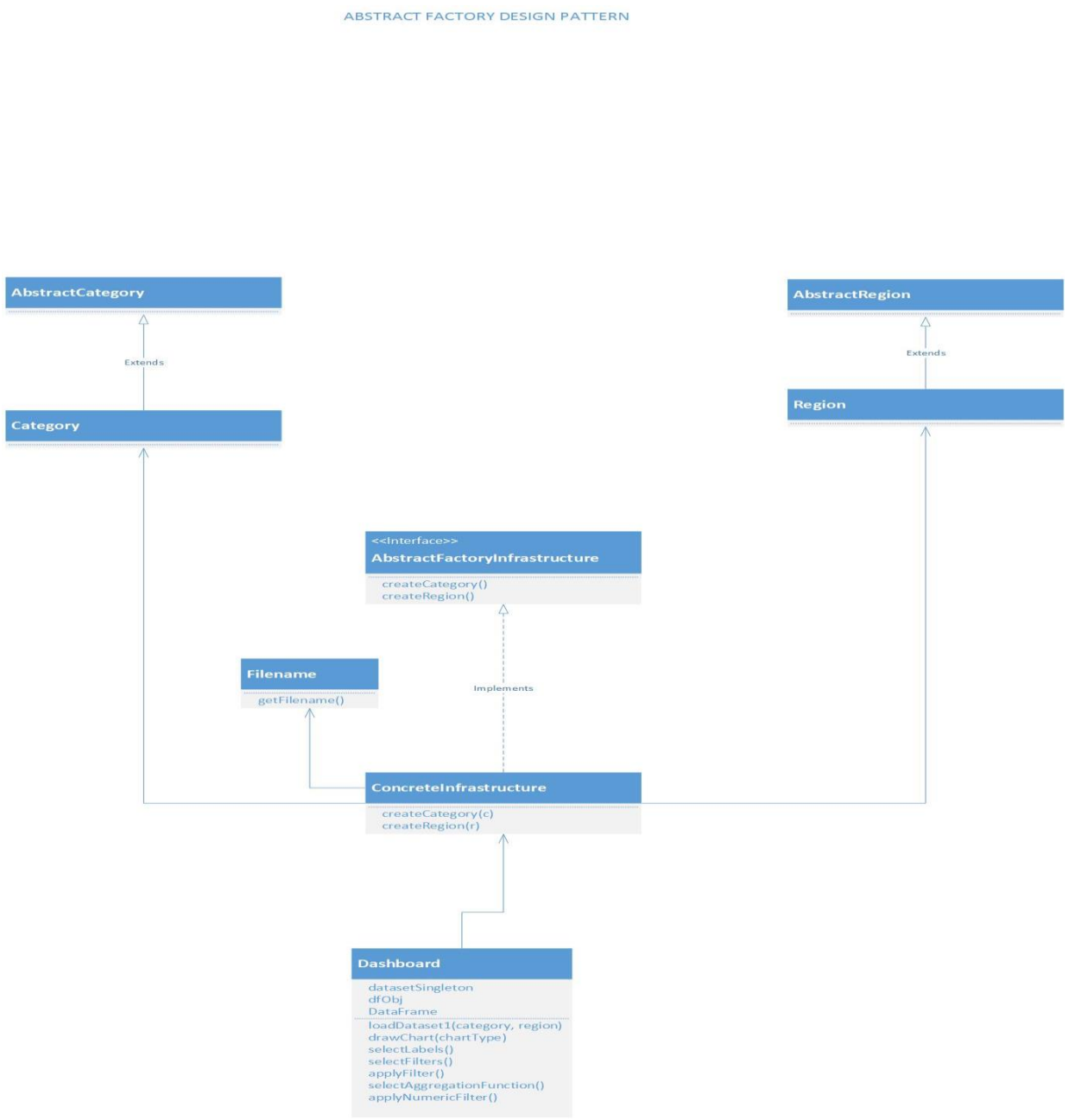


3. SINGLETON DESIGN PATTERN

- This pattern ensures that only one object is created during the class life cycle making it a Singleton Class.
- This class can be accessed by the following classes Dashboard, ChartFactory, ChartConfiguration, etc.
- The class contains a static method **getInstance()**.

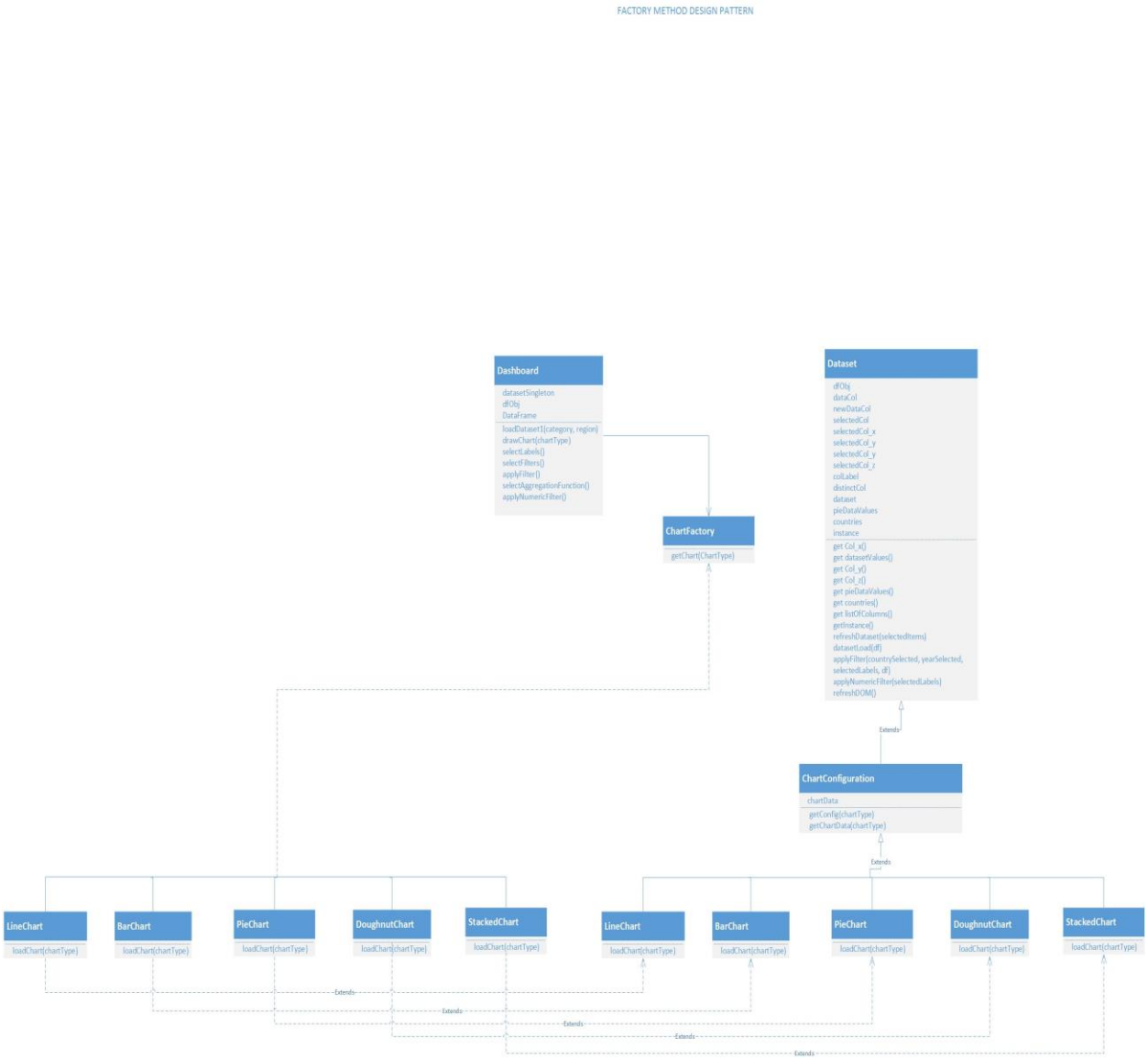
4.**ABSTRACT FACTORY DESIGN PATTERN:**

- Abstract Factory patterns work around a super-factory which creates other factories. This factory is also called as factory of factories. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.
- In Abstract Factory pattern an interface is responsible for creating a factory of related objects without explicitly specifying their classes. Each generated factory can give the objects as per the Factory pattern.



5. FACTORY METHOD DESIGN PATTERN:

- Factory pattern is one of the most used design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.
- In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.



MVC PATTERN