Process MeNtOR 3.0

Uni-SEP

Country Statistics Data Analysis & Visualization System

Design Document

Version:	2.1
Print Date:	
Release Date:	
Release State:	
Approval State:	
Approved by:	
Prepared by:	
Reviewed by:	
Path Name:	
File Name:	
Document No:	

object oriented pty. ltd.

Document Change Control

Version	Date	Authors	Summary of Changes
1.1	2021-03-11	Yiran Shao; Junyi Yang; Zheng Yang; Rui Zhu	Added part 1: Introduction
1.2	2021-03-13	Yiran Shao; Junyi Yang; Zheng Yang; Rui Zhu	Added part 2-3: Major Design Decision and Component Diagram
1.3	2021-03-15	Yiran Shao; Junyi Yang; Zheng Yang; Rui Zhu	Added part 4-5: Detailed UML Class Diagram and use of Design patterns.
1.4	2021-03-22	Yiran Shao; Junyi Yang; Zheng Yang; Rui Zhu	Added part 6-7: Activity plans and Test cases. Finalized document.

Document Sign-Off

Name (Position)	Signature	Date
Yiran Shao	Yiran Shao	2021-03-22
Junyi Yang	Junyi Yang	2021-03-22
Zheng Yang	Zheng Yang	2021-03-22
Rui Zhu	Rui Zhu	2021-03-22

Contents

1 Introduction

1.1 Purpose

This document detailed the requirements of the system Country Statistics Data Analysis & Visualization System.

This document will outline the software design and specifications of the data analysis and visualization system in addition to system architecture, system components, and software requirements developed by the project team based on the provided project description.

1.2 Overview

The SDD document contains the following information:

1. Component Diagram of the system (Architecture). The level of detail and granularity will be at the Java Package level of detail.
2. Detailed Class Diagram for all the created or modified classes in the project. The detail class diagram contains UML notation for each class and its data and methods.

1.3 Resources - References

This SDD follows the design specification specified in

CS2212B-Winter-2020-2021-Project-Description

Country-Statistics-Data-Analysis & Visualization-System-Requirements-Model

References for different technologies used in the project are listed below:

World Bank's API:
https://datahelpdesk.worldbank.org/knowledgebase/articles/898581

JavaAPI:
http://api.worldbank.org/v2/country/can/indicator/SP.POP.TOTL?date=2000:2001&format=json

Draw.io: https://app.diagrams.net/

UMLet: https://www.umlet.com/

2 Major Design Decisions

1. The Country Statistic Data Analysis System uses the MVC architecture style to decouple data models and user interfaces. The code is divided into three parts: model, viewer, and controller.
2. The three parts all utilize the Singleton design pattern to ensure only one instance of these objects are present at an instance in time.
3. Using the observer pattern to decouple each class.
4. Using the proxy pattern to decouple the data model from the operation on the data model.
5. Using mediator pattern to decouple UI from operations on UI.

3 Architecture

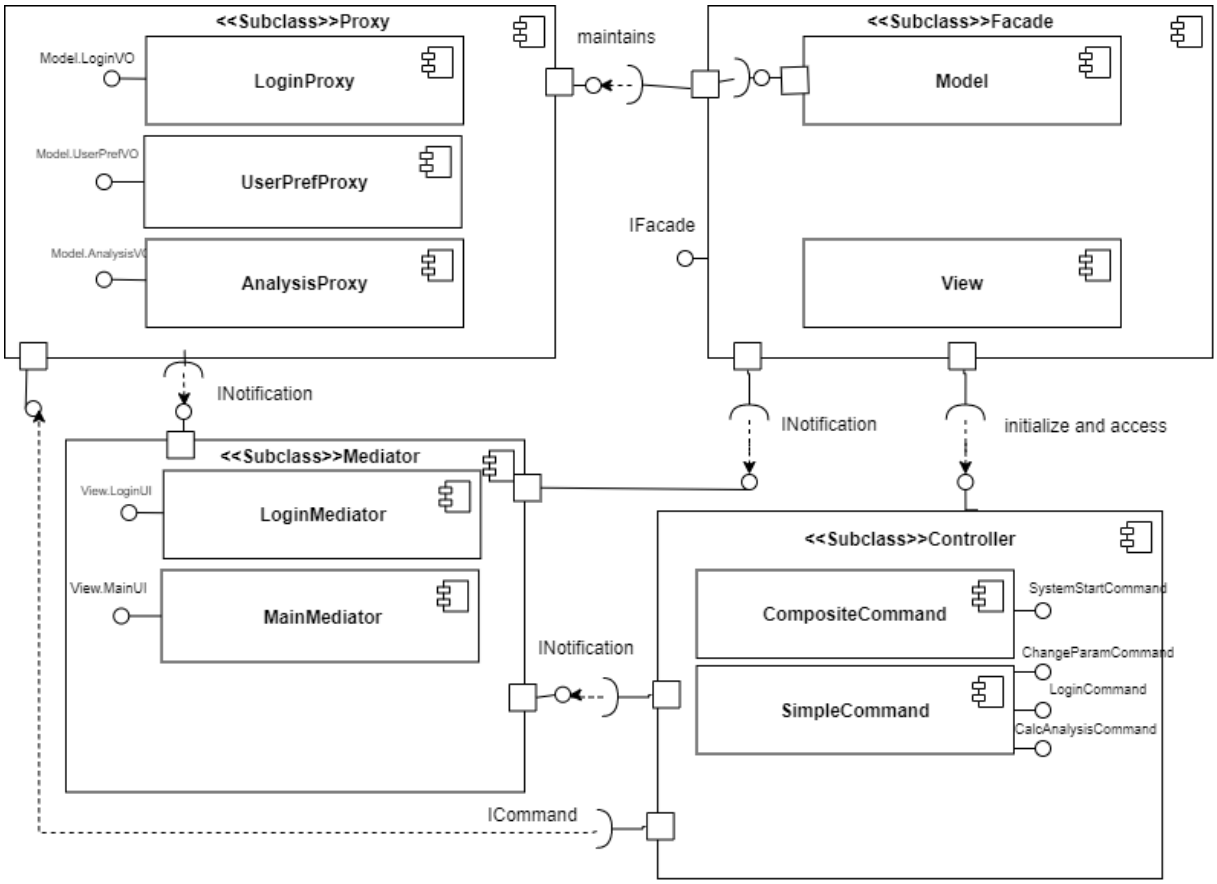

Fig. 3.1 UML Component Diagram

The software architecture pattern employed in this system is built with the call-and-return and hierarchical architectural styles. There are 4 subclasses in the system: Proxy, Controller, Mediator and Facade. Proxies maintain and manipulate specific parts of the data model. Proxies send INotification to the Mediator, require ICommand from the Controller and are maintained by the Facades. Controllers manipulate all the commands by adding, removing or executing them based on the notification received. Controllers provide inference to initialize and access Facade, require iNotification from the Mediator and iCommand from the Proxy. Mediators maintain all the UI panels, listen to all the notifications sent within the UIs and provide notification to other parts of the program. The Mediators send out INotification to all the other components. Facade provides a way of communication between other subclasses of the program. Facades is able to maintain the Proxies, initialize and access the Controllers, send notifications to the Mediators and provide an extra interface iFacade that give access to some public methods.

Above is the package level component diagram of the system (Fig.3.1). Please note that due to space constraints only a highlight of interface methods is shown. Detailed interface methods can be found in the class details section below.

4 Detailed Class Diagrams

4.1 UML Class Diagrams

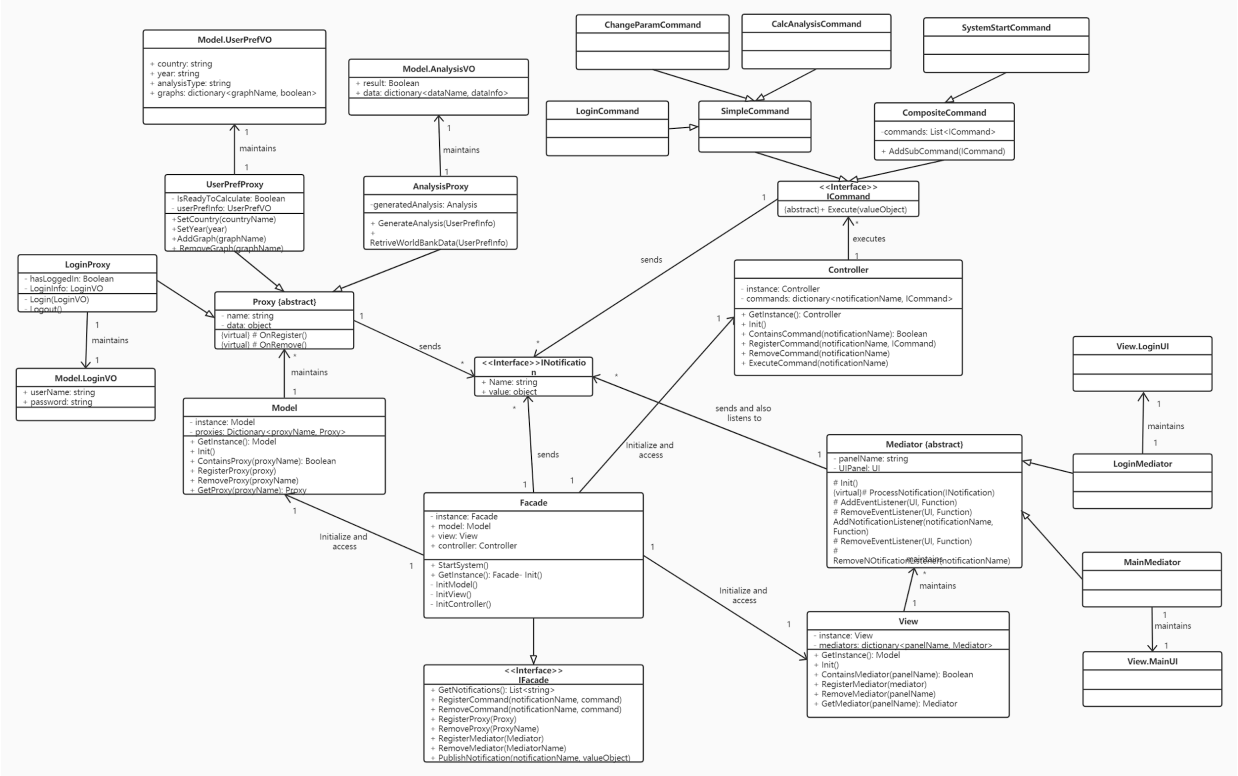

Fig. 4.1 UML Class Diagram

The application is designed in a classic MVC architecture style, which divides the code into 3 layers: model, view and controller.

For the model layer, the Model singleton holds references for all registered proxies. Each proxy holds a reference to a specific part of the data model and is responsible to maintain and manipulate it. The real data object can only be accessed by proxies.

For the controller layer, the Controller singleton holds references for all registered notificationName-command binding. Commands can only be added, removed and executed by Controller. We use commands to implement all the business logic of the application such as calculate a data analysis based on some data, create a new graph, and so on. There are 2 types of commands: SimpleCommand can only be executed, and compositeCommand can contain multiple commands and can be executed sequentially.

For the View layer, the View singleton holds references for all registered mediators. Each mediator takes care of one specific UI panel. For example, loginMediator maintains the LoginPanel and MainMediator maintains the main panel. Mediators can listen to both notifications and UI events and then process them separately.

Notifications are used to communicate between classes. Notifications can be sent by proxies, commands and mediators but can only be listened to and processed by mediators.

Facade provides several easy-to-use APIs that give access to some public methods of Model, View and Controller. Proxies, Mediators and Commands can use these APIs to access and communicate with each other.

Facade Class Details

Class Name	Facade
Method	+ **GetNotifications(): List<string>**: return a list of all notifications that can be sent. + **RegisterCommand(notificationName, command):** Binds a notification name with a command. + **RemoveCommand(notificationName, command) :** Remove a notification-command binding. + **RegisterProxy(Proxy):** Register a new proxy that maintains and manipulates value objects. + **RemoveProxy(ProxyName):** Remove a proxy by its name. + **RegisterMediator(Mediator):** Register a new mediator that listens to and manipulates UI panels. + **RemoveMediator(MediatorName):** Remove a mediator by its name. + **PublishNotification(notificationName, valueObject):** Publish a notification together with some information. + **GetInstance()**: Return the singleton Facade instance.

Proxy Class Details

Class Name	Proxy
Method	{virtual} # OnRegister(): This method is automatically called when creating a proxy. {virtual} # OnRemove(): This method is automatically called when removing a proxy.

Model Class Details

Class Name	Model
Method	+ **GetInstance(): Model :** Return the singleton Model instance. + **Init():** Initialize model. + **ContainsProxy(proxyName): Boolean:** Check if a specific proxy exists. + **RegisterProxy(proxy):** Register a new Proxy. + **RemoveProxy(proxyName):** Remove a proxy by its name. + **GetProxy(proxyName):** Get a proxy by its name.

Controller Class Details

Class Name	Controller
Method	+ **GetInstance(): Controller:** return the singleton Controller Instance. + **Init(): Initialize controller.** + **ContainsCommand(notificationName): Boolean:** Check if a notification name is binding with any command. + **RegisterCommand(notificationName, ICommand):** Bind a command with a notification. + **RemoveCommand(notificationName):** Remove a notification-command binding. + **ExecuteCommand(notificationName):** Execute a command binding with notification name.

Command Class Details

Class Name	Command
Method	**{virtual}+ Execute(valueObject): A virtual method that can be override by any commands to perform a specific operation.**

View Class Details

Class Name	View
Method	+ **GetInstance(): Model:** Get the singleton View instance. + **Init():** Initialize view. + **ContainsMediator(panelName): Boolean:** Check if a mediator exist by panel name. + **RegisterMediator(mediator):** Register a mediator that listens to and manipulates a specific UI panel. + **RemoveMediator(panelName):** Remove a mediator by panel name. + **GetMediator(panelName): Mediator:** Get a mediator by panel name.

Mediator Class Details

Class Name	Mediator
Method	# **Init():** Initialize the mediator **{virtual}# ProcessNotification(INotification):** When receiving a notification, process it based on its name. # **AddEventListener(UI, Function):** Add a UI event listener such that when a user interacts with a UI element, process it based on user input. # **AddNotificationListener(notificationName, Function):** Add a notification listener such that when this specific notification is published, it can be received and processed in the ProcessNotification method. # **RemoveEventListener(UI, Function):** Remove a UI event listener. # **RemoveNotificationListener(notificationName):** Remove a notification listener.

5 Use of Design Patterns

1. Singleton pattern:

   Model, View, Controller and Facade are all singleton classes and are not allowed to be explicitly instantiated.

2. Proxy pattern:

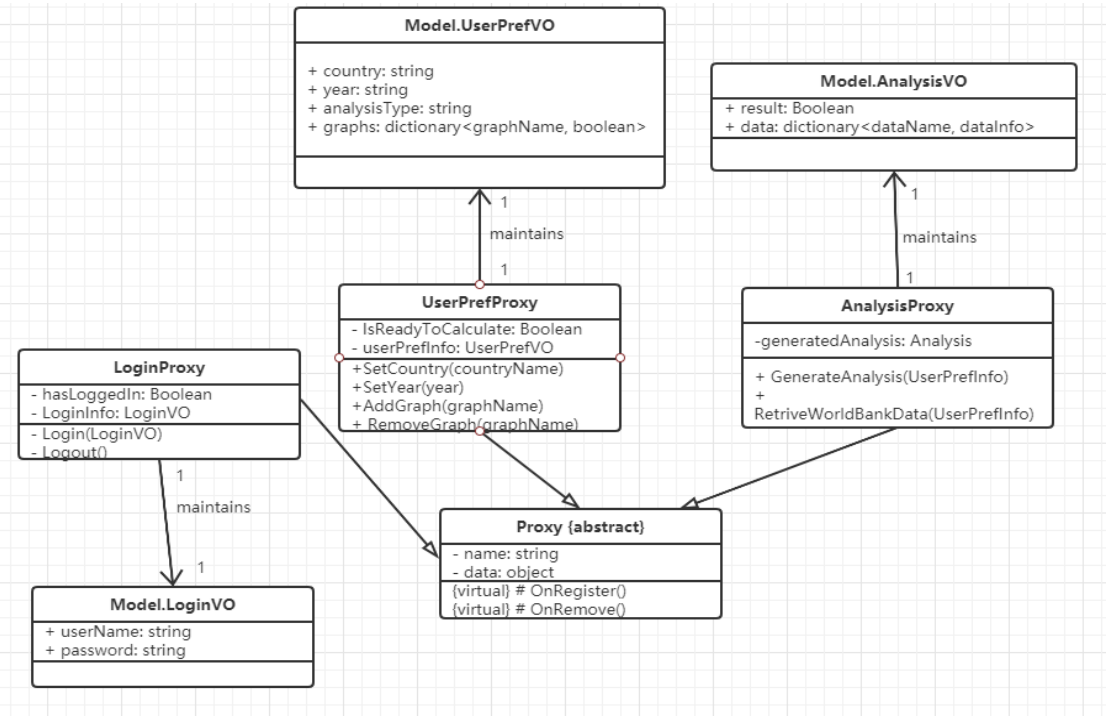

Fig 5.1 UML Class Diagram for Proxy

Proxy pattern is used and each proxy maintains and manipulates a portion of the data model of the application. For example, LoginProxy maintains a data object called LoginVO. Once the LoginProxy is registered by Model, it initializes the LoginVO by reading all username-password combinations from a Json file and stores them into LoginVO. stores pairs of username and password, it has a public method login (tryLoginVO) which compares the LoginVO that is passed as a parameter with the loginVO it maintains, a notification will then be published once the LoginVO is verified.

3. Mediator pattern:

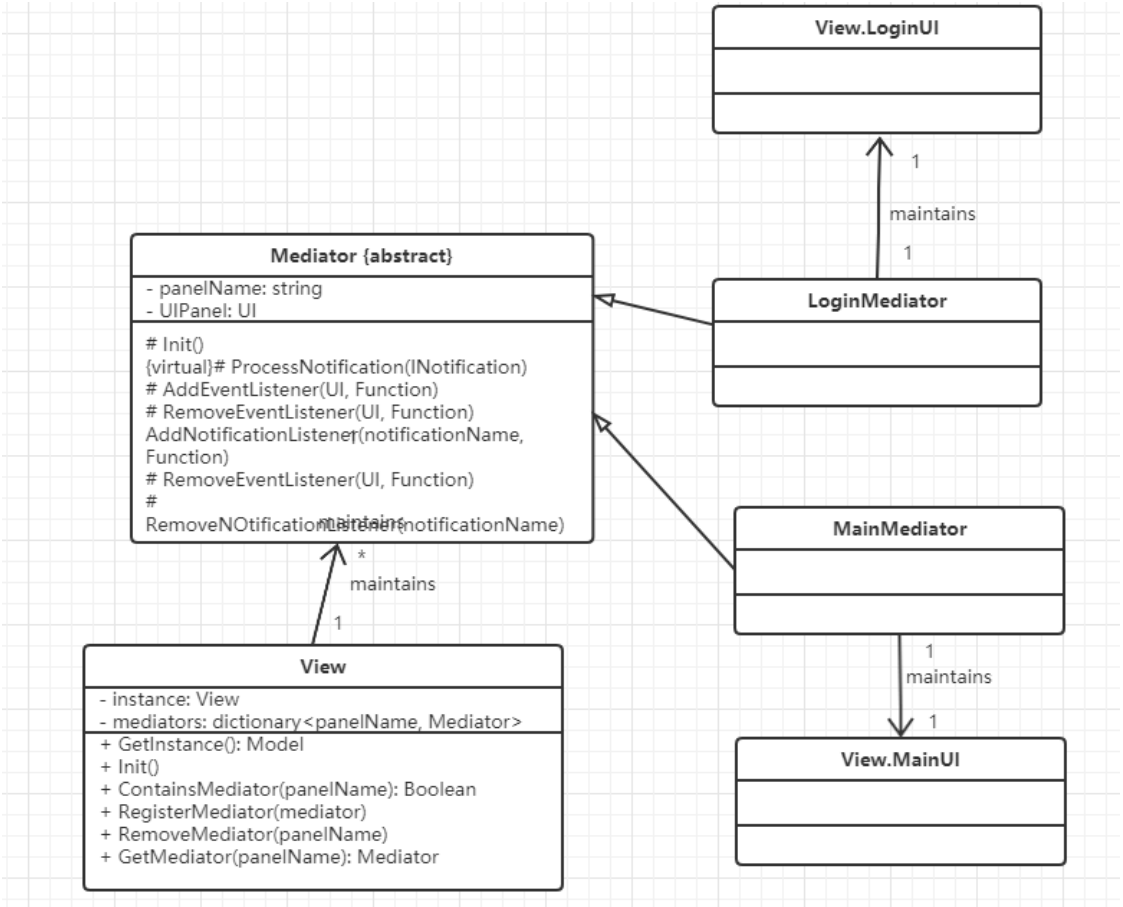

Fig 5.1 UML Class Diagram for Mediator

Mediator pattern is used and each mediator listens to and manipulates a specific UI panel. For example, once the user clicks the “login” button, LoginMediator will collect the user input and publish a notification with the name “tryLogin”. Mediators also listen to notifications and process them. For example, LoginMediator listens to the “LoginResult” notification and manipulates the login panel based on the login result.

4. Facade pattern:

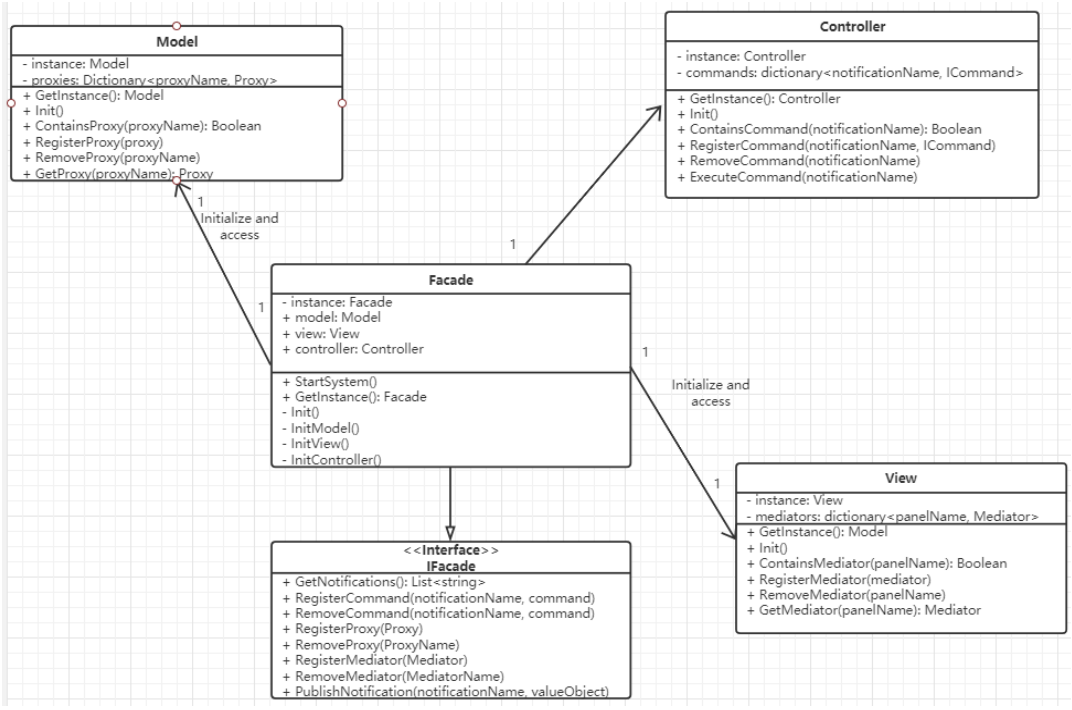

Fig 5.1 UML Class Diagram for Facade

Facade pattern is used. In this application, Facade is a singleton which provides several easy-to-use APIs that give access to some public methods of Model, View and Controller. Proxies, Mediators and Commands can use these APIs to access and communicate with each other.

5. Command pattern:

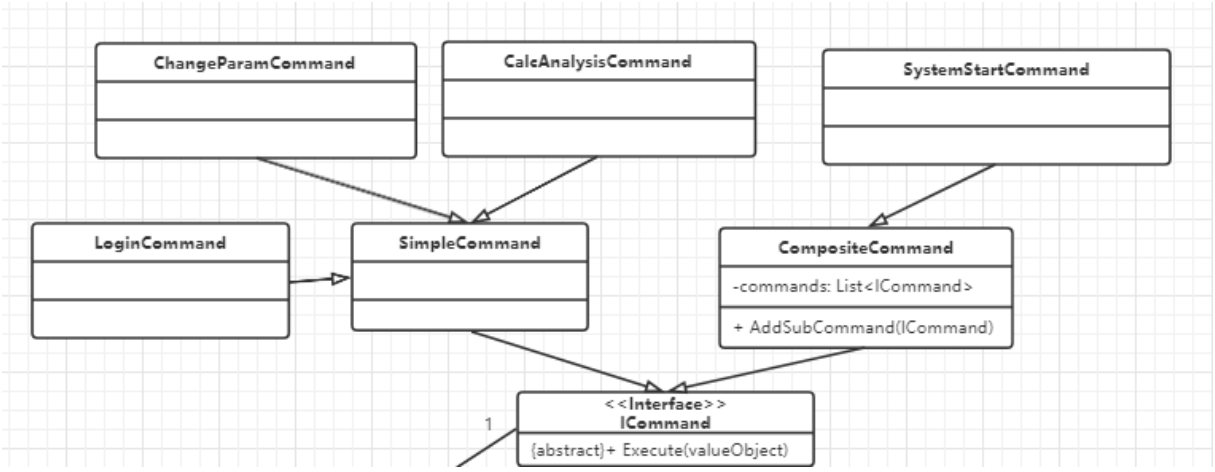

Fig 5.1 UML Class Diagram for Command

Command pattern is used. Each command implements a specific business logic. For example, CalcAnalysisCommand calculates a data analysis based on some data and LoginCommand implements all the business logic of user login.

6. Observer pattern

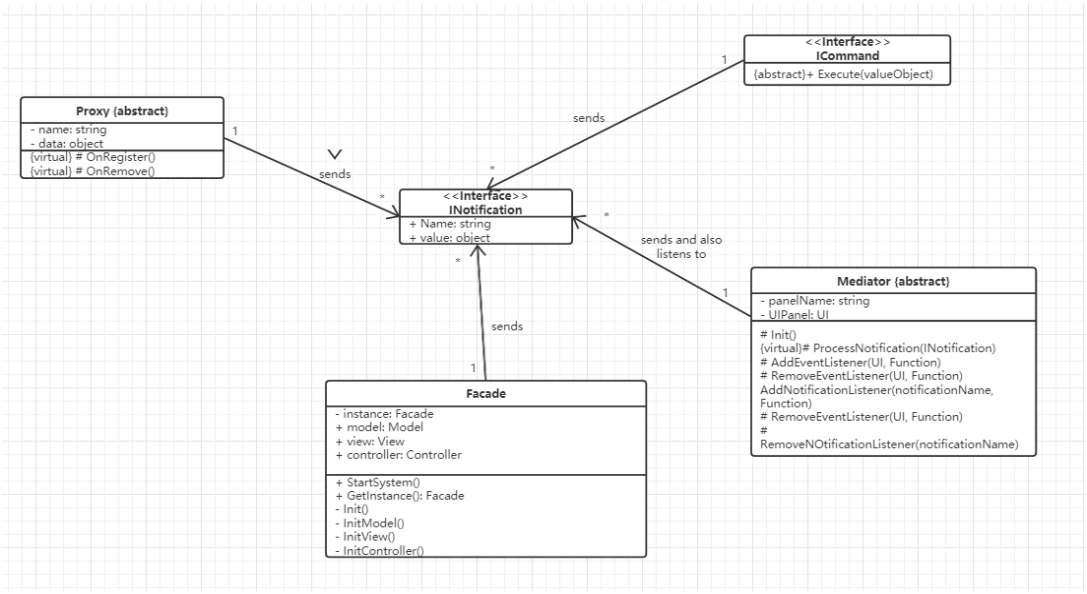

Fig 5.1 UML Class Diagram for Observer

Observer patterns are heavily used in the system to achieve decoupling between classes. Notifications can be published by proxies, commands, mediators. Mediators have to use the AddNotificationListener(notificationName) method to listen to a specific notification and then process it once it is published.

6 Activities Plan

1.1 Project Backlog and Sprint Backlog

In this Section, and assuming you follow a Scrum process model, provide a list of product backlog items so that you can select items for your Sprint backlog. Make sure the product backlog list and the tasks in each product backlog item are consistent with the Gantt Chart in Section 6.1. above.

Story	Estimation	Priority	Status
As a user I want to login to the system.	2	1	To be started
As a user I want to see the list of types of viewers to select from.	3	2	To be started
As a user I want to select a particular type to analyze data.	1	3	To be started
As a user I want to see the list of countries available to select from.	2	4	To be started
As a user I want to select a particular country to analyze data.	1	5	To be started
As a user I want to see the list of years available to select from.	2	6	To be started
As a user I want to select a particular year to analyze data.	1	7	To be started
As a user I want to see the reports and graphs representing the selected data.	4	8	To be started
As a user I want to add viewers.	2	9	To be started
As a user I want to remove viewers.	2	10	To be started

As a user I want to see the error message when I enter incorrect credentials or perform an invalid operation.	1	11	To be started.
Total	21		

1.2 Group Meeting Logs

In this Section you write minutes of each meeting, listing the attendance, what the topics of discussion in the meeting were, any decisions that were made, and which team members were assigned which tasks. These minutes must be submitted with the project report in each deliverable and will provide input to be used for the overall assessment of the project.

Present Group Members	**Meeting Date**	**Issues Discussed / Resolved**
Yiran Shao; Junyi Yang; Zheng Yang; Rui Zhu	26 Jan, 2021	1. Meeting was held through online-phone call. 2. General discussion about the project. 3. Created a timetable for future meetings. 4. Brainstormed the product backlog list for this project. 5. Decided to meet again on 1st February.
Yiran Shao; Junyi Yang; Zheng Yang; Rui Zhu	1 Feb, 2021	1. Meeting was held through online-phone call. We, 2. assigned parts for each group member; which parts may often connect between assigned members; 3. Discussed what actors or functionalities we may need in our project -- relevant diagrams should come out before the next meeting; which parts may require extra collaborations between members; 4. Decided to meet again on 5th February.
Yiran Shao; Junyi Yang; Zheng Yang;	5 Feb, 2021	1. Meeting was held through online-phone call.

Rui Zhu		A generalized version of the actor diagram and user case came out. 2. We added some extra actor type of managers for some invocation of functionality. 3. Administrator actors were removed from our actor diagrams. 4. Some analysis of user cases used in each scenario came out.
Yiran Shao; Junyi Yang; Zheng Yang; Rui Zhu	7 Feb, 2021	1. Meeting was held through online-phone call. During this meeting, 2. Went through all the details that were mentioned in the assignment description. 3. Finalized the deliverable for the first assignment.
Yiran Shao; Junyi Yang; Zheng Yang; Rui Zhu	15 Feb, 2021	1. Meeting was held through online-phone call. During this meeting, 2. Went through the requirements for part 2 of the SRS document 3. Assigned development tasks to group members 4. Decided to meet again on Feb 27
Yiran Shao; Junyi Yang; Zheng Yang; Rui Zhu	27 Feb, 2021	1. Meeting was held through online-phone call. During this meeting, 2. Went through the completed sections of SRS document part 2 3. Assigned evaluation tasks to group members 4. Decided to finalize the document on March 1
Yiran Shao; Junyi Yang; Zheng Yang; Rui Zhu	1 Mar, 2021	1. Meeting was held through online-phone call. During this meeting, 2. Re-evaluated completed and modified sections of SRS document part 2 3. Modified file format 4. Submitted SRS document part 2
Yiran Shao; Junyi Yang; Zheng Yang; Rui Zhu	15 Mar, 2021	1. Meeting was held through online-phone call. During this meeting, 2. Broke-down requirements of SDD document 3. Assigned tasks to group members

		4. Decided to meet again to check task progress on March 19
Yiran Shao; Junyi Yang; Zheng Yang; Rui Zhu	19 Mar, 2021	1. Meeting was held through online-phone call. During this meeting, 2. Evaluated current progress of SDD document 3. Modified task-assignment 4. Decided to meet again to finalize the document on March 22

2 Test Driven Development

Test ID	0001
Category	evaluation of user credentials stored on DB
Requirements Coverage	UC1-Successful-User-Login
Initial Condition	the system has been initiated and runs
Procedure	1. The user selects login 2. The user provides a user name 3. The user provides a password 4. The user logs-in into the system and is presented with the main UI window
Expected Outcome	the login UI closes, and the user is greeted with the main UI panel
Notes	the user should provide a password with at least 8 characters and contains only letters and digits.

Test ID	0002
Category	evaluation of user credentials stored on DB
Requirements Coverage	UC1-Unsuccessful-User-Login
Initial Condition	the system has been initiated and runs
Procedure	1. The user selects login 2. The user provides a combination of username and password that is not stored in the database. 3. The user receives an error message and exits the system.
Expected Outcome	the system terminates.
Notes	the user should provide a password with at least 8 characters and contains only letters and digits.

Test ID	0003
Category	evaluation of user credentials stored on DB
Requirements Coverage	UC1-Empty-User-Login
Initial Condition	the system has been initiated and runs
Procedure	1. The user selects login

	2. The user provides empty username and password and the Login button is pressed. 3. The user receives an error message and exits the system.
Expected Outcome	the system terminates.
Notes	

Test ID	0004
Category	evaluation of analysis types that can be performed by system
Requirements Coverage	UC2-Selection-of-a-New-Analysis-Type
Initial Condition	the user has logged into the system.
Procedure	1. The user selects analysis type. 2. The user selects an analysis type that is different than their previous selection. 4. The viewer list on the main UI is emptied.
Expected Outcome	The viewer list on the main UI is emptied.
Notes	The user can only select analysis types that are provided and implemented by the system.

Test ID	0005
Category	evaluation of analysis types that can be performed by system
Requirements Coverage	UC2-Selection-of-an-Existing-Analysis-Type
Initial Condition	the user has logged into the system.
Procedure	1. The user selects analysis type. 2. The user selects an analysis type that is the same with their previous selection.
Expected Outcome	None.
Notes	The user can only select analysis types that are provided and implemented by the system.

Test ID	0006
Category	evaluation of analysis types that can be performed by system

Requirements Coverage	UC2-Selection-of-Empty-Analysis-Type
Initial Condition	the user has logged into the system.
Procedure	1. The user does not select analysis type. 2. The user presses the Recalculate button 3. The user receives an error message
Expected Outcome	None.
Notes	The user cannot Calculate if the analysis type is not selected.

Test ID	0007
Category	evaluation of the available countries stored in the DB.
Requirements Coverage	UC3-Selection-of-an-Available-Country
Initial Condition	the user has selected the analysis type to be performed.
Procedure	1. The user selects a country from the list which data is available for currently selected analysis type. 2. The user presses the Recalculate button
Expected Outcome	None.
Notes	The user can only select countries that are provided and implemented by the system.

Test ID	0008
Category	evaluation of the unavailable countries stored in the DB.
Requirements Coverage	UC3-Selection-of-an-Unavailable-Country
Initial Condition	the user has selected the analysis type to be performed.
Procedure	1. The user selects a country from the list which data is available for currently selected analysis type. 2. The user presses the Recalculate button 3. The user receives an error message and is asked to re-select the country.
Expected Outcome	The user receives an error message.
Notes	The user can only select countries that are provided in the list.

Test ID	0009
Category	evaluation of the system's behaviour when country is not selected
Requirements Coverage	UC3-Selection-of-an-Empty-Country
Initial Condition	the user has selected the analysis type to be performed.
Procedure	1. The user does not select country. 2. The user presses the Recalculate button 3. The user receives an error message and is asked to select the country.
Expected Outcome	The user receives an error message.
Notes	The user can only select countries that are provided in the list.

Test ID	0010
Category	evaluation of the available years stored in the DB.
Requirements Coverage	UC4-Selection-of-an-Available-Duration
Initial Condition	the user has selected the country to perform the analysis on.
Procedure	1. The user selects a starting and ending year from the list which data is available for currently selected analysis type and country. 2. The user presses the Recalculate button
Expected Outcome	None.
Notes	The user can only select years that are provided and implemented by the system.

Test ID	0011
Category	evaluation of the available years stored in the DB.
Requirements Coverage	UC4-Selection-of-an-Unavailable-Duration
Initial Condition	the user has selected the country to perform the analysis on.
Procedure	1. The user selects a starting and ending year from the list which data is unavailable for currently selected analysis type and country. 3. The user presses the Recalculate button 2. The user receives an error message and is asked to re-select the years.
Expected Outcome	The user receives an error message

Notes	The user can only select years that are provided and implemented by the system.

Test ID	0012
Category	evaluation of the system's behaviour when during is not selected
Requirements Coverage	UC4-Selection-of-an-Unavailable-Duration
Initial Condition	the user has selected the country to perform the analysis on.
Procedure	1. The user does not select start or ending year. 2. The user presses the Recalculate button 2. The user receives an error message and is asked to select the country.
Expected Outcome	The user receives an error message
Notes	The user can only select years that are provided and implemented by the system.

Test ID	0013
Category	evaluation of Adding or Removing Viewers
Requirements Coverage	UC5-Adding-Viewer
Initial Condition	the user has selected the analysis type.
Procedure	1. The user selects a graph that is in the available graph list. 2. The user presses the “add” button. 3. The corresponding graph is added to the viewer list.
Expected Outcome	None
Notes	The user can only select graphs that are compatible with the current analysis type.

Test ID	0014
Category	evaluation of Adding or Removing Viewers
Requirements Coverage	UC5-Adding-Viewer
Initial Condition	the user has selected the analysis type.
Procedure	1. The user selects a viewer from the drop-down menu that is not in the available graph list

	2. The user presses the “add” button
Expected Outcome	The user receives an error message
Notes	The user can only select graphs that are compatible with the current analysis type.

Test ID	0015
Category	evaluation of Adding or Removing Viewers
Requirements Coverage	UC6-Remove-Viewer
Initial Condition	the user has selected the analysis type.
Procedure	1. The user selects a graph from the graph list. 2. The user presses the “remove” button. 3. The corresponding graph is removed from the viewer list.
Expected Outcome	The corresponding graph is removed from the viewer list.
Notes	The user can only remove viewers that are currently in the viewer list.

Test ID	0016
Category	evaluation of Adding or Removing Viewers
Requirements Coverage	UC6-Removing-Viewer
Initial Condition	the user has selected the duration within which the analysis is performed.
Procedure	1. The user selects a viewer from the drop-down menu that is not in the viewer list 2. The user presses the “remove” button
Expected Outcome	The user receives an error message
Notes	The user can only select viewers that are already in the list of viewers.

Test ID	0017
Category	evaluation of Analysis Result
Requirements Coverage	UC7-and-UC8-Display-Result
Initial Condition	the user has set all the parameters for the currently selected analysis type.
Procedure	1. The user presses the recalculate button.

	2. The main UI calculates and shows the viewer list contains all graphs and data analysis.
Expected Outcome	The corresponding graphs and data are shown on main UI.
Notes	None

Test ID	0018
Category	evaluation of Analysis Result
Requirements Coverage	UC7-and-UC8-Display-Result
Initial Condition	the user has set all the parameters for the currently selected analysis type.
Procedure	1. The user presses the recalculate button. 2. The main UI calculates and shows the viewer list contains all graphs and data analysis.
Expected Outcome	The corresponding graphs and data are shown on main UI.
Notes	None