GINA CODY
SCHOOL OF ENGINEERING
AND COMPUTER SCIENCE

SOEN 6441

Advanced Programming Practices

Project Report

Submitted By:

Name:	Student ID:
1. Riddhi VinodBhai Bhuva	40220969
2. Ayushi Chaudhary	40224978

Table of Contents

INTRODUCTION

OBJECTIVE:

The goal of this project is to put into practise coding conventions, useful patterns, refactoring techniques, testing, and to create a software architecture document.

Problem Statement:

Create a system that can read API/JSON data and return results to an application using any language (or a combination of languages).

Design Patterns Implementation:

1. Singleton Design Pattern : Singleton is a creational design pattern that lets us ensure that a class has only one instance, while providing a global access point to this instance.

2. Data Mapper Pattern : A Data Mapper is a type of Data Access Layer that performs bi-directional transfer of data between objects in memory and persistent storage. With the Data Mapper Pattern, the in-memory objects have no idea there is a database, and the database schema is unaware of any objects that use it. This separation is the main purpose of a Data Mapper.

3. Foreign Key Mapping: It comes under the Object Relational Structural Pattern that "Maps an association between objects to a foreign key reference between tables."

IMPLEMENTATION OF DESIGN PATTERNS

Singleton Pattern

It is a creational design pattern that lets us ensure that a class has only one instance, while providing a global access point to this instance.

Purpose:

- Ensure a class has only one instance and provide a global point of access to it.
- Encapsulated "just-in-time initialization" or "initialization on first use".
- Declare a static method that is implicitly called once upon request to obtain an instance of the class.

Fig 1: Class Diagram of Database class implementing Singleton Design Pattern

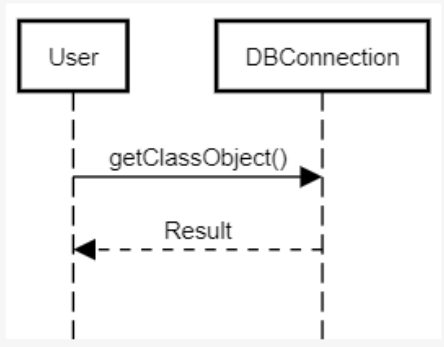

Fig 2: Dynamic Model of Database Class

```
class Database:
    __classObject = None

    def __init__(self):
        self._DBName = "News.db"
        if Database.__classObject != None:
            raise Exception("Singleton Class error")
        else:
            Database.__classObject = self

    @staticmethod
    def getClassObject():
        if Database.__classObject == None:
            Database()
        return Database.__classObject
```

Fig 3: Database class implementing Singleton Design Pattern

In this, the getClassObject() method creates __classObject and initializes it if it is created for the first time and this same instance is returned when attempting to make a connection with the database again.

Data Mapper

The data mapper will help complete the bi-directional transfer of persistence layer and in-memory data.

A layer of mappers that moves data between objects and a database while keeping them independent of each other and the mapper itself. This layer consists of one or more mappers (or data access objects) that perform data transfer. The scope of mapper implementations varies. A generic mapper will handle many different domain entity types, a dedicated mapper will handle one or a few.

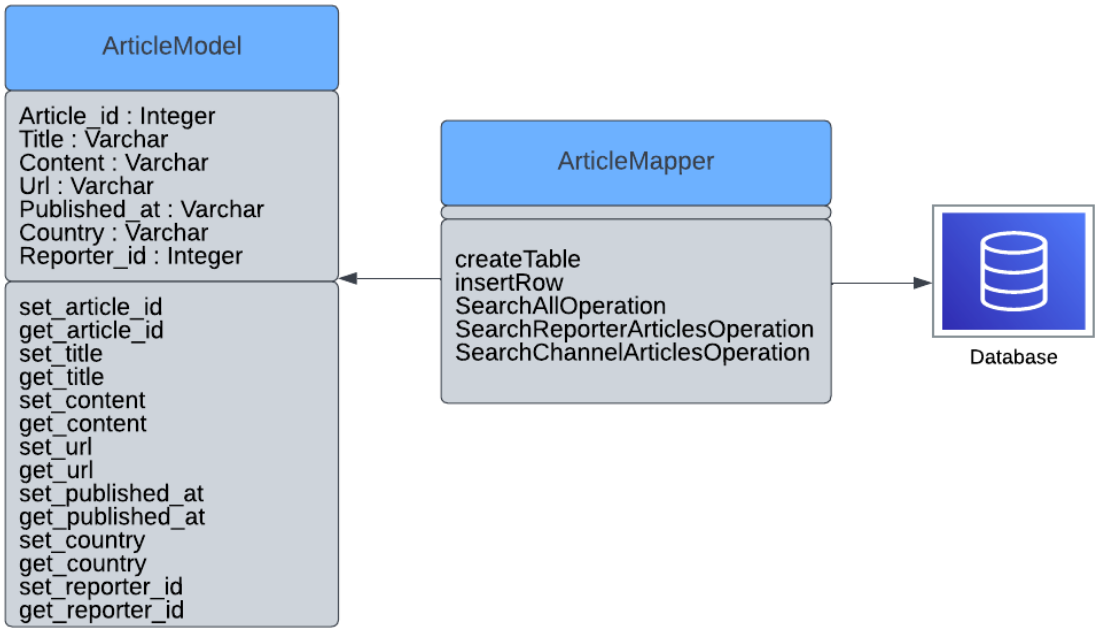

Fig 4: Class ArticleMapper Implementing Data Mapper functionalities

The ArticleModel class contains the class variables Article_id, Title, Content, Url, Published_at, Country, Reporter_id which are initialized using setter methods set_article_id() , set_title() and so on. ArticleMapper class retrieves the values of these variables using the getter methods present in ArticleModel class and use them to create the database news.db and inserting the data fetched from the API in the tables created using the createTable() and insertRow() methods.Therefore, ArticleMapper transports data across object and relational databases while keeping them separate.

In a similar approach, ChannelModel, ChannelMapper , ReporterModel, ReporterMapper classes are also created to serve similar functionalities on the data.

Foreign Key Mapping :

Maps an association between objects to a foreign key reference between tables.

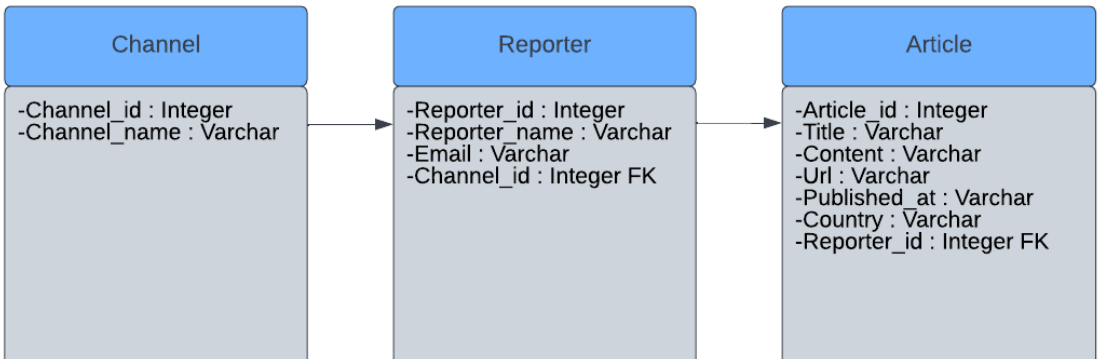

Fig 5: Foreign Key Mapping relation between tables Articles, Reporters and Channel

In our implementation, Reporter_id and Channel_id are made foreign keys to the table Articles and Reporters respectively to access the information required in our application using INNER JOIN query.

REFACTORING

Code refactoring is the process of restructuring existing computer code—changing the *factoring*—without changing its external behavior. Refactoring is intended to improve the design, structure, and/or implementation of the software (its *non-functional* attributes), while preserving its functionality.

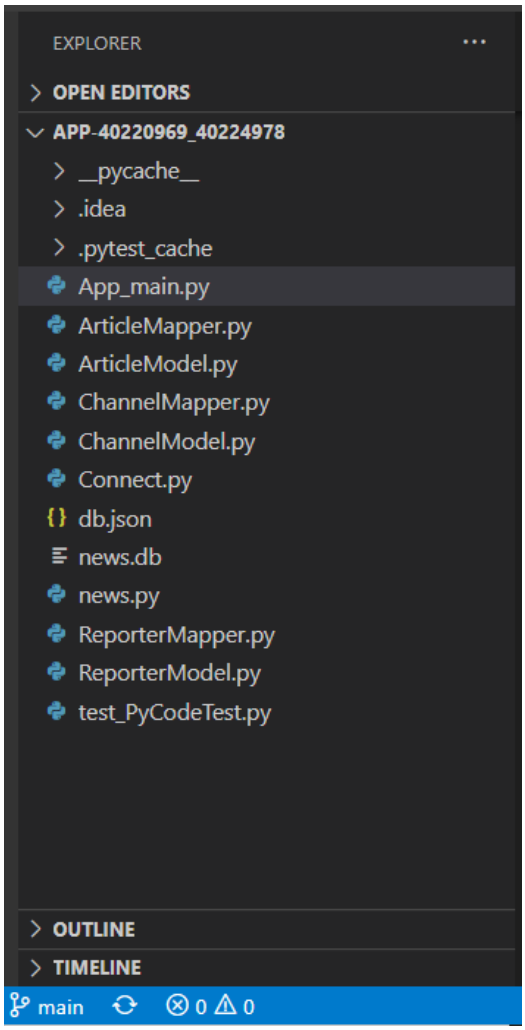

Fig 6: Before Refactoring

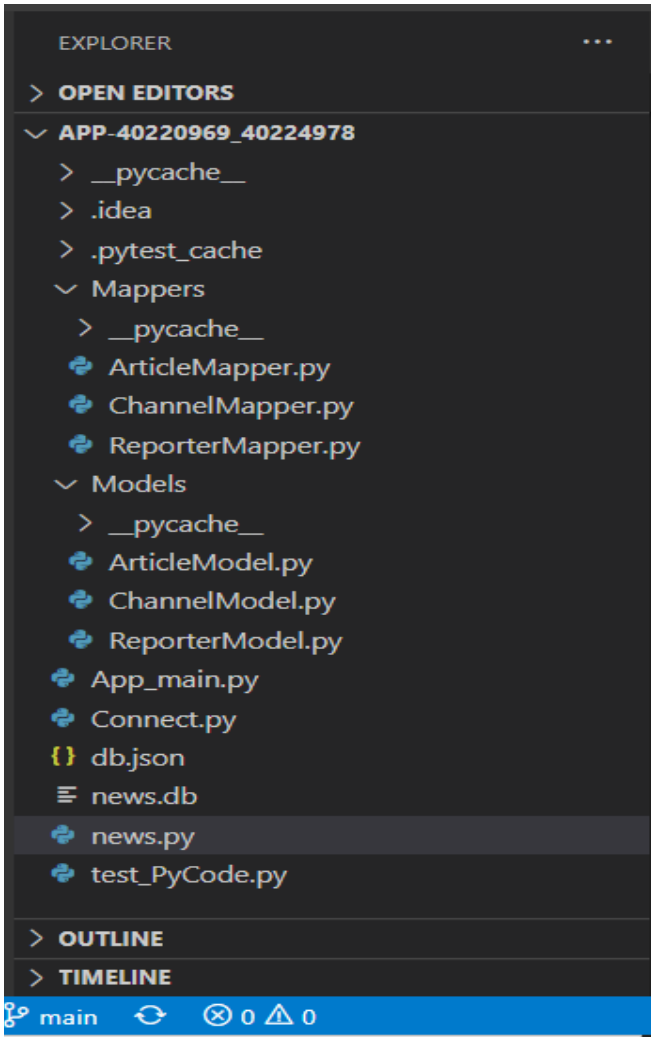

Fig 7: After Refactoring

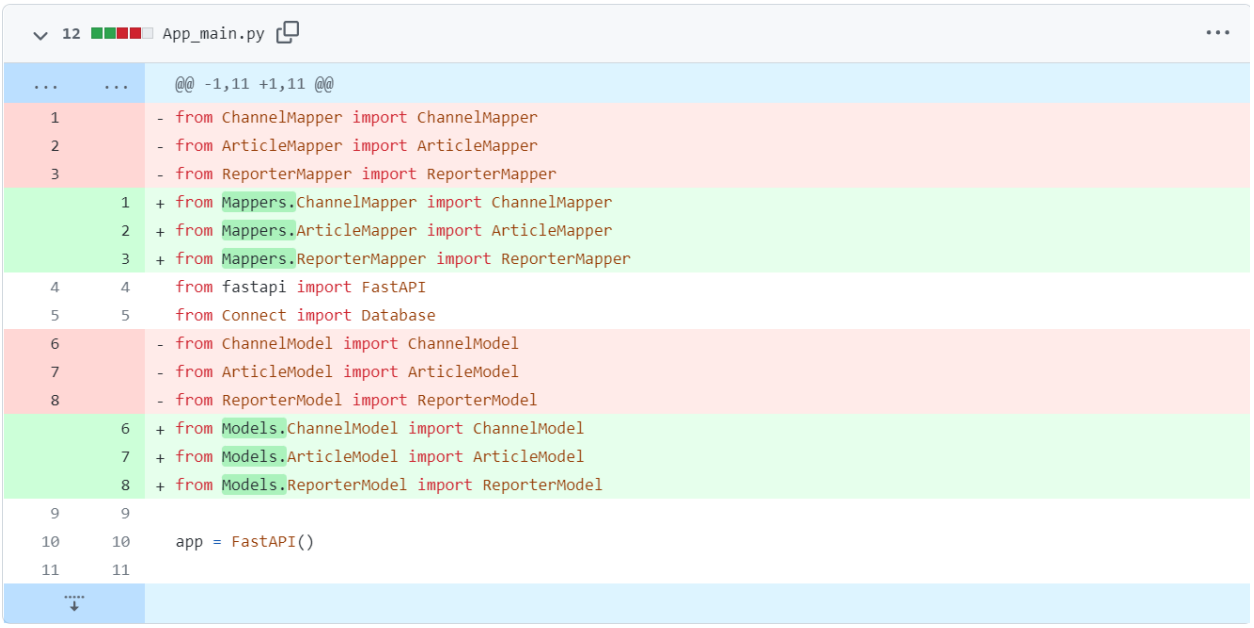

Fig 8: Refactoring App_main.py file

TESTING

The pytest framework makes it easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries.
Features:

- Detailed info on failing assert statements (no need to remember self.assert* names)
- Auto-discovery of test modules and functions
- Modular fixtures for managing small or parametrized long-lived test resources
- Can run unittest (including trial) and nose test suites out of the box
- Python 3.7+ or PyPy 3
- Rich plugin architecture, with over 800+ external plugins and thriving community

Test Cases :

1) Test if the record is inserted successfully in Channel table or test case fails

```
@pytest.mark.one
def test_data_insert_in_ChannelModel():
    assert __ChannelMapper.insertRow(__channel) is None
```

Fig 9: Test Case 1

2) Test if the record is inserted successfully in Reporters table or test case fails

```
@pytest.mark.one
def test_data_insert_in_ReporterModel():
    assert __ReporterMapper.insertRow(__reporter) is None
```

Fig 10: Test Case 2

3) Test if the record is inserted successfully in Articles table or test case fails

```
@pytest.mark.one
def test_data_insert_in_ArticleModel():
    assert __ArticleMapper.insertRow(__article) is None
```

Fig 11: Test Case 3

4) Test if the records are displayed successfully or test case fails

```
@pytest.mark.two
def test_data_display_in_ArticleModel():
    result = __ArticleMapper.SearchAllOperation()
    assert result is not None
```

Fig 12: Test Case 4

5) Test if the Reporter's records are displayed successfully for given channel name or test case fails

```
@pytest.mark.three
def test_data_search_in_ReporterModel():
    __channel.set_channel_name("BBC-News")
    result = __ReporterMapper.SearchReporterOperation(__channel)
    assert result is not None
```

Fig 13: Test Case 5

6) Test if the Article's records are displayed successfully for given reporter name or test case fails

```
@pytest.mark.three
def test_data_reporter_in_ArticleModel():
    __reporter.set_reporter_name("Pranay Parab")
    result =
__ArticleMapper.SearchReporterArticlesOperation(__reporter)
    assert result is not None
```

Fig 14: Test Case 6

7) Test if the Article's records are displayed successfully for given reporter name or test case fails

```
@pytest.mark.three
def test_data_channel_in_ArticleModel():
    __channel.set_channel_name("BBC-News")
    result =
__ArticleMapper.SearchChannelArticlesOperation(__channel)
    assert result is not None
```

Fig 15: Test Case 7

8) Test if code gives error for wrong column name or not

```
@pytest.mark.xfail
@pytest.mark.parametrize("Column, Value", [('Name', 'Pranay
Parab')])
def test_find_wrong_column(Column, Value):
    assert __reporter[Column] is Value
    print("Test will fail because the column name is wrong")
```

Fig 16: Test Case 8

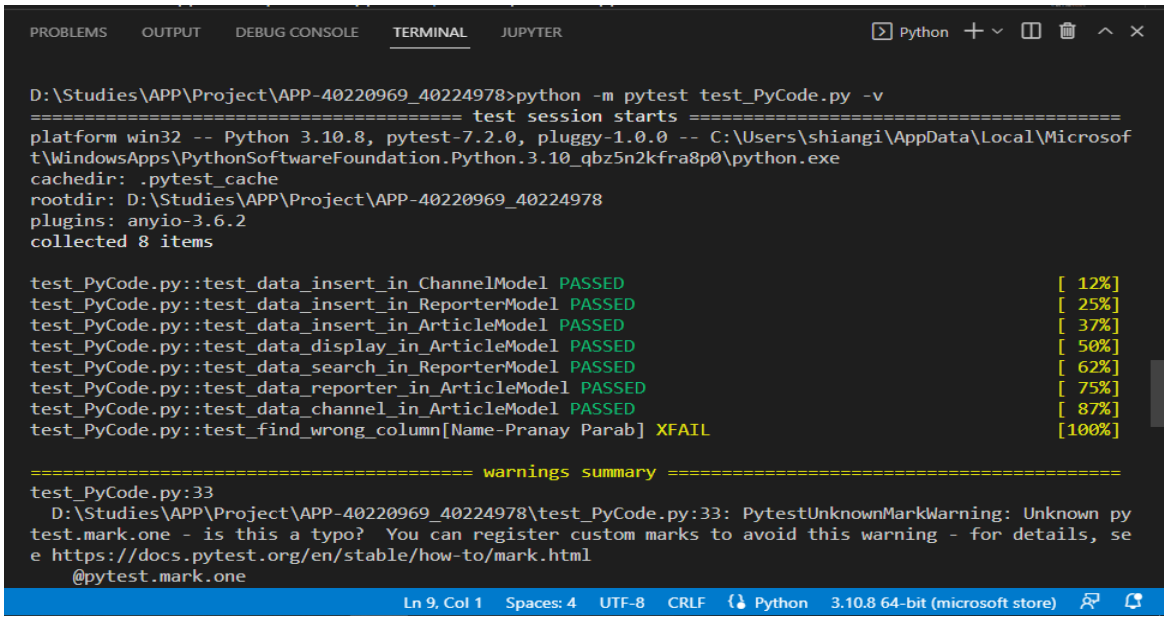

Fig 17: Testing Result

Software Architecture Document

A software architecture Document serves as a roadmap for the project. It helps us quickly identify the software's structure. Without delving into the code, it aids in your understanding of the modules and parts of the product.It is a tool for talking to people about the software, both developers and non-developers.

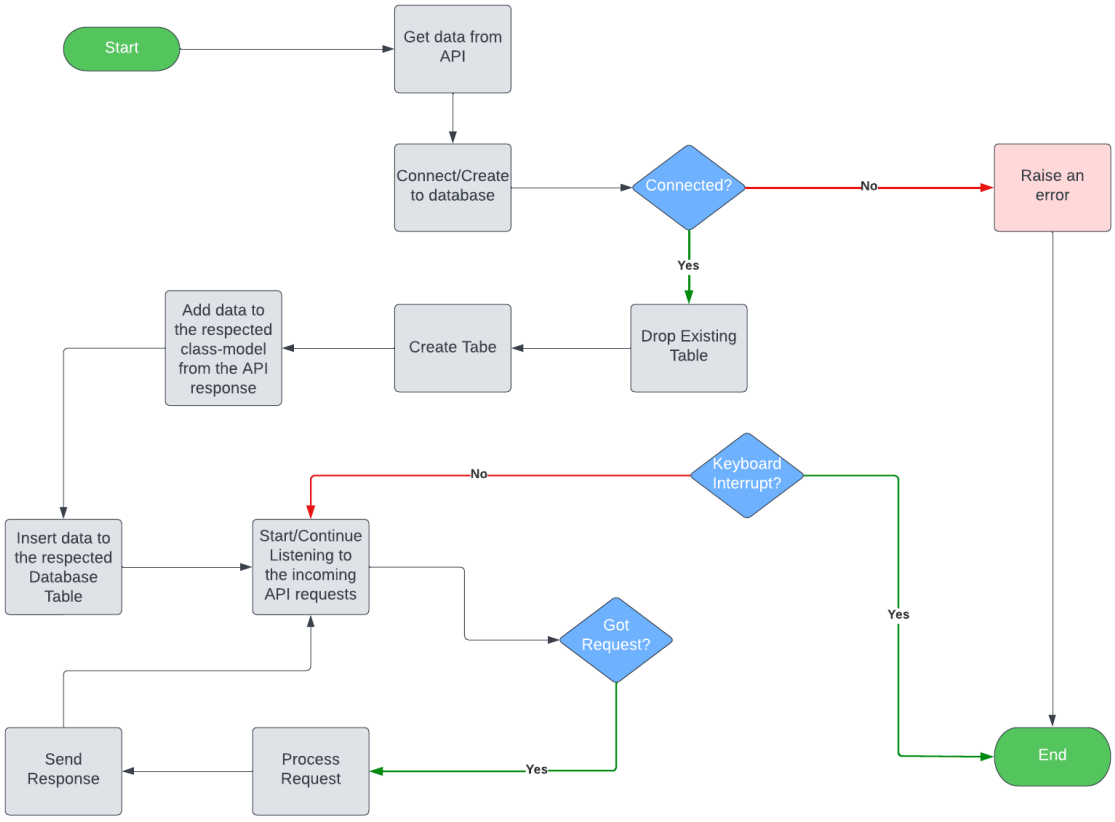

Fig 18 : Flow Chart of News Report System

According to the Flow Diagram Fig.18, we are attempting to retrieve a JSON response from an API (Application Programming Interfaces) before connecting it to the database. It raises an error if it is not connected, and if it connected, it drops theexisting table, creates a new one, and inserts values by repeatedly iterating over

the JSON response from the API. Then, it will continue to receive API requests from the user interface, process them, and reply to the user interface.

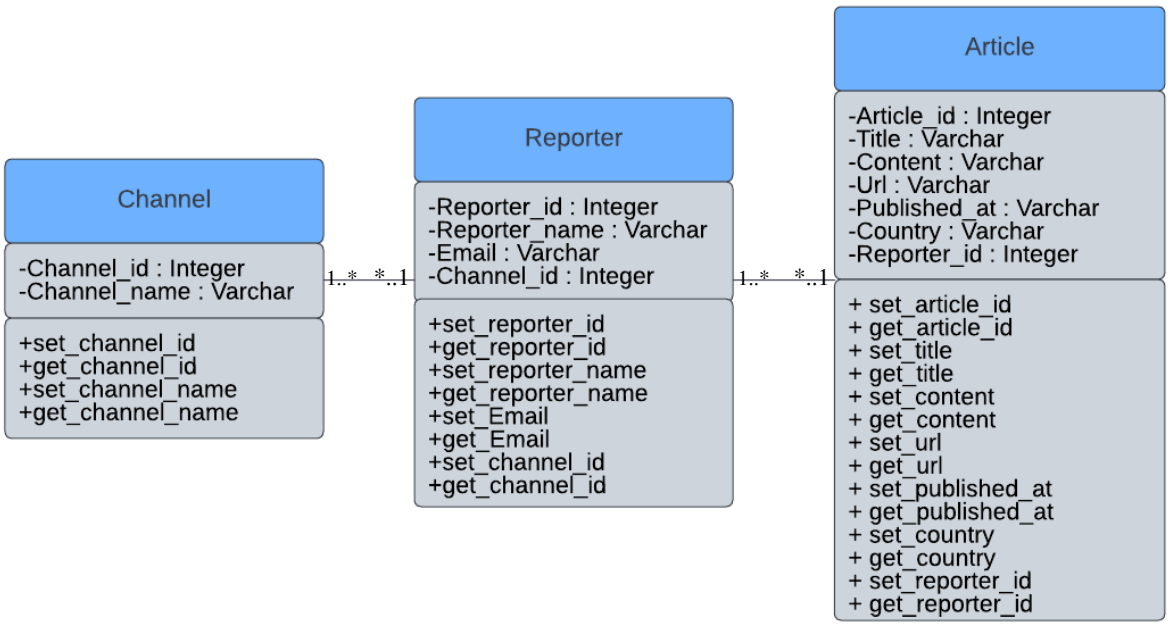

Fig 19: Class Diagrams

The class diagram for the News Report System, which includes all the instance variables and methods needed for each class Articles, Reporters, and Channels is shown in Fig 17.

OUTPUT :

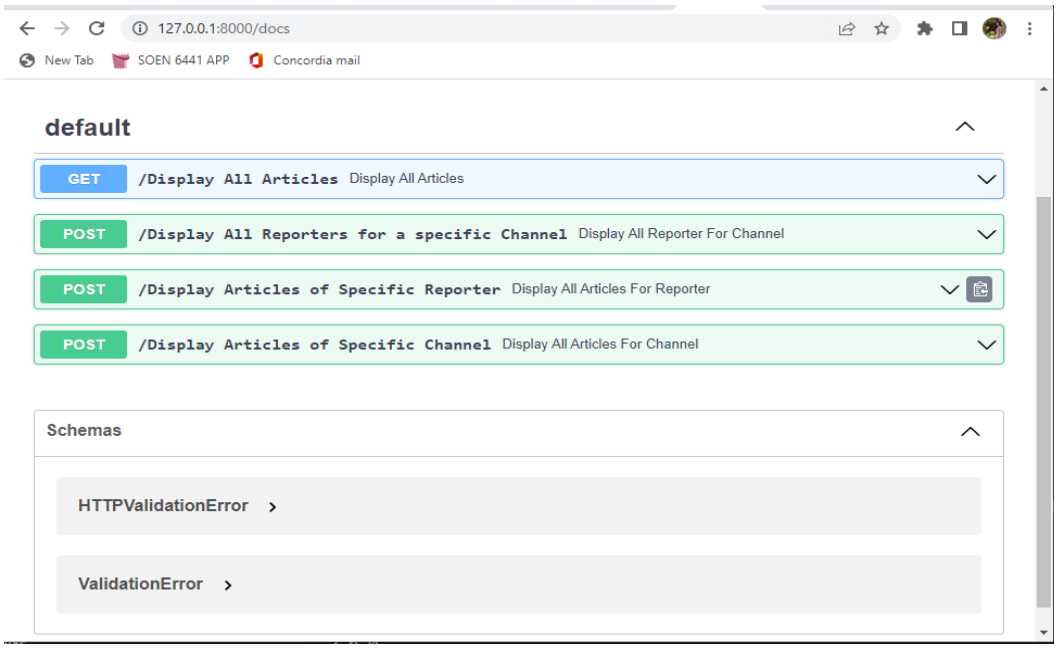

Fig 20: Home page

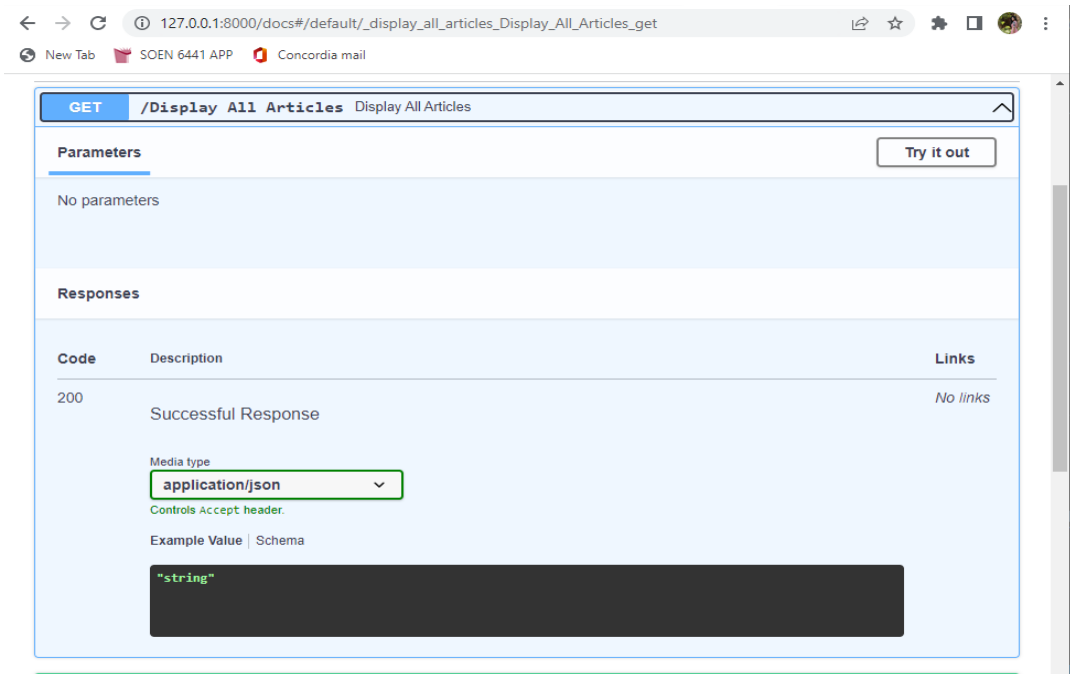

Fig 21: Display all Articles(Screen 1)

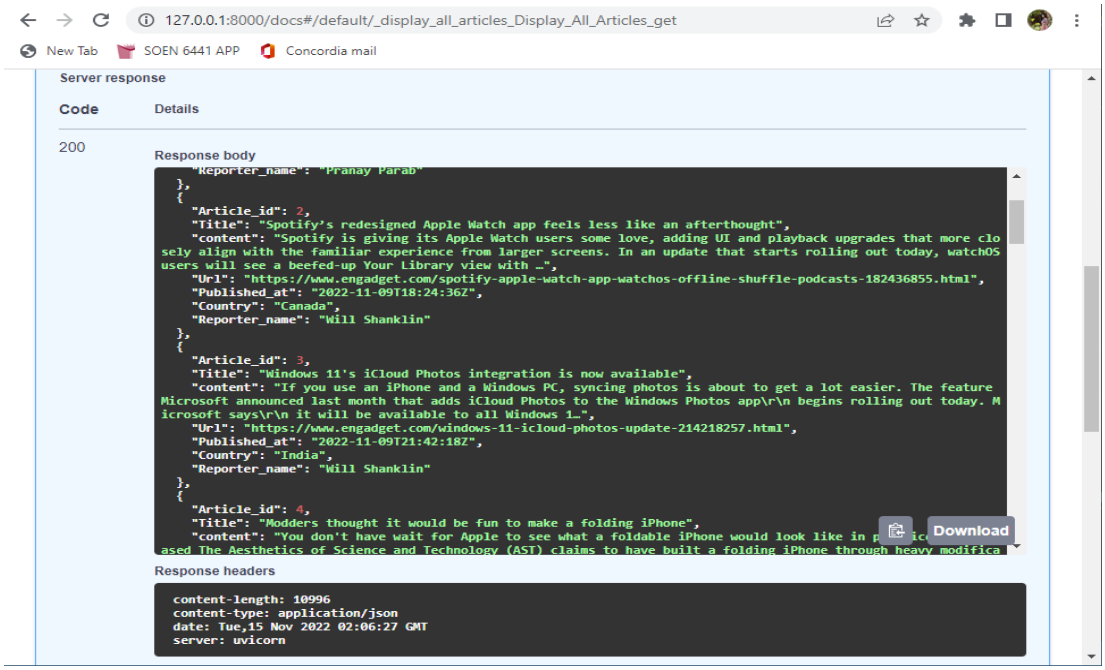

Fig 22: Display all Articles(Screen 2)

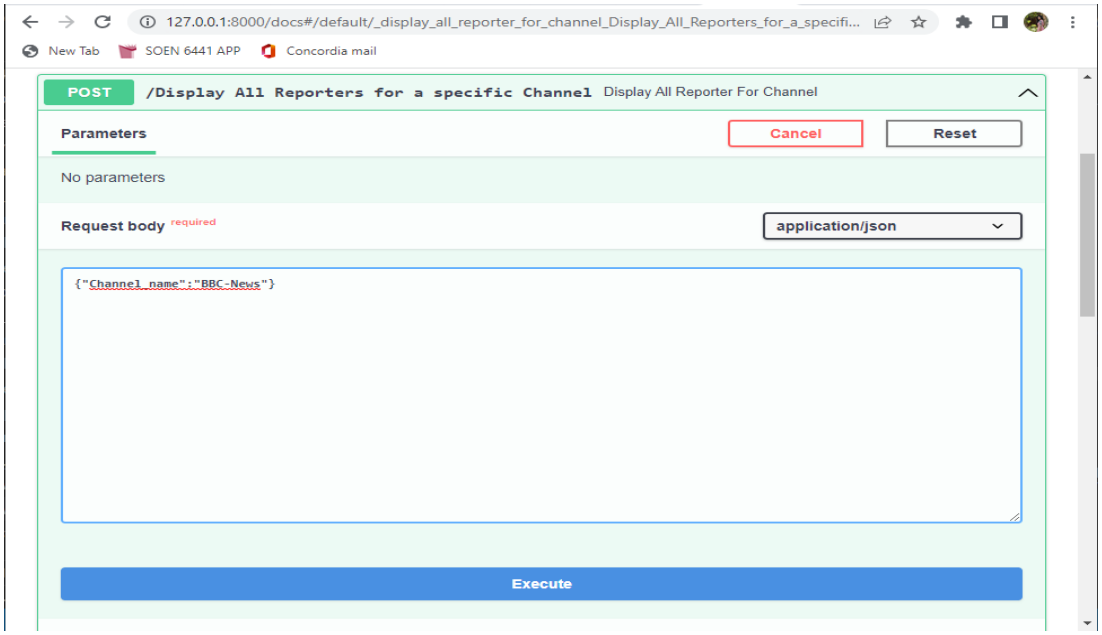

Fig 23: Display all Reporters for specific Channel(Parameterized Query

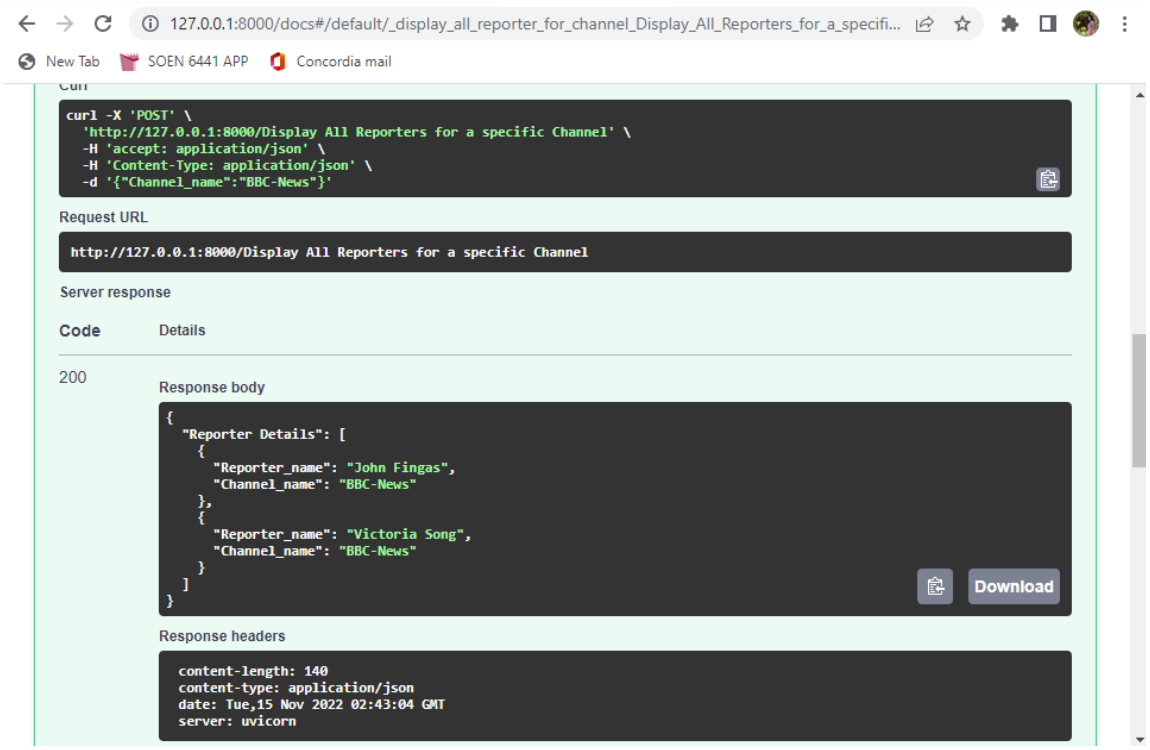

Fig 24: Display all Reporters for specific Channel(Response)

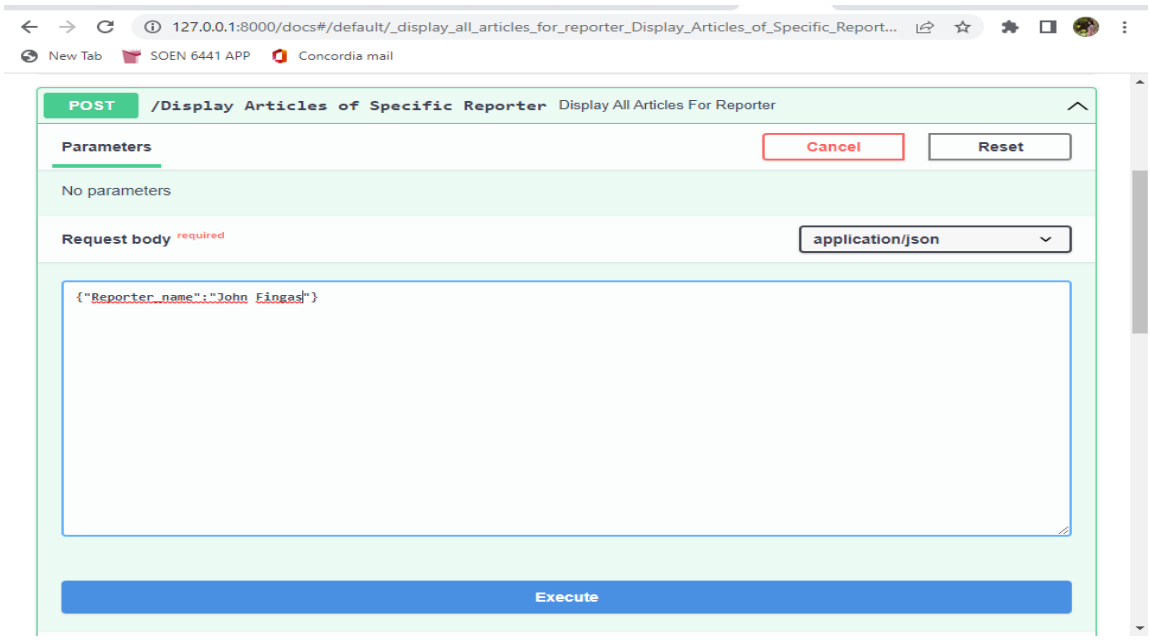

Fig 25: Display all Articles for specific Reporter(Parameterized Query)

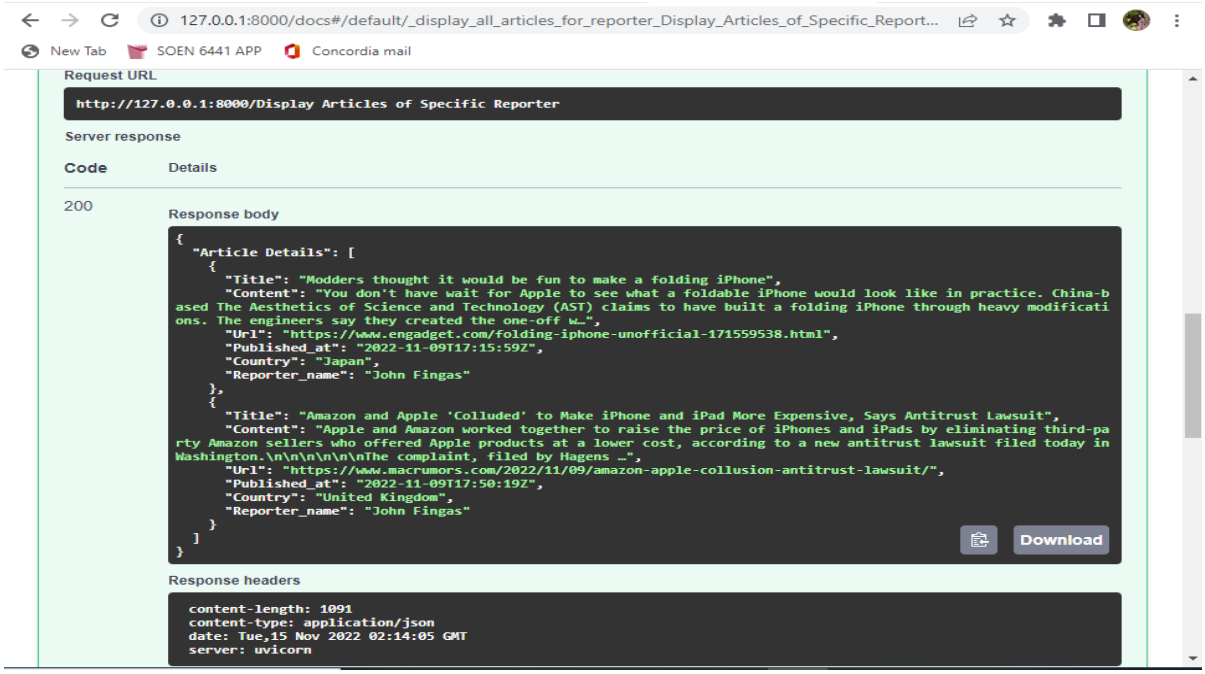

Fig 26: Display all Articles for specific Reporter (response)

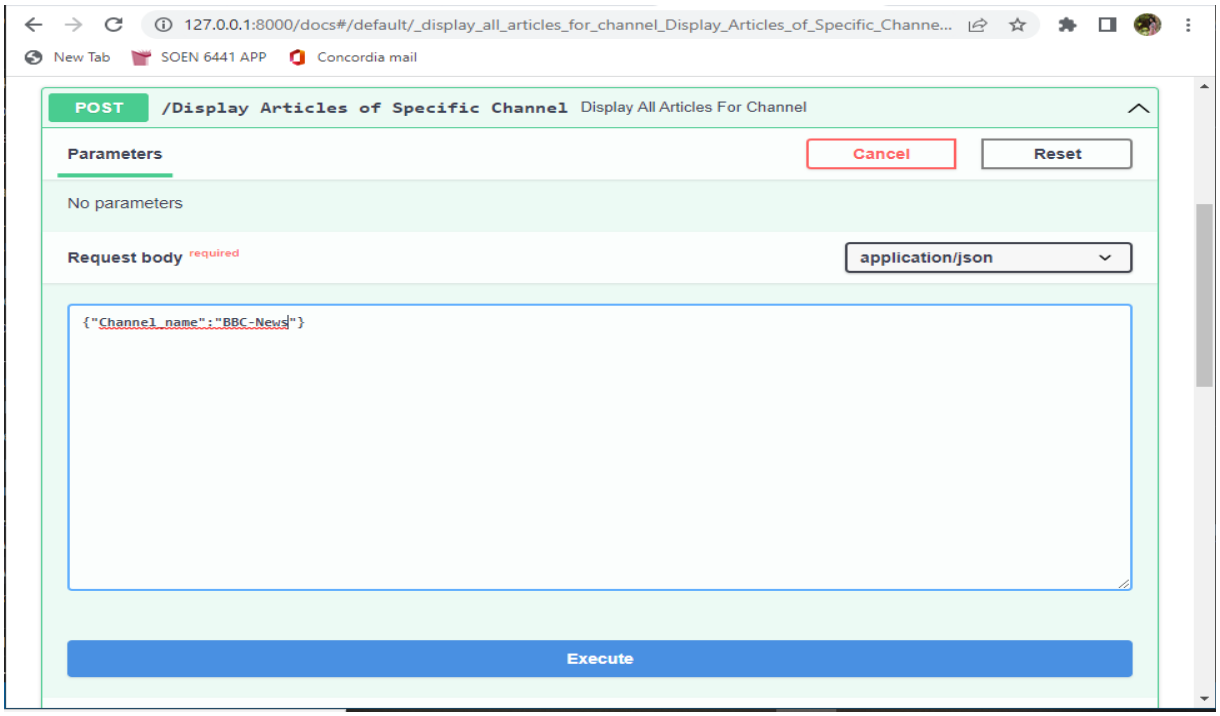

Fig 27: Display all Articles for specific Channel (Parameterized Query)

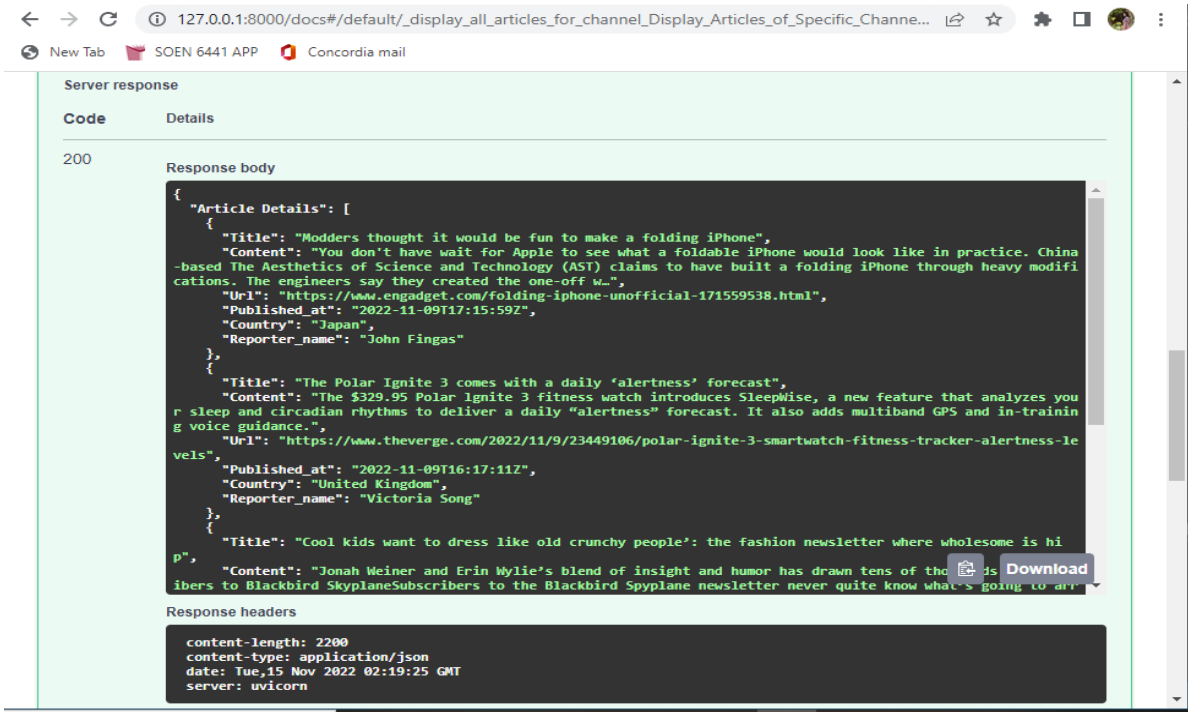

Fig 28: Display all Articles for specific Channel (Response)

References:

- **Refactoring.guru**
- **martinfowler.com**
- **blog.ndepend.com**
- **Techtarget.com**
- **Lucidchart.com**
- **-docs.pytest.org/en/7.2.x**