News Manager System

40225526 – Jimi Mehta

Report

Version 1.0

Github Link: https://github.com/jimimehta/NewsManager-App-Project

Video Link: https://drive.google.com/file/d/1N0Vt2tRVrxA6oNXZ1PrGHpkNUy0CX1U1/view?usp=sharing

Table of Contents

1. Introduction

1.1 Purpose

This document provides an overview of the architecture, testing and functionalities of the News Manager system. It embodies the significant decisions that were made concerning the architecture of the application.

1.2 Scope

This Software Architecture Document gives a global view of the architecture of the News Manager System. This system is developed by one student registered in the SOEN6441 Advance Programming Practices.

The News Manager provides users with the ability to manage and categorize their News, track their News, and create News articles. All those features are available to the users wherever they can find a computer with an Internet connection.

1.3 Definitions, Acronyms, and Abbreviations

HTML = Hypertext Markup Language.
CSS = Cascading Style Sheet
JS = Javascript
JPA= Java Persistence API
UI = User Interface.

1.4 References

"SOEN 6481, Software Architecture" Course Page
[Website] Accessible:
https://users.encs.concordia.ca/~cc/soen6441//.

1.5 Overview

In compliance with the specified purpose and scope, the rest of this document will deal with the following topics:

2. Tools and Technologies Used

2.1 Server-side tools and technologies used

- Spring Boot 2 +
- SpringData JPA (Hibernate)
- Maven 3.2 +
- JDK 1.8
- Embedded Tomcat 8.5+
- Eclipse 2021-12
- Microsoft SQL Database

2.2 Client-side tools and technologies used

- React
- Modern JavaScript (ES6)
- NodeJS and NPM
- VS Code IDE
- Create React App CLI
- Bootstrap 4.5 and Axios HTTP Library

3. Use-Case View

This Use Case view will be used to present the architecturally significant use cases and identify the set of scenarios and use cases that represent important and fundamental functionalities of the system.

3.1 Architecturally significant use cases

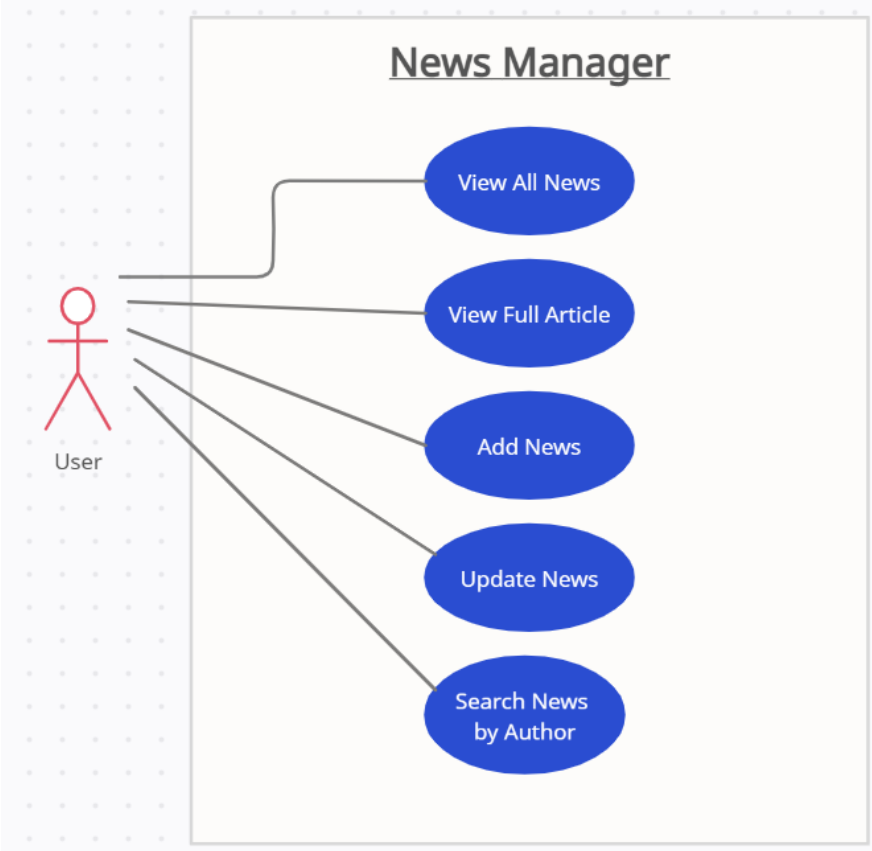


Figure 1. Architecturally Significant Use Case Diagram

4. Data View

The database provides persistence for all data input into the News Manager. The schema in which the relevant data is represented is as illustrated in the Database diagram below.

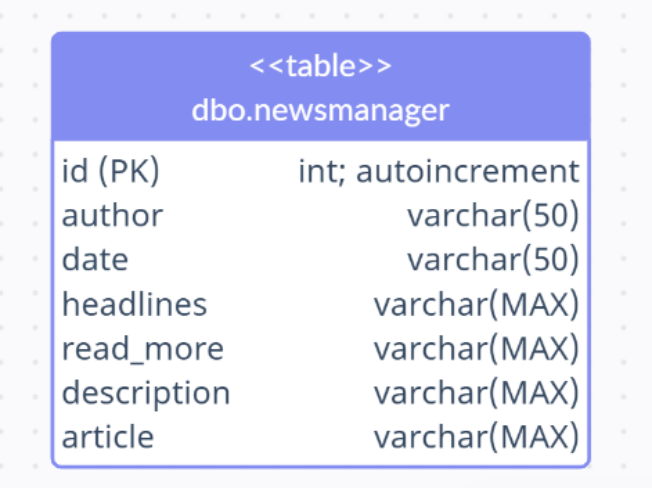


Figure 2. Diagram for the News Manager's database

5. Object Relational Structural Patterns

Single-table inheritance strategy: Map the entire class hierarchy to a single table ("filtered mapping").
The database provides persistence for all data input into the News Manager which is mapped using Single-table inheritance strategy. ORM of News Manager is as illustrated in the diagram below.

Table-per-class inheritance mapping (Vertical mapping)
A table-per-class inheritance strategy maps each class to its own table and maps each attribute to a column in the mapped table.

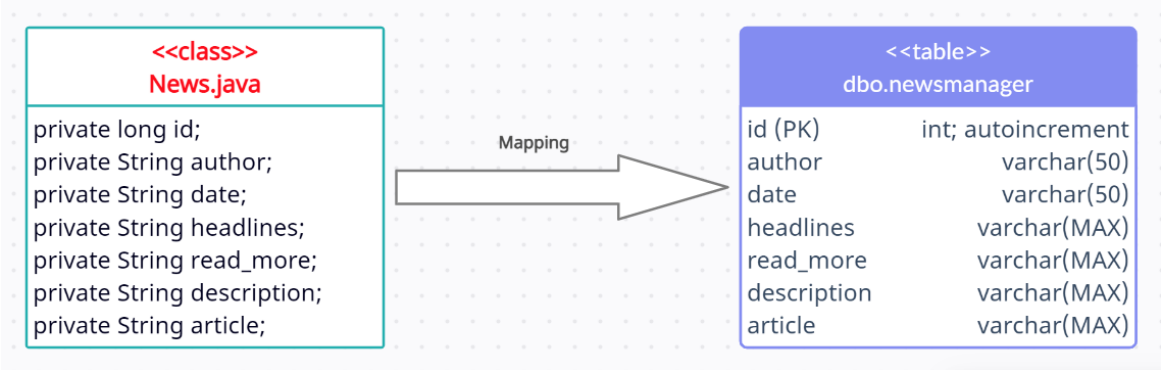


Figure 3. Diagram for the News Manager's ORM

6. Updated Class Diagram:

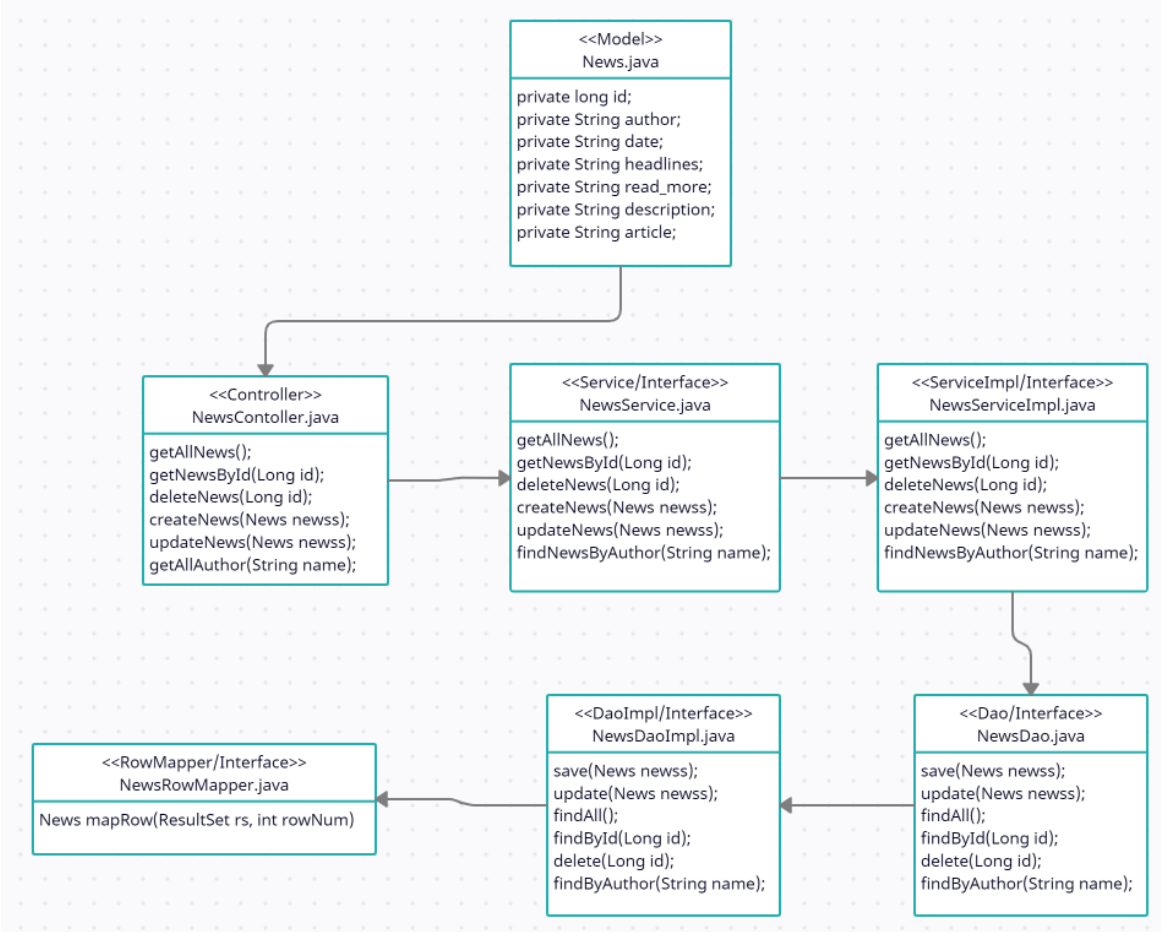


Figure 4. Class Diagram for the News Manager

7. Design Pattern

In News Manager, React is used to handle the UI — frontend, and Spring Boot Java is used to handle the data — backend. And to connect the two of them, we need REST API.

Now it becomes **Model-View-Controller(MVC)**

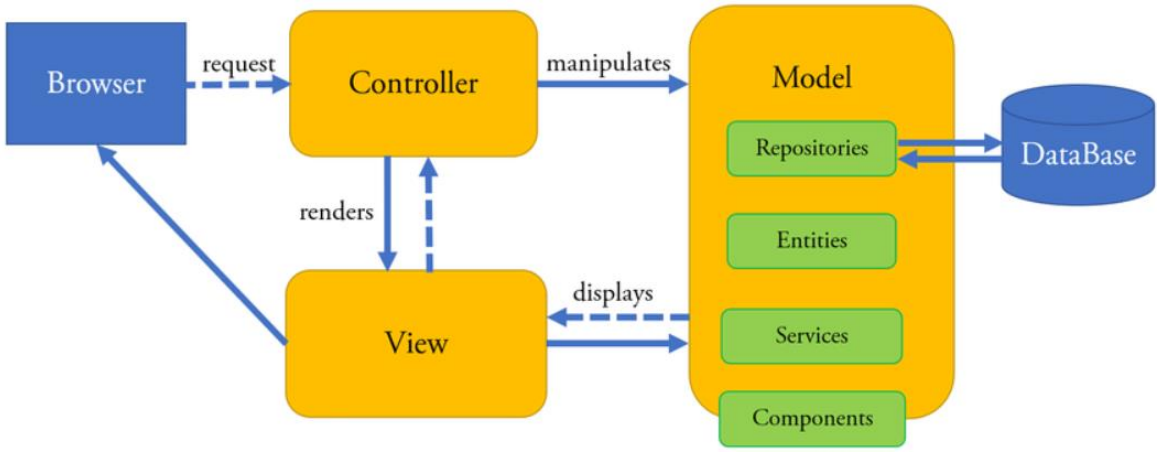


Figure 5. Diagram for the News Manager's DESIGN PATTERN

In News Manager, repository pattern have two purposes; first it is an abstraction of the data layer and second it is a way of centralising the handling of the domain objects.

Now it becomes **Repository Design Pattern.**

The idea with this pattern is to have a generic abstract way for the app to work with the data layer without being bother with if the implementation is towards a local database or towards an online API.

The methods are based on the CRUD methods; Create, Read, Update and Delete.

It can be easily looked at from the diagram 5.

8. Object Relational Structural Patterns

In News Manager, Presentation, Domain and Data Layer are divided as per the diagram. It has data mapping as well as TDG for their operation.

Before:

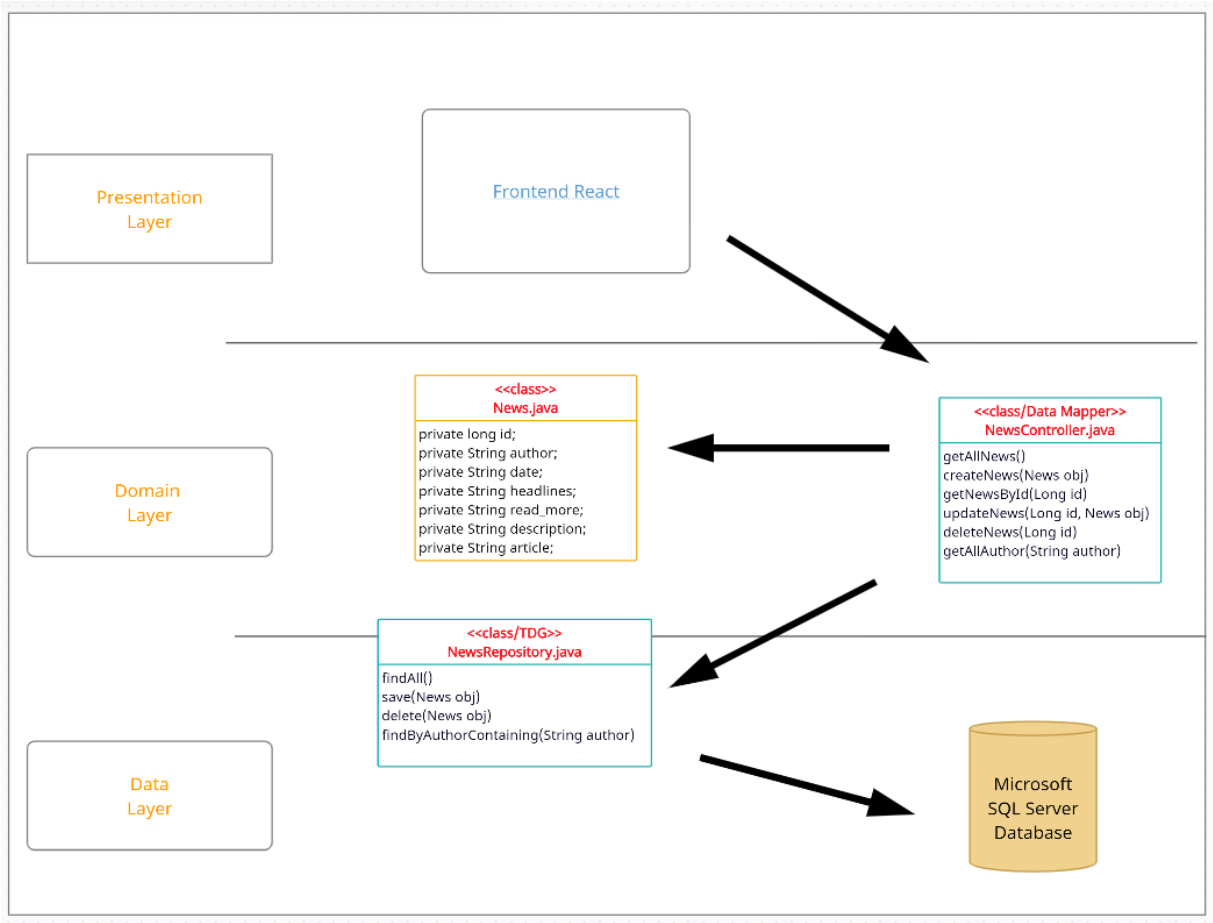


Figure 6. Diagram for the News Manager's Object Relational Structure Pattern

After:
DAO works as ROW DATA GATEWAY/TDG and SERVICE CLASS AS DATAMAPPER

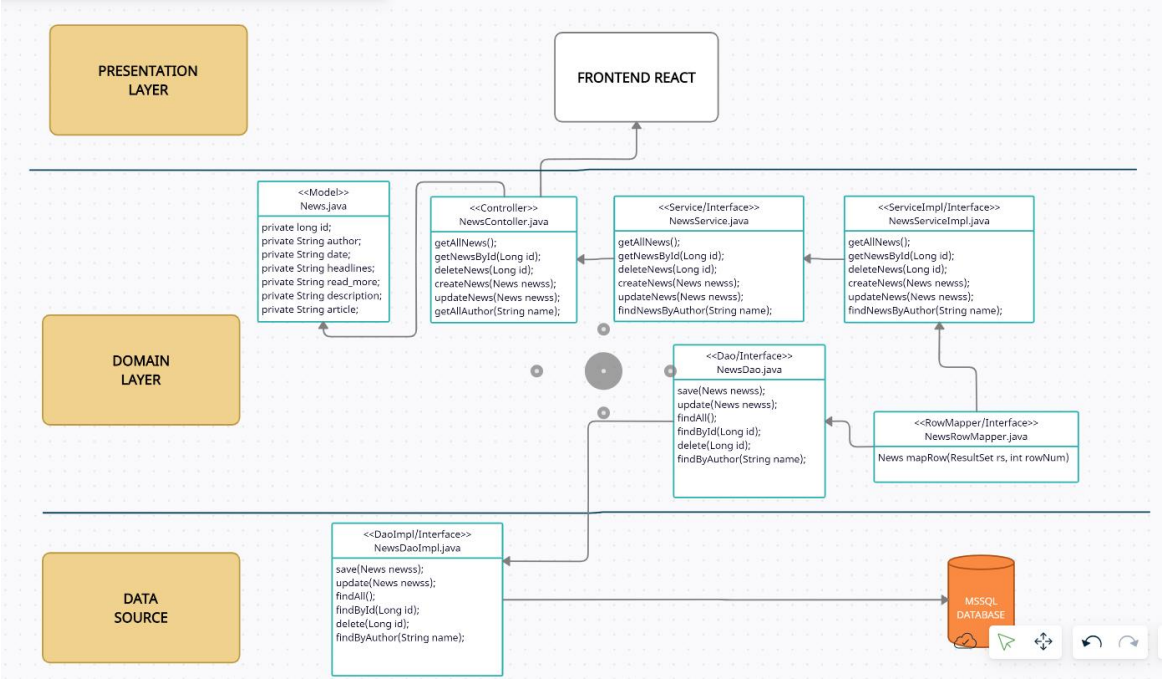


Figure 6. Diagram for the News Manager's Object Relational Structure Pattern

9. Spring Boot React Full-Stack Architecture

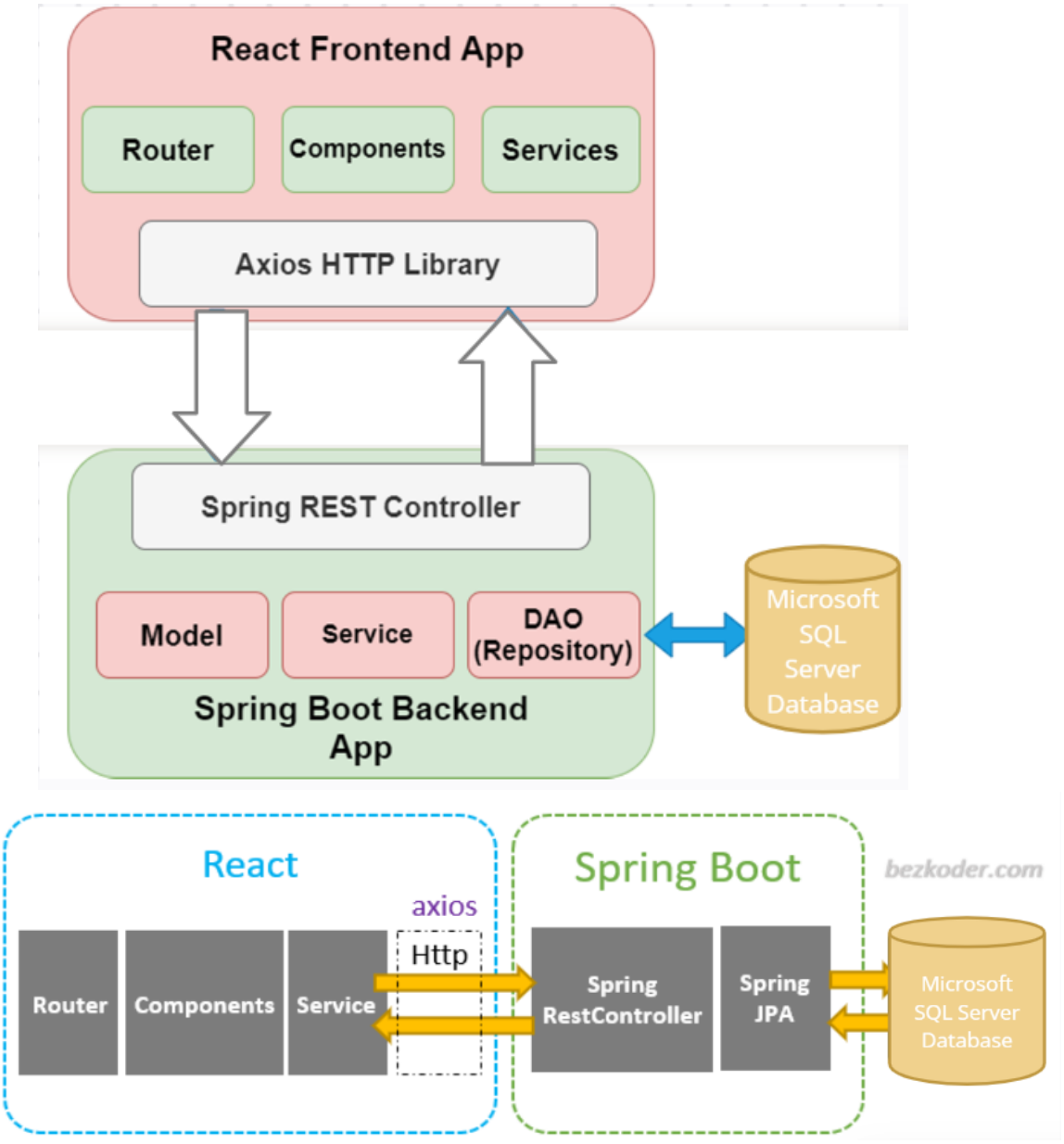


Figure 5. Diagram for the News Manager's Full Stack Architecture

10. Refactoring

Before:

```
@Repository
public class NewsDaoImpl implements NewsDao {

    @Autowired
    private JdbcTemplate jdbcTemplate;


    @Override
    public int save(News newss) {

        return jdbcTemplate.update("INSERT INTO newsmanager(author,date,headlines,read_more,description,article) VALUES(?,?,?,?,?,?)", new Object[] { newss.getAuth
             newss.getDate(), newss.getHeadlines(), newss.getDescription(), newss.getRead_more(),newss.getArticle() });
    }

    @Override
    public int update(News newss) {

        return jdbcTemplate.update("UPDATE newsmanager SET author=?,date=?,headlines=?,read_more=?,description=?,article=? WHERE id=?", new Object[] {
                newss.getAuthor(), newss.getDate(), newss.getHeadlines(), newss.getDescription(), newss.getRead_more(),newss.getArticle(), newss.getId()});
    }

    @Override
    public int delete(Long id) {

        return jdbcTemplate.update("DELETE FROM newsmanager WHERE id=?", new Object[] { id });
    }

    @Override
    public List<News> findAll() {

        return jdbcTemplate.query("SELECT * FROM newsmanager where id = ?", new NewsRowMapper());
    }

    @Override
    public Optional<News> findById(Long id) {
```

After:

```
import java.util.List;



@Repository
public class NewsDaoImpl implements NewsDao {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    private final String INSERT_NEWS_QUERY = "INSERT INTO newsmanager(author,date,headlines,read_more,description,article) VALUES(?,?,?,?,?,?)";
    private final String UPDATE_NEWS_QUERY = "UPDATE newsmanager SET author=?,date=?,headlines=?,read_more=?,description=?,article=? WHERE id=?";
    private final String DELETE_NEWS_QUERY = "DELETE FROM newsmanager WHERE id=?";
    private final String GET_NEWS_BY_ID_QUERY = "SELECT * FROM newsmanager where id = ?";
    private final String GET_NEWS_BY_AUTHOR_QUERY = "SELECT * FROM newsmanager n where n.author = ?";
    private final String GET_NEWSS_QUERY = "SELECT * FROM newsmanager";


    @Override
    public int save(News newss) {

        return jdbcTemplate.update(INSERT_NEWS_QUERY, new Object[] { newss.getAuthor(),
             newss.getDate(), newss.getHeadlines(), newss.getDescription(), newss.getRead_more(),newss.getArticle() });
    }

    @Override
    public int update(News newss) {

        return jdbcTemplate.update(UPDATE_NEWS_QUERY, new Object[] {
                newss.getAuthor(), newss.getDate(), newss.getHeadlines(), newss.getDescription(), newss.getRead_more(),newss.getArticle(), newss.getId()});
    }

    @Override
    public int delete(Long id) {

        return jdbcTemplate.update(DELETE_NEWS_QUERY, new Object[] { id });
    }
```

Before:

```
    @Override
    public News getNewsById(Long id) {

        News newss = new News();
        Optional<News> existingNews = newsDao.findById(id);
        if (existingNews.isPresent())
            newss = existingNews.get();
        return newss;
    }
```

After:

```
    @Override
    public News getNewsById(Long id) {

        News newss = new News();
        if (newsDao.findById(id).isPresent())
            newss = newsDao.findById(id).get();
        return newss;
    }
```

Before:

```
@Repository
public interface NewsRepository extends JpaRepository<News, Long>{


    List<News> findByAuthorContaining(String authorname);

    }
```

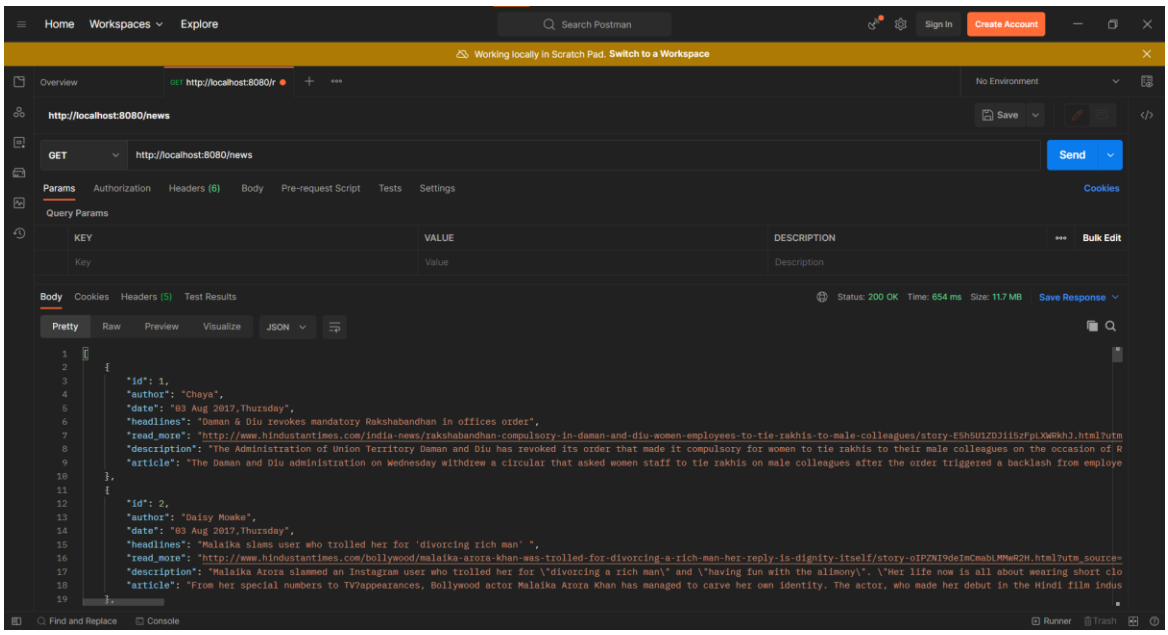
After:

```
@Repository
public interface NewsRepository extends JpaRepository<News, Long>{


    List<News> findByAuthorContaining(Object authorname);

}
```

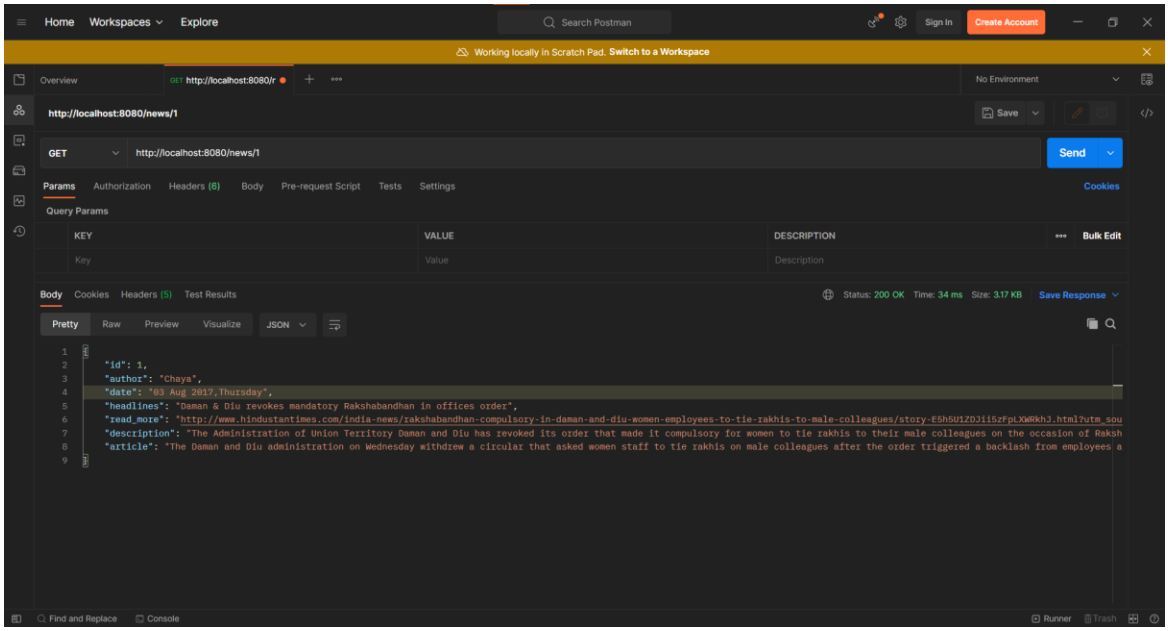
11. Testing Tools

Postman is an API platform for developers to design, build, test and iterate their APIs. Below Screenshots of API TESTING in POSTMAN are illustrated.

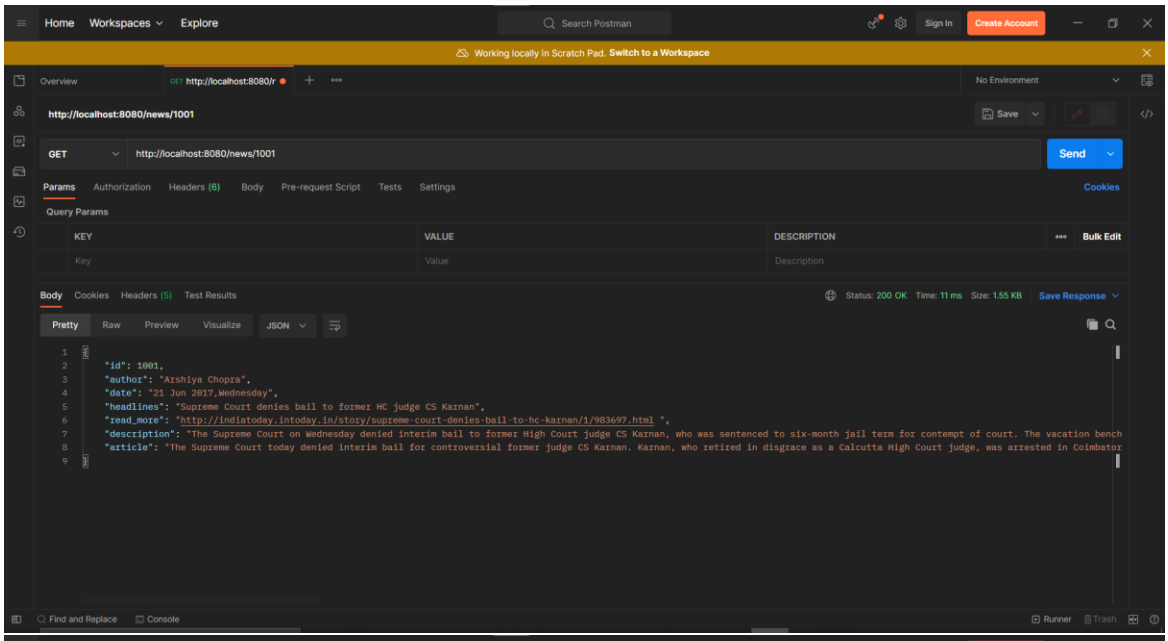
Test 1:

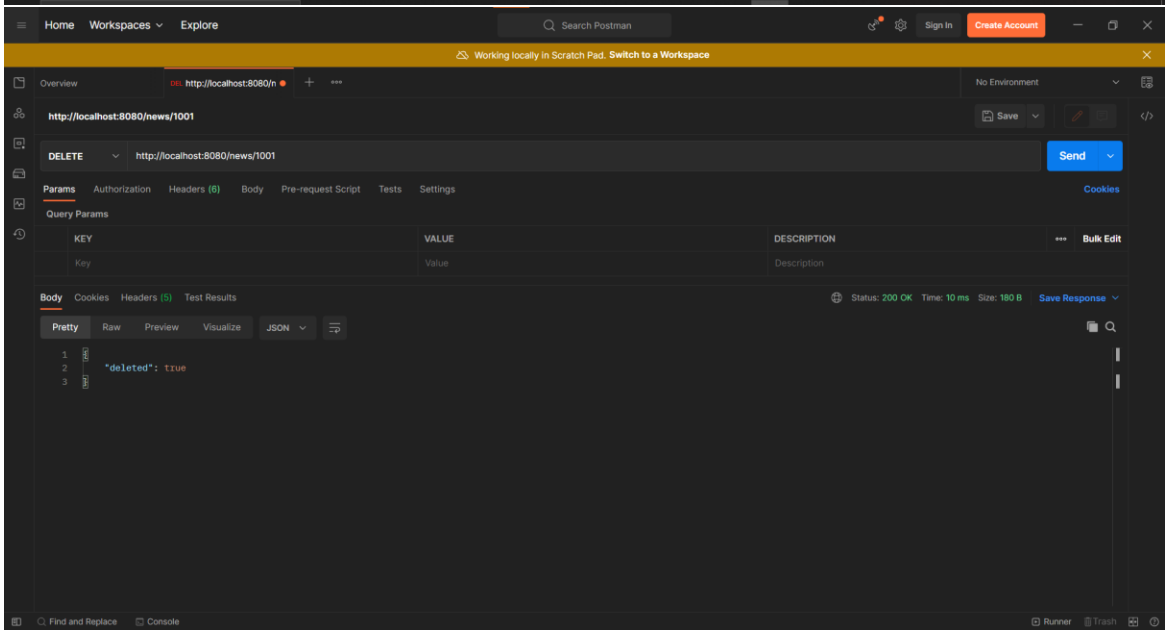


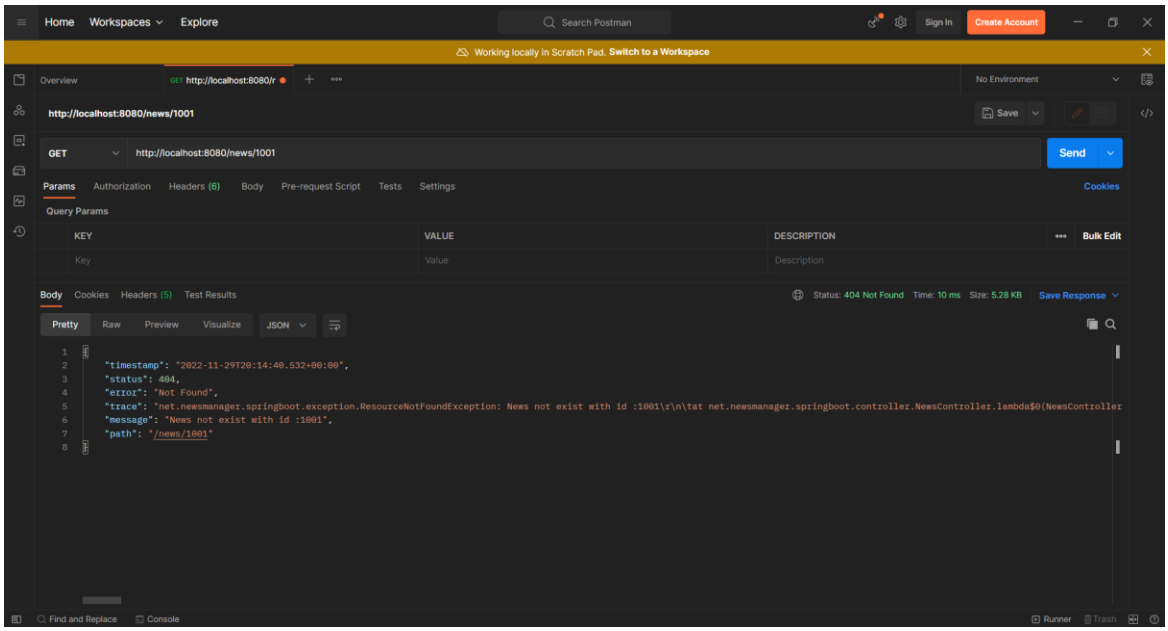
Test 2:



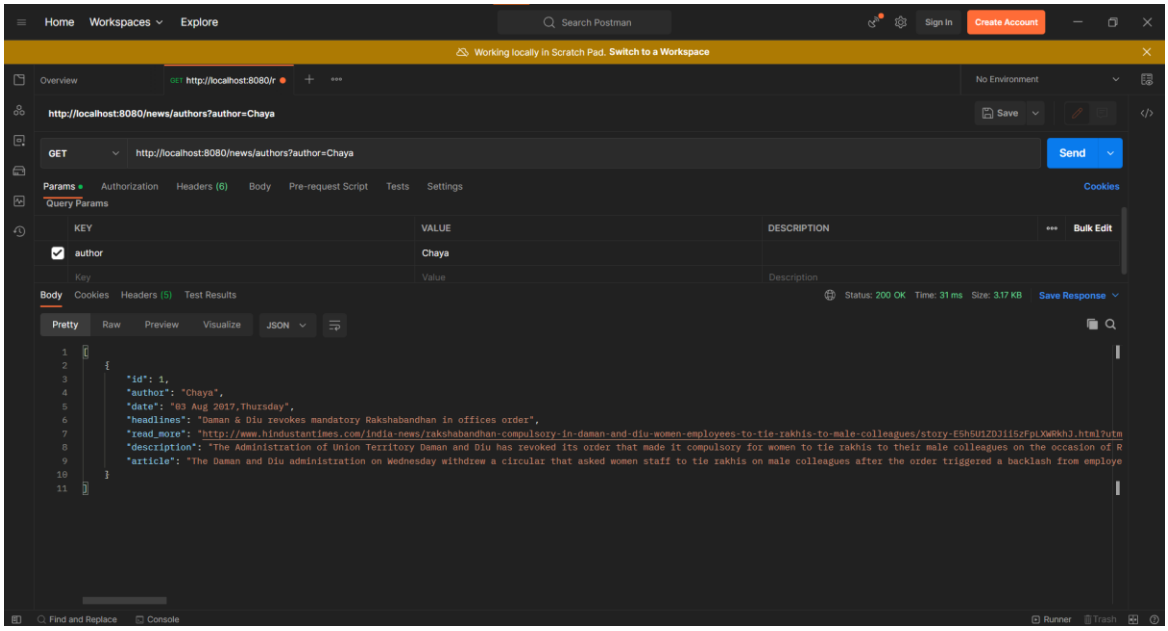
Test 3:


Home
Workspaces
Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Overview
GET http://localhost:8080/r
No Environment
http://localhost:8080/news/1001
Save
GET
http://localhost:8080/news/1001
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 11 ms
Size: 1.55 KB
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1001,
"author": "Arshiya Chopra",
"date": "21 Jun 2017,Wednesday",
"headlines": "Supreme Court denies bail to former HC judge CS Karnan",
"read_more": "http://indiatoday.intoday.in/story/supreme-court-denies-bail-to-hc-karnan/1/983697.html ",
"description": "The Supreme Court on Wednesday denied interim bail to former High Court judge CS Karnan, who was sentenced to six-month jail term for contempt of court. The vacation bench
"article": "The Supreme Court today denied interim bail for controversial former judge CS Karnan. Karnan, who retired in disgrace as a Calcutta High Court judge, was arrested in Coimbator
}
Find and Replace
Console
Runner
Trash


Home
Workspaces
Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Overview
DEL http://localhost:8080/n
No Environment
http://localhost:8080/news/1001
Save
DELETE
http://localhost:8080/news/1001
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 10 ms
Size: 180 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"deleted": true
}
Find and Replace
Console
Runner
Trash


Home
Workspaces
Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Overview
GET http://localhost:8080/r
No Environment
http://localhost:8080/news/1001
Save
GET
http://localhost:8080/news/1001
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 404 Not Found
Time: 10 ms
Size: 5.28 KB
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"timestamp": "2022-11-29T20:14:40.532+00:00",
"status": 404,
"error": "Not Found",
"trace": "net.newsmanager.springboot.exception.ResourceNotFoundException: News not exist with id :1001\r\n\tat net.newsmanager.springboot.controller.NewsController.lambda$0(NewsController
"message": "News not exist with id :1001",
"path": "/news/1001"
}
Find and Replace
Console
Runner
Trash

Test 4:


Home
Workspaces
Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Overview
GET http://localhost:8080/r
No Environment
http://localhost:8080/news/authors?author=Chaya
Save
GET
http://localhost:8080/news/authors?author=Chaya
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
author
Chaya
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 31 ms
Size: 3.17 KB
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 1,
"author": "Chaya",
"date": "03 Aug 2017,Thursday",
"headlines": "Daman & Diu revokes mandatory Rakshabandhan in offices order",
"read_more": "http://www.hindustantimes.com/india-news/rakshabandhan-compulsory-in-daman-and-diu-women-employees-to-tie-rakhis-to-male-colleagues/story-E5h5U1ZDJi15zFpLXWRkhJ.html?utm
"description": "The Administration of Union Territory Daman and Diu has revoked its order that made it compulsory for women to tie rakhis to their male colleagues on the occasion of R
"article": "The Daman and Diu administration on Wednesday withdrew a circular that asked women staff to tie rakhis on male colleagues after the order triggered a backlash from employe
}
]
Find and Replace
Console
Runner
Trash

Test 6: Junit Testing

```
@Test
public void saveNews_success() {
    System.out.print(mockNews.toString());
    newsRepository.save(mockNews);
    List<News> newss = newsRepository.findAll();
    assertThat(newss).isNotNull();
}

@Test
public void findAll_success() {
    Optional<News> newss = newsRepository.findById((long)1);
    assertThat(newss).isEmpty();
}

@Test
  public void should_find_no_news_if_repository_is_empty() {
    List<News> newss = newsRepository.findAll();
    assertThat(newss).isEmpty();
  }


@Test
  public void should_find_news_by_author() {

    List<News> newss = new ArrayList<News>();
    newsRepository.findByAuthorContaining("Chaya").forEach(newss::add);

    assertThat(newss).isEmpty();
  }
```

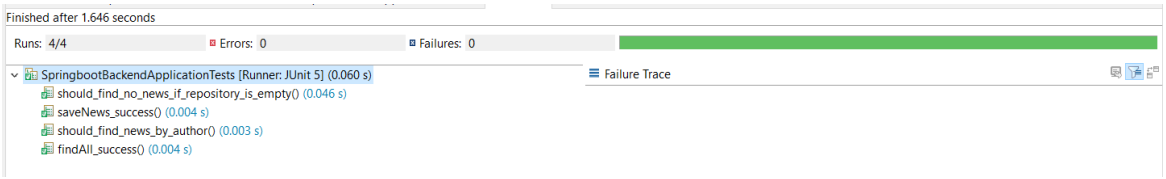


Figure12. Diagram for the News Manager's Testing