Software Design Document for RSS Feed Reader Application

JUNE 5, 2023

Document owner: Nirmani Gunathilaka

**Table of Contents**

1. Overview
   1. Introduction
   2. Purpose
   3. Objective
2. Proposed design
3. Implementation
4. Assumption
5. Future scope
6. Overview
   1. Introduction

This software design documentation outlines the design and implementation of an application that polls a RSS feed every 5 minutes.

* 1. Purpose

The purpose of implementing this application is to manage RSS feed data and store the latest content in a database.

* 1. Objective

This project's object is to fetch RSS feed data periodically to ensure up-to-date retrieval and the latest content accessible in the database.

1. Proposed Design

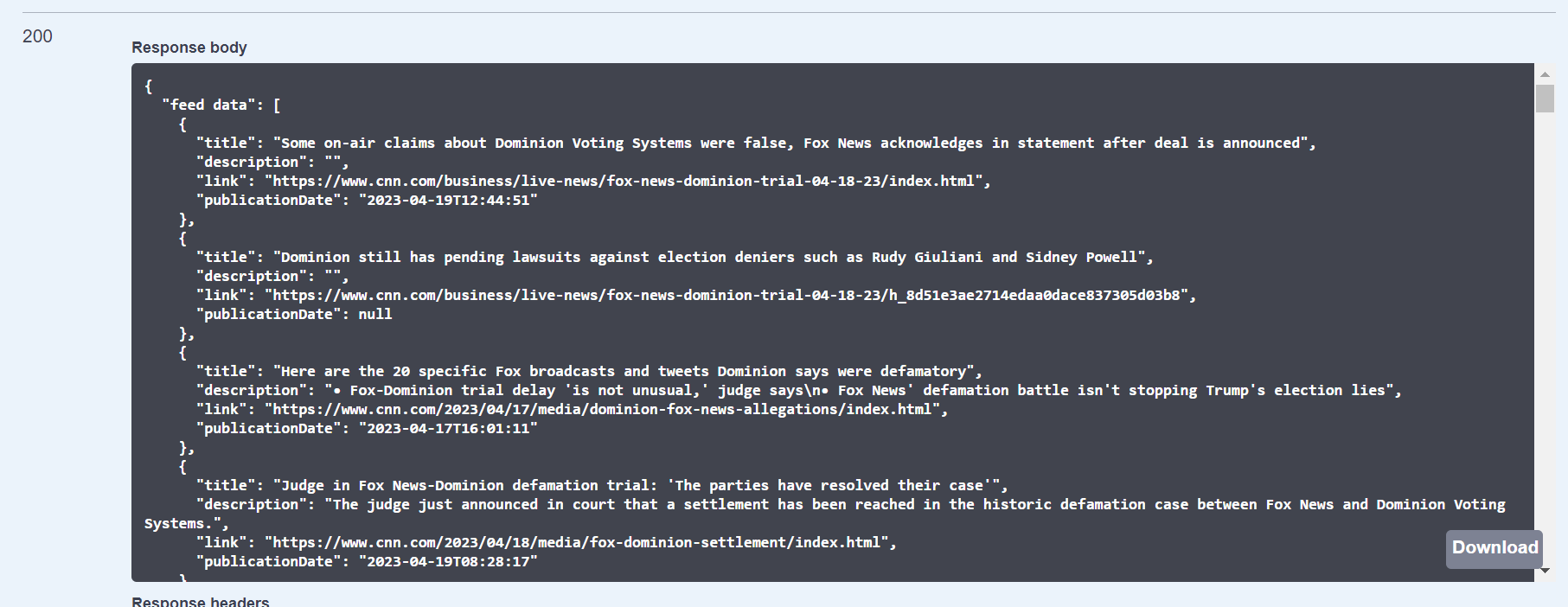
* Retrieve RSS data from the feed URL.
* Extract relevant data from the feed.
* Store the retrieved data in a database. Handling and storing the latest data
* Scheduled the job to execute every 5 minutes, storing data in a database.
* Providing an endpoint to view the latest data with pagination

1. Implementation

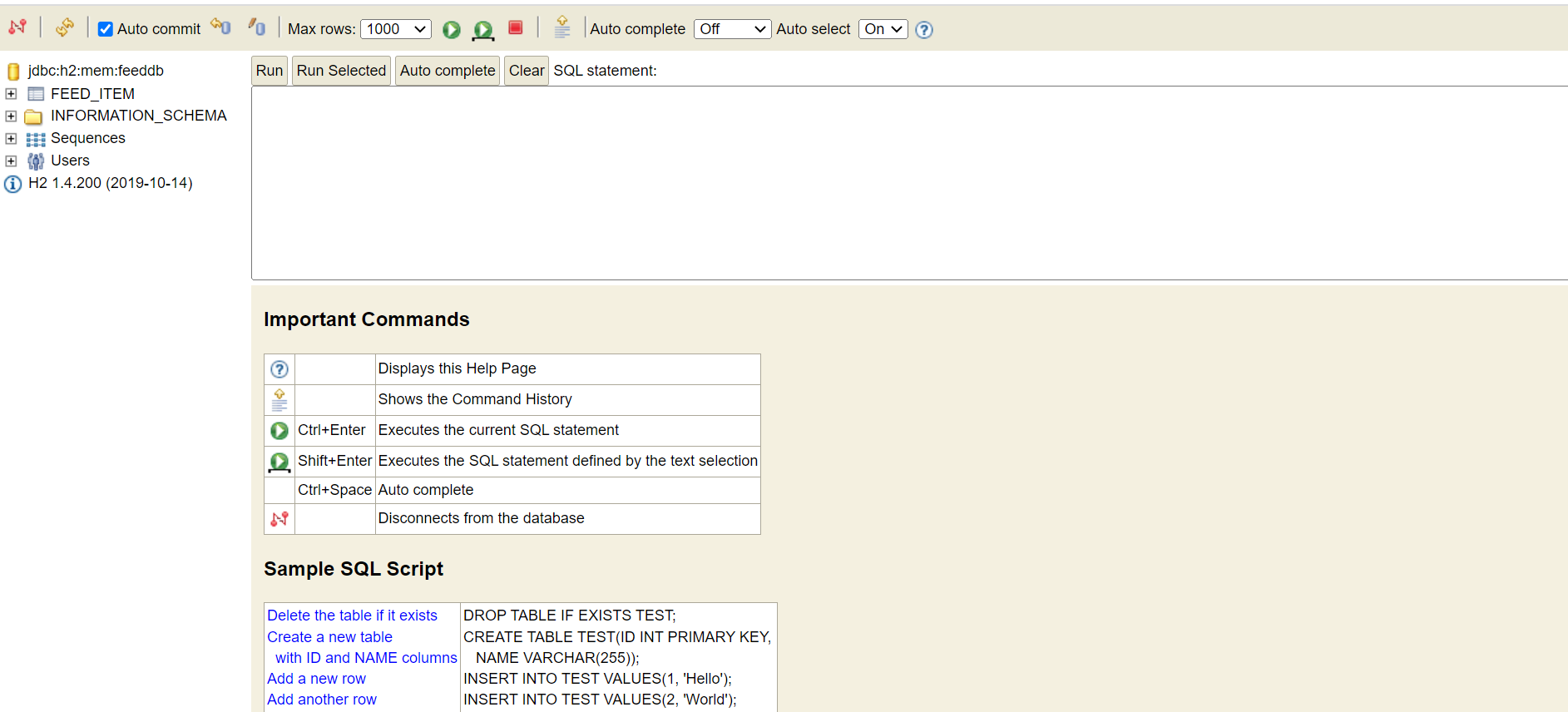
* Implemented a spring boot project using the spring initializer and created the basic structure of the code base. Started implementing version control.
* Chose the CNN feed URL for implementation and fetched the data feed items using the Rome framework. Created new endpoint first to fetch and analyze the item list.

Feed url = <http://rss.cnn.com/rss/cnn_topstories.rss>

Response:



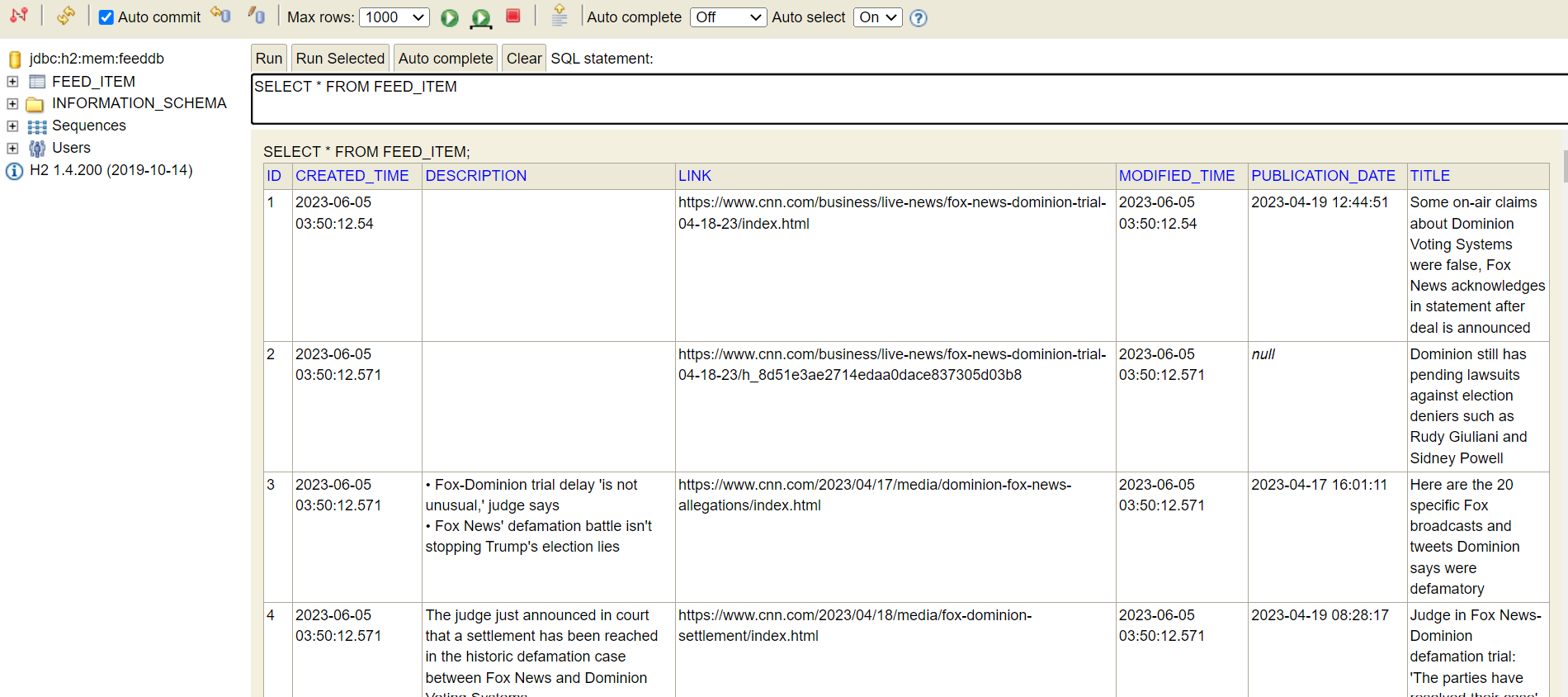
* Configured the H2 database with the Spring Boot application and connected successfully.



* Implemented a cron job scheduler to execute every 5 minutes and fetch the RSS feed items and update the latest content to the database. Stored every timestamp in GMT time zone. Used the @Scheduled annotation. Exposed this as an endpoint also.

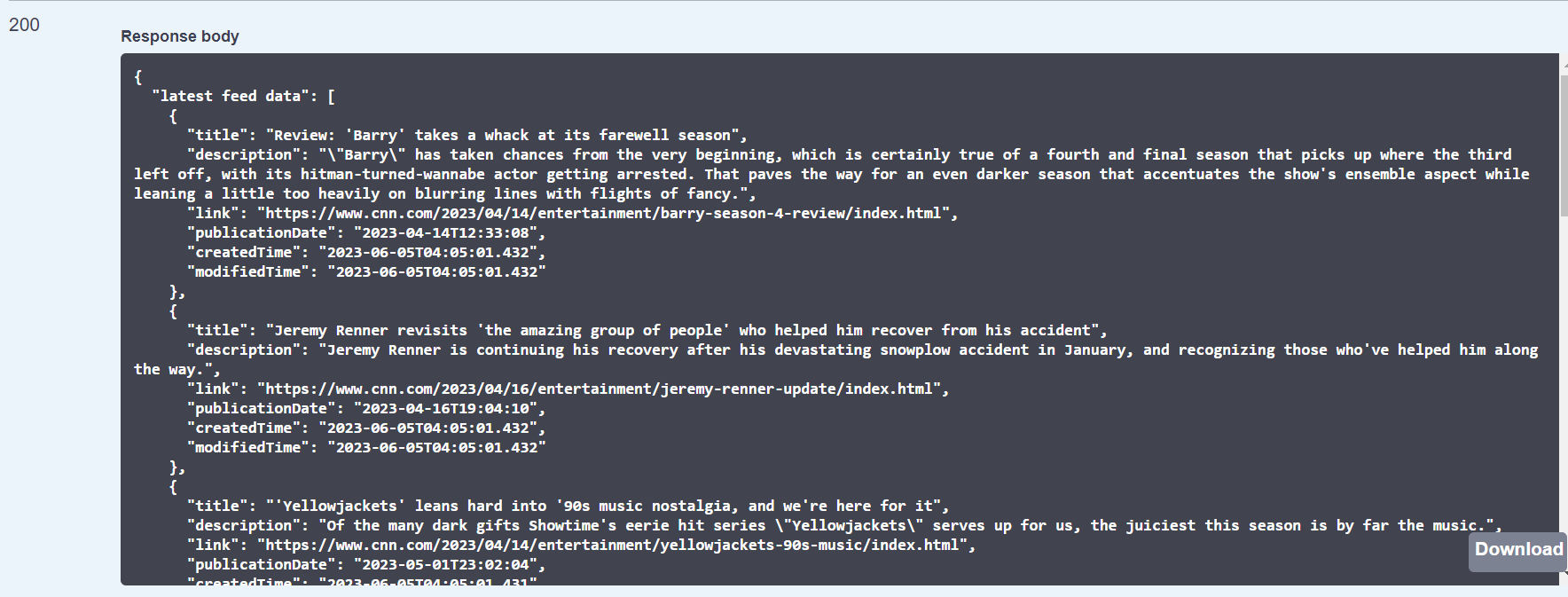
Job Scheduler Cron: 0 \*/5 \* \* \* ?

Stored latest content in database:



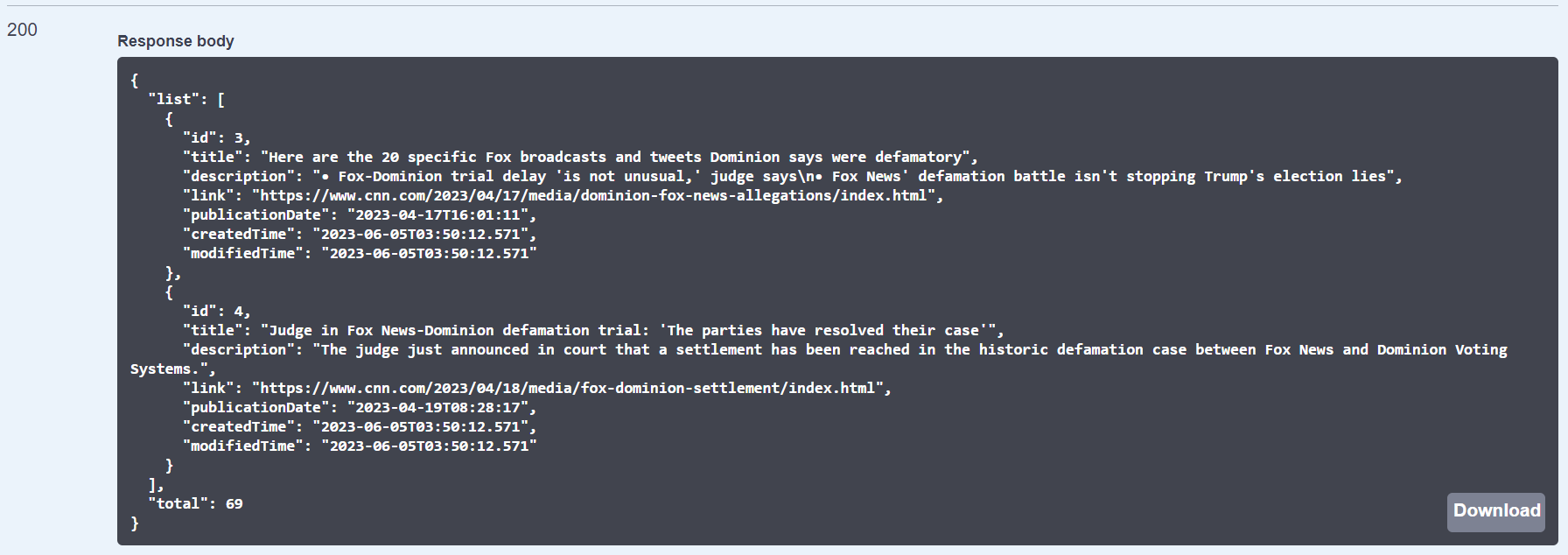
* Implemented an endpoint to retrieve the latest 10 items from the database. Choose the latest items based on created time. used a native JPA query for this.

SELECT \* FROM FEED\_ITEM I ORDER BY I.CREATED\_TIME DESC LIMIT 10



* Implemented an endpoint to retrieve feed items from the database based on pagination.

eg: Paginated data with limit =2, offset = 1 , sort=id and direction=asc



* Exposed all endpoints

GET - <http://localhost:8881/rss/>

POST - <http://localhost:8881/rss/job-scheduler>

GET - <http://localhost:8881/rss/items>

GET - <http://localhost:8881/rss/items-pagination?direction=asc&limit=2&offset=1&sort=id>

* Added exception handling, validations, necessary logs, and comments.
* Conducted dev testing.

1. Assumptions

* Assumed the link of the feed item as unique.

1. Future scope

* Enhance and improve logging mechanism
* Add authentication layer
* Unit testing
* Monitor performance of the scheduler and endpoints