NewsWise: AI-Powered Contextualized News Feed - Phase 1 Deliverables

Github URL: https://github.com/VanshK7/NewsWise

1. Synopsis Document

Project Name: NewsWise: AI-Powered Contextualized News Feed

Introduction:

In today's fast-paced information world, people are overwhelmed by the sheer number of news stories. Navigating this terrain to get relevant, trustworthy, and fair information is a major problem. Furthermore, comprehending complicated issues and spotting potential biases in news sources needs considerable effort and critical thought. NewsWise seeks to solve these issues by offering a tailored and contextualized news experience.

Problem Statement:

- Information Overload: Users struggle to find relevant news amidst the overwhelming amount of content.
- Lack of Context: News articles often lack sufficient context, making it difficult to understand the nuances of complex topics.
- Bias Detection: Identifying and understanding potential biases in news articles is challenging.
- Personalization Limitations: Existing news aggregators often lack sophisticated personalization capabilities, resulting in generic news feeds.

Objectives:

- **Personalized News Feed:** Provide a news feed tailored to individual user interests and preferences.
- **Contextual Understanding:** Utilize AI to analyze news articles and extract key information, including topics, entities, and sentiments.
- **Bias Detection:** Identify and highlight potential biases within news articles, promoting media literacy.
- **Explanation Generation:** Generate simplified explanations of complex topics to enhance user understanding.
- **User Empowerment:** Empower users to make informed decisions by providing contextualized information and insights.
- **Modular Architecture:** Develop a scalable and maintainable system using microservices.

Scope of the Project:

This project encompasses the development of the following core functionalities:

- **News Aggregation:** Fetching news articles from various sources (APIs, RSS feeds, web scraping).
- **Contextualization:** Utilizing an LLM to extract context, identify entities, detect biases, and generate summaries.
- **Personalization:** Recommending news articles based on user interests and article context.
- **User Management:** Managing user profiles, interests, and preferences.
- **API Gateway:** Providing a unified entry point for accessing microservices.
- **Service Discovery:** Implementing service discovery to enable communication between microservices.
- **Configuration Management:** Enabling centralized configuration of microservices.

Technologies Used:

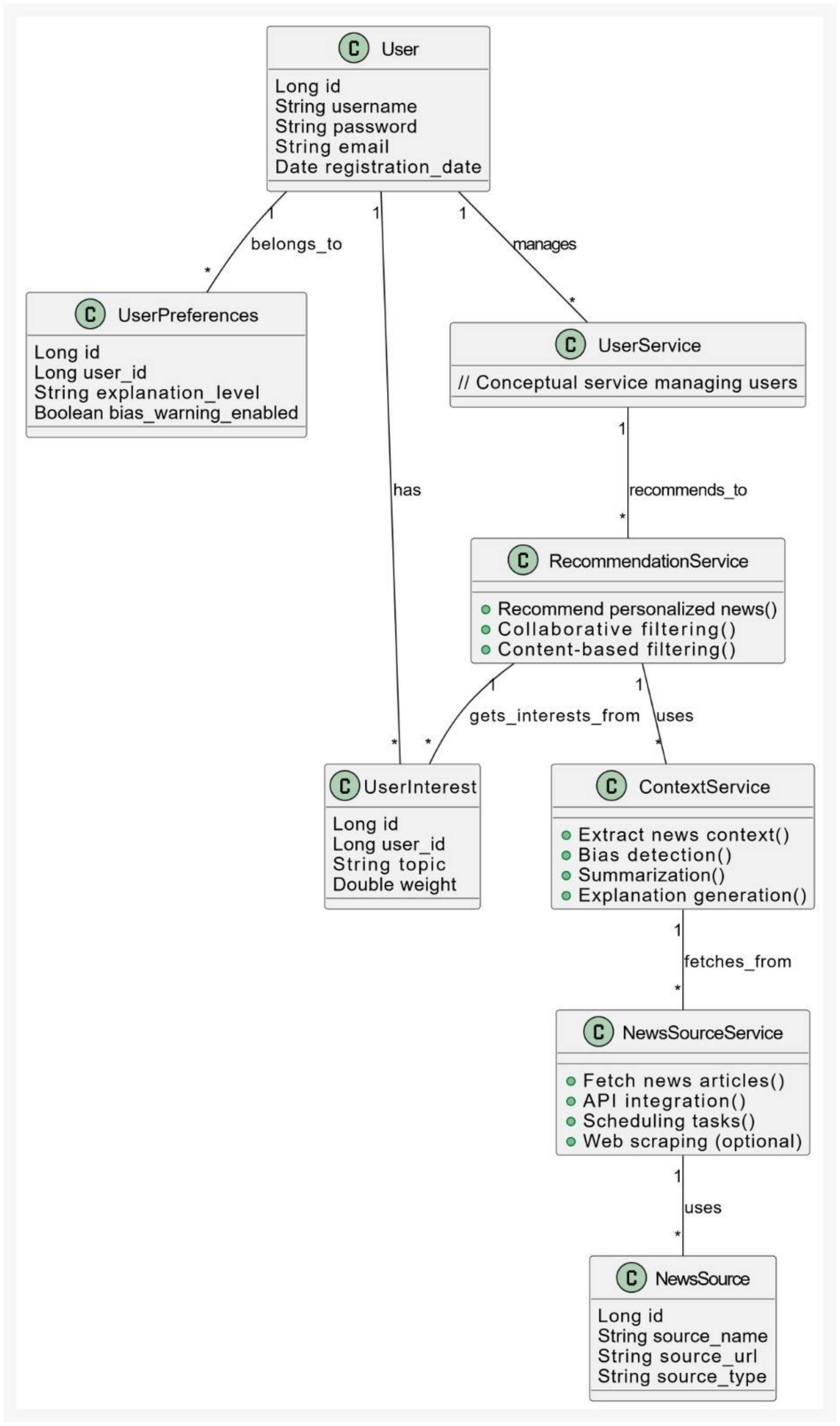
- **Programming Language:** Java
- **Framework:** Spring Boot
- **Database:** MySQL (with Spring Data JPA)
- **Microservices Architecture:** Utilizing RESTful APIs
- **API Client:** Spring WebClient
- **Web Scraping (Optional):** Jsoup (If required for specific news sources)
- **LLM Integration:** OpenAI Java SDK (or similar)
- **Service Discovery:** Eureka
- **API Gateway:** Spring Cloud Gateway
- **Configuration Management:** Spring Cloud Config
- **Authentication & Authorization:** Spring Security, JWT (Recommended)
- **Caching:** Spring Cache
- **Scheduling:** Spring Scheduler
- **Version Control:** Git (with GitHub)
- **Containerization:** Docker
- **Orchestration:** Kubernetes (EKS on AWS)
- **Monitoring:** Spring Boot Actuator, Micrometer, Prometheus, Grafana (ELK Stack or AWS CloudWatch for logging)
- **Resilience:** Resilience4J
- **Cloud Platform:** AWS

Expected Outcome:

- A functional and scalable AI-powered personalized news aggregator.
- A modular system utilizing microservices for maintainability and scalability.
- A user-friendly interface for accessing personalized news feeds, managing interests, and adjusting preferences.
- Integration with external news sources via API calls, RSS feeds, and web scraping (if necessary).

- Implementation of core features like context extraction, bias detection, and summarization using an LLM.
- Secure user authentication and authorization.
- Clear documentation for deployment, maintenance, and future development.

2. ERD (Entity-Relationship Diagram)



Explanation of ERD:

- **Entities:** User, UserInterest, UserPreferences, NewsSource (Optional)
- **Relationships:**
  - User can have multiple UserInterest (One-to-Many).
  - User has one UserPreferences (One-to-One).
  - UserInterest belongs to one User (Many-to-One).
  - UserPreferences belong to one User (Many-to-One).
  - NewsSourceService uses NewsSource (if configured)
  - ContextService uses information fetched from NewsSourceService
  - RecommendationService uses information provided by the UserService and ContextService
  - UserService manages User accounts
-
- **Primary Keys (PK):** id for all tables, autogenerated.
- **Foreign Keys (FK):**
  - user_id in UserInterest and UserPreferences tables referencing User.id.
-
- **Unique Key (UK):** username in the User table.
- **Normalization:** The database design adheres to normalization principles. Specifically:
  - **1NF (First Normal Form):** Each table contains atomic values. No repeating groups.
  - **2NF (Second Normal Form):** In addition to 1NF, all non-key attributes are fully functionally dependent on the primary key.
  - **3NF (Third Normal Form):** In addition to 2NF, no non-key attributes are transitively dependent on the primary key (i.e., no dependencies on other non-key attributes).
- **Microservices and Data Flow:** The diagram illustrates the interactions of different microservices without specifying the exact content of their databases (which are mostly service-specific, outside the core User data). The arrows show how information flows between them.