madlen.io
Esentepe, Büyükdere Cd. No:175/7, Kolektif House
Şişli/İstanbul, 34394

Madlen Software Engineer Interview Project

Objective: Build a backend service that accepts a website URL containing lecture notes, scrapes and organizes the educational content, and stores the data. It also lets users tag contents with additional metadata such as course name and grade level.

Requirements Overview

- Web Scraping: Parse and extract educational content from a given URL.
- Content Structuring: Organize scraped content into logical sections (e.g., titles, sub-sections, bullet points) for consistency.
- Data Persistence: Store both the organized content and metadata in a database for future retrieval.
- Document Tagging: Allow users to tag the scraped content with metadata like course title and grade level.

Task Breakdown

1. Backend API Development

- Requirements:
    1. Create a RESTful API endpoint that:
        - Accepts a URL, course title, and grade level as input.
        - Scrapes the content of the specified URL and organizes it into a structured format.
        - Stores both the organized content and the pagedata in a database.

2. Tech Stack: Python or Java for the API, BeautifulSoup or Scrapy for web scraping, and a database like Sqlite for storage is recommended.

- Implementation Steps:
    1. URL Validation: Verify that the provided URL is accessible and returns a valid HTML response.
    2. Content Extraction: Use scrapers to parse the HTML and extract structured educational content:
        - Extract and organize headers (<h1>, <h2>, etc.) to identify sections and subsections.
        - Extract paragraphs, bullet points, and tables where relevant.
    3. Content Structuring: Organize the extracted content into a structured format based on HTML hierarchy:
        - Use the headers and subheaders as section identifiers.
        - Organize lists, tables, and paragraphs under their respective sections.
    4. Metadata Handling: Include the additional metadata (course title and grade level) along with the scraped content in the final data object.
    5. Database Storage: Store the organized content and metadata in a database.
        - Each entry in the database should contain:
            - URL of the source
            - Scraped content, organized into sections and subsections
            - Course title
            - Grade level
            - Date of scraping (for potential future updates)
            - You may extend here as you wish. Be creative!
    6. API Response: Return a response indicating whether the scraping and storage were successful, along with the ID of the stored entry for reference.

2. Database Design

- Requirements:
    - Create a database schema that allows for efficient storage and retrieval of structured lecture notes along with associated metadata.
    - Use relational DB. Sqlite is preferable for structured data, and easy to set up.

3. Data Retrieval Endpoint

- Requirements:
    - Create an additional API endpoint that allows retrieval of stored lecture notes by:
        - URL
        - Course title
        - Grade level
    - This endpoint should return the organized lecture content and metadata in a structured JSON format for easy use in any frontend or data pipeline.

You are encouraged to extend the project as you wish!