Project Name: Stock Market News and Analysis Platform
Date 24.3.2024
Author: Jani Seppälä

1. Introduction

Often the stock market news and company results are complex pieces of text with a lot of numbers and fancy words including useless business jargon. The goal of this software is to simplify and compress stock market news into a few essential paragraphs. It's designed to quickly inform users about the significance of news items - distinguishing between major and minor developments for specific stocks. By compressing detailed information into more manageable summaries, it aids investors in swiftly understanding the implications of market news, enabling them to make informed decisions regarding their investments.

2. Architecture Design

The architecture of this stock market news simplification software is designed to offer an efficient, user-friendly experience by dividing the system into three main components: front-end, back-end, and database, with integration of external services for additional data.

- **Front-end:** The front-end is developed using React, a powerful JavaScript library for building user interfaces. This choice allows for the creation of a responsive and interactive web application. The front-end consists of several components including login, registration, stock details, and a favorites section, each designed to provide users with a seamless navigation experience throughout the software.
- **Back-end:** Flask, a micro web framework written in Python, is utilized for the back-end. It handles API requests, interfaces with the MongoDB database, and manages user authentication and authorization processes. Flask's simplicity and flexibility make it ideal for this project, allowing for easy integration of additional features or services in the future.
- **Database:** MongoDB, a NoSQL database, is chosen for storing user data, stock information, news articles, and analyses. Its schema-less nature provides the flexibility needed to accommodate the varying structures of stock market news and analysis, making it easier to store and retrieve data efficiently.
- **External Services:** The system integrates with various external APIs to fetch real-time stock data and news. These services enrich the application's content, providing users with up-to-date information and analyses from the stock market. This integration ensures that users have access to comprehensive and current data, enabling them to make well-informed decisions based on the latest market trends.

3. UML Diagrams

UML USE CASE DIAGRAM

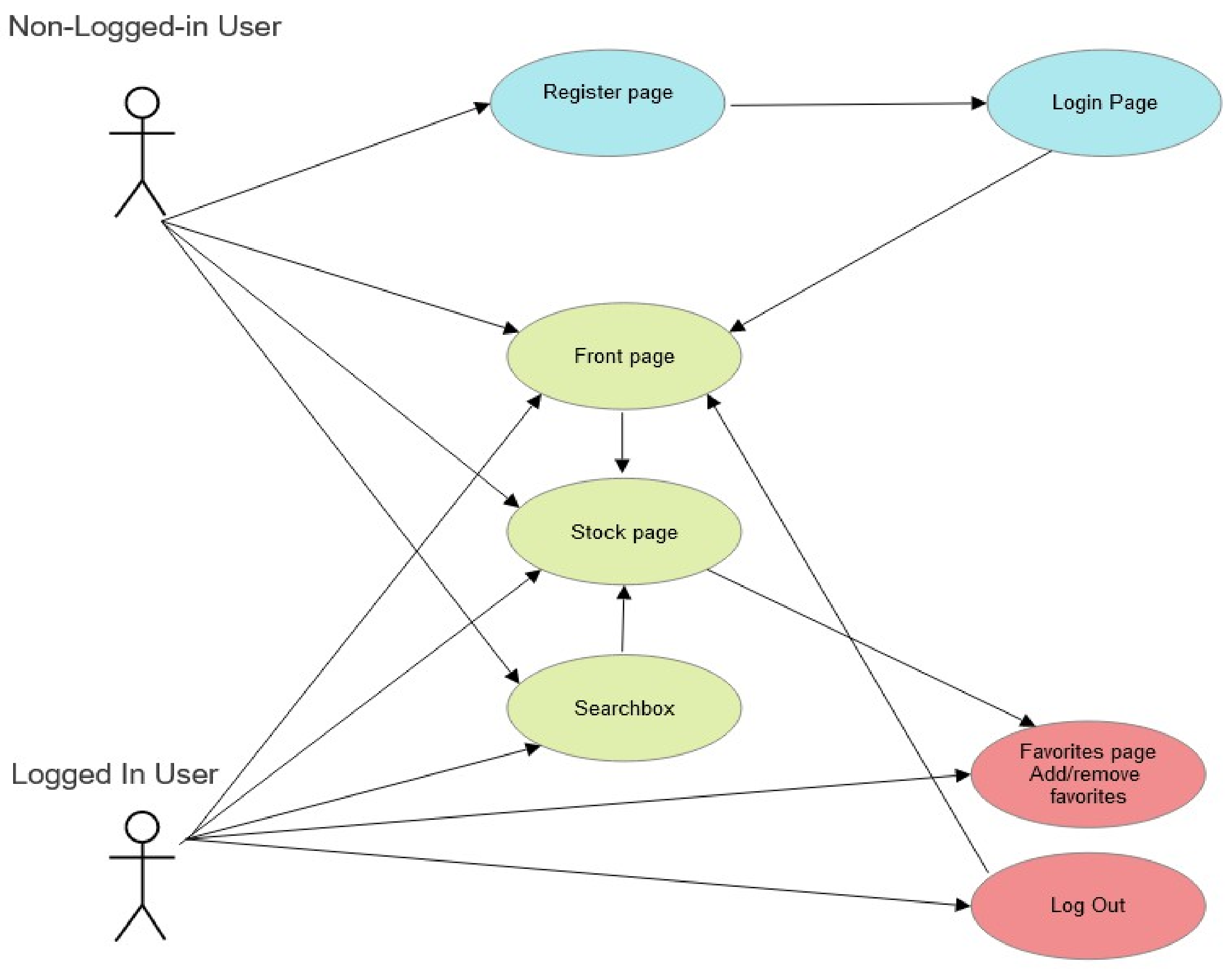


Diagram Description

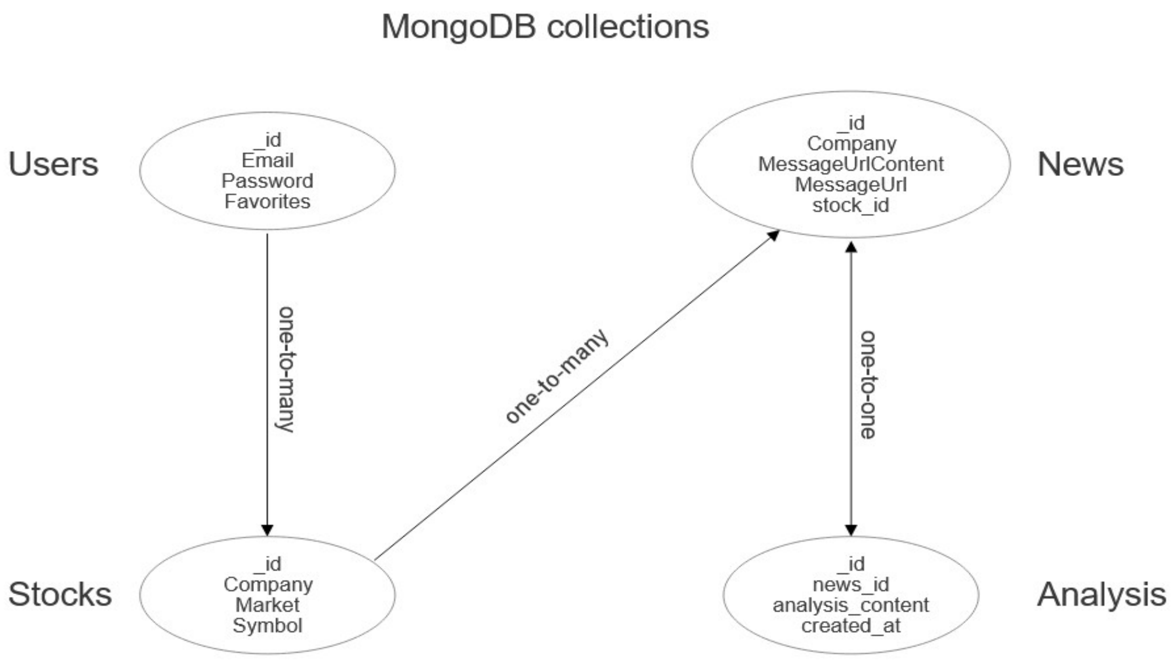
1. **User Interaction Flow:**

   - **Non-Logged-In Users:**
     - Have access to:
       - **Login Page:** Where they can log in to access personalized features.
       - **Registration Page:** Where new users can register. After successful registration, they are redirected to the Login Page.
     - Can view the **Front Page** with all the news but cannot interact with favorite features.
     - **Stock Page:** Can add stocks to their favorites or view specific stock news.
   - **Logged-In Users:**
     - Have access to all pages available to non-logged-in users plus:
       - **Favorite Page:** Where they can view and remove their favorite stocks.

        - Can log out, which redirects them to the Front Page with a logout message displayed.

2. **Front-End Pages:**
    - Front Page
    - Stock Page
    - Favorites Page
    - Registration Page
    - Login Page
3. **Back-End: Flask Application Handling:**
    - User Authentication (Login/Logout)
    - User Registration
    - Favorites Management (Add/Remove)
    - Fetching News and Analysis for Stocks
    - Serving specific stock news on Stock Page
4. **Database: MongoDB named `stock_analysis_app` with Collections:**
    - `users` - Stores user information and their favorites as an array of stock IDs.
    - `stocks` - Stores stock information. Linked to news and analysis documents through stock IDs.
    - `news` - Stores news articles, linked to specific stocks through stock IDs.
    - `analysis` - Stores analyses, linked to specific stocks through stock IDs.
5. **External APIs:**
    - Used by the Flask back-end to fetch real-time stock data and news.

4. Database Design



The database for this project, named `stock_analysis_app`, is structured into four primary collections: `users`, `stocks`, `news`, and `analysis`. Each collection serves a distinct purpose in storing and managing data critical to the application's functionality. Below are the schemas for each collection and the relationships between them:

Users Collection

Schema:

- `_id`: Unique identifier for the user.
- `email`: User's email address.
- `passwordHash`: Hashed password for user authentication.
- `favorites`: Array of `stockIds` that the user has marked as favorites.

Stocks Collection

Schema:

- `_id`: Unique identifier for the stock.
- `company`: Name of the company.
- `market`: Market where the stock is traded.
- `symbol`: Stock ticker symbol.

NewsArticles Collection

Schema:

- `_id`: Unique identifier for the news article.
- `disclosureId`: Unique identifier for the disclosure.
- `categoryId`: Category ID related to the news.
- `headline`: Title of the news article.
- `language`: Language of the news article.
- `languages`: Array of languages in which the news is available.
- `company`: Name of the company related to the news.
- `cnsCategory`: Category of the news.
- `messageUrl`: URL to the full news article.
- `releaseTime`: Time when the news was released.
- `published`: Publication date of the news.
- `market`: Market related to the news.
- `cnsTypeId`: Type ID of the CNS category.
- `attachment`: Array containing any attachments to the news.
- `messageUrlContent`: Content available through the message URL.
- `stock_id`: ID of the stock related to the news.

Analysis Collection

Schema:

- `_id`: Unique identifier for the analysis.

- `content`: Content of the analysis.
- `stock_id`: ID of the stock related to the analysis.

Relationships:

- The `favorites` array in the `Users` collection contains IDs that reference stocks in the `Stocks` collection, allowing users to keep track of their favorite stocks.
- Both `NewsArticles` and `Analysis` collections have a `stock_id` field that links each news article or analysis to a specific stock in the `Stocks` collection. This relationship enables the aggregation of relevant news and analysis for each stock.

This database design supports the application's requirement to efficiently store and retrieve user data, stock information, and related news and analysis. It facilitates quick access to stock-specific news and analysis, thereby enhancing the user experience by providing relevant and timely information.

5. Main technologies used:

- **Python:** The primary programming language for backend development, Python's versatility and readability were pivotal in scripting the server logic, interacting with the database, and processing data.
- **Flask:** A Python micro web framework forms the backbone of server-side logic, facilitating the creation and handling of web server routes and responses with simplicity and flexibility.
- **Flask-PyMongo**: This extension integrates MongoDB with Flask, allowing for seamless data operations within the NoSQL database, which stores user data, stock information, news articles, and user preferences.
- **Flask-JWT-Extended**: Utilized for secure authentication and authorization through JSON Web Tokens, this extension enables secure user sessions and access management.
- **bcrypt**: A robust library for hashing passwords, ensuring that user credentials are securely stored in the database.
- **MongoDB**: Chosen for its flexibility to store varied data structures, facilitating efficient data retrieval and storage.
- **BSON and JSON**: MongoDB stores data in BSON format, while the Flask application frequently converts this data to JSON for API responses, ensuring a smooth data exchange between the database and front-end. This conversion process is critical for manipulating data according to the application's requirements, making data readable for both the server and client sides.
- **JavaScript**: The scripting language powers the application's interactivity on the client side, enabling event handling, DOM manipulation, and dynamic content updates for a smooth and engaging user experience.
- **React**: Leveraging this JavaScript library to craft interactive user interfaces, particularly visible in the dynamic rendering of stock details and news analysis, enhancing the application's responsiveness.

- **axios**: A Promise-based HTTP client employed to communicate with the backend API, fetching data to be displayed and managing user interactions such as adding or removing favorites.
- **Bootstrap & Custom CSS**: The application's styling relies on Bootstrap for general layout and responsiveness, complemented by custom CSS for specific component styling.
- **OpenAI Python SDK & BeautifulSoup**: The OpenAI SDK interacts with GPT models for generating analysis from the prompt, while BeautifulSoup parses HTML content from news sources, extracting and processing textual data for user consumption.

Future additions:

- Add all of nordic market stocks or later whole world (working on it!)
- Make Chatgpt prompt better and add stock history in the prompt among other info (working on it!)
- Test other gpt AI's to see what produces best results
- Make the analysis available in english and other nordic languages beside finnish.
- Make twitter account that post updates and info about this site to get more customers
- Add more info stocks page
- Add more news sources than nasdaqomxnordic
- Add web search for some news that would need more content to know better like appointing new Ceo
- Add web and twitter search for some stocks that are having unexpected volatility but no news
- Add forums for users to discuss the app and stocks
- Add monthly subscription for users that would give them access to more detailed news analysis and perhaps analysis of the company itself