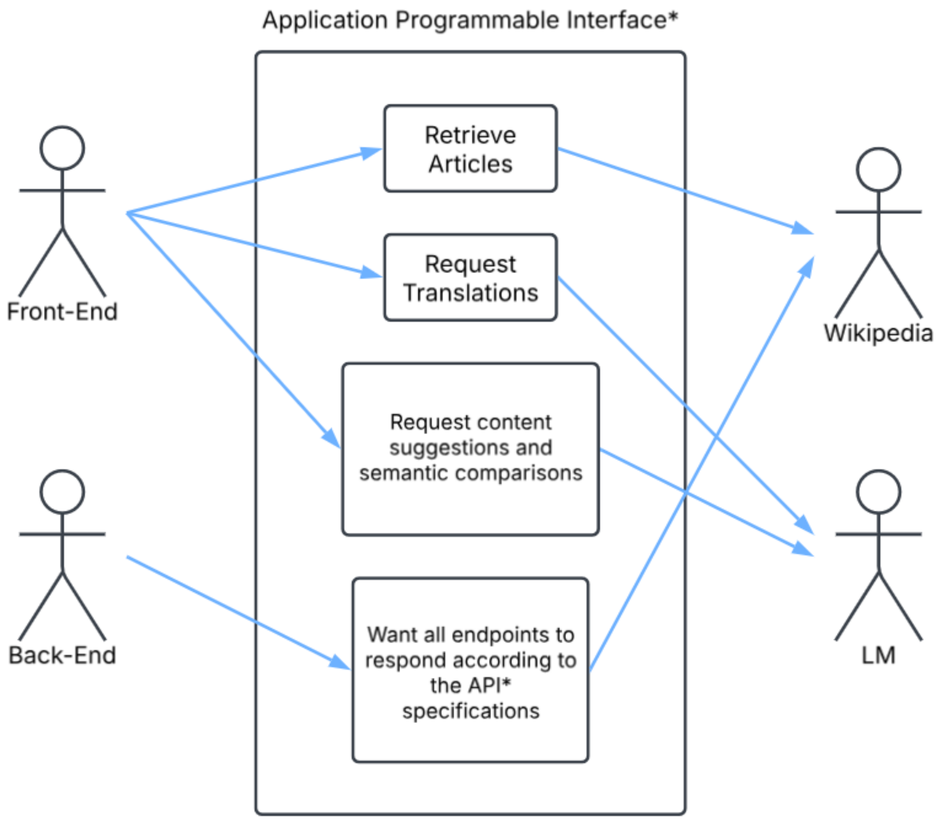
System Requirements

🥐Team Scrum-ptious🥐

Begin with a narrative to give a general overview of the system's *functional* requirements. Provide one big use case diagram illustrating overall system functionality with a specific focus on the features being implemented by the team. Describe each use case in an easy-to-understand natural language (10 pts)

Overview

The high level goal is to enhance Wikipedia's multilingual content by using machine learning for accurate translation and semantic analysis, recommending edits for translational consistency and targeting misinformation. Our task is designing the API and constructing middleware to ensure that front- and back-end developers can send and retrieve information effectively. Primary functions: retrieve articles, request translations, request content suggestions/semantic comparisons, ensure correctness of middleware implementing API.


Application Programmable Interface*
Front-End
Back-End
Retrieve Articles
Request Translations
Request content suggestions and semantic comparisons
Want all endpoints to respond according to the API* specifications
Wikipedia
LM

User Stories

User stories **bolded** are EPICS

1. As a front-end developer, I want to be able to retrieve articles from multiple Wikipedia languages via API, so that I can obtain data for translation comparisons.
    - Size: 3
    - Priority: High
    - Precondition: The requested language is supported by the Wikipedia API
    - Post condition: The article content is visible to user in a specific language
2. As a front-end developer, I want to request translations for the page I am displaying to the end users, so that they can view alternate versions of a page that they can understand.
    - Size: 3
    - Priority: Medium
    - Precondition: Articles in other languages have been fetched from Wikipedia
    - Postcondition: The translated article is available
3. **As a front-end developer, I want to request content suggestions and semantic comparisons via the API from the LM so that I can present users with a comparison view of the original and translated content.**
    - Size: 21
    - Priority: Medium
    - Precondition:
        - Wikipedia articles have been retrieved
        - LMs are available and configured for use
    - Postcondition: The requested suggestions and comparisons are provided
4. **As a backend developer, I want an API so that I can communicate with other components of the system consistently and reliably.**
    - Size: 13
    - Priority: High
    - Precondition: An established link/connection with the UI, back-end, and Wikipedia API exists.
    - Postcondition: Communication flows smoothly.

Nonfunctional Requirements

Provide a separate list of any relevant *nonfunctional* requirements (5 pts)

- **Security**: API requests must be authenticated using API keys to prevent unauthorized access.
- **Performance**: The API should process requests within a short time under normal load conditions.
- **Scalability**: The middleware should support a large amount of requests without significant performance degradation.
- **Error Handling**: The API should return standardized error messages with appropriate HTTP status codes.
- **Logging & Monitoring**: All incoming requests and outgoing responses should be logged for debugging and analytics.
- **Consistency**: All API responses should follow a uniform structure to ensure seamless front-end integration.

Glossary

Include a glossary that defines all relevant terms that may have a special meaning in the context of the system (10 pts)

- **Wikipedia:** A free online encyclopedia where people can read and edit articles in many languages. In this project, Wikipedia is used as the main source for retrieving articles for translation and comparison.
- **UI – User Interface:** An interactive graphical representation of the program and data that allows users to interact with the system.
- **LM – Language Model:** A machine-learning model using algorithms to process human language to generate translations, analyze content, and provide semantic comparisons.
- **API – Application Programming Interface:** A collection of method definitions available for usage by modules of the program.
- **API Endpoints:** URLs where the API receives requests to handle.
- **Middleware:** A component that sits between different parts of a software system, controlling how data flows between them.
- **Front-end developer:** A developer working in the UI app of the Symmetry repository responsible for integrating API responses and designing user interactions
- **Back-end developer:** A developer working on the Wikipedia APIs and LM APIs, responsible for handling data retrieval, processing, storage, and security.
- **End users:** People who interact with the UI application to request translations, compare content, and access Wikipedia data through the system.
- **CORS - Cross-Origin Resource Sharing:** Rules to ensure that only specific domains can access an API, preventing unauthorized web requests
- **API keys:** Unique credentials used to authenticate and control access to an API, ensuring that only authorized clients can send requests.
- **HTTP requests:** Communication method used in front-end application to communicate with backend APIs
  - **POST:** HTTP request that has a request JSON body for handling complex data that can’t be sent in query parameters
  - **GET:** HTTP request for retrieving data using query parameters used for getting data from Wikipedia API
- **Body parameters:** Parameters included in HTTP request, typically in JSON format
- **Query parameters:** Parameters included in HTTP request, typically within the URL