Sprint 1 Report

🥐Team Scrum-ptious🥐

What functionality does the system have at the end of this sprint? List user stories that you successfully implemented during this sprint (5 pts)

- User Story 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.
- User Story 2:
    - As a front-end developer, I want to request translations for the page I am displaying so that end users can view the article in different languages.
- User Story 3:
    - As a backend developer, I want to configure middleware for API access to ensure secure data access.

Key functionalities Implemented:

User Story 1: We enabled the retrieval of articles in multiple Wikipedia languages via the API. To verify functionality, we examined the endpoints using Postman and Swagger UI, confirming that they returned successful JSON responses as expected.

User Story 2: We ensured that the endpoint for Wikipedia translation scraping is implemented and that the API will handle the correct Wikipedia articles, and the translation (which is available for that article) as parameters.

User story 3: We made sure that middleware is properly configured, and CORS origin handling is implemented in order to manage resource access from outside domains.

Did you end up making any changes to any of these user stories? Did you break down any further user stories? Did you identify any new user stories during this sprint and, if so, did you add them to the product backlog or decide to implement them right away? Explain (5 pts)

We did not make any changes to our user stories during this sprint. However, some of our user stories depend on the work of other teams for full implementation, particularly the UI team and ML team since we are the middle ground between multiple different

systems. Additionally, we identified security risks in our current implementation, which led us to generate new user stories related to the API security improvements.

User Story 2: In order to fully display to end users the article in different languages, we will need to retrieve the improved translation of the article from the ML, as well as the highlighted differences between the two. If successfully passed and rendered by the UI, this story will be fully functional. However, we have completed the task within our scope, and will continue this story throughout future sprints.

User Story 3: We were able to successfully configure middleware access, but this is not secure.

```
31      # Allow all origins (be cautious with this in production)
32      app.add_middleware(
33          CORSMiddleware,
34          allow_origins=["*"],  # You can specify the allowed origins here
35          allow_credentials=True,
36          allow_methods=["*"],
37          allow_headers=["*"],
38      )
```

As you can see, all origins are allowed to access the resources of the backend through our API, which is very insecure. This has generated a new user story of securing this implementation, and ensuring we practice secure software design for our future implementations

There also were a few bugs and design choice concerns which were raised by our tech lead throughout development, which lead to further research during this sprint:

JSON > XML: We conducted targeted research of XML versus JSON to determine the optimal data format for our project. A suspicion was raised that XML might reduce some of the processing required for a lot of the LM's that the backend team are planning to implement, and might be a better choice over the current XML implementation. However, we have deduced that the JSON would overall be a better design decision for data representation since unique XML schemas would have to be developed for each LM, and this would prove to add more complexity to our implementation than some light pre-processing of raw Wiki data would. Our project is also already tailored to JSON as it is developed using FastAPI, so that would require a considerable amount of labor to convert the already developed bones of our project to a new data representation.

Porting issue investigation: Our technical lead had difficulties running the application. The previous maintainer of the project identified a conflict with another application binding to the same port as a likely candidate. While it turns out that our tech lead's issues were an installation issue, our investigation did turn up the fact that the middleware is launched with a hard-coded port from an odd place internally. We are tasked with communicating this issue to the bundling team, who will be responsible for adding a configuration and launching the middleware.

Elastic search: The application is supposed to use either a URL or a page title. The UI implementation only currently uses the URL API, even if the content is not a URL. The UI is not supposed to be responsible for in-depth logic not directly related to presentation, so we will need to retool the API for clarity of use and add more logic to determine the type of input ourselves in the middleware prior to making a request to Wikipedia's API.

What are the "lessons learned" at the end of this sprint? What would you do differently next time? Explain (5 pts)

- **Intergroup development**: Our code interacts with that of both the ML team and UI team. To avoid conflicts and confusion, we plan to reorganize our code to add clarity of ownership and schedule a meeting with both teams to discuss how to code effectively and properly branch off of each other’s work.

- **Regional time clarity**: Because our team, the client, and two of the other teams are in different time zones, it has been challenging to schedule meetings. For example, our team is in Eastern Daylight Time (EDT), while the others are in Arizona, which uses Mountain Standard Time (MST) and does not have daylight saving time. Top search results automatically "helpfully" convert MST to MDT for comparison. We have learned to check the time zones carefully when scheduling, use time converters to ensure everyone is on the same page, and confirm meetings with everyone.

- **Accountability**: Setting reminders and holding each other accountable for specific subdivisions of work helps ensure consistent progress and avoid extended group sessions for bulk work. This will help lower team stress levels and improve the quality of our work.

- **Documentation**: Actively reading documentation (e.g., about CORS), using tools like Postman, and exploring built-in solutions are very helpful to enhance our understanding of the project. This helps us solve problems more efficiently and propose better ideas and solutions.

Provide an updated numbered list of all user stories yet to be implemented; indicate pre- and post-conditions (5 pts)
Remaining User Stories:

4. As a back-end developer, I want to store content from both Wikipedia and the LM, so that I can perform comparisons between the original article and the LM-generated content to identify semantic differences.

- Size: 5
- Priority: Medium
- Precondition: Wikipedia articles and LM translations have been successfully retrieved
- Postcondition: The semantic comparison results indicate whether the translation is valid.

5. As a back-end developer, I want to ensure that APIs align with the front-end team's requirements so that the data delivered is in a format that can be seamlessly integrated into the UI, ensuring a consistent and efficient user experience.

- Size: 3
- Priority: High
- Precondition: HTTP API request endpoints are already defined
- Postcondition: API requests are functional for the front-end

6. As a back-end developer, I want an API that retrieves both the original and LM-translated versions of an article so that other front end developers can render and display this content.

- Size: 5
- Priority: High
- Precondition: Both original article content from Wikipedia and the translated content from the LM are available
- Postcondition: Both the original and LM-translated articles are usable by front-end developers.

7. As a back-end developer, I want consistent response structures across all endpoints so that consuming services can reliably use the data.

- Size: 3
- Priority: High
- Precondition: All API endpoints have been implemented.
- Postcondition: Every API endpoint returns responses in a standardized JSON format as defined in the API specifications.

8. As a back-end developer, I want error-handling mechanisms that return meaningful error responses so that consuming services can gracefully handle failures.

- Size: 5
- Priority: High
- Precondition: API endpoints are in place and operational.
- Postcondition: Each endpoint returns standardized error responses (with appropriate HTTP status codes and descriptive error messages) when failures occur.

9. As a back-end developer, I want logs of incoming requests and outgoing responses so that I can debug issues and monitor API behavior.

- Size: 3
- Priority: Medium
- Precondition: API endpoints are live and handling traffic
- Postcondition: API requests and responses are recorded in a centralized logging system.

<u>New User Stories:</u>

10. As a back-end developer, I want our configured middleware to be exceptionally secure and have selective origin parameters.

- Size: 3
- Priority: Medium
- Precondition: Rules exist for middleware specifications (CORS handling, API key management, etc.).
- Postcondition: The APIs are secured to only allow certain domains, types of requests, and authorized users.

11. As a developer, I want my API to be able to fetch articles based on the URL or Title given so that users don’t have to rely solely on the URL.

- Size: 5
- Priority: Medium

- Precondition: The input provided is either a valid URL or a valid title corresponding to an existing Wikipedia article.
- Postcondition: The API successfully retrieves and returns the article when given either a URL or title.

12. As a developer, I want the port used by the internal server process to be configurable so that I can use Symmetry in tandem with other locally hosted processes.

- Size: 3
- Priority: Medium
- Precondition: The server currently runs on port 8000 when being launched locally, which can lead to crashes when other applications are using it.
- Postcondition: The internal server will be able to provide dynamic ports so that there are no port conflicts with other running applications.

13. As a developer, I want the repository to be modular so that it's easy to get to files and to be understandable

- Size: 3
- Priority: High
- Precondition: The codebase is maintained in a centralized version control system with limited modularity.
- Postcondition: The repository is restructured into clear, well-defined modules or components, and files are organized in a way that makes them easily accessible and understandable to all developers.

Given the current functionality of the system and taking into account the pre- and post-conditions, identify a subset of user stories to be implemented during the next sprint. Be sure that the cumulative size of the selected user stories is about 1/4 of the size of the full backlog. Describe the functionality that your (partially implemented) system will have at the end of this sprint. (5 pts)

Sprint 2:

6. As a back-end developer, I want an API that retrieves both the original and LM-translated versions of an article so that front end developers can render and display this content

- Size: 5

- Priority: High
- Precondition: Both original article content from Wikipedia and the translated content from the LM are available
- Postcondition: Both the original and LM-translated articles are usable by front-end developers.

7. As a back-end developer, I want consistent response structures across all endpoints so that consuming services can reliably use the data.

- Size: 3
- Priority: High
- Precondition: All API endpoints have been implemented.
- Postcondition: Every API endpoint returns responses in a standardized JSON format as defined in the API specifications.

8. As a back-end developer, I want error-handling mechanisms that return meaningful error responses so that consuming services can gracefully handle failures.

- Size: 5
- Priority: High
- Precondition: API endpoints are in place and operational.
- Postcondition: Each endpoint returns standardized error responses (with appropriate HTTP status codes and descriptive error messages) when failures occur.

13. As a developer, I want the repository to be modular so that it's easy to get to files and to be understandable

- Size: 3
- Priority: High
- Precondition: The codebase is maintained in a centralized version control system with limited modularity.
- Postcondition: The repository is restructured into clear, well-defined modules or components, and files are organized in a way that makes them easily accessible and understandable to all developers.

<u>If possible:</u>

12. As a developer, I want the port used by the internal server process to be configurable so that I can use Symmetry in tandem with other locally hosted processes.

- Size: 3
- Priority: Medium

- Precondition: The server currently runs on port 8000 when being launched locally, which can lead to crashes when other applications are using it.
- Postcondition: The internal server will be able to provide dynamic ports so that there are no port conflicts with other running applications.

Description: For Sprint 2, we propose implementing these user stories, which together have a cumulative size of 16 points, if possible 19, of 52.

For User Story 6: The API will be able to retrieve the new article generated by the LM with the original article, and comparisons between the two. The API will then send this information to the front end for rendering and display.

User Story 7: We must ensure that every API endpoint returns responses in a standardized JSON format. The consistency will enable consuming services to reliably integrate and use the data.

User Story 8: We must introduce error-handling mechanisms that provide meaningful error responses and descriptive messages, ensuring that any API failures can be managed gracefully.

User Story 13: The repository must be restructured so it can be more well-defined and have modular components. This will make it easier for future developers to navigate, understand, and maintain the codebase.

User Story 12: We must make the internal server's port configurable to avoid conflicts with other locally hosted processes.