# Sprint 2 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)**

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

- Size: 5
- Priority: High
- Precondition: All schemas have been designed.
- 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 organized into easily understandable and maintainable packages.

- 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.

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.

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

- Size: 2
- 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.

**Key functionalities Implemented:**

**User Story 7:** Have designed (and redesigned) multiple possible schemas which are possible for the return of compared data from the backend. Have constructed a general We have a plan for what we will be implementing as our comparison endpoint, but we are still waiting on the ML team to finish the implementation of the schema.

**User Story 8:** Implemented multiple different error-handling messages for previously unconsidered cases, such as:

- A general code for edge cases not considered
- If the language to be compared to is not selected

- The Wiki API is not configured correctly or sends malformed data

This user story will be revisited when the comparison endpoint is implemented.

**User Story 13:** We have prototyped a version of the repository with a more modular structure under the general API folder, which separates the logic for the middleware, the logic for the UI API, as well as the logic for the ML API.

**User Story 12(migrated):** Decided this task was better suited for the platform team after a conference with the tech lead; handoff successful.

**User Story 10:** Secured the middleware CORS handling to make sure resources can be accessed with authorization.

**User Story 11:** Have successfully augmented the fetch endpoint to accept either the URL or just the title of the desired article from the user on the initial fetch request.

**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)**

**User Story 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.)*: This user story was not completed during our current sprint, as we had the specification for the MVP redefined, and will likely not be tackling LM translated articles in our development cycle.

**User Story 12:** After further examining the logistics of the configuration of the port declaration, we decided this would be better off for another team

**User Story 7:** This took up a majority of our sprint. We needed to have multiple meetings discussing how to implement the comparison schema and how the front end and back end would be passing information. After these meetings, we had to design multiple schemas in coordination with the ML team for a design that they would be willing to implement.

**INSTEAD**: User Stories 13, 10, and 11 were added to our sprint to make up for the lack of User Story 6. These user stories also made sense in terms of the momentum of the team and direction from our tech lead throughout the sprint.

**NEW STORIES:**

**14**. As a front-end developer, I want to be able to have the user select what language their source article is in via a drop-down menu

- Size: 5
- Priority: Low
- Precondition: Article fetch endpoints are fully implemented
- Postcondition: The user will be able to select the article only using the title of the article and by selecting the language in the drop-down spinner

**15**. As a developer, I want endpoints to follow consistent naming conventions so requests are intuitive and not repetitive

- Size: 2
- Priority: Medium
- Precondition: All endpoints have been designed, with some unnecessary declarations
- Postcondition: Every API endpoint follows robust naming conventions and has a unique function

**CHANGED 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.

**CHANGED TO**

**4**. As a front-end developer, I want the target language article to be returned in the same language as the source article so that the user can understand the differences

- Size: 13
- Priority: Medium
- Precondition: MVP is fully functional; comparisons can be made between two articles from Wikipedia in different languages
- Postcondition: The ML can translate Wikipedia articles to the user's desired language, and the comparisons will still be accurate and displayed to the front end.

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

- **Intergroup Development:** Unfortunately, there was a massive miscommunication between our tech lead and the three different development teams. Two of the groups (API, UI) believed that the MVP would incorporate the translation of the target article back to the source language, but this was not what the ML team expected. The ML team was correct, but this required multiple messages between the product owner and the tech lead, as well as a meeting between the 3 teams. In the end, we figured out the conflict, but this massively delayed our development this sprint. We learned it is extremely important to make sure all development teams are on the same page regarding implementation goals before starting to correspond together.

- **Compromise:** Within our group, our members are extremely busy with work or other personal issues. We have learned this sprint how to pick up each other's slack and make sure that we can finish our goals when some of our team is not available. Our group would not have been able to complete much more work with the extra hands on deck, and most of our shortcomings came from miscommunication between the tech lead and the teams.

- **Regional Time Clarity:** In this sprint, we have three separate meetings (the most we've had in one sprint), all involving members of different time zones. We were able to communicate our time zones correctly and make sure that all meetings were accessible and every member could attend. We used regional time zone tools on https://schej.it/ in order to ensure this.

- **Paired Work:** We successfully combated the issue of an overreliance on entire group work this sprint. While it is beneficial to be working on some things with the entire group, productivity is maximized when it is possible to work asynchronously alone or with a few members at a time. We would frequently pair up into groups of 2-3 people when working on a majority of the tasks in the backlog and then recap to the group using our Discord server for team communication.

**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 front-end developer, I want the target language article to be returned in the same language as the source article so that the user can understand the differences

- Size: 13
- Priority: Medium
- Precondition: MVP is fully functional; comparisons can be made between two articles from Wikipedia in different languages
- Postcondition: The ML can translate Wikipedia articles to the user's desired language, and the comparisons will still be accurate and displayed to the front end.

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: 5
- 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 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 the source language and the target language are available in Wikipedia
- Postcondition: Both the original and target articles show comparisons in the front end.

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

- Size: 5
- 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.

**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: Low
- Precondition: API endpoints are live and handling traffic
- Postcondition: API requests and responses are recorded in a centralized logging system.

**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 3:**

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: 5
- Priority: High
- Precondition: HTTP API request endpoints are already defined
- Postcondition: API requests are functional for the front end

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

- Size: 5
- Priority: High
- Precondition: Both original article content from the source language and the target language are available from the back end
- Postcondition: Both the original and target articles show comparisons in the front end.

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

- Size: 5
- 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.

**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: Low
- Precondition: API endpoints are live and handling traffic
- Postcondition: API requests and responses are recorded in a centralized logging system.

**Description**: For Sprint 3, we will be focused on implementing the functionality of the MVP. This entails complete schema configuration between the front end and the back end, with all necessary endpoints in place and functional. This version of the project will only support comparing one article in the user language provided by Wikipedia and the target language translation of the article also provided by Wikipedia. The UI will be able to highlight the comparisons made by the backend of these two articles and render them in a different view.

Points: 18/64