Sugar Labs (Sugar)
Institute (SugarLabs) Proposal (GSoC 2024)

Project Title: Add an AI-assistant to the Write Activity
Project Length: 350 Hours
Project Difficulty: Hard
Coding Mentor: Walter Bender
Assisting Mentor: Ibiam Chihurumnaya

Name: Soham Sarode

Email: sohamsarode2312@gmail.com

Github Profile: https://github.com/soham2312

First Language: I speak English Proficiently

Location: India(GMT+5:30)

Education: Pre-Final Year for BTech CSE (IIIT Jabalpur)

To convince the project team that I'm a good fit for **adding an AI assistant to the Write Activity**, I would emphasize the following points:

1. **Expertise in NLP and AI:** I possess strong skills in natural language processing (NLP) I would like to take you to the personal project of Codeforces Recommendation System and artificial intelligence (AI), including experience with Hugging Face for the GPT-2 model in transformers.

2. **Technical Proficiency:** I am proficient in using tools like PyTorch, transformers library, and GPT-2 models, as demonstrated in the provided code snippet. I have experience working with large language models and understand their capabilities and limitations.

3. **Chatbot Development:** I have prior experience in developing chatbots using TensorFlow, which has given me insights into creating conversational AI systems.

4. **NLP Project Experience:** I have worked on natural language processing projects before, including developing a recommendation system for Codeforces. This experience has enhanced my understanding of NLP techniques and their practical applications.

5. **Understanding Project Goals:** I have a clear understanding of the project's objectives, particularly in enhancing the writing process with grammar correction and AI-assisted suggestions.

6. **Collaborative Approach:** I value collaboration, open communication, and feedback, and I am committed to working closely with the project team to deliver a high-quality solution aligned with the project's objectives.

By demonstrating my expertise, technical skills, understanding of project goals, and collaboration capabilities, I am confident that I can effectively contribute to adding an AI assistant to the Write Activity in Write Activity for Sugar.

Sugar Labs Contributions

1. Replaced WebL10n with i18next in FoodChain activity

    - Issue: #1378
    - Pull request: #1525
    - **Outdated Localization Framework:** The existing localization framework, WebL10n, in the FoodChain activity, is outdated and no longer actively maintained, potentially causing compatibility issues and lacking modern localization features.
    - **Modernization and Enhancement:** Migrating to i18next, an actively maintained internationalization framework, ensures modern localization capabilities, ongoing support, and improved compatibility with evolving technology standards for the FoodChain activity.

2. Replaced WebL10n with i18next in EbookReader activity

    - Issue: #1378
    - Pull request: #1531
    - **Outdated Localization Framework:** The existing localization framework, WebL10n, in the EbookReader activity is outdated and no longer actively maintained, potentially causing compatibility issues and lacking modern localization features.
    - **Modernization and Enhancement:** Migrating to i18next, an actively maintained internationalization framework, ensures modern localization capabilities, ongoing support, and improved compatibility with evolving technology standards for the EbookReader activity.

    .

3. Replaced WebL10n with i18next in MarkDown activity

    - Issue: #1378
    - Pull request: #1540
    - **Outdated Framework:** The existing localization framework, WebL10n, in the Markdown activity is outdated and no longer maintained, potentially leading to compatibility issues and lacking modern localization features.
    - **Modernization and Support:** Migrating to i18next, an actively maintained internationalization framework, ensures modern localization capabilities, ongoing support, and compatibility with evolving technology standards for the Markdown activity.

Other Organizations Contributions

C2Si Organization

1. Addressing Hardcoded Data and Database Reflection in Backup Email Component
    - Issue: #64
    - The issue with the 'backup email' component arises from hard-coded data and actions not reflecting in the database. A solution is needed to establish a database connection, fetch users excluding the logged-in user, and limit the results to four records.

2. Connecting the Firebase database to the component of 'Backup email' in the user dashboard
    - Issue: #64
    - Pull request: #65
    - The user has added functionality to the 'backup email' component connected from the database in the 'Forms' component and implemented Firebase Database Connection for the Backup Email Component in the User Dashboard.

3. Integrate Redux for a voting feature and Firebase integration
    - Issue: #131
    - Pull request: #132
    - This pull request adds Redux action and reducer for managing the state of the voting feature and integrates Firebase for data storage and retrieval, improving scalability and performance. It also includes functionality to check and handle user votes efficiently.
4. Fixed the behavior of upvote/downvote buttons to prevent negative voting counts or unlimited voting behavior.
    - Issue: #133
    - Pull request: #134
    - This pull request addresses a critical issue where users can cast unlimited upvotes/downvotes and negative votes within the voting system, which compromises data integrity and opens the system to potential abuse. The fix includes validation checks to limit users to a single upvote or downvote per item and ensures that negative votes are not allowed. This enhancement improves the reliability and security of the voting feature, providing a more accurate representation of user feedback.

Stdlib-Js Organization

1. [RFC]: Improve Type Declarations for @stdlib/utils/group-in
    - Issue:#1084
    - Pull request: #1383, #1435, #1448
    - **Enhance Type Declarations:** Improve TypeScript type declarations for @stdlib/utils/group-in to address significant type information loss, currently typed as 'any', when returning results.
2. [RFC]: Add C Implementation for @stdlib/math/base/special/cot
    - Issue:#1663
    - **Implement Cotangent Function:** Add a C implementation for @stdlib/math/base/special/cot to enhance the standard library's functionality.
    - **Function Details:** Create stdlib_base_cot, a double-precision cotangent function that takes a double-precision floating-point number x as input and returns the cotangent of x.

Project Details

What are you making?

Project Description: Adding an AI assistant to the Write Activity in Sugar Activities

Overview:

The Write Activity in Sugar Activities provides a platform for peer editing and collaborative writing. However, it lacks advanced support for grammar correction and AI-assisted writing, which are crucial for improving the writing process. This project aims to enhance the Write Activity by integrating an AI assistant that can provide feedback on written content and suggest improvements.

Objectives:

- **Grammar Correction:** Implement grammar correction functionality using natural language processing techniques to identify and rectify grammatical errors in the written text.
- **AI-Assisted Writing Suggestions:** Develop AI models to analyze the text and provide suggestions for improving writing style, clarity, and coherence.

- **Integration with Write Activity**: Integrate the AI assistant seamlessly into the Write Activity interface, ensuring a user-friendly experience for writers.
- **Workflow Optimization:** Optimize the workflow for grammar correction and writing suggestions to provide real-time feedback and suggestions as users write.
- **Technical Implementation:** Python Development: Utilize Python programming language for implementing natural language processing algorithms and AI models.
- **Sugar Activity Development:** Leverage experience with Sugar activities to integrate the AI assistant into the Write Activity interface.
- **Large Language Models (LLMs) and Chatbots:** Utilize LLMs and chatbot technologies to develop the AI assistant for grammar correction and writing suggestions.
- **Fine-Tuning Models:** Fine-tune pre-trained language models (such as GPT-2) on writing-specific datasets to improve the accuracy and relevance of suggestions.
- **User Experience (UX) Design:** Focus on UX design principles to ensure that the AI assistant seamlessly integrates into the Write Activity, providing a smooth and intuitive user experience.

Code Integration:

The provided code showcases the process of importing a pre-trained GPT-2 language model, fine-tuning it on custom writing data, and using it to generate text based on given prompts. This code can serve as a foundation for integrating the AI assistant's functionality into the Write Activity, specifically for providing AI assistance.

Challenges:

- **Data Privacy and Security:** Ensure data privacy and security measures are in place, especially when processing user-generated content.
- **Model Accuracy:** Continuously improve the accuracy and relevance of AI-assisted suggestions through iterative model training and testing.
- **Real-time Feedback:** Implement real-time feedback mechanisms to provide immediate suggestions as users write, without impacting performance.

Conclusion:

By adding an AI assistant to the Write Activity in Sugar Activities, the project aims to revolutionize the writing experience by offering advanced grammar correction, AI Assisted writing

suggestions, thereby enhancing the overall quality of written content.

How will it impact Sugar Labs?

Integrating an AI assistant into the Write Activity in Sugar Activities will have a significant impact on user experience, particularly for children using Sugar Labs' projects:

- **Enhanced Learning Experience:** Children will have access to advanced grammar correction and AI-assisted writing suggestions, promoting better learning outcomes and improving their writing skills over time.
- **Engaging and Interactive:** The AI assistant adds an interactive element to the Write Activity, making writing more engaging and encouraging children to explore and express their ideas with confidence.
- **Personalized Feedback:** The AI-assistant can provide personalized feedback tailored to each child's writing style and skill level, creating a supportive environment for learning and growth.
- **Empowering Creativity:** By offering suggestions and corrections, the AI assistant empowers children to be more creative in their writing while also learning proper grammar and writing techniques.

- **Accessible Learning Tools:** Integrating modern AI technologies makes learning tools more accessible and intuitive for children, promoting self-directed learning and exploration.
- **Positive Impact on Education:** The project's focus on user experience for children ensures that Sugar Labs' educational tools remain engaging, effective, and impactful in supporting children's learning journeys.

What technologies (programming languages, etc.) will you be using?

The technologies that will be used for this project are as follows:

- **Python:** Python will be used for implementing NLP algorithms, developing AI models, and integrating the AI Assistant within the Sugar activities framework.
- **Natural Language Processing (NLP):** NLP techniques will be employed for grammar correction, text analysis, and generating AI-assisted writing suggestions within the Write Activity.
- **Large Language Models (LLMs):** LLMs such as GPT-2 will be utilized for generating text and providing context-aware suggestions based on NLP analysis.
- **Chatbot Technologies:** Chatbot technologies will enhance user interactions with the AI-assistant, utilizing NLP for understanding user queries and providing relevant responses.
- **Sugar Activities Framework:** Development within the Sugar activities framework will include integrating NLP-based

functionalities for grammar correction, writing feedback, and interactive user experiences.

Plan of Action: Grammar Checker and Fine-Tuning for Text Generation

Goal:

The goal is to implement a grammar checker and fine-tuning mechanism for text generation within the Write Activity in Sugar project.

Requirements:

- Python programming skills
- Knowledge of natural language processing (NLP) techniques
- Familiarity with AI models and fine-tuning processes
- Access to GPT-2 or similar large language models
- Development environment set up for Sugar project

Functionalities:

Grammar Checker: Implement an NLP-based grammar checker to identify and correct grammatical errors in written text.

- I have created a prototype for an NLP-based grammar and spell checker to identify and correct errors in written text.
- Colab Link: Link
- Video Link: Link

Fine-Tuning for Text Generation: Fine-tune a pre-trained AI model (e.g., GPT-2) on custom writing data to improve text generation quality and relevance.

- I have developed a prototype for text generation using the GPT-2 model from Hugging Face, which was further fine-tuned with custom datasets to enhance the quality and relevance of the generated text
- Colab Link: Link
- Video Link: Link

Design Figma File: Design
Prototype Video: Video

User Experience (UX) Improvements for Grammar Checker:

- Seamless Integration: Integrate the grammar checker directly into the Write Activity interface, allowing users to access grammar correction tools without switching between different applications.
- Interactive Feedback: Provide real-time feedback on grammatical errors as users type, with interactive suggestions that users can accept or ignore with ease.

User Experience (UX) Improvements for AI-Assisted Writing:

- Non-Intrusive Suggestions: Display AI-generated writing suggestions in a non-intrusive manner, such as pop-ups or tooltips, to avoid disrupting the writing process.
- Contextual Relevance: Ensure that AI-generated suggestions are contextually relevant and displayed based on the user's current writing context, enhancing the usability of the AI assistant.

Steps to Implement:

Set Up Development Environment:

Install necessary libraries and tools, including Python, NLP libraries (e.g., NLTK, spaCy), transformers library for AI models(Hugging Face), and Sugar development environment.

Data Collection and Preprocessing

Gather Text Samples for Grammar Checking:

- Collect a diverse dataset of text samples containing various grammatical errors such as spelling mistakes, punctuation errors, sentence structure issues, etc.
- Include text from different genres and styles to ensure the grammar checker can handle a wide range of writing styles and contexts.

Fine-Tuning Dataset Collection:

- Gather a dataset of writing samples that reflect the target writing style and content for fine-tuning the AI model.
- Include a mix of sentences, paragraphs, and longer text passages to capture the nuances of the writing style and context.

Data Preprocessing for Grammar Checking and Fine-tuning:

- Tokenization: Tokenize the text data into words, sentences, or tokens for processing.
- Cleaning: Remove any irrelevant or noisy data, such as special characters, HTML tags, or non-textual content.
- Normalization: Normalize the text data by converting it to lowercase, removing accents, and standardizing abbreviations.
- Formatting: Format the data into appropriate input formats for NLP tasks, such as tokenized sequences or structured data for training the grammar checker and fine-tuning the AI model.

Implement Grammar Checker:

- Develop an NLP-based grammar checker using Python and appropriate NLP libraries.
- Train the grammar checker on the collected dataset to learn grammatical rules and error patterns.

Fine-Tuning for Text Generation:

- Select a pre-trained AI model (e.g., GPT-2) for text generation.
- Fine-tune the AI model using the collected dataset to adapt it to writing style and content specifics.

Integration with Write Activity:

- Integrate the grammar checker and fine-tuned AI model into the Write Activity of Sugar project.
- Implement user interface components for accessing grammar checking and text generation functionalities.

Testing and Validation:

- Test the grammar checker and text generation features within the write activity of sugar Activity to ensure functionality and accuracy.
- Validate the results by comparing generated text with expected outputs and evaluating grammar correction accuracy.

Optimization and Deployment:

- Optimize the performance of the grammar checker and AI model for efficient use within write activity.

- Deploy the updated write activity with grammar checking and fine-tuned text generation capabilities.

By following these steps, the plan aims to successfully implement a grammar checker and fine-tuning mechanism for text generation within the write ativity project, enhancing the writing experience for users.

Tools for Measuring Performance

- Accuracy: Measure correctness in grammar checking and text generation.
- F1 Score: Balance precision and recall for performance assessment.
- BLEU Score: Evaluate text generation similarity to human-written samples.
- ROUGE Score: Assess summary quality and overlap with reference text.
- Perplexity: Measure language model uncertainty for text generation.
- Confusion Matrix: Detailed breakdown of correct and incorrect predictions.

Optimizing the ALgorithm/Model

Identify Bottlenecks: Conduct thorough performance analysis to identify bottlenecks such as inefficient layers or

computational-heavy operations that significantly impact model speed and efficiency.

Optimize Hyperparameters: Fine-tune hyperparameters using techniques like grid or random search to achieve optimal model performance while reducing computation time and resource usage.

Some methods I will use for optimization of javaScript

Identify Bottlenecks:

Profiling Tools: Use profiling tools such as TensorFlow Profiler or PyTorch Profiler to analyze the execution time and resource usage of different model components. These tools provide insights into which parts of the model are taking the most time or consuming the most resources, helping you focus on optimizing those areas.

Optimize Hyperparameters:

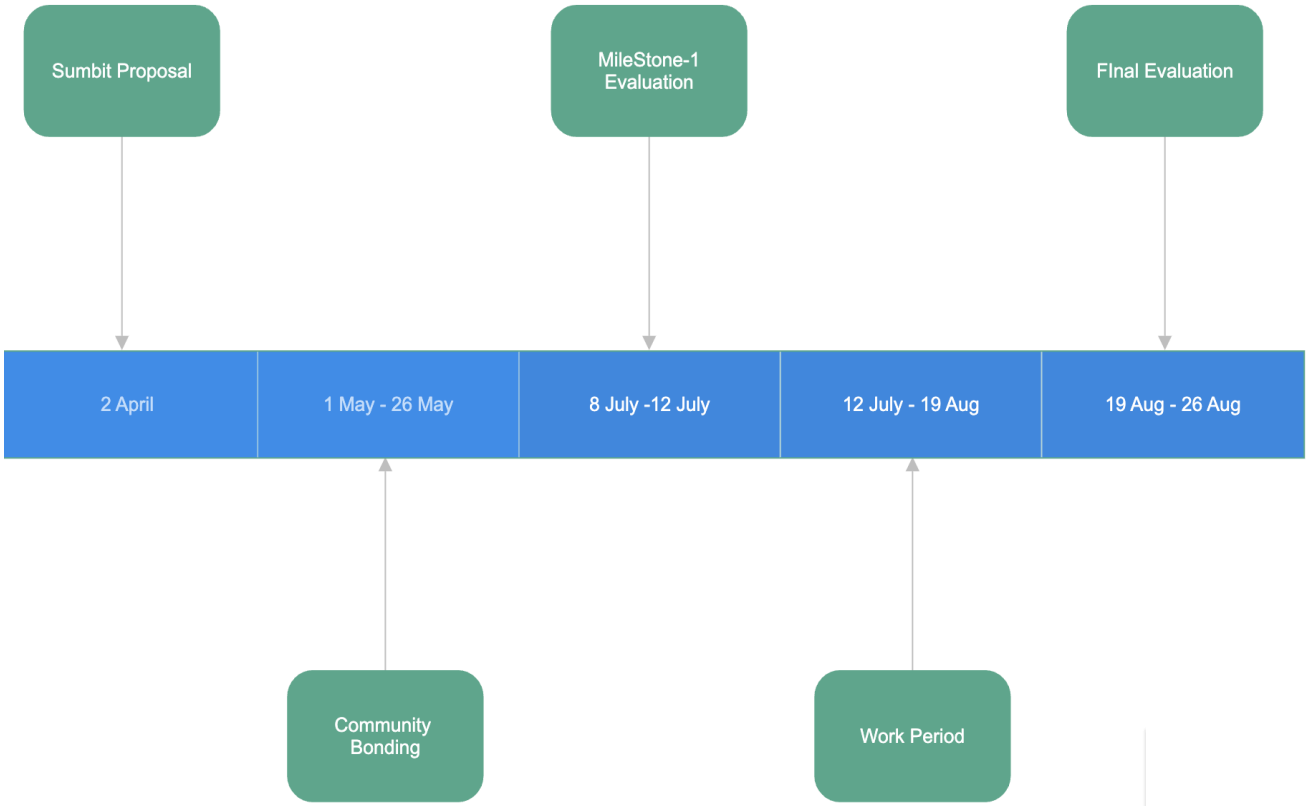
Bayesian Optimization: Use Bayesian optimization techniques such as Gaussian Processes (GP) or Tree-structured Parzen Estimators (TPE) to efficiently search for optimal hyperparameters. Bayesian optimization models the hyperparameter space probabilistically and iteratively explores promising regions, making it efficient for large search spaces and computationally expensive models.

GSoC 2024 has two evaluations, once after every 5 weeks. Highlight the work you plan to complete before each evaluation.

Timeline outlines project events chronologically, while Milestone 1 focuses on foundational tasks like UX changes and grammar checker development. Milestone 2 advances to fine-tuning, optimization, and integration, preparing the project for evaluation and deployment.

TimeLine

Important Dates



After Submission Of Proposal

From April 3 to May 1, the project will enter a phase focused on exploring its intricacies further. This involves delving into the project details, comprehending its nuances, and assembling all the requisite learning resources essential for its seamless development. Additionally, this period will entail referencing various repositories that have previously undertaken the implementation of AI Assistant for writing activity, a crucial step in gathering insights and leveraging existing knowledge to enhance project outcomes.

From May 1 to May 26, the project will focus on community bonding, including engaging with the mentor. This phase involves building rapport within the project community, fostering open communication channels, and collaborating closely with the mentor to discuss and refine project goals and requirements. Discussions with the mentor will be

crucial in gaining valuable insights, receiving guidance, and aligning strategies to ensure a successful and impactful development process.

Milestone 1

UX Changes and Seamless Integration:

- Integrate the grammar checker into the Write Activity interface with seamless access.
- Ensure interactive feedback for real-time error suggestions.

Data Collection and Preprocessing:

- Gather text samples for fine-tuning and grammar checking.
- Preprocess data for NLP tasks and grammar checker training.

Grammar Checker and Fine Tuned Model Implementation:

- Develop and implement the NLP-based grammar checker.
- Test the accuracy and functionality of the grammar checker and fine-tuned model with collected data.

Validation and Documentation Preparation:

- Validate grammar checker results and refine as needed.
- Prepare initial documentation for grammar checker usage.

No.	Description of PR / action	Start Date	Expected Date for PR
1.1	UX Changes and Seamless Integration	22-26 May	4-8 June
1.2	Data Collection and Preprocessing	6-8 June	10-12 June
1.3	Grammar Checker and Fine Tuned Model Implementation	12-14 June	18-20 June
1.4	Validation and Documentation Preparation	20-22 June	24-25 June

1.5	Finalized the implementation and UX integration into the write activity	25-30 June	1-3 July
1.6	Prepare for midterm evaluation	4-8 July	

Milestone 2

Fine-Tuning and Model Optimization (July 1 to July 10):

- Fine-tune AI model for text generation based on collected data.
- Optimize hyperparameters for grammar-checking accuracy.

Testing and Validation (July 11 to July 20):

- Test grammar checker and text generation features for accuracy and reliability.
- Validate results and refine models as necessary.

Integration and Deployment (July 21 to August 10):

- Integrate grammar checker and AI text generation into write activity.
- Optimize performance and deploy features for user testing.

Finalize Documentation(August 11 to August 20):

- Complete comprehensive documentation for grammar checker and AI-assisted writing features.
- Prepare for final evaluation and feedback collection from users and mentors.

No.	Description of PR / action	Start Date	Expected Date of PR

2.1	Fine-Tuning and Model Optimization	12-18 July	18-20 July
2.2	Testing and Validation	20-26 July	26-28 July
2.3	Integration and Deployment	29 Jul- 5 Aug	5-7 Aug
2.4	Review all implementations, discuss changes and new functionalities with your mentor, and update documentation accordingly.	7-15 Aug	15-17 Aug
2.5	Finalize Documentation	17-20 Aug	20-22 Aug
2.6	Prepare for final evaluation	22-26 Aug	

Final Goal of the Project

The final goal of the project is to successfully integrate a user-friendly grammar checker and AI-assisted text generation functionalities into the write activity project, ensuring seamless access, interactive feedback, and accurate suggestions for users. This goal encompasses all aspects of development, testing, optimization, integration, and documentation to deliver a robust and impactful solution for enhancing the writing experience within Sugar

Mention how much time will you spend each week working on your project

I will be working 40 hours a Week

How will you report progress between evaluations?

Google Summer of Code (GSoC), I plan to regularly interact with my mentor on every functionality implemented and provide weekly progress reports to ensure alignment with project goals and receive valuable feedback and guidance.

Discuss your post-GSoC plans. Will you continue contributing to Sugar Labs after GSOC ends?

After the GSoC period, I am eager to continue contributing to the Sugarizer project, particularly focusing on the AI domain. I am passionate about enhancing models, refining responses, and optimizing response time to continually improve the user experience. My goal is to make significant strides in advancing AI capabilities within the SugarLabs Organization, not only benefiting my assigned project but also collaborating on other AI projects to collectively elevate the organization's AI functionalities. I am committed to ongoing learning, innovation, and making meaningful contributions that positively impact the Sugarizer project and its users.