AI assistant for Pippy Activity

Proposal - Google Summer of Code 2024

Mukund Choudhary

Personal Details:

- Full Name: Mukund Choudhary
- Email: 21bcs055@smvdu.ac.in
- GitHub Username: hackorlyf
- Language: English
- Proficiency in programming languages: C, C++, Python & JavaScript
- Location and Timezone: India/ India Standard Time (GMT+5:30)

Introduction:

My name is Mukund Choudhary, and I am thrilled to present my proposal for the Google Summer of Code 2024 with Sugar Labs. I am a passionate software developer hailing from India, currently pursuing my studies in Bachelor of Technology in Computer Science and Engineering at Shri Mata Vaishno Devi University. With a keen interest in open-source development and a strong foundation in computer science, I am excited about the opportunity to contribute to the Sugar Labs community and make a meaningful impact through my proposed project.

Throughout my academic and professional journey, I have actively engaged in software development projects, honing my skills in various programming languages and technologies. My journey with Sugar Labs began with my enthusiasm for leveraging technology to support learning and education, aligning closely with Sugar Labs' mission to provide accessible and empowering tools for learners worldwide.

Beyond technical proficiency, I am deeply committed to fostering collaboration, learning, and community engagement. I believe that the synergy of diverse perspectives and expertise within the open-source community is instrumental in driving innovation and creating impactful solutions. As such, I am eager to collaborate with mentors and fellow contributors to bring the proposed project to fruition and contribute to the continued success of Sugar Labs.

Contributions to Sugarlabs:

- **#3805 (merged):** Completed Documentation in js/planetInterface.js to specify its correct use.
- **#3801 (closed):** Added JS style documentation for jquery-3.7.1.js (not to make changes in library files).
- **#3789 (merged):** Rectified and Completed Documentation for block.js.
- **#3787 (merged):** Added and rectified JS documentation for the event listener function and other functions.
- **#1534 (closed):** Too fast player in Maze Web activity.
- **#3774 (merged):** Added and Updated documentation in block.js.
- **#3768 (merged):** Completed JS Style documentation for base64Utils.js and basicblocks.js.
- **#3764 (merged):** Complete JS style documentation for artwork.js.
- **#9 (closed):** Refactor input handling to prevent duplicate player instantiation.
- **#3754 (merged):** Added JS Style documentation for activity.js.
- **#3746 (merged):** Added JS style documentation for temperament.js.
- **#3741 (merged):** Added JSDoc style documentation for WidgetBlocks.js.
- **#3729 (merged):** Started documentation in WidgetBlocks.js.
- **#1536 (bug):** Found a bug in sugarizer. The player was spawned without face and color from time to time.
- **#3701 (merged):** Removed an extra if condition which was added for the resizing issue when the download toolbar used to appear at the bottom of Chrome. Also added an Exceptional handler in activity.js.
- **#3812 (merged):** Started documentation in protoblocks.js.
- **#3830 (open):** Added and Completed documentation for various functions in protoblocks.js.

Project Proposal:

Project Title:
Add an AI assistant to the Pippy Activity

Overview:
Pippy is the Sugar "learn to program in Python" activity. It comes with lots of examples and has sufficient scaffolding such that a learner could modify an existing Sugar activity or write a new one. This project aims to add "co-pilot"-like assistance to Pippy. A learner should be able to ask the AI to provide example Python code to help them navigate the

language and explore possibilities in a more open way than the collection of Pippy examples affords. (The Pippy examples are geared towards activity development, which is largely how to navigate the basics of the Sugar toolkit and some GTK basics. This would be much broader in scope.)

The challenge, beyond the plumbing, is to design and implement a sensible workflow such that the AI is helpful but not in the way.

Why is Pippy AI Assistance needed:

Expanding the learning possibilities for Pippy users through the integration of an AI assistant stems from the recognition of the diverse learning styles and needs of learners, especially beginners. While Pippy provides a solid foundation for learning Python programming within the context of Sugar Labs, the addition of an AI assistant introduces a dynamic and interactive element that complements traditional learning methods.

The AI assistant serves as a personalized guide for learners, offering contextualized support and guidance tailored to individual learning trajectories. By enabling learners to interact with the AI through natural language queries, they can receive instant feedback, clarification, and code suggestions, fostering a more immersive and engaging learning experience.

Moreover, the AI assistant augments the existing resources within Pippy by providing a broader range of examples and explanations beyond the scope of traditional activity development. This expansion of resources empowers learners to explore Python programming concepts in greater depth and complexity, facilitating a deeper understanding of core concepts and principles.

Additionally, the AI assistant mitigates the risk of learners getting stuck or discouraged when encountering challenges or complexities in their coding journey. By offering timely assistance and suggestions, the AI helps learners overcome obstacles more effectively, promoting a sense of accomplishment and progress.

Overall, the integration of an AI assistant into Pippy enhances the learning possibilities by providing personalized support, expanding resources, and mitigating barriers to learning, thereby empowering learners to grasp concepts faster and navigate the Python programming landscape with confidence and ease.

Architectural Flow:

1. The user interacts with the Pippy Activity interface.
2. The user requests assistance from the AI assistant through a designated UI component or natural language input.
3. Pippy Activity sends the user's request to the AI assistant module.
4. The AI assistant module processes the user's request using natural language processing (NLP) techniques.
5. The AI assistant module retrieves relevant Python code examples, explanations, or suggestions based on the user's request and the context of their current activity.
6. The AI assistant module sends the generated response back to the Pippy Activity.
7. Pippy Activity displays the AI assistant's response to the user, either as text or within the Pippy interface.
8. User interacts with the provided assistance, potentially incorporating it into their coding activities or asking follow-up questions.
9. The cycle repeats as the user continues to interact with the Pippy Activity and requests assistance from the AI assistant as needed.

Downsides to Using a Co-pilot-like Interface:

While an AI co-pilot interface can offer valuable assistance to learners, there are potential downsides that should be considered:

- Dependency on AI: Relying too heavily on an AI co-pilot may hinder learners from developing critical thinking and problem-solving skills. Learners may become dependent on AI for generating code solutions rather than learning to solve problems independently.
- Overreliance on Suggestions: Learners may be tempted to blindly accept code suggestions from the AI without fully understanding the underlying concepts. This could lead to superficial learning and a lack of deeper comprehension of programming principles.
- Limitation of Creativity: A co-pilot-like interface may limit learners' creativity and exploration. Instead of experimenting with different approaches and solutions, learners may stick to the suggestions provided by the AI, thereby stifling innovation and originality.
- Privacy and Security Concerns: Introducing an AI co-pilot raises potential privacy and security concerns, particularly if the assistant interacts with sensitive data or requires access to user information. Ensuring the privacy and security of user data should be a priority when implementing such a feature.

Language Model Recommendation and Cost Consideration:

In selecting the Language Model (LLM) for the AI assistant in Pippy, it's essential to prioritize open models that align with Sugar Labs' ethos as a FOSS (Free and Open Source Software) organization. Considering the financial constraints of our target users, especially children without a budget for AI, cost-effective solutions are paramount. Therefore, I recommend exploring open-source LLMs that offer robust capabilities without incurring significant expenses.

Recommended Open Models:

1. OpenAI's GPT (Generative Pre-trained Transformer) series:
    - **GPT-2 and GPT-3**: While the full versions of GPT-2 and GPT-3 are proprietary, OpenAI has released smaller, open-source versions that can still provide valuable language generation capabilities.
    - **Potential Cost**: Since OpenAI offers smaller versions of GPT-2 and GPT-3 for free, utilizing these versions could minimize costs associated with AI implementation.

2. Hugging Face's Transformers Library:
    - This library provides access to various pre-trained models, including GPT variants, BERT, and more, fostering flexibility and customization in model selection.
    - **Potential Cost**: Hugging Face's library is open-source and free to use, aligning with the FOSS principles of Sugar Labs.

3. Google's BERT (Bidirectional Encoder Representations from Transformers):
    - Although developed by Google, BERT has been widely adopted in the open-source community and can serve as a powerful tool for natural language understanding tasks.
    - **Potential Cost**: While Google's pre-trained BERT models are freely available, there might be costs associated with utilizing Google Cloud services for training or fine-tuning custom models. However, pre-trained models can be leveraged without incurring additional expenses.

By leveraging these open models, we can ensure accessibility and affordability while providing a high-quality AI assistant experience within Pippy. Additionally, the utilization of open-source frameworks and tools will enable seamless integration with Sugar Labs' existing infrastructure, fostering collaboration and innovation within the community.

Timeline:

Week 1-2: Exploration and Planning

- Dive deep into the existing codebase of the Pippy Activity, understanding its architecture, and how components interact.
- Initiate discussions with mentors to define specific objectives and milestones for the project.
- Outline a detailed plan for integrating the AI assistant into the Pippy Activity, considering both technical requirements and user experience enhancements.
- Set up the development environment, ensuring all necessary tools and dependencies are in place.

Week 3-4: Initial Implementation

- Begin implementing basic functionalities of the AI assistant within the Pippy Activity framework.
- Design a user-friendly interface for interacting with the AI assistant, ensuring seamless integration with the existing Pippy interface.
- Conduct initial rounds of testing to identify and address any technical challenges or compatibility issues.
- Regularly communicate progress with mentors, seeking feedback and guidance as needed.

Week 5-6: Functional Enhancement

- Enhance the capabilities of the AI assistant by integrating natural language processing (NLP) capabilities, enabling it to interpret and respond to user queries more effectively.
- Implement features such as code suggestion and error correction to assist learners in navigating Python programming concepts.
- Conduct usability testing sessions with target users to gather feedback on the AI assistant's functionality and user interface.
- Refine the design and functionality based on user feedback, prioritizing improvements that enhance the learning experience.

Week 7-8: Integration and Optimization

- Integrate the AI assistant seamlessly into the Pippy Activity workflow, ensuring it complements existing features without causing disruption.
- Optimize the performance of the AI assistant, addressing any bottlenecks or inefficiencies in its implementation.
- Conduct comprehensive testing across different environments and platforms to ensure compatibility and reliability.
- Document the integration process and provide clear instructions for users on how to utilize the AI assistant within Pippy.

Week 9-10: Advanced Features and Testing

- Implement advanced features of the AI assistant, such as providing relevant Python code examples based on user queries.
- Conduct thorough testing and debugging to identify and resolve any remaining issues or bugs.
- Perform stress testing to assess the scalability and robustness of the AI assistant under various conditions.
- Prepare for the first evaluation by summarizing progress, achievements, and challenges encountered during the first phase of the project.

Week 11-12: Finalization and Documentation

- Finalize all components of the AI assistant, ensuring it meets all specified requirements and objectives.
- Create comprehensive documentation and tutorials for users, covering installation, usage, and troubleshooting of the AI assistant within Pippy.
- Prepare a detailed report highlighting key accomplishments, technical insights, and lessons learned during the development process.
- Conduct a review of the project with mentors, soliciting feedback and addressing any remaining issues or concerns.

Week 13-14: Review and Refinement

- Review the project's progress and performance against initial goals and milestones.
- Address any feedback or suggestions provided by mentors and community members, making necessary refinements or adjustments.
- Conduct final rounds of testing to ensure the stability and reliability of the AI assistant in real-world usage scenarios.

- Prepare for the second evaluation by documenting progress and achievements made during the second phase of the project.

Week 15-16: Finalization and Post-GSOC Planning

- Finalize all deliverables and documentation required for the final evaluation.
- Conduct a comprehensive review of the project to ensure completeness and readiness for submission.
- Discuss post-GSOC plans with mentors and community members, outlining intentions for continued contribution and support to Sugar Labs.
- Prepare a roadmap for future development and improvement of the AI assistant, identifying potential areas for further enhancement and collaboration.

Hours per Week: I dedicate approximately 25-28 hours per week to this project.

Progress Reporting:

I will maintain regular communication with my mentors through weekly progress reports, detailing the tasks completed, challenges faced, and plans for the upcoming week. Additionally, I will utilize project management tools such as GitHub to track progress and share code updates with my mentors for feedback.

Post GSoC Plans:

After GSOC ends, I am committed to continuing my contributions to Sugar Labs. I will actively engage with the community, address any ongoing issues or feature requests related to the AI assistant in Pippy, and explore opportunities for further enhancements and collaborations within Sugar Labs.