Basic Details

- **Full Name**: Archit Rakeshkumar Agrawal
- **Email**: architagrawal000@gmail.com
- **GitHub Username:** https://github.com/architagrawal
- **Matrix chatroom**: @archit0000:matrix.org
- **Your first language**: Hindi, Gujarati. I prefer English for communication with community
- **Location and Time zone**: Tempe, Arizona (MST, GMT-7)
- **Share links, if any, of your previous work on open-source projects:** I am new to the open-source community. I take GSOC as an opportunity to get started contributing my skills for open-source projects.
- **Convince us that you will be a good fit for this project, by sharing links to your contribution to Sugar Labs.** https://github.com/sugarlabs/sugar/pull/983
- **My Brief Introduction**:
    - I am currently pursuing my studies in computer science at Arizona State University, where I've delved deep into AI topics through coursework such as Data Mining, focusing on various data analysis techniques, and Data Intensive Systems for Machine Learning, which provided insights into managing ML systems for Big Data applications. I've closely followed the progress of Large Language Models on platforms like Hugging Face.
    - In my role as a Student Software Developer at ASU, I'm developing a corrective retrieval-augmented generation (RAG) based chatbot system. This system empowers faculty members to design courses effectively using technologies like Python, Langchain, OpenAI, Prompt Flow, and Semantic Kernel. Additionally, I've designed and implemented APIs and webpages for quiz platforms and question banks within the ASU Online courses platform, significantly enhancing the educational experience for students. My hands-on experience and dedication to staying abreast of emerging technologies position me well to contribute meaningfully to projects at the intersection of AI and education.

Project Details

- **What are you making?**
    - InfoSlicer serves as an add-on activity for Sugar desktops, offering teachers a seamless way to create new documents by simply dragging and dropping content from downloaded Wikipedia articles and subsequently publishing them.
    - This summer, my aim is to enhance InfoSlicer by integrating new features that enable the generation of summaries for both original and edited Wikipedia articles. Additionally, I plan to implement functionality that allows teachers to automatically generate lesson plans based on selected themes and the information extracted from Wikipedia articles. A crucial aspect of these new features will be the implementation of a validation layer to ensure the accuracy and reliability of the generated summaries and lesson plans.
- **How will it impact Sugar Labs?**
    - The new features will ease the process for users to create new articles from existing articles. This promotes collaboration and knowledge-sharing within the community.

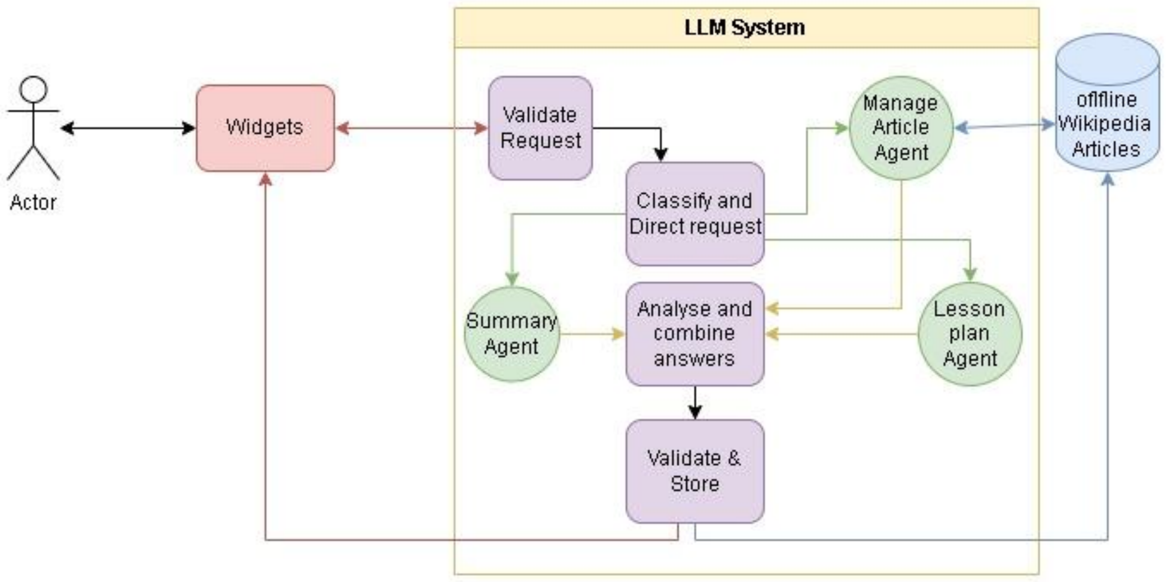
- The integration of the lesson plan feature will aid teachers to create custom lesson plans based on relevant Wikipedia content, enhancing the educational experience for teachers and students. This aligns with Sugar Labs' mission of promoting innovative and accessible educational tools worldwide.

- **What technologies (programming languages, etc.) will you be using ?**
  - **UI:** GTK3. As the UI for sugar is based on it.
  - **Vector DB for embeddings**: FAISS, Chroma
  - **LLM**: Ollama, Langgraph and Langchain, semantic-kernel and guidance
  - **Validation and Guardrail the output**: Deepeval and guardrail-ai
  - **Monitor the flow during development**: Prompt Flow

- **Break down the entire projects into chunks and tell us what will you work on each week.**

  - **Deliverables:**
    - Interactive widgets facilitating user interaction with the Large Language Model (LLM) chatbot.
    - Development of a Free and Open Source (FOS) LLM integrated chatbot system.
    - Comprehensive documentation providing a user guide for navigating and utilizing the chatbot system effectively. This documentation will include instructions on installation, usage, and troubleshooting to ensure seamless integration and user satisfaction.

  - **System Design:**



    - Web Interface: Add two tabs on top using gtk3 and an interface for user and chatbot interaction.
    - Agents responsible for each task implemented on langchain and langGraph. I will also explore semantic kernels and microsoft guidance to improve the performance in later stages.
    - A central LLM system that handles the flow of answer generation. The steps include validating and checking the user request for prompt injection, classifying the request and getting an answer from relevant

    agents, analyzing and aggregating their responses to form an answer, finally validating the answer using DeepEval and guardrailing for improper output.
    - I plan to test some pre-trained LLM available locally through Ollama at each stage of development and compare overall results.
    - For privacy concerns, I plan to use FAISS/Chroma as vector databases as they will be initialized for each session. One drawback is for each source article, the system will generate embeddings each session
    - For privacy concerns, I will not be using any databases. No interaction will be stored between sessions. Drawback of this is, there will be no possibility of persistent chat and chat history won't be available for users.

  - **Implementation**
    - Design widgets and an UI for interaction in:
      - Summary tab
      - Lesson plan tab
    - Agents responsible for each process:
      - Manage articles
      - Generate summary
      - Create Lesson plan
    - Develop prompts for LLM to:
      - Validate the request for prompt injection and required data,
      - Classify and direct the request,
      - Summarize documents
      - Edit summary
      - Create lesson plan
      - Edit lesson plan
      - Aggregate the response from agents
    - Validate with deepeval and guardrail the answer with guardrail-ai:
      - Define test cases for sample interaction in deepeval
      - Define a specific format of answers to be avoided with guardrail ai
      - Utilize LLM itself to validate the generated answers

  - **Timeline**
    - Week 1-2: Implement widgets and basic UI connecting with the LLM system.
    - Week 3-4: Load LLM models. Implement request validation and logic to classify requests.
    - Week 5-6: Develop prompts for each type of request. Compare and choose LLM with optimal results.
    - Week 7-8: Implement agents
    - Week 9-10: Aggregate responses from agents, validate and guardrail the output, and connect the chain of thought.
    - Week 11-12: Test the chatbot, modify and integrate the system, document setup and usage for users.

- **If you will be off-the-grid for a few days, then mention those in the timeline:** I have no other commitments during the GSOC timeline.
- **GSoC 2024 has two evaluations, once after every 5 weeks. Highlight the work you plan to complete before each evaluation.** For the first evaluation, I plan to have UI and a basic chatbot flow ready. Till second evaluation the chatbot system will be ready and only the final 2 weeks task will be left. Kindly refer to time line.

- **How many hours will you spend each week on your project?** 40 hours per week
- **How will you report progress between evaluations?** I will share weekly updates of my progress on tasks of the week and tasks for the next week through mailing lists. I will keep in touch with project mentors through matrix and update on each mini-milestone achieved.
- **Discuss your post GSoC plans. Will you continue contributing to Sugar Labs after GSOC ends?** I am interested in working in the edTech space and activities developed on sugar. Post GSoc, I will be resuming my school. I may find it difficult to dedicate the same time that I will be in summer, but plan to dedicate at least 10 hours per week to develop new and interesting functionalities or maintain the projects.
    - I would like to look into possibilities to add the activities to sugarizer as it took me a lot of tries to set up the system locally. Having access on the fly saves hassle is what I think.
    - I see multiple projects that are based on LLM for this GSOC. I am sure different approaches will be used by contributors, each having a different LLM system. These systems require a bulky LLM package to be downloaded for each activity and have a greater system resource usage. Having multiple LLM for different activities is not feasible for sugar is what I think. I would like to connect all these activities to a central LLM system post GSOC.