

# Google Summer of Code 2024

## AI Chatbot for Chat Activity

Sachin Singhal

### Section 1: About You

My name is [Sachin Singhal](#). I am a second-year undergraduate student at the [Swami Keshvanand Institute of Technology Jaipur](#) from [India](#), pursuing Computer Science Engineering as my major.

**What project are you applying for?**

["Integrate a chatbot tailored to engage with our younger-skewing learners."](#)

**Why are you interested in working with [Sugarlabs](#), and on your chosen project?**

I began contributing to Sugar Labs in October 2023 with the goal of exploring the open-source community and participating in [GSOC 2024](#). Contributing to Sugar Labs has helped me learn [Python, LLMs/Chatbots, and Sugar activities](#), introducing me to the test-driven environment. Whenever I encountered challenges, mentors were readily available for assistance. Through my involvement with the [Chat Activity](#) team, I gained valuable insights into [Python, model training, datasets, and real-world applications](#). Now, I am proficient with the source code and have been working on the [Chat Activity](#) since day one, gaining a deep understanding of potential issues and how to resolve them. Completing this project would be an invaluable learning experience for me, and I aim to streamline the integration process for improved developer-friendliness.

**Prior Experience**

For the past one and a half years, I've been immersed in **Machine Learning and Artificial Intelligence**, primarily focusing on training chatbots using Python. Over the last 5-6 months, I've dedicated my efforts to contributing to Sugar Labs, specifically within the [ChatActivity](#) team.

1. My contributions involve creating pull requests, assisting fellow team members, identifying and documenting bugs within the project, and actively resolving them.
2. I also provide support to other contributors, successfully addressing one query and making efforts to assist with another.
3. Additionally, I developed a chatbot for [The Ministry of Consumer Affairs, Government of India](#) during a hackathon.
4. Furthermore, I created a Finance Bank Chatbot using Python for [MasterCard](#).
5. I've developed several models using **datasets** to implement in applications, incorporating **real-world dynamics**.

### **Links to PRs.**

- [#46](#) Activate a Storytelling mode.
- [#45](#) Hugging face deployment.
- [#39](#) Enhancing more features.
- [#44](#) Resolved Issue
- [#43](#) Resolved Issue

A complete list of my issues can be found [here](#).

I did volunteer work with mentor of ChatActivity @Ibiam Chihurumnaya and @Walter Bender. My PRs can be found [here](#).

### **Project Size**

I am applying for a large project (~175 hours).

### **Project Timeframe**

13 June 2024 to 22 September 2024

### Contact info and timezone(s)

Primary Email and Hangout: [singhal3.sachin7@gmail.com](mailto:singhal3.sachin7@gmail.com)

Secondary Email: [b220499@skit.ac.in](mailto:b220499@skit.ac.in)

Contact Number: +91 7852099010

Linkedin: [in/melosachin](https://www.linkedin.com/in/melosachin)

GitHub: [@12sachingupta](https://github.com/@12sachingupta)

Personal Portfolio: [Link](#)

Resume:  **Singhal.Sachin.Software.Exp(1).pdf**

Company Website: [visit](#)

Discord: [sachinhasaksita](#)

Time Zone: Kolkata, India (GMT+5:30)

Preferred mode of Communication: Hangout, Email, Google meet, Discord.

### Time Commitment

I am having summer vacation from 1 June to 16th July. In this time frame, I would be able to devote **~42 hr/week**. After that, I would be able to devote **~25 hr/week**. which may increase if the need arises. ( **Note:** can be extended if the need arises).

### Essential Prerequisites

- I am able to run a **Python backend test** target on my machine.

```
legion@pop-os: ~/Desktop/OpenSource/oppia

Tasks still running:
  core.controllers.editor_test (started 01:28:58)
-----
[datastore] Mar 14, 2022 1:29:01 AM io.gapi.emulators.grpc.GrpcServer$3 operationComplete
[datastore] INFO: Adding handler(s) to newly registered Channel.
[datastore] Mar 14, 2022 1:29:01 AM io.gapi.emulators.netty.HttpVersionRoutingHandler channelRead
[datastore] INFO: Detected HTTP/2 connection.
20:00:03 FINISHED core.controllers.editor_test: 65.0 secs
Stopping Redis Server(name="sh", pid=382839)...
Stopping Cloud Datastore Emulator(name="sh", pid=382720)...

+-----+
| SUMMARY OF TESTS |
+-----+

SUCCESS   core.controllers.editor_test: 92 tests (61.6 secs)

Ran 92 tests in 1 test class.
All tests passed.

Done!
(P.3.7) legion@pop-os:~/Desktop/OpenSource/oppia$
```

- I am able to run all the **trained model tests** at once on my machine.

## Other Summer Obligations

Currently I do not have any summer obligations.

## Communication Channels

I am active on Emails, Hangouts, and Discord. I can work with whatever platform that my mentor prefers. Meetings can be held every week to discuss progress in the project.

## Section 2: Proposal Details

### ***Problem Statement***

Link to PRD (or N/A if there isn't one)	<a href="#">Add an AI chatbot to the chatbot activity</a>
Target Audience	<ul style="list-style-type: none"><li>Younger Learners Various Age group</li></ul>
Core User Need	<ul style="list-style-type: none"><li>The core user need for this project is to enhance the interactive chat program by integrating a chatbot tailored to <b>engage younger learners, requiring proficiency in Python, Sugar activities, and LLMs/Chatbots.</b></li><li>Users seek a chatbot that seamlessly integrates into conversations, catering to the <b>unique age demographic of learners, necessitating tuning to align with their preferences and understanding levels.</b></li></ul>
What goals do we want the solution to achieve ?	<ul style="list-style-type: none"><li>Enhance the interactive nature of the Chat Activity by seamlessly integrating a <b>chatbot into conversations.</b></li><li>Cater to the needs of younger learners by tuning the chatbot to align with their age group, thereby promoting engagement and effective learning experiences.</li></ul>

## Section 3: Project Development

### AI chatbot for chat activity

#### Specifications:

- **Natural Language Processing:** AI chatbot capable of understanding and generating human-like responses.
- **Topic Versatility:** Ability to discuss academic subjects and personal experiences.
- **Privacy Protection:** Ensures confidentiality and privacy when discussing personal matters.
- **Context Awareness:** Retains context from previous messages to maintain coherent conversations.
- **User Engagement:** Designed to keep users engaged through interactive and meaningful interactions.

#### Introduction:

Welcome to our AI chatbot, designed for seamless communication and interaction! Whether you're discussing academic topics or sharing personal anecdotes, our chatbot is here to facilitate enriching conversations. With a focus on privacy and context retention, rest assured that your interactions are secure and personalized. Let's start chatting and exploring the endless possibilities together!

## Project Goals

During this GSOC term, I would deliver the following:

- **Integration:** Integrating an AI chatbot into the existing Chat Activity program involves incorporating the chatbot's functionality seamlessly within the interface, allowing users to interact with both human participants and the AI bot within the same conversation. This integration requires implementing communication protocols and user interface elements to enable smooth interaction between the chatbot and users.
- **Customize:** Customizing the chatbot for a younger audience involves adapting its language and responses to be age-appropriate, using simpler vocabulary and more relatable examples. Additionally, the chatbot's content and topics should align with the educational interests and developmental level of the target audience, ensuring engagement and relevance for younger users.
- **Engaging and Interactive:** To ensure the chatbot is engaging and interactive, it should incorporate features such as quizzes, games, and storytelling elements to captivate users' attention. Additionally, allowing for user input and feedback, as well as dynamically adjusting responses based on user interaction, can enhance engagement and encourage continued participation in the conversation.
- **Tuning:** Fine-tuning the chatbot's responses involves aligning them with the educational objectives and subjects discussed in the Chat Activity, ensuring relevance and coherence with the learning material. Additionally, providing comprehensive documentation and guidance for maintaining and updating the chatbot integration ensures its longevity and enables future improvements to meet evolving educational needs and goals.

#### Some More Proposed Features:

- **Personalization Options:** Allow users to customize their chatbot experience by choosing from different themes, avatars, or personalities, enhancing engagement and making the interaction more enjoyable.
- **Learning Progress Tracking:** Implement a feature that tracks users' progress

and knowledge acquisition through the chatbot conversations, providing feedback and suggestions for further learning opportunities.

- **Multilingual Support:** Incorporate multilingual capabilities into the chatbot, enabling users to interact in different languages and promoting inclusivity and accessibility for diverse user populations.
- **Integration with Educational Resources:** Provide seamless integration with educational resources such as textbooks, videos, or interactive simulations, enhancing the chatbot's capabilities as a supplementary learning tool.

## Implementation

- **Integrating AI Chatbot into Chat Activity:** Utilize Python libraries such as NLTK or TensorFlow to integrate an AI chatbot into the Chat Activity interface, enabling seamless interaction between users and the bot.

```
# Import the necessary libraries for integrating the chatbot
import chatbot_module

# Instantiate the chatbot within the Chat Activity interface
chatbot = chatbot_module.Chatbot()

# Add functionality to the Chat Activity interface to interact with the chatbot
def handle_chat_message(message):
    if message.startswith('/chatbot'):
        response = chatbot.respond(message)
        display_response(response)
    else:
        display_message(message)

# Code to display messages and responses within the Chat Activity interface
```

- **Customizing for Younger Audience:** Modify the chatbot's responses and vocabulary using age-appropriate language and examples to cater to the younger target audience.



```
# Define a function to customize chatbot responses for younger audience
def customize_response_for_younger_users(response):
    # Add logic to simplify language and use relatable examples
    return simplified_response

# Implement the customization function within the chatbot module
chatbot.set_customization(customize_response_for_younger_users)
```

- **Implementing Engagement Features:** Add interactive features such as quizzes and games within the chatbot interface to enhance user engagement.

```
# Implement interactive features such as quizzes and games within the chatbot module
def handle_user_input(user_input):
    if user_input.startswith('/quiz'):
        quiz_question = get_random_quiz_question()
        display_quiz_question(quiz_question)
    elif user_input.startswith('/game'):
        game_instructions = get_game_instructions()
        display_game_instructions(game_instructions)

# Update the chatbot to handle user input and trigger interactive features
chatbot.set_interaction_handler(handle_user_input)
```

- **Fine-tuning Educational Alignment:** Adjust the chatbot's responses to align with the educational goals and topics covered in the Chat Activity, ensuring relevance and coherence.

```
# Define a function to fine-tune chatbot responses for educational goals
def fine_tune_response_for_educational_topics(response):
    # Add logic to align responses with educational objectives and topics
    return fine_tuned_response

# Implement the fine-tuning function within the chatbot module
chatbot.set_fine_tuning(fine_tune_response_for_educational_topics)
```

- **Documentation for Maintenance:** Provide comprehensive documentation and guidelines for maintaining and updating the chatbot integration, ensuring its longevity and usability.

```
# Create documentation outlining the chatbot's features, customization options, and
documentation = """
AI Chatbot Documentation:
- Features: List of chatbot features and capabilities
- Customization: Instructions for customizing chatbot responses
- Maintenance: Guidelines for maintaining and updating the chatbot integration
"""

# Save the documentation to a file for future reference
with open('chatbot_documentation.txt', 'w') as file:
    file.write(documentation)
```

- **Multilingual Support:** Implement translation capabilities within the chatbot to support multiple languages, enhancing accessibility for diverse user populations.

```

from googletrans import Translator

def translate_message(message, target_language='en'):
    translator = Translator()
    translated_message = translator.translate(message, dest=target_language).text
    return translated_message

# Usage:
translated_response = translate_message("Hello, how are you?", target_language='fr')

```

- **Parental Controls:** Develop a feature allowing parents to set time limits or restrict certain topics within the chatbot conversation, ensuring age-appropriate content for their children.

```

def parental_controls(message):
    if "math" in message:
        if is_parental_control_enabled() and not is_math_topic_allowed():
            return "Sorry, this topic is restricted by parental controls."
    return None

# Usage:
restricted_response = parental_controls("Let's learn advanced calculus!")

```

- **Collaborative Learning Activities:** Enable collaborative learning sessions where users can join group discussions or solve problems together within the chatbot interface.

```

def start_group_discussion(topic):
    # Code to initiate a group discussion on the specified topic
    pass

# Usage:
start_group_discussion("Science exploration")

```

- **Feedback Mechanism:** Integrate a feedback system allowing users to rate chatbot responses and provide suggestions for improvement, enhancing user

experience

```
def collect_feedback(response, rating):  
    # Code to collect and process user feedback  
    pass  
  
# Usage:  
collect_feedback("Great explanation!", rating=5)
```

- **Gamification Elements:** Incorporate gamification elements such as badges and rewards to incentivize user engagement and motivate participation in chatbot interactions.

```
def award_badge(user_id, badge_type):  
    # Code to award a badge to the user  
    pass  
  
# Usage:  
award_badge("user123", "Super Learner")
```

- **Augmented Reality Integration:** Implement AR functionality within the chatbot interface, allowing users to visualize concepts or interact with virtual objects in a 3D environment.

```
def show_ar_content(topic):  
    # Code to display augmented reality content related to the specified topic  
    pass  
  
# Usage:  
show_ar_content("Space exploration")
```

- **Storytelling Mode:** Activate a storytelling mode where the chatbot narrates educational stories or adventures, fostering creativity and imagination in users.

```
def start_storytelling_mode():  
    # Code to initiate storytelling mode  
    pass  
  
# Usage:  
start_storytelling_mode()
```

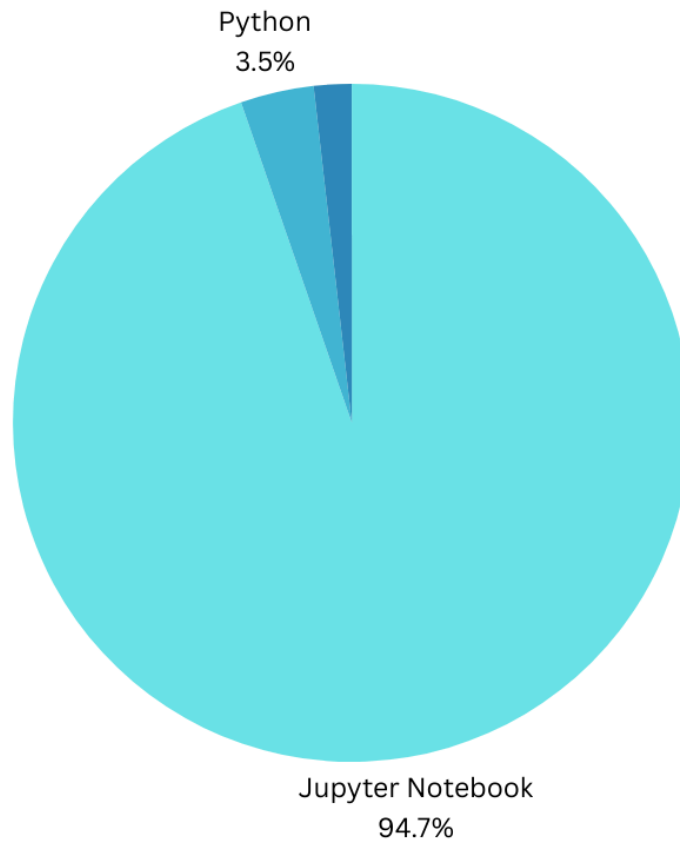
- **Emotional Intelligence:** Incorporate sentiment analysis capabilities to detect and respond to users' emotions, allowing the chatbot to provide empathetic and supportive interactions.

```
from textblob import TextBlob  
  
def analyze_sentiment(message):  
    blob = TextBlob(message)  
    sentiment_score = blob.sentiment.polarity  
    return sentiment_score  
  
# Usage:  
sentiment_score = analyze_sentiment("I'm feeling excited about learning!")
```

## My Demo Chatbot for Chat Activity

*It is Discord AI chatbot for char Activity*

### Tools and Technologies:



	Accuracy	Cross Platform	User Database
Firefox	98.789	Yes	50
Discord Server	99.895	Yes	25
Application	89.173	Not Applicable	35

**Hugging Face Deployment:** Deploy the chatbot model on the Hugging Face platform for easy access and scalability, allowing for future predictions and replication.

```
from transformers import pipeline

# Load chatbot model from Hugging Face
chatbot = pipeline("conversational")

# Usage:
H response = chatbot("Hello, how are you?")
```

### ⚡ Hosted inference API ⓘ

💬 Conversational

Input a message to start chatting with  
**r3dhummingbird/DialoGPT-medium-joshua.**

Send

This model is currently loaded and running on the Inference API.


</> JSON Output

🖥 Maximize



# Welcome to #bot!

This is the start of the #bot channel.

 [Edit Channel](#)



Message #bot



GIF




ONLINE — 2



JoshuaBot BOT



lynn 



```

from transformers import AutoTokenizer, AutoModelWithLMHead

tokenizer = AutoTokenizer.from_pretrained("r3dhummingbird/DialoGPT-medium-joshua")

model = AutoModelWithLMHead.from_pretrained("r3dhummingbird/DialoGPT-medium-joshua")

# Let's chat for 4 lines
for step in range(4):
    # encode the new user input, add the eos_token and return a tensor in Pytorch
    new_user_input_ids = tokenizer.encode(input(">> User:") + tokenizer.eos_token_id)
    # print(new_user_input_ids)

    # append the new user input tokens to the chat history
    bot_input_ids = torch.cat([chat_history_ids, new_user_input_ids], dim=-1)

    # generated a response while limiting the total chat history to 1000 tokens,
    chat_history_ids = model.generate(
        bot_input_ids, max_length=200,
        pad_token_id=tokenizer.eos_token_id,
        no_repeat_ngram_size=3,
        do_sample=True,
        top_k=100,
        top_p=0.7,
        temperature=0.8
    )

    # pretty print last output tokens from bot
    print("JoshuaBot: {}".format(tokenizer.decode(chat_history_ids[:, bot_input_ids.size()-1:])))

```

## Structure of this Project

- **model\_train\_upload\_workflow.ipynb:** Notebook to be run in Google Colab to train and upload the model to Hugging Face's Model Hub.
- **discord\_bot.py:** Script to be imported into a Repl.it Python Discord.py project.

- discord\_bot.js: Script to be imported into a Repl.it JavaScript Discord.js project.

**This is a Discord AI Chatbot that uses the Microsoft DialoGPT conversational model.**

## Timeline/Project Plan:

GSoC is round about 12 week duration, with about 25 days of Community Bonding Period in Addition.

I will be spending **10% time in fixing the bugs** left out in the current version of the app. **80% time on adding new features** to the Bot.

Remaining **10% time on testing the bot , preparing Wiki and FAQ for the template**. The detailed timeline is linked below:

**Legend:** Importance and Time Devoted:  >  >  > 

Time Frame	Start Date	End Date	Task	
Community Bonding	7 <sup>th</sup> May	26 <sup>th</sup> May		
	27 <sup>th</sup> May	29 <sup>th</sup> May	Requirement Gathering and Documentation	
	29 <sup>th</sup> May	1 <sup>st</sup> June	UI enhancement of the Chatbot	
	1 <sup>st</sup> June	5 <sup>th</sup> June	Implementing Edit any instance functionality	
	5 <sup>th</sup> June	10 <sup>th</sup> June	Implementing Reset Password Functionality	
	10 <sup>th</sup> June	20 <sup>th</sup> June	Implementing QR code functionality along with Send emails on getting an answer on instance functionality	

	20 <sup>th</sup> June	25 <sup>th</sup> June	UI enhancement of the bot including fonts update and an initial testing of the application	
Phase 1 Evaluation	25 <sup>th</sup> June	28 <sup>th</sup> June		
	26 <sup>th</sup> June	12 <sup>th</sup> July	Implementing archive an instance session functionality along with initial documentation for the application.	
	12 <sup>th</sup> July	24 <sup>th</sup> July	Implementing Partial implementation of Real time remote control functionality including some UI improvements.	
Phase 2 Evaluation	24 <sup>th</sup> July	26 <sup>th</sup> July		
	25 <sup>th</sup> July	30 <sup>th</sup> July	UI enhancement of the bot including Python implementation for Model Training )	

	30 <sup>th</sup> July	7 <sup>th</sup> August	Remaining implementation of the bot..	
	7 <sup>th</sup> August	11 <sup>th</sup> August	Implementing Previous notification feature along with UI enhancement of the bot including necessary loaders implementation	

	11 <sup>th</sup> August	17 <sup>th</sup> August	Omniauth Implementation And Follow any instance implementation Along with Thorough Testing, Exception Handling and distributing the bot among beta testers (if allowed).	
	17 <sup>th</sup> August	19 <sup>th</sup> August	Preparing Documentations, Wiki and FAQs and a Webcast on the Final Product.	

I can assure you that if I get selected to work with Sugarlabs this summer, I definitely will try my level best to make this project successful and will love to continue working with other Sugarlabs's projects even after the summer.

I would love to mentor in any Educational programs that Sugarlabs is a part of. Further, I would really love to work under such great mentors of Sugarlabs.

Also for some reasons, if I am not selected this year even then I'll try to contribute to this and other projects as much as possible and retry again next year.

Looking forward to working with you.

Thanks And Regards

Sachin Singhal

## **Future Work for the AI Chatbot Integration Project:**

1. **Enhanced Personalization** : Implement advanced personalization techniques to tailor the chatbot experience based on individual user preferences and learning styles.
2. **Advanced NLP Capabilities** : Explore and integrate state-of-the-art natural language processing (NLP) models to improve the chatbot's understanding of complex language nuances and context.
3. **Integration with Learning Management Systems** : Integrate the chatbot with learning management systems (LMS) or educational platforms to provide seamless access to educational resources and track user progress.
4. **Voice and Speech Recognition** : Enhance the chatbot's capabilities with voice and speech recognition technology, enabling users to interact with the bot through voice commands.
5. **Emotional Intelligence** : Further develop the chatbot's emotional intelligence by incorporating sentiment analysis and empathy modeling to provide more empathetic and supportive responses.
6. **Collaborative Learning Features** : Implement collaborative learning features such as group projects, peer review, and collaborative problem-solving sessions facilitated by the chatbot.
7. **Integration with Augmented Reality (AR) and Virtual Reality (VR)** : Explore integration with AR and VR technologies to create immersive learning experiences and simulations within the chatbot interface.
8. **Continuous Learning and Improvement** : Continuously update and refine the chatbot's knowledge base and responses based on user feedback, new learning materials, and advancements in AI technology.
9. **Expansion to New Platforms and Devices** : Extend the reach of the chatbot by making it accessible on a wider range of platforms and devices, including mobile apps, smart speakers, and chat applications.
10. **Research and Collaboration** : Collaborate with educators, researchers, and AI experts to explore innovative approaches to AI-driven learning and develop new features to enhance the chatbot's educational value.