GSoC-2023 Project Proposal
Adversarial Data using Gradio and EvalAI

1 Basic Details

- **Full Name:** Ayush Rathore
- **Location:** Bilaspur, Chattisgarh, India
- **Time Zone:** UTC +5:30 Indian Time Zone
- **Education:** B.Tech in Computer Science & Engineering at Rajiv Gandhi Institute of Petroleum Technology (Batch of 2024)
- **Email:** 20cs3018@rgipt.ac.in
- **GitHub:** ayushr1
- **Skype:** live:.cid.97082b778ed2b0f9
- **Portfolio:** ayushr1.netlify.app
- **Resume:** Ayush-Rathore-Resume.pdf

2 Statement of Motivation

2.1 What is your motivation for participating in Google Summer of Code?

Being part of the Google Summer of Code is an exceptional opportunity to boost my career, get global recognition, gain valuable experience in software development, and contribute to organisations that are solving real-world challenges. The prospect of working alongside industry experts and collaborating with top open-source organisations is genuinely thrilling. As an ardent software developer, I have always been driven to solve real-world problems, and my passion for open source has been my driving force since my high school days when I got my first Android phone and got into the world of open source. My enthusiasm for solving problems, fixing existing bugs, and adding features to open-source software has never wavered. I am motivated to participate in the Google Summer of Code because it presents a remarkable chance to tackle challenging projects that address actual problems faced by society and learn a lot of new things which will help immensely in my career. Under the guidance of experienced mentors, I will learn new skills, work collaboratively with developers from diverse backgrounds, and enhance my communication abilities. Contributing to open-source projects and the Google Summer of Code is the perfect platform to achieve my goals.

2.2 Why did you choose CloudCV?

The initial spark for my interest in CloudCV came from my skills, prior experience, and aligning interests. In particular, I was drawn to the concept and use case of EvalAI, which motivated me to delve into its codebase and gain a better understanding of its inner workings. I discovered EvalAI which aims to build a centralized platform to host, participate, and collaborate in Artificial Intelligence (AI) challenges organized around the globe and hope to help in benchmarking progress in AI, with numerous participants, submissions, and evaluation metrics while providing hosts with the flexibility to customize their challenges as needed. Once I started working on this project, I knew that I couldn't turn back. These experiences provided me with valuable insights into the technical aspects of the project and made me capable to contribute to its growth and development through the Google Summer of Code program.

2.3 Why this project idea?

I have chosen Adversarial Data collection using Gradio and EvalAI as my project idea. I chose this because it provided an excellent opportunity for me to apply my knowledge and skills in a practical domain and integrate multiple things to get it working. As an individual who is passionate about building solutions with the potential to make a substantial impact on a large scale, I am excited at the prospect of contributing to the creation of a simple yet effective solution for gathering data for machine learning models. What intrigues me most about this project is the prospect of starting it from scratch. I have always found such tasks to be a testament to my technical acumen, I have built large-scale personal projects such as AR-Beast Kernel, Saarthi, cpp2latex, Upastithi and many more from scratch earlier and I am eager and can take on the challenge of building this project on an even bigger scale. My experience in similar projects in the past has equipped me with the expertise and confidence necessary to succeed. I am excited to be a part of this project and confident in my ability to contribute significantly to its success. My extensive experience and expertise in building similar solutions make me a suitable candidate for this project. I look forward to working collaboratively with the team to develop an innovative solution that can have a meaningful impact.

2.4 What are your expectations from us during and after the successful completion of the program?

I am truly excited to be a part of this community and contribute to the work at CloudCV. First and foremost, I am eager to forge lasting bonds with members of the CloudCV family. Having a welcoming environment that encourages candid communication and teamwork is crucial to me. I think that by collaborating closely with my mentors and other contributors, I may benefit from their knowledge, improve my comprehension, and develop myself.

Second, I'm determined to provide the CloudCV project with useful contributions. I am certain that my knowledge and abilities can benefit the project, and I am willing to put in a lot of effort to make sure I achieve the project's objectives. I appreciate criticism and ideas for my work since I firmly think that both professional and personal development benefits much from constructive critique.

Finally, I pledge to keep lines of communication open with my mentors and the

larger community. To make sure that everyone is on the same page and that the project is progressing successfully, I think constant communication is crucial. Regular communication will allow me to ask for help when I need it, advance more quickly, and produce better work.
Finally, I'm excited to go on contributing to the open-source community through CloudCV after the Google Summer of Code programme is a success. I think the GSoC project is a great chance for me to learn and progress as a developer, and I'm enthusiastic about the idea of taking on bigger tasks and having a real impact on the project's future.

2.5 What are you hoping to learn?

I am excited at the prospect of gaining valuable experience working alongside seasoned professionals on a significant project like CloudCV. Developing my development skills is particularly enticing, and I am eager to build the solution and familiarise myself with the more complex components of the program and further expand my development knowledge. Moreover, I am confident that staying connected with qualified individuals in the community will provide me with an incredible opportunity to learn and grow. Collaborating on this project has been a great experience, and I have already gained insights into the best practices that we need to follow to maintain high-quality code. Furthermore, I believe that staying connected with such qualified individuals in the community would be a remarkable learning opportunity for me. It has been a great experience collaborating on this project and learning about the practices we need to follow to maintain high-quality code. The codebase is well-managed, with various checks integrated to ensure that syntax and formatting errors are caught before merging a new pull request. These practices have made it easier for new contributors to get started with the project. This experience will help me gain a more comprehensive understanding and take my skills to the next level. I am excited to contribute to this project and make a positive impact.

3 Experiences

3.1 Have you taken Computer Vision, Machine Learning, Artificial Intelligence, Natural Language Processing, and Deep Learning courses?

In my computer science program, I've had the opportunity to study courses in data analysis and statistical methods, data mining, artificial intelligence, and deep learning., I've even completed courses on machine learning from AndrewNG from Coursera (certificates) to deepen my knowledge. I also have elementary exposure to Computer Vision and have made projects utilising MATLAB libraries to build intelligent traffic signal time based on congestion.

3.2 What kind of projects have you worked on in the past? What technologies did you use?

I have experience in development in the backend, frontend, cloud, automation, and Linux kernel development with experience building 10+ full-scale projects and many small projects which are listed below.
I have used technologies such as Python, Django, Flask and JavaScript, and NodeJS for Backend. HTML/CSS, ReactJS for frontend. I have experience setting up infrastructure on AWS for web apps and also have set up CI/CD pipelines. The list of my projects is as follows

Upastithi **- Django, Python, MySQL** -

- Composed a Web App performing **CRUD** operations to digitally log attendance of RGIPT students, **deployed** and in use, enabling 2+ faculties, 5+ tutors and **300**+ students to log, monitor & track attendance.

Saarthi **- NodeJS, ReactJS, MongoDB** -

- Constructed an education platform with courses fetched from **YouTube REST API** with a progress tracker and notes sharing community inventory that increased productivity by 10%.

Cpp2LaTeX **- Flask, Python, Algorithms** -

- Implemented a tool converting C++ functions to LaTeX pseudo-code, reducing time spent on professional writing by over 90%.
- Won the IIT Indore Hackathon.

Dynamic Instructions Calculation **- Python, Flask, Bash** -

- Devised a **GDB**-based bash and python **script** to get instruction analysis for a given program with 100% precision.
- Implemented **REST API** service with the **script** on the **server** to find instructions analysis of the first 10000 instructions.

Sampark **- NodeJS, ReactJS** -

- Developed a peer-to-peer 2-way video chat using **WebRTC** & **SocketIO** web application for real-time video communication.
- Selected as top 23 in NIT-Rourkela Hackathon among 5000+ participants.

3.3 What is your experience with Python and Javascript?

I have experience in Python as well as JavaScript programming. I got introduced to Python and JavaScript in my first year and have been using them extensively in my projects for the last three years. I have made the projects as shown above on the above technology extensively. All the projects are deployed and some of them are used in production. Also I have had volunteer work experience which are as follows

Computer Programming Tutor - RGIPT

- Taught programming in **C** and **Python** to over 57 junior students, guided in solving **100+** problems of basic Algorithms.

Technical Head - ACM - SC

- Automated bulk emailing using python scripts and designed the official website, leading to **99%+** time savings.
- Lectured on **Open Source Development, Git,** and **GitHub** and tutored workshops for over 100 attendees.

4 CloudCV Contributions

4.1 Pull Requests

- Fix #3884: Improve Validate Challenge Config to handle corner cases #3890

  Impact: This script handles every corner case which could happen while creating a challenge.

  (+97 -58 Lines)
- Enhancement: Add a Script to create challenges locally #71

  Impact: The script would help in testing, validating, and creating challenges locally which would help contributors to facilitate better testing and evaluation.

  (+298 Lines)
- Fix #342: Fix error on get all challenges #344

  Impact: This fixed the major issues in eval ai cli where we were not able to get challenges.

  (+3 -3 Lines)

4.2 Issues

- Documentation: Host challenge: Fix duplication 3913

5 Other open-source contributions

- Kernel Adiutor: Fix Temperature Calculation bug. #532
- Kernel Adiutor: ThunderCharge: Fix Charge current not changing bug. #518

AR_Beast Kernel **- Linux Kernel, Android, App, C** - Open Source

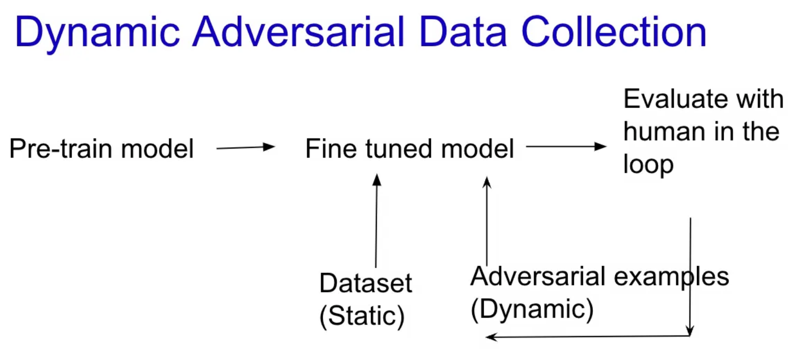
- Built Custom Linux Kernel for 4 Android Devices as an open source project, which helped **9000+** users to get an optimized device experience.
- Coded device-specific Thermal control driver & Current Control driver in C, which reduced heating, increased performance by **10**% and improved charging time by **30** min.
- Overhauled Android App to easily configure over 50 features offered by the kernel, reaching over **30,000** downloads.

6 Project Details

6.1 Project Description

This project aims to develop an infrastructure that enables the collection of adversarial data for models submitted to EvalAI. This is an important task as machine learning models can be vulnerable to adversarial attacks, which can cause the model to make incorrect predictions or misclassify data. Static benchmarks, while widely used to evaluate your model's performance, are fraught with many issues: they saturate, have biases or loopholes, and often lead researchers to chase increment in metrics instead of building trustworthy models that humans can use.

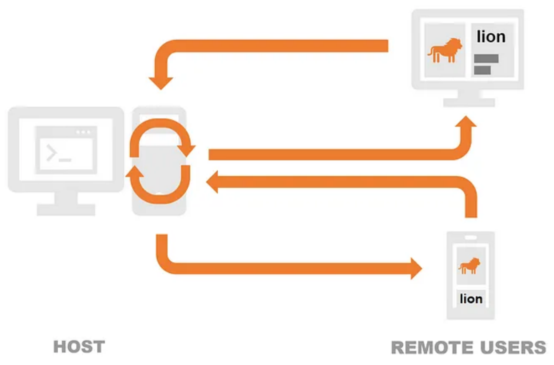
Dynamic adversarial data collection (DADC) holds great promise as an approach to mitigate some of the issues of static benchmarks. In DADC, humans create examples to fool state-of-the-art (SOTA) models. This process offers two benefits it allows users to gauge how robust their models are; it yields data that may be used to further train even stronger models. This process of fooling and training the model on the adversarially collected data is repeated over multiple rounds leading to a more robust model that is aligned with humans.


Dynamic Adversarial Data Collection
Pre-train model
Fine tuned model
Evaluate with human in the loop
Dataset (Static)
Adversarial examples (Dynamic)

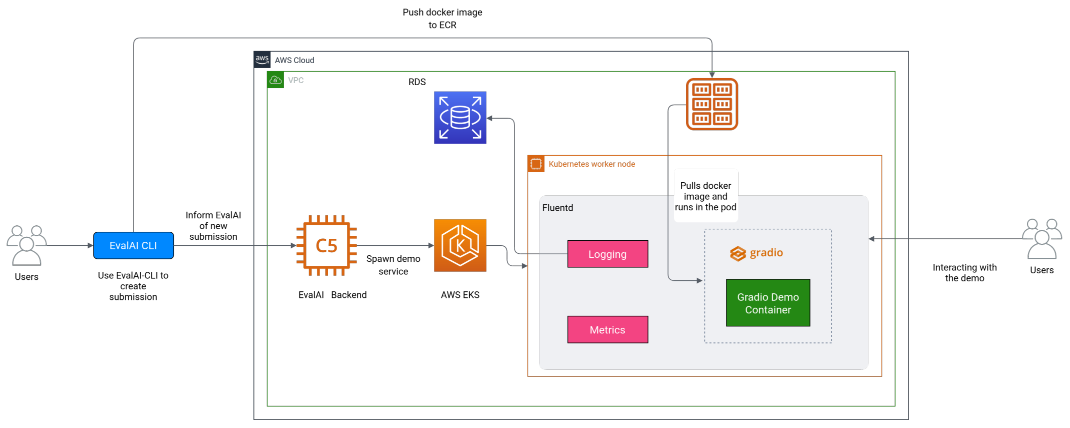
By integrating Gradio with EvalAI's code upload challenge pipeline and deploying models as web services, the project aims to capture user interactions and provide a dataset for each submission that can be used to evaluate the robustness of the model. This will help to improve the quality of machine learning models by identifying and addressing vulnerabilities.

The approach of using Gradio, a platform for building web-based interfaces for machine learning models, is the best way to go because it provides an intuitive and interactive way for users to interact with the model. This will make it easier for users to provide feedback on the model's performance, which will ultimately lead to better evaluations of the model's robustness.

The project will be delivered in several stages to ensure that the integration of Gradio with EvalAI is done efficiently and effectively.



This project is an important step towards improving the quality of machine learning models by identifying and addressing vulnerabilities through the collection of adversarial data. The integration of Gradio with EvalAI is an efficient and effective way to achieve this goal, and the project will be delivered in several stages to ensure that the integration is done correctly.



6.2 Deliverables

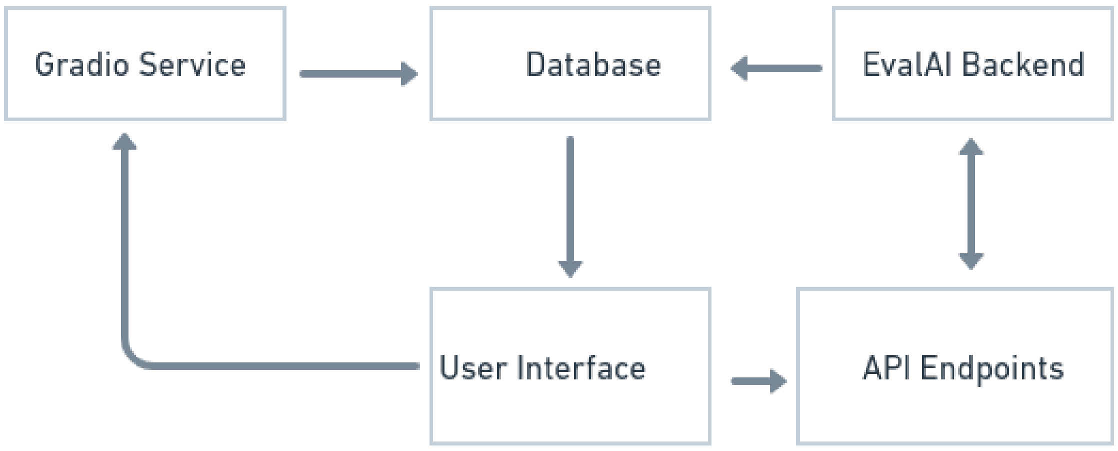
The project will have the following Deliverables:

1. Build system design for EvalAI and Gradio integration.
2. Setup Gradio integration for a single VQA demo as a proof of concept.
   (a) Create a Gradio interface wrapper that is modular for model inputs.
   (b) Integrate the Gradio interface with the dockerized model for inference.
3. Set up auto launching of a worker service for deploying a Gradio app using a celery task from EvalAI Django backend.
4. Add frontend controls to deploy a Gradio web service for a single submission made by a participant to a code upload challenge.
5. Add support to log interactions made by a user on the Gradio web service and push the interactions to a database table specific to a demo.
6. Add API support to export the logged interactions for a demo.
7. Implement frontend changes to allow users to download the logged interactions in a standard dataset format.

7 Project Implementation

7.1 Build system design for EvalAI and Gradio integration

The first deliverable of the project is to build a system design for integrating Gradio with EvalAI. Here is a detailed procedure on how to achieve this:



1. Define the scope of the integration:- This involves determining what the integration will achieve and what data will flow between Gradio and EvalAI. In this project, the integration aims to collect adversarial data for models submitted to EvalAI. The data flow will involve users interacting with the Gradio web service, which records their interactions and saves them to a database. This data will be used to evaluate the robustness of the model submitted to EvalAI.

2. Identify the key components required for the integration:- These components include the Gradio web service, the database for storing user interactions, and the API endpoints for communication between Gradio and EvalAI. The Gradio web service will be used to record user interactions and save them to the database. The database will store these interactions, which can be used to evaluate the robustness of the model. The API endpoints will allow communication between Gradio and EvalAI, enabling the submission of models and the retrieval of user interactions.

3. Define the API endpoints required for communication between Gradio and EvalAI :- These endpoints will allow Gradio to submit models to EvalAI and retrieve user interactions from the database. Some of the API endpoints that may be required include:

   - `/api/models/`: This endpoint could be used by Gradio to submit models to EvalAI.
   - `/api/interactions/`: This endpoint could be used by Gradio to retrieve user interactions from the database.

4. Develop a high-level architecture diagram to visualize the integration:- This diagram will show how the different components of the integration interact with each other. The architecture diagram should include the following components:

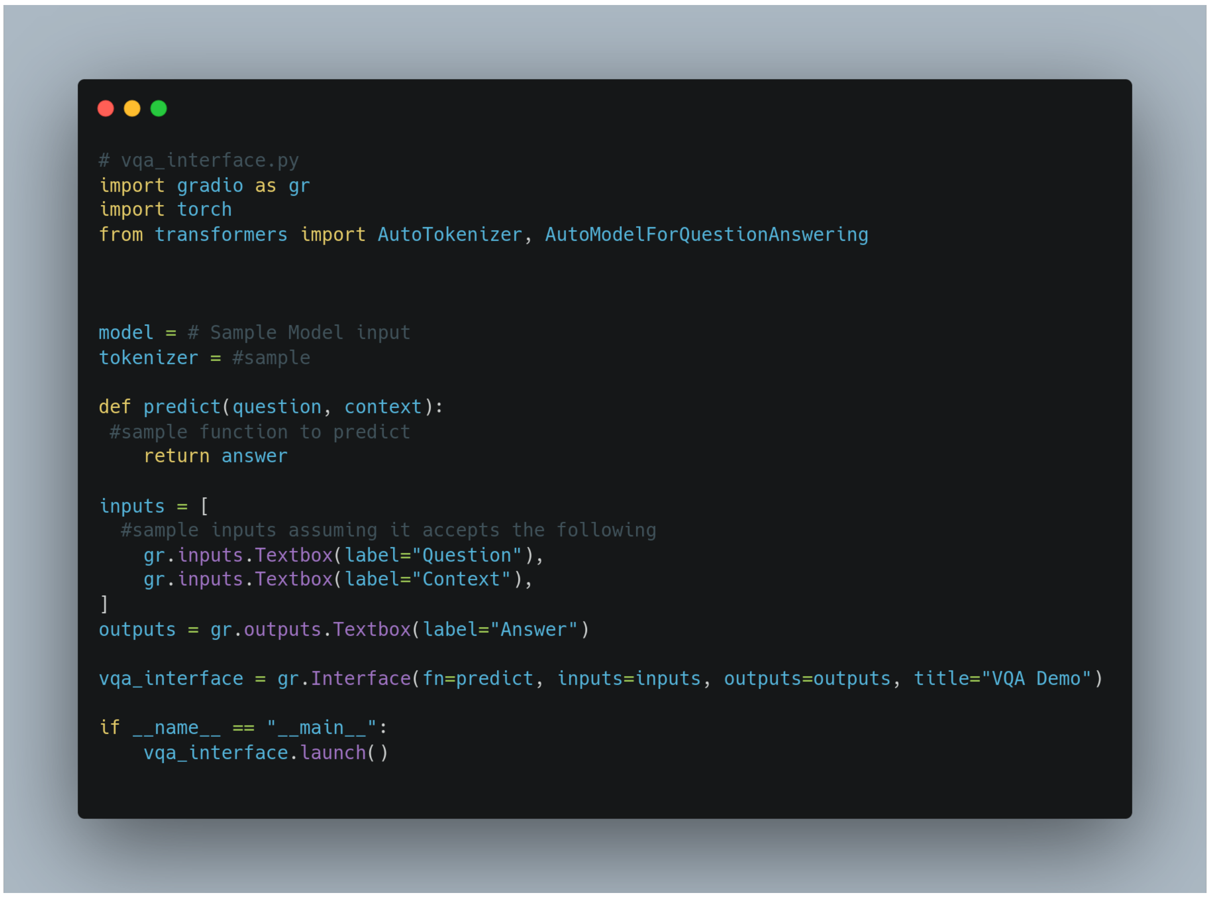
   - Gradio web service: This component will be responsible for recording user interactions and saving them to the database.

- Database: This component will store the user interactions.
- EvalAI backend: This component will receive models from Gradio and allow users to submit models to the platform.
- API endpoints: These endpoints will allow communication between Gradio and EvalAI, enabling the submission of models and the retrieval of user interactions.
- User interface: This component will allow users to interact with the Gradio web service and submit models to EvalAI.

7.2 Setup Gradio integration for a single VQA demo as a proof of concept

To achieve this, follow these steps:

1. Select a VQA model and dataset for the demo. We can find models on HuggingFace which we can use for the demo.
2. Create a modular Gradio interface for the VQA model:
   (a) The Gradio interface wrapper should be designed to be modular so that it can be used with different types of models.
   (b) The input and output formats for the wrapper should be defined based on the requirements of the model that will be used with it.
   (c) The Gradio interface wrapper can be implemented using Gradio's API, which provides a simple way to define the input and output formats and connect them to the underlying model.
3. Integrate the Gradio interface with a dockerized model for inference:
   (a) Dockerize the VQA model so that it can be easily deployed and run in a containerized environment.
   (b) Write a script to integrate the Gradio interface with the dockerized model.
   (c) The script should include the necessary commands to start the docker container, load the model into memory, and make predictions using the Gradio interface.
   (d) Test the integration to ensure that the Gradio interface can successfully make predictions using the dockerized model.



```python
# vqa_interface.py
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForQuestionAnswering



model = # Sample Model input
tokenizer = #sample

def predict(question, context):
 #sample function to predict
    return answer

inputs = [
  #sample inputs assuming it accepts the following
    gr.inputs.Textbox(label="Question"),
    gr.inputs.Textbox(label="Context"),
]
outputs = gr.outputs.Textbox(label="Answer")

vqa_interface = gr.Interface(fn=predict, inputs=inputs, outputs=outputs, title="VQA Demo")

if __name__ == "__main__":
    vqa_interface.launch()
```

This is a sample code snippet By following these steps, we can successfully set up a Gradio integration for a single VQA demo as a proof of concept.

7.3 Set up auto launching of a worker service for deploying a Gradio app using a Celery task

To implement this we would be proceeding as below:

1. Configure Celery on EvalAI's Django backend

2. Write a Celery task that can launch a Gradio app. The task should include the necessary commands to:

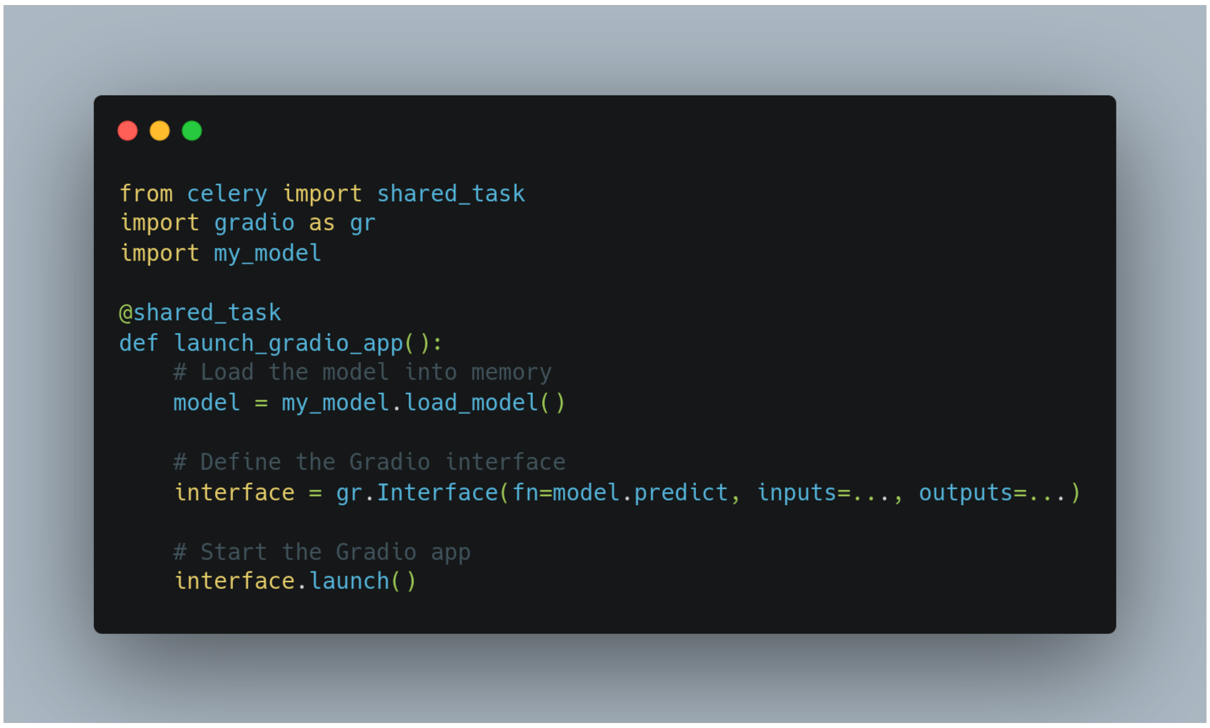
   (a) Start the Gradio app.

   (b) Load the model into memory.

   (c) Make predictions using the Gradio interface.

3. Configure EvalAI's Django backend to automatically launch the Celery task when a code upload challenge is submitted. This can be done by Using Celery's beat scheduler or by Triggering the task

4. Test the task to ensure that it can launch a Gradio app successfully submitting the challenge and verifying that the Gradio app is launched and ensuring model can make predictions using the Gradio interface.



```python
from celery import shared_task
import gradio as gr
import my_model

@shared_task
def launch_gradio_app():
    # Load the model into memory
    model = my_model.load_model()

    # Define the Gradio interface
    interface = gr.Interface(fn=model.predict, inputs=..., outputs=...)

    # Start the Gradio app
    interface.launch()
```

7.4 Add frontend controls to deploy a Gradio web service for a single submission made by a participant to a code upload challenge

This simplifies the process of deploying and testing models for participants, providing an easy-to-use and user-friendly interface using Gradio. Secondly, it enables participants to iteratively improve their models by testing them with various inputs and parameters using the Gradio interface. This allows for more efficient and effective model development. Finally, the addition of front-end controls offers a more interactive and engaging experience for participants, which can boost their engagement and satisfaction with the challenge. Here is a detailed procedure on how to achieve this deliverable:

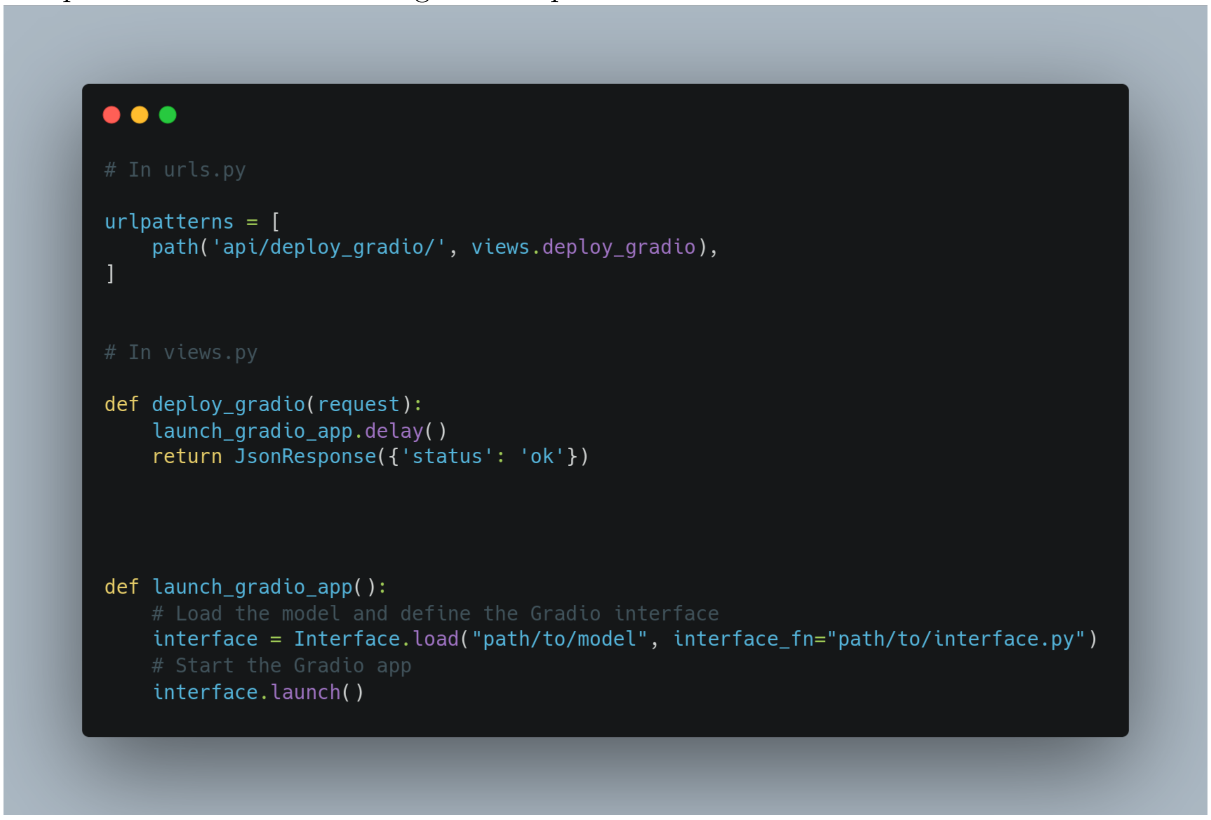
1. Modify the EvalAI front to allow users to deploy a Gradio web service for their submission. This involves adding a new button or link to the submission view that triggers the deployment process. The front end can also display the status of the deployment and any errors or warnings that occur during the process.

2. When the user clicks the button or links to deploy the Gradio web service, the front end should send a request to the backend to initiate the deployment process. This can be done using a Django API endpoint that triggers the Celery task to launch the Gradio app.

3. Once the Gradio web service is deployed, the front end should display the URL of the service to the user. This URL can be generated dynamically by the backend and returned to the front as a response to the deployment request.

4. Test the frontend changes to ensure that users can deploy a Gradio web service successfully. This involves submitting a code upload challenge and verifying that the Gradio web service is deployed and accessible from the URL provided by the front end.

Sample Code demonstrating the api which will be called from front end



```python
# In urls.py

urlpatterns = [
    path('api/deploy_gradio/', views.deploy_gradio),
]


# In views.py

def deploy_gradio(request):
    launch_gradio_app.delay()
    return JsonResponse({'status': 'ok'})




def launch_gradio_app():
    # Load the model and define the Gradio interface
    interface = Interface.load("path/to/model", interface_fn="path/to/interface.py")
    # Start the Gradio app
    interface.launch()
```

7.5 Add support to log interactions made by a user on the Gradio web service and push the interactions to a database table specific to a demo

By logging user interactions, the collected data can be used to evaluate the robustness of the models. To add support to log interactions made by a user on the Gradio web service and push the interactions to a database table specific to a demo, the following steps can be followed:

1. Create a database table to store user interactions. The table should include columns for relevant information such as user ID, timestamp, inputs, and outputs.

2. Add logging script to Gradio interface that will collect user interactions such as inputs and outputs from the web service. The logging script should include code to connect to the database and push the collected data to the relevant table.

3. Test the logging script and ensure it correctly captures user interactions and pushes the data to the database. To do this, make several test submissions to the Gradio web service and check that the relevant data has been added to the database.

4. To keep the data organized, create a separate table for each demo in the database. This can be achieved by adding a new table for each demo that will be using the Gradio web service.

5. Modify the logging script to store the data collected from each demo in the corresponding table. This can be achieved by passing the demo ID as an argument

to the logging script, which will then use this to determine the correct table to store the data in.

6. Test the logging script to ensure that it is correctly storing data in the demo-specific tables. To do this, make several test submissions for each demo and check that the relevant data has been added to the correct table.

7.6 Add API support to export the logged interactions for a demo

Here is a detailed procedure on how to add API support to export the logged interactions for a demo:

1. **Define the API endpoint for exporting logged interactions**:- The API endpoint should be defined in the EvalAI Django backend code. The endpoint URL should be unique and follow RESTful API conventions. For example, the endpoint URL could be something like :- `api/logged_interactions/<demo_id>/`

2. **Write an API endpoint that can retrieve logged interactions from the database**:- This API endpoint should be responsible for querying the database and retrieving the logged interactions for a specific demo. It should then convert the logged interactions data into a format that can be easily downloaded, such as CSV or JSON.

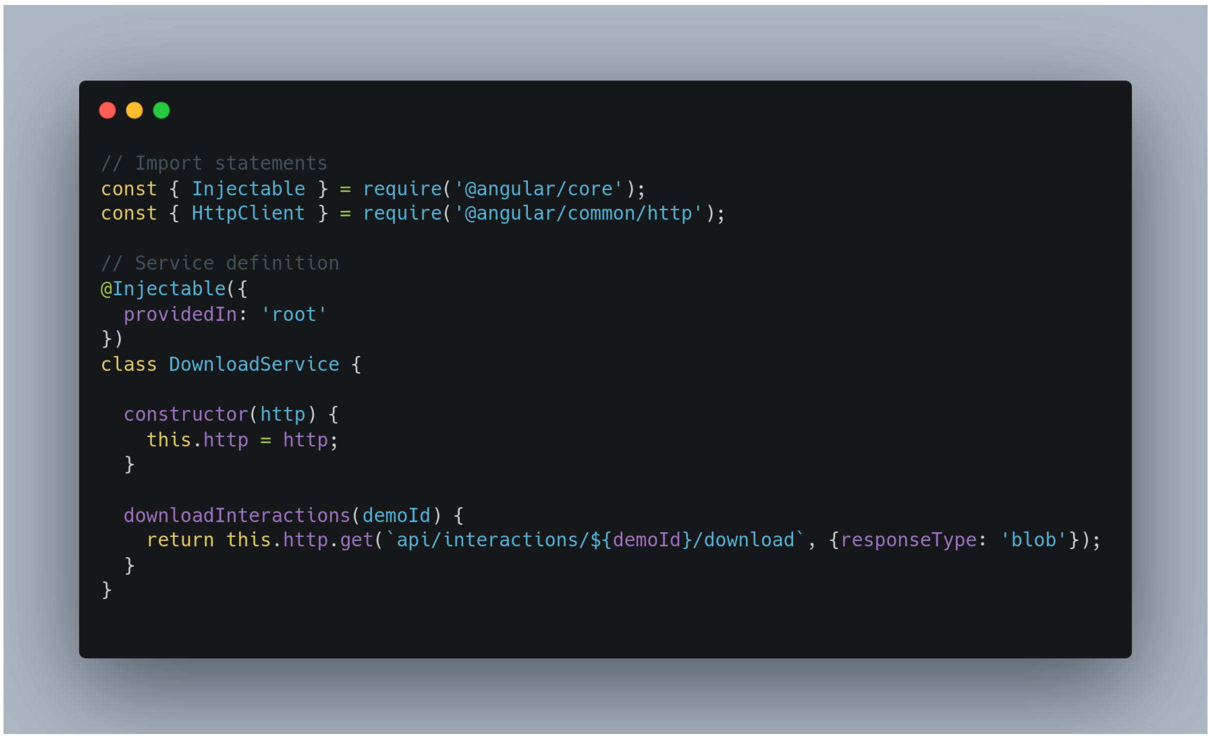
3. **Test the API endpoint**: To test the API endpoint, use a tool such as Postman to request the endpoint URL with the appropriate demo ID parameter. Verify that the returned data is in the correct format and contains the correct logged interaction data.

7.7 Implement frontend changes to allow users to download the logged interactions in a standard dataset format.

To implement the frontend changes that allow users to download the logged interactions in a standard dataset format, we can follow the steps below:

1. Determine the standard dataset format you want to use for the logged interactions. Examples of standard dataset formats include CSV, JSON, and XML. We can use any of these formats.

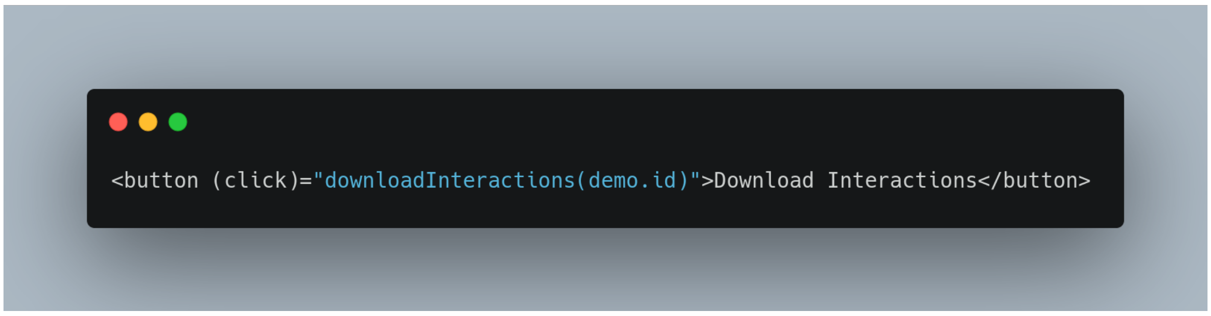
2. Create a service in Angular that will allow users to download the logged interactions in the chosen standard dataset format. This view will fetch via API. Here's an example implementation of the service:



```
// Import statements
const { Injectable } = require('@angular/core');
const { HttpClient } = require('@angular/common/http');

// Service definition
@Injectable({
  providedIn: 'root'
})
class DownloadService {

  constructor(http) {
    this.http = http;
  }

  downloadInteractions(demoId) {
    return this.http.get(`api/interactions/${demoId}/download`, {responseType: 'blob'});
  }
}
```

In this example, we assume that there is an API endpoint that returns the interaction data for a specific demo in CSV format. The downloadInteractions method of the service uses Angular's HttpClient to make an HTTP GET request to the API endpoint and set the responseType to blob to download the file as a blob object.

3. Add a link or button to the EvalAI front that will allow users to access the view created in step 2. Here's an example implementation:



```
<button (click)="downloadInteractions(demo.id)">Download Interactions</button>
```

In this example, we assume that there is an Angular component that renders a page for a specific demo (demo.id). The button uses the (click) event to call the downloadInteractions method of the DownloadService with the appropriate demo ID.

4. Test the frontend changes to ensure that users can download the logged interactions in CSV format. Create a demo and deploy a Gradio web service for the demo. Interact with the Gradio web service and make some submissions. Then, click on the "Download Interactions" link to download the interactions in CSV format. Verify that the downloaded file contains the expected data.

8 Project Schedule

8.1 GSoC period

Period	**Duration**	**Goals**
Community Bonding	May 4 - May 28	- Explore EvalAI's code base to understand its different components and Discuss project requirements with mentors to better understand their expectations. - Learn relevant skills, get familiar with Gradio
Week 1	May 29 - Jun 4	- Build system design for EvalAI and Gradio integration
Week 2	Jun 5 - Jun 11	- Setup Gradio integration for a single VQA demo as a proof of concept
Week 3	Jun 12 - Jun 18	- Buffer Period - Create a Gradio interface wrapper that is modular for model inputs
Week 4	Jun 19 - Jun 25	- Integrate the Gradio interface with dockerized model for inference
Week 5	Jun 26 - Jul 2	- Buffer Time for anything left - Setup auto-launching of a worker service for deploying a Gradio app using a celery task from EvalAI Django backend
Week 6	Jul 3 - Jul 9	- Continue setup auto-launching of a worker service for deploying a Gradio app using a celery task from EvalAI Django backend - Buffer Time
Midterm Evaluation	Jul 10 - Jul 14	- Submit midterm evaluations
Week 7	Jul 15 - Jul 21	- Add frontend controls to deploy a Gradio web service for a single submission made by a participant to a code upload challenge
Week 8	Jul 22 - Jul 28	- Buffer Time for anything left - Add support to log interactions made by a user on the Gradio web service and push the interactions to a database table specific to a demo
Week 9	Jul 29 - Aug 4	- Add API support to export the logged interactions for a demo
Week 10	Aug 5 - Aug 11	- Implement frontend changes to allow users to download the logged interactions in a standard dataset format
Week 11	Aug 12 - Aug 18	- Finalise any pending work and Ensure the project is well-documented and tested.
Final Evaluation	Aug 21 - Aug 28	- Submit final work product and mentor evaluation

8.2 Pre GSoC period

To continue contributing to EvalAI, I plan to focus on fixing issues, adding new features, and improving the codebase. I will begin by ensuring that any outstanding pull requests I have submitted are fully addressed and merged. If any further action is required, I will communicate with my mentors. Furthermore, I plan to contribute to the project by taking on new challenges and exploring areas of the codebase that I am less familiar with. This will allow me to expand my knowledge and skills, as well as provide valuable contributions to the project. I would take the opportunity to work on new issues and take on more responsibilities within the project.

8.3 Post GSOC period

Contributing to open source is not just something I am doing for GSOC. It has been a valuable experience for me to solidify my skills and help others. I am passionate about expanding my knowledge and sharing it with others. Even after the GSOC period ends, I will ensure that the projects I have implemented have proper documentation for knowledge transfer. I plan to continue contributing to this project and making the platform better.

9 Time

I am fully dedicated to devoting my complete focus and effort to the project from start to finish. During my summer break, from May 5th to August 31st, which is the complete period for GSoC I have no prior engagements or internships, which allows me to commit more than 40 hours per week to the project. I am enthusiastic about investing my time and energy into the project to ensure its successful completion. My usual work hours are from 11 AM to 2 AM (Indian Standard Time), but I am flexible and can adjust my schedule to accommodate the project's needs. Furthermore, I am open to spending additional time learning new skills and tackling any obstacles that may arise during the project. I have already allocated ample time to develop a comprehensive plan, including buffer days, and to acquire any necessary skills to ensure that I remain on track and deliver high-quality work. However, I am willing to devote extra hours to ensure the project's timely completion and exceed expectations.

References

[1] Cloud-CV GSoC Ideas. GitHub. Retrieved April 4, 2023, from https://github.com/Cloud-CV/GSoC-Ideas/issues/60.

[2] Cloud-CV GSoC 2023 Proposal Template. GitHub. Retrieved April 4, 2023, from https://github.com/Cloud-CV/GSoC-Ideas/wiki/GSOC-2023-Proposal-Template.

[3] EvalAI Documentation. Retrieved April 4, 2023, from https://evalai.readthedocs.io/.