VisionPulse (दिव्य दृष्टि)

AI-Enhanced Learning and Interaction Assistant for Visually Impaired people at IIIT Delhi.

Submitted By:
Akash Kushwaha 2021514
Lakshay Kumar 2021061
Manav Mittal 2021538
Shreyas Kabra 2021563
Tanmay Singh 2021569
Utkarsh Venaik 2021570

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Introduction

VisionPulse is our revolutionary idea in the educational domain, specifically designed to cater to the needs of visually impaired students at IIIT Delhi. By leveraging the multifaceted capabilities of Computer Vision and Natural Language Processing, VisionPulse aims to build a Large Multi-Model (LMM) system that addresses the unique challenges faced by these students.

Our model is powered with object recognition techniques, deciphering complex scenarios in a localized environment and delivering precise, actionable knowledge about the surroundings to turn daily campus navigation and social encounters into a vivid experience.

Our model also incorporates features like real-time obstacle detection, spatial location detection, aiding in reading emails, reading, understanding, and conveying text-based instructions in front of the user, enhancing the feasibility and easing the individual's life. VisionPulse will be able to perform actions based on voice commands and be capable of generating audio-based responses as well.

Introduction

Our contribution of VisionPulse to the IIIT Delhi community can act as a gateway in the realm of education for visually impaired students by creating a more inclusive, empowering, and independent educational experience for them.

Our project has the potential to become one of the breakthrough contributions in the academic landscape which leverages existing tools and technology to a significant effect, which will help bridge the gaps in education for differently-abled students and create a healthy, inclusive environment where every student can thrive & unlock their full potential to scale great heights & achieve all their dreams.

Problem Statement

Visually impaired students at IIIT Delhi face a significant challenge: the lack of accessible, context-aware educational support tailored to their unique needs.

Existing tools often fall short, limiting their ability to seamlessly interact with peers, instructors, and their physical environment.

This gap restricts their participation in both academic and social activities.

Problem Statement

VisionPulse’s Solution:

VisionPulse seeks to address this challenge by harnessing advanced Computer Vision and Natural Language Processing techniques to create a Large Multi-Modal (LMM) system.

This innovative solution aims to provide visually impaired students with:

- Real-time obstacle detection
- Location Identification (Visual Question Answering)
- Agent-based system for reading and sending emails
- LLAMA 3 Guard
- General Chat tool
- Student ERP Portal Agent (SQL Agent)
- Agent for retrieving IIIT Delhi Policies

Motivation & Literature Survey

Motivation:

- Visually impaired people face a lot of challenges doing usual tasks & are often alienated from activities which most of us can do without much fuss. Hence, ***the primary motivation of our work was to empower the visually impaired people to do everyday tasks at ease & help them overcome the challenges that they face often.***
- Inform the user whether they can move forward and provide real-time alerts, such as beeping, when obstacles are detected.
- Since the visually impaired person may not be able to access documents like IIIT Delhi policies, etc., we developed a model that has the knowledge related to IIIT Delhi built within it, which can be accessed and queried by the user at ease.

Motivation

- Detect the user's current location and provide guidance on routes to reach their desired destination.

- Integrate the student's mailbox by developing an email agent integrated with SMTP to enable users to read and write emails using audio prompts eg: "write a formal mail to admin B.Tech inquiring about the last date of fee payment".

- Allow users to control and navigate the system through voice commands. ***We aim to tailor each feature of this system to enhance its usability on the IIIT Delhi campus.***

Literature Review

1. **Bootstrapping Language Image Pre-training (BLIP)** Li et al. introduce BLIP, a Vision-Language Pretraining framework that achieves state-of-the-art performance by using CapFit to generate and filter synthetic captions, enhancing dataset quality. The framework includes a multimodal mixture of encoder-decoder models with pre-training objectives like image-text contrastive learning and image-conditioned language modeling. VisionPulse will utilize BLIP for Visual Question Answering
2. **Pathways Language and Image (PaLI)** PaLI models vision and language tasks jointly, leveraging large language models (LLMs) and Vision Transformers (ViTs) to process multimodal inputs. It achieves state-of-the-art performance in image captioning, visual question-answering, and scene-text understanding. VisionPulse will use PaLI-VQA for question-answering tasks.
3. **LLama 3** The Llama 3 series includes models designed for multilingual tasks, coding, reasoning, and tool utilization, with a dense Transformer having 405 billion parameters. It achieves performance comparable to GPT-4 and integrates image, video, and speech capabilities. VisionPulse will utilize various fine-tuned Llama VQA models for language functionalities.
4. **Wu-Palmer Similarity Score (WUPS)** Malinowski et al. introduce the WUPS score to evaluate system-generated answers, accounting for semantic fuzziness between classes. It penalizes underestimation and overestimation of answers, providing a measure that balances precision and forgiveness. VisionPulse will incorporate WUPS for performance evaluation.

Proposed Contributions

- Leverage Large Multi-modal Models for answering questions based on knowledge extracted from visual data.
- Identify objects in a localized environment.
- Provide detailed and descriptive answers to user queries.
- Read, understand, and answer questions based on text found in images, such as checking expiry dates on food items, reading documents, or identifying items on a menu.
- Identify locations by reading and recognizing banners or signs (e.g., identifying ‘Brewbakes’ or ‘Center of Artificial Intelligence’).

Proposed Contributions

- Inform the user whether they can move forward and provide real-time alerts, such as beeping, when obstacles are detected.

- Pipeline to annotate visual feed using audio feed for knowledge base creation.

- Since the visually impaired person may not be able to access documents like IIIT Delhi policies, etc., the model should have knowledge related to IIIT Delhi built within it, which can be accessed and queried by the user at ease.

- Parallel Pipeline to process the video and audio feed to text with “smart storage” for future QA based on past context using RAG.

Proposed Contributions

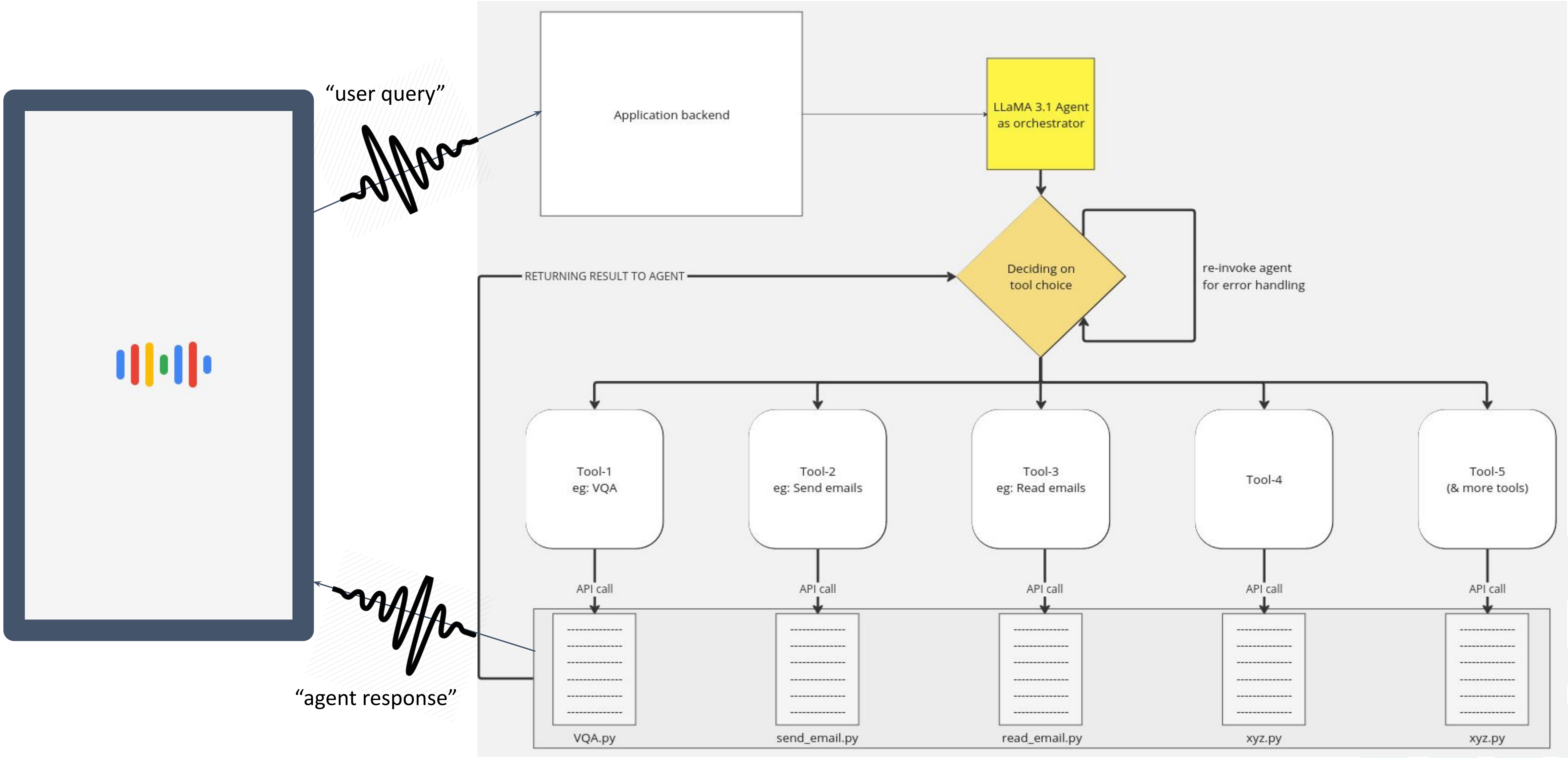
- Detect the user's current location and provide guidance on routes to reach their desired destination.

- Recognize and identify faces relevant to the user.

- Integrate the student’s mailbox by developing an email agent integrated with SMTP to enable users to read and write emails using audio prompts eg: “write a formal mail to admin-btech@iiitd.ac.in inquiring about the last date of fee payment”.

- Allow users to control and navigate the system through voice commands. We aim to tailor each feature of this system to enhance its usability on the IIIT Delhi campus.

Model Architecture Schematic



"user query"

Application backend

LLaMA 3.1 Agent as orchestrator

Deciding on tool choice

re-invoke agent for error handling

RETURNING RESULT TO AGENT

Tool-1 eg: VQA

Tool-2 eg: Send emails

Tool-3 eg: Read emails

Tool-4

Tool-5 (& more tools)

API call

API call

API call

API call

API call

VQA.py

send_email.py

read_email.py

xyz.py

xyz.py

"agent response"

Application Features

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

LLaMA 3.1 Based Agent Orchestrator

1. Agent llama based
2. contains many tools
3. some tools are agents in itself
4. agent directs query to tool
5. error proof feedback mechanism: even if direction fails it automatically retries
6. Speech to text and text to speech using OpenAI whisper / Google Chirp

Real-time Obstacle Detection

Dataset Collection:

Manually collected various images of different locations of IIIT Delhi focusing on obstacles visually impaired individuals may encounter at different body levels.

Head-Level Perspective:

Objects at eye level or above (e.g., door frames, signs, branches).

Images captured at head height to simulate a natural viewpoint.

Knee-Level Perspective:

Objects closer to the ground (e.g., chairs, steps, small animals).

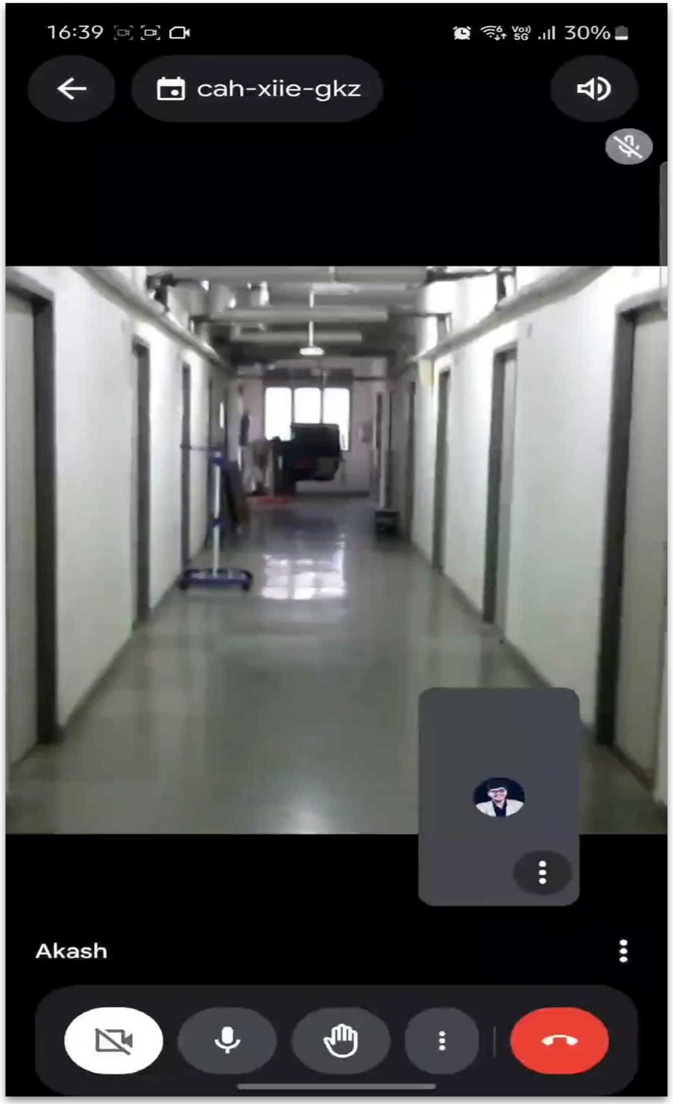
Images taken from knee height to account for low-lying obstacles.

Capturing images from different body levels significantly enhances the robustness and comprehensiveness of our obstacle detection system.

Obstacle

Non Obstacle

Obstacle Detection Demo



Real-time Obstacle Detection

Dataset Description:

Training Set:

Class 0 (obstacle): 80 images

Class 1 (Non obstacle): 87 images

Testing Set: 40 images

Model Approach:

We used **Teachable Machine** to train a CNN based binary classification model.

Input (224x224x3) -> 3 CNN layers -> Output (2 classes)

The **preprocessing steps** involve:

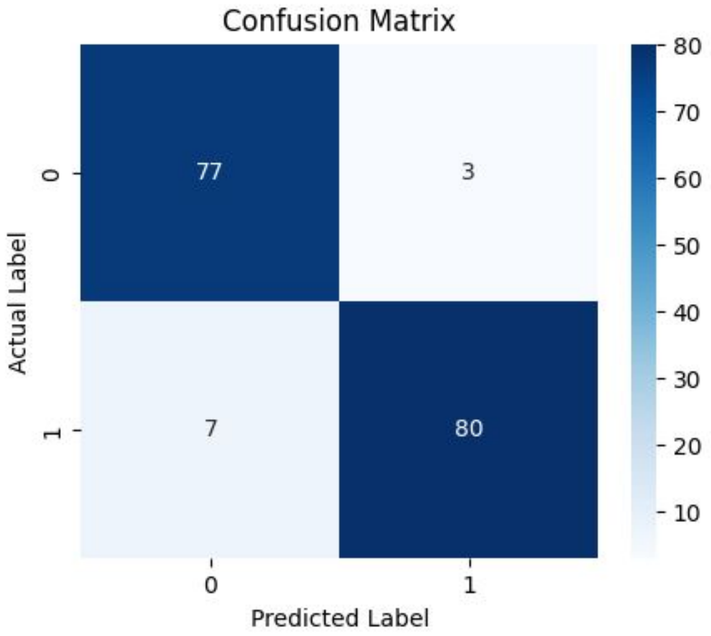
1. Resizing Images: Convert all input images to 224x224 dimensions to match the model's expected input size.
2. Normalization: Scale pixel values to the range [0, 1] by dividing by 255, ensuring consistent input to the model.
3. Splitting Dataset: Divide the dataset into training, validation, and test sets to train and evaluate model performance.

It is compiled with the Adam optimizer, binary cross-entropy loss, and accuracy as the metric.

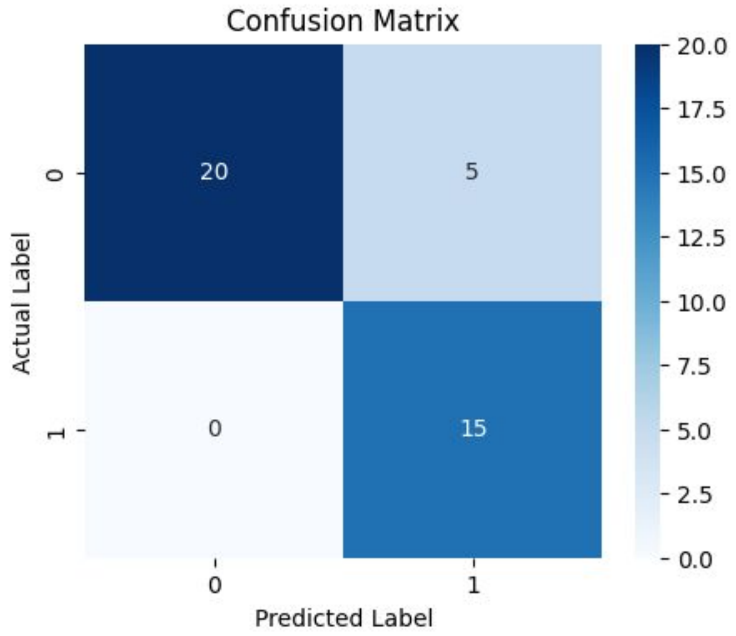
Real-time Obstacle Detection

Results:

Metric	Accuracy	Total Inference time	Avg Inference time	False Negative Rate	Recall
Training Set	0.9401	0.6435	0.0039	0.0805	0.9195
Testing Set	0.8750	0.1614	0.0040	0.0000	1.0000

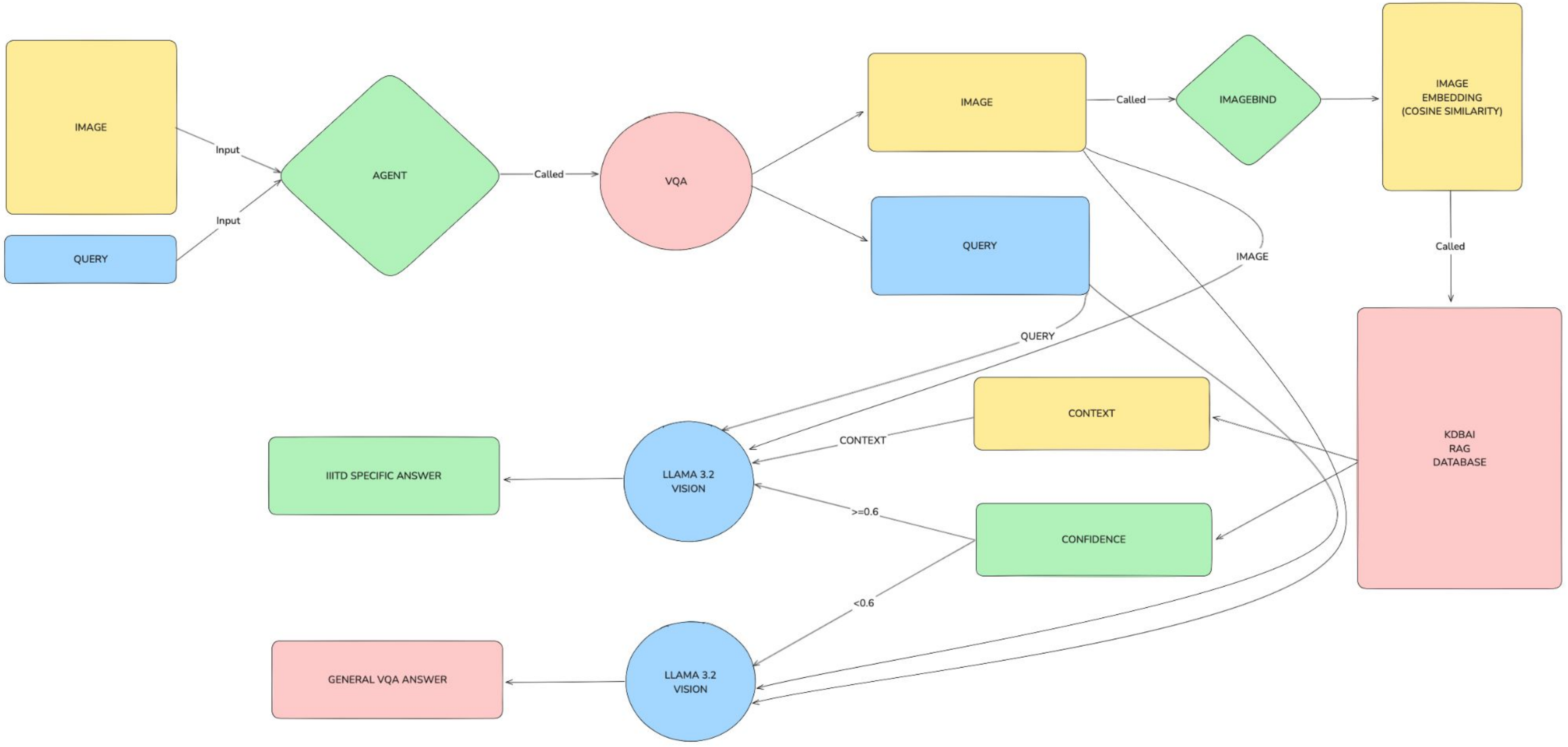


Training Set



Testing Set

Visual Question Answering Tool (RAG-Based VQA)



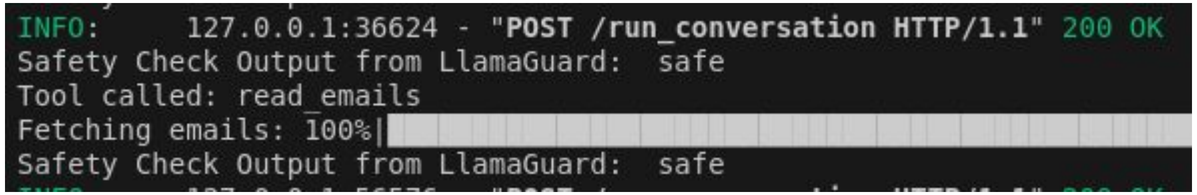
Receive Emails Tool

In our system, the **IMAP server** is connected to the student's Gmail account, allowing it to automatically receive and manage incoming emails. The system will notify the user of new emails, read the latest email aloud, and convert its content into speech for easy consumption by visually impaired students.

Input:

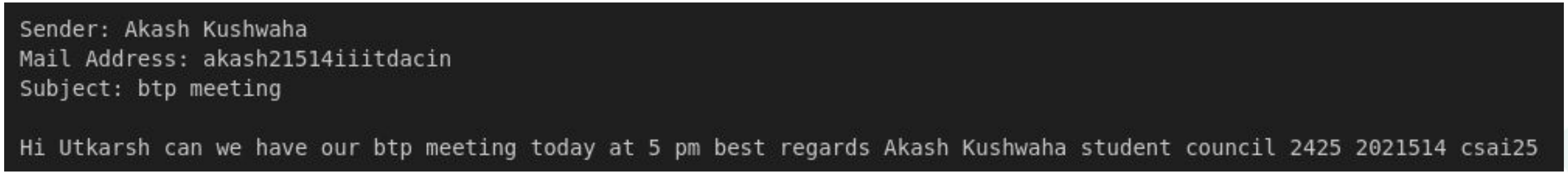
```
"read any new mail"
```

Llama Guard: Llama orchestrator selects appropriate tool to be used for this input which is **read email** tool in this scenario and then Llama guard detects any harmful content and verify the mail as safe or unsafe.



```
INFO:     127.0.0.1:36624 - "POST /run_conversation HTTP/1.1" 200 OK
Safety Check Output from LlamaGuard:  safe
Tool called: read_emails
Fetching emails: 100%|
Safety Check Output from LlamaGuard:  safe
```

Once an email is received, it is read aloud, providing users with easy access to email content without needing to see the screen.



```
Sender: Akash Kushwaha
Mail Address: akash21514iiitdacin
Subject: btp meeting

Hi Utkarsh can we have our btp meeting today at 5 pm best regards Akash Kushwaha student council 2425 2021514 csai25
```

Send Emails Tool

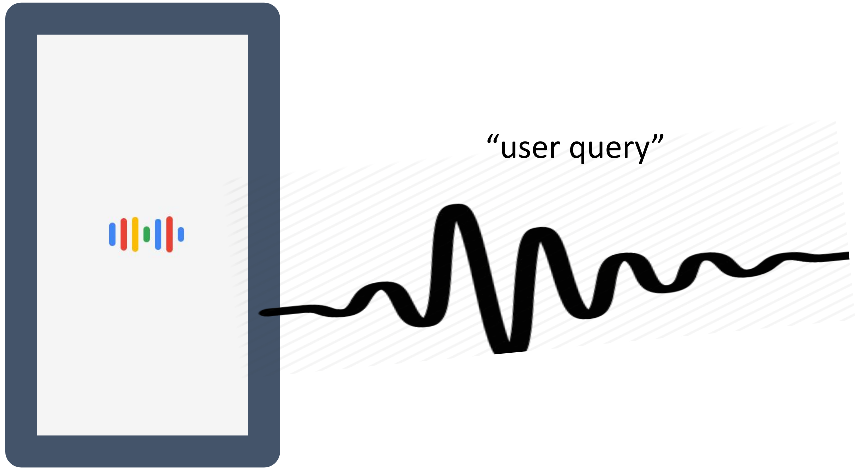
- The Send Emails tool uses a LLAMA model in the front which has access to a plethora of tools, whose access is controlled by an orchestrator.

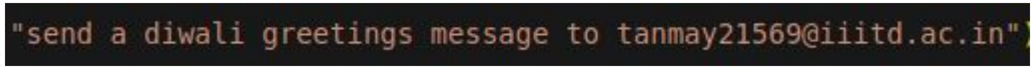
- The query made to the LLAMA model is classified with the help of the LLAMA orchestrator & classifies it, whether its a 'send' email query.

- Once the query is classified, the orchestrator invokes access to the 'send_email' tool, which does the job of sending the email using another LLAMA model in its base.

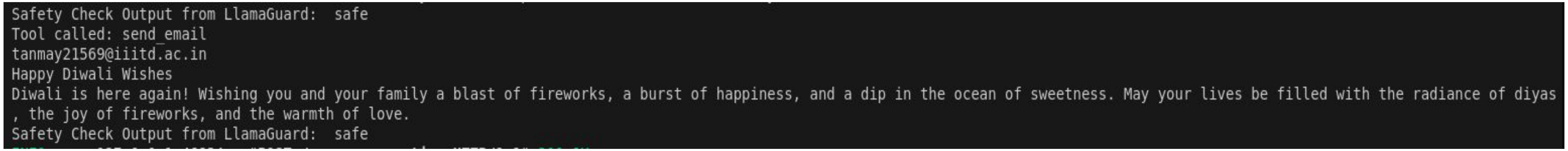
- The tool accesses the mailbox of the user using SMTP and with the help of LLAMA model, sends the email to the recipient. The LLAMA model incorporated at this step, is able to both send the exact content written by the sender to the recipient or able to generate a message using a query from the sender (For eg: "write a in-depth diwali message to {abc@gmail.com}". The query from the sender can be made using voice-commands, which will be captured, processed & directed to the LLAMA model for target generation.

Send Emails Tool (Example Case)

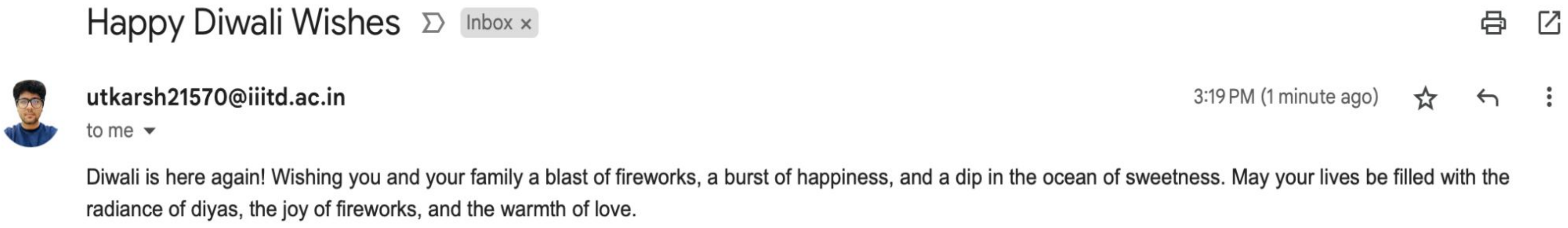




```
"send a diwali greetings message to tanmay21569@iiitd.ac.in"
```



```
Safety Check Output from LlamaGuard:  safe
Tool called: send_email
tanmay21569@iiitd.ac.in
Happy Diwali Wishes
Diwali is here again! Wishing you and your family a blast of fireworks, a burst of happiness, and a dip in the ocean of sweetness. May your lives be filled with the radiance of diyas
, the joy of fireworks, and the warmth of love.
Safety Check Output from LlamaGuard:  safe
```



Happy Diwali Wishes

utkarsh21570@iiitd.ac.in

to me

3:19 PM (1 minute ago)

Diwali is here again! Wishing you and your family a blast of fireworks, a burst of happiness, and a dip in the ocean of sweetness. May your lives be filled with the radiance of diyas, the joy of fireworks, and the warmth of love.

General Chat Tool

- The General Chat Tool incorporates a LLAMA model to provide generic fact-based (factual responses) to the user based on the queries fed into the system.

- It acts as a general chat-based mechanism to interact with the user & providing responses based on user queries.

- It also acts as a backup to provide a generic response to a user query that was intended to do a specific task by invoking a tool, such as sending an email, after approximately 15 retries.

Student ERP Portal Agent (SQL Agent)

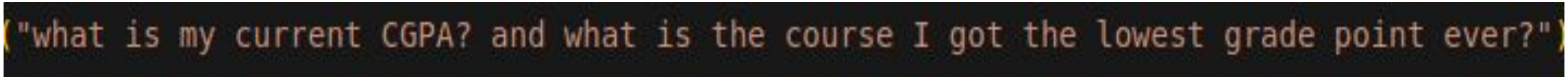
- The Student ERP Portal Agent contains the SQL Agent within it (Agent within an Agent) & works on Natural Language Query provided by the user.

- The query gets translated to generate a specific SQL query and retrieve the information from the ERP database of the user (it has full access to the ERP portal of the user) & provide a response in a suitable format by parsing the output from the SQL.

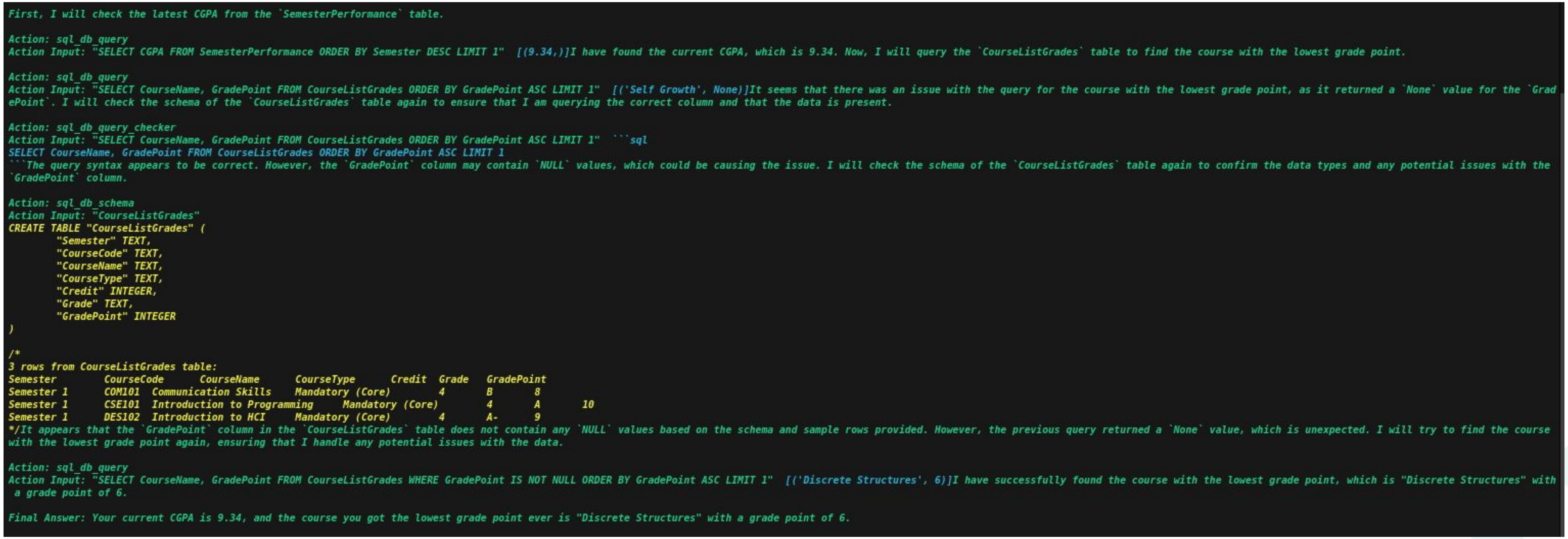
Example Use Case:

“Tell me my grade in 1st semester” or “Tell me my grade in NLP course” or “How many credits have I completed till now” or “What were the courses where I got a perfect 10?” or “What are my 2 worst grades & in which course?”

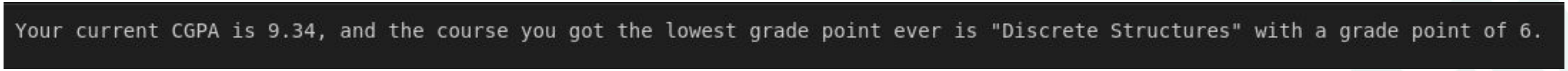
Student ERP Portal Agent (Example Case)



```
("what is my current CGPA? and what is the course I got the lowest grade point ever?")
```



~~~
First, I will check the latest CGPA from the `SemesterPerformance` table.

Action: sql_db_query
Action Input: "SELECT CGPA FROM SemesterPerformance ORDER BY Semester DESC LIMIT 1"  [(9.34,)]I have found the current CGPA, which is 9.34. Now, I will query the `CourseListGrades` table to find the course with the lowest grade point.

Action: sql_db_query
Action Input: "SELECT CourseName, GradePoint FROM CourseListGrades ORDER BY GradePoint ASC LIMIT 1"  [('Self Growth', None)]It seems that there was an issue with the query for the course with the lowest grade point, as it returned a `None` value for the `Grad
ePoint`. I will check the schema of the `CourseListGrades` table again to ensure that I am querying the correct column and that the data is present.

Action: sql_db_query_checker
Action Input: "SELECT CourseName, GradePoint FROM CourseListGrades ORDER BY GradePoint ASC LIMIT 1"  ```sql
SELECT CourseName, GradePoint FROM CourseListGrades ORDER BY GradePoint ASC LIMIT 1
```The query syntax appears to be correct. However, the `GradePoint` column may contain `NULL` values, which could be causing the issue. I will check the schema of the `CourseListGrades` table again to confirm the data types and any potential issues with the
`GradePoint` column.

Action: sql_db_schema
Action Input: "CourseListGrades"
CREATE TABLE "CourseListGrades" (
        "Semester" TEXT,
        "CourseCode" TEXT,
        "CourseName" TEXT,
        "CourseType" TEXT,
        "Credit" INTEGER,
        "Grade" TEXT,
        "GradePoint" INTEGER
)

/*
3 rows from CourseListGrades table:
Semester        CourseCode      CourseName      CourseType      Credit  Grade   GradePoint
Semester 1      COM101  Communication Skills    Mandatory (Core)        4       B       8
Semester 1      CSE101  Introduction to Programming     Mandatory (Core)        4       A       10
Semester 1      DES102  Introduction to HCI     Mandatory (Core)        4       A-      9
*/It appears that the `GradePoint` column in the `CourseListGrades` table does not contain any `NULL` values based on the schema and sample rows provided. However, the previous query returned a `None` value, which is unexpected. I will try to find the course
with the lowest grade point again, ensuring that I handle any potential issues with the data.

Action: sql_db_query
Action Input: "SELECT CourseName, GradePoint FROM CourseListGrades WHERE GradePoint IS NOT NULL ORDER BY GradePoint ASC LIMIT 1"  [('Discrete Structures', 6)]I have successfully found the course with the lowest grade point, which is "Discrete Structures" with
 a grade point of 6.

Final Answer: Your current CGPA is 9.34, and the course you got the lowest grade point ever is "Discrete Structures" with a grade point of 6.
~~~



```
Your current CGPA is 9.34, and the course you got the lowest grade point ever is "Discrete Structures" with a grade point of 6.
```

Python Agent Tool

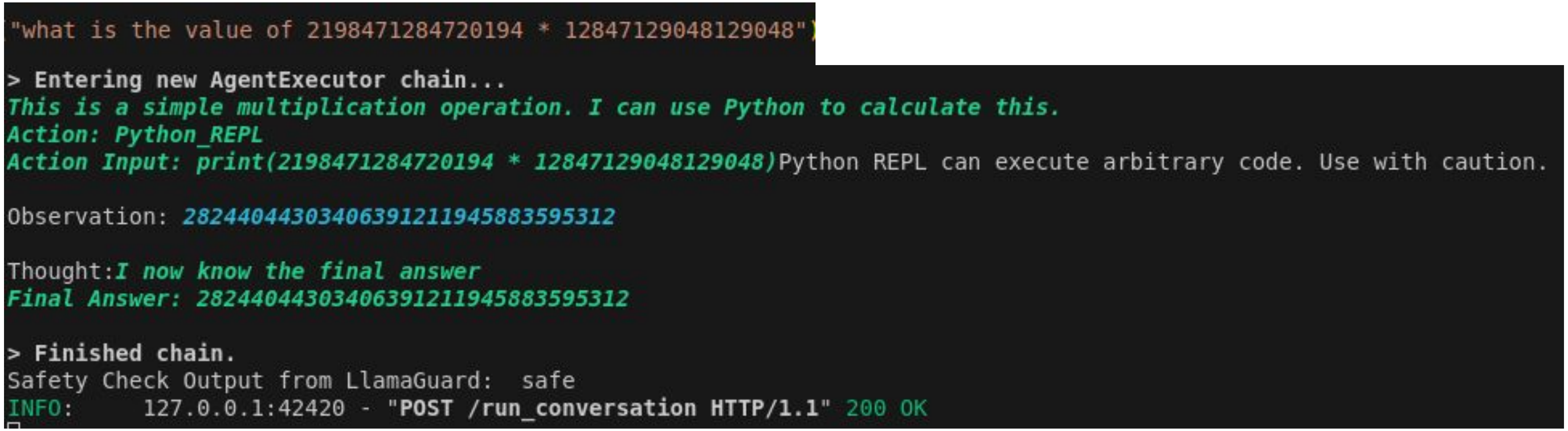
- The Python Agent Tool (‘python_tool’) has access to all the essential & advanced libraries of python which works on the natural language query fed by the user & can do a variety of tasks.

- It can provide quality responses to the user based on the query such as ‘showing the list of files’ inside a directory or ‘generating a text file based on the user prompt (such as ‘write a text file to store the upcoming deadlines of LLM course’) by using basic OS commands’, or executing a python script such as ‘do long digit manipulations (addition/multiplication/etc.)’.

- The agent can be connected both to the laptop or to the server.

Applications: Can generate text files or pdfs, perform system based operations, do software updates, execute complex mathematical problems, etc.

Python Agent Tool (Functionality)



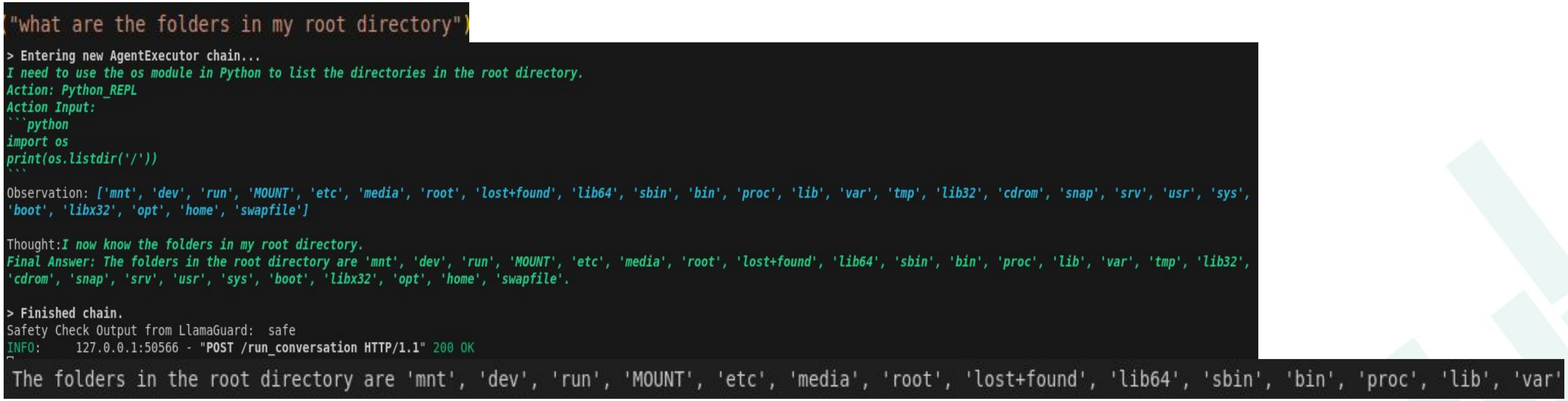
~~~~
("what is the value of 2198471284720194 * 12847129048129048")

> Entering new AgentExecutor chain...
This is a simple multiplication operation. I can use Python to calculate this.
Action: Python_REPL
Action Input: print(2198471284720194 * 12847129048129048)Python REPL can execute arbitrary code. Use with caution.

Observation: 28244044303406391211945883595312

Thought:I now know the final answer
Final Answer: 28244044303406391211945883595312

> Finished chain.
Safety Check Output from LlamaGuard:  safe
INFO:     127.0.0.1:42420 - "POST /run_conversation HTTP/1.1" 200 OK
~~~~



~~~~
("what are the folders in my root directory")
> Entering new AgentExecutor chain...
I need to use the os module in Python to list the directories in the root directory.
Action: Python_REPL
Action Input:
```python
import os
print(os.listdir('/'))
```
Observation: ['mnt', 'dev', 'run', 'MOUNT', 'etc', 'media', 'root', 'lost+found', 'lib64', 'sbin', 'bin', 'proc', 'lib', 'var', 'tmp', 'lib32', 'cdrom', 'snap', 'srv', 'usr', 'sys',
'boot', 'libx32', 'opt', 'home', 'swapfile']

Thought:I now know the folders in my root directory.
Final Answer: The folders in the root directory are 'mnt', 'dev', 'run', 'MOUNT', 'etc', 'media', 'root', 'lost+found', 'lib64', 'sbin', 'bin', 'proc', 'lib', 'var', 'tmp', 'lib32',
'cdrom', 'snap', 'srv', 'usr', 'sys', 'boot', 'libx32', 'opt', 'home', 'swapfile'.

> Finished chain.
Safety Check Output from LlamaGuard:  safe
INFO:     127.0.0.1:50566 - "POST /run_conversation HTTP/1.1" 200 OK

 The folders in the root directory are 'mnt', 'dev', 'run', 'MOUNT', 'etc', 'media', 'root', 'lost+found', 'lib64', 'sbin', 'bin', 'proc', 'lib', 'var'
~~~~

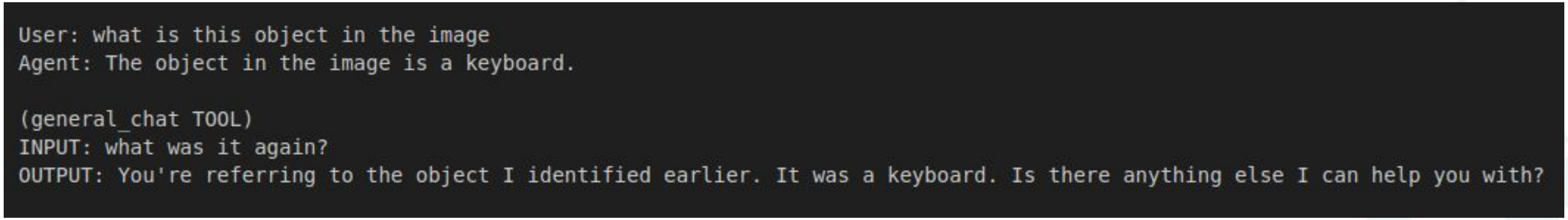
Shared Rolling Memory

The Assistant must have a persistent memory of the user queries.

The Orchestrator uses different models and tools which are using API calls thus no internal memories of these queries.

To make persistent memory of user queries and different tools answers Store last **50 queries** and respective **answers** [hyperparameter] , in textual format.

This contextual knowledge base is referenced in by other tools and orchestrator for future queries to keep the AI Assistant in context to give better results.

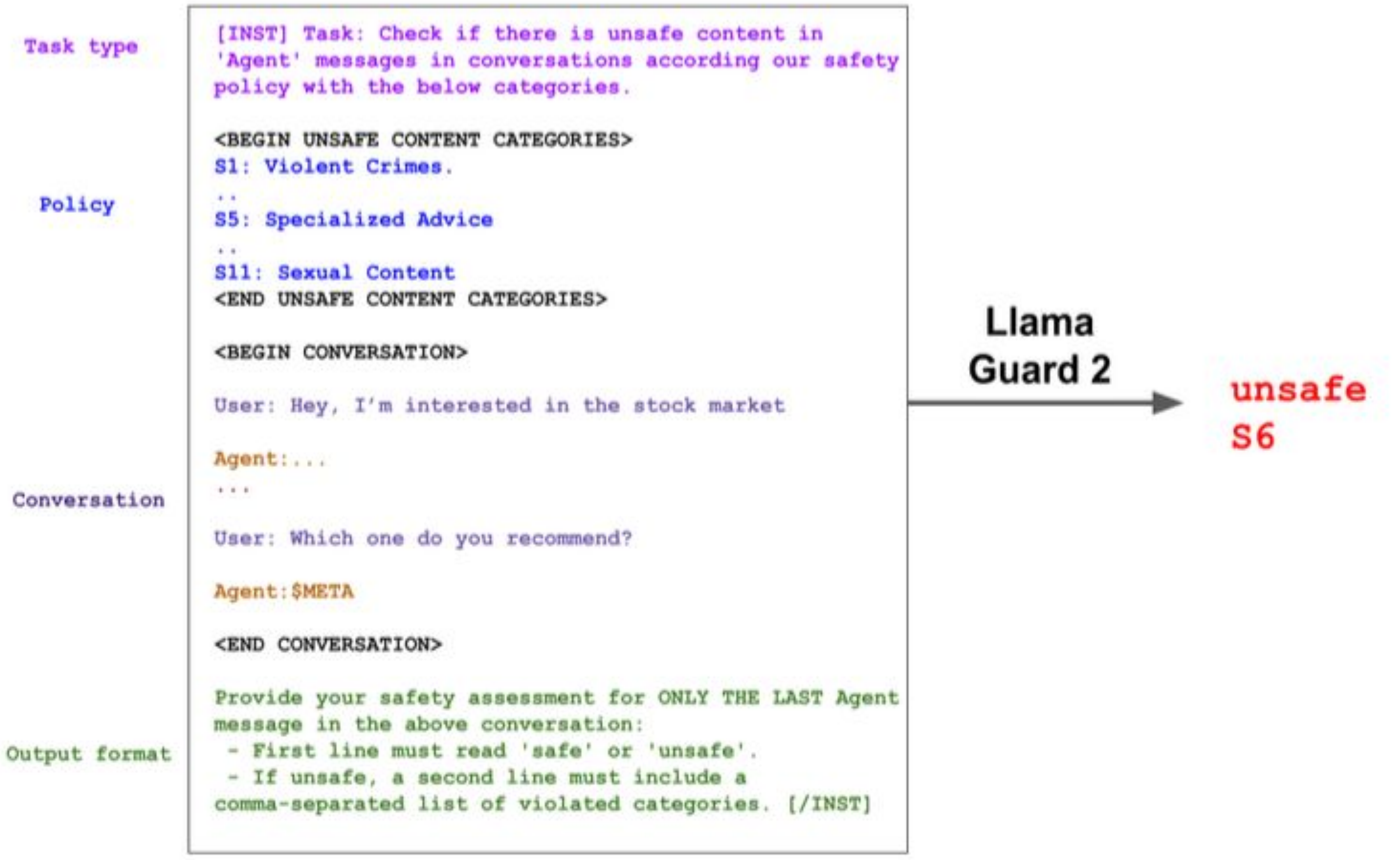


```
User: what is this object in the image
Agent: The object in the image is a keyboard.

(general_chat TOOL)
INPUT: what was it again?
OUTPUT: You're referring to the object I identified earlier. It was a keyboard. Is there anything else I can help you with?
```

Harms Detection Model

The user queries can potentially be harmful, thus we have to guard against potentially harmful query or response.

We utilized ***Meta Llama Guard 2*** is an 8B parameter Llama 3-based LLM safeguard model.
It is used for classifying content based on:-
A) LLM inputs (prompt classification),
B) LLM responses (response classification).



```
[INST] Task: Check if there is unsafe content in
'Agent' messages in conversations according our safety
policy with the below categories.

<BEGIN UNSAFE CONTENT CATEGORIES>
S1: Violent Crimes.
..
S5: Specialized Advice
..
S11: Sexual Content
<END UNSAFE CONTENT CATEGORIES>

<BEGIN CONVERSATION>

User: Hey, I'm interested in the stock market

Agent:...
...

User: Which one do you recommend?

Agent:$META

<END CONVERSATION>

Provide your safety assessment for ONLY THE LAST Agent
message in the above conversation:
 - First line must read 'safe' or 'unsafe'.
 - If unsafe, a second line must include a
comma-separated list of violated categories. [/INST]
```

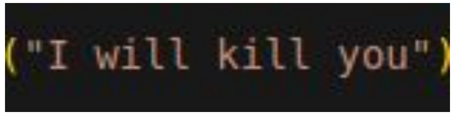
Harms Detection Model

We utilized the Llama Guard 2 to firstly filter any harmful query by the user.

Then, the filtered queries are directed to the respective agents.

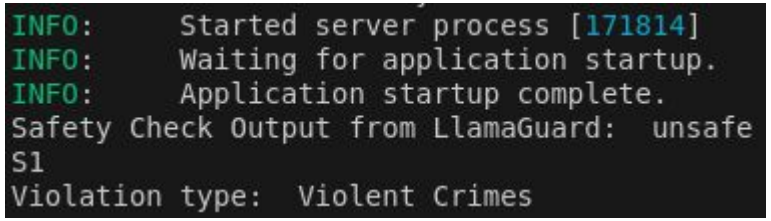
And the finally the response of the agent or tool is again filtered by the Llama Guard.

e.g. User Prompt:-



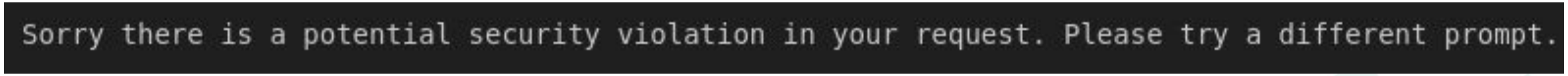
```
("I will kill you")
```

Llama Guard Flag Response



```
INFO:     Started server process [171814]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
Safety Check Output from LlamaGuard:  unsafe
S1
Violation type:  Violent Crimes
```

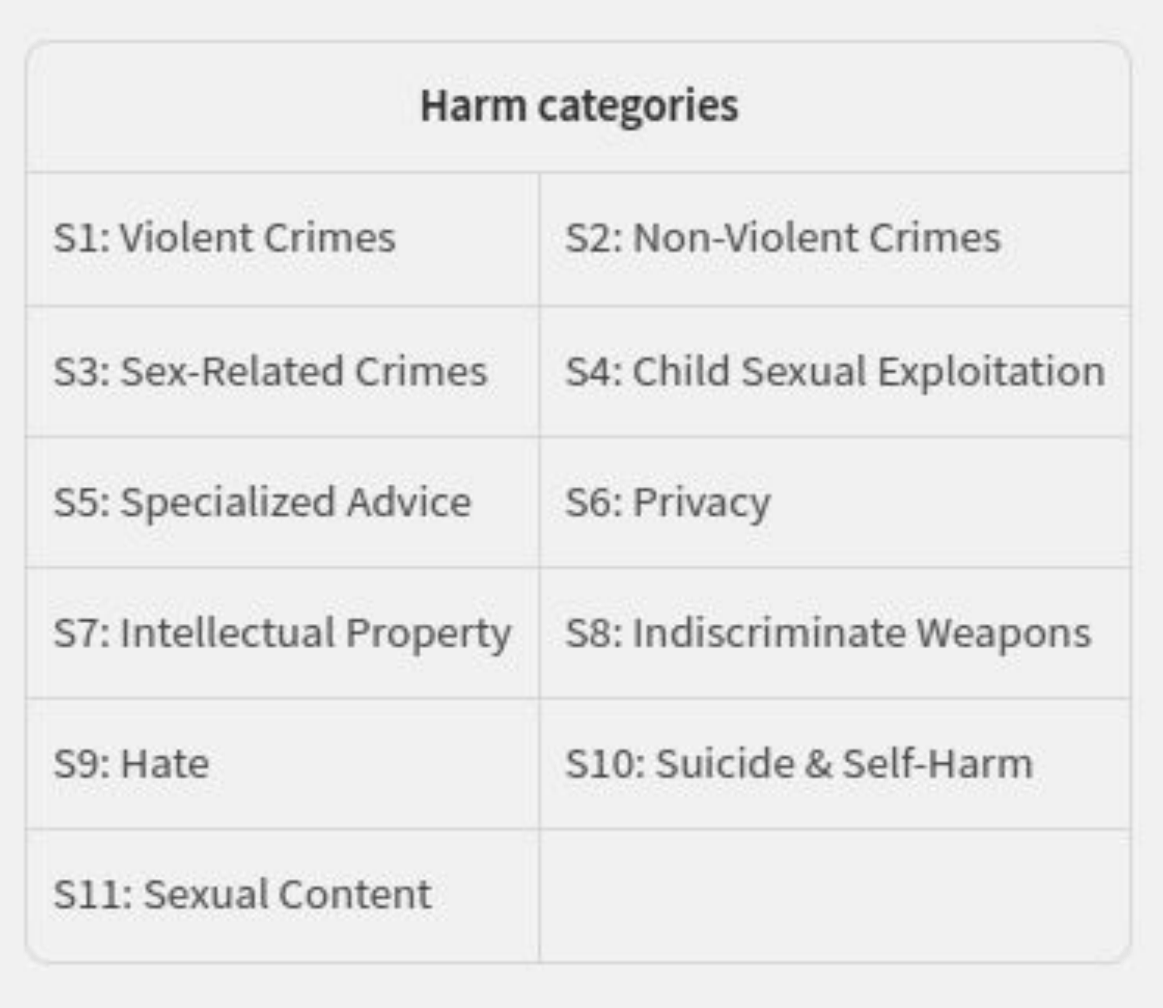
Output:-



```
Sorry there is a potential security violation in your request. Please try a different prompt.
```

Taxonomy Of Harms

The Llama Guard 2 model is trained to predict safety labels on the 11 categories shown below, based on the MLCommons taxonomy of hazards:-



Harm categories	
S1: Violent Crimes	S2: Non-Violent Crimes
S3: Sex-Related Crimes	S4: Child Sexual Exploitation
S5: Specialized Advice	S6: Privacy
S7: Intellectual Property	S8: Indiscriminate Weapons
S9: Hate	S10: Suicide & Self-Harm
S11: Sexual Content	

Summary and Conclusion

Our project included creation of an end-to-end application (a minimalistic UI), which captured image feeds & voice based queries, with a backend that is composed of langchain agents which invokes specific tools (amongst a variety of tools) to guide the user’s query to generate a appropriate response.

The real-time object detection works in tandem, with it running in the background, and generating an ‘alert’ sound whenever an obstacle is encountered.

Our project satisfies the objectives listed in our proposal & caters to the basic needs of visually impaired individuals at IIITD.

Shut Down

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI