

# GSOC 2025 PROPOSAL

## OWASP Nest bot as an AI agent/assistant

### BASIC INFORMATION

Name	Dishant Miyani
Email	<a href="mailto:dishantmiyani1804@gmail.com">dishantmiyani1804@gmail.com</a>
GitHub	<a href="https://github.com/Dishant1804">Dishant1804</a>
LinkedIn	<a href="https://www.linkedin.com/in/dishantmiyani">dishantmiyani</a>
Contact	(+91) XXXXXXXXXX
TimeZone	Indian Standard Time (UTC +5:30)
Location	Gujarat, India

### PROJECT DETAILS

**Project Title :** OWASP Nest bot as an AI agent/assistant.

**Project size :** Large.

**Proposed Duration :** 350 Hours.

**Project Mentors :** Arkadii Yakovets, Kateryna Golovanova, Tamara Lazerka

### DELIVERABLES

The following are the deliverables for GSOC 2025 for this project :

- Develop a classifier model that will be trained on OWASP's data.
- New Nest bot with the AI assistance capabilities.
- Testing and production ready mechanism for the Nest bot AI assistant.
- Usage documentation of the Nest bot assistant.

- **Project Overview**

OWASP Slack channel has a Slack bot which is called **Nest bot**, Nest bot is responsible for giving out the answers of the command that we type in the message. Currently, Nest bot operates based on commands (e.g., `/contribute`, `/events`, `/chapter`, etc.). **Nest bot has pre-programmed commands**, and it can only respond to the data that is associated with each command.

However, in certain cases, users frequently ask questions that do not match any existing command. These might include:

- General questions about the OWASP Foundation.
- Specific questions about OWASP projects (like how to set them up or contribute).
- Requests for directions to the correct Slack channel for a particular topic.

When Nest bot cannot answer these queries, manual input/guidance is required, which may sometimes result in unsolved queries or repetitive answering of questions by the members of the organization.

I propose to solve this issue by upgrading the **Nest bot with AI capabilities to function as an intelligent assistant**. With the help of the OWASP's data, we can get accurate and context aware responses by enhancing the capability of **Large Language Models (LLMs)**. LLMs are AI models trained on vast amounts of text data, capable of understanding and generating human-like text. To ensure the LLM provides answers that are accurate and specific to OWASP, we will use a technique called **Retrieval-Augmented Generation (RAG)**.

With the help of AI Nest bot we will achieve the following results :

- The ability to answer frequently asked questions (FAQs) and general queries accurately about OWASP.
- Intelligently direct users to the most appropriate Slack channels or resources for their specific needs.

- **Few terminologies**

- What is RAG?

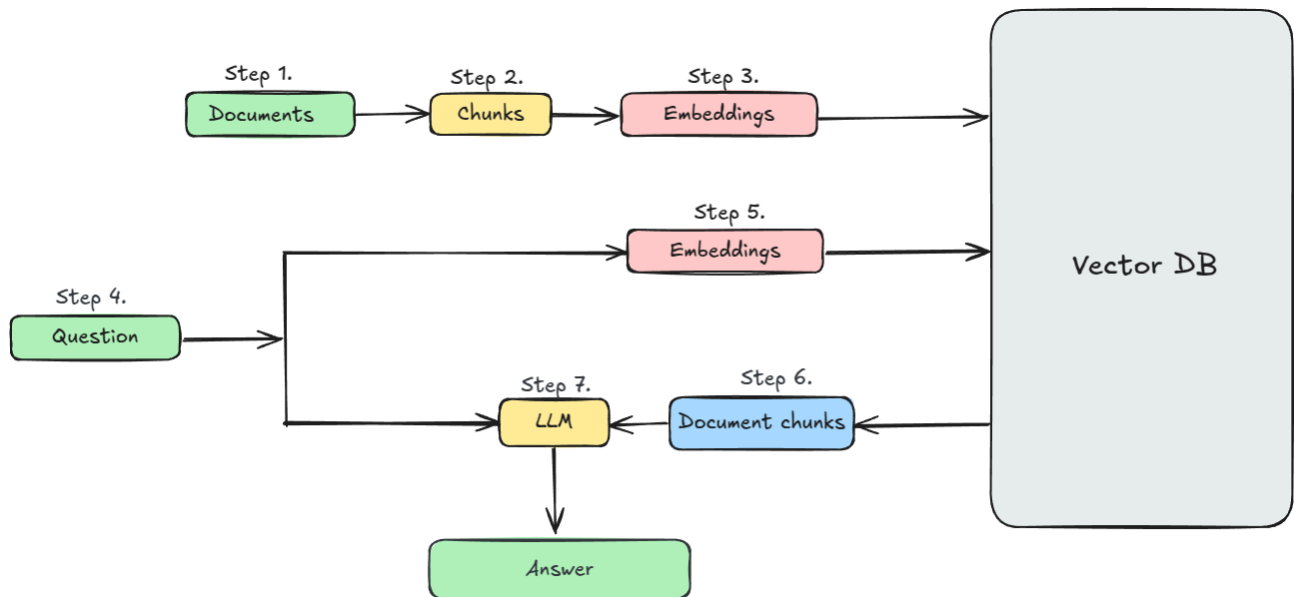
- A technique that enhances the LLM response by retrieving relevant information from an external knowledge base before generating a response, improving accuracy and context relevance.

- Why RAG?

- Without RAG, the LLM takes the user input and creates a response based on information it was trained on — or what it already knows. With RAG, an information retrieval component is introduced that utilizes the user input to first pull information from a data source. The user query and the relevant information are both given to the LLM. The LLM uses the new knowledge and its training data to create better responses.

- What are embeddings?


- It is a numerical vector representation of text that captures semantic meaning, which helps in efficient similarity search of text.



*Sample RAG workflow diagram*

## ● Execution Plan

- Community bonding period (May 8th - June 1st)
  - Introduction to the mentors and to the community
  - Discuss with mentors and community to see if any more improvements are required or not.
  - Interact with mentors to get my doubts/technical design decisions cleared if there are any left.
  - Since the community bonding period is almost for a month we will utilize the time and set up the requirements that will be needed to implement Nest bot accurately.
  - Fetching all the data related to OWASP, from Slack conversations and GitHub repositories of OWASP, by using some existing methods in the code base and creating new methods if required.
  - Creating a django model to store the vector embeddings of the fetched data.
  - This is the sample model for storing the fetched data and its embeddings:

 IndexedDocument
<ul style="list-style-type: none"><li>● id: BigAutoField() «PK»</li><li>● text_content: TextField()</li><li>● source_of_data: CharField(max_length=200)</li><li>● category: CharField(max_length=100)</li><li>● embedding: VectorField()</li></ul>

- After fetching and cleaning all the data, text\_content needs to be divided into chunks (small pieces) so that we will reduce the noise in data and create small and relevant portions of that data, overlapping the chunks will help to retain the context.
- Setting up commands for creating embeddings and chunks of the document. Overlapping of chunks will be performed with

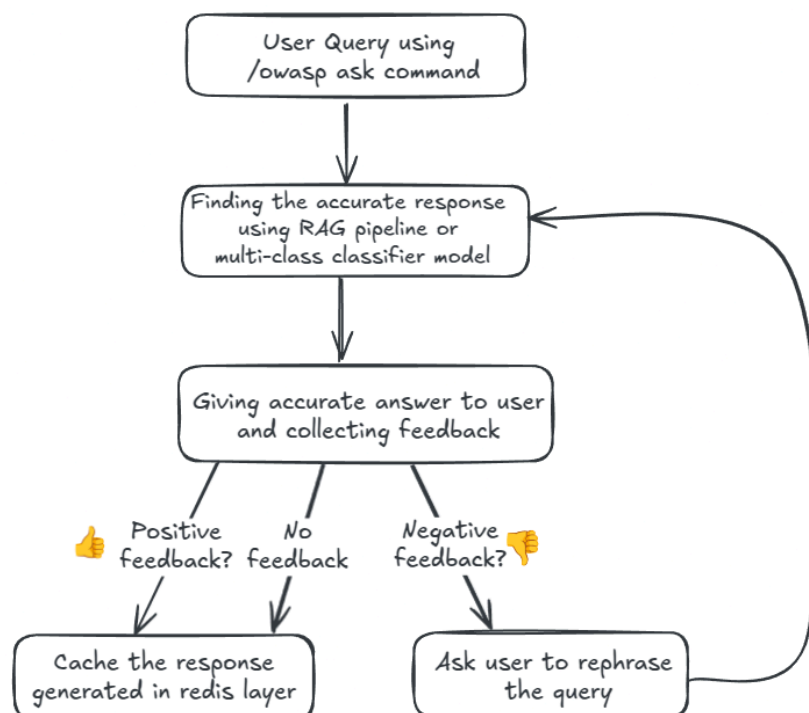
the help of a sliding window mechanism which will give us 10-15% overlapping near the boundaries of chunks.

- We will prepare the chunked data using the sliding window method and create its command and call the bulk update method to create its vector embeddings which is numerical representation of that data.
- Creating a new Slack channel for testing and a new command (e.g. `/owasp ask <query>`) to call the AI Nest bot and ask a query to the Nest bot by calling the command in Slack conversations.
- Setting up the initial Slack message flow in the new testing channel for the AI Nest Bot using Slack bolt API interactions.
- Implementing initial structure of message flow from user query to bot and then giving the acknowledgement to the user followed by the response that will be generated by the query processing pipeline.

○ Week 1 (June 2 - June 8)

- Setting up the Redis caching layer to cache the RAG + LLM generated response, this cached response can be used in future conversations when similar questions are asked.
- Redis is an open-source, in-memory data structure store that is used to cache the data.
- We will implement built-in semantic search mechanism of Redis, which can search the similarities between the embeddings of user's query and embeddings of previously cached query with the `distance_threshold` of 0.2 to 0.4, if there is a cache hit, we give the response instantly and if it is a cache miss then we go to the next steps to answer the query.
- By allowing the `distance_threshold` between 0.2 to 0.4, we allow for slightly more flexible matching (lower the `distance_threshold` stricter is the semantic search).

- Storing the count of positive and feedback of the response in the caching layer. This will help us to discard the cached response after the particular response reaches a certain number of negative feedback.
  - This approach will act very powerful when it comes to answering frequently asked questions (FAQs), because it gives fast responses, reduces the number of API calls, and reduces the cost.
- Week 2 (June 9 - June 15)
- Setting up an interactive feedback mechanism for user to give feedback to the generated response by clicking on 'helpful 👍' or 'not helpful 👎' button. If the user is not satisfied with the answer, we will ask to rephrase the query for better output.
  - Creating a method to fetch the positive cached responses and store them in the database.
  - We will set up an automated job for fetching and storing the cached responses that will run weekly or bi-weekly as per preference.
  - This ensures the caching layer does not get bloated and gives out a quick response.
  - This is the basic sample of message flow diagram:



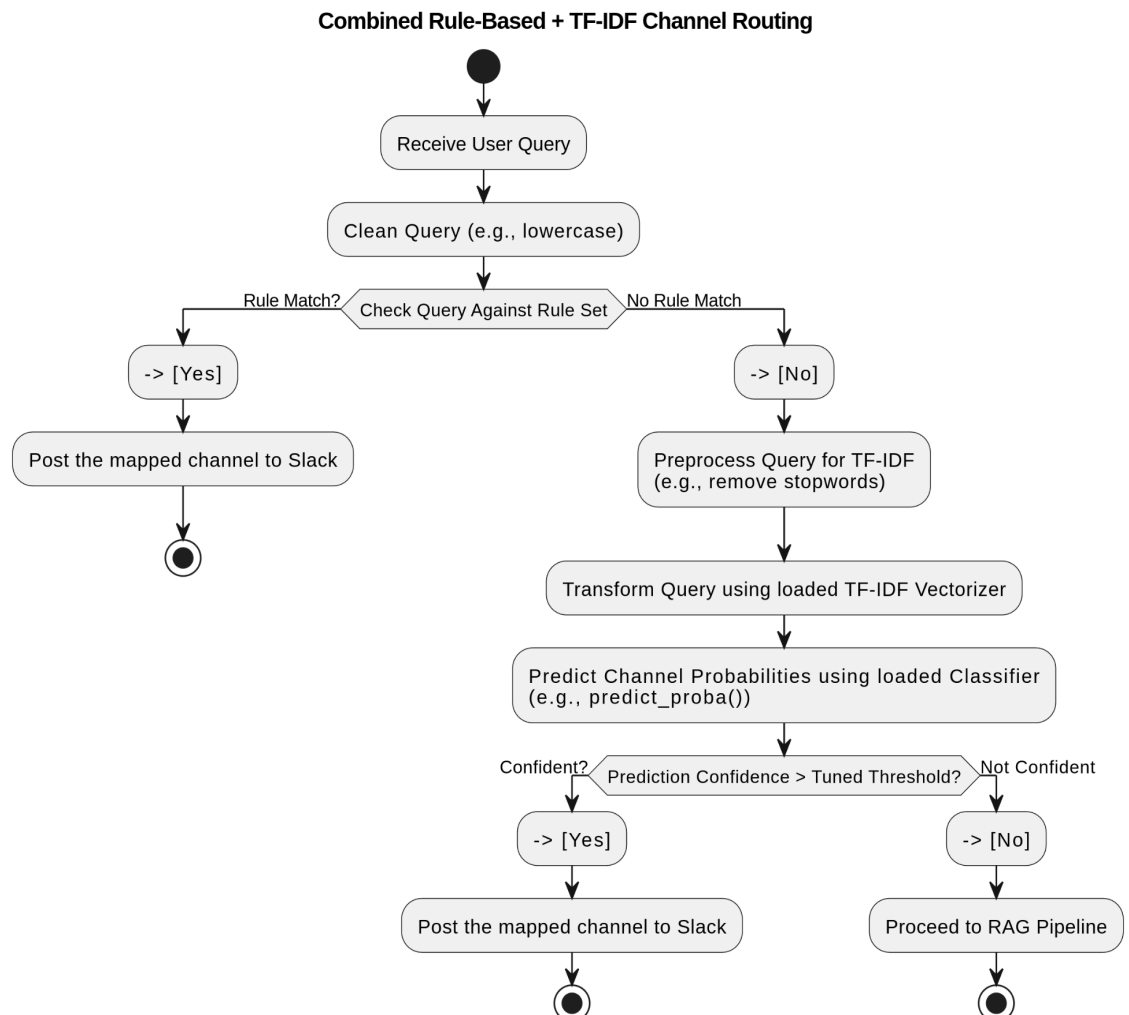
○ Week 3 (June 16 - June 22)

- If the query is related to a specific channel we will direct the user to a specific OWASP Slack channel, by using a combination of **Rule Based approach** and **TF-IDF (Term frequency-Inverse document frequency)**.
- This will be a multi-layered approach, where the rule based approach will be the first layer. In this layer we will define a **rule set** which will include all the **major keywords, abbreviations and common phrases** (e.g. 'ASVS', 'AppSec standard') along with the **synonyms and multi-word variants** (e.g. 'OWASP ZAP', 'OWASP Zed attack proxy'), this will help us to identify which channel to direct the user to, very easily.
- To handle cases where multiple keywords might match, a **Longest Match First strategy** will be implemented, favoring the **longest matching keyword** to capture the most specific intent (e.g. 'Juice shop' will be more prioritized over 'shop')
- If the first layer has ambiguity due to multiple similar matches, we move towards more refinement. The second layer is the **combination of TF-IDF and a multi-class classifier**.
- TF-IDF is a statistical measure used to evaluate how important a word is to a document (or message) in a collection. Words frequent in a specific channel's messages but rare across all channels receive higher TF-IDF scores for that channel, indicating they are characteristic keywords.
- The data that we have fetched from the OWASP Slack channels and OWASP repositories is not clean enough to train our machine learning model.
- We will clean the data by removing the punctuations and stopwords (words that occur very frequently and do not hold meaning for data, e.g. 'where', 'the', 'is', 'was', etc.) This filter helps us to keep the meaningful data.

- We will split the cleaned data into a train/test set and fit the `TfidfVectorizer` on training only and transform training data.
  - We will use Scikit-learn's `TfidfVectorizer` which will calculate the TF-IDF score for each term in the vocabulary for each training message.
  - This transforms the text data into a matrix of numerical feature vectors, where each vector represents a message based on the importance of its constituent words.
- Week 4 (June 23 - June 29)
    - The model we are going to use here will be developed with the **Scikit-learn library** (a popular open source library to build ML models). We will train the classifiers like `Multinomial Logistic Regression` or `Linear Support Vector Machine` or `Multinomial Naive Bayes` on the training TF-IDF vectors. The one which gets us the best accuracy will be used here.
    - This **A/B testing of classifiers** will help us choose the best and accurate classifier for our needs.
    - The evaluation of the model will be done on the metrics like **accuracy\_score**, **F1 score**, **precision recall**. This will help us to identify which model is performing well.
    - The fitted `TfidfVectorizer` object and the trained `classifier` object will be saved in `.pkl` format.
  - Week 5 (June 30 - July 6)
    - We will combine the above two layers here and create a structural path for the query processing based on confidence thresholds.
    - The incoming user query will undergo cleaning and preprocessing which is same as that of used during the TF-IDF training with the help of fitted `TfidfVectorizer` object.



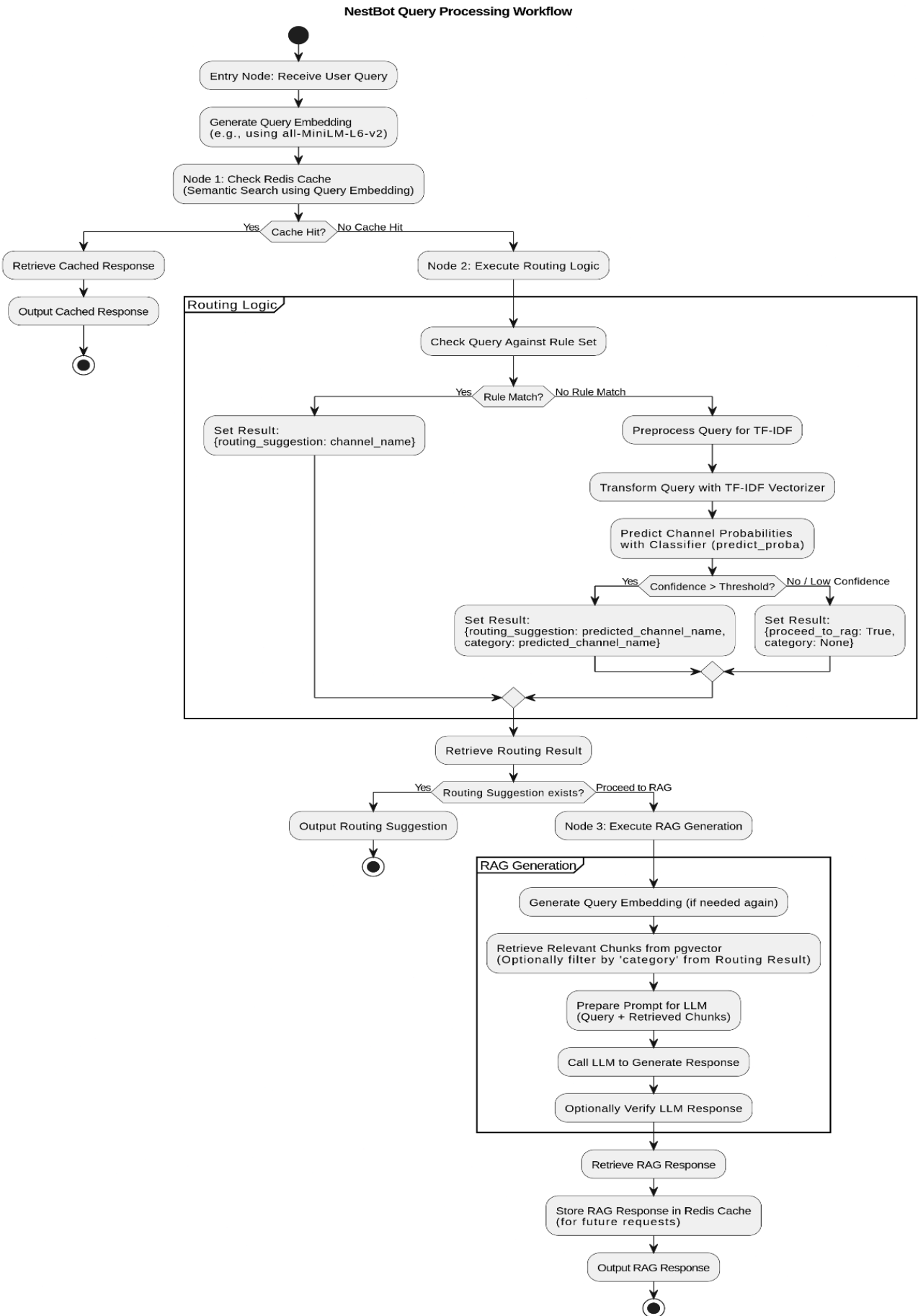
- The loaded `TfidfVectorizer` will transform the cleaned query into its TF-IDF feature vector.
- The trained `classifier` object will then use the `predict_proba()` method on this vector to get the probability distribution across all target channels.
- The channel with the highest probability is identified as the prediction.
- We will only suggest the channel predicted by the TF-IDF classifier if its probability (confidence score) exceeds a set threshold (e.g., 0.9 - to be tuned during testing). This prevents suggesting channels based on weak statistical signals.
- If a prediction is made above the confidence threshold, we will suggest the predicted channel. Otherwise, we will pass the query to the RAG pipeline as this query can be a general query which is not related to any of the projects.



○ Week 6 (July 7 - July 13)

- We will create a **Retrieval-Augmented Generation (RAG) pipeline** we will create embeddings of user's query to search for relevant data in the embeddings stored in database, It retrieves the corresponding `text_content` from the database and finally, a Large Language Model (LLM) constructs the answer using the retrieved context.
- The RAG pipeline we will be implementing here is a **combination of LangChain and LangGraph**. LangChain is a framework for building linear execution chains, while LangGraph, a part of LangChain, allows us to define **graph workflows** instead of simple linear chains.
- LangGraph lets us define and build a graph workflow of query processing with the help of conditional nodes and edges. We define start nodes, conditional nodes along with tool nodes if required and end nodes in our graph workflow.
- The LangGraph node will call the **Redis semantic search** method and look whether there is any previous relevant response to this query, if there is the cached response for a similar query it is a cache hit and we will return the response.
- In case of cache miss we will move to the next node, this is the node where we will call the classifier model which was trained and saved earlier, it will have a conditional node which will return the user the Slack channel if the confidence of response is more than the defined threshold, if it does not satisfy the threshold we get to know that it is a general query then we will pass the query to RAG response generation node.
- This RAG node of the LangGraph will give us the generated response by taking the embeddings of the user's query and we will use this to find the relevant information from the database.
- After getting the relevant context from the database we will give the context along with the query to the OpenAI-LLM model to generate the response that will solve the user's query.

Sample workflow diagram of all the 3 LangGraph nodes combined



- Mid Evaluations (July 14 - July 18)
  - Delivering the integrated development prototype which has caching, routing and RAG workflow capabilities.
  - Documenting the working and implementation of the trained ML model.
  - Refining the approach if needed and discussing the further implementations of the project.
- Week 7 (July 19 - July 25)
  - We will **verify the response generated** by the OpenAI-LLM and if the response is verified upon comparing it with the retrieved `text_content` from the database and give the generated response to the user.
  - If the answer is not relevant, we send it back to the LLM for refinement based on the data source. This process ensures that the **final answer provided to the user is accurate, verified, and specific to OWASP**.
  - By writing a **proper prompt** to the LLM by **creating a proper prompt template**, this data validation layer will also make sure there are **no hallucinations** in response or any kind of informal answer given to the end user.
  - Performing **A/B testing different RAG LLM prompt strategies** and **retrieval parameters** on the pipeline to try and find a more accurate and suitable approach for our needs.
- Week 8 (July 26 - Aug 2)
  - We will connect the new Slack channel that was created earlier with the whole RAG pipeline.
  - We will map the command to call the RAG pipeline if any user asks the query using `/owasp ask <query>` command.
  - This **pilot testing phase** will help us understand how the RAG pipeline behaves when the user asks the query.
  - We will store the **summary of the new conversation** as text and divide the summary into smaller pieces (chunks) and create embeddings of the conversations in the database to train model.

○ Week 9 (Aug 3 - Aug 10)

- Writing the **tests for all the methods and the commands** that are defined and implemented above.
- Writing the tests for the Slack channel and Slack command to ensure that the Slack events are called properly.
- Implementing the testing mechanism for the RAG pipeline.
- Using **LangChain evaluators** we will **test the entire workflow** and ensure that the expected results are produced by pipeline.
- Testing the queries that are out of the scope for OWASP and asserting that the RAG pipeline **handles the response gracefully**.

○ Week 10 (Aug 11 - Aug 17)

- Implementing the **logging, monitoring, and performance tracking of the RAG pipeline** with the help of **Signoz tool**.
- Signoz is an open-source tool which is used to get the logs, metrics, and performance details of the RAG pipeline. We will self-host it by creating a docker file for it. It also gives us all the metrics via a **built-in dashboard** so that we will **track all the performance details**.
- With the help of Signoz, I plan to track the metrics such as **pipeline latency, cache hit rate, LLM response time**, these metrics will help us in altering the thresholds and do slight modifications to improve performance.
- **Rolling out the new Nest bot in a controlled environment**, and gathering feedback from the community members and mentors about the usability and accuracy of the response.

○ Week 11 (Aug 18 - Aug 25)

- After taking all the feedback we will **implement the feedback** and test it.
- Implementing a **rollback mechanism** so that we can move to the previous versions of the new Nest bot.

- Gradually we will **roll out the new Nest bot in production** by first implementing it in a few owasp-specific channels.
  - We will closely **monitor all the metrics** and ensure everything is working fine.
  - We will prepare the **continuous integration pipeline** for the new Nest bot. This will help us every time we make some changes and deploy the Nest bot again during the testing as well as during production.
- Week 12 (Aug 26 - Sep 1)
    - Making the **final changes** to the new Nest bot if required or changes suggested by the mentor.
    - Finally **documenting the usage instructions** of the new Nest bot we will roll out in owasp-community channel.
  - Final Evaluation (Sep 2 - Sep 8)
    - Submitting all the **deliverables along with the documentation** of the project.
    - Project evaluation of the deliverables by mentors.

## ● Estimated project timeline

TIME PERIOD	TASKS
<b>Community bonding period</b> [May 8th - June 1st]	<ul style="list-style-type: none"> <li>- Introducing myself to the community and mentors.</li> <li>- Discussing the technical plan of the project with mentors in depth.</li> <li>- Fetching the OWASP specific data from Slack conversations and GitHub repositories.</li> <li>- Creating model and storing the data required for processing and answering user's query</li> <li>- Creation of a separate Slack channel and command for testing.</li> </ul>

<p><b>Week 1</b> [June 2 - June 8] <b>Redis Caching &amp; Semantic Search Setup</b></p>	<ul style="list-style-type: none"> <li>- Setting up Redis caching with semantic search to store and retrieve RAG + LLM responses based on query similarity using a distance threshold</li> <li>- Tracking feedback in cache to discard low-quality responses and optimize FAQ handling by reducing API calls and latency.</li> </ul>
<p><b>Week 2</b> [June 9 - June 15] <b>Feedback mechanism</b></p>	<ul style="list-style-type: none"> <li>- Implementing an interactive feedback mechanism via Slack interactions to get feedback on generated responses.</li> <li>- Creating automated jobs to fetch and store the cached responses.</li> </ul>
<p><b>Week 3</b> [June 16 - June 22] <b>Query Mapping &amp; TF-IDF Vectorization</b></p>	<ul style="list-style-type: none"> <li>- Implementing rule-based keyword mapping to route queries using keywords, synonyms, and prioritization of longer matches.</li> <li>- Cleaning the data and applying TF-IDF to convert messages into numerical vectors for training a classifier.</li> </ul>
<p><b>Week 4</b> [June 23 - June 29] <b>Classifier Training &amp; Evaluation</b></p>	<ul style="list-style-type: none"> <li>- We will train multiple classifiers (Logistic Regression, SVM, Multinomial Naive Bayes) using Scikit-learn on TF-IDF vectors and select the best through A/B testing.</li> <li>- Model evaluation will be done using accuracy, F1, precision, and recall, and the best-performing classifier along with the TF-IDF vectorizer will be saved.</li> </ul>
<p><b>Week 5</b> [June 30 - July 6] <b>Confidence-Based Query Routing</b></p>	<ul style="list-style-type: none"> <li>- We will integrate rule-based and TF-IDF layers into a structured query pipeline using a confidence threshold for predictions.</li> <li>- Queries below the threshold will be routed to the RAG pipeline, ensuring only high-confidence predictions are used for channel suggestions.</li> </ul>
<p><b>Week 6</b> [July 7 - July 13] <b>RAG Pipeline Implementation (LangChain/LangGraph)</b></p>	<ul style="list-style-type: none"> <li>- We will build a RAG pipeline using LangChain and LangGraph, integrating semantic search, classifier, and LLM response generation into a structured workflow.</li> <li>- The workflow will route queries through cache, classifier, or RAG nodes, using embeddings to retrieve context and generate accurate response.</li> </ul>
<p><b>Mid-Evaluation</b> [July 14 - July 18]</p>	<ul style="list-style-type: none"> <li>- Review progress with mentors and receive feedback.</li> <li>- Address any necessary adjustments or refinements to the implemented model or modifications in approach.</li> </ul>

<p><b>Week 7</b> [July 19 - July 25] <b>Validation and testing of workflow</b></p>	<ul style="list-style-type: none"> <li>- Adding a response validation and response filtering node to the graph workflow.</li> <li>- A/B testing of the pipeline to find a suitable approach to achieve expected results.</li> </ul>
<p><b>Week 8</b> [July 26 - Aug 1] <b>Slack Integration with RAG, Pilot Testing &amp; Data Collection</b></p>	<ul style="list-style-type: none"> <li>- Connecting the Slack channel and mapping the /owasp ask &lt;query&gt; command to the RAG pipeline.</li> <li>- Rolling out pipeline for initial pilot testing.</li> <li>- Fetching and summarizing new Slack conversations and storing them for future model training.</li> </ul>
<p><b>Week 9</b> [Aug 1 - Aug 10] <b>Testing of commands and pipeline</b></p>	<ul style="list-style-type: none"> <li>- Writing the tests for all the new commands/methods implemented in the project to avoid errors and unexpected results.</li> <li>- Testing out the RAG response with LangChain evaluators and handling out of scope queries gracefully.</li> </ul>
<p><b>Week 10</b> [Aug 11 - Aug 17] <b>Logging, monitoring and performance tracking</b></p>	<ul style="list-style-type: none"> <li>- Implementing the logging of errors, monitoring the resources and tracking the performance of the response generated via Signoz tool.</li> <li>- Rolling out a new Nest bot in a controlled environment and gathering feedback from users and mentors.</li> </ul>
<p><b>Week 11</b> [Aug 18 - Aug 25] <b>Pre-Launch and monitoring</b></p>	<ul style="list-style-type: none"> <li>- Implementing the feedback given by the users and mentors</li> <li>- Gradually rolling out the new AI Nest bot for owasp-specific channels.</li> <li>- Tracking and monitoring the bot more closely.</li> <li>- Implementing CI (continuous integration) pipeline for the new Nest bot.</li> </ul>
<p><b>Week 12</b> [Aug 26 - Sep 1] <b>Final changes and documentation</b></p>	<ul style="list-style-type: none"> <li>- Making final changes if required or suggested by a mentor.</li> <li>- Documenting the usage instructions of the new Nest bot and launching it in the OWASP-community Slack channel.</li> </ul>
<p><b>Final Evaluation</b> [Sep 2 - Sep 8]</p>	<ul style="list-style-type: none"> <li>- Delivering all the mentioned deliverables for evaluation</li> <li>- Ensure that the codebase is well documented, structured and implementations are kept very clear</li> </ul>



## ABOUT ME

I am a pre-final year student pursuing a degree (Bachelor of Technology in Information Technology) at Parul University, India. I developed a significant interest in programming and algorithms during my first year of college.

I am always interested in learning and exploring the depths of programming. I have built various projects such as the multiple-disease-classifier which is an ML-based classifier used to detect various chronic diseases such as kidney-stone, brain tumor, and pneumonia. I have also worked with pre-trained LLMs and developed RAG-based chatbots to chat with document's information using LangChain.

I was interested in Web Development, Artificial Intelligence and Machine Learning and I found OWASP Organization and Nest project interesting as it will be a great opportunity to learn more and implement technology and be a part of an open-source project.

## MOTIVATION

I was in my second year of college when I was introduced to the world of open-source. I was also interested in implementing the learnings and gaining more knowledge and experience of real-world projects via making contributions. I always believe in giving back to the community as much as I can. I found OWASP Nest Project and joined their Slack channel.

For the initial information about the Nest project, I tried to interact with the Nest bot, and I also got answers to a few questions via commands, but I could not get the answers to general questions as well as project related questions from Nest bot. I then went through the list of channels that were in OWASP organization's Slack channel and found project-nest channel. Finally, I asked my doubts about the project and started contributing to the project.

This motivated me to extend the functionality of the Nest bot and add automation to it using AI, as it is a way to reduce the manual efforts of answering the repeated questions asked by new contributors and new members within the OWASP organization.

## BASIC APPROACH TOWARDS PROBLEM

The first basic approach to solve the problem of answering the common questions was by implementing some automation in the Nest bot. I had previously worked with basic RAG chatbots, so the first approach that came to my mind was to retrieve all the data of OWASP and generate the context-aware response to the user's query via Nest bot and give it to the user.

## EXPECTATIONS FROM THE PROJECT

Working on a real-world project will be a great experience for me, I will learn a lot during the implementation of the entire project. I will get to know what are the things that are to be taken care off while working on the real-world project and also implement the best coding practices. Interacting with the OWASP community and implementing their feedback will help me grow both as a developer and as an open-source contributor. OWASP as a community is very supportive, So I just want to gain more hands-on experience of working on the real-world project and learn a lot from this project.

## PAST CONTRIBUTIONS

I have worked on a variety of issues and gained decent experience within OWASP Nest. Here's the list of some of the contributions I have made

- **Merged PR's**

- [#631](#) - Project details card migration to ChakraUI.
- [#712](#) - Refactor /gsoc command to fetch data from algolia index
- [#714](#) - Add audience attribute to project schema
- [#715](#) - Migrating buttons, tooltips and anchor tags to ChakraUI
- [#770](#) - Multi-stage docker build
- [#942](#) - Design and implement OWASP committee schema
- [#1011](#) - Extraction of common properties from schema files
- [#1043](#) - Implement owasp.org post previews
- [#1197](#) - About page for OWASP Nest
- [#1269](#) - Docker base image migration

- **Reviewed PR's**

- [#611](#) - Implementation of skeleton from ChakraUI
- [#663](#) - Shifted modal to ChakraUI dialog box
- [#780](#) - Update card layout
- [#1069](#) - Smooth animation for expanding/collapsing repositories list

## COMMITMENTS

- During the entire summer period I am available and there are no clashes and no other commitments during the summer break. I will manage to devote 30 - 35 hours a week to the project, which will be a minimum of 360 hours.
- I do not plan on taking up any internships or other activities this summer, as I want to keep myself available for this project. I will be having my college exams from July 12 to July 16.
- I will manage that time by working extra during the holidays and weekends for 6-8 hours a day to complete the project.
- I will complete the work well before the proposed deadline of each week to ensure that there is no lag with the proposed project timeline.
- I will give the mentors weekly updates about the project, I can alter the frequency depending on the mentor's preference.

## EXTENDED TIMELINE

- In the event of unforeseen obstacles, I am prepared to allocate additional time to the project in subsequent weeks.
- My priority is to deliver a robust and complete solution, and I am dedicated to seeing and implementing the project through to its fullest potential.
- I am willing to work in an extended time period in case the project is not completed, or any uncertainty occurs regarding the project.

## POST GSOC PLANS

I'm not applying for GSOC under any other organization other than this project in the OWASP organization. I am heart-warmed by the incredibly welcoming community and mentors I've found in the project-nest team, and I really want to be a part of it.

Post the GSOC period, I would love to continue contributing by adding more features to project-nest. I'll be part of a great OWASP-community and continue to contribute to the project and take it to the next level by implementing new features and improving the existing ones.

## REFERENCES

- [Redis semantic caching](#)
- [Pg-Vector package](#)
- [An Improved Text Sentiment Classification Model Using TF-IDF](#)
- [understanding RAG architecture and its working](#)
- [Slack api integration for AI interactions](#)
- [SigNoz self hosting](#)