EFFICIENT ONBOARDING TROUBLESHOOTING

AND

KNOWLEDGE MANAGEMENT

Team Members:

Omar Anan Abouromia

Mohamed Fahd

Mentors:

Eng. Mohammed El-Bastawesy

Eng. Mohamed Alrawy

Why we need Efficient Onboarding Troubleshooting and Knowledge Management webchat?

Problem Statement:

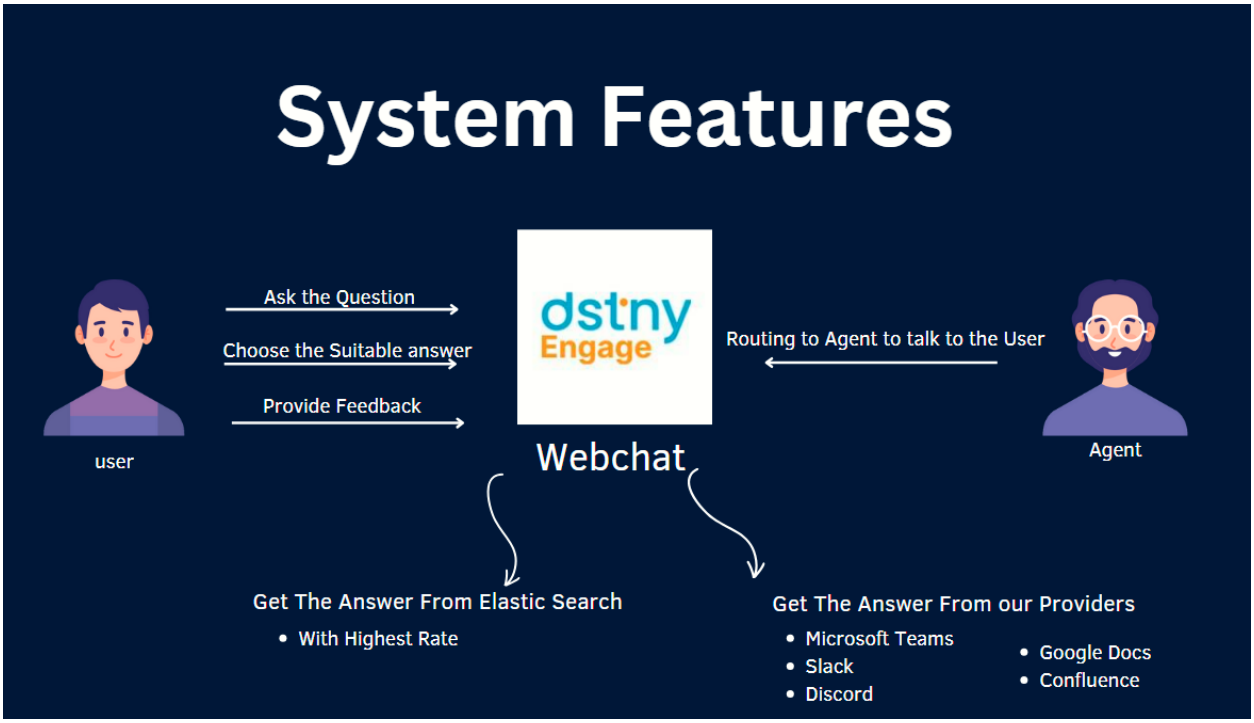
Inefficient Onboarding: It involves a considerable amount of time and resources to get new team members up to speed with our internal procedures, tools, and systems.

Knowledge Management: We are facing difficulties in effectively organizing, accessing, and updating the vast amount of knowledge within the organization. This includes information on best practices, guidelines, past project insights, and internal documentation.

Ineffective Communication: Communication and collaboration across multiple platforms lead to miscommunication and a lack of cohesive knowledge sharing.

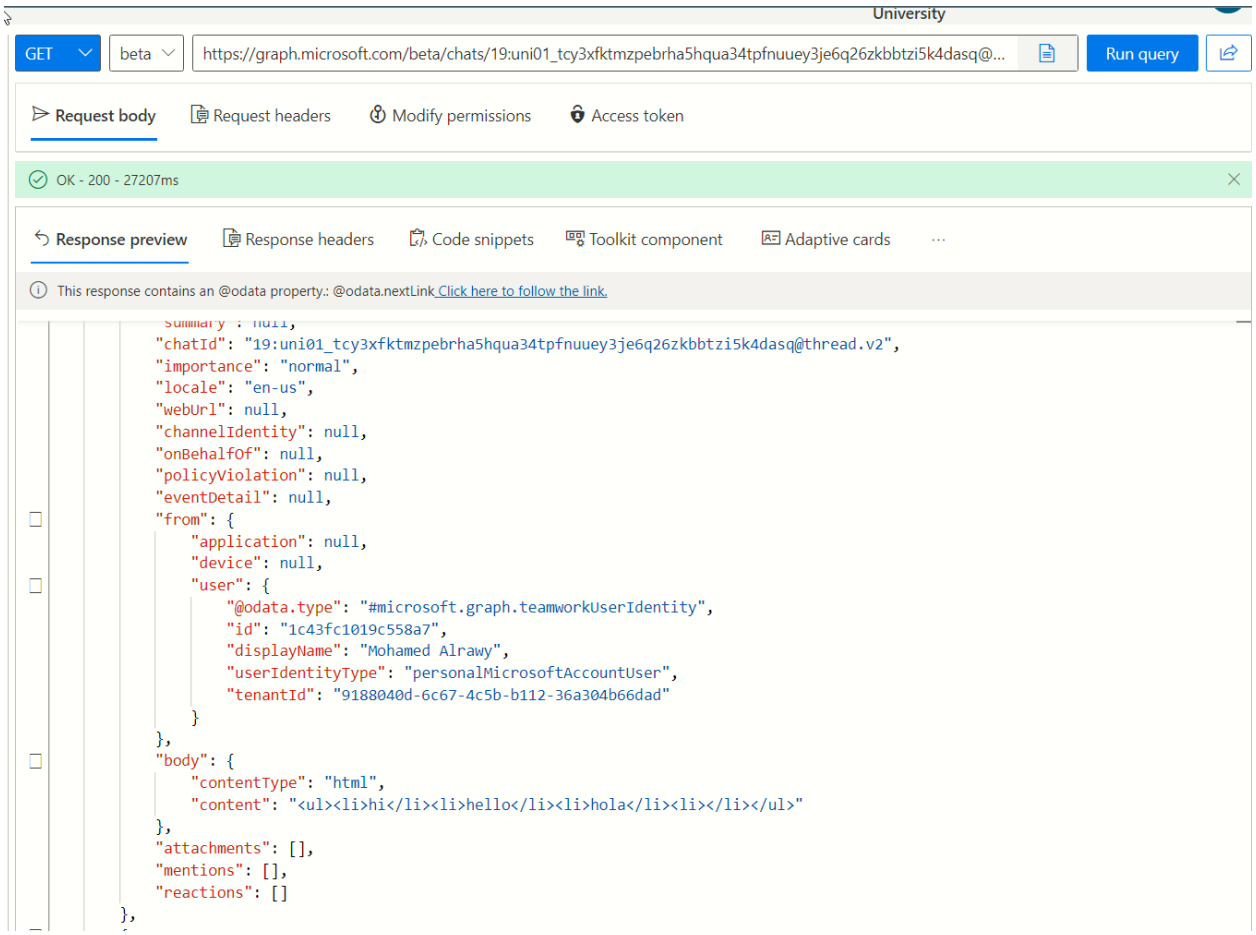
Solution:

We propose the development of a chatbot with the capability to retrieve answers from multiple sources, including an Elasticsearch database and the following providers: Slack, Microsoft Teams, Discord, Confluence, and Google Docs.



How to Integrate Microsoft Teams (as Chat)?

- Use **Microsoft Graph Explorer** with my access as normal user in organization.
- Use chat ID ➔ to GET data of chat between me and others by access each chat by its ID
- Add the access token provided in graph Microsoft and add it in the header.



```
const teamsFetchAllMessages = async () => {
  const chatEndpoint =
"https://graph.microsoft.com/beta/chats/19:uni01_tcy3xfktmzpebrha5hqua34tpfnuuey3je6q26zkbbtzi5k4dasq@thread.v2/messages"
;

  let allMessages = [];

  let pageUrl = chatEndpoint;
  while (pageUrl) {
    try {
      const response = await axios.get(pageUrl, {
        headers: {
          Authorization: `Bearer ${accessToken}`,
          "Content-Type": "application/json",
        },
      });
```

After Getting all the content of chat .. what's next ?

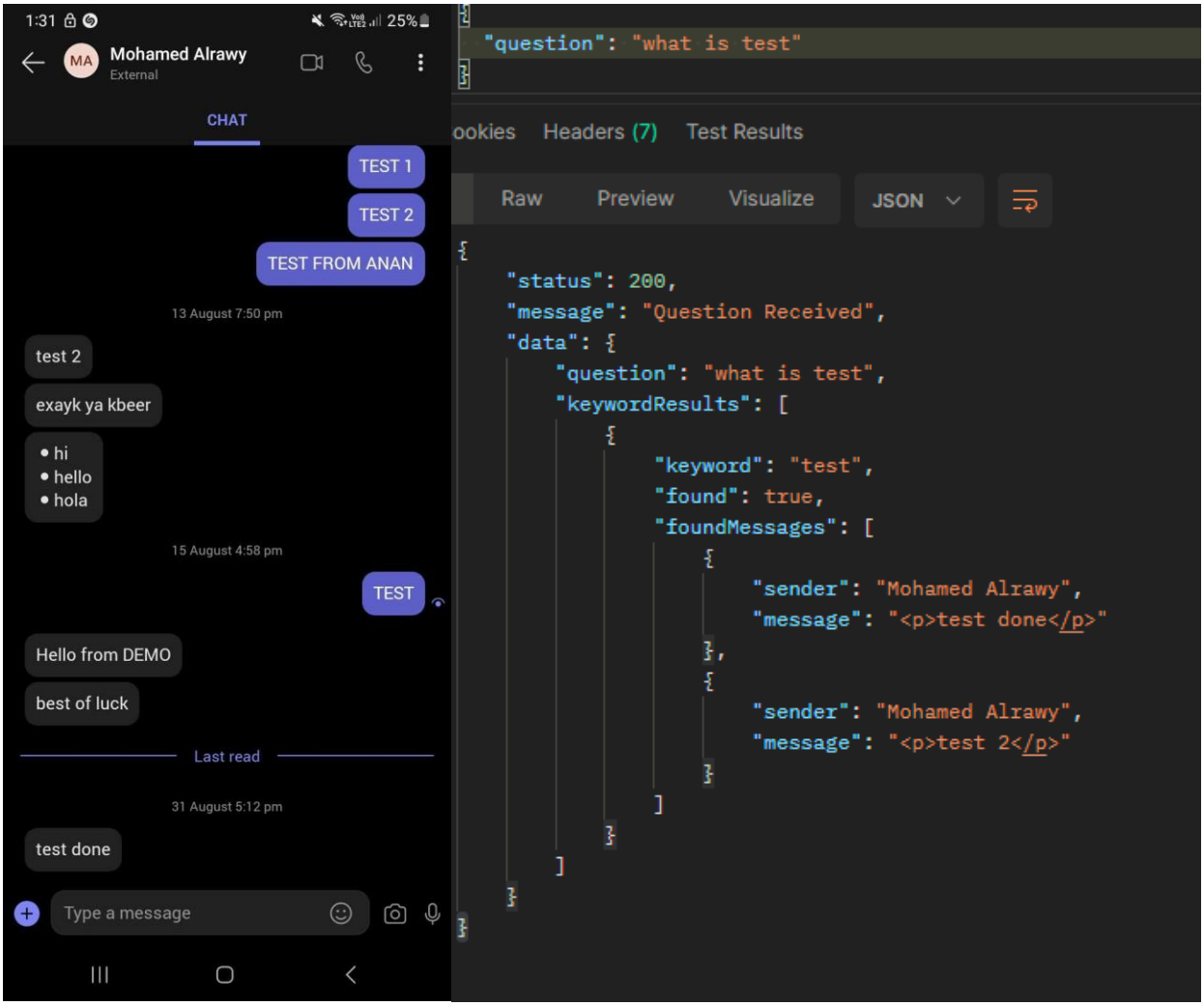
Start to Search in the retrieved messages

How the Search take place ?

- Start to slice the question and find the technical keywords by skipping the stop words
- Sending the Technical founded keywords to search Query to start the search
- Return the username and his message
-

```
async function teamsSearchQuery(keyword) {
  try {
    const messages = await teamsFetchAllMessages(keyword);

    const matchedMessages = messages
      .filter((message) => message.body.content.includes(keyword))
```


1:31
25%
Mohamed Alrawy
External
CHAT
TEST 1
TEST 2
TEST FROM ANAN
13 August 7:50 pm
test 2
exayk ya kbeer
• hi
• hello
• hola
15 August 4:58 pm
TEST
Hello from DEMO
best of luck
Last read
31 August 5:12 pm
test done
Type a message
"question": "what is test"
ookies
Headers (7)
Test Results
Raw
Preview
Visualize
JSON
{
"status": 200,
"message": "Question Received",
"data": {
"question": "what is test",
"keywordResults": [
{
"keyword": "test",
"found": true,
"foundMessages": [
{
"sender": "Mohamed Alrawy",
"message": " test done "
},
{
"sender": "Mohamed Alrawy",
"message": " test 2 "
}
]
}
]
}
}

How to Integrate with Stack Platform?

using ***Slack API***

First Step: Create Slack App

- Go to ➔ https://api.slack.com/
- Create new workspace ➔ Provide a new name for your workspace.

Second Step: Configure App Permissions

- Navigate to **OAuth & Permissions** section.
- Scroll down to user token scope ➔ Choose the scope of permissions your app will need
- Here are the permissions that I have used

User Token Scopes

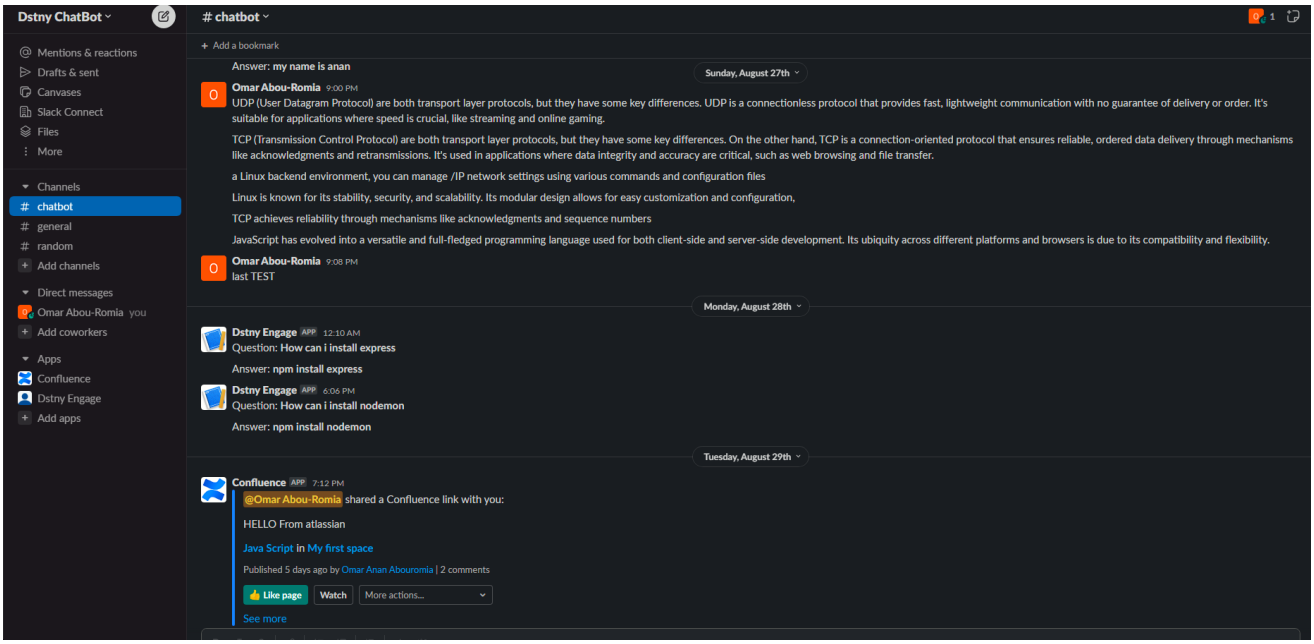
Scopes that access user data and act on behalf of users that authorize them.

OAuth Scope	Description
admin	Administer a workspace
channels:history	View messages and other content in a user's public channels
channels:read	View basic information about public channels in a workspace
channels:write	Manage a user's public channels and create new ones on a user's behalf
channels:write.invites	Invite members to public channels
channels:write.topic	Set the description of public channels
search:read	Search a workspace's content
team.preferences:read	Allows Dstny Engage to read a workspace's preferences

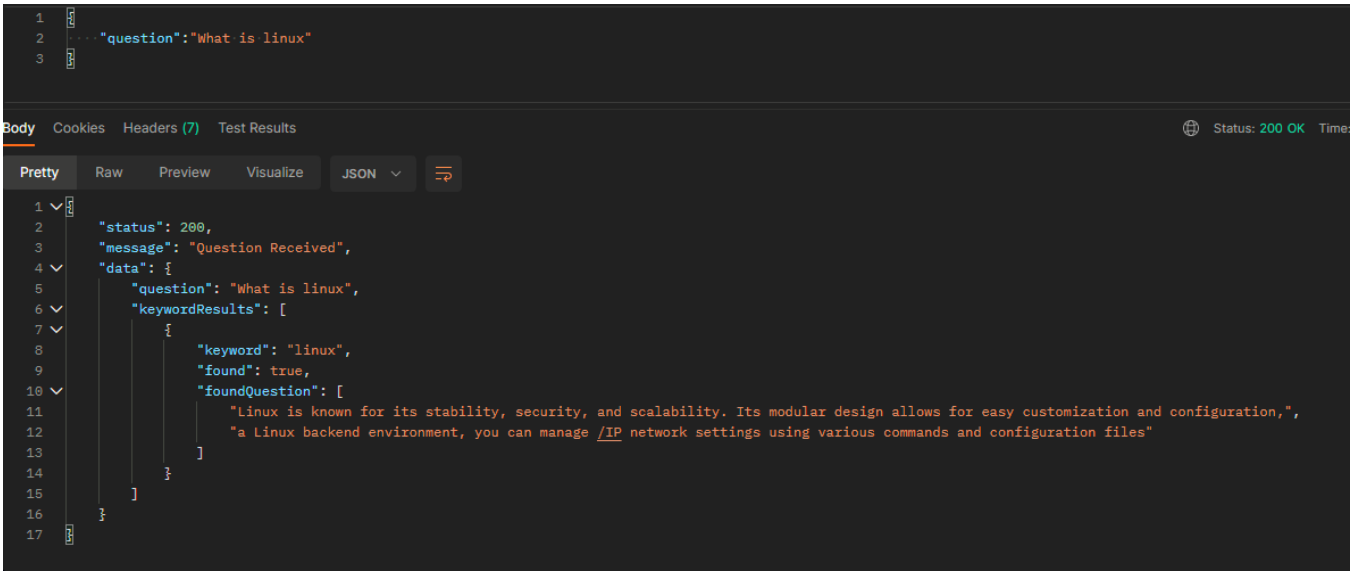
1. **@slack/web-api** ➔ The official Slack Web API client for Node.js.
2. Post message in Slack using **Web Client** ➔https://hooks.slack.com/service/ {Channel path}
3. GET all Message during inside the channel ➔ **web.conversations.history**
4. Search Query ➔use endpoint provided by salck https://slack.com/api/search.all \
   ➔we can search for keywords by giving array of keywords found inside the question that I handle it's logic inside the code

```
1  async function slackSearchQuery(keyword) {
2    try {
3      const headers = {
4        "Content-Type": "application/x-www-form-urlencoded",
5        Authorization: `Bearer ${token}`,
6      };
7
8      const data = new URLSearchParams({
9        query: keyword,
10       count: 10,
11       sort: "timestamp",
12       sort_dir: "desc",
13     }).toString();
14
15     const response = await axios.post(
16       "https://slack.com/api/search.messages",
17       data,
18       { headers }
19     );
20
21     const result = response.data;
```

Messages in slack channel



Endpoint to search inside the Messages



How to integrate with Discord?

- Create an app using the following link: https://discord.com/developers/
- Navigate to BOT and make your BOT permission Administration.
- Take the generated Token to be used in your code.

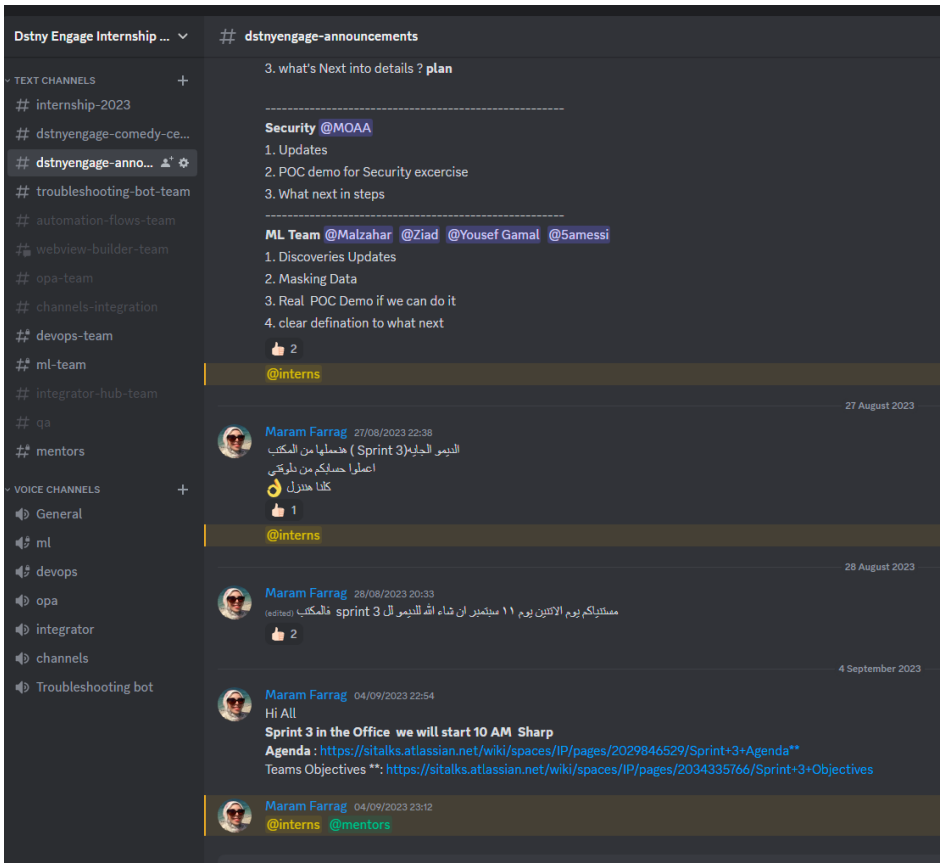
How to Search inside in Discord Channel?

- First of all you need permission from the administrator if the organization
- After that you need to get all the messages from your channel

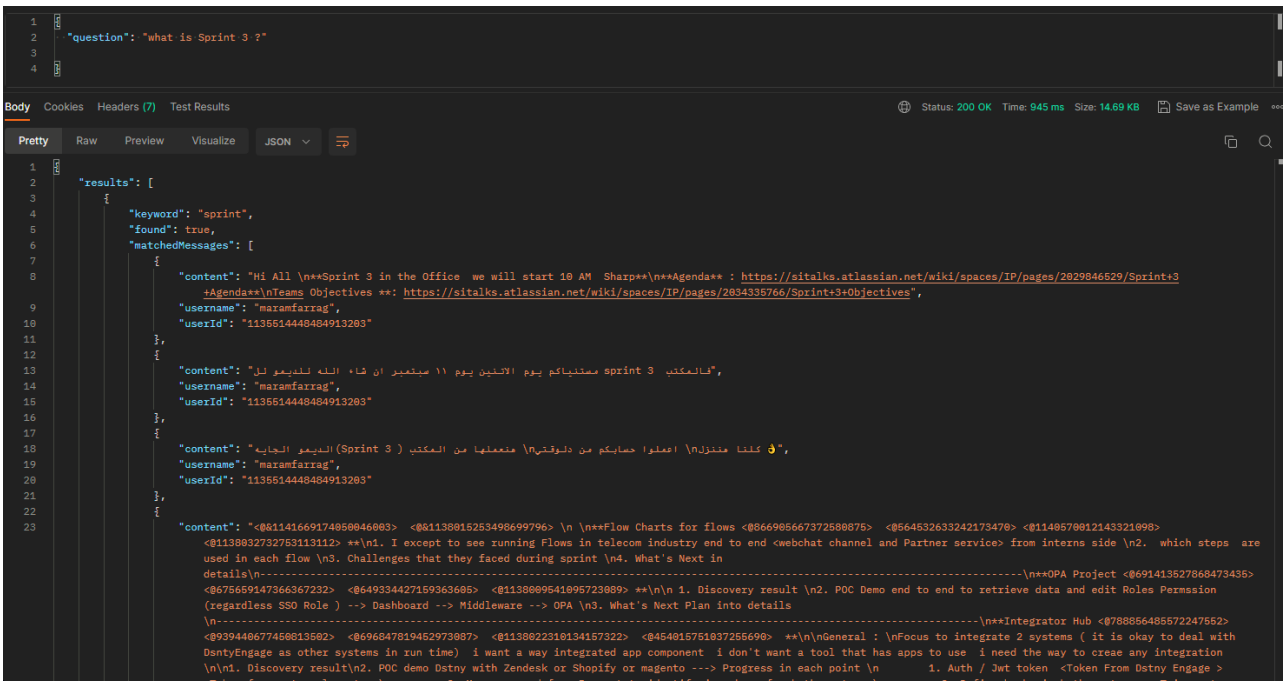
```
const response = await axios.get(
  `https://discord.com/api/v10/channels/${channelID}/messages`,
  {
    headers: {
      Authorization: `Bot ${discordToken}`,
    },
  }
);
```

- After POST the question and found the Technical Keywords ➔ start searching inside the retrieved messages
- Return the **founded messages** and the **username**

Messages from the Channel



Endpoint to search inside the Messages



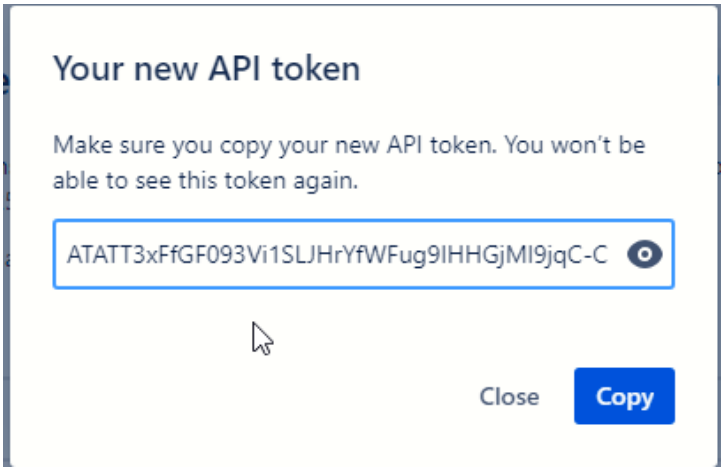
Confluence

How to integrate with Atlassian confluence?

1. Username ➔ your personal email
2. Password ➔ your API Token

```
const auth = {
  username: username,
  password: password,
};
```

3. Page ID ➔ to get access of your page
4. Restful API ➔ to GET or POST data



How to get data from Confluence?

Using the REST API

Expansion: The Confluence REST API uses resource expansion: some parts of a resource are not returned unless explicitly specified. This simplifies responses and minimizes network traffic.

To expand part of a resource in a request, use the `expand` query parameter and specify the entities to be expanded. If you need to expand nested entities, use the `.` dot notation. For example, the following request will expand information about the requested content's space and labels:

```
GET /wiki/rest/api/content/{id}?expand=space,metadata.labels
```

```
async function confluenceGetPageContent(pageId) {
  try {
    const response = await axios.get(
      `${confluenceBaseUrl}/rest/api/content/${pageId}?expand=body.storage,title`,
      {
        auth,
        headers: {
          "Content-Type": "application/json",
        },
      }
    );

    return {
      title: response.data.title,
      body: response.data.body.storage.value,
    };
```

How is the search done in Confluence?

- After getting all the content of the Page ➔ Start search inside this content

```
async function confluenceSearchQuery(keyword) {
  try {
    const pageContent = await confluenceGetPageContent(pageId);

    const keywordFoundInTitle = pageContent.title
      .toLowerCase()
      .includes(keyword);
    const keywordFoundInBody = pageContent.body.toLowerCase().includes(keyword);

```

Search Query and the Blocker

- When I search inside the document if my keyword that I search for it inside the document ➔ All the Content of the Document will be retrieved
- Solution : **The BOLD Article Headline** to Break the Search

Content Inside the Confluence Page

Dstny Engage Troubleshooting ChatBot

OA Owned by Omar Anan Abouromia •••
Last updated: Aug 31, 2023 • 2 min read • See how many people viewed this page

JAVASCRIPT

often celebrated as the "language of the web," has revolutionized the way we interact with the internet. Originally developed as a scripting language for web browsers, JavaScript has evolved into a versatile and full-fledged programming language used for both client-side and server-side development. Its ubiquity across different platforms and browsers is due to its compatibility and flexibility.

LINUX

At the hart of much of today's computing infrastructure lies the Linux operating system. Linux's open-source nature, security features, and customization options have led to its widespread adoption across servers, embedded systems, and even smartphones. Its robust command-line interface (CLI) provides developers and administrators with granular control over the system, making it an ideal choice for managing servers and networking equipment.

Linux's architecture provides the foundation for hosting applications and services. It manages processes, memory, and file systems efficiently, contributing to the stability and scalability of modern software systems. The marriage of JavaScript's versatility with Linux's capabilities has led to the development of powerful applications that leverage the best of both worlds.

TCP/UDP

TCP and UDP are two fundamental networking protocols that underpin internet communication. They both operate at the transport layer of the OSI model, but with distinct characteristics.

Endpoint to search inside the Messages


```
{
  "question": "what is computing "
}
```


Body Cookies Headers (7) Test Results — Status: 200 OK Time: 580 ms Size: 1.15 KB Save as Example

Pretty Raw Preview Visualize JSON

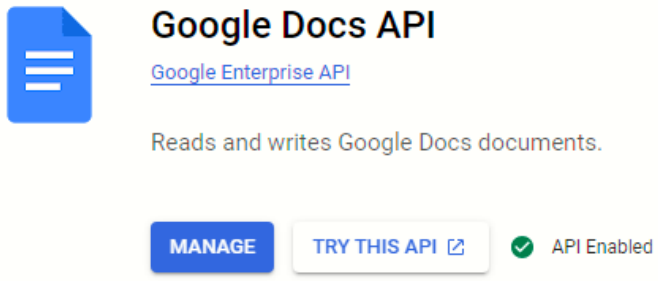

```
{
    "results": [
        {
            "keyword": "computing",
            "articleFound": true,
            "articleContent": "<p><strong>LINUX</strong></p><p>At the hart of much of today's computing infrastructure lies the Linux operating system. Linux's open-source nature, security features, and customization options have led to its widespread adoption across servers, embedded systems, and even smartphones. Its robust command-line interface (CLI) provides developers and administrators with granular control over the system, making it an ideal choice for managing servers and networking equipment.</p><p>Linux's architecture provides the foundation for hosting applications and services. It manages processes, memory, and file systems efficiently, contributing to the stability and scalability of modern software systems. The marriage of JavaScript's versatility with Linux's capabilities has led to the development of powerful applications that leverage the best of both worlds.</p>"
        }
    ]
}
```

How to Integrate to Google Docs? ➔ using **Google Console Cloud**

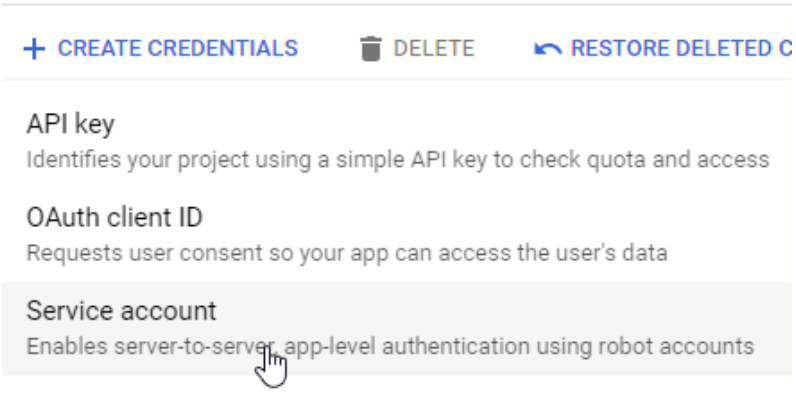
First Step: Create Google Project on Google Console Cloud

- Go to ➔ https://console.cloud.google.com/
- Create new Project ➔ Provide a new name for your Project.

Second Step: Enable Google Docs Api through Library



Third Step: Create Service Account

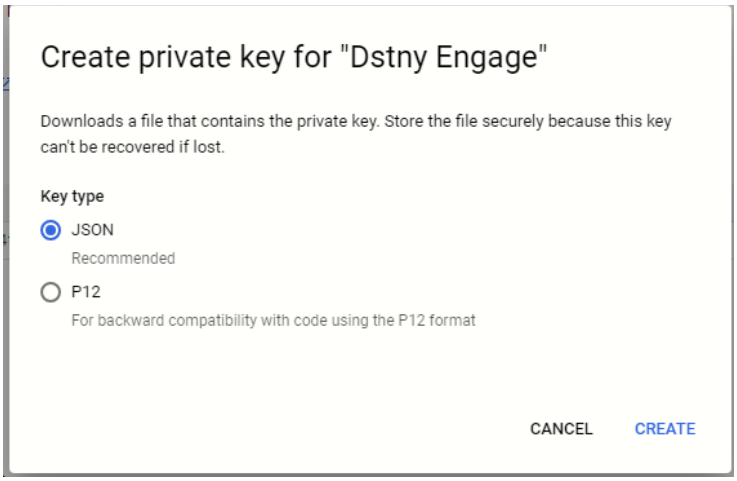


fourth Step : Share the generated @gserviceaccount.com with your Google Docs

dstny-engage@dstny-engage.iam.gserviceaccount.com

Fifth Step: Create a KEY as Credentials

➔ To be used as your credentials during the implementation phase



How to get the page content and start Search inside the content?

1. Using googleapis package ➔ npm install googleapis
2. Get all the content of the page using googledocs.documents.get
3. Then start searching by the Founded Technical Keywords

```
1  async function googleSearchQuery(keyword) {
2    try {
3      const auth = new google.auth.GoogleAuth({
4        keyFile: "credentials.json",
5        scopes: "https://www.googleapis.com/auth/documents",
6      });
7
8      const client = await auth.getClient();
9      const googleDocs = google.docs({ version: "v1", auth: client });
10     const document = await googleDocs.documents.get({
11       documentId,
12     });
13     const content = document.data.body.content;
```

Search Query and the Blocker

- When I search inside the document if my keyword that I search for it inside the document ➔ All the Content of the Document will be retrieved
- Solution : **The BOLD Article Headline** to Break the Search

Content Inside the Google Docs Page

JavaScript:
Often NEW LAST TROUBLESHOOTING celebrated as the "language of the web," has revolutionized the way we interact with the internet. Originally developed as a scripting language for web browsers, JavaScript has evolved into a versatile and full-fledged programming language used for both client-side and server-side development. Its ubiquity across different platforms and browsers is due to its compatibility and flexibility.

Linux: The Command Center

At the heart of much of today's computing infrastructure lies the Linux operating system. Linux's open-source nature, security features, and customization options have led to its widespread adoption across servers, embedded systems, and even smartphones. Its robust command-line interface (CLI) provides developers and administrators with granular control over the system, making it an ideal choice for managing servers and networking equipment.
Linux's architecture provides the foundation for hosting applications and services. It manages processes, memory, and file systems efficiently, contributing to the stability and scalability of modern software systems. The marriage of JavaScript's versatility with Linux's capabilities has led to the development of powerful applications that leverage the best of both worlds.

TCP and UDP: Networking Essentials

TCP and UDP are two fundamental networking protocols that underpin internet communication. They both operate at the transport layer of the OSI model, but with distinct characteristics.

TCP, the Transmission Control Protocol, is known for its reliability. It establishes a connection-oriented communication channel between two devices, ensuring that data is delivered accurately and in the correct order. This reliability comes at the cost of increased overhead due to acknowledgment and error-checking mechanisms.

UDP, the User Datagram Protocol, takes a different approach. It is connectionless and provides a simple, low-overhead means of transmitting data. While UDP does not guarantee delivery or order, it's well-suited for applications where speed and efficiency are paramount, such as real-time streaming and online gaming.

Endpoint to search inside the Messages


```
{
  "question": "What is the difference between DstnyIntegration javascript and Backend and Algorithm and Linux"
}
```

ody Cookies Headers (7) Test Results Status: 200 OK Time: 4.14 s Size: 1.84 KB Save as Example

Pretty Raw Preview Visualize JSON

```
{
  "results": [
    {
      "keyword": "dstnyintegration",
      "articleFound": false,
      "articleContent": ""
    },
    {
      "keyword": "javascript",
      "articleFound": true,
      "articleContent": "JavaScript: \nOften NEW LAST TROUBLESHOOTING celebrated  as the \"language of the web,\" has revolutionized the way we interact with the internet. Originally developed as a scripting language for web browsers, JavaScript has evolved into a versatile and full-fledged programming language used for both client-side and server-side development. Its ubiquity across different platforms and browsers is due to its compatibility and flexibility.\n\n\n\n\n"
    },
    {
      "keyword": "backend",
      "articleFound": false,
      "articleContent": ""
    },
    {
      "keyword": "algorithm",
      "articleFound": false,
      "articleContent": ""
    },
    {
      "keyword": "linux",
      "articleFound": true,
      "articleContent": "Linux: The Command Center\n\nAt the heart of much of today's computing infrastructure lies the Linux operating system. Linux's open-source nature, security features, and customization options have led to its widespread adoption across servers, embedded systems, and even smartphones. Its robust command-line interface (CLI) provides developers and administrators with granular control over the system, making it an ideal choice for managing servers and networking equipment.\n\nLinux's architecture provides the foundation for hosting applications and services. It manages processes, memory, and file systems efficiently, contributing to the stability and scalability of modern software systems. The marriage of JavaScript's versatility with Linux's capabilities has led to the development of powerful applications that leverage the best of both worlds.\n\n"
    }
```

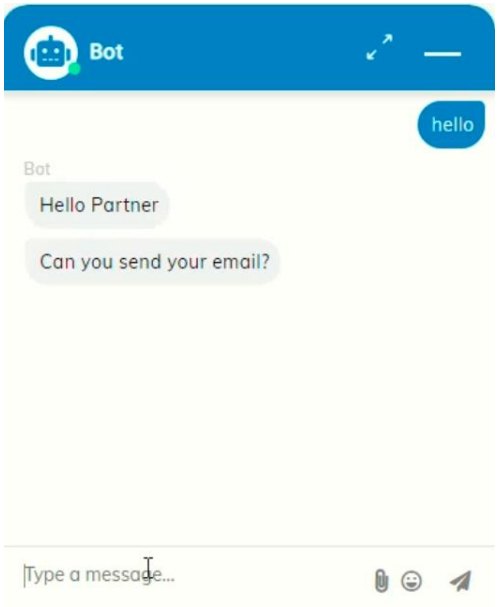
What did we use to build our Webchat?

1. Automation SDK
2. Integrate with partner
3. Create Endpoint to POST the Question and return the Founded Query from our 5 Providers

How we generate our Flow in the Webchat?

First:

1. we start with greeting message, and we use randomText method to generate random Greeting text
2. Ask the user to enter his email and and we use userInput method to save the input of the user

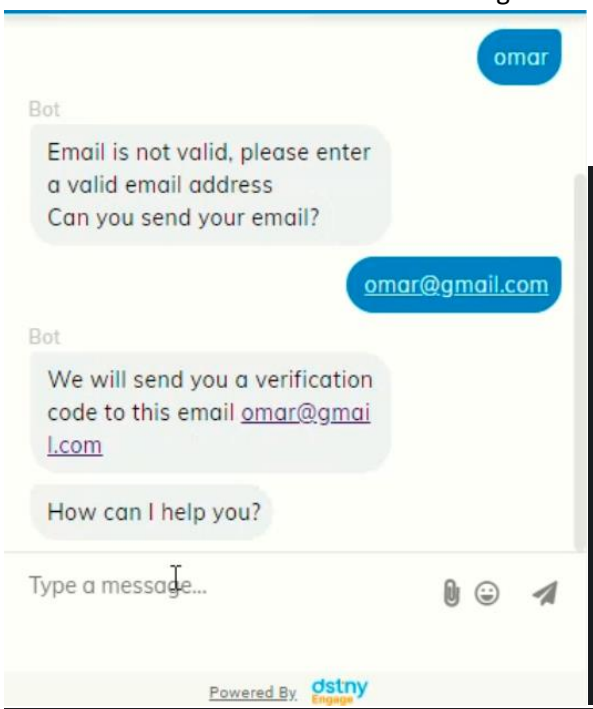


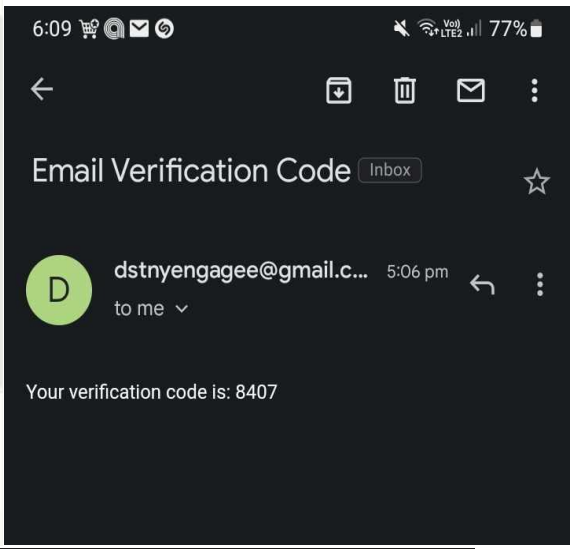

```
.randomText([
    ["Hello Partner", 1],
    ["Welcome Commander", 1],
    ["Greeting Developer", 1],
  ])

  .userInput({
    question: "Can you send your email?",
    contextParam: "userEmail",
```

Second:

Validate the user email and send a message that verification code will be generate

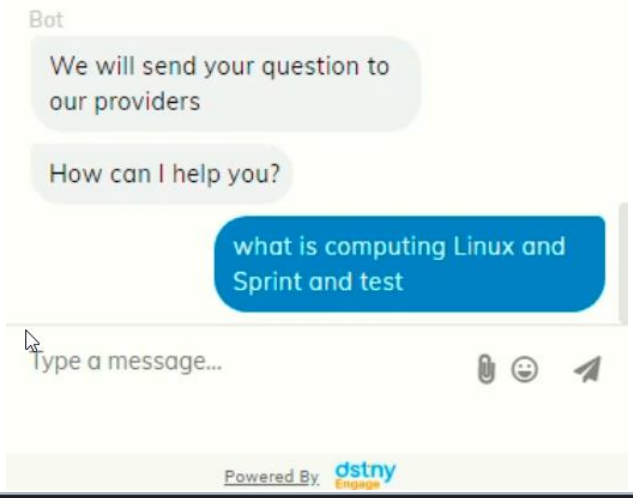





```
validation: {
      regex: "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,}$",
      errorMessage: "Email is not valid, please enter a valid email address ",
    },
  })
  .api(
    "http://localhost:3000/verification-code",
    "POST",
    {},
    { email: "{{params.userEmail}}" }
  )
  .text([
    [
      "We will send you a verification code to this email {{{params.userEmail}}}",
      1,
    ],
  ])
```

Third:

1. Ask the user to how can we help him
2. Use userInput to save user question in variable
3. Send the user question in the body of api function to start our searching




```
.userInput({
      question: "How can I help you?",
      contextParam: "userQuestion",
    })
    .api(
      "http://localhost:3000/question",
      "POST",
      {},
      { question: "{{params.userQuestion}}" }
    )
```

Fourth:

1. After getting our response from the endpoint with the suitable Format
2. We show our response in the webchat using dynammicCarousel method

```
.dynamicCarousel("{{{api.response.json.results}}}", {
    id: "{{id}}",
    mediaURL: "{{{image_url}}}",
    title: "{{{title}}}",
    subTitle: "{{{short_description}}}",
    buttons: [
      new FlowButton(
        "{{id}}",
        "Show More",
        {
          id: "{{id}}",
          title: "{{title}}",
          short_description: "{{short_description}}",
        },
        new WebchatFlow("b1", "sdk")
          .text([["Show More to {{payload.title}}"]])
          .text([[" {{payload.short_description}}"]])
```

POST http://localhost:3000/question Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

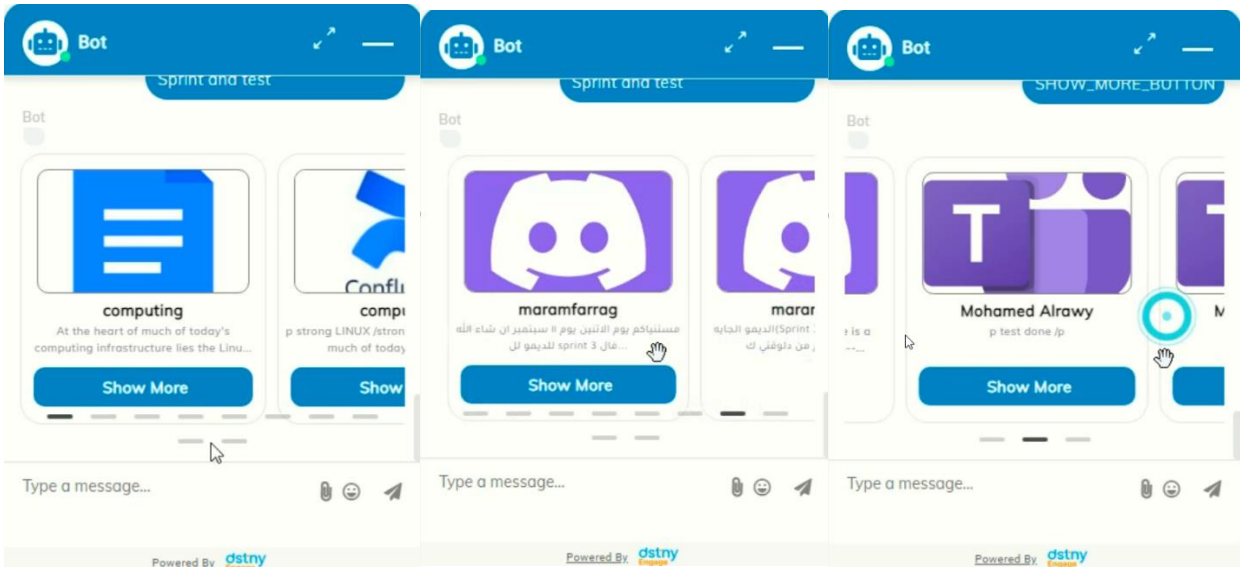
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
  "question":"what is computing and sprint dstny ?"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 28.83 s Size: 12.21 KB Save as Example

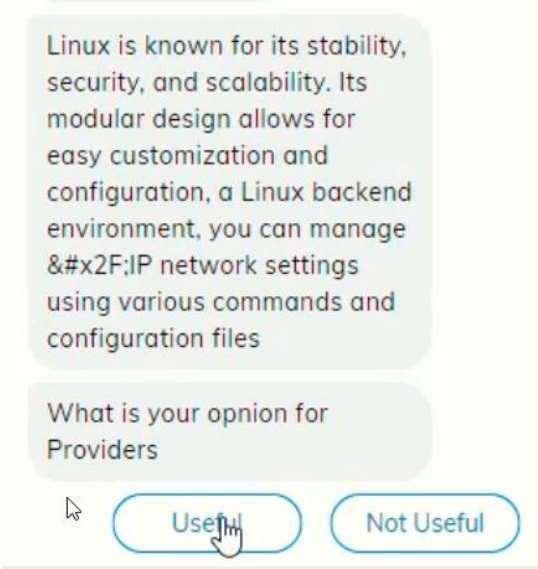
Pretty Raw Preview Visualize JSON


```
{
  "results": [
    {
      "id": "81c3540d-5dbb-4331-8753-f208f12b6f79",
      "source": "google_docs",
      "title": "computing",
      "short_description": "At the heart of much of today's computing infrastructure lies the Linux operating system. Linux's op",
      "image_url": "https://storage.googleapis.com/gweb-uniblog-publish-prod/images/Google_Docs.width-500.format-webp.webp"
    },
    {
      "id": "837be62c-bcd9-4410-8971-256f08813dc1",
      "source": "confluence",
      "title": "computing",
      "short_description": "<p><strong>LINUX</strong></p><p>At the hart of much of today's computing infrastructure lies the Linux operating system. Linux's open-source nature, s",
      "image_url": "https://media.licdn.com/dms/image/C4D0BAQHR3-muCN3-4A/company-logo_200_200/0/1611699233211?e=2147483647&v=beta&t=9ih0eEQjXVMMb5gA76KKXUV3T-VJkTN812f769UiBr4"
    },
    {
      "short_description": "Hi All \n**Sprint 3 in the Office  we will start 10 AM  Sharp**\n**Agenda** : https://sitalks.atlassian.net/wiki/spaces/IP/pages/2829846529/Sprint+3
        +Agenda**\nTeams Objectives **: https://sitalks.atlassian.net/wiki/spaces/IP/pages/2034335766/Sprint+3+Objectives",
      "title": "maramfarrag",
      "userId": "113551444848491320 3",
      "id": "a4995f22-e39f-4d22-8bcf-84cd5ede4705",
      "keyword": "sprint",
      "image_url": "https://play-lh.googleusercontent.com/0oO5sAneb9lJP6l8c6DH4aj6f85qNpplQVHmPmbbBxAukDnlO7DarDW0b-kEIHa8SQ"
    },
    {
      "short_description": "فالمكتب  sprint 3 مستنياكم يوم الاتنين يوم ١١ سيتعبر ان شاء الله للديمو لل",
      "title": "maramfarrag",
      "userId": "113551444848491320 3",
      "id": "fdab96b3-69cb-43aa-a322-ec31f26ff190",
      "keyword": "sprint",
      "image_url": "https://play-lh.googleusercontent.com/0oO5sAneb9lJP6l8c6DH4aj6f85qNpplQVHmPmbbBxAukDnlO7DarDW0b-kEIHa8SQ"
    },
    {
      "short_description": "الديمو الجايه(Sprint 3 ) متعملها من المكتب \nكلنا هننزل اعملوا حسابكم من دلوقتي\n",
```



Fifth:

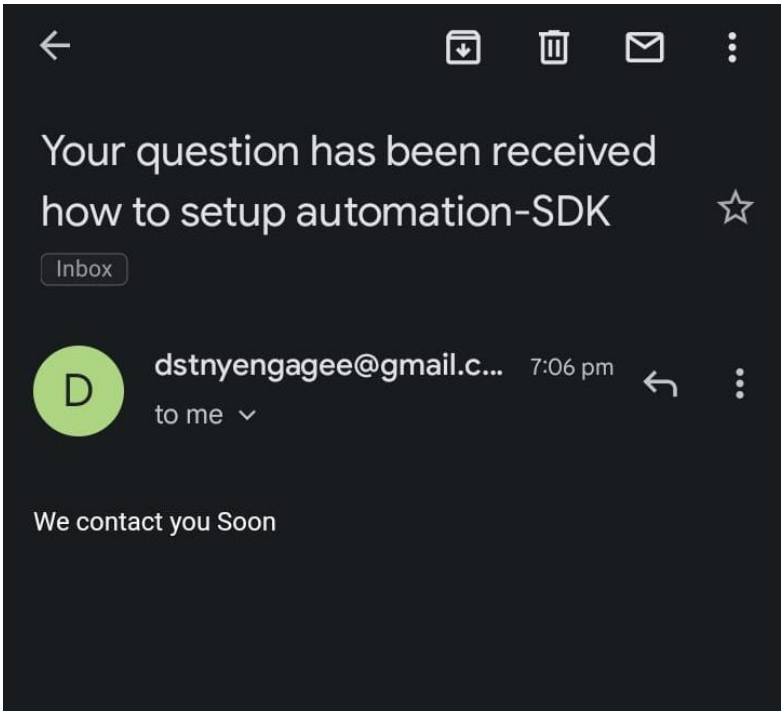
1. By clicking on show more button the we show the user the body of fetched message he select
2. Then we ask the user for feedback using quickReply method




```
.quickReply("What is your opnion for Providers", [
            new FlowButton("1", "Useful", "1", firstFlow()),
            new FlowButton("2", "Not Useful", "2", thirdFlow()),
          ])
```

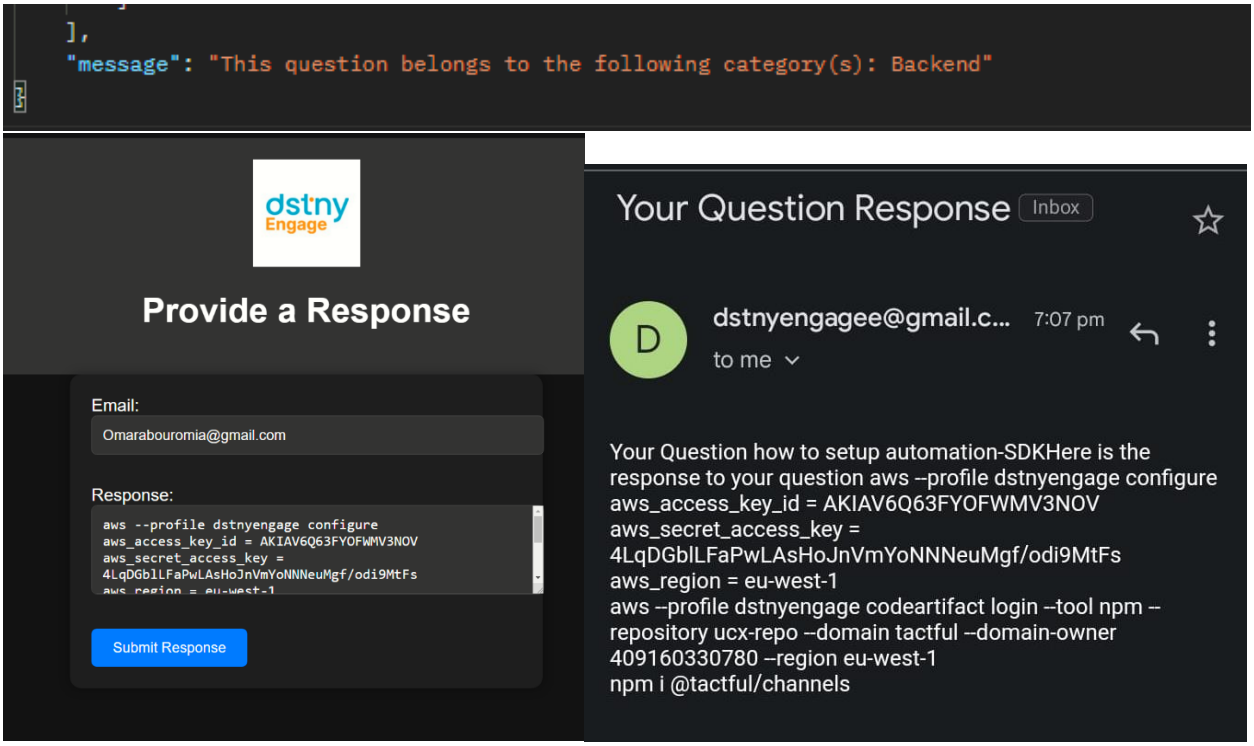
Sixth:

1. If useful is chosen then we take the answer of our providers and save it in our database and send to user thankful message
2. If not useful then we send the question the categorized team depend on the category of the question to answer the user question




```
.text([["one of our team will contact you soon"]])
.api(
  "http://localhost:3000/userData",
  "POST",
  {},
  { email: "{{params.userEmail}}", question: "{{params.userQuestion}}" }
);
```

After categorized the question the categorized team will send email with response



The Backend Server

technology used in the backend server.:

1. ElasticSearch
2. Postgresql
3. Node js
4. Express

The ElasticSearch

Why:

Elasticsearch is a powerful and flexible search engine that is widely used for various purposes, including matching questions with existing questions. Here are some reasons why you should consider using Elasticsearch for this task:

1. High-Speed Search: Elasticsearch is designed for fast and efficient full-text searching. It can quickly retrieve relevant results, making it ideal for matching questions against a large database of questions.

2. Scalability: Elasticsearch is horizontally scalable, which means you can easily add more nodes to handle increased data and query loads. This is essential if your question database is expected to grow over time.

3. Relevant Scoring: Elasticsearch uses a scoring system to rank search results based on relevance. You can configure scoring algorithms to ensure that the most relevant questions are presented at the top of the search results.

4. Fuzzy Matching: Elasticsearch supports fuzzy searching, which allows you to find similar questions even if there are slight variations, typos, or synonyms in the query. This is especially useful for user-generated content where questions may not be perfectly structured.

5. Full-Text Analysis: Elasticsearch provides advanced text analysis capabilities, such as tokenization, stemming, and stop-word removal. This enables you to process and search questions effectively, even when users express questions differently.

6. Faceted Search: Elasticsearch supports faceted search, allowing users to filter and refine their search results based on various attributes or categories. This is useful for enhancing the user experience by enabling them to narrow down their search results.

7. Real-Time Data Indexing: Elasticsearch excels at indexing new data in real-time. If your question database is dynamic and frequently updated, Elasticsearch ensures that new questions are quickly available for search.

8. Open Source and Active Community: Elasticsearch is open-source, with a vibrant community that actively contributes to its development and offers support. This makes it a cost-effective solution with a wealth of resources.

9. RESTful API: Elasticsearch provides a simple RESTful API, making it easy to integrate with various programming languages and frameworks. This allows you to build custom applications and interfaces to interact with your question database.

10. High Availability: Elasticsearch can be configured for high availability and data redundancy, ensuring that your question database is resilient to hardware failures.

In summary, Elasticsearch is a robust choice for matching questions with existing questions due to its high-speed search capabilities, scalability, and advanced features for text analysis and query flexibility. It's well-suited for a wide range of applications that require efficient and accurate search functionality.

The Postgresql

Why Use PostgreSQL with Elasticsearch?

In our project, we have opted to use PostgreSQL in conjunction with Elasticsearch to provide a robust and consistent data storage and search solution. This combination offers several advantages that contribute to the success of our application:

1. Data Persistence:

PostgreSQL serves as a reliable database for storing structured data. It ensures data integrity and provides transactional support, making it ideal for preserving the core information in our application.

2. Complex Data Modeling:

PostgreSQL excels at handling complex data models, supporting relational data structures, and allowing us to define relationships between different

entities. This is particularly useful when we need to store additional information related to the questions in our Elasticsearch index.

3. Data Consistency:

By maintaining a PostgreSQL database in sync with our Elasticsearch index, we ensure data consistency. This is crucial for applications where data integrity is paramount. Any changes or updates to our questions are first made in the PostgreSQL database before being reflected in Elasticsearch.

4. Rich Querying and Reporting:

PostgreSQL offers a wide range of SQL querying capabilities, enabling us to perform complex data analysis, generate reports, and extract valuable insights from the stored information. This complements Elasticsearch's search functionality, allowing us to cover a broader spectrum of data-related tasks.

5. Historical Data and Audit Trails:

With PostgreSQL, we can easily implement historical data tracking and audit trails, which is valuable for maintaining a comprehensive history of changes to the questions over time. This adds an extra layer of data accountability and security.

6. ACID Compliance:

PostgreSQL adheres to ACID (Atomicity, Consistency, Isolation, Durability) principles, ensuring that data remains reliable, even under high concurrent access and modification. This level of reliability is crucial for mission-critical applications.

7. Seamless Integration:

The PostgreSQL-Elasticsearch combination is highly versatile. We've integrated these two systems effectively, allowing Elasticsearch to focus

on providing fast and efficient search results, while PostgreSQL handles data storage and management. This enhances the overall performance of our application.

8. Backup and Disaster Recovery:

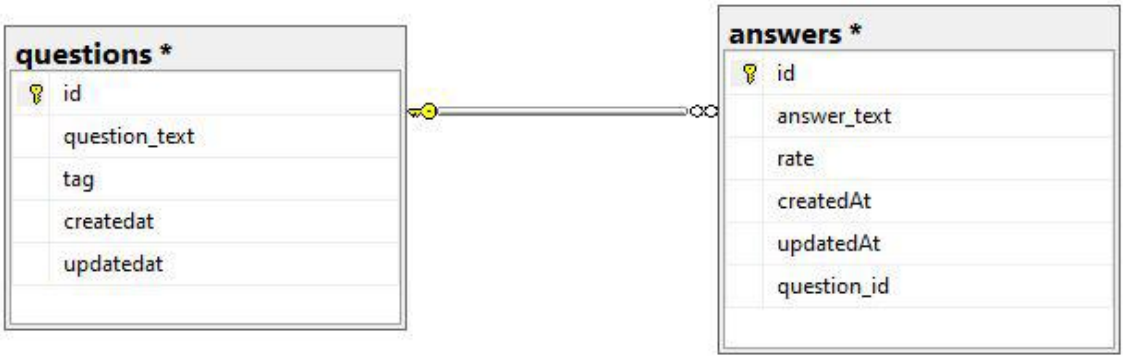
PostgreSQL's backup and replication features enable us to implement solid disaster recovery strategies. This ensures that our data is safe even in the event of unforeseen issues.

9. Data Enrichment:

PostgreSQL can be used to enrich the questions with additional context, metadata, or related information. This enriched data can then be made available for more advanced searching and filtering through Elasticsearch.

By utilizing the combined power of PostgreSQL and Elasticsearch, we strike a balance between structured data storage and efficient full-text searching. This comprehensive solution helps us maintain data consistency, reliability, and advanced querying capabilities, all of which are essential for a successful and high-performance application.

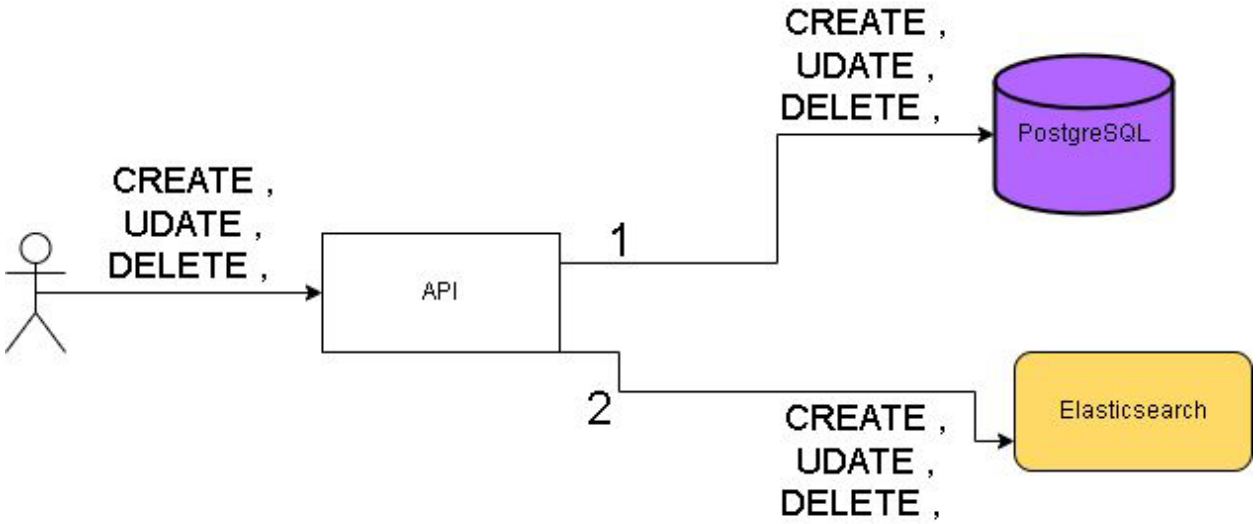
Database ERD:

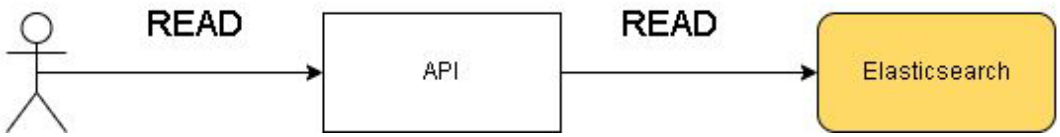


The Relationship between questions table and answers table:

one to many relationships, one question can has many answers.

Typical architecture with PostgreSQL and Elasticsearch



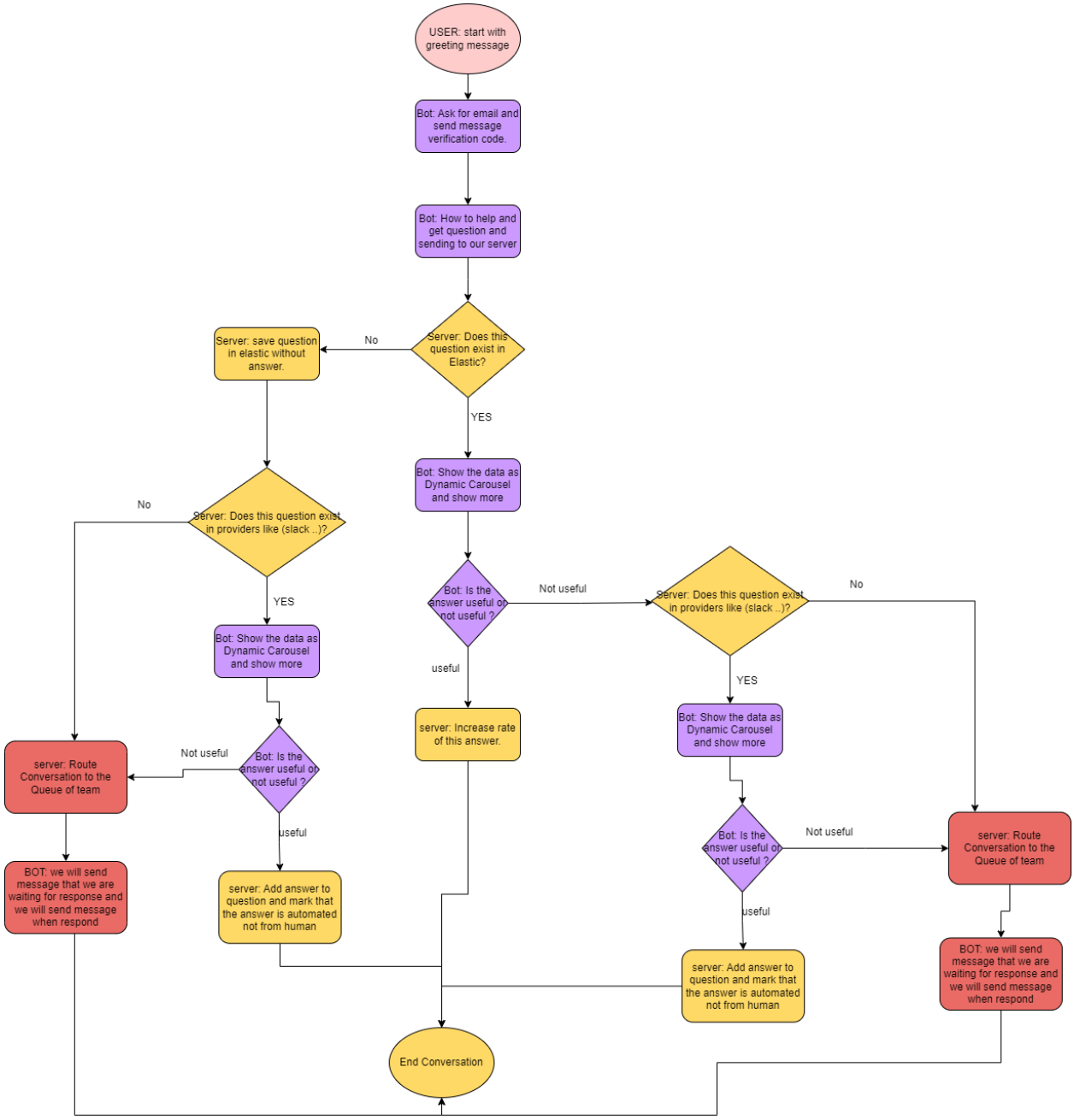


Details:

- when we use the create, update, and delete operation we must do these operations to PostgreSQL and Elasticsearch to make data consistency.

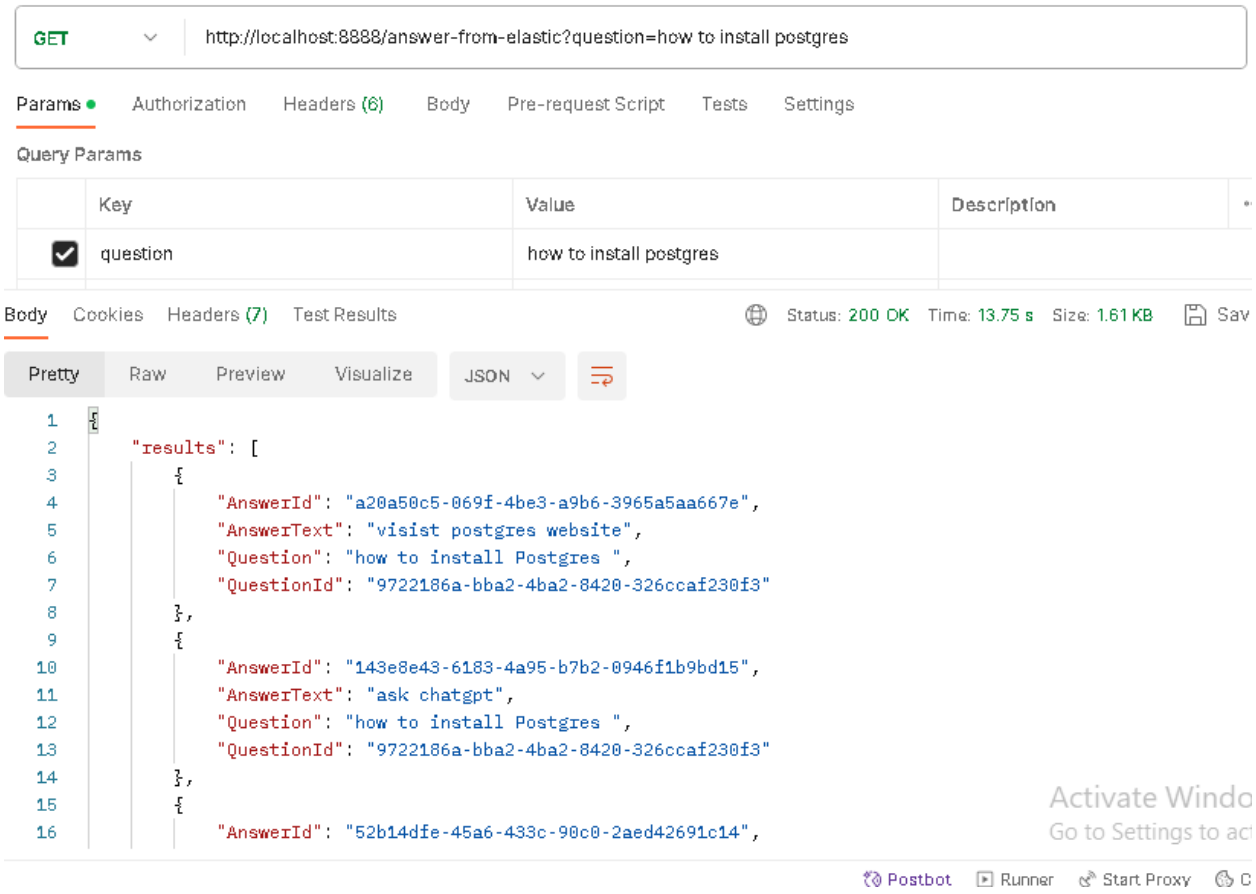
- when we use the Read operation we read from Elasticsearch.

The flowchart of the bot flow



The API Endpoints in our API

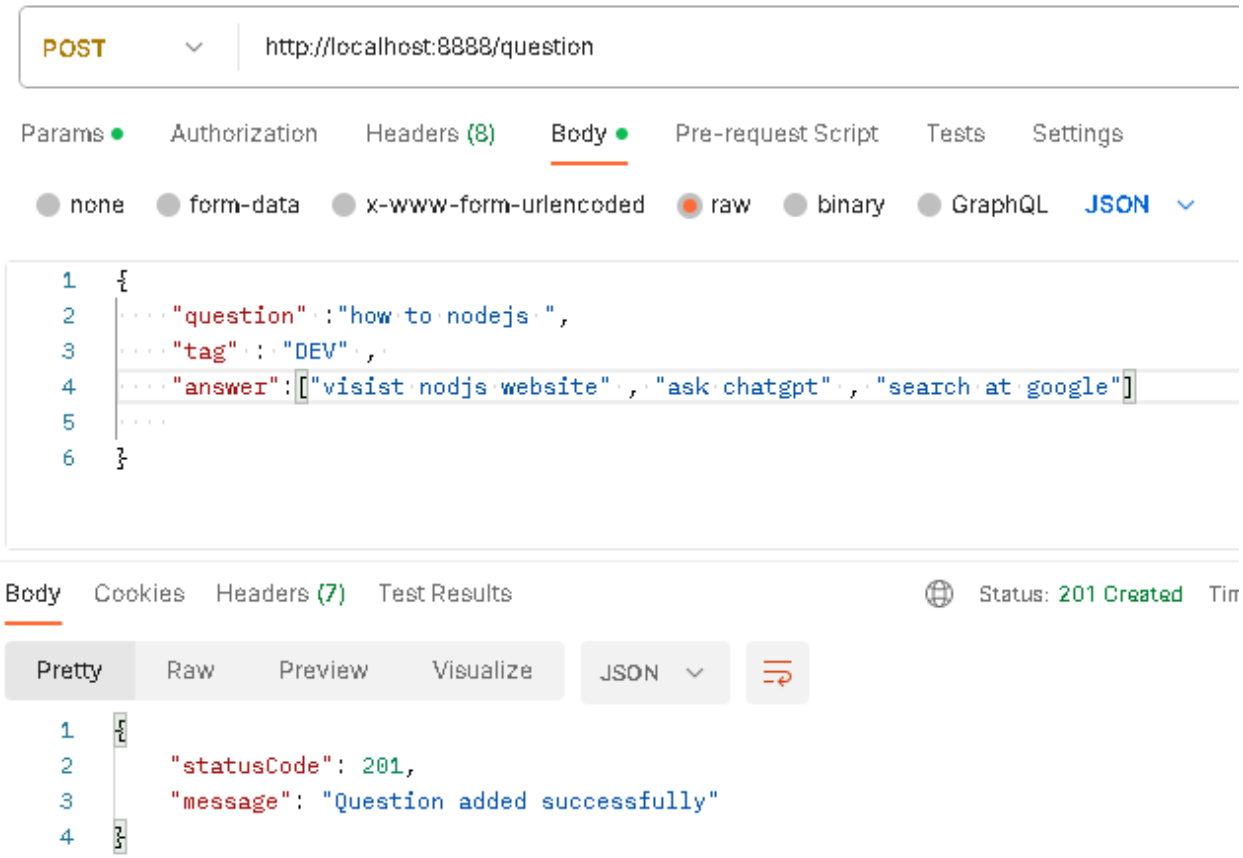
GET: http://localhost:8888/answer-from-elastic?question=how to install postgres



Description:

- Get match questions from elasticsearch and results in this form to able to put it in dynamic carousel

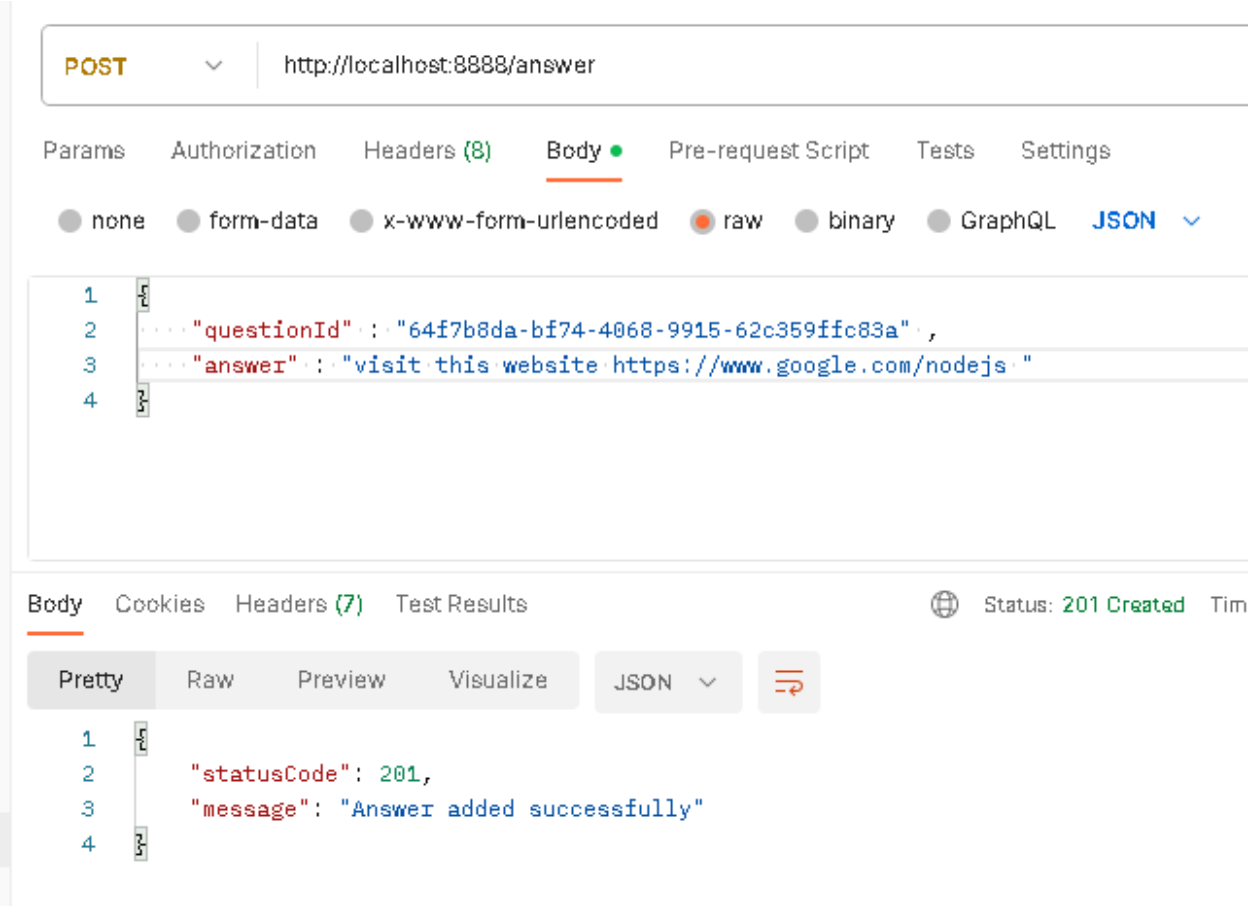
POST: http://localhost:8888/question



Description:

- add question with array of answers.

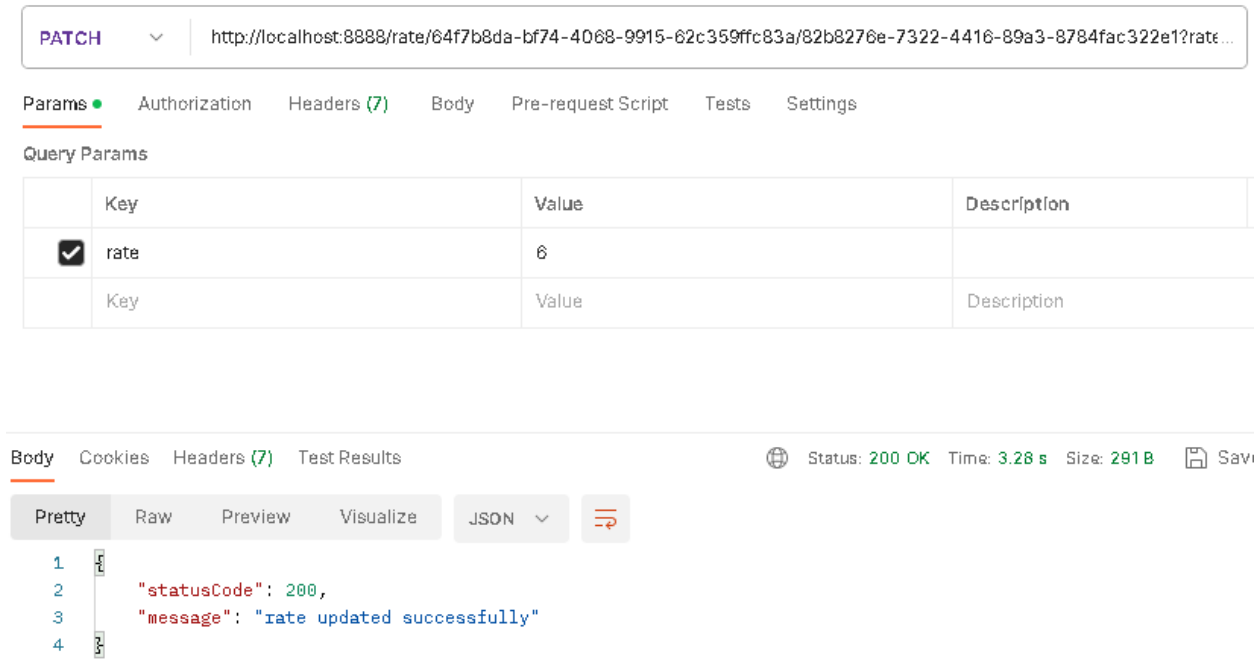
POST: http://localhost:8888/answer



Description:

- add the answer to specific question.

PATCH: http://localhost:8888/rate/64f7b8da-bf74-4068-9915-62c359ffc83a/82b8276e-7322-4416-89a3-8784fac322e1?rate=6

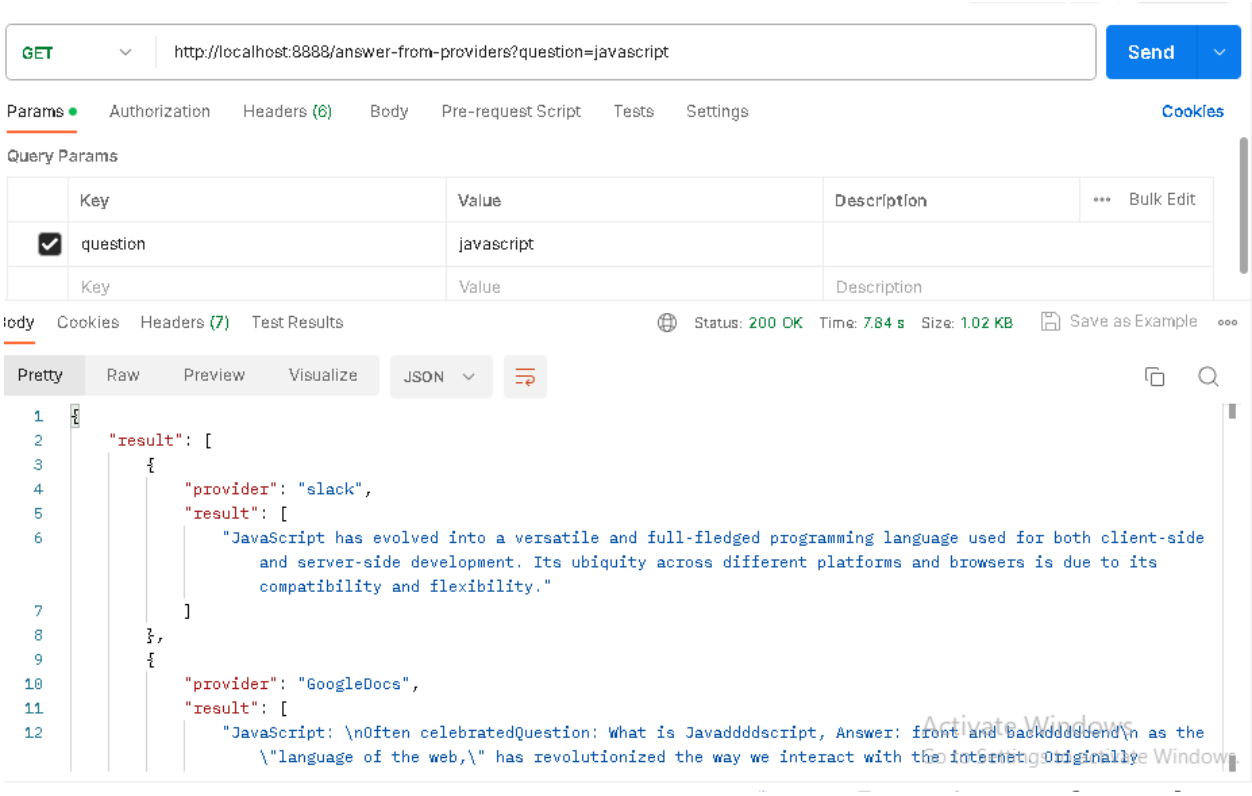


Description:

- Update rate for a specific answer,

we take question id and answer id from URL and take rate from querey string.

GET: http://localhost:8888/answer-from-providers?question=javascript



Description:

- get answers from providers like slack etc

Presentation:
https://www.canva.com/design/DAFuFDYlyxw/lyQpv805RxVJ_56ub8xSeg/edit?utm_content=DAFuFDYlyxw&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

GitHub Link: https://github.com/tactful-ai/Troublshooting-Bot.git

First Flow:
https://drive.google.com/file/d/10mlSuWSaFGfXMT_X2SmlKmX8r_QwSgLF/view?usp=sharing

Second Flow:

https://drive.google.com/file/d/1JoqycNp_Nr6LtftkUXF5FrV7KQHaQPnX/view?usp=sharing

Final Flow:

https://drive.google.com/file/d/1PUPiuQM0EA7u64pcARvArGWFvg25COi3/view?usp=sharing

Thank you , Dstny Engage