MIT World Peace University

NASA Space Apps Challenge 2023

STAR App LLM for Querying Technical Documentation.

NASA Space Apps Challenge 2023

Prepared By

Krishnaraj Thadesar
Sahaj Mishra
Neeraj Varma
Saubhagya Singh
Aaron Philip
Yashvardhan Tekawade

October 8, 2023

Contents

1 Problem Statement

Greetings, space enthusiast! Are you tired of slogging through technical requirements, only to find omissions and inconsistencies that could spell disaster for your mission? What if there was an Artificial Intelligence (AI)-powered app called STAR (Standards Technical Assistance Resource) that could streamline the process and offer requirement recommendations? Your challenge is to develop the approach, code, or procedure for STAR, so that with STAR as a copilot, mission designers can blast off with even greater confidence, knowing that they have the right requirements in place.

2 Screenshots

2.1 Login

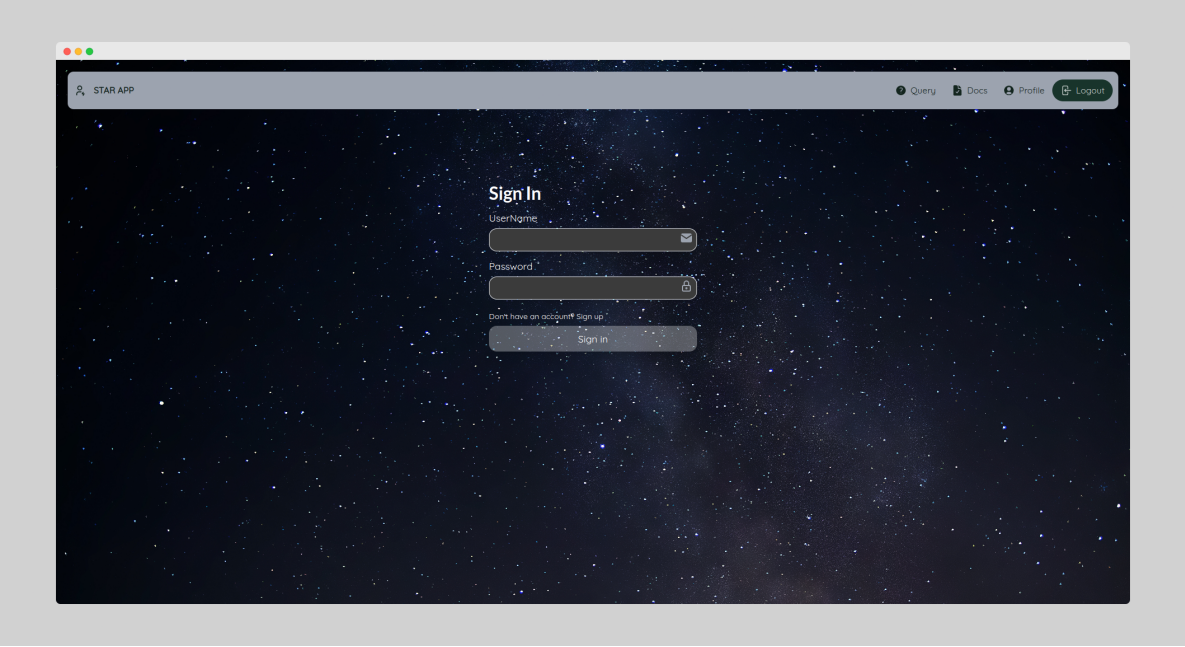

Figure 1: signin

2.2 Query Page

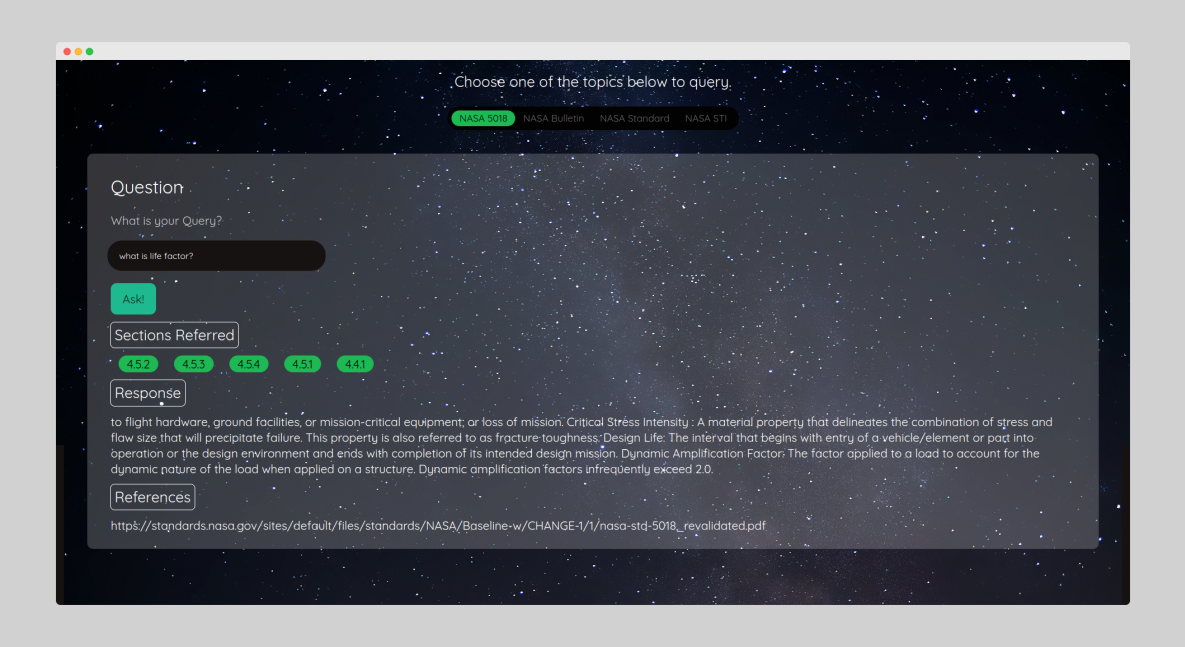

Figure 2: Query

2.3 Answer

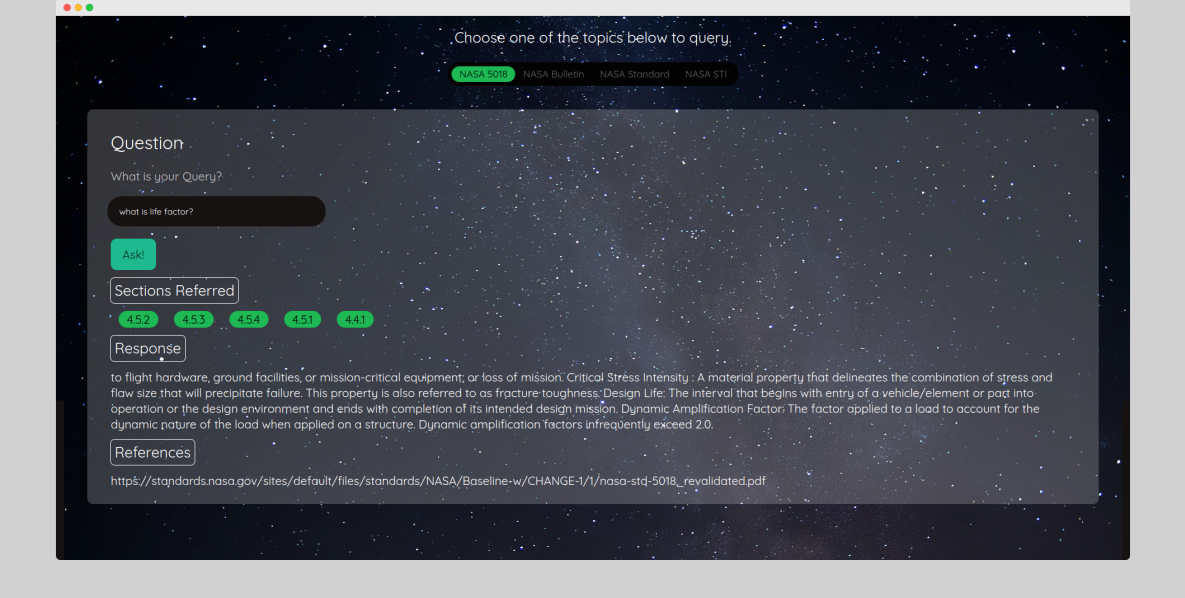

Figure 3: Answer

2.4 Docs

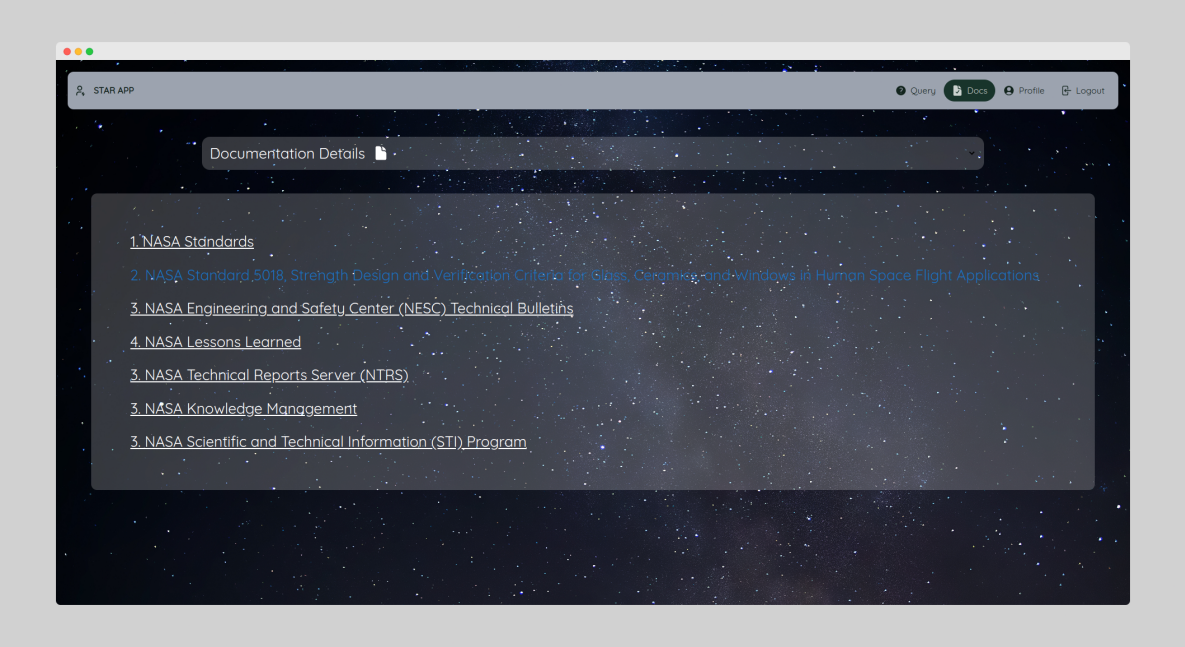

Figure 4: Docs

2.5 Server

```
C:\STAR\backend\query\views.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 08, 2023 - 20:55:27
Django version 4.2.6, using settings 'website.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[08/Oct/2023 20:55:48] "GET //query/nasa/?query=life+factor HTTP/1.1" 200 801
[08/Oct/2023 20:56:05] "GET //query/nasa/?query=what+is+life HTTP/1.1" 200 756
[08/Oct/2023 20:56:17] "GET //query/nasa/?query=what+is+life+factor%3F HTTP/1.1" 200 883

(.venv) C:\STAR\backend>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 08, 2023 - 21:28:26
Django version 4.2.6, using settings 'website.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[08/Oct/2023 21:30:00] "GET //query/nasa/?query=life+factor HTTP/1.1" 200 801
[08/Oct/2023 21:39:49] "GET //query/nasa/?query=life+factor HTTP/1.1" 200 801
[08/Oct/2023 21:42:07] "GET //query/nasa/?query=life+factor HTTP/1.1" 200 801
[08/Oct/2023 21:42:21] "GET //query/spacetech/?query=tell+me+about+software+control+categories HTTP/1.1" 200 293
[08/Oct/2023 21:42:40] "GET //query/standards/?query=tell+me+about+coated+window+panes HTTP/1.1" 200 534
```

Figure 5: Django Server

3 Features

1. Ability to ingest and convert documents into queryable information
2. Ability to text with the contents of the document
3. Different document sources to choose from
4. Ability to view previous queries of a user
5. User authentication and security
6. Ability to view source documents directly
7. Segmented answers that provide section,response,ref
8. Interactive UI
9. Dark mode

4 Files and Directories

4.1 Backend

4.1.1 views.py

1. The file defines several functions for user registration, login, and querying NASA-related information.
2. The functions use various libraries such as Django, faiss, and bcrypt for authentication and data retrieval.
3. The functions retrieve relevant documents based on user queries and return the answers, sections, and references as a JSON response.
4. The file also includes a function to remove special characters from text and a function to retrieve user query data from the database.
5. The file uses regular expressions to match patterns in the retrieved documents and extract relevant information.

4.1.2 registeruser

1. The code defines a function called registeruser that handles user registration requests.
2. The function checks if the request method is POST and parses the request body as JSON data.
3. The function validates the email format and checks if the required fields (username, password, and email) are present in the request data.
4. The function concatenates the username and password, hashes the concatenated value using bcrypt, and inserts the user data into a database table called UserReltion.
5. The function returns a JSON response indicating whether the user registration was successful or not.

4.1.3 loginuser

1. The code defines a function called loginuser that handles user login requests.
2. The function checks if the request method is POST and parses the request body as JSON data.
3. The function retrieves user data from the database based on the provided username and checks if the provided password matches the hashed password in the database. If the authentication is successful, the function returns a JSON response indicating that the login was successful. If the authentication fails, the function returns a JSON response indicating that the username or password is invalid, or that the user was not found in the database.

4.1.4 nasa

1. The code defines a function called nasa that handles NASA-related queries.
2. The function uses the GooglePalmEmbeddings class to create embeddings for the query and the FAISS class to retrieve relevant documents based on the query.
3. The function retrieves the query and username from the request parameters and checks if the query contains the word "summary".
4. The function extracts relevant information from the retrieved documents using regular expressions and returns the answers, sections, and references as a JSON response.
5. The function also has a commented-out section that appends the query to the queriesdata field in the UserReltion table in the database.

4.1.5 bulletin

1. The code defines a function called bulletin that handles requests related to NASA bulletins.
2. The function uses the GooglePalmEmbeddings class to create embeddings for the query and the FAISS class to retrieve relevant documents based on the query.
3. The function retrieves the query from the request parameters and checks if it is empty. If it is empty, the function returns a JSON response with an error message and a 400 status code.
4. The function retrieves relevant documents based on the query and extracts relevant information using regular expressions. It then returns the answers, sections, and references as a JSON response.
5. The function also has a commented-out section that appends the query to the queriesdata field in the UserReltion table in the database.

4.1.6 standards

1. The code defines a function called standards that handles requests related to NASA standards.
2. The function creates a GooglePalmEmbeddings object and a FAISS object to retrieve relevant documents based on the query.
3. The function retrieves the query from the request parameters and checks if it is empty. If it is empty, the function returns a JSON response with an error message and a 400 status code.

4. The function retrieves relevant documents based on the query and extracts relevant information using regular expressions. It then returns the answers, sections, and references as a JSON response.

5. The function returns a JSON response with the answers, sections, and references to the relevant NASA standards.

4.1.7 spacetech

1. The code defines a function called spacetech that handles requests related to space technology.

2. The function creates a GooglePalmEmbeddings object and a FAISS object to retrieve relevant documents based on the query.

3. The function retrieves the query from the request parameters and checks if it is empty. If it is empty, the function returns a JSON response with an error message and a 400 status code.

4. The function retrieves relevant documents based on the query and extracts relevant information using regular expressions. It then returns the answers, sections, and references as a JSON response.

5. The function returns a JSON response with the answers, sections, and references to the relevant space technology documents.

4.1.8 queriesrequest

1. The code defines a function called queriesreuest that handles GET requests related to user queries.

2. The function retrieves the username from the request parameters and checks if it is empty. If it is empty, the function returns a JSON response with an error message and a 400 status code.

3. The function retrieves query data from the database based on the provided username using a SQL query.

4. The function extracts the query data from the SQL query result and returns it as a JSON response. If the provided username is not found in the database, the function returns a JSON response with an error message and a 404 status code.

4.2 Machine Learning

User Interaction This feature involves handling interactions with the user. Users input their queries or questions through a user interface, and the bot is responsible for processing these inputs and providing relevant answers.

Query Processing In this step, the bot takes the user's query and prepares it for further analysis. It may involve text preprocessing, removing irrelevant information, or extracting key terms to improve the accuracy of the search.

Query Embedding Query embedding is a technique used to convert the user's query into a numerical vector representation. This numerical representation helps in comparing the query with other text data for similarity analysis.

Similarity Search Similarity search is a core component of the bot's functionality. It involves comparing the query embedding to embeddings of documents or content within the PDFs. The goal is to find the most relevant documents or passages that closely match the user's query.

Finetuning After retrieving potential matches, the bot may perform additional analysis to refine the results. This could include ranking the matches by relevance, filtering out less relevant information, or conducting deeper contextual analysis to ensure accuracy.

Displaying Output Finally, the bot presents the answers or information to the user in a user-friendly format. This can include displaying the relevant PDF passages, summarizing the content, or providing direct answers to the user's query.

5 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : python 3.11 and node js.

6 Summary

1. We have successfully implemented a working Space app for assisting Astronauts and NASA Space engineers to greatly help their accessibility of NASA's technical documentation.

2. We used Backend Technologies like Django and Python with LLM libraries like langchain, PyTorch and Tensorflow. We also used Front End Frameworks like React with Javascript for implementing the Website.

3. We learnt quite a bit about the NASA Space documentation and its many intricacies in the process which in turned motivated us to create the app.

4. We hope that this work can prove itself to be useful at least in concept to fellow Engineers and developers alike.