The Rig Veda APP's

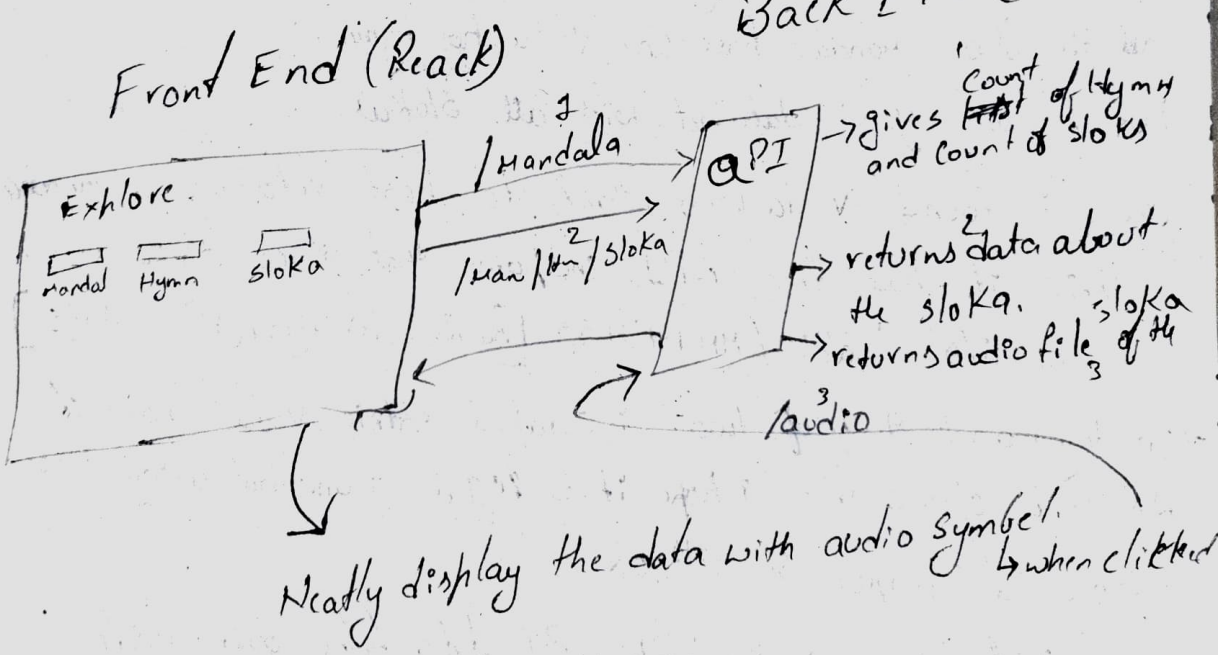
Sept 28 The initial idea was to get the dataset and make an veda Explorer that anyone can explore, The Rig veda and get any slok with audio and translation.

The Development plan is being made, but then i got an idea on adding chat GPT like bot to it, for only answering veda question.

I got idea like trainning LLM with the dataset, but i don't have resources to deploy it, so, i got an idea like

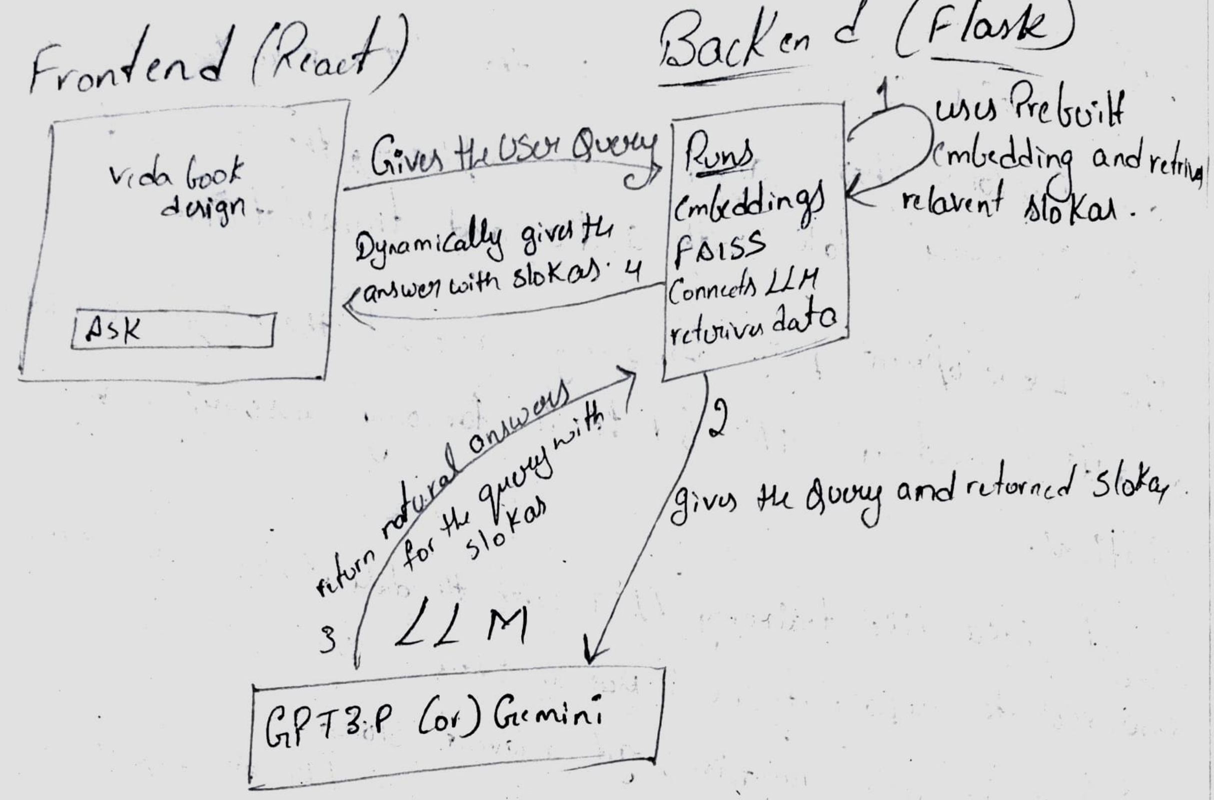
→ Based on user question, get relavent sloka
→ feed it to get natural answers from LLM like GPT 3.5/gemini Lite.
→ Display sloka and answer.

The structure. for veda Explorer. (Migth change through development)



* This uses Json file as Rigveda dataset not mongo DB As it requires paid subs- for high traffic

Structure for chat Bot:



Developement Until Now:

-> while searching for Rig veda dataset mostly (Commonly) i found a data set where only mandal 1 sloks were fully their all the other mandals have one sloka per hymn.
But i want good data set with all slokas.

-> Then i found veda web, thanks to host reference at announcem

-> Their it has free API endpoint that gives sloka and it full details "api/MMHHHSS [mandaal (M), Hymn (H), sloka (S)]

-> on day 1 and Half of two, i scraped entrie veda from it As it is open source, i hope it is legal, i am not using it for commercial purpose so it's ok.

-> Then i parsed all the RAW API data files and made initial per mandal .json files and merged them for final-dataset.

-> Later i got audio file dataset from RV-audio Repo from github, ~~the~~ but it has all sloks combined into one mp3 per Hymn.

-> I designed a formula to split the entire Hymn .MP3 like into single few sloka MP3 files using slokas count and length of each sloka, i am middly satisfied with the result but it is good for now.

-> then on evening of Day 2 i designed and implemented React + flask communication.

-> At first i used MongoDB to store the dataset but it is more then 16MB so i split it for mandal

-> Later when i search sloka it became difficult and traffic increasing, so i made it local JSon files and backend directly accesies it.

Data set structure (For Mandal)

```
{ "mandala": 1,
  "hymns":[
   { "hymn_number": 1;
     "stanzas" : [
       { "stanza_number": 1,
         "location": "01.001.01"
         "Sankrit": " ...",
         "Translitoration": " ...",
         " ... _scheme": "IAST",
         "Translations":{
           "griffith": "...",
           "macdonell": "...",
           "oldenburg": "...",
       } "Padas":{ {... }}
}; }
```

→ Finally at last of second day : got working model for "IvE".

Day 3

Today : focused on creating Embeddings for the dataset to create the chat bot

→ All the Morning : spent Coding Embedding creation using gemini-embedding-v1 but it ~~didn't~~ reached limits before embedding all. (it took half day to realize)

→ So : switched to local model (bge-base-en-v1.5) it took 3 hours to create all embeddings for each sloka

→ yes : embedded each sloka (1024) rather than combining all slokas in Hymn into one embedding..

→ Then : ~~created~~ merged all embeddings to create final FAISS index and slokas_mapping.

→ Then : tested the ~~En~~ index Semantic Search using Py Script.

- : have given a search query
- It encodes it with the model(embeddingmodel) → creates vector
- Compare/Search the vector with FAISS index and gives 3 indexes
- Retrieves Text using sloka-mapping (returns the slokas)

→ It performs very good, it giving slokas related to the search term.

Journal continues / Documentation of development.

→ Today : only focuses on chat bot creation (backend) only.

→ I changed backend file structure to modular approach.
  ⇒ from one file to two folders/modules

  ↳ chat_bot
  ↳ sloka_explorer.

each has it codes that handles the api and all of them integrated with app.py and it's routes.

* The chatbot processing flow is as follows.

→ User gives the query to chatbot.

* The driver function (get_answer) will handle all the process
* first what the user want's to will extended using extract_intents_gemini
* based on the intent the below functions will perform.

  ↳ Fetch sloka by location
  ↳ Semantic Search
  ↳ Answering question

all the results will be the context for LLM (gemini), along with query and sloka's retrived a summarized answer will be returned by LLM and it will be returned to frontend.

* Now After testing the backend with Postman via Coustum API Call : Move on to frontend.

-> Before that i moved all the data to backend/data folder.

* frontend :

Created frontend Prototype. It took me like aweek because i got Sick for few days

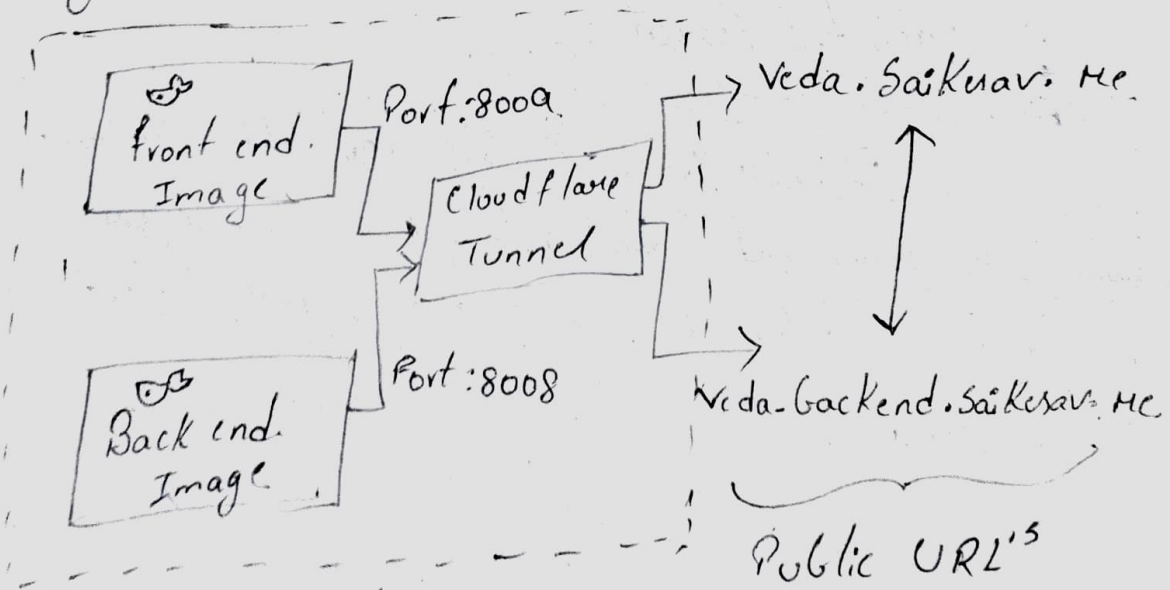
-> By 9th and 10th the initial version of the frontend was built. The design is very simple and easy to understand later i will change/redesign ~~their~~.

-> 11 and 12th and Continues.

Tested Both components invidually and integrated them. I used individual docker files

-> After testing i want to deploy it in my Server.

So i changed dockerfiles and created this/below deployment structure.



Deployment Method

I used Docker Orchestration technique to make a way of integrating multiple Containers to work/communicate under single Network.

Approach:

I used Docker to Containized my front end & Backend.

→ As i already have code to make individual Image's now i tried Combined by images into one image. But the overall size has become too much to be a single image. This is a Problem (Centeralized Image was 8GB+)

→ This is too much and creates a Overload in the System and also it is not an efficient way of using Docker.

→ So i change the structure. I Created two Seperate Docker Images for frontend and Backend Then i

frontend → 1.2GB

Backend → 6+GB Including audio files

used a Separate cloudflare Tunnel Image As shown in the side Drawing and Created a Docker-compose file to Common Bridge network.

→ Then Deployed the Build both Images and Uploaded to Docker Hub and pulled into the Server and built the Composed Image and hosted via the URL veda-saikiran.me

From Now on i started using Git-version control As (V1.0) has developed and cleaned for without any Sensitive data like API's in them. So i used .gitignore & .dockerignore files. to tranfor sesitive data from the Code. If still you found any API key's don't use them Please. Else your

* Applying Threading to Backend

~~From~~ ~~15 to 21~~

Concurrent User Handling

* ~~15-17~~ :

As Prototype version is Ready. And also on 13-14 : implemented session control and logging features. After deployment : tested the website and found when multiple users logged in at once and used the AI the answers are mixing up and results in error from backend.

-> So used Python Threading and to create multiple treads of the backend instances and modualrized resource sharing between instances thus acheiving multi thread performance,

when user access the api for like chatbot or something then ~~based~~ : separate thread will be created for that process and handls the request.

* By the Threading and Resource sharing : achieved.

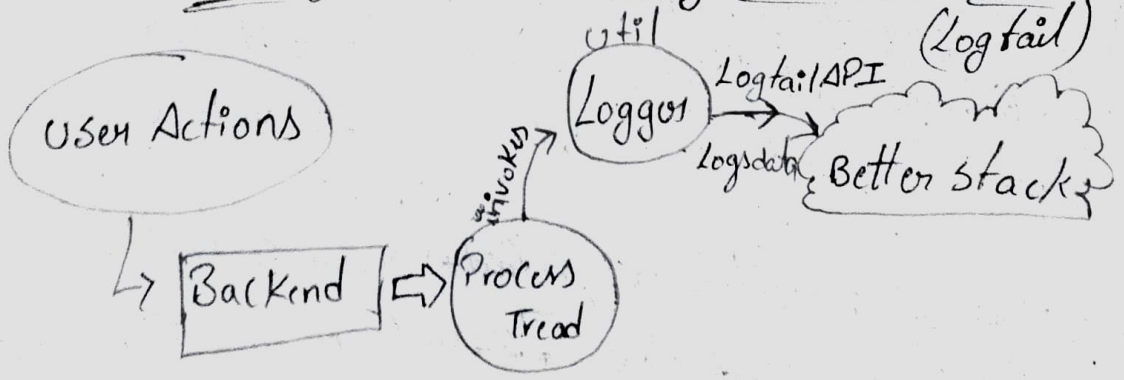
-> User Isolation

-> Efficiend Memory Usage.

-> Resource sharing.

---

* Centeralized cloud Logging with Better Stack *



After Deployment : want to know how Semantic Search, chat BOT responsed to user, is the results are upto the user required. For that i developed Cloud logging using better stack.

-> I used logtail API token and created a source point (HTTPS) requests to send the logs. These are highly Encrypted channel. So no data will be breached.

* I log's the below data into the server from the user.

-> Anonymized IP address
-> Total time to handle the request.
-> User interaction data
-> System responses

for more indept details see Backend/utils/logging_utils.py.

After developing these features i deployed them used Previously mention Orchestration method.

For now initial version of the Project is done From now on refinement and redesign/optimizing the features is Pending.

Phase 1 development Done.

version 1.0

* From Now On every Code change is recorded.
* You Can use Commit History to view the changes.

* Phase 2 : UI/UX Refinements (oct 15-17, 2025)
  -> Redesigned HomePage and added quotes
  -> updated veda Explore UI and added Progress Bar
  -> Homepage animation is not up to mark So Removed it, improved chatbot and Semantic Search UI.
  -> Reduced LOC (Line of Code) in Frontend.

* Phase 3 : Feature Enchantment (oct 17-18, 2025).
  -> Now i noticed the audio (sloka audio) files it's size is 1GB, it is increasing backend size. So i upload them int GitHub. (Reduce backend size 8 -> 7GB)
  -> I add audio file retrival Module from Github into My backend. and no frontend change as same working.

  -> Today while Scrolling reels i saw a video when Someone moving an object with hand gestures, their seeing that i got an idea, why do i add an object in the webpage that user can interacte with So i search google and found a 3D Model of OM. i put that homepage.
  -> Then added new images in the webpage as icons.
  -> Till Now My backend is about 6+GB and frontend is about 1GB.

→ The 6GB is because for Semantic Search model i am using "bge-base-en-v1.5" it's about 480MB and it requires multiple variants of model to run in Docker Image in the next phase i want to optimize this and optimize the docker Image.

Phase 4 Production Readyiness: (18-19, 2025)

→ Today entire Semantic Search model is changed. old model alone takes upto 480x3 MB + other dependencies.

→ To reduce that i used all-MiniM-L6-V2 a 90MB model. Smaller then base-en ~~but~~ and performs lower but i tested the smaller dataset using the Mini it perform upto the standard. i satisfied with the performence in testing so implemented it for entire dataset.

→ First it created embeddings for each sloka using newer model and created new FAISS index and transfered the ~~to~~ index and embeddings to backend and changed backend code ~~too~~ use newer model. The performance is good. but Semantic Search formula giving errors so i changed it to adapet to the newer model.

* Reducing Backend Docker Image Size:

→ After using lower model for Semantic Search Docker image Docker Image size reduce 6 → 3.5 GB. But still i want smaller size so i rewrote DockerFile and reduced Imag to 2.4 GB. For more info see Commit History (dbfc68, 996417)

* Frontend : Added Credits & Privacy Policy Pages.
if you see those page for in details info about those website.
→ Reduced 3D Model texture file size. 6 MB → 2 MB so that every time page loads faster.
→ changed Title from Rig veda to Eternal veda.

* Feature Completion
→ added Navigate button for sloka Explorer to go to next sloka.
→ changed Frontend Color Palette.

For Backend : implemented API Authentication.
→ Till now my backend is exposed to internet anyone can access LLM, Semantic model and entire dataset this is Risky as my data and system resources can be breached.
So : implemented API Auth, i used Decorator and Auth Module to check every API request.
→ I set up an API key in frontend and same in backend. on every request backend checks API key is matching with it's own key and returned the result (error) if no else. gives access to the request resources.

For now I think this is the Final Phase, so i merged both backend & Frontend Git Head's into one and Published the Repo into Git Hub.
→ Don't worry every API key in Commit History has been stopped and it will not work :)

Thanks For Reading This Documentation.

The One Month Development Journey.

This development Journey is a very special one in my CS life, it teached me new things and i learned a lot ~~abi~~ about Sanatan Dharma while building this website. Even if i not win, still i will not have any regrets, because, i think. this Hackathon is not about winning, it's about giving Eternal Knowledge to others. I am very thankfull to be a part of it. I also thank "Indra in Pixels by Ashris" @indiainPixels for inspiring me to create this.

Still if you have any Doubts or if you want to Know more about this project or want to collaborate with me in future or want to be Friends - you can Contact me via my website https://www.saikesav.me.

:)

Saikesav