GovHack[1]
Team V-SARC
An Accurate and Trustworthy Chatbot for Data Interactions

Proof of Concept
User Manual

Date: August 31, 2025
Project: NTGAssist
Version 1.0

[1]GovHack and the GovHack Logo are registered trademarks of GovHack Australia Limited. The GovHack logo is used here for reference purposes only. All rights in the GovHack name and logo are reserved by GovHack Australia Limited.

Contents

Preface

This user manual is provided as a guidance to set up the *Proof of Concept (PoC)* for the presented challenge of **An Accurate and Trustworthy Chatbot for Data Interactions**.

The PoC is intended to demonstrate the objectives of the challenge including dynamic selection of data sources, conversational capabilities of the *Retrieval Augmented Generation* use of AI to overcome common issues associated with the generative AI.

Instructions include below are for developers, evaluators and spectators who wish to explore the software functionalities purposed in the solution for the challenge. Certain pre-requisites are listed to ensure the host system is capable of spinning up the PoC.

Disclaimer

This software and accompanying documentation are provided strictly for proof-of-concept and demonstration purposes only. The software is experimental and may contain errors; it is not intended for production use. The use of any copyrighted logos, including the GovHack logo, does not imply endorsement, sponsorship, or affiliation. All rights in such logos are reserved by their respective owners. Use of this software is at your own risk, and the authors accept no liability for any damages resulting from its use.

Team Contact Information

Name	Title	Contact information
Team V-SARK		
Ayesh Jayasekara	Team Lead	Email: ejkpac@gmail.com
Chathura Janadara	Member	Email: jim.smith@demo.com
Ravindu Supun	Member	Email: jim.smith@demo.com
Vigneshwaran	Member	Email: jim.smith@demo.com
Kavini	Member	Email: jim.smith@demo.com

Prerequisites for Host Systems

In order to ensure the host system is capable of running the **Large Language Model *LLM*** and associated system components, following specifications are recommended as a minimum.

- RAM – 16GB available space minimum
- Storage – 6GB Free & Available storage
- Processor - Intel or Apple M series recommended

Software Requirements

In order to setup the sources, following software installations are required.

- Java Runtime - 17 or above
- Homebrew
- NPM latest version
- GIT
- Docker
- Maven

LLM Setup

In order to avoid third party paid service LLM AI service providers, the PoC is designed to use a localised version of LLM. Our team has chosen **Mistral** for these purposes since it offers right balance of reasoning capabilities and resource intensity. However, by nature the model can be swapped to more powerful choice if resource permits and will not theoretically impact the purpose of the demonstration although such was never test due to limited time availability.

Follow instructions below to set up *Mistral* locally.

Verify installation:

```
brew --version
```

Install Ollama CLI via Homebrew:

```
brew install ollama
```

Verify installation:

```
ollama --version
```

Pull required model and embedding model:

```
ollama pull mistral
ollama pull nomic-embed-text
```

To test running model execute following:

```
ollama run mistral "Hello Mistral"
```

Your LLM model and embedding model is now ready to be used. Proceed to next step.

Vector Database Setup

Vector database contains the AI readable metadata in a form of multidimensional vector space. Higher the dimensions, higher the accuracy of information retrieval. However, in the local setup with extremely limited hardware resources the dimensions are kept to minimal viable default provided by *nomic-embed-text* of **768 dimensions**.

The information retrieval phase embedded to the information retrieval service *(described in next chapter)* is set to distance calculation method of *cosine* accordingly.

To spin up a vector store:

```
docker run -p 6333:6333 -p 6334:6334 qdrant/qdrant
```

This will sping up a vector database to be used in next step.

Ingestion & Chat Service Setup

As with any RAG based solution, a vector database is a critical component of the architecture. It serves the purpose of the main *knowledge base* for the system.

To simplify the setup process, the document ingestion component and chat completion component are built into unified micro-service. This is written in *Java* using *Spring Framework* as it was deemed comfortable with team expertise of the members.

Following are the steps to setup the microservice locally.

Clone the sources from GitHub Repository:

```
git clone git@github.com:AyeshJayasekara/GovHack-2025-Backend.git
```

Build the executable from sources (Execute following within the cloned directory that contains the *pom.xml* file):

```
mvn clean install -DskipTests
```

To spin up the service from the JAR file generated:

```
cd target
java -jar service-1.0.0.jar
```

At this point, the service should spin up. It is important to note that the Mistral service is required at the startup. If any problems persist, repeat instructions above and make sure that service is healthy before attempting to troubleshoot.

Populating the Vectorised Knowladge Base

At this point the local setup is ready to be populated with metadata of the system. To simplify this process a meta-data file has been prepared for following data sources as outlined in the challenge.

- **Data Set 1:** Freedom of Information Statistics on Data.gov.au
- **Data Set 2:** Employee Leave Tracking Data on Kaggle

See GovHack 2025 Backend - DATA Directory for metadata we prepared as part of data preparation step.

Once you are ready execute below to instruct the backend service to populate the knowledge base.

```
curl --location --request POST 'localhost:8080/rag/ingest'
```

Congratulations! You are now ready to explore the capabilities of the PoC system.

If you do not want to set up the front-end application to test the system in user-friendly way, you can interact with the system from terminal in the form of HTTP requests.

Use the following cURL code snipped as a template and alter the ***question*** parameter with your desired prompt to retrieve a response from the system. As to be expected, the terminal view may not be most reader friendly so we recommend to use the provided frontend system.

```
curl --location 'localhost:8080/ask' \
--header 'Content-Type: application/json' \
--data '{
    "question": "What is happening with leave patterns in my team?"
}'
```

User Interface Setup

Our team has come up with a friendly way to interact with the RAG AI solution. Again, given the expertise of team members we have chosen *Expo Framework* given its versatile capabilities to quicker go-to-market. Follow the instructions below to set it up on your system.

Clone the sources from GitHub Repository:

```
git clone git@github.com:AyeshJayasekara/GovHack-2025-Frontend.git
```

Navigate to the cloned repository folder to install the required packages:

```
npm install
```

Once done, execute following to run the application:

```
npx expo start
```

A QR code will be displayed on the terminal, scan it to navigate to the demo mobile application to interact with the system.

Note: You mobile device must be on the same network (e.g. connected to same WiFi). Also, you are required to install the freely available mobile application named *Expo Go* from your respective mobile application store (See below).

- **Apple Store:** Expo Go on the App Store
- **Google Play Store:** Expo Go on Google Play

Post Cleanup

The downloaded models and related artifacts maybe deleted from your system once you are satisfied with the demonstration.

The LLM model you downloaded may have consumed a lot of storage from your system. Execute following to remove it:

```
ollama rm mistral
```

Similarly, the vector datastore may also be removed from your system. You can use *Docker Desktop* if its installed in your system. Or otherwise execute following:

```
docker ps -a
```

Above will list out any docker containers you ran previously. Note the container ID and execute following with subsituting container ID placeholder.

```
docker stop <ContainerID>
docker rm <ContainerID>
docker rmi qdrant/qdrant
```

Other components including the user interface and unified microservice can be deleted by removing the relevant folders where you have cloned them to.

Thank you! Hope you have enjoyed working with a smarter agent to crunch your data!