KARNA - BOT

For smarter conversations

TEAM 3X | Github Link

Table of Contents

Scope of the document

This document will provide the detailed explanation of Karna bot, a chatbot system helping end customers to interact with business solution providers to use their services. These interactions are based on conversations done on well known messenger apps.

The Idea of Karna bot was started to provide innovative solutions for an open mobility challenge hosted by Namma Yatri.

About Karna Bot

Karna Bot is a process automation tool which can help users to complete certain business processes like booking a ride over conversations. Karna Bot helps orchestrate conversation responses as a task of a process.

About Namma Yatri

Namma Yatri is an open network mobility application built with the idea to provide multi-modal service to the commuters without the involvement of any middlemen.

Problem statements

1. **Booking Rides Without App**

   Mobile/Website applications provide great UX to interact with the businesses. But from the end user persona installing multiple applications for different providers to use some of their services tends to be a hassle.

   Also, in a country like India where the majority of the population is not well familiar with websites and applications, using digital services like booking a ride will become difficult on applications.

2. **Map Cost Improvements**

   Premium map APIs are used to auto-complete user pickup and drop locations. Goal is to reduce these map costs.

3. **Customer Acquisition**

   To grow the number of new users to use Namma Yatri provided services.

Solution approaches:

1. **Booking Rides Without App**

   Humans being social creatures feel connected when they have conversations. And booking a ride with sharing certain details as a part of conversation would always be a simple, effortless choice for the majority of us. But as said for businesses it would be unviable to place humans to conduct these repetitive conversations as a counterpart.

   a. Automatic messaging systems like Karna Bot can help to manage these conversations on scale with the least amount of human interventions.
   b. Whatsapp a well known messenger application with a huge user base can be integrated as an user interface with Karna bot to complete the ride booking.

2. **Map Cost Improvements**

   a. Karna bot can interpret the location data from the conversation and forward the same to the backend servers.

b. Choice of Whatsapp messenger as the user interface will be advantageous here as it already has the maps feature for the end customers which will help them to pick their desired pick up and drop location for the ride booking.

3. **Customer Acquisition**
    a. Using conversations as a solution for most of the user queries and to provide services will subtly lead to the increase of user count.
    b. Messenger apps will help us in removing some of the tedious steps like installing an app, logging in for a session just to book a ride.

Karna Bot Features

- Ride booking.
- Ride cancellation.
- Multi-language conversations support.
- Starred (Favorite) locations management.
- Feedback provision for the ride taken.
- Ride status tracking.
- Namma Yatri's open data redirection.
- Support, FAQ sections and more.

System Features

- Integration ready for multi-platform messenger applications.
- Scale ready for multiple conversations with different users.
- Conversation Isolation.

- State aware conversations i.e. maintaining state data for each conversation active.
- Integrated message template engine with multi-language support.
- Horizontal scalable.
- Auto close conversations after threshold time of user inactiveness.

Technicals of Karna Bot

Technologies used

1. **Camunda:** Camunda is an open-source workflow and decision automation platform offering process orchestration capabilities. It uses BPMN 2.0 and DMN for business and decision process models.
   Conversation workflows for Karna bot are built using components of BPMN 2.0 and DMN. Hence, onboarding a new conversation workflow for Karna bot is as simple as sketching out a new business process using BPMN diagrams.

2. **Redis:** Redis is an open-source, in-memory data structure store used as a database, cache, message broker, and streaming engine.
   Karna bot uses Redis as a cache layer to cache multilingual message templates and other data.

3. **MongoDB:** MongoDB is an open-source schema-less database used to store data as a document data structure composed of field and value pairs.
   Karna bot uses MongoDB as a data-store to save records of users and message template entities.

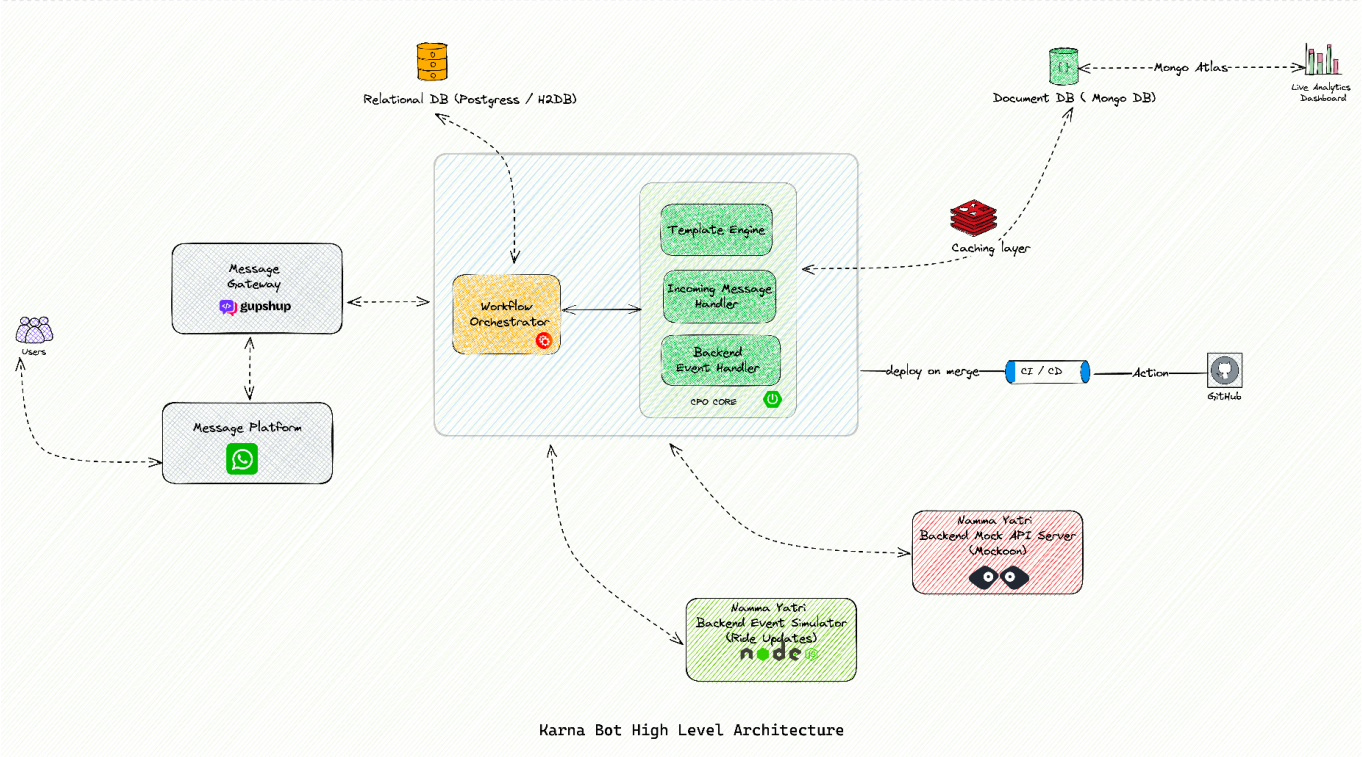
4. **Spring Boot:** Spring boot is an open-source Java based framework used to create microservices.
   Karna bot is built using Spring boot based Java app embedded with Tomcat web server and Camunda process engine.

5. **Mockoon:** Mockoon is an open-source project which helps in designing and running mock REST APIs over a server.
   Karna bot requires interaction with a backend server handling all the business logics such as finding nearby rides. With the help of Mockoon, mock REST APIs were created to imitate business logics for Namma Yatri ride booking service.

6. **Gupshup:** Gupshup is a conversation messaging platform providing APIs to help enable Whatsapp messaging for businesses.
   Karna bot uses Gupshup's REST APIs to build and deliver conversation responses as a Whatsapp message to the end user.

High Level Design

The KarnaBot is built using a microservices architecture, with each service responsible for a specific aspect of the chatbot's functionality. The system architecture is designed to be scalable, fault-tolerant, and resilient.



The following are the key system components

Message Platform : This component acts as the primary interface between the user and the bot. Users can send messages or interact with the bot through various platforms such as WhatsApp, Slack, or Telegram.

Message Gateway : This component receives messages from the message platform and forwards them to the bot engine. It is responsible for handling platform-specific logic and converting it to a generic format that can be understood by the bot engine. For the WhatsApp platform integration, we have used the gupshup.io platform.

Workflow Orchestrator : This component manages the workflow of the bot and persists conversation state. It is responsible for deciding the next state of the conversation based on the user's input. We have used the Camunda BPMN engine for this component, which has its own relational database to store the state of the conversation.

Template Engine : This component generates messages and responses to user queries quickly and efficiently in multiple languages.

Incoming Message Handler : This component handles incoming messages from the Message Gateway.

Backend Event Handler : This component handles backend events from the service provider platform, such as Namma Yatri, to receive users' ride updates.

Document DB : his component maintains user configuration and template configuration and helps in the analytical dashboard.

Caching Layer : This component caches user and template context to provide a faster response to the user.

Live Analytics Dashboard : This component provides insights into the usage of the Karna Chat bot, such as the number of users, user trends, and active conversations. This information can be used to improve the bot's performance and user experience.

Namma Yatri Backend Mock Server : This component is responsible for simulating the backend APIs of the service provider platform, Namma Yatri, and providing the simulated data to the bot. The mock server provides an interface that mimics the behavior of the actual backend APIs, allowing the bot to interact with it as if it were interacting with the real APIs

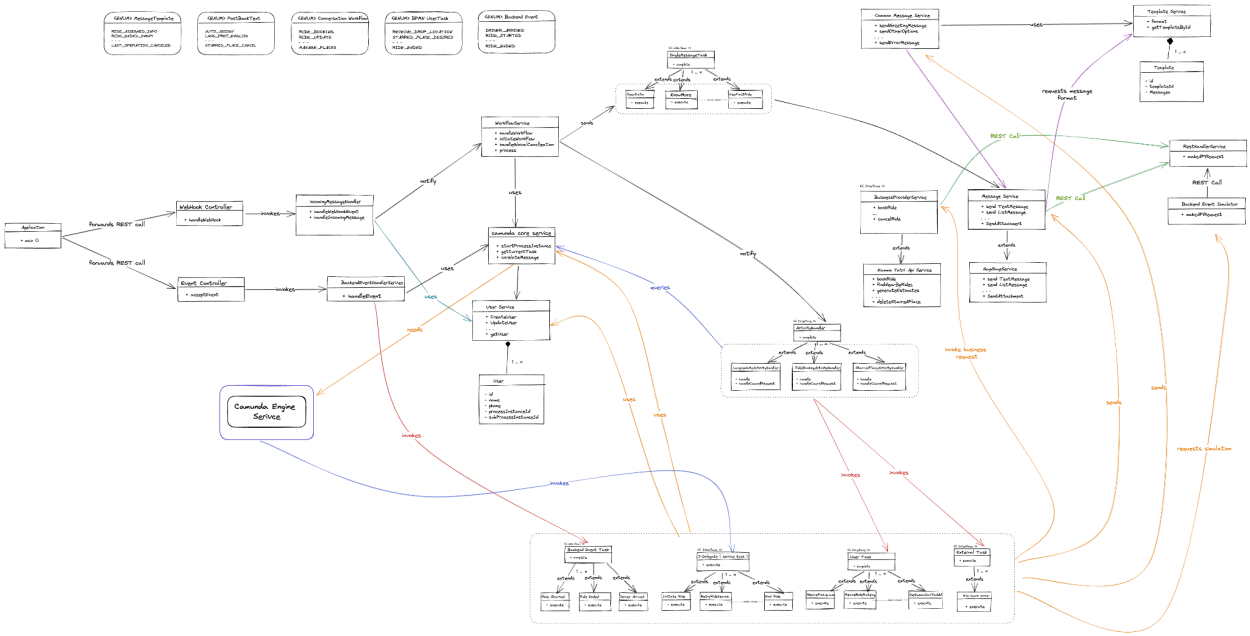
Backend Event Simulator Host : This component is responsible for simulating the backend events from the service provider platform, such as Namma Yatri, and sending the simulated events to the bot. The simulator host provides an interface that mimics the behavior of the actual backend event streams, allowing the bot to receive the simulated events as if they were real. In this case we simulated the ride updates event of Namma Yatri.

CI / CD Pipeline : to deploy the code to the cloud and manage the deployment and versioning of the code. We've used github actions to handle CI/CD

Low Level Design

1. Class Diagram

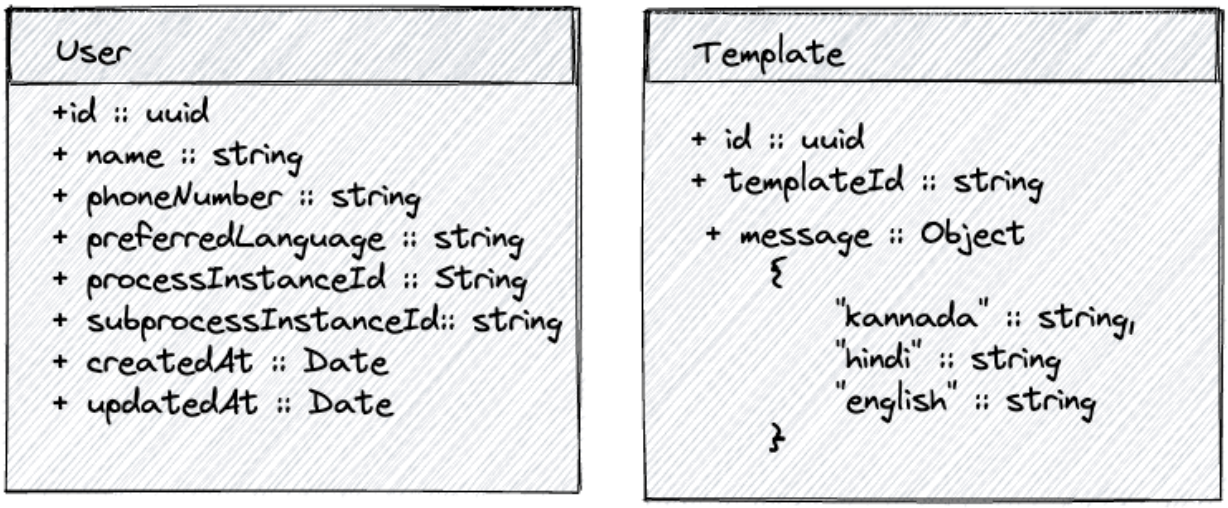
The low-level class diagram illustrates the implementation details of the Karna chatbot, including the classes and their relationships. (Image link in References)



Karna Bot Low Level Class Diagram

2. DB Schema

The low-level DB Schema diagram illustrates the implementation details of the schema and it's relations



Karna bot DB Schema

End users onboarding strategy

Following simple options can be availed to onboard a new user for interacting with Karna bot. The main goal here is to remove the friction of user onboarding on to the business.

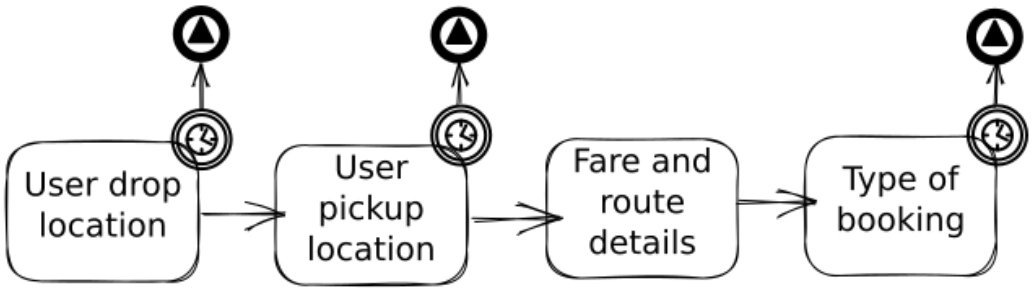
1. **QR codes:** QR codes will be placed at the city hotspots, popular pickup points, and marketing campaigns. On scanning this code users will be redirected to the messenger app like Whatsapp with a greet message prefilled in the text area, ready to send on a click to the business contact.
2. **Web Links:** Web links can be placed on the site or can be shared through other means. These links would be working similar to the QR code scans

Ride booking workflow overview

How booking workflow works

Booking flow as a process can be divided into 3 different subprocesses

1. **Acquiring ride data from the user.**



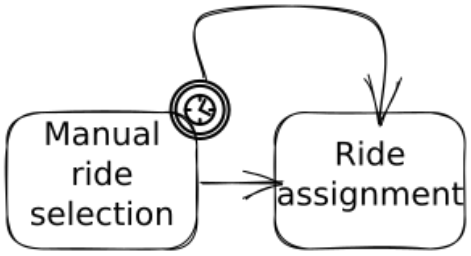
This is the beginning of the booking journey where users will be sharing all the necessary details required for the booking. The tasks involved in this sub flow are

- Drop location request - A message requesting users drop location with the list of favorite places if present will be sent to the user.

  Users can share the pinned location from Whatsapp or can select their favorite places from the list if they have added any in the system before.
- Pickup location request - This task is similar to the drop location request and captures the location as user pickup.
- Calculation of fare and route details - A REST call is performed to the Namma Yatri backend mock server to calculate the fare for the selected route.

  The response with route and fare details is formatted as a message and sent to the user.
- Booking type selection - With fare and route details user will be provisioned with two different booking types to move forward as button options in the same message.
  - Automatic ride assignment.
  - Choose a ride manually between multiple ride options.

2. Ride assignment based on the type of booking.



From the options provided users would either proceed with Automatic ride assignment which is faster as the ride gets assigned without

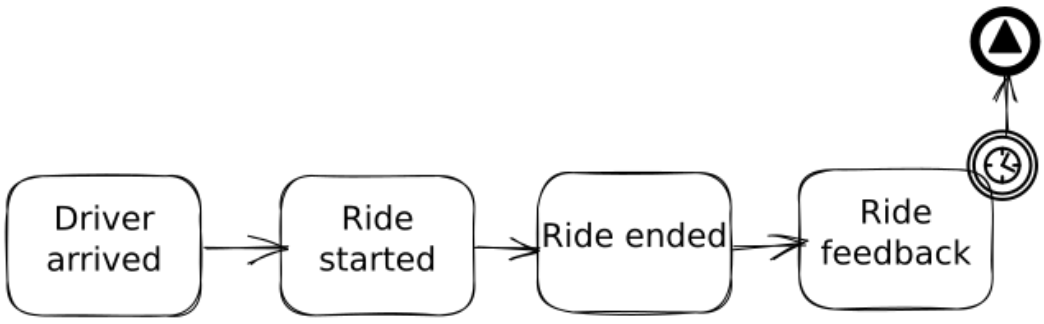
any consensus between driver and ride user or

Choose manually from the given options. A REST call to fetch the nearby rides is executed with the type of booking as a request parameter. Tasks involved in this subprocess are

- Manual ride selection - For booking type manual users will be provided with the list of rides found nearby comprising of the ride fare quoted by the drivers in the message.

  Users will choose and send one of the preferred rides from the list.
- Ride assignment - Regardless of the booking type a REST call to book the ride for the user is made to the mock backend server. On the success call the users are informed on the message about the same.

3. **Ride updates to the user.**



This subprocess is about notifying the users about the ride updates post booking. The webhook calls for all these updates would be ideally called from Namma Yatri backend server, currently there is an event simulator which will be updating the Karna bot about the ride updates on behalf of the backend server. This subprocess includes the following tasks

- Driver arrived - Notifying the users about the driver's arrival and requesting them to share the OTP with the driver to start the ride. This OTP is generated during the ride assignment request.
- Ride started - After the OTP validation on the server ride start event hook will be called from the simulator. The Karna bot will notify the same event update as a message.
- Ride ended - The trigger for the ride end event is when the driver completes the ride from their side. The Karna bot will notify the same event update as a message.
- Ride feedback - This message will be sent to the users right after the ride ends requesting the users about the experience and feedback for the ride.

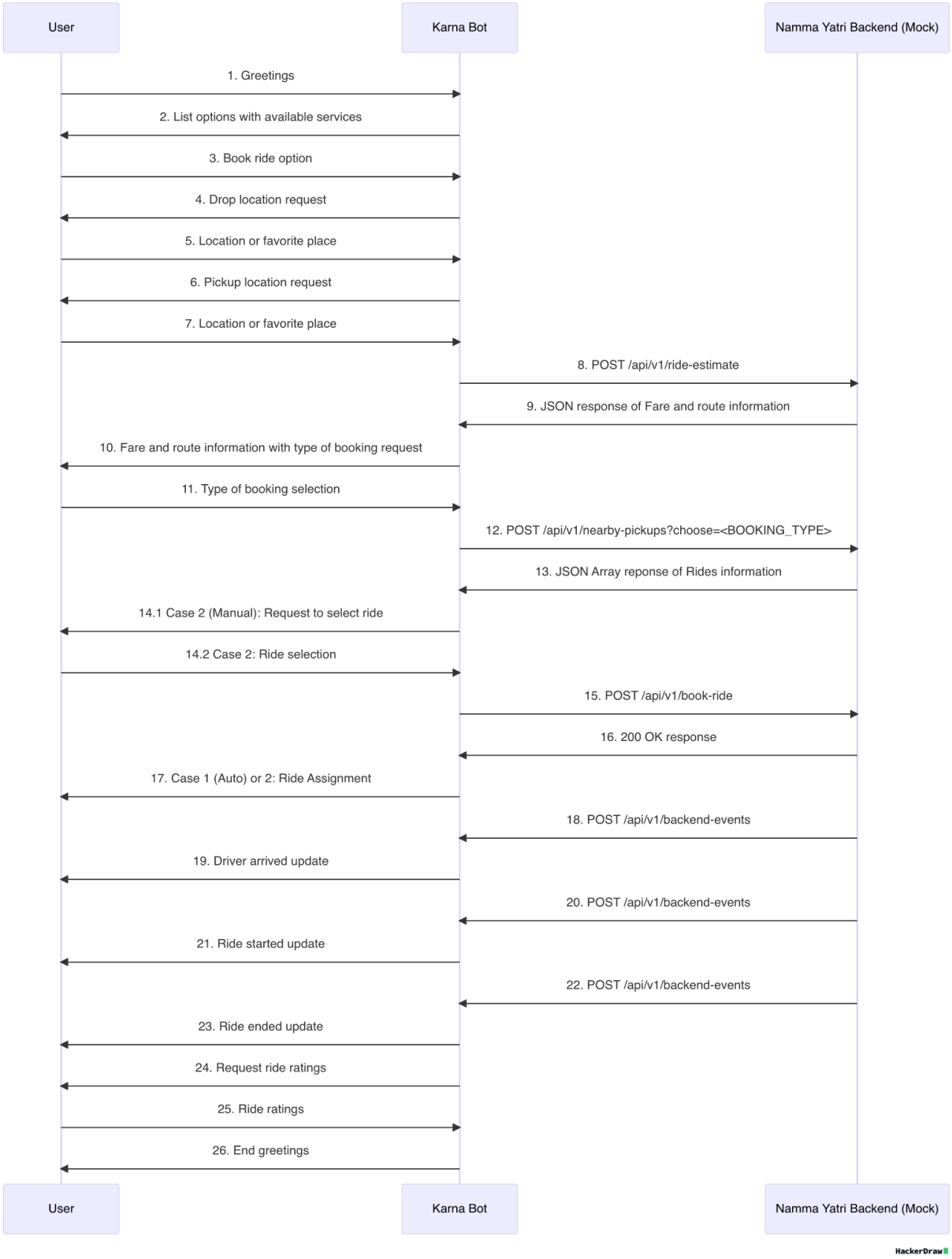
Exceptional cases in the booking workflow

1. **No response closures** - Karna bot will be waiting for a specific time for every task which involves users' response. On reaching this threshold time a message stating “closing the booking flow for no response” will be sent to the users.
2. **Global cancellation** - Karna bot supports global cancellation i.e. after starting any of the business flows like ride booking if the users send “cancel” as a text message the respective business flow will be canceled and a flow cancellation notification message will be sent to the users. Users can start the process again from the main menu options.
3. **Cancel booking CTA’s** - Some of the messages such as ride assignment sent to the users will consist of a cancel booking CTA. Users can use these CTA to cancel the booking process if they don’t want to proceed further with the

process. Result of this is similar to the global cancellation where the underlying process flow will be canceled and the user will be notified about the same.

4. **Retry ride search** - There can be a case where the Namma yatri backend server will not be able to find the nearby ride for the users. Karna bot will be waiting for a specific time for the ride search. If the threshold time is reached Karna bot will start the retry search case based on the following conditions.
    a. If the booking type selected by the users is manual ride then the next retry ride search will be performed with automatic ride assignment booking type.
    b. Karna bot will be retrying for a max retry attempt set and once this threshold is reached, Karna bot will close the flow and notify the user about no rides found and request the user to retry again later.

Booking workflow sequence diagram



Further read for booking workflow

- Booking workflow BPMN :

https://github.com/nishanb/Namma-Bot/blob/master/assets/bpmn/booking_flow-v1.png

- Ride update workflow BPMN :

https://github.com/nishanb/Namma-Bot/blob/master/assets/bpmn/ride_update_flow-v1.png

Other workflows supported by Karna Bot

Starred (Favorite) places management workflow

Users might have certain places which they would like to use more frequently. Searching them on the maps every time will be a time consuming repetitive task. Starred places workflow solves this problem where users can add their favorite locations and provide them a tag or nickname. These starred places can also be deleted if not required.

The addition or removal of the starred places will reflect in the list of starred places shared with destination or pickup location messages during ride booking flow.

Starred places workflow comprises following tasks.

- Desired action request - Users have the option to add a new starred place or delete existing starred place. A message requesting users to choose between these two actions is sent.

Add new starred place -

- Starred location request - A message requesting the user to share the location of the place which has to be starred is sent.

- Tag name for starred location - Users are requested to share the name for the starred location to be stored.
- Add operation notification - A REST call to add the starred for the user requested is executed and users are notified about the status of starred place addition.

Delete starred place -

- Selection of starred places - A REST call to fetch all the starred places for the requested user is executed and a message comprising a list of all the starred places is sent to the users. Users are requested to select one of the options to be deleted. If there are no starred places for the user, a message notifying users about “no starred places present” is sent.
- Delete operation notification - A REST call to delete the selected starred place for the user requested is executed Users are notified about the status of starred place deletion.

Further read for starred (Favorite) places management workflow

- Starred places flow BPMN :
https://github.com/nishanb/Namma-Bot/blob/master/assets/bpmn/manage_starred_place-v1.png

Language change workflow

Support for different languages for application usage is present in most of the prominent applications considering the diverse range of people interacting with it. Karna bot also supports conversations with different languages apart from English. All the workflows supported by Karna bot can also be completed using Hindi and Kannada language. Karna bot is adaptable enough for adding other regional languages.

Users can change the language by completing three simple tasks.

- Preferred language selection - Karna bot provides 3 different language options as a message, users can select one of the preferred languages to move forward.
- Language change confirmation - Users are requested to confirm the language change by selecting yes or no options provided in the next message.
- Change in language notification - Post confirmation of change in preferred language, Karna bot updates the user's preferred language in the datastore. And a notification for the same operation is sent to the users as a message.

Further read for language change workflow

- Language change flow BPMN :

https://github.com/nishanb/Namma-Bot/blob/master/assets/bpmn/language_change_flow-v1.png

Analytics and Dashboard

To support the legacy of Namma yatri's open data dashboards, Karna bot also has some of the dashboards created on its data with the help of Mongo Atlas. The data that can be visualized on these are

- Users registered for usage of Karna bot.
- Preferred languages by users onboarded.
- Number of active conversation Karna bot handling.
- Total active message templates count.
- Bar chart of users onboarding with respect to time.

Karna bot analytics and dashboards are still a work in progress.

Beta tests performed

After the MVP development of Karna bot it moved through alpha testing within the team where all end to end tests were performed on the bot.

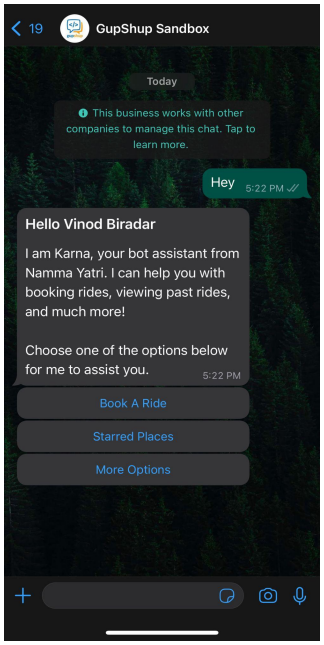
Now, Karna bot is being hosted on one of the self managed servers and onboarding links/QR codes have been shared with a closer community to test it out as end users.

People were impressed with the idea and shared their valuable feedback on Google forms. One of the major achievements for Karna bot was that no users have encountered technical issues (First 50).

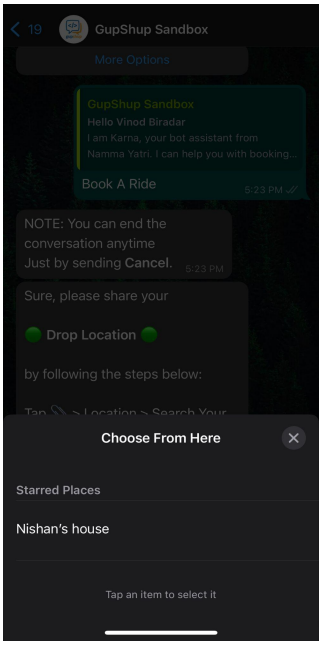
Future enhancements

- Replacing mock server with Namma Yatri server SDK integration
- Sharing driver's live location as message to end users after ride assignment.
- Maintaining conversation history for users.
- Building adapter design pattern for messaging client configuration to easily integrate with multiple message platforms.
- Building admin dashboards to manage templates and conversation flow.
- Integration of alerting tool for app exceptions alerts.
- Integrate with PostHog for product analytics, user analytics and more.
- Manual takeover of bot replies and handling it by humans.
- Campaign management using Karna bot.

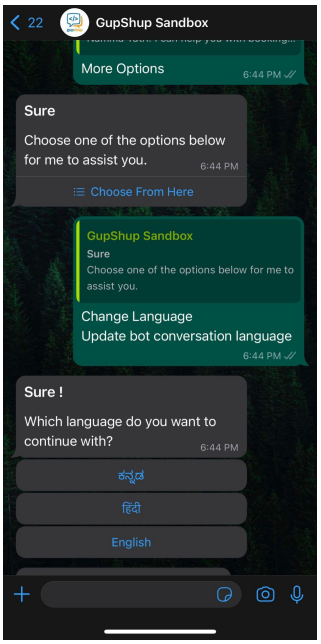
Some Snapshots of Karna bot responses on Whatsapp



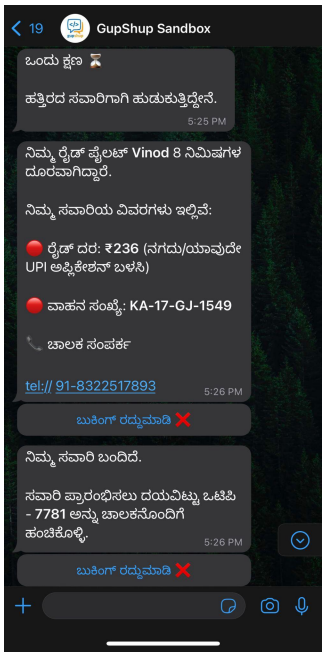
1



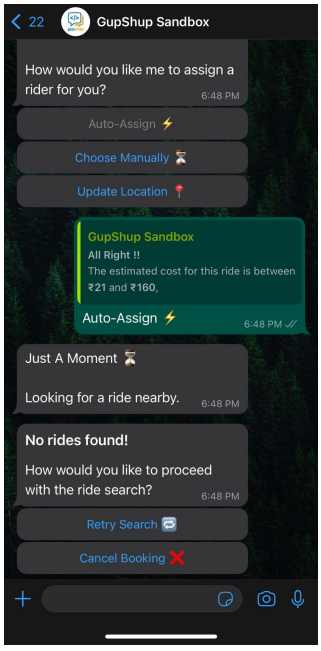
2



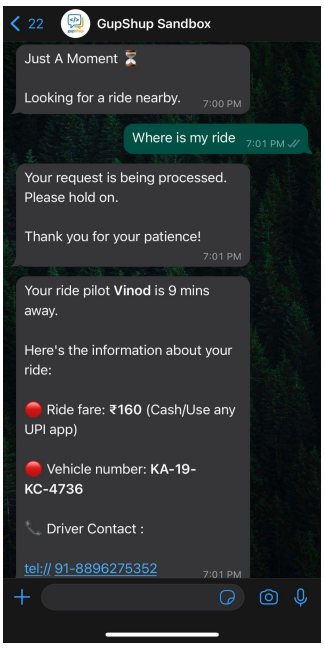
3



4



5



6

1. Greet message with available workflows
2. Providing destination location from favorites
3. Language change message

4. Ride assignment message
5. Retry booking selection
6. Handling invalid user inputs.

Reference links

- **Karna Bot GitHub:** https://github.com/nishanb/Namma-Bot
- **Beta test users feedback:** https://docs.google.com/spreadsheets/d/1hQdV6n86K6z3zfIY9w26ChCvwShFJAp0KP0lBSWvfhM/edit?usp=sharing
- **High Level Design :** https://github.com/nishanb/Namma-Bot/blob/master/assets/design/karna-bot-hld-v5.png
- **Low Level Design - Class Diagram:** https://github.com/nishanb/Namma-Bot/blob/master/assets/design/karna-bot-lld-v3.png
- **Karna bot analytics dashboard:** https://charts.mongodb.com/charts-test-hwppi/public/dashboards/5f6a3bd3-8ed8-44e9-8480-f9915f290cc7
- **BPMN Diagrams :** https://github.com/nishanb/Namma-Bot/tree/master/assets/bpmn
- **Namma Yatri's open dashboard:** https://nammayatri.in/open/
- **Camunda:** https://camunda.com/
- **Spring Boot:** https://spring.io/projects/spring-boot
- **Mockoon:** https://mockoon.com/docs/latest/about/
- **GupShup:** https://www.gupshup.io/
- **MongoDB:** https://www.mongodb.com/

- **Redis:** https://redis.io/docs/