**Challenge 1: Intelligent Search –**

To enable intelligent searches for better searchability via the UHI network

**Objective:** Intelligent search gives the users more personalized and accurate search results, leveraging AI technologies like machine learning, semantic search, computer vision and natural language processing. The objective of building intelligent search using UHI network is to break down silos, allowing information to be extracted from practically any data source, leveraging a common protocol and ecosystem, thereby standardizing the way health is delivered.

**Benefits:**

* Delivers faster and smarter results for the user
* Provides a single point of access to the enterprise content sources
* Allows data to be enhanced, searched, and analyzed in both structured and unstructured formats
* Reduces time spent by a user in searching for information, making healthcare delivery more efficient

**Features:**

* ***Implemented Features***
  + Autocorrect
  + Autosuggest
  + Conversion of unstructured text into structured text
  + Local language support (API – 100% completed, integration WIP – applicable for any language translation)
  + Doctor search and discoverability –
    - Specialty search
    - Location awareness
    - Price recommendation
    - Rating plan (80% facility recommendation and 20% patient satisfaction rating)
    - Doctor picture
    - Mode of consultation (physical vs teleconsultation)
  + Both Android and Web applications developed
* ***Suggested Features (Future scope of enhancement)***
  + Unstructured/semi-structured data (pdf or jpeg/png with presenting physical symptoms by the patient) can be used as search criteria for autosuggestion by system for specialty/doctor to be consulted

**Assumptions:**

* Since we do not have consumer end points build, we will mimic the NDHM APIs in our application, to showcase the leveraging of UHI network for intelligent search
* All the HRPs must implement the new UHI protocol for search for standard usability
* As a POC, a small training data with 20 samples is considered and annotated with these entities. With more data samples the model can be improved

**Technical Design**

1. **Architecture Diagram for scalable data models used in the project**

**Diagram

Description automatically generated**

The Intelligence search code resides in UHI and individual HRP need to implement recommendations provided by the UHI. i.e. for example if UHI request for to provide all facility locations then it has to respond with all facilities with latitude and longitude details. Similarly they need to return the corresponding data for other parameters as well.

**Diagram

Description automatically generated**

1. **Model APIs**

The following APIs have been developed for this challenge track –

1. **Autocorrect: https://intelli-search-csh.herokuapp.com/autocorrect**
   1. This API allows for autocorrect. Input can be a word or multiple words.
   2. For dataset for Autocorrect – words in English vocabulary have been extracted from NLTK Corpus and saved into a txt file
2. **extractEntities: https://intelli-search-csh.herokuapp.com/extractEntities**
   1. This API allows us to convert semi-structured data to structured format. The model extracts the following entities – Hospital, Location, Specialty, and Physician Name from the search query
   2. The Annotated Data is present in annotator.json
   3. The code to train the model is present in <https://github.com/AiswaryaSrinivas/Hackathon/blob/main/SPacy_Training.ipynb>
   4. The model is trained on Google Colab using spacy 3.2.1
3. **Autopredict (Autosuggest):** [**https://intelli-search-csh.herokuapp.com/autopredict**](https://intelli-search-csh.herokuapp.com/autopredict)
   1. The API can be used to autosuggest Symptoms, Diseases and Specialty in Intelligent Search (if we have names of doctors, we can autosuggest doctors as well)
   2. The list of symptoms is extracted from <https://www.kaggle.com/datasets/itachi9604/disease-symptom-description-dataset>
   3. Symptoms, Diseases and Specialty are autosuggested based on similarity of the input text with the set of symptoms, diseases, specialty etc. (if the list are extended as new symptoms are entered – this model can also take into consideration the newer symptoms as well)
   4. API takes two parameters – “text” and “text\_type”
   5. To predict symptoms – pass parameter text\_type:”symptoms”
   6. To predict diagnosis – pass parameter text\_type:”diagnosis”
   7. To predict specialty – pass parameter text\_type:”search”

**How to set-up**

Please refer readme file of each folder.

**Workflow with UI**

1. **Autocorrect Workflow**
   1. **Search page**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

* 1. **User inputs specialist type (incorrectly)**

Graphical user interface, text, application

Description automatically generated

* 1. **Autocorrection for user**

Graphical user interface, text, application

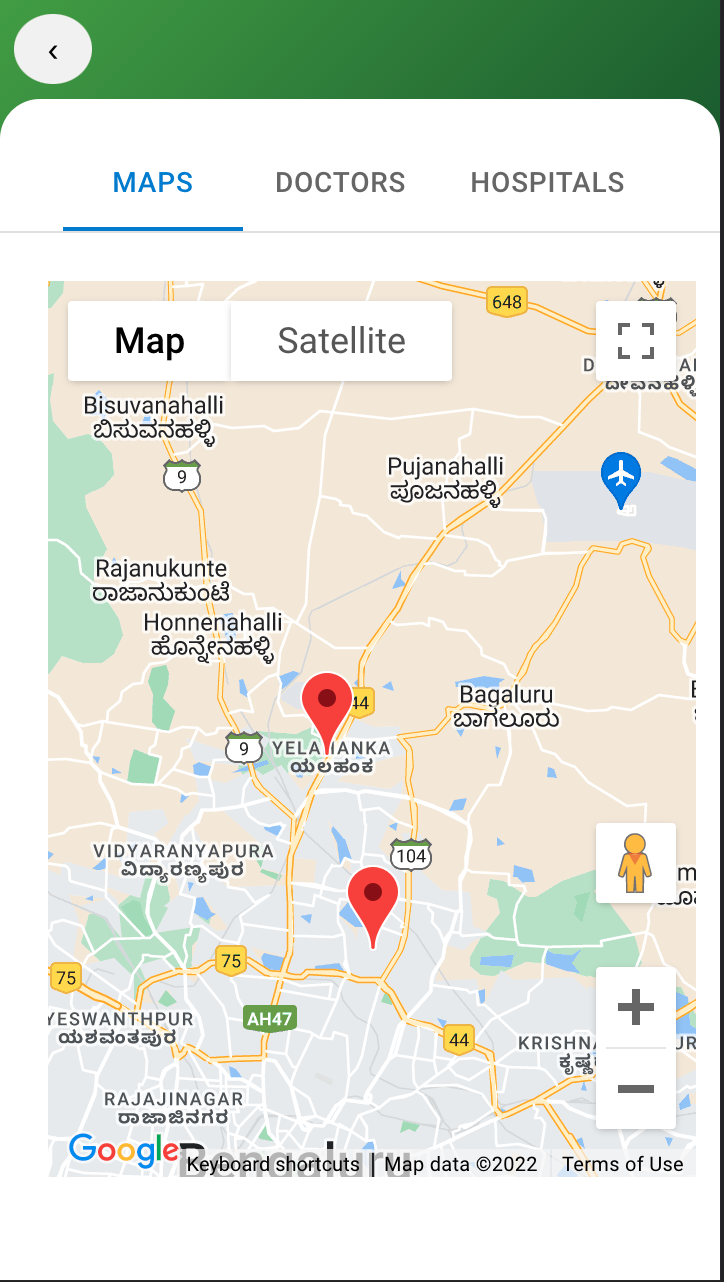
Description automatically generated

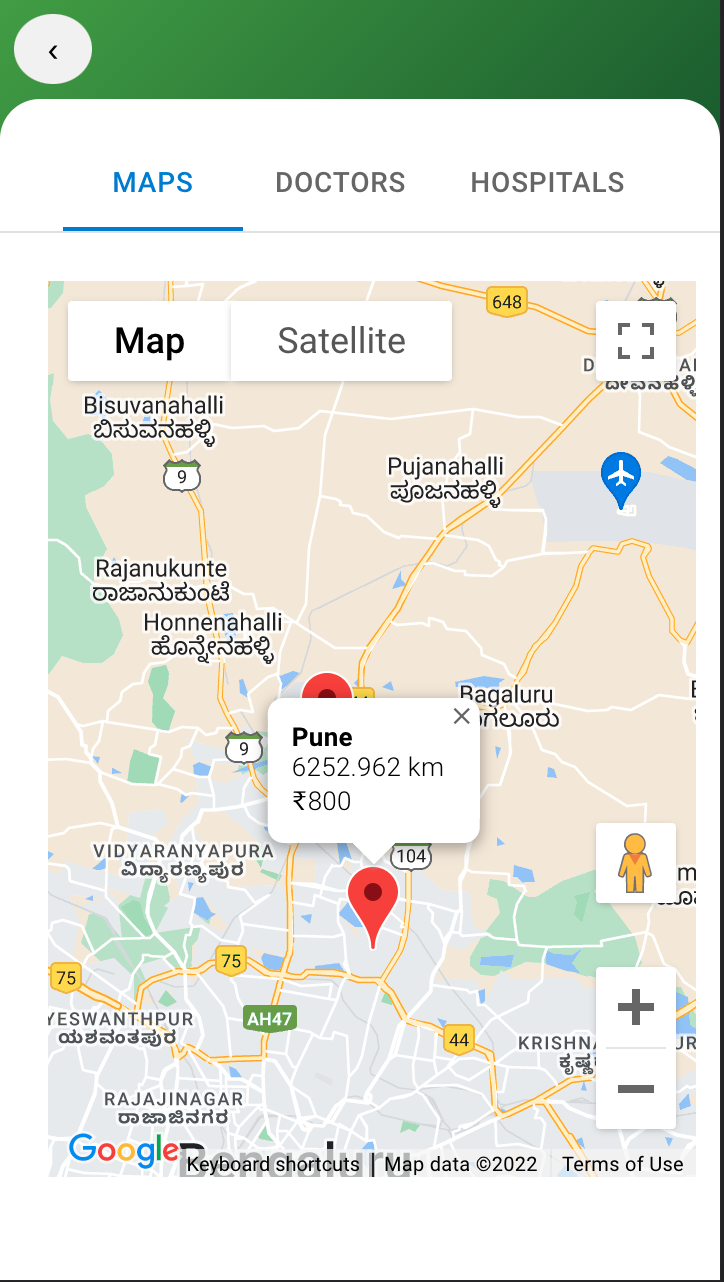
1. **Autosuggest Workflow**
   1. **Specialty autosuggestion**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

* 1. **Doctor search & discoverability (mapping parameters like location, pricing, rating, doctor picture, mode of consultation)**

****

****

**Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated**

**Workflow with Video**

We will demo the app during review.

**Sample data**

* **Autocorrect**

If a user enters a word with an incorrect spelling, the system will make intelligent, autocorrect recommendations to the user. For example, when searching for a cardiologist, the user inputs “cardilgist” in the search field, the system will return an autocorrected recommendation to the user.

**Graphical user interface, text, application, email

Description automatically generated**

* **Autosuggest**

When a user inputs a text, the system autosuggests symptoms, diagnosis and specialty based on similarity of the input text with the text type set of symptoms, diagnosis, specialties etc.   
For example, if a user inputs “Cardi” in the specialty search field, the system intelligently returns autosuggestions like “Cardiovascular/Pulmonary”.

**Graphical user interface, text, application, email

Description automatically generated**

If a user inputs “limb” in the diagnosis search field, the system intelligently autosuggests list of provisional diagnosis for the user to select from.

**Graphical user interface, text, application, email

Description automatically generated**

If a user inputs incomplete word or letters like “kn” in the symptom search field, the system intelligently autosuggests all possible symptoms with the letters kn in them, for the user to select from.

**Graphical user interface, text, application, email

Description automatically generated**

* **Structured and unstructured data**

Semi-structured data gets converted to structured format, as the model extracts entities including Hospital, Location, Specialty, Physician name from the search query.

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**For the language translation**

API: 'http://localhost:3000/translate'

Input body

{

“text”:”Text to be translated”,

“from”:”language” //default “en”,

“to”:”language to be converted”

}

Returns the converted language

The workflow of language translation

Approach 1:

Step 1: EUA’s will make an API call to UHI’s API gateway for local language to English translation

Step 2: EUA’s will pass returned English text back to UHI’s API gateway for any models to extract meaning information or they can use the translated text in their applications

Approach 2:

Step 1: EUA’s will make an API call to UHI’s API gateway using local language along with model to consume.

Step 2: API gateway will convert the local language to English, pass the translated text to model to extract meaningful information and return the information either in English or local language.