**Challenge 5: Optimizing prescription and diagnostic generation through speech or handwritten script** - Build tool which will allows for structured data entry through speech or handwritten script into a tool which can generate health records

**Objective:** A major challenge in India is the vast prevalence of hand-written prescriptions, with a very low readability rate, thereby resulting in errors in comprehension by the pharmacists or other stakeholders, leading to incorrect medications and severe health outcomes for the patient. A digitized prescription/diagnosis partly solves this challenge, however, has associated costs and requires a doctor to invest significant time in interacting with an electronic health record system, thereby reducing the patient-doctor facetime. Patient record addition and retrieval consumes 49% of doctor’s time in using an EHR system (C Sinsky et al.). To mitigate these challenges, there is an unmet need for creation of innovative speech-to-text or handwritten script conversion tools, leveraging AI technologies like natural language processing, machine learning and computer vision, to optimize digital prescription and diagnosis generation.

**Benefits:**

* Reduces amount of time consumed by a doctor in creating patient records
* Minimizes interaction of a doctor with a system and increases patient-doctor facetime
* Improves patient health outcomes
* Enables entry of structured data

**Features:**

* ***Implemented Features***
  + Speech-to-text and handwriting text capabilities for patient and doctor interaction
  + Both Android and Web applications developed
* ***Suggested Features (Future scope of enhancement)***
  + Extension to other charting features like vitals, lab test recommendation

**Technical Design**

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

**Diagram

Description automatically generated**

1. **Model APIs**

The following APIs have been developed for this track –

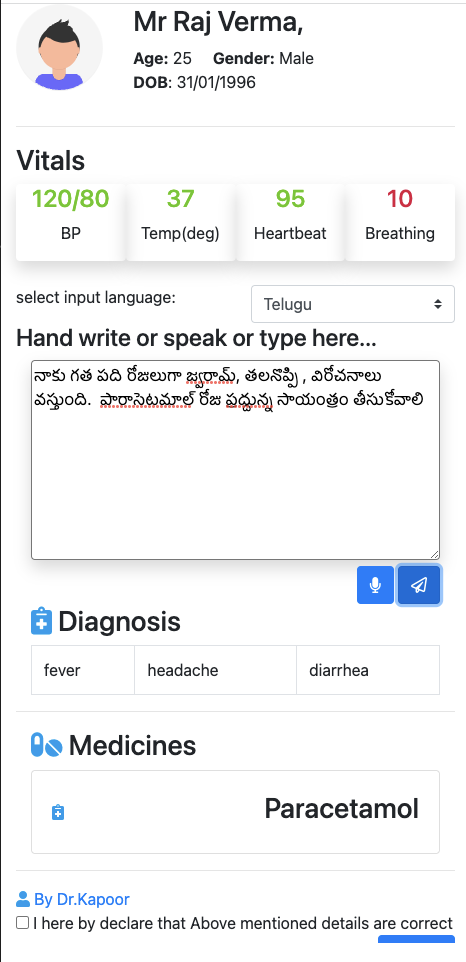
1. **extractDisease:** [**https://intelli-search-csh.herokuapp.com/extractDisease**](https://intelli-search-csh.herokuapp.com/extractDisease)
   1. This API allows extraction of disease from text
   2. The BIOBERT model is fine-tuned for NER disease extraction on NCBI Data
2. **extractMedication:** [**https://intelli-search-csh.herokuapp.com/extractmedication**](https://intelli-search-csh.herokuapp.com/extractmedication)
   1. This API allows extraction of medication, dosage, and frequency information from the text
   2. The model is fine-tuned to extract this info using MED7 Data

**How to set-up**

Please refer git hub repos read me files for how to set up

**Workflow with UI**

1. **Charting of patient information by user – write or speak capability**



* 1. **Handwriting capability**

**Graphical user interface, text, application

Description automatically generated**

* 1. **Speech-to-text capability**

**Graphical user interface, text, application

Description automatically generated**

1. **Extraction of patient information (diagnosis, prescription)**

**Table

Description automatically generated**

**Workflow with Video**

We will covering this during the demo

**Sample data**

* **Extraction of diseases**

When a user enters presenting symptoms of a patient (through speaking or handwriting), the system extracts recommended diagnosis. For example, on input “patient complains of diarrhea and fever for three days”, the system returns and extracts “diarrhea, fever” in the patient record.

**Graphical user interface, text, application, email

Description automatically generated**

Graphical user interface, text, application, email

Description automatically generated

* **Extraction of medicines**

**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.