LibreHealth

PROJECT : *Create web components for FHIR Resources*

BASIC INFORMATION:

NAME	: SHASHWAT
GITLAB	: shashwat
GITHUB	: geforce6t
FORUM	: shashwat
CHAT	: (shashwat)
COLLEGE	: NATIONAL INSTITUTE OF TECHNOLOGY TIRUCHIRAPPALLI
DEGREE AND YEAR	: BTECH, 2ND YEAR
TIMEZONE	: Indian Standard Time (UTC +05:30)

PROJECT INFORMATION:

GOALS :

This project bases its goal on creating an EHR system using the webcomponents implementing FHIR resources developed during the previous versions of GSoC.
For the implementation some new resource based components will be required as well which will be created simultaneously.

The plan is to use ReactJs along with Redux(Check here) for state management and import and use the old and newly created Polymer components. We will be using React because of its fast and flexible features along with easily creatable PWA (Progressive Web Application).

PROJECT BACKGROUND :

An EHR is a digital real-time, patient-centered record that makes information available instantly and securely to authorized users. This involves a lot of data and information transfer requiring a standard data model for improving interoperability which is given by the FHIR (Fast Healthcare Interoperability Resources) Specification.

In the previous version of GSoC this project was based on creating web components for these FHIR resources with a goal to simplify the making of an EHR. Components for following Resources and data types have already been developed :

Resources:

- Allergy
- Location
- Medication
- Observation
- Organisation
- Patient
- practitioner

Data Types: these are the basic type based components on which a resource field depends on, creating these can reduce the extra effort to create web components separately for a field of a resource.
Components for following data types are already created:

- Coding
- Period

- Range
- ratio

The basic implementation for a resource is to create the field type web components with / without using the data type webcomponent and finally combine them to create a *get/create* type web component which will be ultimately used by the EHR developer.

PROJECT IMPLEMENTATION:

The procedure for implementing an EHR involves mapping an existing workflow and recreating it by the means of the EHR so that the process can be improved or simplified.

During GSoC 2018, under this project a polymer application was developed which combined all the web components (developed at that time) into a PWA, the following tabs were developed:

- Create / search patient
- Create location
- Create organisation

The current project will be built upon this application by adding and implementing workflows. Some of the key workflows that can be considered while implementing the EHR are :

- ○ Patient check-in
- ○ Patient visits
- ○ e-Prescribing
- ○ Appointment scheduling
- ○ Laboratory orders
- ○ Referral generation and management
- ○ Office discharge
- ○ Billing

This list of workflows is not exhaustive and considering the limited time we have for GSoC , the implementation of the following 5 workflows would be possible.

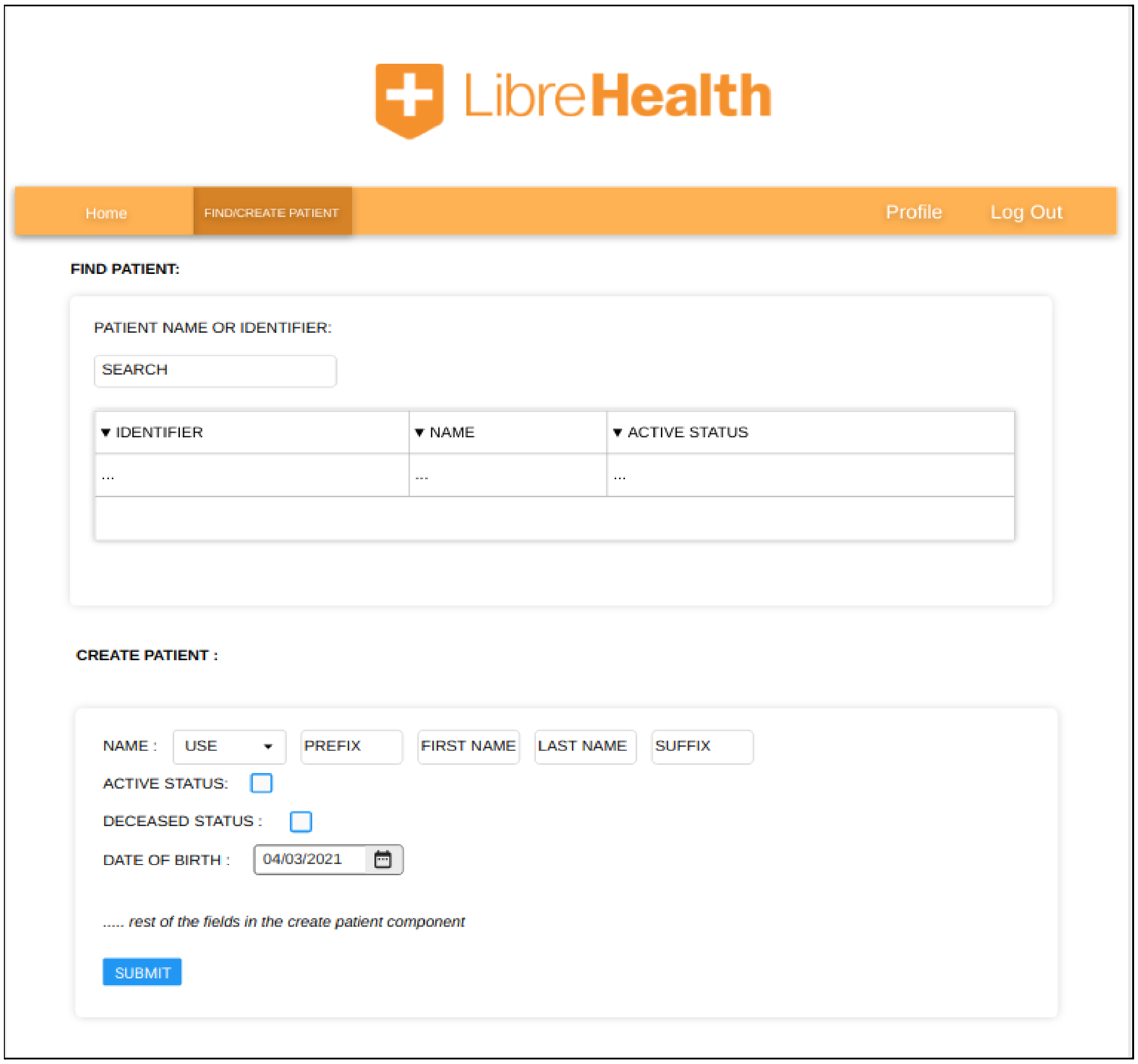
- Check-in
- Visit
- e-Prescription
- Appointment
- Lab orders

Each workflow involves professionals with different roles, so each workflow will be different for different users based on their role and privileges. For ex - during a visit the workflow of a provider will be different to that of a nurse.

Check-in : (front-desk receptionist)

The check-in workflow mainly focuses on the interaction between a patient and a front-desk receptionist. After logging into the EHR the receptionist will have the option to find a specific patient that exists in the EHR and modify/edit the patient information, else if the patient is new the receptionist can create a new patient entry into the EHR by entering the information provided by the patient. The screens below are meant for receptionists.

Fig 1A (fhir-search, fhir-create-patient)



FIND PATIENT:

PATIENT NAME OR IDENTIFIER:

▼ IDENTIFIER	▼ NAME	▼ ACTIVE STATUS
...	...	...

CREATE PATIENT :

NAME : USE PREFIX FIRST NAME LAST NAME SUFFIX

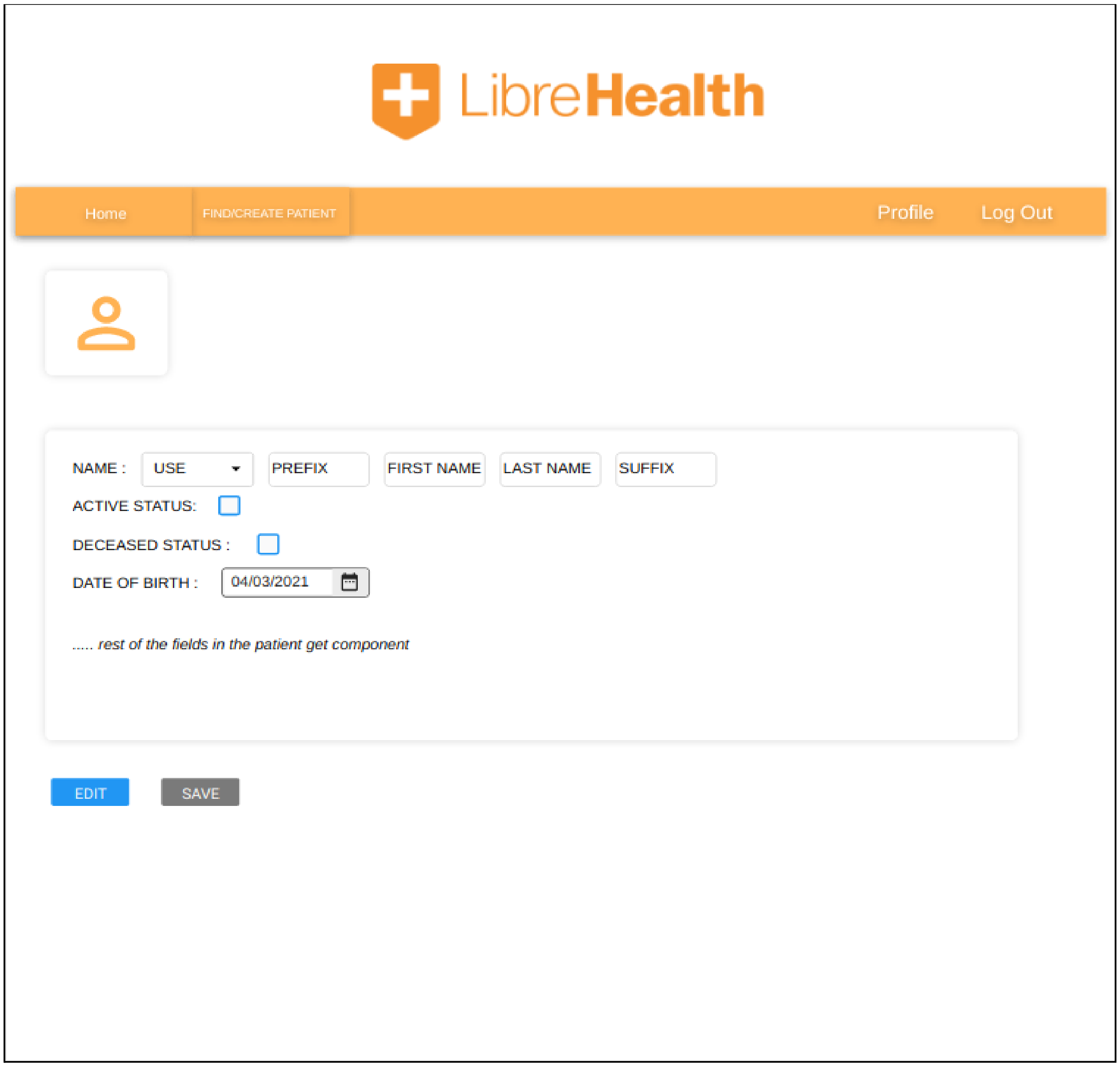
ACTIVE STATUS:

DECEASED STATUS :

DATE OF BIRTH : 04/03/2021

..... rest of the fields in the create patient component

Fig 1B (fhir-patient-get)



NAME : USE PREFIX FIRST NAME LAST NAME SUFFIX

ACTIVE STATUS:

DECEASED STATUS :

DATE OF BIRTH : 04/03/2021

..... rest of the fields in the patient get component

In Fig 1A the patients can be searched by the name or the identifier, on clicking an entry of this search the details of the patient are shown which can be edited (Fig 1B)

Web Components used :

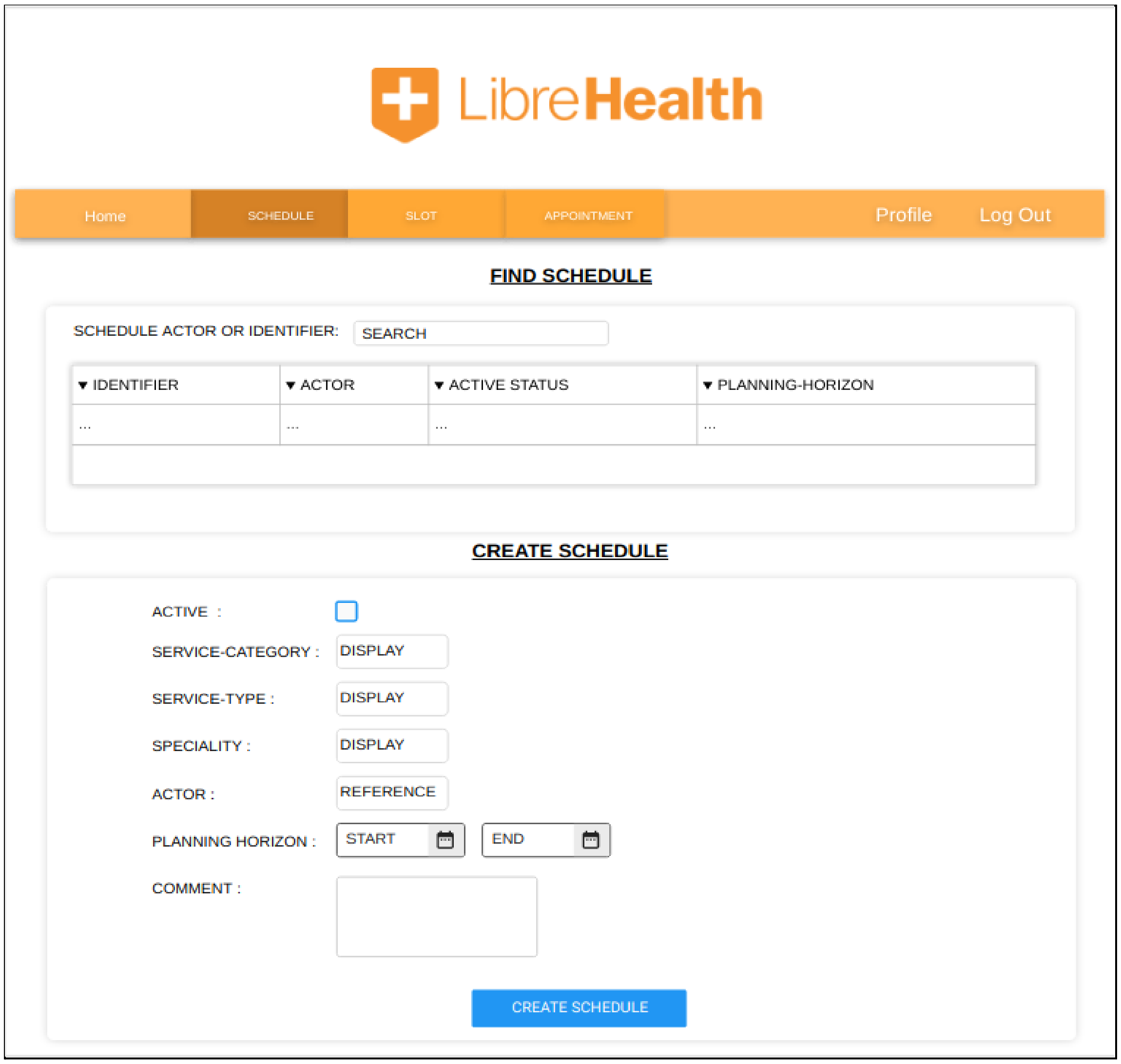
- fhir-create-patient
- fhir-patient-get
- fhir-search

Appointment : (Scheduler)
FHIR has a defined Scheduling workflow for the implementation of Appointments using the following resources :

- Schedule
- Slot
- Appointment

The **Schedule** resources provide a container for time-slots(Slot resource) that can be booked using an appointment; it does not provide any information about actual appointments. **Slot** resources are the time-slots that can be booked using an appointment. These define an interval status information of the Schedule resource. The **Appointment** resource has a reference to the Slot resource and contains the specifications about the appointment.

Fig 2A



FIND SCHEDULE

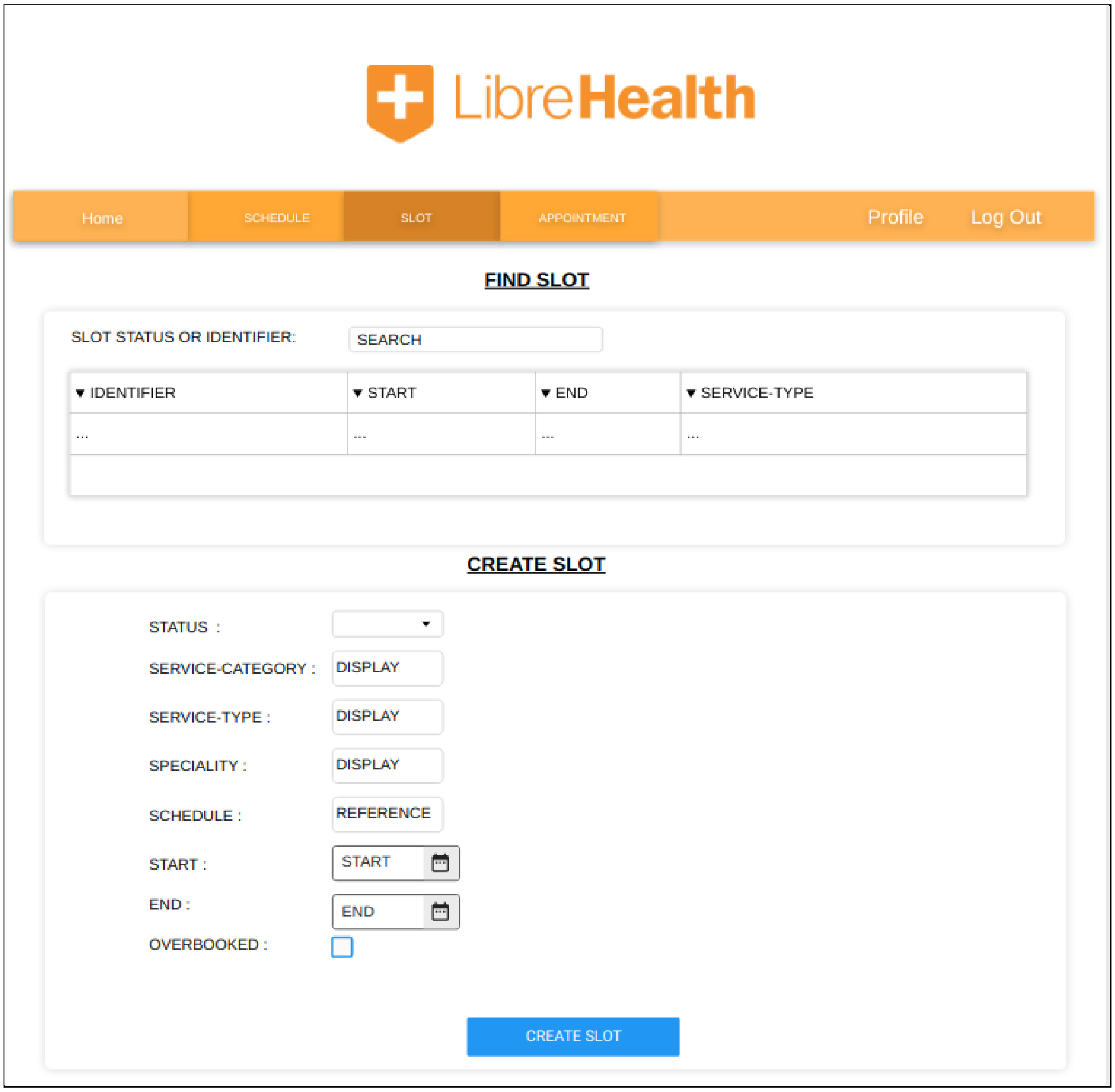
SCHEDULE ACTOR OR IDENTIFIER: SEARCH

▾ IDENTIFIER	▾ ACTOR	▾ ACTIVE STATUS	▾ PLANNING-HORIZON
...	...	...	...

CREATE SCHEDULE

ACTIVE :	☐
SERVICE-CATEGORY :	DISPLAY
SERVICE-TYPE :	DISPLAY
SPECIALITY :	DISPLAY
ACTOR :	REFERENCE
PLANNING HORIZON :	START END
COMMENT :	

Fig 2B



FIND SLOT

SLOT STATUS OR IDENTIFIER: SEARCH

▾ IDENTIFIER	▾ START	▾ END	▾ SERVICE-TYPE
...	...	...	...

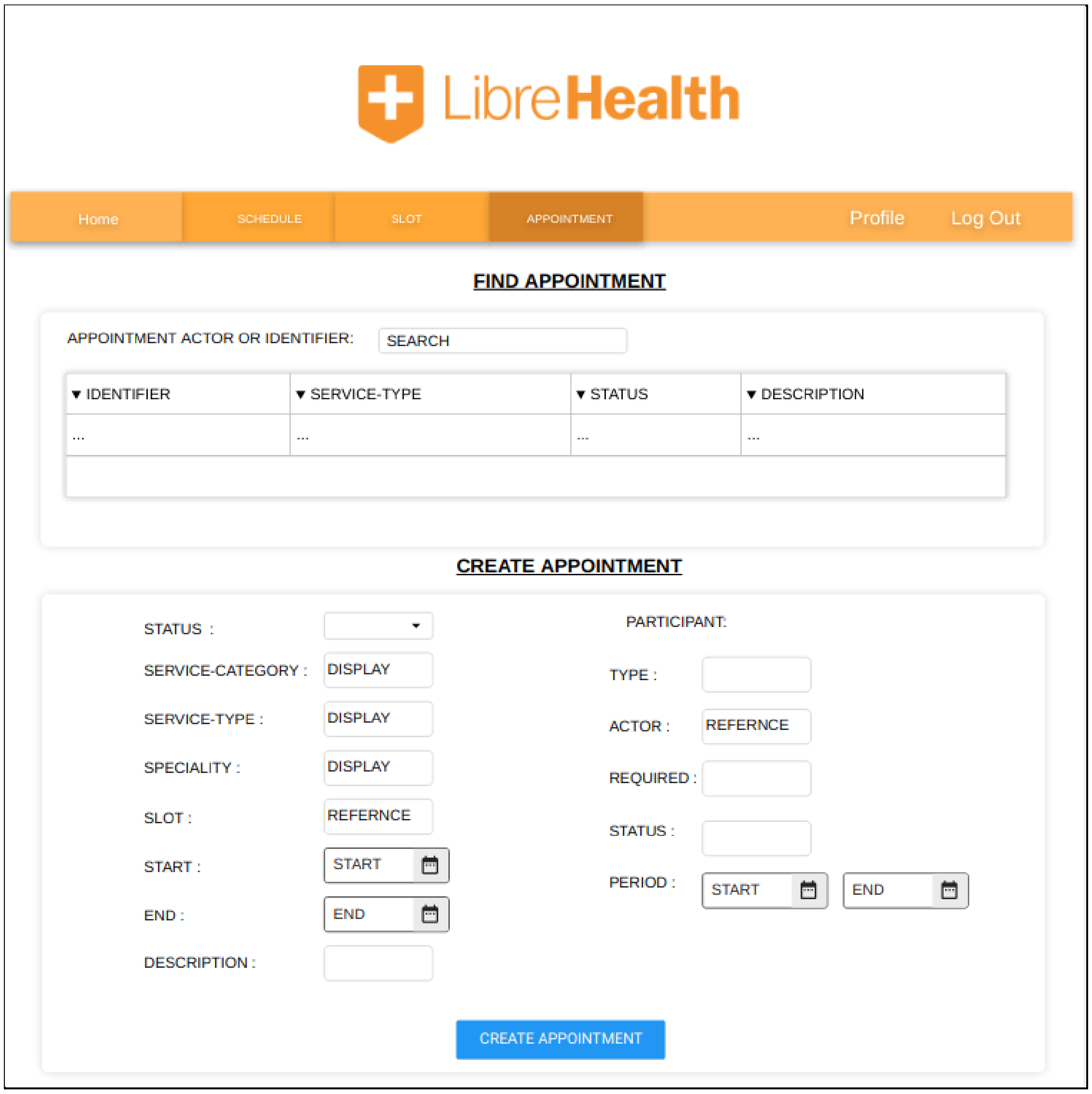
CREATE SLOT

STATUS :	
SERVICE-CATEGORY :	DISPLAY
SERVICE-TYPE :	DISPLAY
SPECIALITY :	DISPLAY
SCHEDULE :	REFERENCE
START :	START
END :	END
OVERBOOKED :	☐

CREATE SLOT

The figure 2A, 2B and 2C are the EHR screens for the **scheduler**, primarily the scheduler creates a schedule resource then allocates slots which have the schedule as a reference. Fig 2A and 2B shows the screens for the creation and display of the schedule and slot based resource. The screens are in order, first the creation of a schedule component -> slot component -> appointment component.
After the slots are created the scheduler can create an appointment using a slot as a reference.

Fig 2C



FIND APPOINTMENT

APPOINTMENT ACTOR OR IDENTIFIER: SEARCH

▾ IDENTIFIER	▾ SERVICE-TYPE	▾ STATUS	▾ DESCRIPTION
...	...	...	...

CREATE APPOINTMENT

STATUS :		PARTICIPANT:	
SERVICE-CATEGORY :	DISPLAY	TYPE :	
SERVICE-TYPE :	DISPLAY	ACTOR :	REFERNCE
SPECIALITY :	DISPLAY	REQUIRED :	
SLOT :	REFERNCE	STATUS :	
START :	START	PERIOD :	START END
END :	END		
DESCRIPTION :			

CREATE APPOINTMENT

Webcomponents required:

- Create-schedule
- Create-slot
- Create-appointment

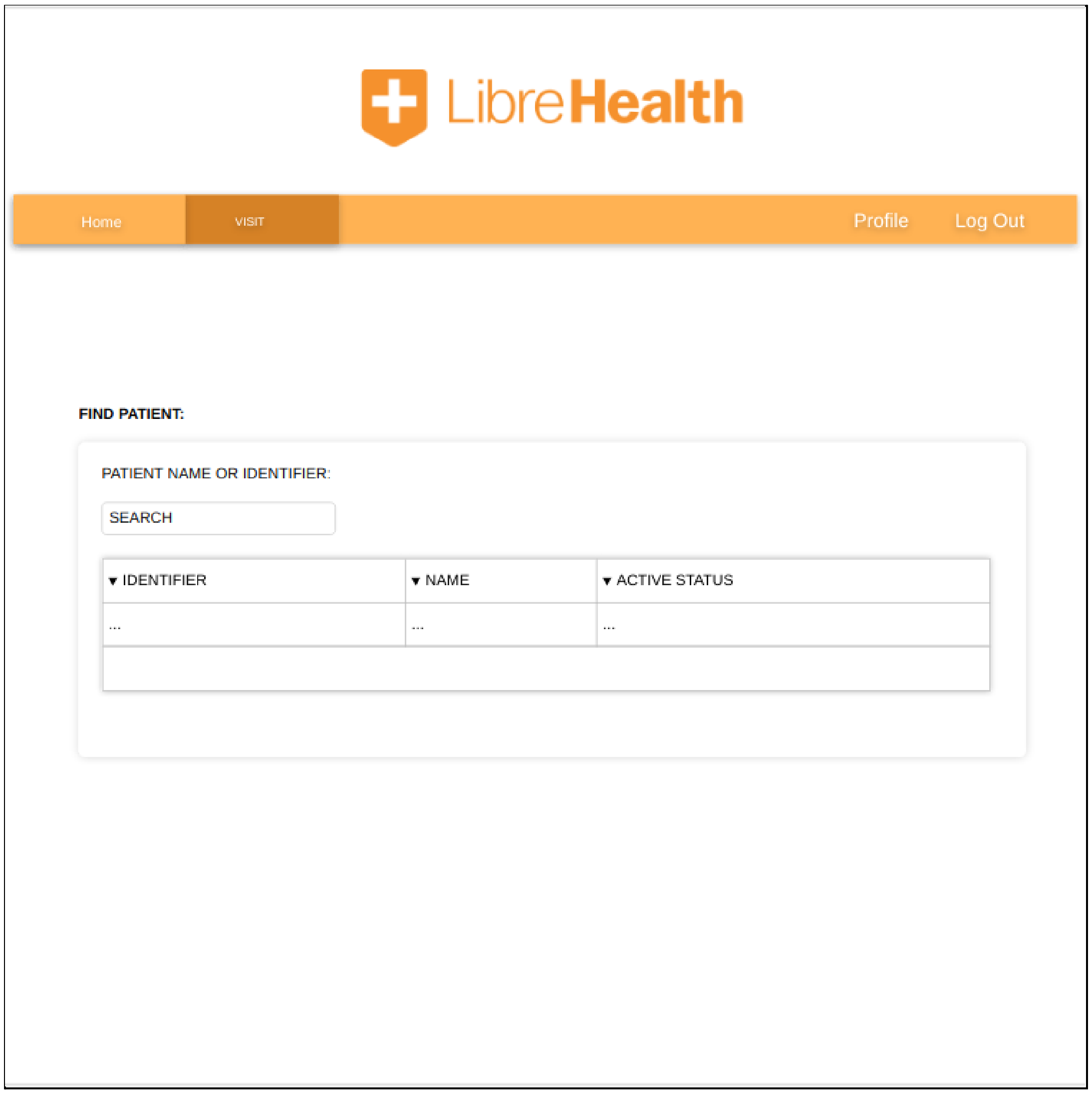
Visit : (Nurse, Provider)

The visit workflow primarily involves the nurse and the provider along with the patient, the nurse checks the arrival of the patient, takes the vitals, verifies and creates medications and allergies.

Provider then checks the patient and performs exams and orders lab tests if necessary. FHIR does not have a visit resource so every operation related to a visit event is handled by the modified encounter resource.
In FHIR, Appointment is used for establishing a date for the encounter,
When the patient arrives and the visit is about to start, then the appointment will be marked as fulfilled, and linked to the newly created encounter.

We have the following screens for the **NURSE :**

Fig 3A : The nurse logs into the system and finds and selects the patient



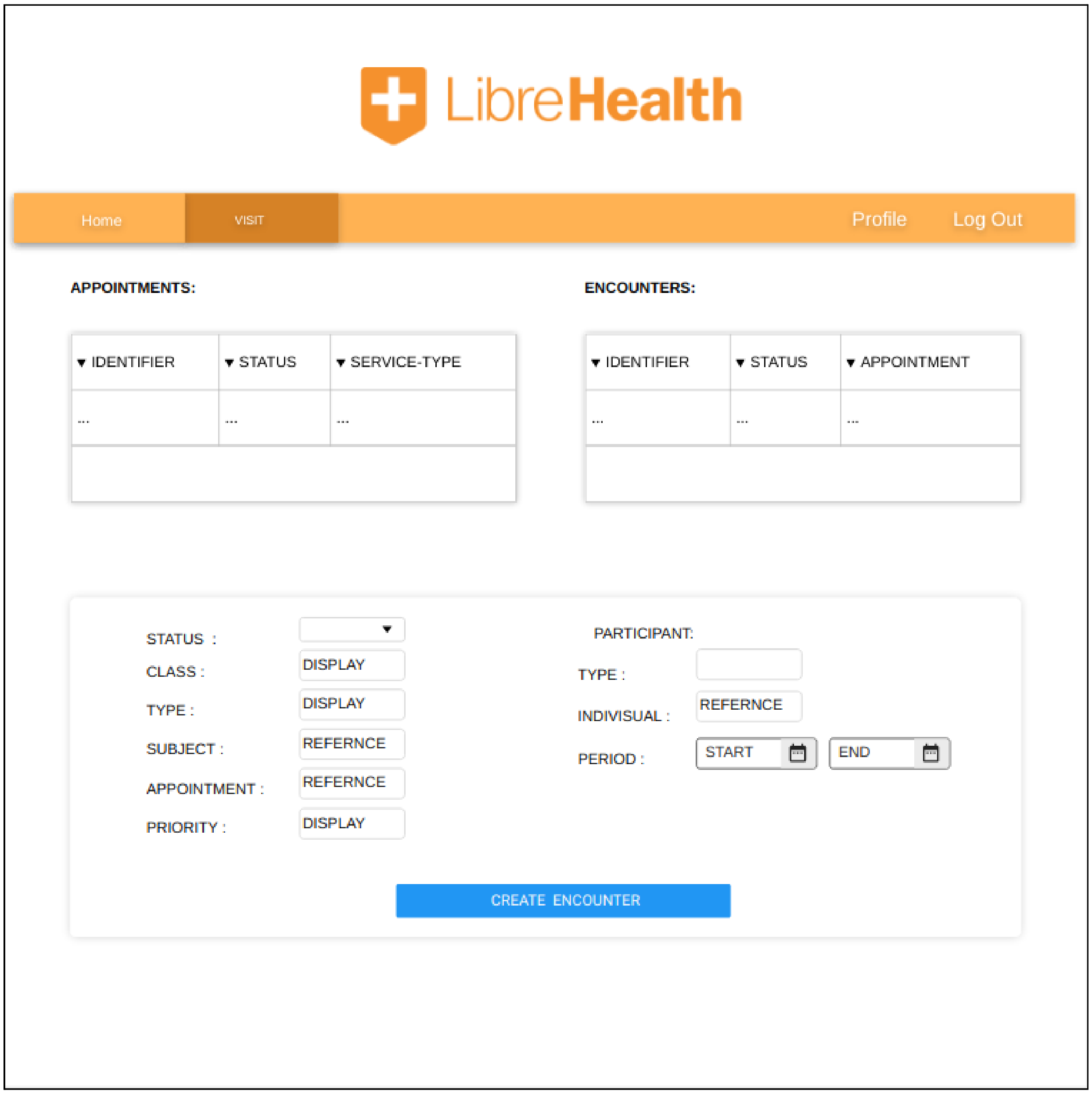
FIND PATIENT:

PATIENT NAME OR IDENTIFIER:

SEARCH

▾ IDENTIFIER	▾ NAME	▾ ACTIVE STATUS
...	...	...

Fig 3B : this screen contains the appointments and encounters of the selected patient, the encounter is created which fulfills a specified appointment.



Home | VISIT | Profile | Log Out

APPOINTMENTS:

▾ IDENTIFIER	▾ STATUS	▾ SERVICE-TYPE
...	...	...

ENCOUNTERS:

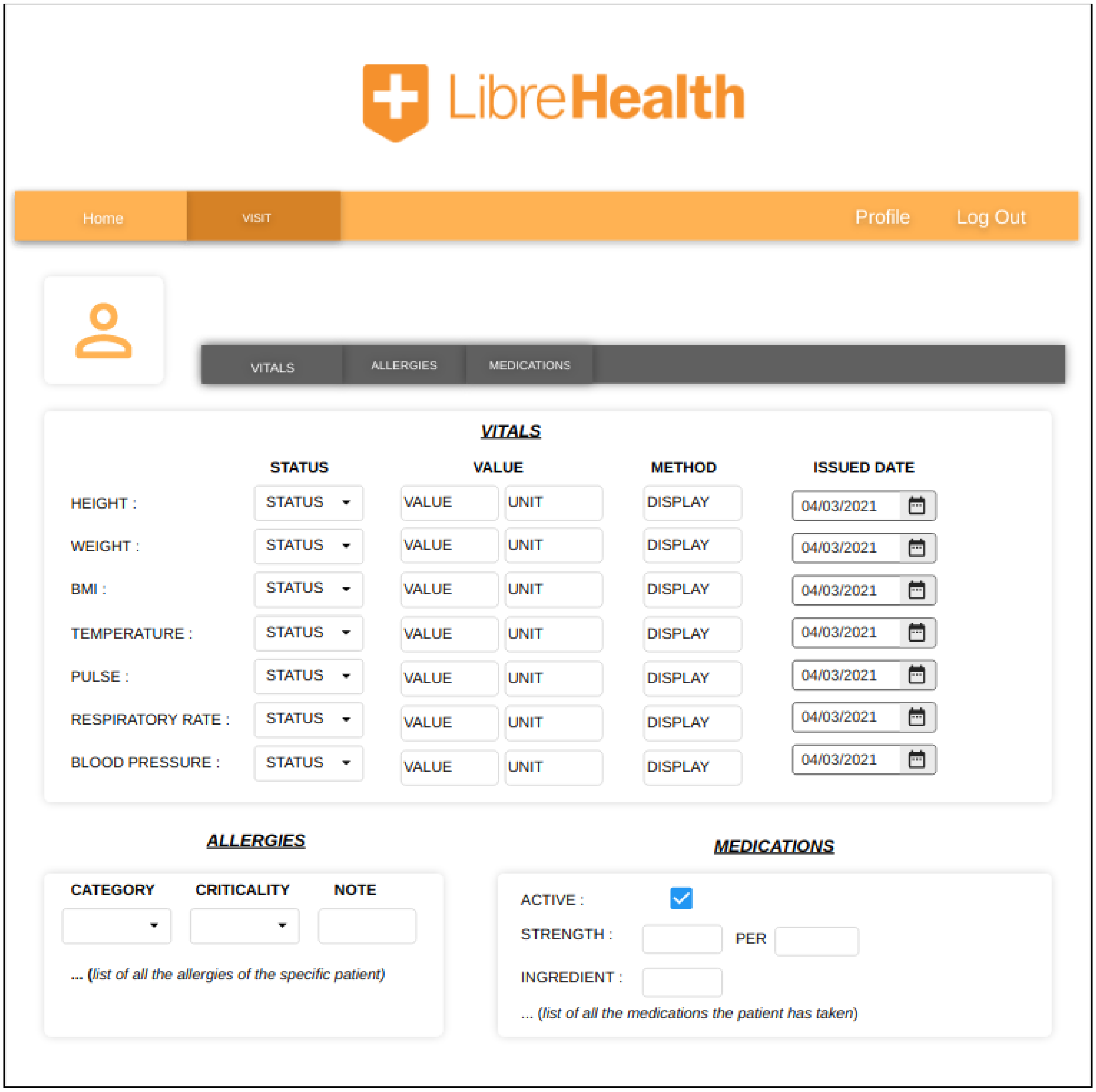
▾ IDENTIFIER	▾ STATUS	▾ APPOINTMENT
...	...	...

STATUS :	▾
CLASS :	DISPLAY
TYPE :	DISPLAY
SUBJECT :	REFERNCE
APPOINTMENT :	REFERNCE
PRIORITY :	DISPLAY

PARTICIPANT:

TYPE :		
INDISVISUAL :	REFERNCE	
PERIOD :	START	END

Fig 3C : The dashboard screen with the basic information including vitals (vitals is of observation resource type), allergies and medications



VITALS

	STATUS	VALUE		METHOD	ISSUED DATE
HEIGHT :	STATUS	VALUE	UNIT	DISPLAY	04/03/2021
WEIGHT :	STATUS	VALUE	UNIT	DISPLAY	04/03/2021
BMI :	STATUS	VALUE	UNIT	DISPLAY	04/03/2021
TEMPERATURE :	STATUS	VALUE	UNIT	DISPLAY	04/03/2021
PULSE :	STATUS	VALUE	UNIT	DISPLAY	04/03/2021
RESPIRATORY RATE :	STATUS	VALUE	UNIT	DISPLAY	04/03/2021
BLOOD PRESSURE :	STATUS	VALUE	UNIT	DISPLAY	04/03/2021

ALLERGIES

CATEGORY	CRITICALITY	NOTE

... *(list of all the allergies of the specific patient)*

MEDICATIONS

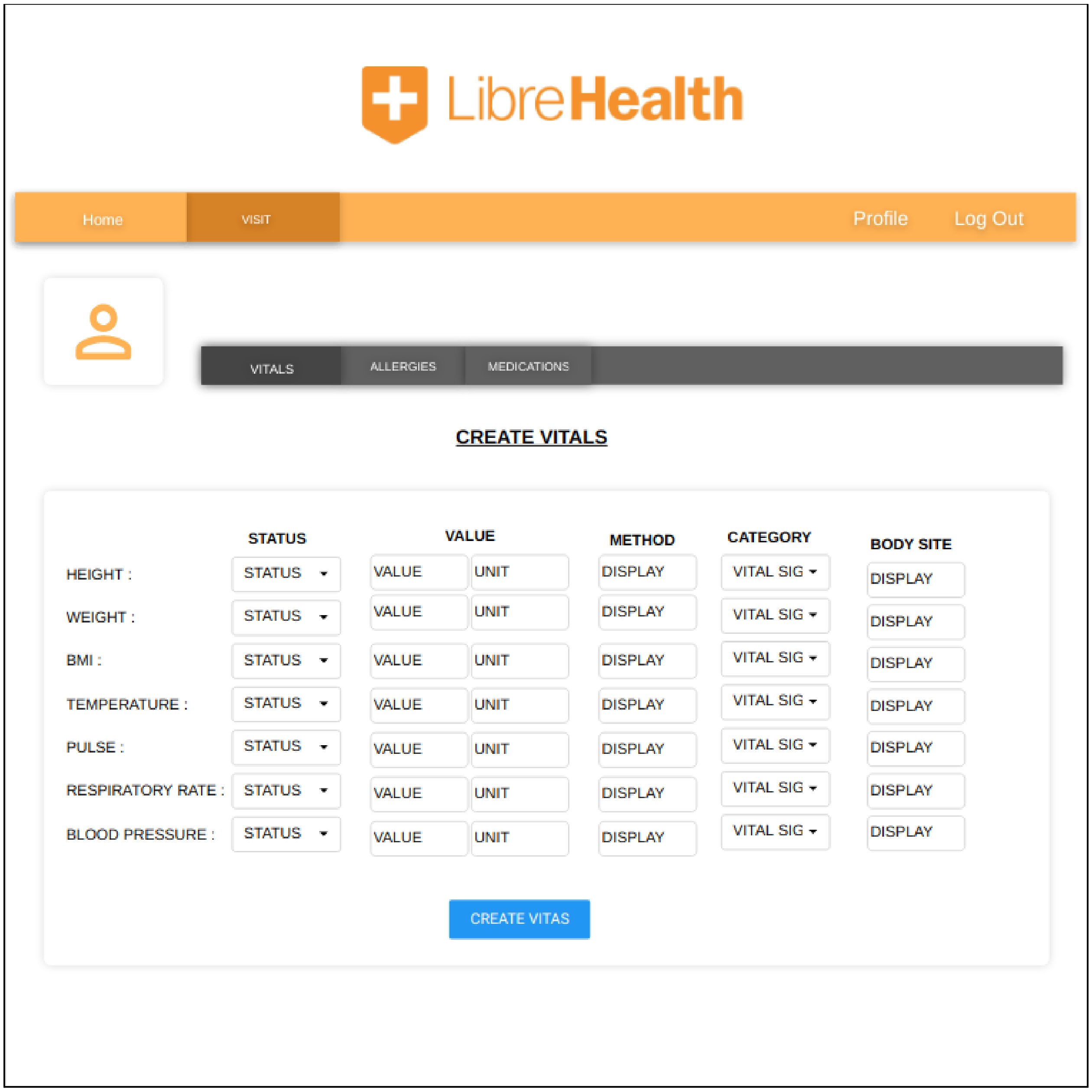
ACTIVE : ✓

STRENGTH : PER

INGREDIENT :

... *(list of all the medications the patient has taken)*

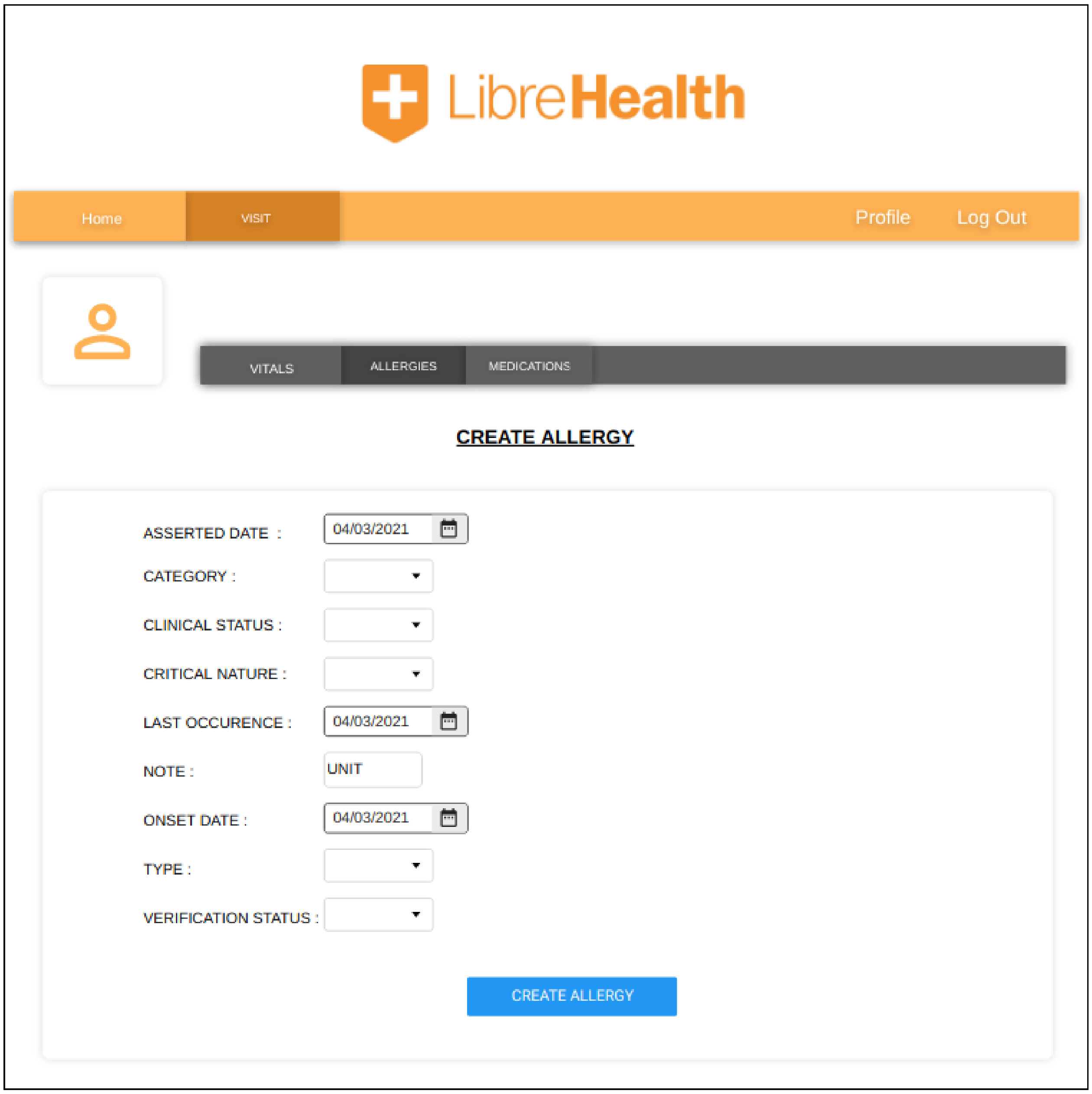
Fig 3D : the screen for creating vitals, vitals is the implementation of observation resource, observation has a category field which could be set to vital signs. This screen has a bundle of observation resource entries.



CREATE VITALS

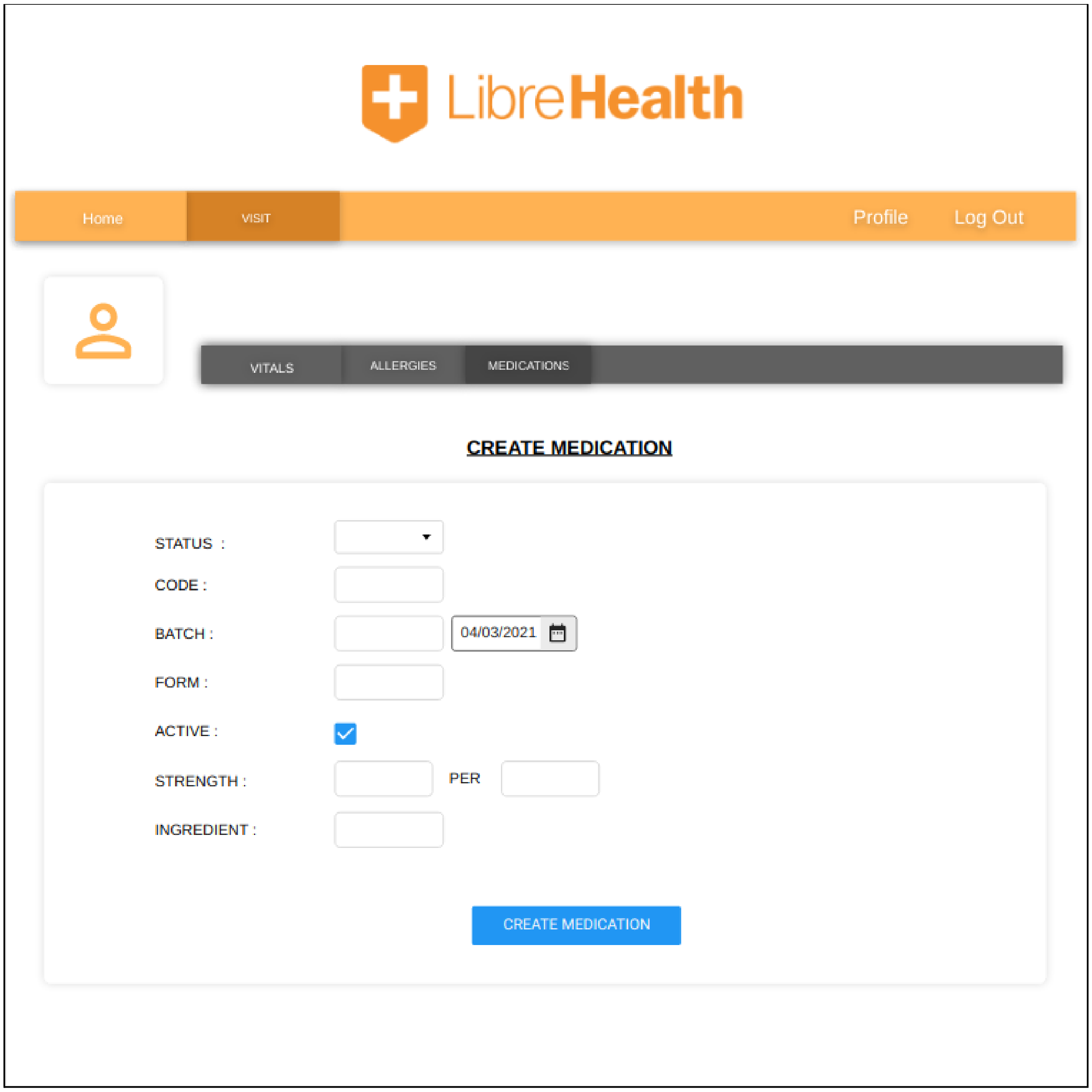
	STATUS	VALUE		METHOD	CATEGORY	BODY SITE
HEIGHT :	STATUS	VALUE	UNIT	DISPLAY	VITAL SIG	DISPLAY
WEIGHT :	STATUS	VALUE	UNIT	DISPLAY	VITAL SIG	DISPLAY
BMI :	STATUS	VALUE	UNIT	DISPLAY	VITAL SIG	DISPLAY
TEMPERATURE :	STATUS	VALUE	UNIT	DISPLAY	VITAL SIG	DISPLAY
PULSE :	STATUS	VALUE	UNIT	DISPLAY	VITAL SIG	DISPLAY
RESPIRATORY RATE :	STATUS	VALUE	UNIT	DISPLAY	VITAL SIG	DISPLAY
BLOOD PRESSURE :	STATUS	VALUE	UNIT	DISPLAY	VITAL SIG	DISPLAY

Fig 3E : screen to create allergy, the field type components of allergy are already created, for this screen a create type component combining all the field type components of allergy resource will be required.



ASSERTED DATE :	04/03/2021
CATEGORY :	
CLINICAL STATUS :	
CRITICAL NATURE :	
LAST OCCURENCE :	04/03/2021
NOTE :	UNIT
ONSET DATE :	04/03/2021
TYPE :	
VERIFICATION STATUS :	

Fig 3F : The screen to create the medication the patient is already taking



STATUS :	
CODE :	
BATCH :	04/03/2021
FORM :	
ACTIVE :	✓
STRENGTH :	PER
INGREDIENT :	

The medication resource gets implemented in Fig 3F , this is different from the resource that is meant to order medication (medication request). This resource focuses on the information about the medicine including ingredients and strength.

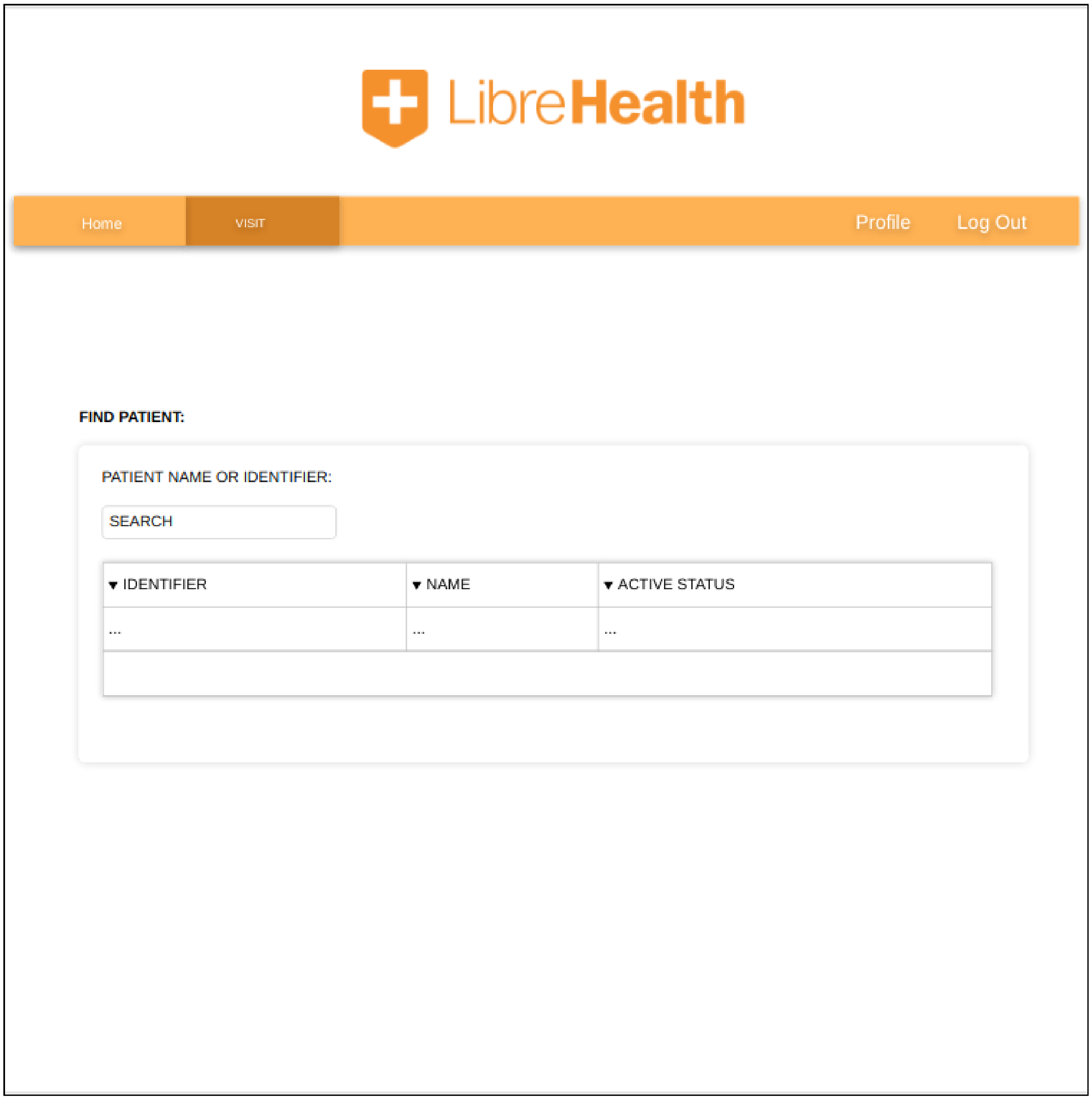
Webcomponents required (for nurse workflow):

- Create-medication
- Create-allergy
- Create-observation

- Create-encounter

We have the following screens for the **Provider :**

Fig 4A : the provider logs into the system and selects the patient



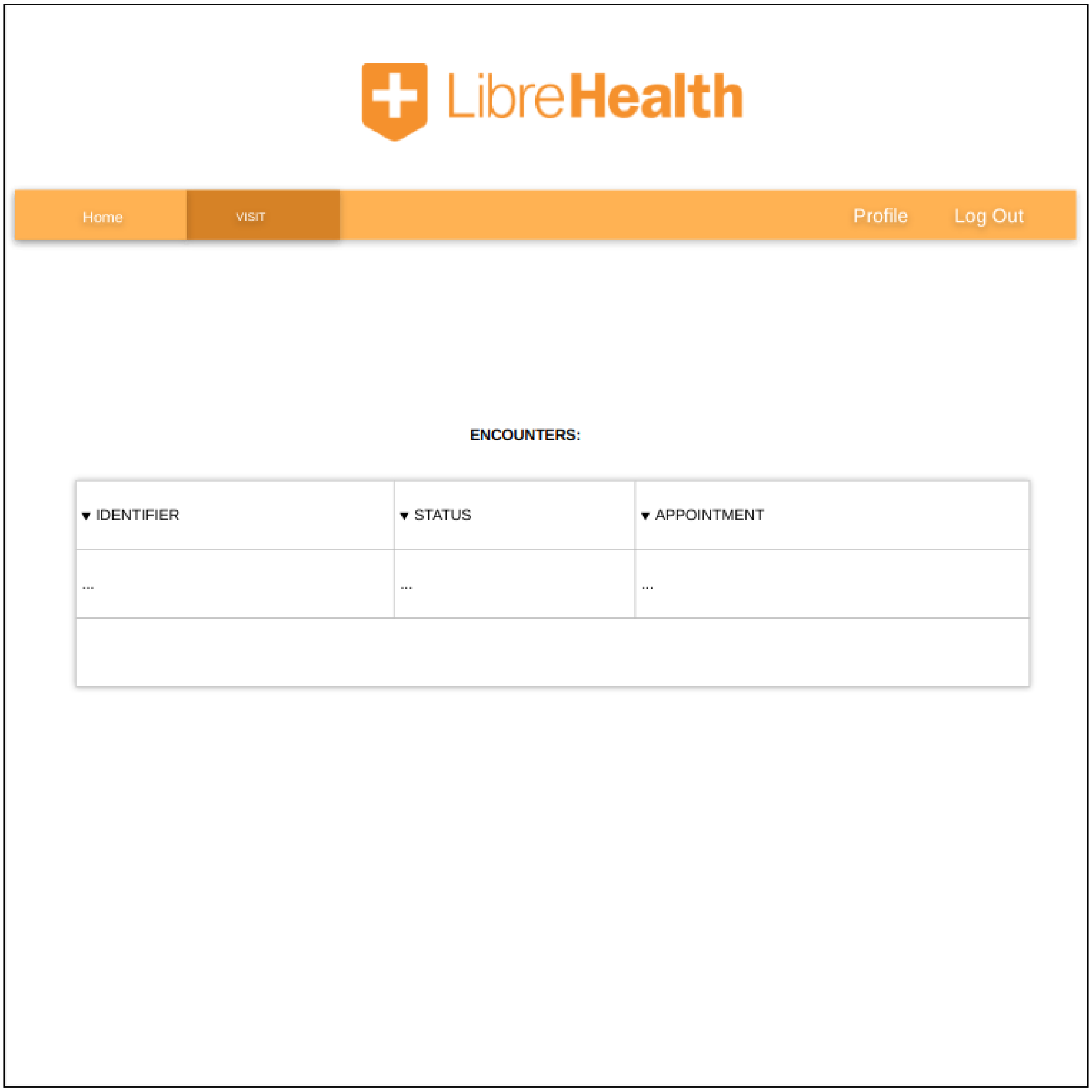
FIND PATIENT:

PATIENT NAME OR IDENTIFIER:

SEARCH

▾ IDENTIFIER	▾ NAME	▾ ACTIVE STATUS
...	...	...

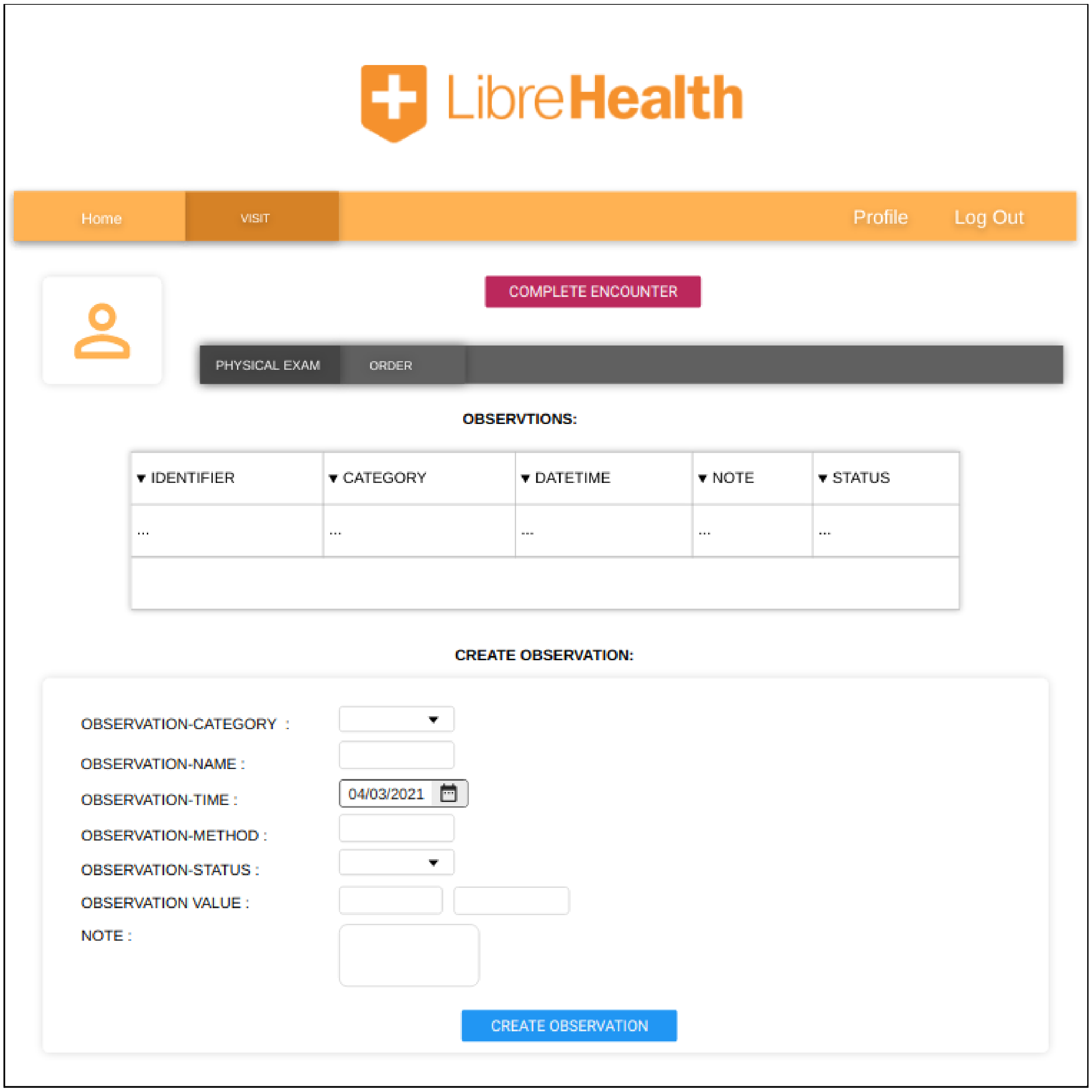
Fig 4B: the following screen contains the encounters related to the selected patient , the provider selects the appropriate encounter



ENCOUNTERS:

▾ IDENTIFIER	▾ STATUS	▾ APPOINTMENT
...	...	...

Fig 4C: the provider can take physical exams and create the observation



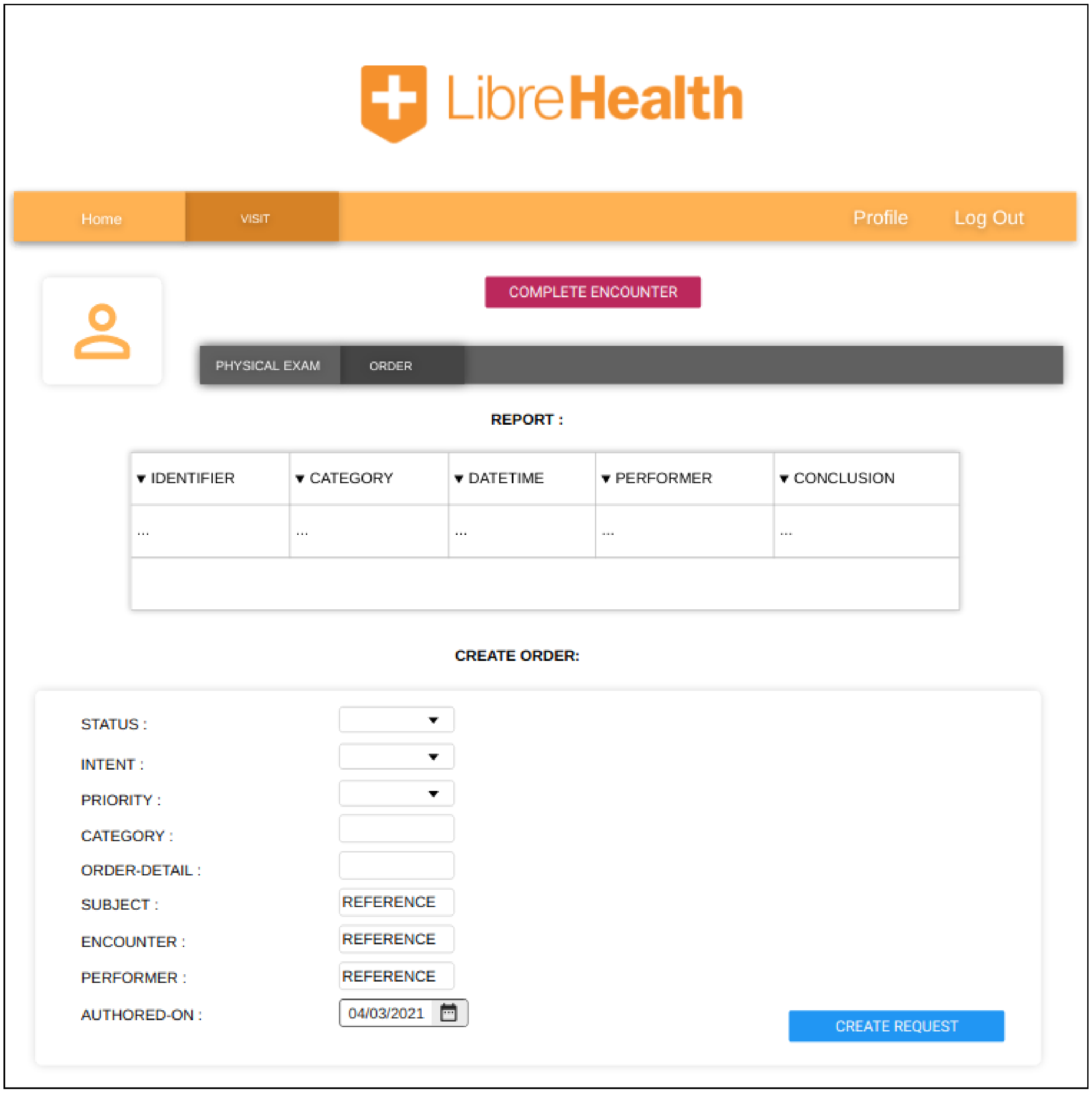
OBSERVTIONS:

▾ IDENTIFIER	▾ CATEGORY	▾ DATETIME	▾ NOTE	▾ STATUS
...	...	...	...	...

CREATE OBSERVATION:

OBSERVATION-CATEGORY :	
OBSERVATION-NAME :	
OBSERVATION-TIME :	04/03/2021
OBSERVATION-METHOD :	
OBSERVATION-STATUS :	
OBSERVATION VALUE :	
NOTE :	

Fig 4D : this screen is for ordering a lab test (the service request component is used here), this is a part of the order workflow,
The reports received from the lab are also displayed here. The provider also has the option to complete the encounter.



REPORT :

▾ IDENTIFIER	▾ CATEGORY	▾ DATETIME	▾ PERFORMER	▾ CONCLUSION
...	...	...	...	...

CREATE ORDER:

STATUS :	
INTENT :	
PRIORITY :	
CATEGORY :	
ORDER-DETAIL :	
SUBJECT :	REFERENCE
ENCOUNTER :	REFERENCE
PERFORMER :	REFERENCE
AUTHORED-ON :	04/03/2021

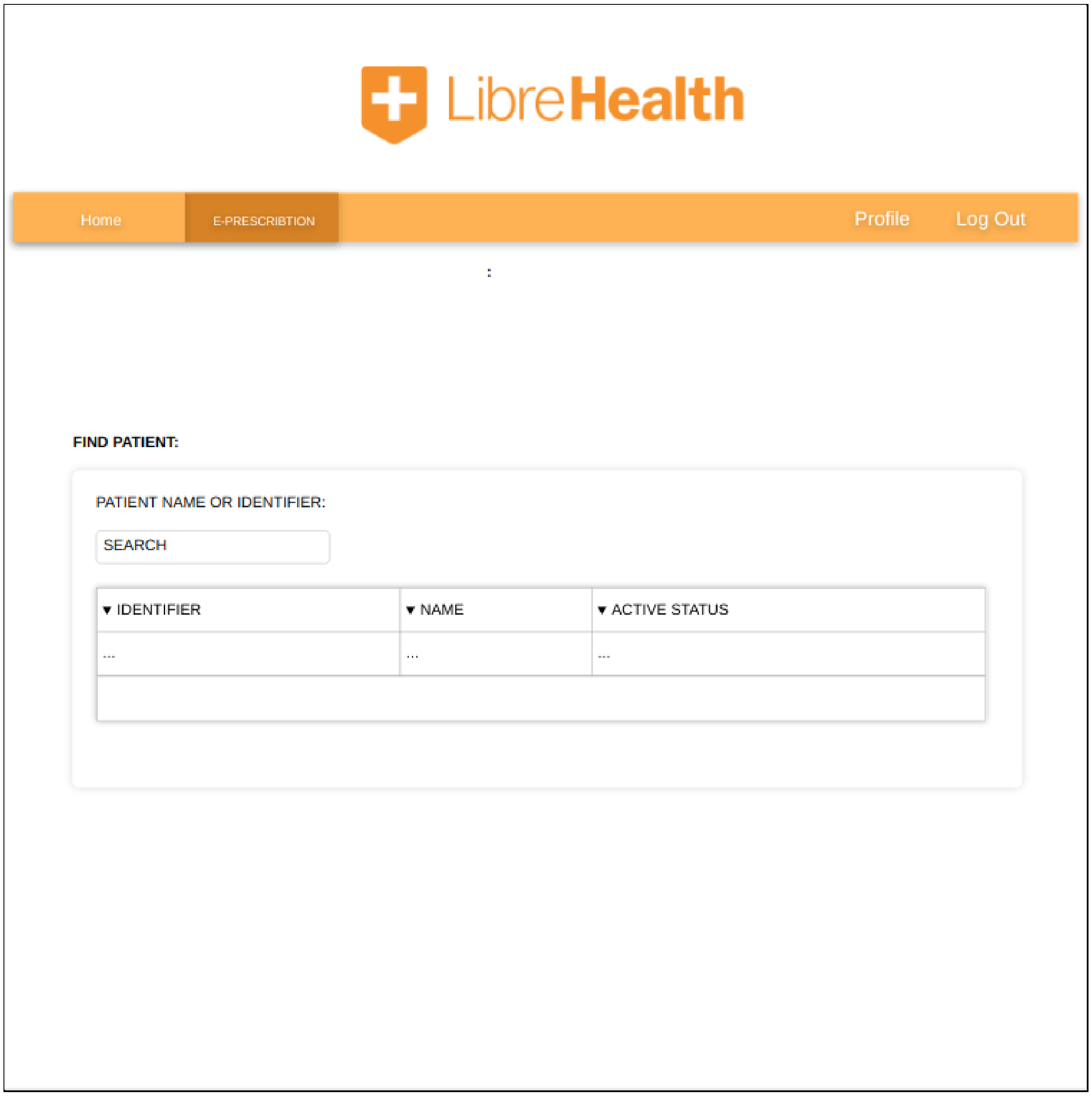
The webcomponents used (for the provider workflow):

- Create service request
- Create observation

e-prescription : (prescriber)
This workflow involves the prescriber and the pharmacy. The prescriber checks the patient and orders the medication. The pharmacy receives all the orders and completes the request. Creating the screens for the pharmacist is not covered in the project because of the limited time.

The screens for the **Prescriber:**

Fig 5A: the prescriber find the patient



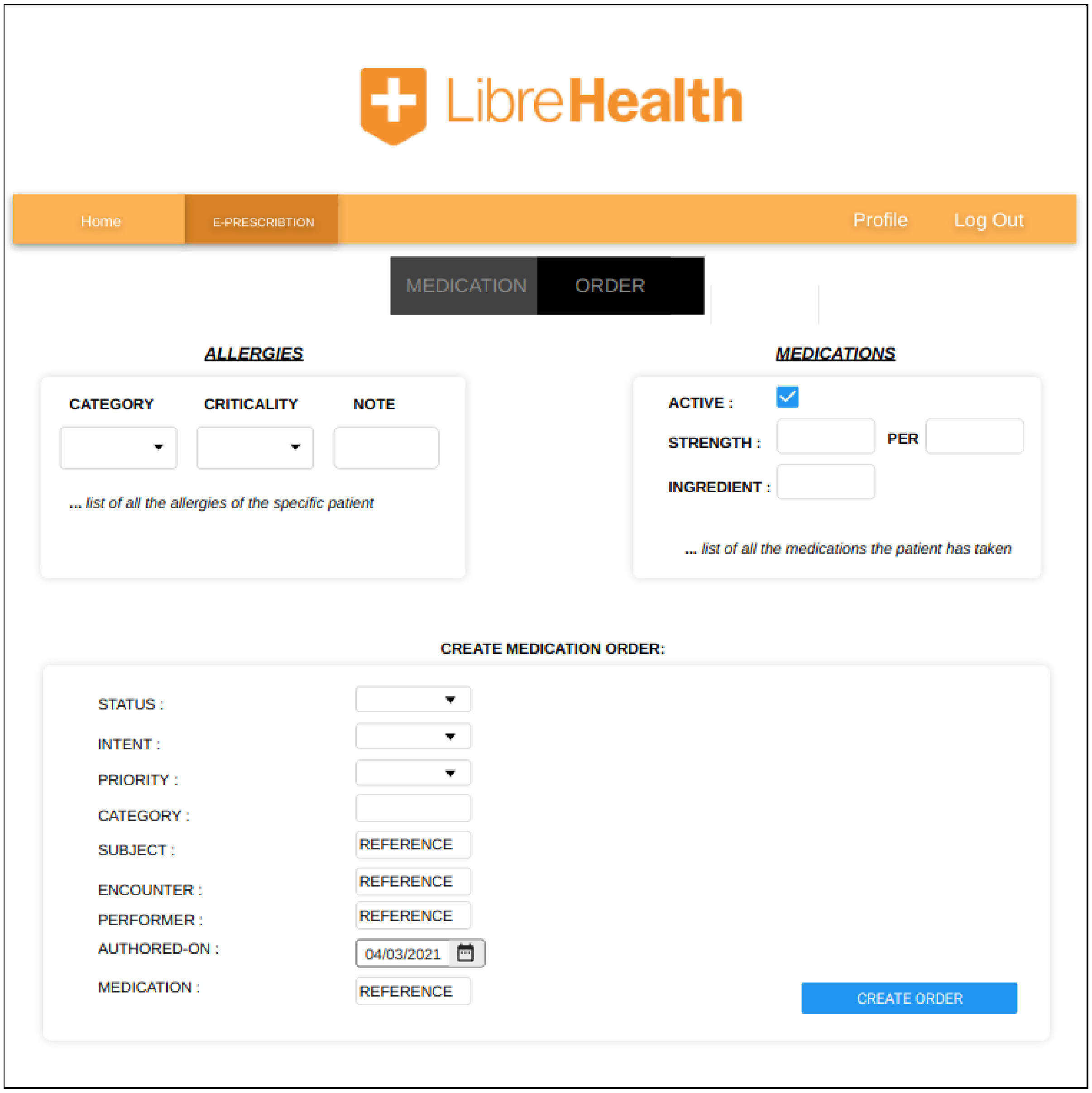
FIND PATIENT:

PATIENT NAME OR IDENTIFIER:

SEARCH

▾ IDENTIFIER	▾ NAME	▾ ACTIVE STATUS
...	...	...

Fig 5B : The screen for making a medication request / order



ALLERGIES

CATEGORY	CRITICALITY	NOTE

... list of all the allergies of the specific patient

MEDICATIONS

ACTIVE : ✓

STRENGTH : **PER**

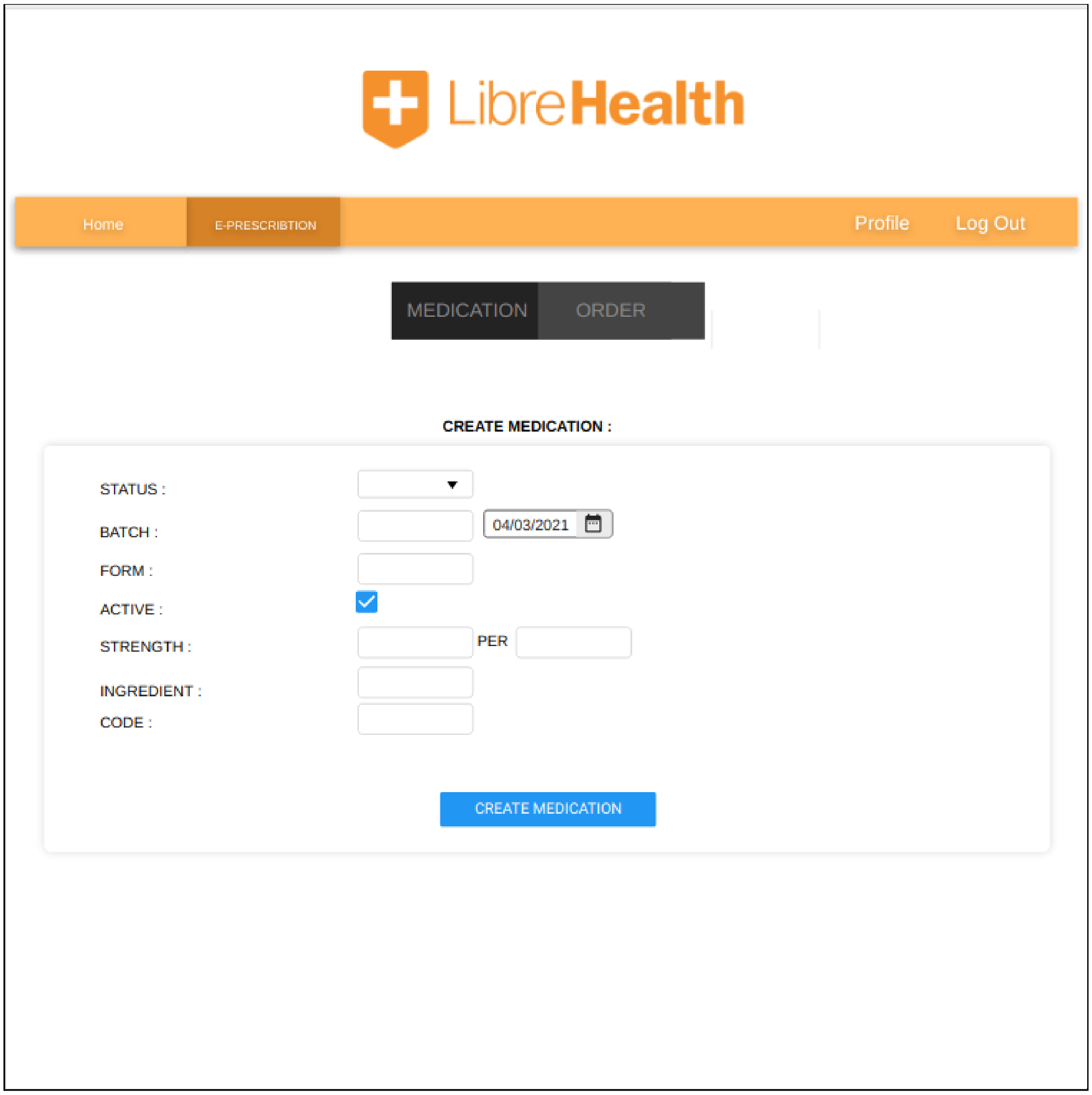
INGREDIENT :

... list of all the medications the patient has taken

CREATE MEDICATION ORDER:

STATUS :	
INTENT :	
PRIORITY :	
CATEGORY :	
SUBJECT :	REFERENCE
ENCOUNTER :	REFERENCE
PERFORMER :	REFERENCE
AUTHORED-ON :	04/03/2021
MEDICATION :	REFERENCE

Fig 5C : the medication request has a reference to the medication resource



CREATE MEDICATION :

STATUS :	
BATCH :	04/03/2021
FORM :	
ACTIVE :	✓
STRENGTH :	PER
INGREDIENT :	
CODE :	

The `medication resource` contains the specifications about the medicine, the `medication request resource` references this resource and does not contain the details about the medicine, hence creation of `medication resource` entry is essential for a request.

Webcomponents required:

- Create-medication
- Create-medication request

order : (provider)

The order workflow involves the provider and laboratory, the provider checks the patient and orders laboratory tests which the lab receives. The lab then performs the test and creates a report which can be seen by the provider. The screen for the provider is already created in **Fig 4D,** the screens of the laboratory are not covered due to limited time for GSoC.

USING LIT-ELEMENT COMPONENTS WITH REACT:

Creating the react application and importing the polymer components can be done using the webcomponent repo as a git submodule within the main application.

There are a few things to be noted :

- React currently doesn't have a way to listen to native DOM events (preferring, instead, to use it's own proprietary SyntheticEvent system), nor does it have a way to declaratively access the current DOM element without using a ref. The good thing about the current web components we have is that most of event handling is completely done inside the web component repo so we don't usually have to traverse the dom and create event listeners from the react application.

- In case of requirement of event handling we will make use of React's `useRef` hook to create a reference to the native DOM element, React's `useEffect` and `useState` hooks can be further used to communicate with the values property.

- So as long as the communication of react with the dom is limited the application will not require extra modifications, therefore the application will be designed in a specific manner where the lit-components handle all dom related events and react the UI of the EHR.

The Lerna package will be required to be installed as a dependency in the application. lerna.json file will be required as well to configure the lerna options.

Below is the basic sample package.json :

```
{
 "name": "my-app",
 "version": "0.1.0",
 "private": true,
 "dependencies": {
   "lerna": "^4.0.0",
   "react": "^17.0.2",
   "react-dom": "^17.0.2",
   "react-scripts": "4.0.3",
   "web-vitals": "^1.0.1"
 },
 "workspaces": [
   "lh-toolkit-webcomponents/packages/*"
 ],

 "scripts": {
   "start": "react-scripts start",
   "build": "react-scripts build",
   "test": "react-scripts test",
   "eject": "react-scripts eject"
 }
}
```

Below is the sample lerna.json :

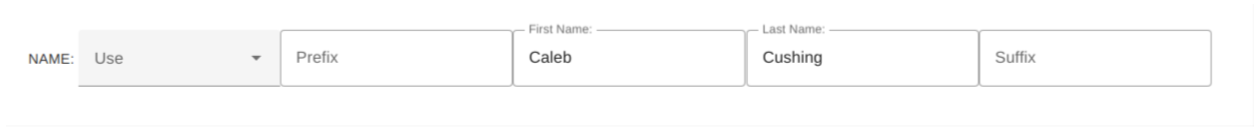
```
{
    "lerna": "4.0.0",
    "version": "0.0.1",
    "npmClient": "yarn",
    "packages": [
      "lh-toolkit-webcomponents/packages/*",
    ],
    "useWorkspaces": true
}
```

`lerna bootstrap --use-workspaces` command will be required to install all the dependencies of the packages into the react application and then the components can be imported and used.

Below is a sample App.js file where a webcomponent is imported and used :

```
import './App.css';
import '@lh-toolkit/fhir-human-name/fhir-human-name.js';
import React, { useRef, useEffect } from "react";

function App() {
 const elementRef = useRef();

 useEffect(() => {
   const divElement = elementRef.current;
   console.log(divElement.value);
 });

 return (
   <div className="App">
     <fhir-human-name ref={elementRef}
url="http://hapi.fhir.org/baseDstu2/Patient/175556"></fhir-human-name>
   </div>
 );
}

export default App;
```

Below is the output :



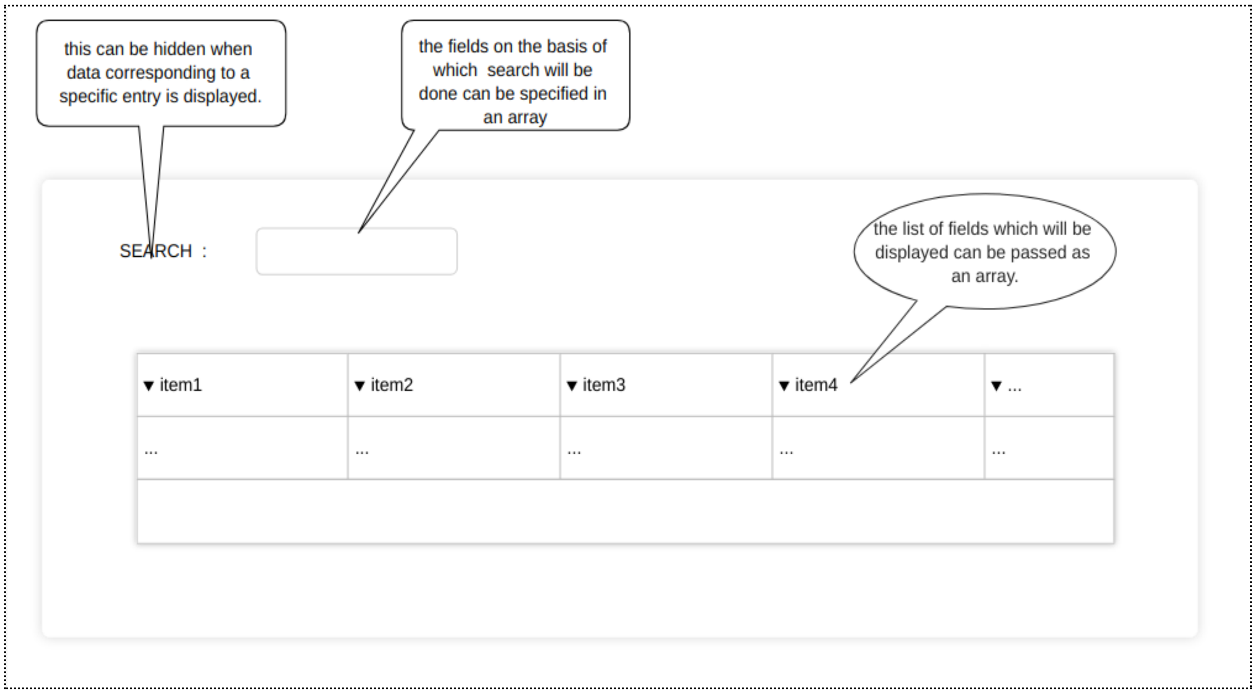
SEARCH AND LOGIN COMPONENT:

Most of the above screens contain a search component. There are two approaches for this implementation:

- Creating search components for each resource based component.
- Create a general search component compatible with all resources.

The repository has a `fhir-search component` from which inspiration can be taken while creating these, the second approach is preferable but if it presents issues the first approach can be taken.

Here is a mockup of the general search component:


this can be hidden when data corresponding to a specific entry is displayed.
the fields on the basis of which search will be done can be specified in an array
the list of fields which will be displayed can be passed as an array.
SEARCH :
▾ item1
▾ item2
▾ item3
▾ item4
▾ ...
...

Conditional searching is available in FHIR
A general format can be given by :

```
GET [base]/[type]?name=value&...{&_format=[mime-type]}}
```

ADMINISTRATOR:

Apart from these screens the EHR must also contain an administrator tab for the admin user, the admin/super user will have all of the privileges and the ability to create new entries into the EHR for the resources. Creating a new user and assigning a role will be the responsibility of the admin, moreover a lot of components which are not getting directly used in the EHR but as a reference can be created by the admin.

COMPONENTS REQUIRED :

Based on the EHR screens displayed in the upper sections, components for the following resources will be required apart from the already existing ones:

- Schedule |
- slot |
- Appointment |
- Medication (field type components already exists, `create` component required)
- Observation (field type components already exists, `create` component required)

- Allergy (field type components already exists, `create` component required)
- Encounter |
- Service request |
- Medication request |

Most of the above mentioned resources have many fields which are of a similar data type, hence creation of field based components for these would be repetitive and unnecessary.

Based on this notion the creation of following data type based web component would significantly reduce the work :

- Reference
- Codeable concept
- Code
- Boolean

The webcomponent repository has Gitlab issues addressing the creation of some of these data type components , these can be worked upon before the inception of the coding period.

The creation of a component is dependent on its requirement for the EHR, the components will be created once its utility and the fields required for the EHR is confirmed.

TIMELINE:

PRE COMMUNITY BONDING PERIOD :

- ➔ Contributing to the webcomponent repo primarily to the creation of data type web components.(creation of reference(merge request already present) and codeable concept type will be a high priority)
- ➔ Explore more about EHR and FHIR.
- ➔ Setup a React application and attempt implementation of one basic workflow.

COMMUNITY BONDING PERIOD :

- ➔ Finalise the EHR screens and workflows with detailed UI UX design.
- ➔ Familiarize with other participants and mentors.
- ➔ Continuous communication with the mentors.
- ➔ Develop CD/CI to integrate unit tests and integration tests using SauceLabs.
- ➔ Design skeleton PWA to show end-to-end architecture..

CODING PHASE :

WEEK	TASK
Week 1	- Setup the project - Implement the basic login, home screen - Implement the **check-in workflow**
Week 2 and Week 3	- Create the web components for the appointment workflow (including docs and tests) - Implement the **appointment workflow**
Week 4 and week 5	- Create the web components for the visit workflow (including docs and tests) - Implement the **visit workflow** (this includes the **order workflow**)
Week 6	- Complete pending work if any. - Solve bugs and issues
Week 7 and week 8	- Create the components for the e-prescription workflow (including docs and tests) - Implement the **e-prescription workflow**
Week 9	- Create the admin page, user profile page and pwa integrations
Week 10	- Complete pending work if any. - Solve bugs and issues

PREREQUISITES AND CONTRIBUTIONS

Creating web components for a resource: (Merge Request)
Identify list of missing components : COMPONENTS REQUIRED :

MERGE REQUESTS :

- Refactoring package structure (OPEN)
- Creating reference type component (OPEN ISSUE)
- Text fixture and component modification (MERGED)

ISSUES :

- Refactoring package structure (LINK)

ABOUT ME

I am a second year student from National Institute of technology tiruchirappalli, India pursuing a degree in Bachelor of Technology. I have been a full-stack developer at my college based organisation Delta Force since August 2020. I have a keen interest in exploring new things. My interest in working on projects that impact and help people brought me to LibreHealth.

I am willing to commit **45-50 hrs per week** from the beginning of the coding period till July end (till my college remains closed for the summer break), from august beginning till the end of the program I am willing to commit **40-45 hrs per week.**

I do not have any major commitments till late August and hope to spend quality time making open source contributions for LibreHealth.

MAJOR PROJECTS:

PRAGYAN PREMIER LEAGUE | (ReactJs + Golang)
Pragyan Premier League is a virtual cricket league in which matches are algorithmically simulated.The application goes live every year during Pragyan (a college fest).

I was a part of the team that implemented new and real-time features for the application using Golang and Gorilla among other technologies including ReactJs and redux.

The game is live at this point of time : PPL (as of april 7)

PROJECT MANAGEMENT WEBSITE **|** (Reactjs + laravel)
Website for management of projects being carried out by Delta Force (college based organisation).

Worked as a full stack developer implementing Laravel in the Backend and React in the frontend. Link

OPEN SOURCE CONTRIBUTIONS:
Apart from LibreHealth i have made some significant open source contributions for bench routes

TECH STACK:

Languages:
- C++
- Python
- Javascript
- Golang
- php

web development:
- Nodejs
- Laravel
- Gin
- ReactJs
- redux
- Polymer
- Flask
- nextJs

GSoC

Have you previously participated in GSoC ? when ? under which project ?
-> I haven't participated in GSoC before.

Are you also applying to other projects?
-> NO