Fullstack challenge

Patient Tracking System - MERN/MEAN

[Problem Statement]

Develop and Enhance a full stack web app to manage patients at a hospital.

You are expected to upgrade a web app written in NodeJS, Angularjs, Angular and React stack, And for your database use MongoDB.

The source code for this application has been provided along with a description of the tasks needed to accomplish.

****With the problem statement, we have also provided the skeleton of the solution which is incomplete. You are expected to understand the skeleton thoroughly and complete most of the tasks specified here.****

To run skeleton code use:

Angular 2:

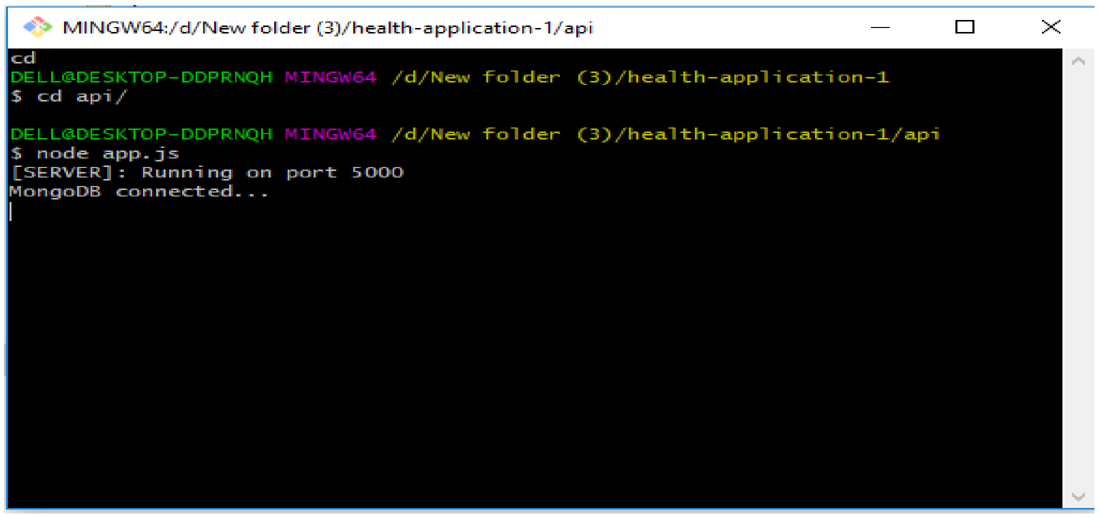
ng serve

Angular 1:
grunt serve

React :
npm start

node [backend]:
Path : projectName/api/app.js
cmd : node app.js

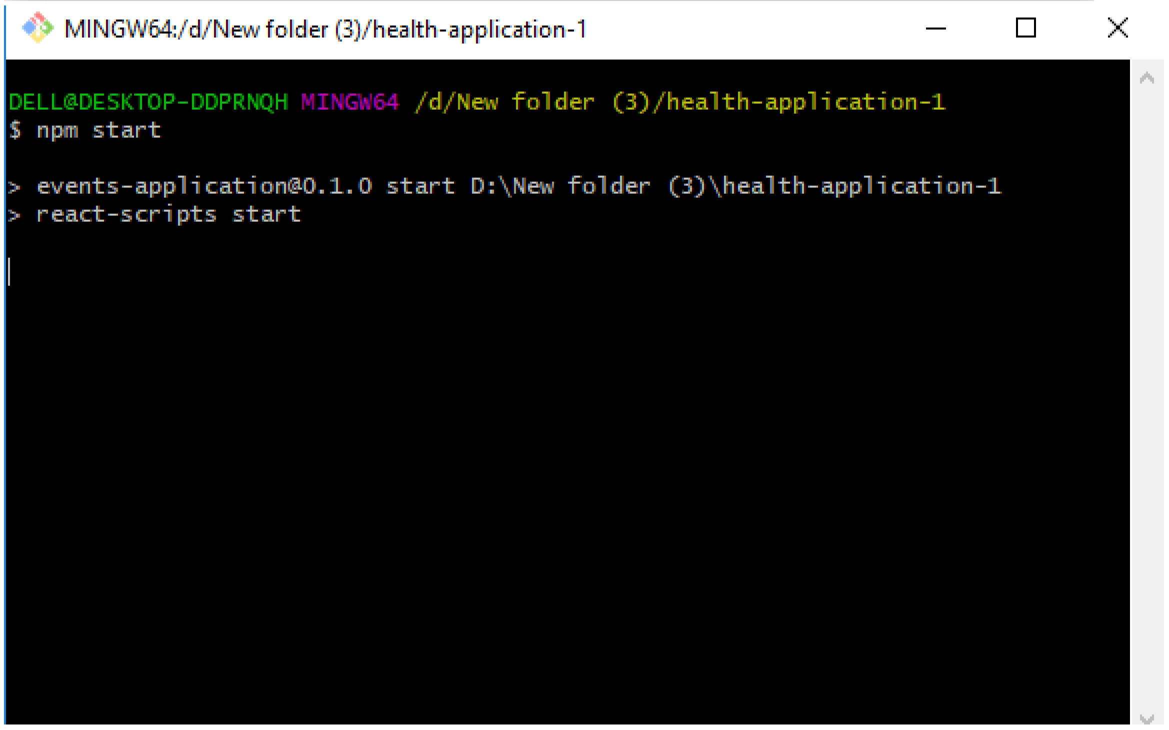
Sample Screenshot :



MINGW64:/d/New folder (3)/health-application-1/api

```
cd
DELL@DESKTOP-DDPRNQH MINGW64 /d/New folder (3)/health-application-1
$ cd api/

DELL@DESKTOP-DDPRNQH MINGW64 /d/New folder (3)/health-application-1/api
$ node app.js
[SERVER]: Running on port 5000
MongoDB connected...
```



MINGW64:/d/New folder (3)/health-application-1

```
DELL@DESKTOP-DDPRNQH MINGW64 /d/New folder (3)/health-application-1
$ npm start

> events-application@0.1.0 start D:\New folder (3)\health-application-1
> react-scripts start
```

You are required to extend the application and build the following additional features:

[Minimum requirement] (level 0)

[Frontend Task]

Add Patient Component: It should consist of the following features:

- **Patient Table**: The table should have four columns, Patient Name, Age, Phone Number, and Address. The table contains all the a patients. Also, the table should contain patient's age in the table. UI for the same needs to be created.
- **Add Drugs Icon**: On Clicking the **Add Drugs button**, the view should show a modal to add drugs and how to take them for that patient.
- **Add Patient button**: On clicking **Add Patient button,** the view should switch to **Add Patient Component**.
- UI layer is responsive. Mobile UI should match that of mock provided. Provide a breakpoint at 960px

[Backend Task] -

- **Endpoint for updating a patient.** Create an endpoint that allows for updating a patient.
- **Endpoint for getting all patients.** Create an endpoint that allows for getting all patients from the MongoDB database.
- Proper REST API calls at (\add), (\update\<id>) and (\delete\<id>)
- Proper unit tests and code coverage for Crud functionalities

[Plus point] (level 1)

[Frontend Task]

Add Patient Component: It should consist of the following features:

- **Patient Name Input**: The user should be able to enter the patient name in it.
- **Age Input**: The user should be able to enter the patient age in in it
- **Phone Number Input:** The user should be able to enter the phone number of the patient in it.
- **Address Input:** The user should be able to enter the patient address in it
- **Save Button**: The patient should be added and should redirect to the **View Patients Component** where the new patient should be reflected along with the added drugs.
- Sort and filter based on date, category and alphabetical order
- Implement additional features like tags bookmarks ratings
- Implementing bulk delete functionality
- Responsive web design for all the views
- Provide timestamp for add/modify activities on the home page
- Proper validations should be given if forms are used(messaging/alerts etc.,)

[Backend Task]

- **Endpoint for Creating a Patient.** Create an endpoint that allows for adding a Patient. It should accept as body at least:
    - firstName [string with length between 3 and 24 characters]
    - lastName [string with length between 3 and 24 characters]
    - Phone [10 digit numeric input]
    - Address [string with max length 1000 characters]
    - consultedBy [[string with length between 3 and 24 characters]]
    - Age [number between 0 and 100]
    - Gender [string either male or female]
    - Consulted [boolean]
    - Created_at [date-time stamp]
    - Complains [string with max length 1000 characters]
    - Results [string with max length 1000 characters]
    - Prescriptions [string with max length 1000 characters]

All attributes are required except **[consulted, complains, created_at, results, prescriptions]**
Pagination of 20 items. After reaching the end the application should make another call for the database for the next 20 items.

Implementation of Login/Logout functionality, on the home screen, including user authentication. Either use Google API for authetication or create a manual login system and store passoword in hash.

[Plus point] (level 1)

[Frontend Task]

Add Drugs for Patient: It should consist of the following features:

- **Name of drug**: The user should be able to enter the of the drug in it
- **Number of times**: The user should be able to enter the number of times in it.
- **The method of taking the drug drop down**: The user should be able to select how the drug is to be taken
- **Save Button**: The drug should be added to t
- he list of drugs for the patient and should redirect to the **View Patients Component** where the new patient should be reflected along with the added drugs.
- Provide UI test cases using Karma, Selenium etc.
- Connecting the complete end to end flow through routing mechanism

- Acceptable scaffolding/asset management and code management by using gitlab
- Session and state management across the application/Lazy Loading

[Backend Task]

The Update Endpoint should work for this too

- Use of git to manage source code. Create different branches for FrontEnd, Backend etc,
- Use of proper design patters like Factory, Service, Singleton for all CRUD functionalities

[Advanced] (level 2)

[Frontend Task]

Create Update Patient Component: It should consist of the following features:

- **Patient Name Input**: The user should be able to enter the new patient name in it. By default, the name being edited should already be in the box.
- **Age Input**: The user should be able to enter the new patient age in in it. By default, the new age being edited should already be in the box.
- **Phone Number Input:** The user should be able to enter the new phone number of the patient in it. By default, the new phone number being edited should already be in the box.
- **Address Input:** The user should be able to enter the new patient address in it. By default, the new address being edited should already be in the box.
- **Update Button**: The patient should be updated and should redirect to the **View Patients Component** where the updated patient should be reflected along with the added drugs.

Delete Patients: It should consist of the following features:

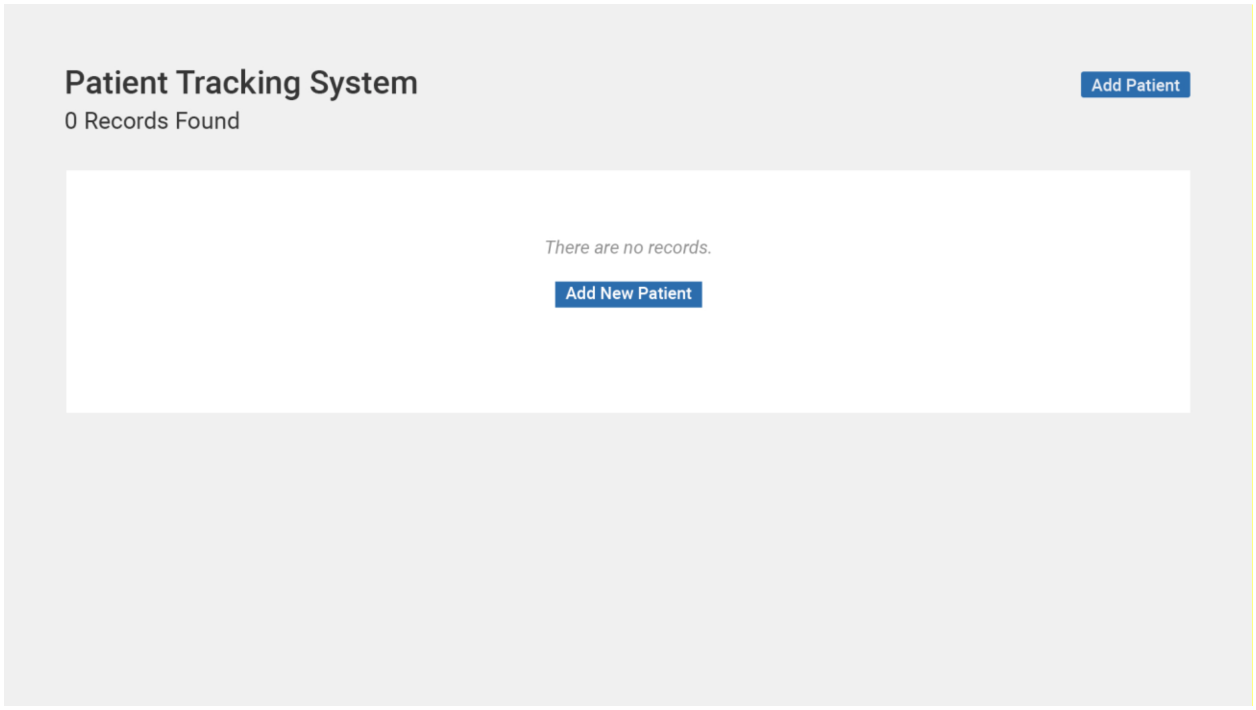
- **Delete patient button**: The user should click the archive patient button/icon and the patient should be removed from created patients and added to archived patients.
- When trying to delete a patient, a password should be required, otherwise a message stating otherwise should be displayed. This password should be **“1234”**

[Backend Task]

- **Endpoint for updating a patient.** Create an endpoint that allows for updating a patient.
- **Endpoint for deleting a patient.** Create an endpoint that allows for deleting a patient given their ID

[Mockup References]:

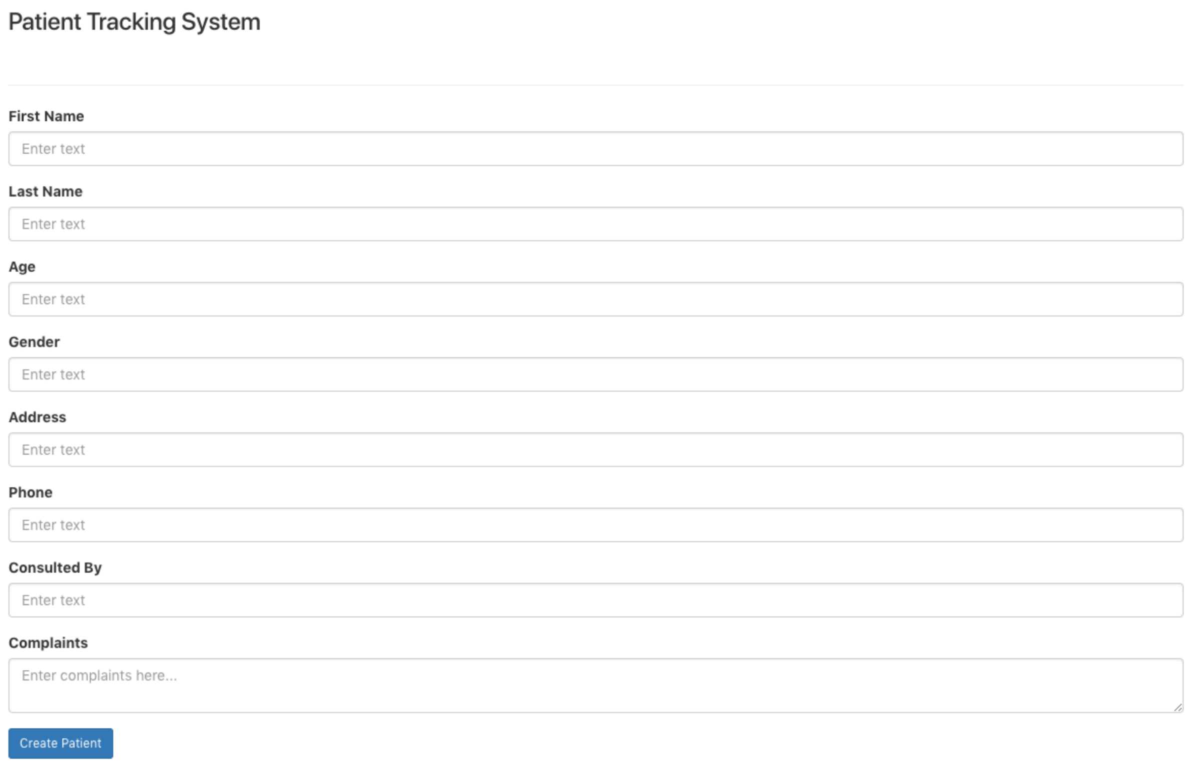
Home Page


Patient Tracking System

0 Records Found

There are no records.

On making a call at (\add)


Patient Tracking System

First Name

Enter text

Last Name

Enter text

Age

Enter text

Gender

Enter text

Address

Enter text

Phone

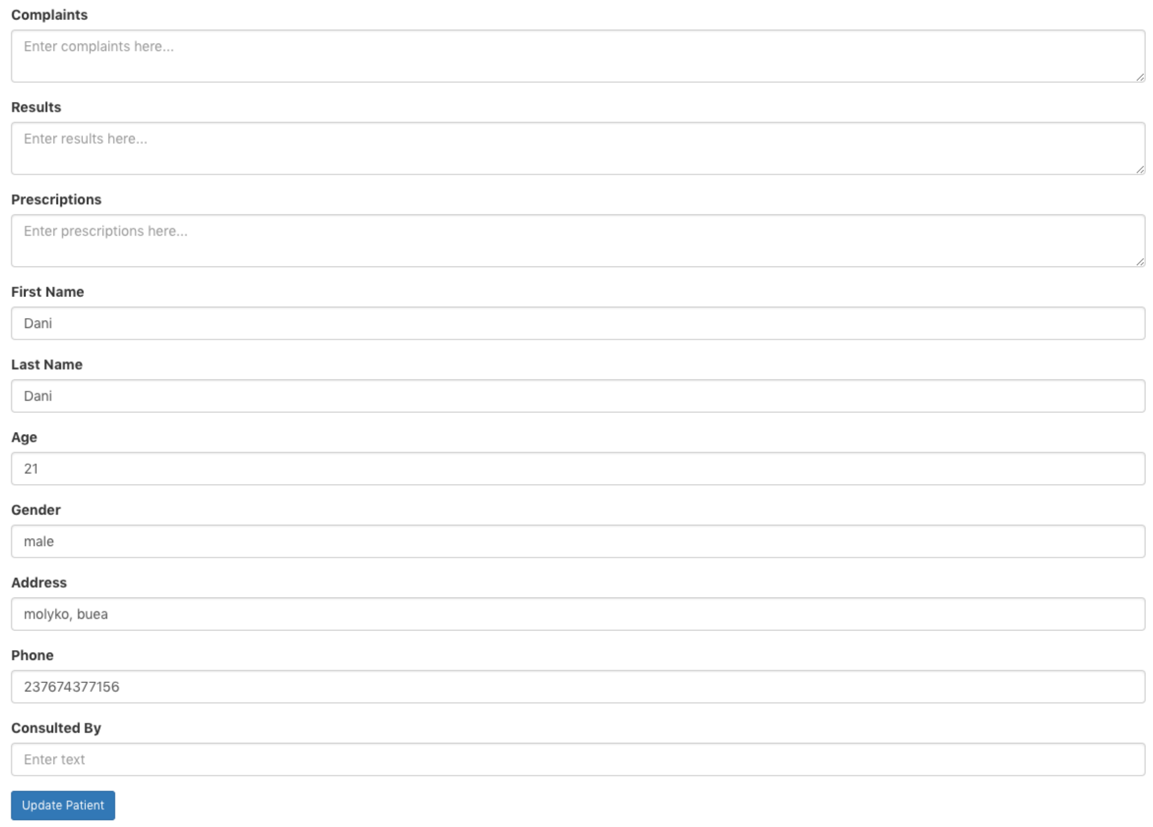
Enter text

Consulted By

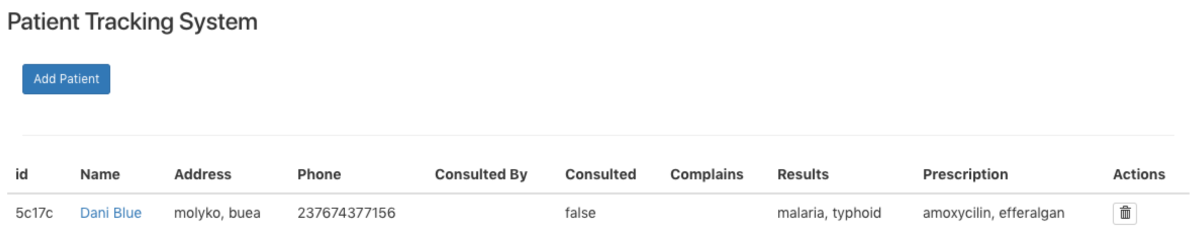
Enter text

Complaints

Enter complaints here...



Complaints	Enter complaints here...
Results	Enter results here...
Prescriptions	Enter prescriptions here...
First Name	Dani
Last Name	Dani
Age	21
Gender	male
Address	molyko, buea
Phone	237674377156
Consulted By	Enter text



Patient Tracking System

id	Name	Address	Phone	Consulted By	Consulted	Complains	Results	Prescription	Actions
5c17c	Dani Blue	molyko, buea	237674377156		false		malaria, typhoid	amoxycilin, efferalgan	🗑

On making a call at (\update/\d)

Patient Tracking System

First Name

Enter text

Last Name

Enter text

Age

Enter text

Gender

Enter text

Address

Enter text

Phone

Enter text

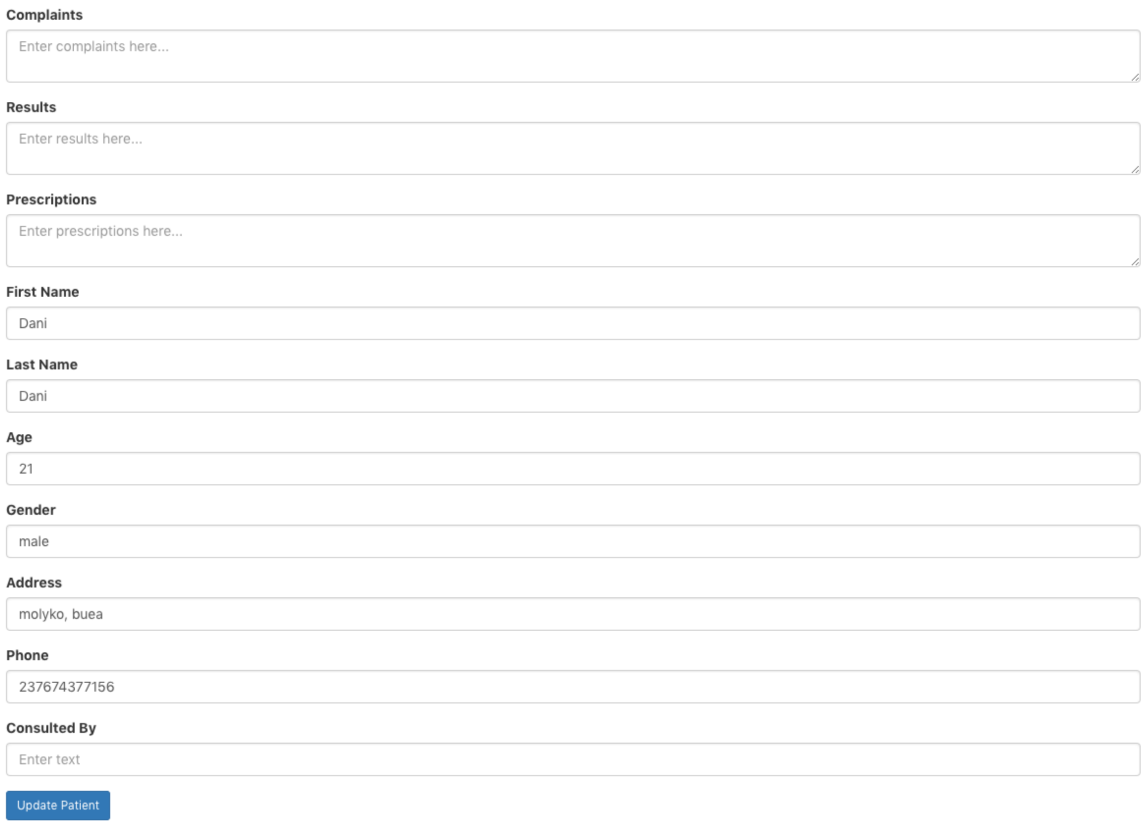
Consulted By

Enter text

Complaints

Enter complaints here...

Create Patient



Complaints

Enter complaints here...

Results

Enter results here...

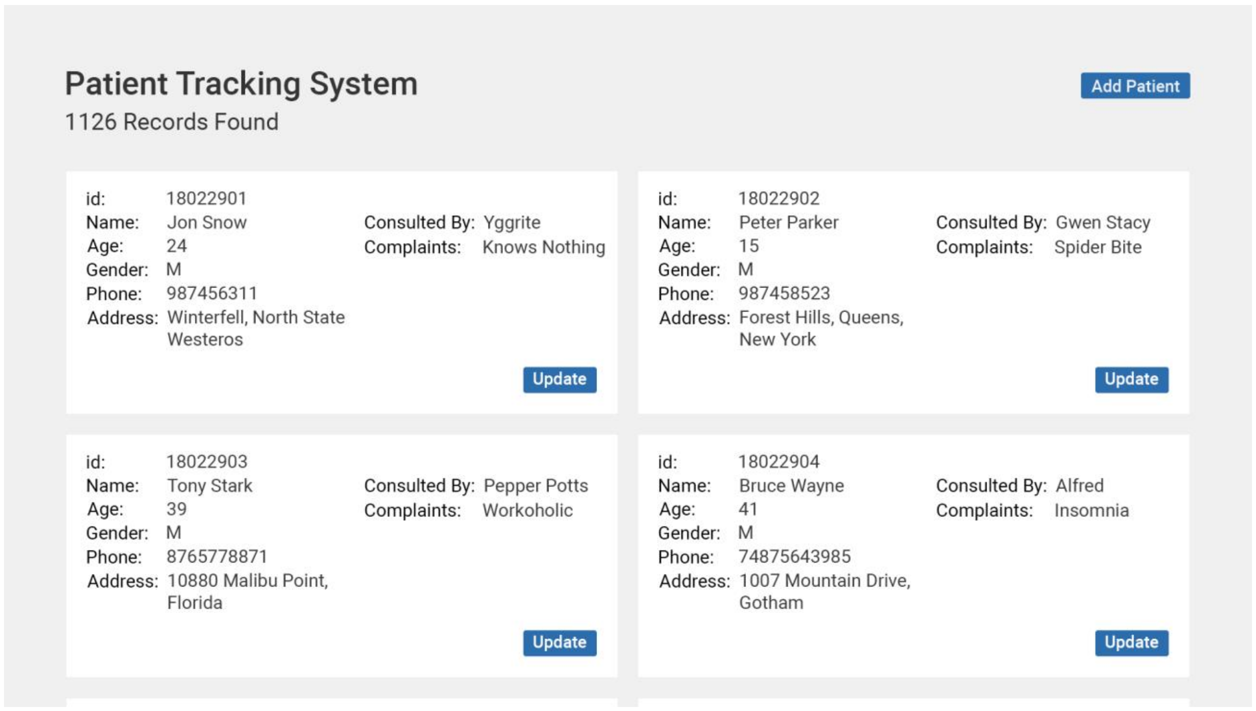
Prescriptions

Enter prescriptions here...

First Name	Dani
Last Name	Dani
Age	21
Gender	male
Address	molyko, buea
Phone	237674377156
Consulted By	Enter text

Recommended Mockup HomePage

Web UI



Patient Tracking System
1126 Records Found
Add Patient

id:	18022901		
Name:	Jon Snow	Consulted By:	Yggrite
Age:	24	Complaints:	Knows Nothing
Gender:	M		
Phone:	987456311		
Address:	Winterfell, North State Westeros		

Update

id:	18022902		
Name:	Peter Parker	Consulted By:	Gwen Stacy
Age:	15	Complaints:	Spider Bite
Gender:	M		
Phone:	987458523		
Address:	Forest Hills, Queens, New York		

Update

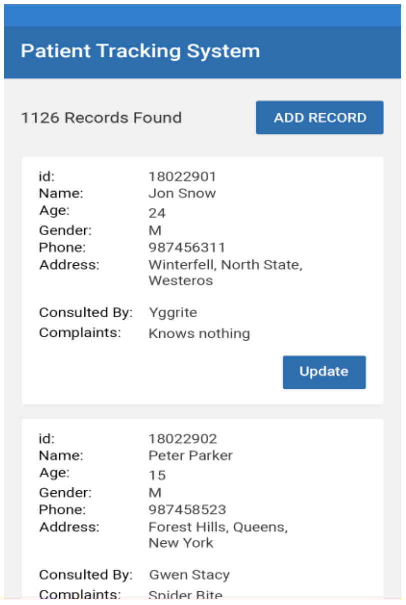
id:	18022903		
Name:	Tony Stark	Consulted By:	Pepper Potts
Age:	39	Complaints:	Workoholic
Gender:	M		
Phone:	8765778871		
Address:	10880 Malibu Point, Florida		

Update

id:	18022904		
Name:	Bruce Wayne	Consulted By:	Alfred
Age:	41	Complaints:	Insomnia
Gender:	M		
Phone:	74875643985		
Address:	1007 Mountain Drive, Gotham		

Update

Mobile UI



Patient Tracking System

1126 Records Found
ADD RECORD

id:	18022901
Name:	Jon Snow
Age:	24
Gender:	M
Phone:	987456311
Address:	Winterfell, North State, Westeros
Consulted By:	Yggrite
Complaints:	Knows nothing

Update

id:	18022902
Name:	Peter Parker
Age:	15
Gender:	M
Phone:	987458523
Address:	Forest Hills, Queens, New York
Consulted By:	Gwen Stacy
Complaints:	Spider Bite

[Accepted Solution]

- You accepted solution will allow for a user to create patients, edit, update and delete patients
- Data will be saved to a MongoDB database
- The REST/RESTful API should be built in NodeJS

[Solution Approach]

- No knowledge of CSS is required, though you are required to build the HTML and bootstrap forms
- You are also required to build the REST API using NodeJS and Express with MongoDB
- Also, you are required to configure routing for the application using react-router-dom which has already been installed.
- All the necessary files have been created, so you don't need to create any extra files.

[Guide]

Code Skeleton:

TechStack :

Frontend: Angular, Angularjs, Reactjs

Backend : NodeJS,

Database : MongoDB