

2020

## Disease Navigation Application

Naveena Varna  
*Grand Valley State University*

Follow this and additional works at: <https://scholarworks.gvsu.edu/cistechlib>

---

### ScholarWorks Citation

Varna, Naveena, "Disease Navigation Application" (2020). *Technical Library*. 364.  
<https://scholarworks.gvsu.edu/cistechlib/364>

This Project is brought to you for free and open access by the School of Computing and Information Systems at ScholarWorks@GVSU. It has been accepted for inclusion in Technical Library by an authorized administrator of ScholarWorks@GVSU. For more information, please contact [scholarworks@gvsu.edu](mailto:scholarworks@gvsu.edu).

# **Disease Navigation Application**

Naveena Varna

A Project Submitted to

GRAND VALLEY STATE UNIVERSITY

In

Partial Fulfillment of the Requirements

For the Degree of

Master of Science in Applied Computer Science

School of Computing and Information Systems

<December> <2020>



## **Abstract**

Telemedicine is in the current line of requirements of modern society. Automated symptom-based disease detection and appointment booking is a well-known research topic in the field of informatics and system design. That helps in elevating awareness and early detection of disease at the ease of home through appointment booking.

The purpose of the application is the capability to articulate disease symptoms, gathering all relevant information of the patient and a recommendation system that eventually evaluates all symptoms and maps effectively to a specialty department. Using the natural language processing-based approach, the user can input text of symptoms to be mapped with Google Search API and then finally parse them and generate recommendations based on the user input. Then it enables the user to choose a doctor and book an appointment based on the availability of that doctor.

The application is developed by using HTML, CSS, and JavaScript to build the front end and using microservice API exposed as REST built with the Django Python framework to build the back end. Using an SQLite database for the back end of the application helps to store the data locally and makes the application functional. In addition, appointment booking, and disease recommendation are two inevitable features from an end-user perspective.

## **Introduction**

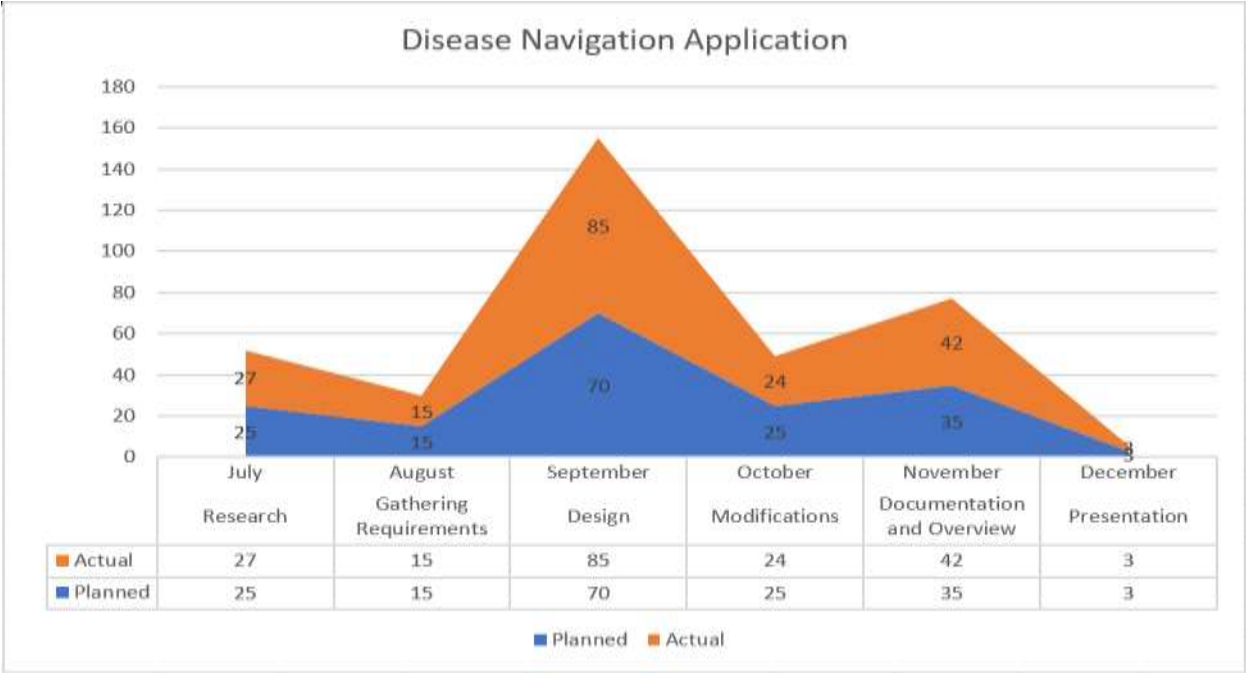
Telemedicine is in the current line of the requirement of modern society. That helps in elevating awareness and early detection of disease at the ease of home. In this project, I have designed an application named – Novi Med. This is a web application built with the Python Django framework. It can articulate disease symptoms, gathering all relevant information of the patient and a recommendation system that eventually evaluates all symptoms and maps effectively to a specialty department. And then it enables the user to choose a doctor and book an appointment based on availability. If a slot is available, the booking will be confirmed, and the appointment number will be displayed over the screen of the app. In this application appointment booking, and disease recommendation are two inevitable features of this app from an end-user perspective. In the following sections, we will be discussing them in more detail.

The website will allow the user (patient) to be able to get the probable disease that the user may be facing. This is done by the disease detector that uses a database of diseases and symptoms and then presents the under with probable diseases. The patient can then book appointments with different doctors specialized in a department.

The website has three user types of users: Doctors, Admin, and User (Patient). A Doctor will be available for a patient to get an appointment for a disease that has been detected by the detector. The administrator will manage or control the website. The user can get detection for the symptoms and get an appointment with a doctor. The Disease Navigation comprises a disease database that helps users to know what are the probability that user is

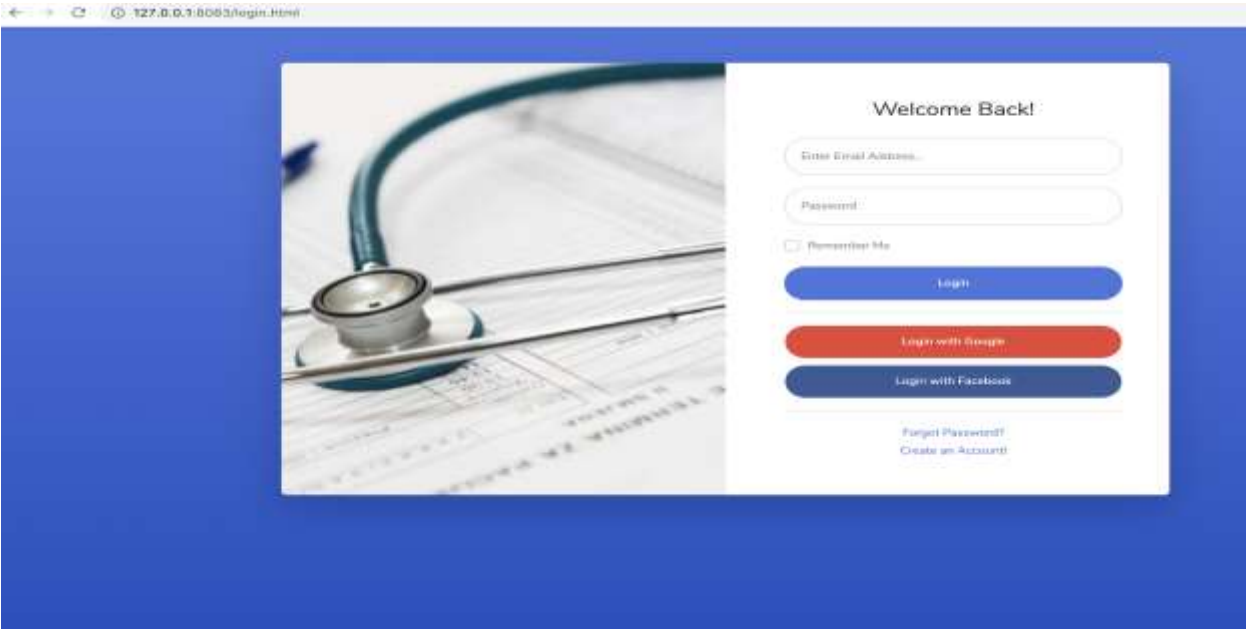
suffering from a disease and get an appointment to a specialized doctor in the disease. The doctor can view the appointments made and see the patients.

## Project Management



## Organization Requirements

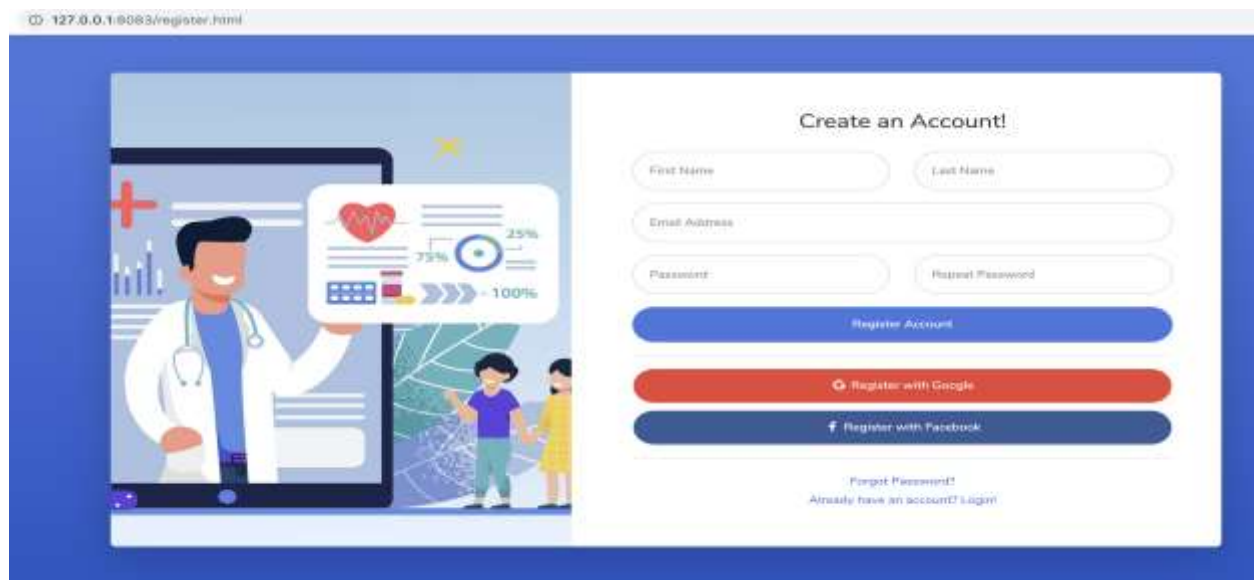
### Login/Sign Up screen



Login screen contains two fields Email(username) and password. Here users must put their credentials in textboxes given and then hit the button of Login. Once clicked it will trigger a back-end API to validate user login. If the user has provided valid input, then login will be successful, and it will redirect to the homepage. In case of failure it will display an alert message of - incorrect credential.

### Sign Up Page

In case of new user, they will be redirected to register.html page as and when sign up link is clicked. Sign up page contains a web form with Name, Email and Password field using which a user will login and use this application going forward.



### Home Page:

Home page of this application is based on a default dashboard view where a squeezable left panel contains other feature link like - disease diagnosis, History, Comments etc.

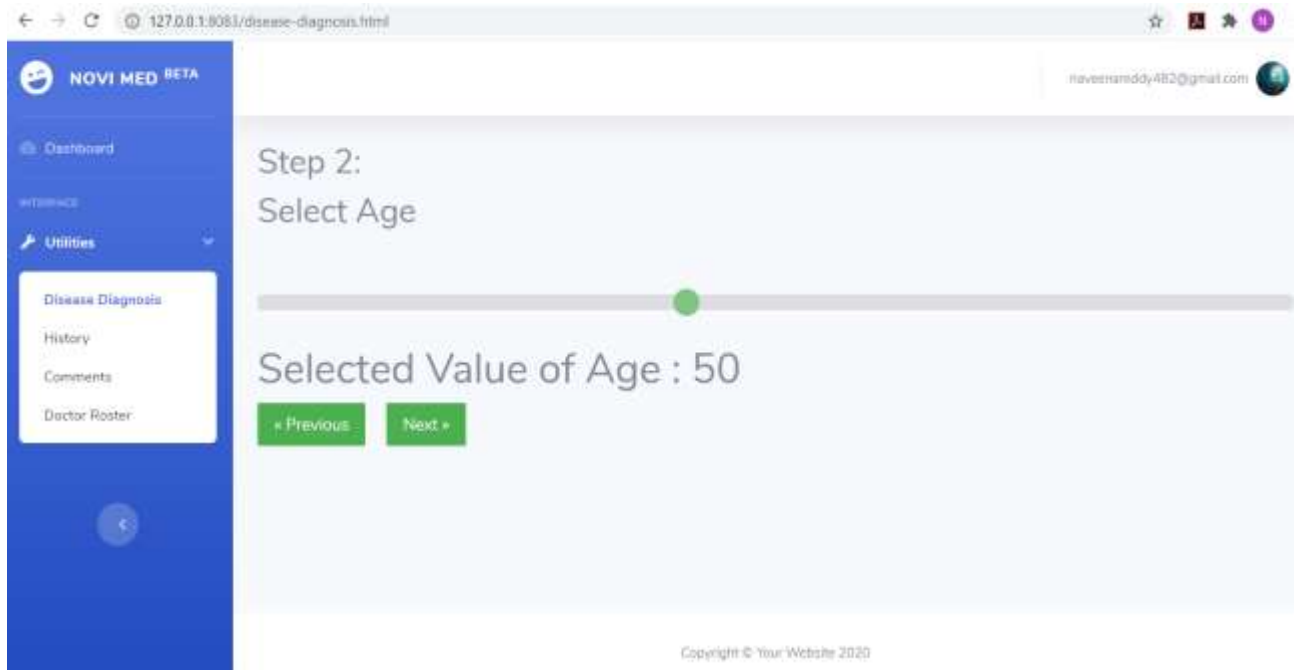


## Disease Diagnosis Page

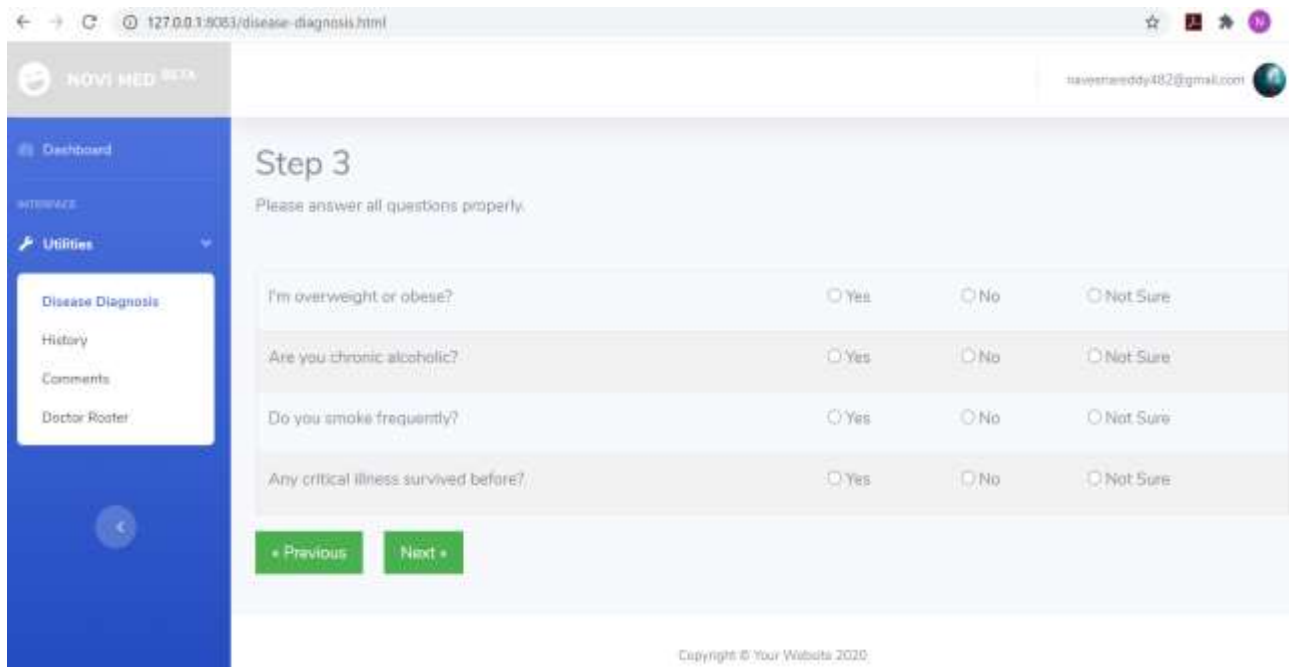
Disease diagnosis page consists a series of questionnaires through which a user will input all its relevant details which will be processed by App to diagnose the disease. The above screen is the first step. Where a user has to select his/her sex in an interactive way.



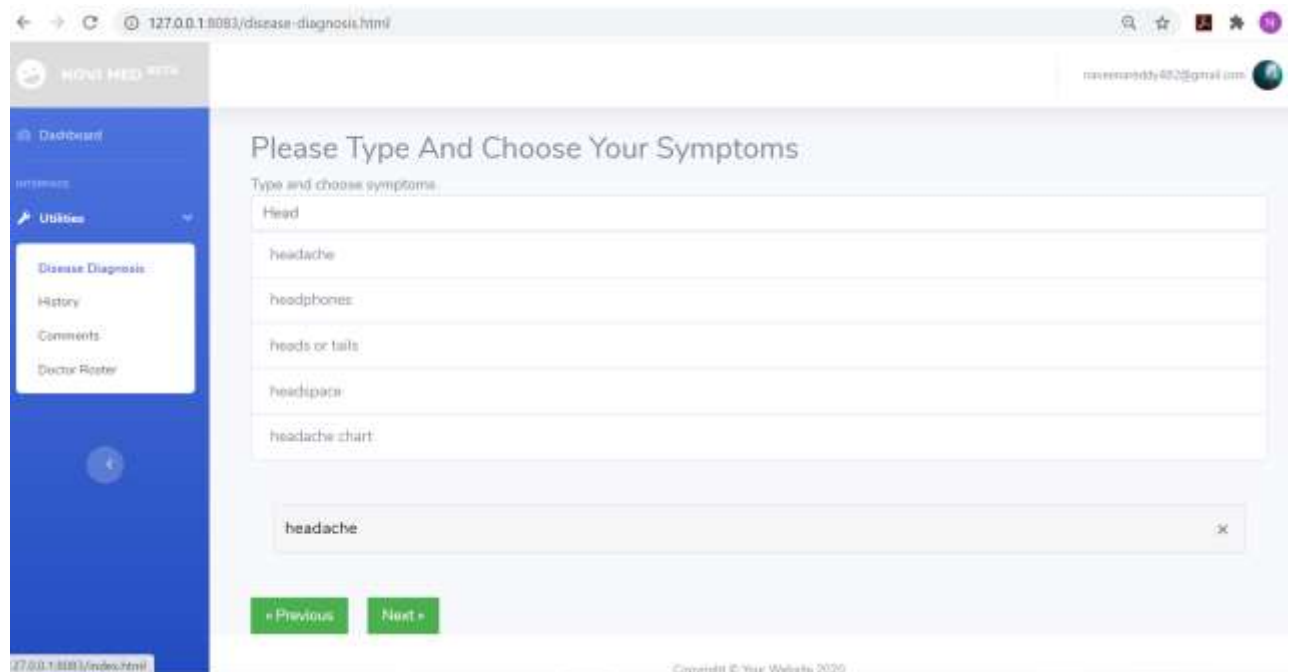
Next step is about selecting age, user will slide the slider to select his/her age and then next button will be clicked.



Next step users must answer some basic questionnaires which will help to generate more accurate recommendations.

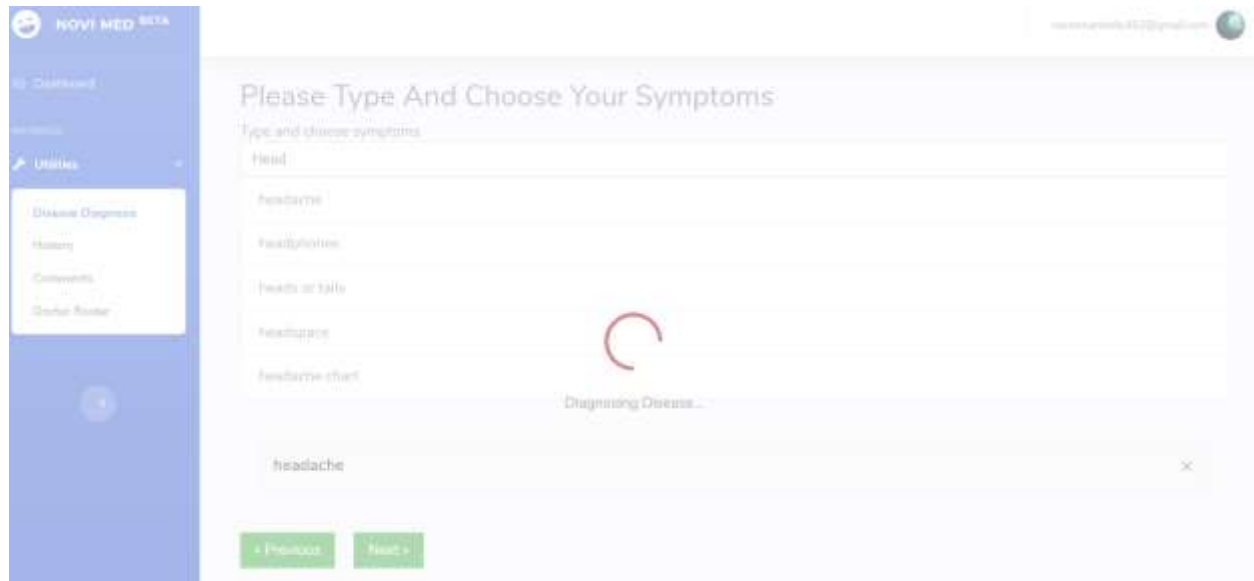


In this step users will type their symptoms in the shown textbox and the recommendation system will display recommended symptoms which the user will select. That way symptoms will be added to the applications. Also, they can be deleted by clicking on the close icon on symptoms.

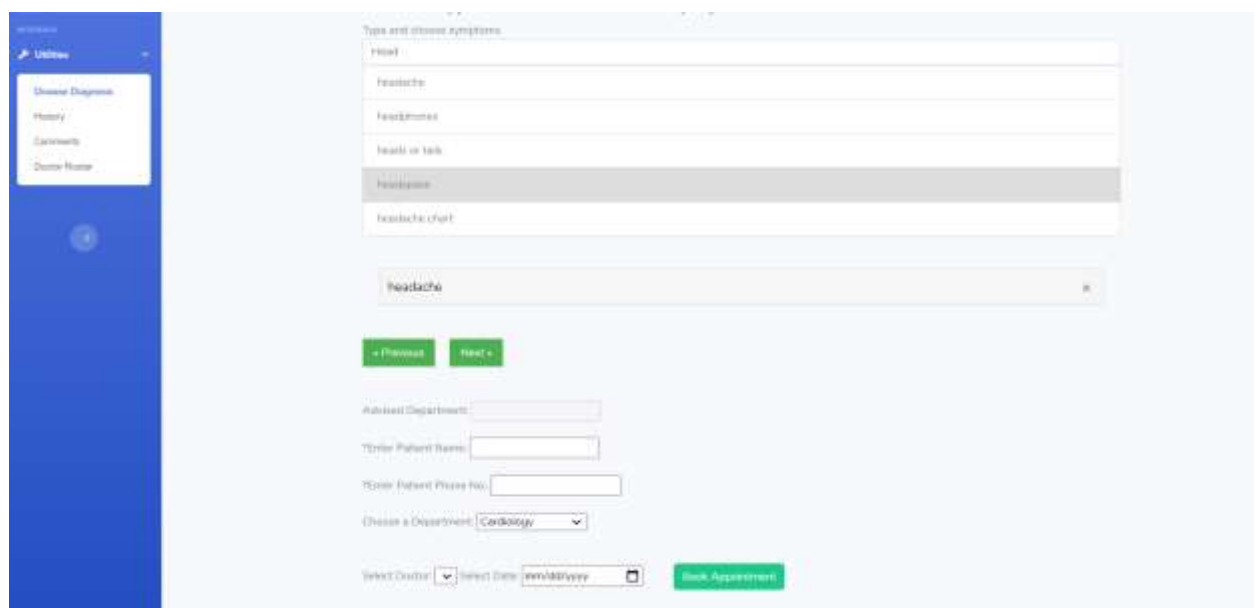




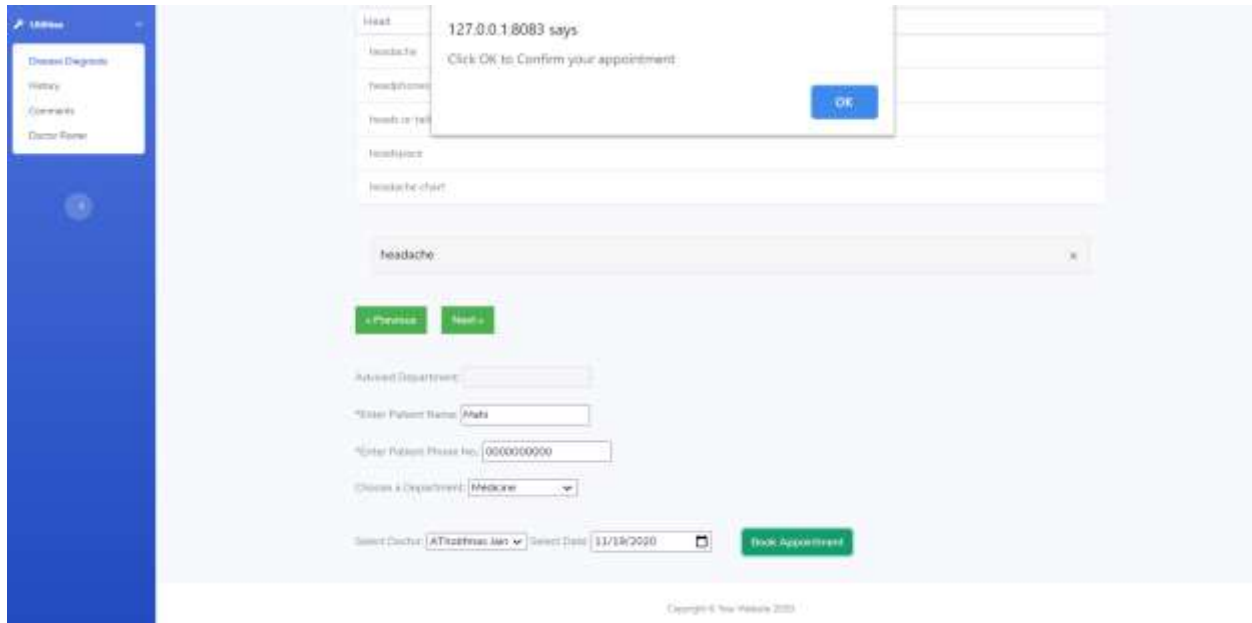
Once all symptoms are entered the user will click on the Next button that will trigger the diagnosis of symptoms by the back end of the application as shown in the screen below.



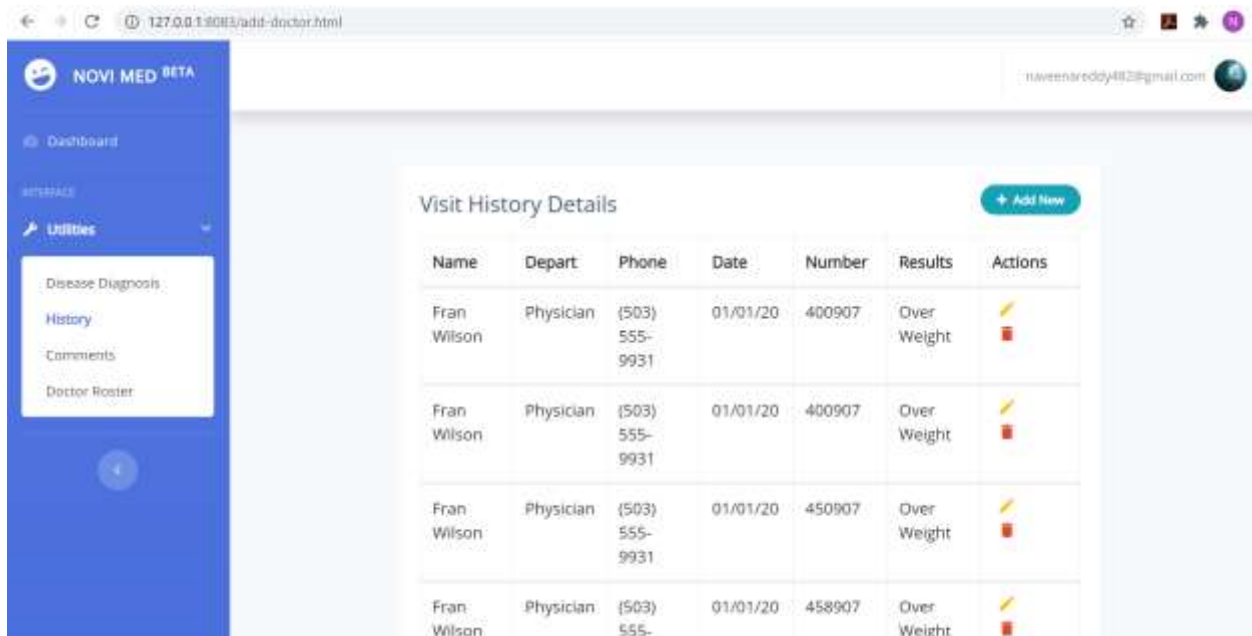
On completion of diagnosis a screen will be displayed with recommendation and appointment booking option as below.



Now the user will select the department, then select doctor and choose a date of appointment and finally click the book appointment button. Details are filled. Appointment will be confirmed only when doctor is available on selected date otherwise not. Above screen showing appointment confirmed status also displaying the appointment number here. In case of a doctor unavailable. It should display the same alert.



This module screen will enable the admin user to add patient visit summary details or update any existing details into the back-end database.



Comment-User can send the message to organization (Delete appointment or Any assistance user need).

ED BETA

naveenareddy482@gmail.com

Enter Department Name:

Enter Your Name-Provide Comments:

**\*\*Enter comma separated string of keyword.\*\***

SEND

Copyright © Your Website 2020

BETA

naveenareddy482@gmail.com

Enter Department Name:

Enter Your Name-Provide Comments:

**\*\*Enter comma separated string of keyword.\*\***

SEND


naveenareddy482@gmail.com

Enter Department Name:

Enter Your Name-Provide Comments:

**\*\*Enter comma separated string of keyword.\*\***

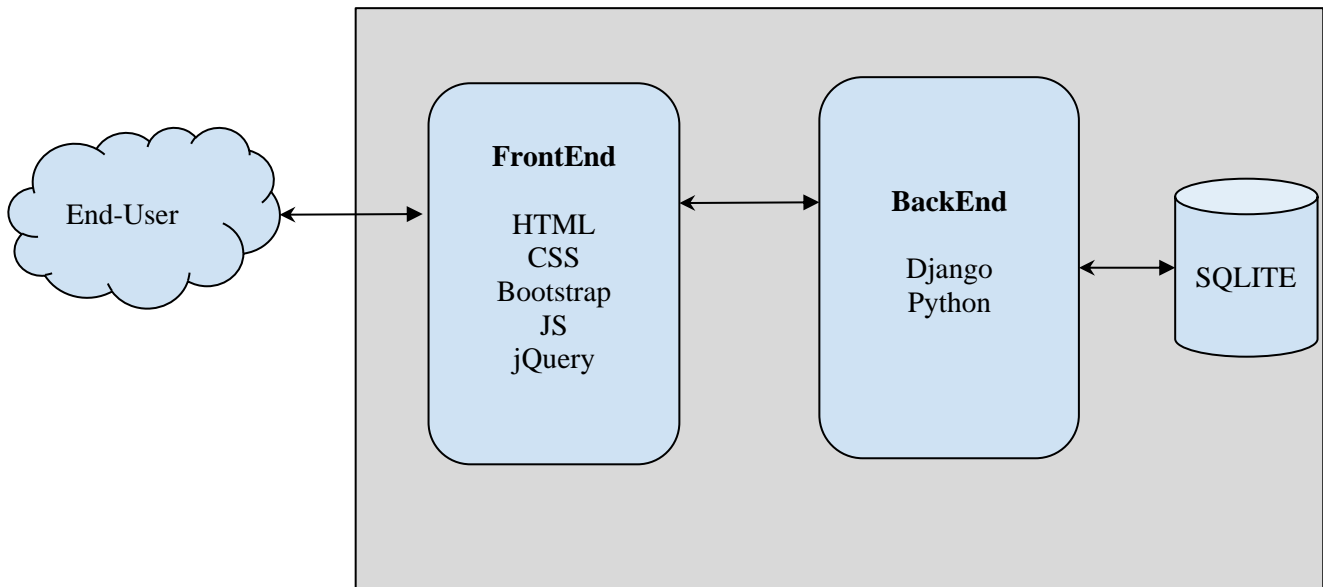
SEND



Sending Comments...

## Implementation

In this section we are going to discuss the design architecture of this application. We have chosen Microservice Design Pattern as it has many cutting-edge advantages that makes **App** more resilient and highly scalable. Application consists of two-layer front end and back end layer. Front end layer is built using bootstrap, html, CSS, JavaScript & jQuery. And the back end is a pure microservice API exposed as REST built with the Django python framework. Below diagram depicts the scenario more clearly.



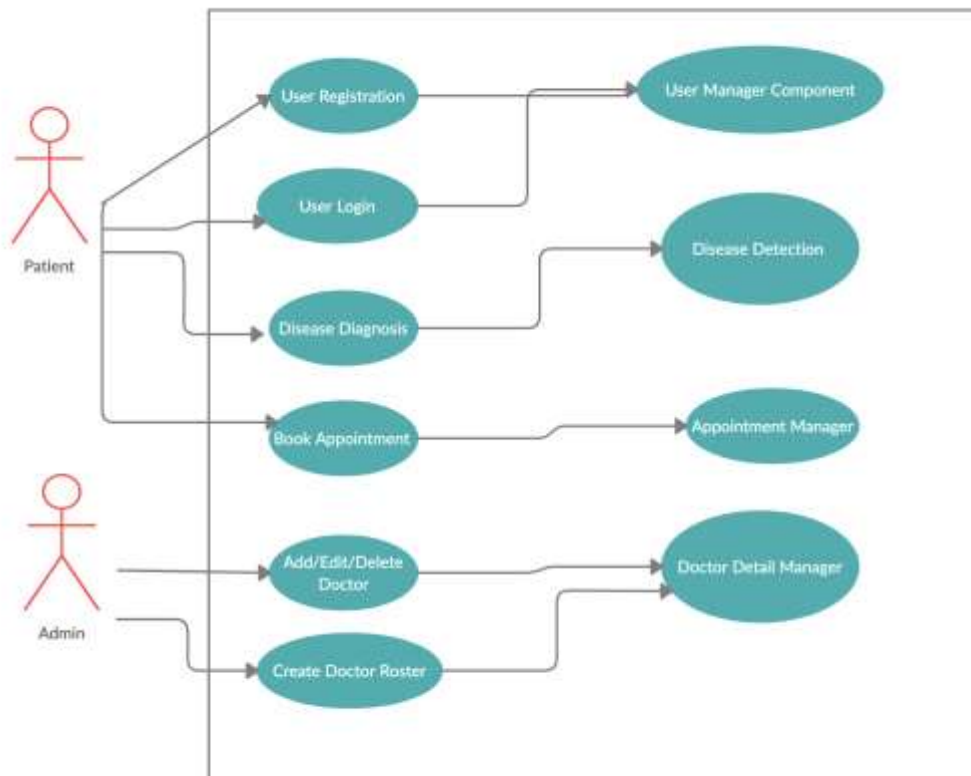
End user will render the web pages through the browser. Each front-end web page is mapped with a different back end Rest API as described in below table. Backend python script will serve the API request call of the frontend layer, it will use SQLite database as lightweight data storage. Where it will perform all sorts of crud operations.

REST API Referencing Table

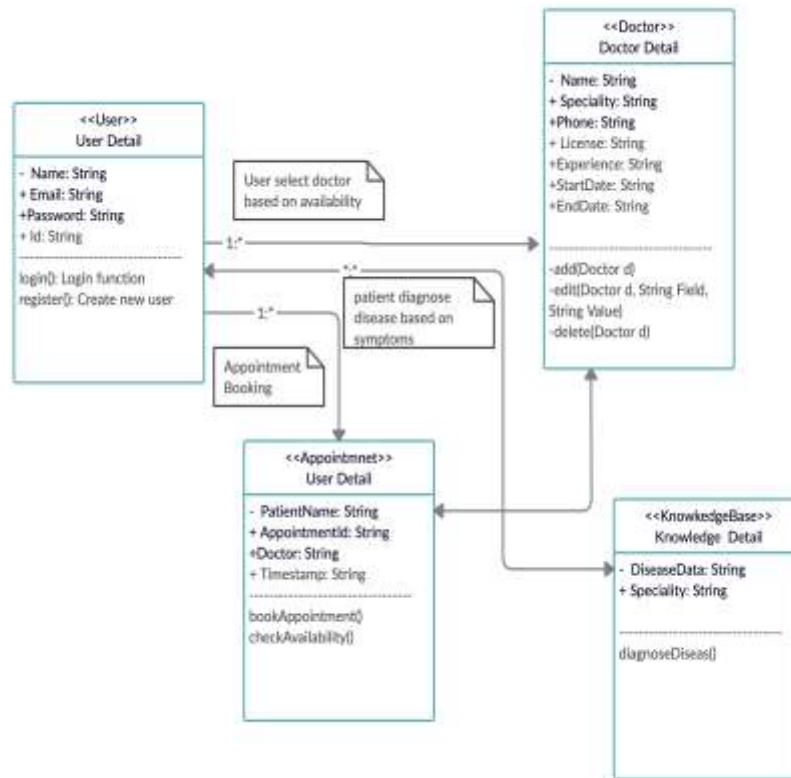
| API Name                      | Type | Request(Sample)   | Significance                                   |
|-------------------------------|------|---|--|
| <a href="#">add_patient</a>   | POST | { "Name": "AA BB", "Email": "sample@gmail.com", "Password": "*****" } | It creates an user profile for the patient     |
| <a href="#">login_patient</a> | POST | { "Email": "sample@gmail.com", "Password": "*****" }                  | It validates an patient user login             |
| <a href="#">get_diagnosis</a> | POST | { "Symptoms": "" }  | It returned a diagnosis recommendation.        |
| <a href="#">add_doctor</a>    | POST | { "Name": "Bilal Jain", "Phone": "1222222", "Speciality": "Medicine", | It creates a new doctor profile to the system. |

|                  |      |  |   |
|------------------|------|--|---|
|                  |      | "License": "US34590", "Experience": "6 YEARS", "Degree": "MD"                              |   |
| save_slot        | POST | {"Doctor": "Davis Das", "Dept": "Cardiology", "Start": "08/22/2020", "End": "08/25/2020" } | It updates doctor specific availability in the system.                  |
| book_appt        | POST | { }  | It creates new appointment for a patient                                |
| getall_doctor    | POST |  | Returns list of all doctor details in JSON formatted.                   |
| getall_dept      | POST |  | Returns list of all department details in JSON formatted.               |
| get_doctorByDept | POST |  | Returns list of all doctors of a specific department in JSON formatted. |

### Use Case Diagram



## Class Diagram:



## Results, Evaluation, and Reflection

In this section we are going to discuss all observations regarding results and evaluation metrics.

We have achieved an optimized performance in terms of computation time and accuracy. We will be discussing all aspects of it one by one.

### User Registration Average Response

| User registration request count | Processing Time |
|---------------------------------|-----------------|
| <u>1</u>                        | 0.552 ms        |
| <u>10</u>                       | 0.78 ms         |
| <u>13</u>                       | <u>1.2 ms</u>   |

### User Login Average Response

| User login request count | Processing Time |
|--------------------------|-----------------|
| <u>1</u>                 | 0.042 ms        |
| 10                       | 0.28 ms         |

|    |        |
|----|--------|
| 15 | 1.9 ms |
|----|--------|

Disease Diagnosis Average Response

| Disease Diagnosis request count | Processing Time | Accuracy |
|---------------------------------|-----------------|----------|
| <u>1</u>                        | 2 ms            | 96%      |
| <u>10</u>                       | 14 ms           | 89%      |
| 15                              | 111ms           | 87.625%  |

Book Appointment Average Response

| Appointment Booking Request | Processing Time |
|-----------------------------|-----------------|
| <u>1</u>                    | 167 ms          |
| 10                          | 299 ms          |
| 15                          | 2.5 sec         |

Doctor Details Average Response

| Request count | Processing Time |
|---------------|-----------------|
| <u>1</u>      | 0.042 ms        |
| 10            | 0.28 ms         |
| 15            | 1.9 ms          |

## Conclusion and Future Work

As per the current scope we have almost covered the use cases and program requirements provided in this implementation. But to make this **Novi Med – Decease Navigation App** as a full functional package there are many other features which need to be added. Some of them are Appointment tracker, Doctor Feedback, Mail and Message based integration etc. Still this application is quite capable enough to detect disease early based on the algorithms implemented and then recommend the patient towards a new appointment booking. Also, it has another two admin panels which will enable admin users to add doctors and maintain their availability roster. As it is built in Microservice architecture hence all components are independent and self-driven. So, having a high code reusability factor and it is very easy to integrate with other third-party applications. In terms of scalability and production readiness there is scope to add a docker file and containerize the application and then run that container in Kubernetes cluster.

## Bibliography

<https://doi.org/10.1016/j.ins.2018.01.001>

A. S. Hussein, W. M. Omar, X. Li and M. Ati, "Efficient Chronic Disease Diagnosis prediction and recommendation system," *2012 IEEE-EMBS Conference on Biomedical Engineering and Sciences*, Langkawi, 2012, pp. 209-214, doi: 10.1109/IECBES.2012.6498117.

Binal A. Thakkar, Mosin I. Hasan, Mansi A. Desai, "Healthcare decision support system for swine flu prediction using naïve bayes classifier", "IEEE", 101-105, 2010.

Sellappan P., Rafia A., "Intelligent heart disease prediction system using data mining techniques", IEEE/ACS, 2008.

<https://www.djangoproject.com/>

<https://www.w3schools.com/w3css/>

<https://www.w3schools.com/html/default.asp>

[https://www.w3schools.com/bootstrap/bootstrap\\_ver.asp](https://www.w3schools.com/bootstrap/bootstrap_ver.asp)

<https://app.creately.com/manage/recent>

[www.wikipedia.org](http://www.wikipedia.org)