SOFTWARE DESIGN DOCUMENT

for

HomeDoc-Doctor Appointment Website

Prepared by:

1. Aswin K Suresh(CCE20CS019)
2. Anchita Sajeev(CCE20CS016)
3. Akash Antony(CCE20CS005)
4. Shravan Freddy(CCE20CS054)

Submitted to : Ms.Jasmine Jolly

Assistant Professor

CSE Department

Guide

Contents

1 System Architecture Design…………………………………………………….3

2 Application Architecture Design………………………………………………4

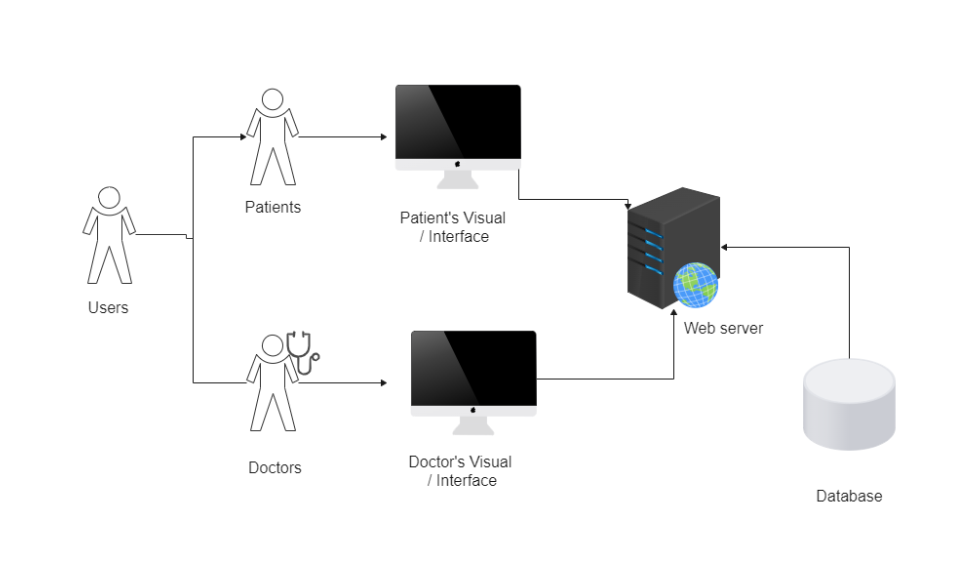
3 GUI Design………………………………………………………………………………5

4 API Design………………………………………………………………………………6

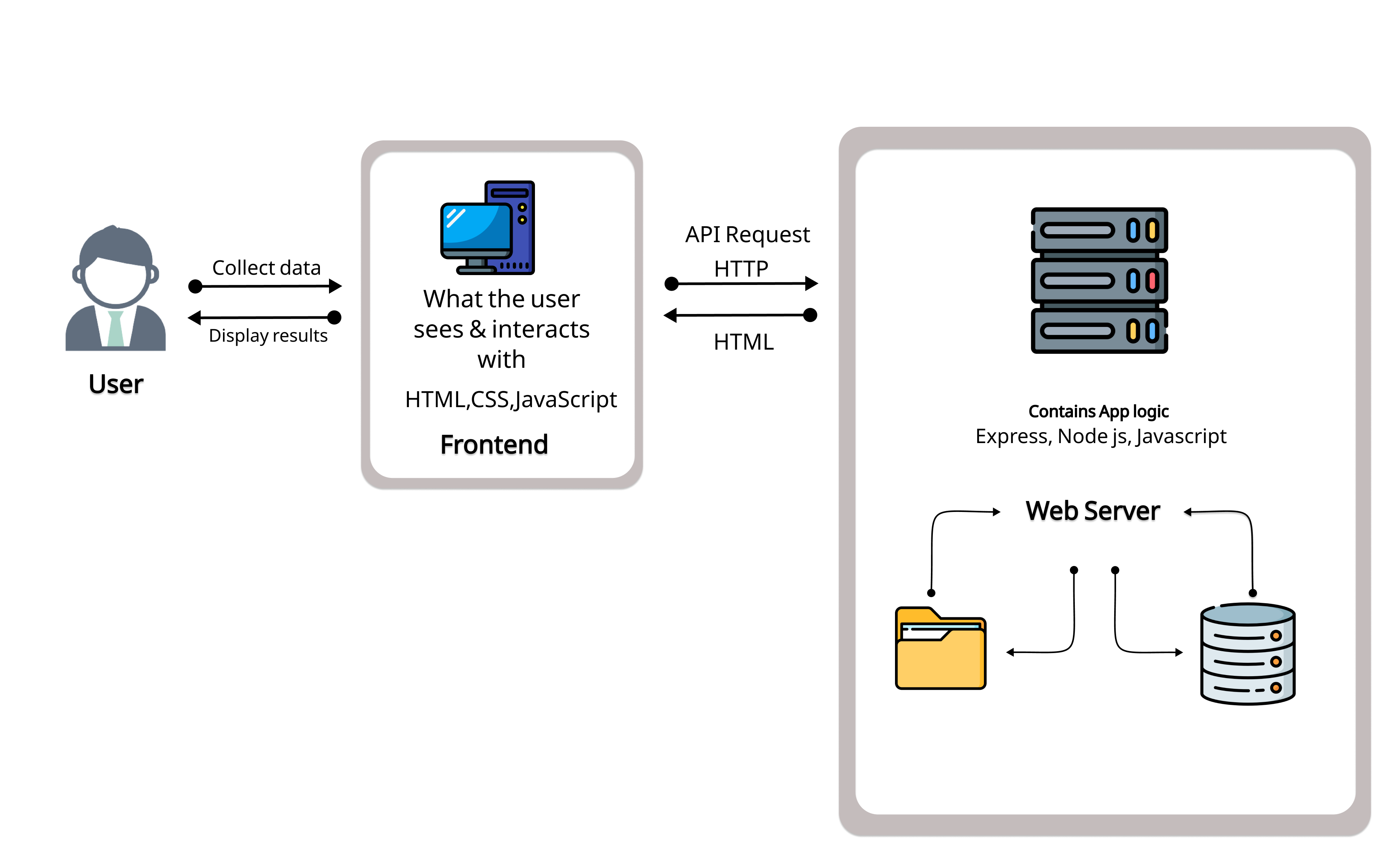
5 Data Base Design……………………………………………………………………..8

6 Technology Stack……………………………………………………………………..9

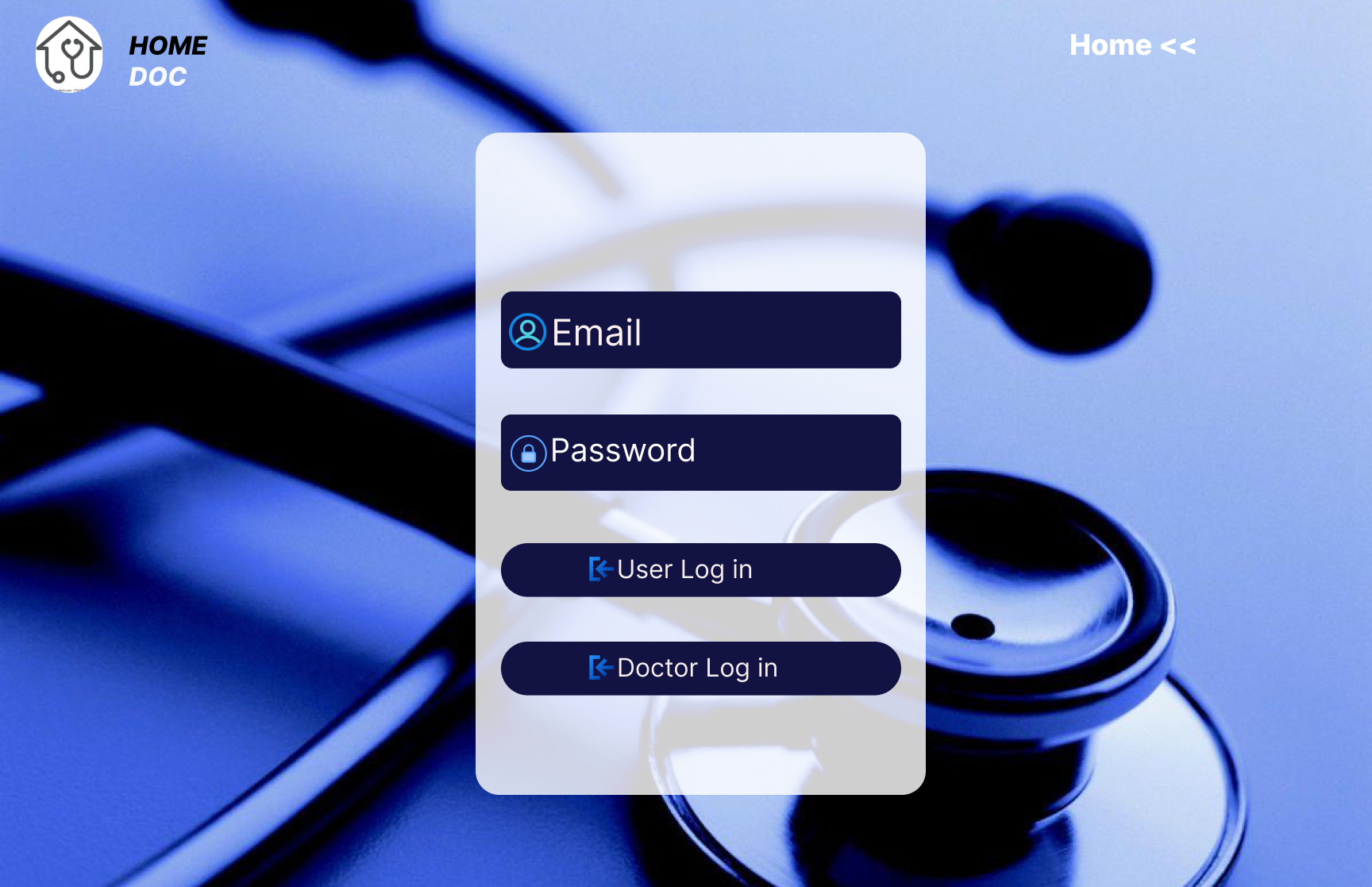
1.System Architecture Design

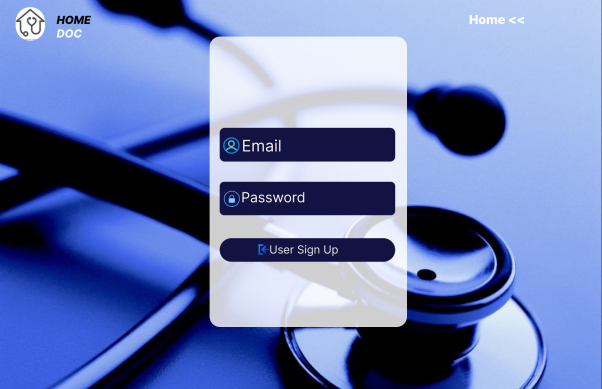


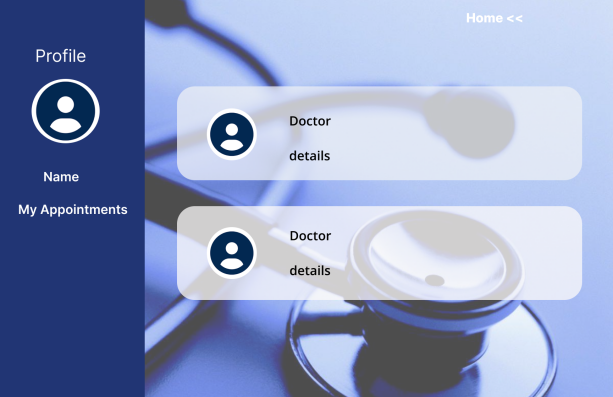
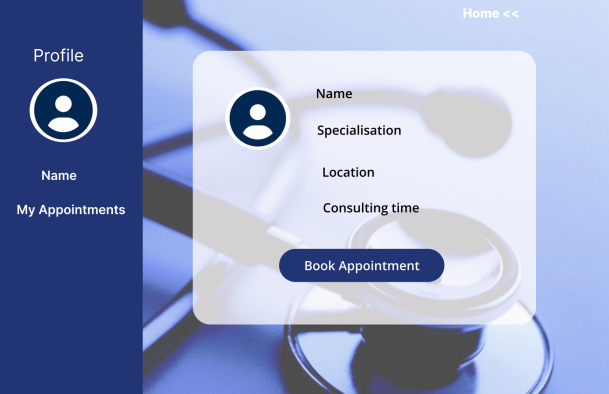
2.Application Architecture Design

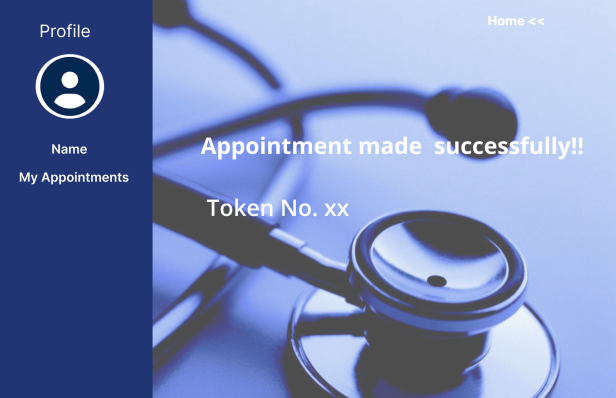
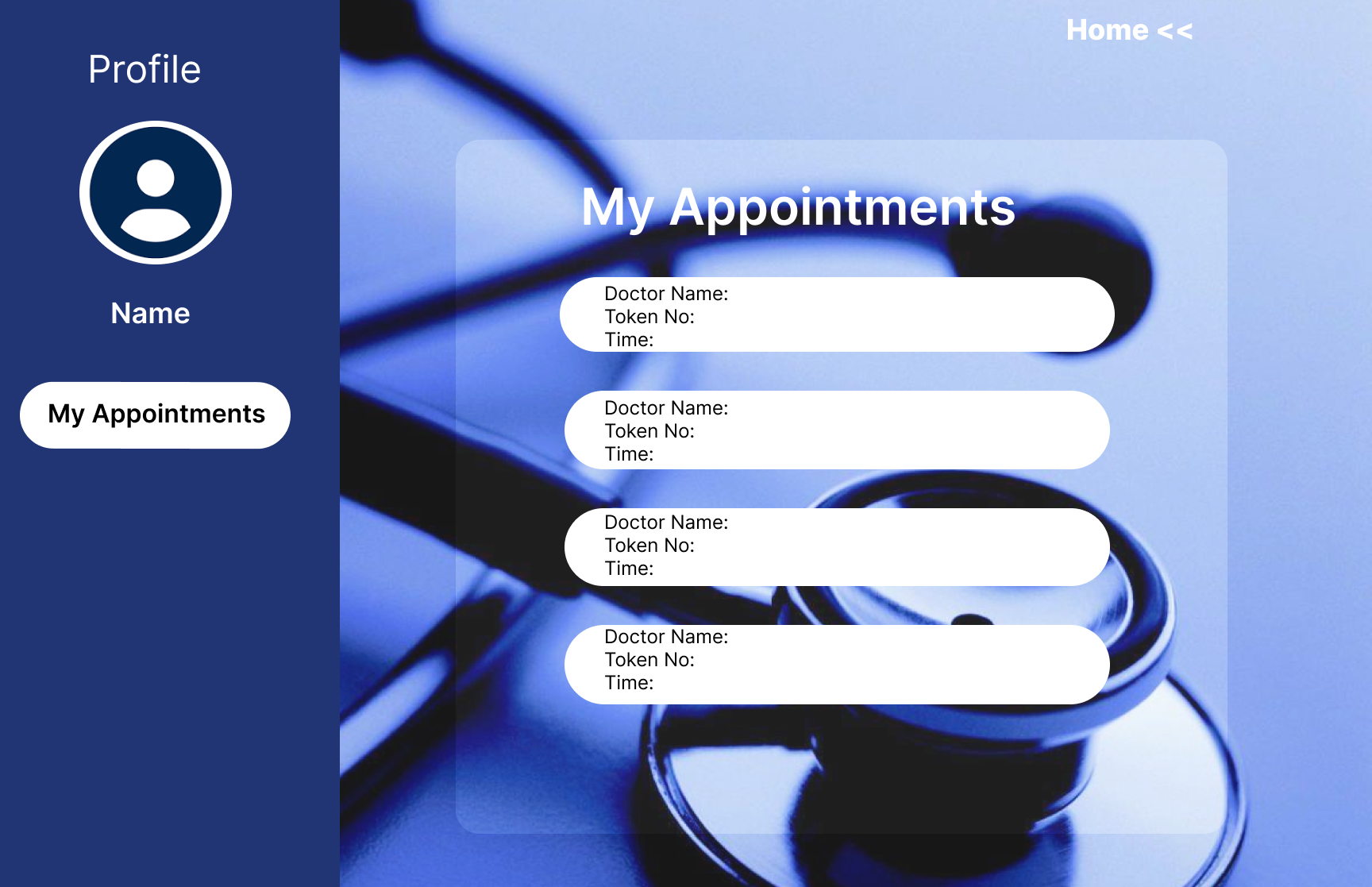


1. GUI Design

1. API Design

* **User Registration API:**

Endpoint: POST /register

Request Body: { name, email, password }

Response: HTTP 201 (Created) if successful, HTTP 400 (Bad Request) if there's an error

* **User Login API:**

Endpoint: POST /login

Request Body: { email, password }

Response:

HTTP 201 (Created) with rendered "search" view if login is successful for a user

HTTP 201 (Created) with redirection to "/doctorview/:doctorName" if login is successful for a doctor

Rendered "login" view with error message for invalid login details

* **Search Doctors API:**

Endpoint: GET /doctors

Query Parameters: { field }

Response: Rendered "doctors" view with doctors data based on the search query

* **Get Doctor Details API:**

Endpoint: GET /appointment/:id

Path Parameter: doctorId

Response: Rendered "appointment" view with doctor details based on the provided doctorId

* **Book Appointment API:**

Endpoint: POST /appointment

Request Body: { datePicker, patient\_Id, patientName, doctorName, doctorLocation, consultationTime, doctorSpecialization }

Response:

HTTP 201 (Created) with success message and token number if appointment is booked successfully

HTTP 409 (Conflict) if appointment is already booked for the same patient and doctor

* **Cancel Appointment API:**

Endpoint: POST /cancel-appointment/:appointmentId

Path Parameter: appointmentId

Response: Updated appointment list in JSON format

* **Get Patient Appointments API:**

Endpoint: GET /patientview/:userId

Path Parameter: userId

Response: Rendered "patientview" view with the patient's appointments based on the provided userId

* **Get Doctor Appointments API:**

Endpoint: GET /doctorview/:name

Path Parameter: doctorName

Response: Rendered "doctorview" view with the doctor's appointments based on the provided doctorName

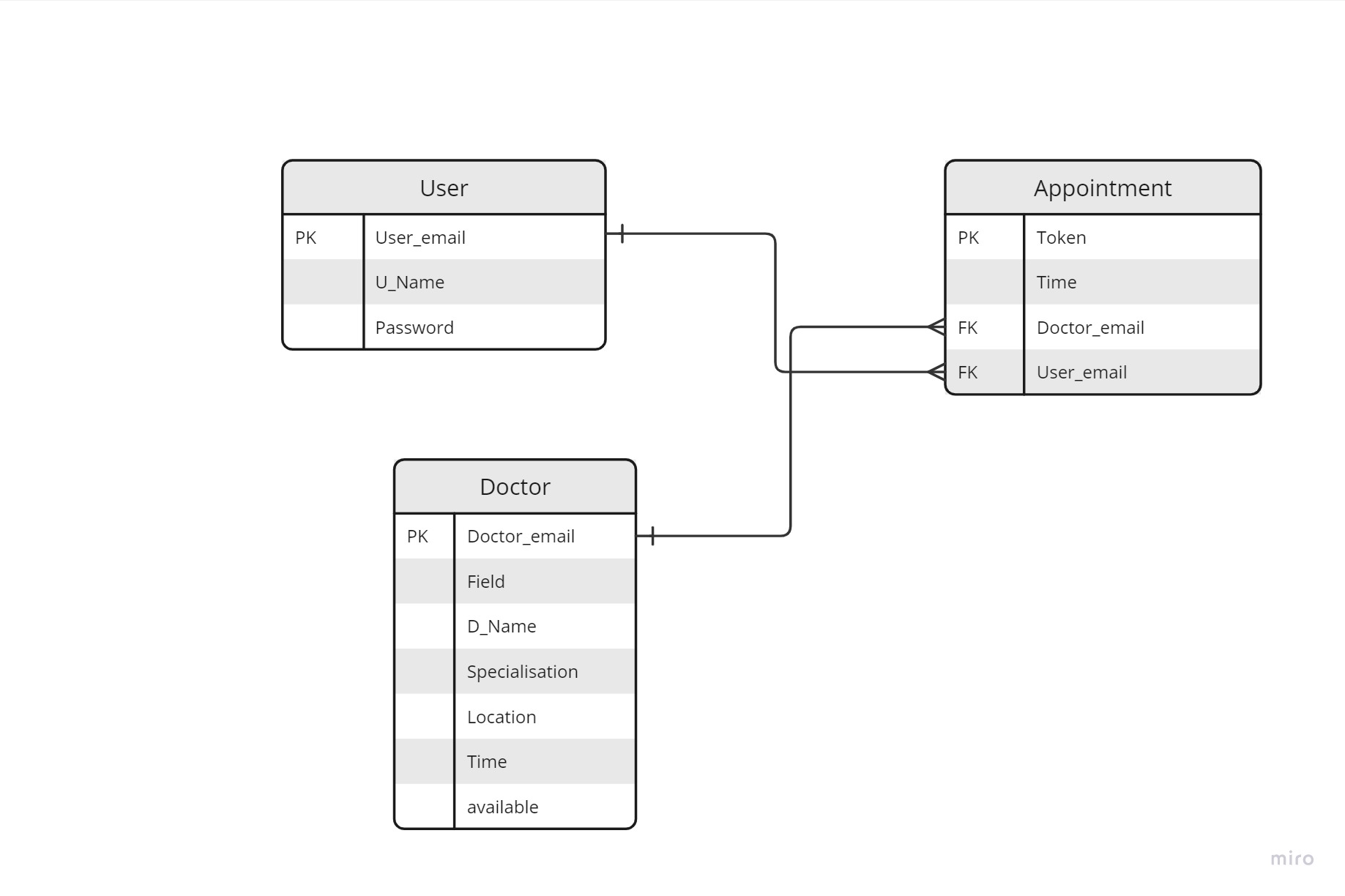
* **Update Doctor Availability API:**

Endpoint: POST /update-availability

Request Body: { doctorName, available }

Response: HTTP 200 (OK) if availability status is updated successfully

1. Data Base Design



1. Technology Stack

**Front end tech stack**

*  HTML
*  CSS
*  JavaScript

**HTML**

HTML (Hypertext Markup Language) defines the basic layout and structure of our pages.

**CSS**

All the styling that’s applied to the web pages comes from CSS

**JavaScript**

The dynamic elements of the page come from the JavaScript code.

**Back end tech stack**

The back end or server-side tech stack consists of many components, like database, server,

framework, and operating system.

* Express
* Node.js
* MongoDB

**Frameworks**

Frameworks provide a structure to the application and have common utilities so that you don’t have to write every piece of logic from scratch. We have used Express framework.

**Runtime environment**

Runtime environment is the software in which an application is run. Runtime provides crossplatform compatibility. We have used Node.js as our runtime environment.

**Databases**

Modern data platforms like MongoDB further enhance the functionality of a database by providing intelligent querying, aggregation, and transformation features, in addition to data storage and retrieval. We have used MongoDB for our database.

