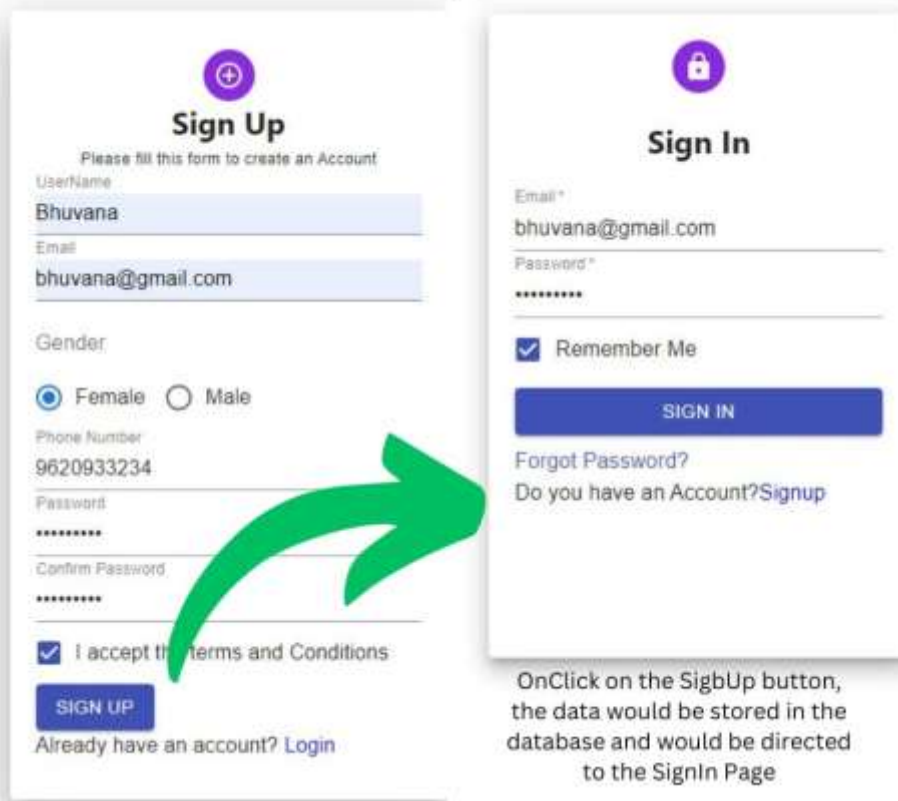




Authored by:  
Bhuvaneshwari H,  
Web Intern at Digital Shark Technology,  
Date: 14/04/2024,  
Title: Hospital Management

## SIGN UP AND THE SIGN IN PAGE



The image shows two side-by-side forms. The left form is titled 'Sign Up' with a plus icon. It contains fields for Username (Bhuvana), Email (bhuvana@gmail.com), Gender (Female selected), Phone Number (9620933234), Password, and Confirm Password. There is a checkbox for 'I accept the terms and Conditions' and a 'SIGN UP' button. Below the button is a link 'Already have an account? Login'. The right form is titled 'Sign In' with a lock icon. It contains fields for Email\* (bhuvana@gmail.com) and Password\*. There is a 'Remember Me' checkbox and a 'SIGN IN' button. Below the button are links for 'Forgot Password?' and 'Do you have an Account? Signup'. A large green arrow points from the 'SIGN UP' button on the left form to the 'SIGN IN' form on the right.

OnClick on the SigbUp button,  
the data would be stored in the  
database and would be directed  
to the SignIn Page

AND IF I TRY TO SIGN-UP WITH THE SIGNED-UP EMAIL:



The image shows the 'Sign Up' form with an error message overlay. The error message is in a red-bordered box and says 'localhost:3000 says Username or Email already exists.' with an 'OK' button. The form fields are the same as in the previous image, but the 'SIGN UP' button is replaced by a 'LOADING' button.

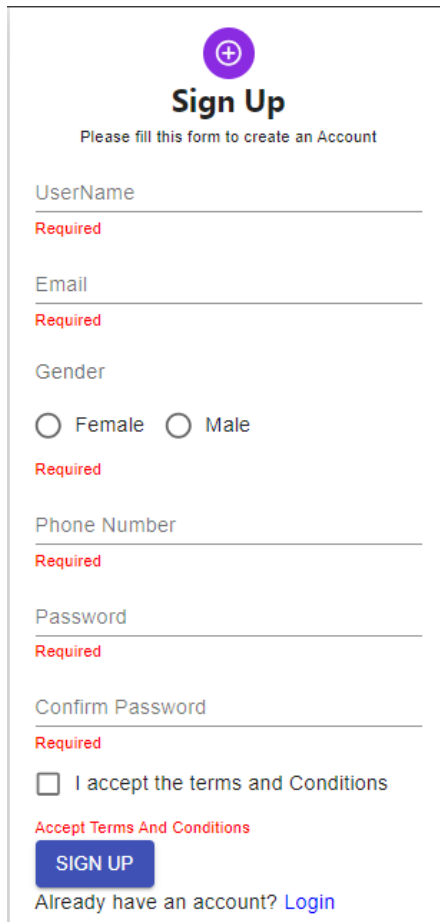
Upon succesfull Signing In, it will be directed to the Dashboard. And all the Form Validation's are done using Formik and Yup.

## FORMIK AND YUP:

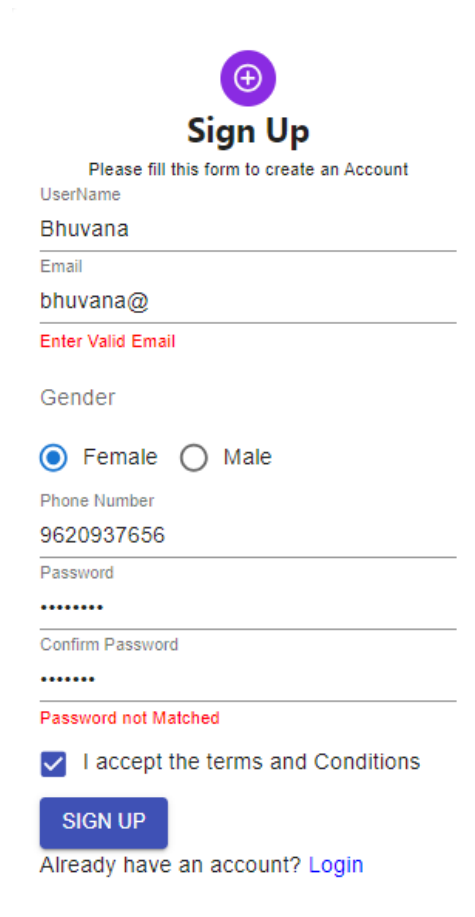
Formik is a React library for managing form state and submission, simplifying form handling by abstracting away complexities.

Yup: Yup is a JavaScript schema builder for value parsing and validation, providing a fluent API for defining validation rules and ensuring data integrity.

### FORMIK'S VALIDATION:



The image shows a 'Sign Up' form with a purple plus icon in a circle at the top. Below the icon is the text 'Sign Up' and 'Please fill this form to create an Account'. The form has five input fields: 'UserName', 'Email', 'Gender', 'Phone Number', and 'Password'. Each field has a red 'Required' error message below it. The 'Gender' field has two radio buttons, 'Female' and 'Male', both of which are unselected. Below the 'Password' field is a 'Confirm Password' field, also with a red 'Required' error message. At the bottom of the form, there is a checkbox labeled 'I accept the terms and Conditions' which is unchecked, followed by a red error message 'Accept Terms And Conditions'. A blue 'SIGN UP' button is at the bottom, and a link 'Already have an account? Login' is below it.



The image shows a 'Sign Up' form with a purple plus icon in a circle at the top. Below the icon is the text 'Sign Up' and 'Please fill this form to create an Account'. The form has five input fields: 'UserName', 'Email', 'Gender', 'Phone Number', and 'Password'. The 'UserName' field contains 'Bhuvana'. The 'Email' field contains 'bhuvana@' and has a red error message 'Enter Valid Email' below it. The 'Gender' field has two radio buttons, 'Female' and 'Male', with 'Female' selected. The 'Phone Number' field contains '9620937656'. The 'Password' field contains '.....' and the 'Confirm Password' field contains '.....'. Below the 'Confirm Password' field is a red error message 'Password not Matched'. At the bottom of the form, there is a checked checkbox labeled 'I accept the terms and Conditions'. A blue 'SIGN UP' button is at the bottom, and a link 'Already have an account? Login' is below it.

## AT THE FRONTEND:

```
const validationSchema = Yup.object().shape({
  username: Yup.string().min(3, "It's Too Short").required("Required"),
  email: Yup.string().email("Enter Valid Email").required("Required"),
  gender: Yup.string().oneOf(["male", "female"], "Required").required("Required"),
  phoneNumber: Yup.number().typeError("Enter Valid Phone
Number").required("Required"),
  password: Yup.string()
    .min(8, "Password Minimum Length should be 8")
    .matches(/^(?=.*[!@#$$%^&*])/, "Password must contain at least one special
character")
    .required("Required"),
  confirmpassword: Yup.string().oneOf([Yup.ref('password')], "Password not
Matched").required("Required"),
  termsAndConditions: Yup.boolean().oneOf([true], "Accept Terms And Conditions") //
Updated for boolean type
})
```

# CRUD OPERATIONS (FOR DOCTOR'S PAGE)

## CREATE OPERATION

ADD DOCTOR

Name Akshay	Department Surgeon
Experience 15	Gender Male
City Mysore	Country India
Specialization Orthopedic	Patients Served 3000

☒ Doctor Available

ADD +

As soon as you enter the Doctor's data in the form, the data gets stored in the database and gets updated in the Table

```
{
  "_id": ObjectId('6616b82a6b60213ae2c67775'),
  "name": "Akshay",
  "department": "Surgeon",
  "gender": "Male",
  "experience": "15",
  "city": "Mysore",
  "country": "India",
  "specialization": "Orthopedic",
  "patientsServed": "3000",
  "doctoravailable": true,
  "createdAt": 1712764970046,
  "updatedAt": 1712764970046
}
```

Unity LifeLine

Inserted Successfully

Dashboard  
Doctors  
Patients

DOCTORS

Experienced Doctors + Add Doctor

Nandini	Surgeon	12	Female	Banglore	India	
Aishwarya	Nurse	1	Female	Banglore	India	
Vinay	Surgeon	15	Male	Mysore	India	
Akshay	Surgeon	15	Male	Mysore	India	

## AT THE BACKEND:

## MODEL DEFINITION:

```
module.exports = {
  attributes: {
    name: { type: 'string', required: true },
    department: { type: 'string', required: true },
    experience: { type: 'string', required: true },
    gender: { type: 'string', required: true },
    city: { type: 'string', required: true },
    country: { type: 'string', required: true },
    specialization: { type: 'string', required: true },
    patientsServed: { type: 'string', required: true },
    doctoravailable: { type: 'boolean', defaultsTo: false },
  }
};
```

## CONFIGURE ROUTE:

```
'POST /create': 'DoctorController.create',
```

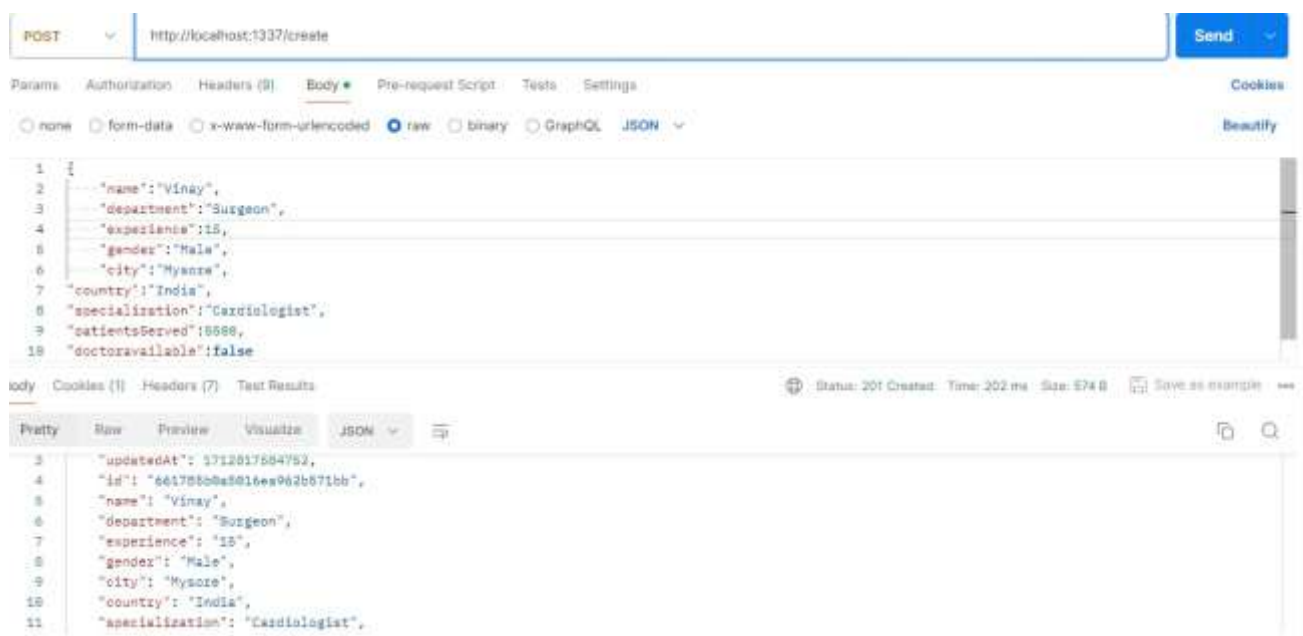
## BACKEND CONTROLLER'S LOGIC:

```
async create(req, res) {
  try {
    const doctor = await Doctor.create(req.body).fetch();
    return res.status(201).json(doctor);
  } catch (error) {
    return res.status(500).json({ error: 'Server Error' });
  }
},
```

## FRONTEND LOGIC:

```
const handleAddSubmit = async (data) => {
  console.log('Adding new doctor:', data);
  try {
    const response = await axios.post('http://localhost:1337/create', data);
    console.log('New doctor added:', response.data);
    setDoctors(prevDoctors => [...prevDoctors, response.data]);
    setIsPopoverOpen(true);
    setTimeout(() => setIsPopoverOpen(false), 2000);
    toggleDrawer();
  } catch (error) {
    console.error('Error adding doctor:', error);
  }
};
```

## TESTING THROUGH POSTMAN



## FORMIK'S VALIDATION WITH YUP

The image displays two versions of a web form titled "ADD DOCTOR".

**Left Screenshot (Empty Form):** The form contains eight input fields arranged in two columns. Each field has a red error message below it: "Name is required", "Department is required", "Experience is required", "Gender is required", "City is required", "Country is required", "Specialization is required", and "Patients Served is required". At the bottom left is a checkbox labeled "Doctor Available", and at the bottom right is a blue button labeled "ADD +".

**Right Screenshot (Filled Form):** The form is filled with the following data: Name "Ramesh", Department "Heart", Experience "2yrs", Gender "Female", City "Mysore", Country "India", Specialization "Cardiologist", and Patients Served "Thousand". The error message for "Experience" is "Experience must be a number". The "Doctor Available" checkbox is unchecked. The "ADD +" button is at the bottom right.

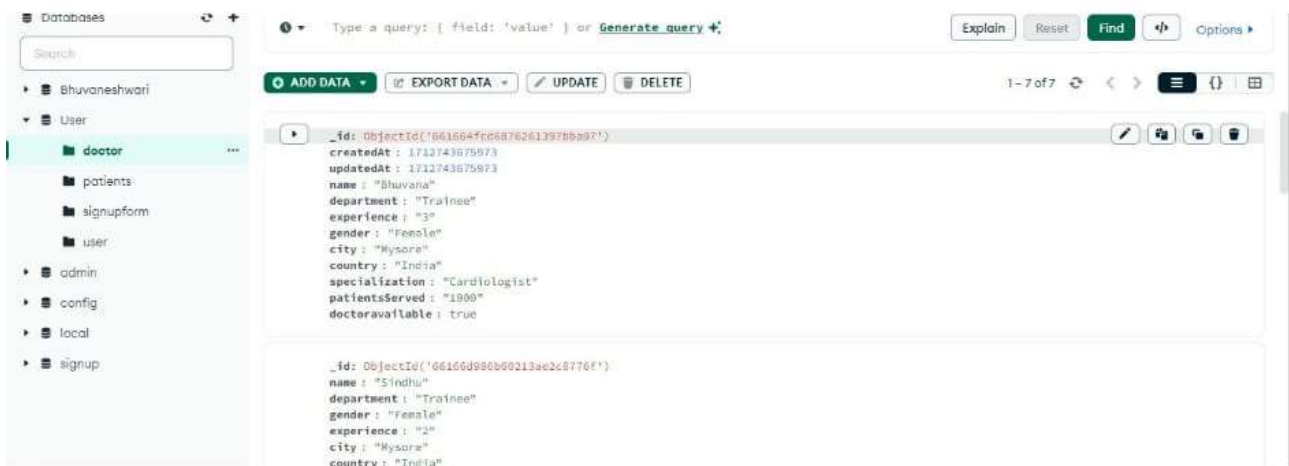
```
const validationSchema = Yup.object().shape({
  name: Yup.string().required('Name is required'),
  department: Yup.string()
    .required('Department is required')
  experience: Yup.number().typeError('Experience must be a number').required('Experience is required'),
  city: Yup.string().required('City is required'),
  country: Yup.string().required('Country is required'),
  specialization: Yup.string().required('Specialization is required'),
  patientsServed: Yup.number().typeError('Patients Served must be a number').required('Patients Served is required'),
});
```

## READ OPERATION



Profile	Name	Department	Experience	Gender	City	Country	Actions
	Bhuvana	Trainee	3	Female	Mysore	India	
	Sindhu	Trainee	2	Female	Mysore	India	
	Nandini	Surgeon	12	Female	Bangalore	India	

The data will be fetched from the database and will be displayed in the table through GET Method



### AT THE BACKEND:

### CONFIGURE ROUTE:

```
'GET /doctors': 'DoctorController.doctors',
```

### BACKEND CONTROLLER'S LOGIC:

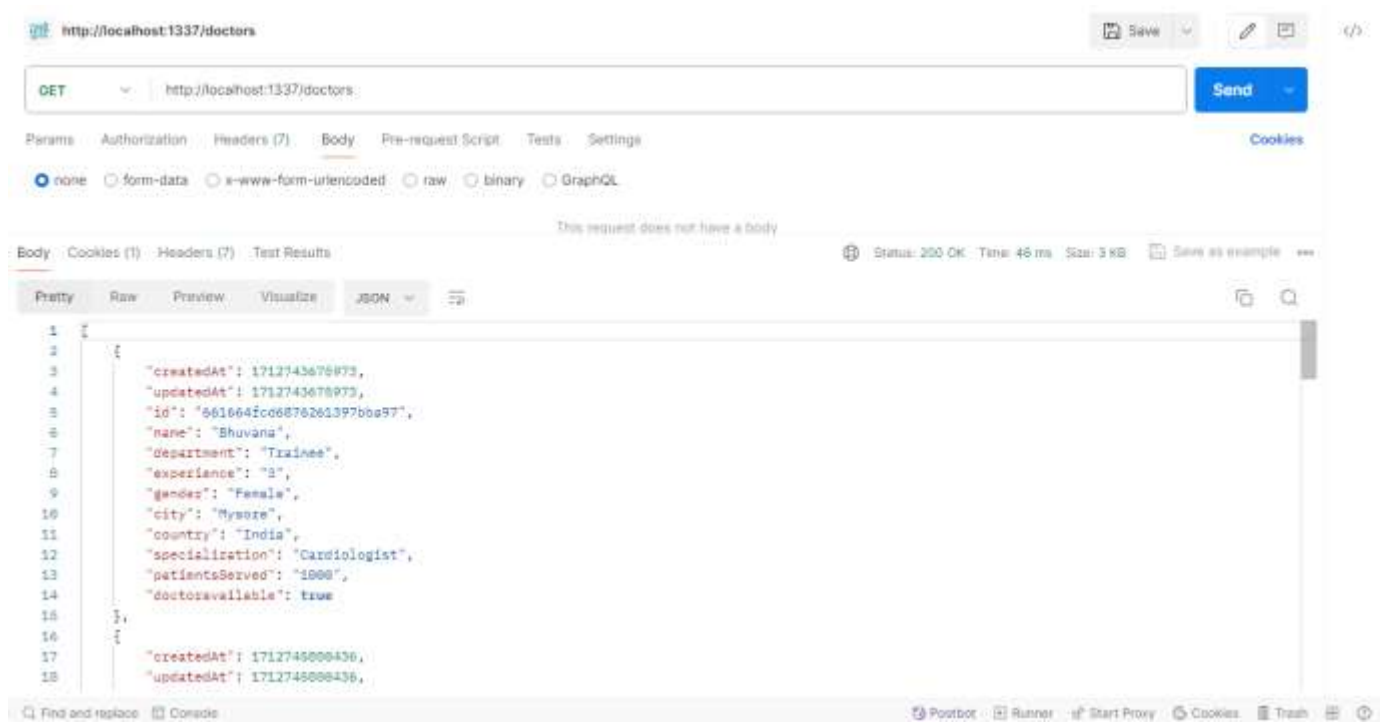
```
async doctors(req, res) {
  try {
    // Fetch doctors data from the database (e.g., using a model)
    const doctors = await Doctor.find(); // Assuming Doctor is your model

    // Send the fetched data as a response
    return res.json(doctors);
  } catch (error) {
    // Handle errors
    console.error('Error fetching doctors:', error);
    return res.status(500).json({ error: 'Server Error' });
  }
},
```

## FRONTEND LOGIC:

```
const fetchData = async () => {
  try {
    // Make GET request to your backend API endpoint to fetch doctors data
    const response = await axios.get('http://localhost:1337/doctors');
    setDoctors(response.data); // Set the fetched data to the state
  } catch (error) {
    console.error('Error fetching data:', error);
  }
};
```

## TESTING THROUGH POSTMAN:











## UPDATE OPERATION

Unity LifeLine

Dashboard  
Doctors  
Patients

DOCTORS

Experienced Doctors Add Doctor

Profile	Name	Department	Experience	Gender	City	Country	Actions
	Ananya	Nurse	3	Female	Banglore	India	 
	Apoorva	Nurse	1	Male	Banglore	India	 

Updating Name Ananya to Ananya S A

Unity LifeLine

Dashboard  
Doctors  
Patients

UPDATE DOCTOR

Name: Ananya S.A

Department: Nurse

Experience: 3

Gender: Female

City: Banglore

Country: India

Specialization: Orthopedic

Patients Served: 1000

☐ Doctor Available

Unity LifeLine

Updated Successfully

Dashboard  
Doctors  
Patients

DOCTORS

Experienced Doctors Add Doctor

Profile	Name	Department	Experience	Gender	City	Country	Actions
	Ananya.S.A	Nurse	3	Female	Banglore	India	 
	Apoorva	Nurse	1	Male	Banglore	India	 
	Abhishek	Surgeon	15	Male	Mysore	India	 
	Apeksha	Surgeon	11	Female	Mysore	India	 

Before



After

```
{
  "_id": ObjectId('6618f9c75714e35cebd742c7'),
  "createdAt": 1712912839081,
  "updatedAt": 1712912839081,
  "name": "Ananya",
  "department": "Nurse",
  "experience": "3",
  "gender": "Female",
  "city": "Banglore",
  "country": "India",
  "specialization": "Orthopedic",
  "patientsServed": "1000",
  "doctoravailable": true
}
```

```
{
  "_id": ObjectId('6618f9c75714e35cebd742c7'),
  "createdAt": 1712912839081,
  "updatedAt": 1712912839081,
  "name": "Ananya.S.A",
  "department": "Nurse",
  "experience": "3",
  "gender": "Female",
  "city": "Banglore",
  "country": "India",
  "specialization": "Orthopedic",
  "patientsServed": "1000",
  "doctoravailable": true
}
```

## AT THE BACKEND:

### CONFIGURE ROUTE:

```
'PUT /update/:id': 'DoctorController.update',
```

### CONTROLLER'S LOGIC:

```
async update(req, res) {
  const doctorId = req.param('id');
  const newData = req.allParams();
  try {
    const updatedDoctor = await Doctor.updateOne({ id: doctorId }).set(newData);
    if (!updatedDoctor) {
      return res.notFound('Doctor not found.');
```

## AT THE FRONTEND:

```
const handleUpdateSubmit = async (data) => {
  try {
    console.log('Editing doctor:', editingDoctor);
    console.log('Updated data:', data);

    // If editingDoctor exists, it means we're updating an existing doctor
    const updatedDoctor = { ...editingDoctor, ...data };

    // Make PUT request to backend API to update the doctor
    await axios.put(`http://localhost:1337/update/${editingDoctor.id}`, updatedDoctor);
```

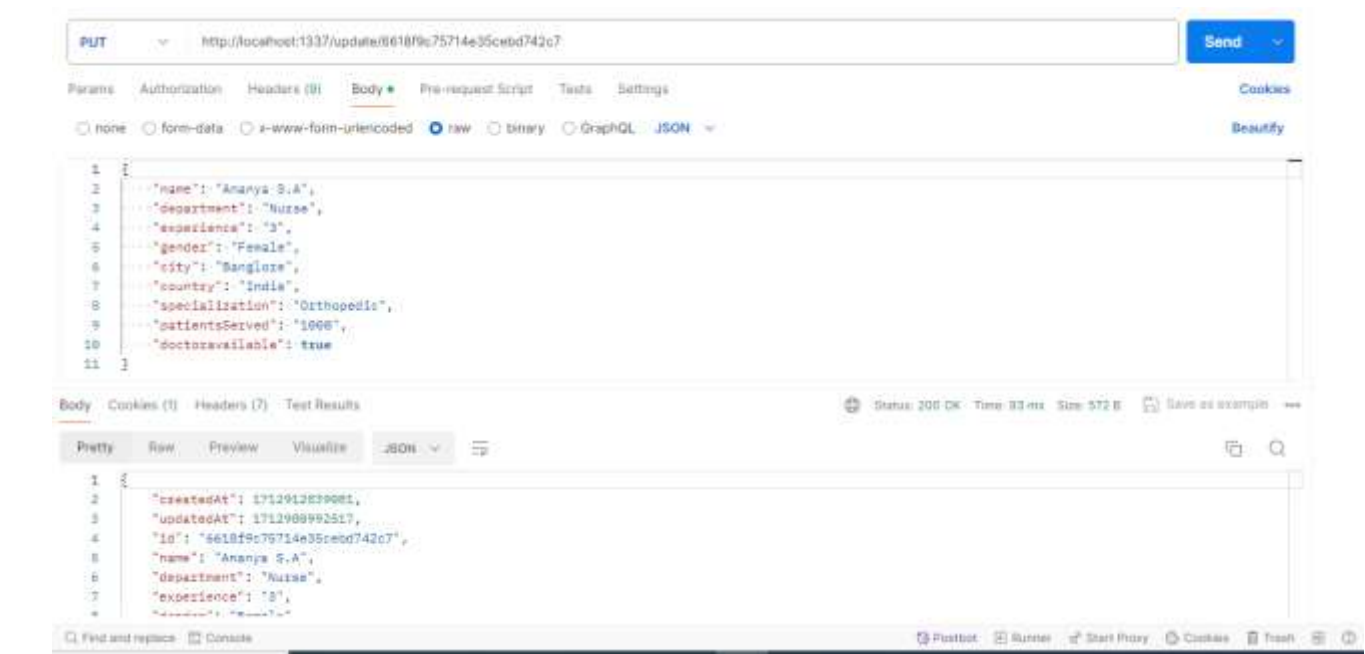
```

// Update the doctor object in the state with the new data
setDoctors(prevDoctors =>
  prevDoctors.map(doctor =>
    doctor.id === editingDoctor.id ? updatedDoctor : doctor
  )
);
setIsUpdateSuccessOpen(true);

toggleDrawer(); // Close the drawer after submission
} catch (error) {
  console.error('Error updating doctor:', error);
  // Handle error gracefully (e.g., display error message)
}
};

```

## TESTING THROUGH POSTMAN:



## DELETE OPERATION



OnClick on the Delete icon, a prompt appears stating "Are you sure you want to delete this doctor", If yes, the doctor gets deleted both at the frontend and backend

---

### AT THE BACKEND:

#### CONFIGURE ROUTE:

```
'DELETE /deleteDoctor/:id': 'DoctorController.deleteDoctor',
```

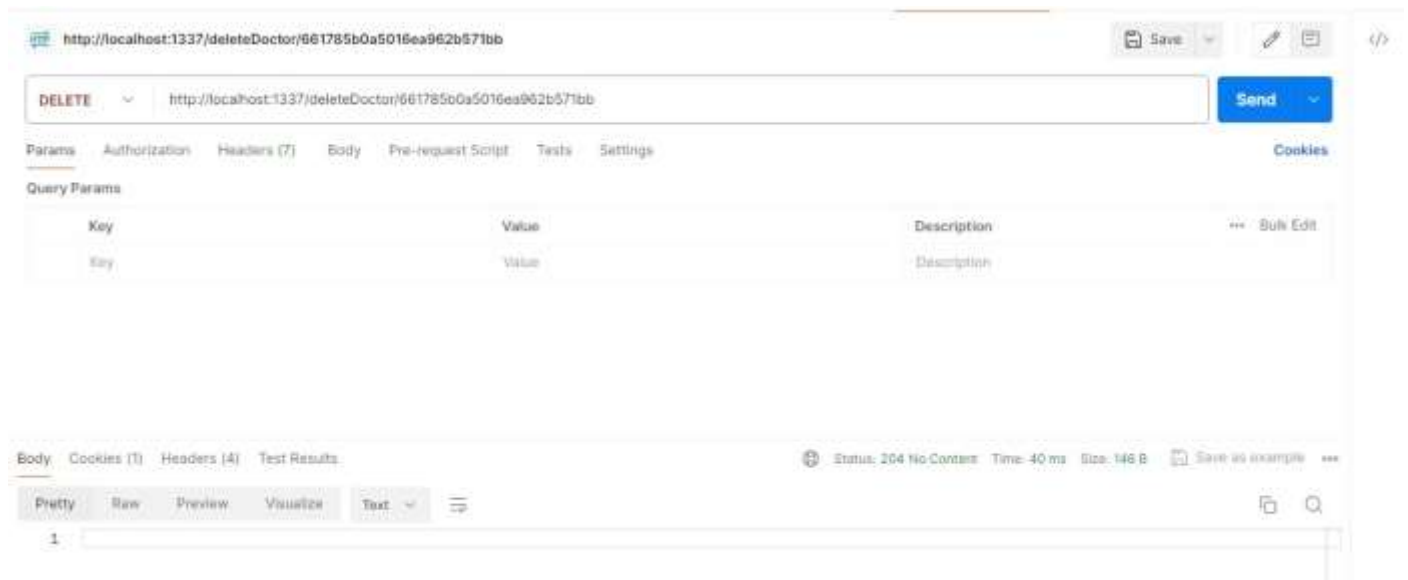
#### BACKEND CONTROLLER'S LOGIC:

```
async deleteDoctor(req, res) {
  const { id } = req.params;
  try {
    // Find the doctor by ID and delete it
    const deletedDoctor = await Doctor.destroyOne({ id });
    if (!deletedDoctor) {
      return res.status(404).json({ error: 'Doctor not found' });
    }
    res.status(204).end(); // Respond with 204 No Content on successful deletion
  } catch (error) {
    console.error('Error deleting doctor:', error);
    res.status(500).json({ error: 'Server Error' });
  }
}
```

## FRONTEND LOGIC:

```
const handleDeleteClick = async (id) => {  
  console.log('Deleting doctor with ID:', id); // Add this line  
  try {  
  
    // Make DELETE request to backend API to delete the doctor  
    await axios.delete(`http://localhost:1337/deleteDoctor/${id}`);  
    // Update the doctors state by filtering out the deleted doctor  
    setDoctors(doctors.filter((doctor) => doctor.id !== id));  
  } catch (error) {  
    console.error('Error deleting doctor:', error);  
  }  
};
```

## TESTING THROUGH POSTMAN:



## CHECKING DOCTOR'S AVAILABILITY

ADD DOCTOR

Name Vinay	Department Surgeon
Experience 15	Gender Male
City Mysore	Country India
Specialization Cardiologist	Patients Served 5000

☒ Doctor Available

ADD +

Dr. Vinay is available for his patients, and onMouseover of his profile, it shows that Vinay is Available

```
_id: ObjectId('661785b0a5016ea962b571bb')
createdAt: 1712817584752
updatedAt: 1712817584752
name: "Vinay"
department: "Surgeon"
experience: "15"
gender: "Male"
city: "Mysore"
country: "India"
specialization: "Cardiologist"
patientsServed: "5500"
doctoravailable: true
```

Unity LifeLine

Dashboard  
Doctors  
Patients

DOCTORS

Experienced Doctors Add Doctor

Profile	Name	Department	Experience	Gender	City	Country	Actions
 Vinay is available	Vinay	Surgeon	15	Male	Mysore	India	 
	Apeksha	Surgeon	11	Female	Mysore	India	 
	Ananya	Nurse	3	Female	Banglore	India	 
	Apoorva	Nurse	1	Male	Banglore	India	 

UPDATE DOCTOR

Name Apoorva	Department Nurse
Experience 1	Gender Male
City Banglore	Country India
Specialization Orthopedic	Patients Served 100

☐ Doctor Available

UPDATE

Apoorva not available for her patients, and onMouseover of his profile, it shows that she is not Available

```
_id: ObjectId('6618f9fd5714e35cebd742c8')
createdAt: 1712912893961
updatedAt: 1712912893961
name: "Apoorva"
department: "Nurse"
experience: "1"
gender: "Male"
city: "Banglore"
country: "India"
specialization: "Orthopedic"
patientsServed: "100"
doctoravailable: false
```

Unity LifeLine

Dashboard  
Doctors  
Patients

DOCTORS

Experienced Doctors Add Doctor

	Ananya	Nurse	3	Female	Banglore	India	 
 Ananya is not available	Apoorva	Nurse	1	Male	Banglore	India	 
	Shiva	Trainee	1	Male	Banglore	India	 
	Abhishek	Surgeon	15	Male	Mysore	India	 



## AT THE BACKEND:

### CONFIGURE ROUTE:

```
'GET  
/doctors/fetchDoctorAvailability/:doctorId':'DoctorController.fetchDoctorAvailability',
```

### CONTROLLER'S LOGIC:

```
async fetchDoctorAvailability(req, res) {  
  try {  
    const doctorId = req.params.doctorId; // Extract doctor ID from request params  
    const doctor = await Doctor.findOne({ id: doctorId }); // Find doctor by ID    if (!doctor) {  
      return res.notFound('Doctor not found');  
    }    const availabilityStatus = doctor.doctoravailable ? 'available' : 'not available';  
    return res.json({  
      doctor: doctor.name, // Assuming 'name' attribute represents the doctor's name  
      doctoravailable: availabilityStatus  
    });  
  } catch (error) {  
    console.error('Error fetching doctor availability:', error);  
    return res.serverError('Internal server error');  
  }  
},
```

### FRONTEND LOGIC:

```
const fetchDoctorAvailability = async (doctorId) => {  
  try {  
    const response = await  
    axios.get(`http://localhost:1337/doctors/fetchDoctorAvailability/${doctorId}`);  
    const doctorData = response.data;  
    console.log(doctorData);  
    let status;  
    if (doctorData.doctoravailable==='available') {  
      status = 'available';  
    } else {  
      status = 'not available';  
    }  
    console.log(status);  
    setAvailability(`${doctorData.doctor} is ${status}`);  
  } catch (error) {  
    console.error('Error fetching doctor availability:', error);  
  }  
}
```

```
    setAvailability('Error fetching availability');  
  }  
};
```

## TESTING THROUGH POSTMAN:

The screenshot shows a Postman interface for a GET request. The URL is `http://localhost:3337/doctors/fetchDoctorAvailability/8618f9c75714e35cebd742c7`. The response status is 200 OK, with a time of 24 ms and a size of 303 B. The response body is displayed in JSON format:

```
{  
  "doctor": "Ananya",  
  "doctorAvailability": "available"  
}
```



## VIEW DOCTOR DETAILS



OnClick on the particular doctor 's profile, that particular doctor 's profile would be showed which includes his/her Name, Specialization In, Number of Patients Served And his/her Experience



### FRONTEND LOGIC:

```
const handleProfileClick = async (id) => {
  try {
    const response = await axios.get(`http://localhost:1337/doctors/${id}`);
    const doctorData = response.data;
    handleDialogOpen(doctorData);
  } catch (error) {
    console.error('Error fetching doctor details:', error);
  }
};
```

## EXPERIENCED DOCTORS



Unity LifeLine

Dashboard  
Doctors  
Patients

DOCTORS

Experienced Doctors Add Doctor

Profile	Name	Department	Experience	Gender	City	Country	Actions
	Vinay	Surgeon	15	Male	Mysore	India	
	Ananya	Nurse	2	Female	Bangalore	India	
	Apoorva	Nurse	1	Male	Bangalore	India	
	Shiva	Trainee	1	Male	Bangalore	India	

OnClick on the Experienced Doctor's Button, only those doctors who have experience of more than 10 years would be displayed



Unity LifeLine

Dashboard  
Doctors  
Patients

DOCTORS

Experienced Doctors Add Doctor

Profile	Name	Department	Experience	Gender	City	Country	Actions
	Vinay	Surgeon	15	Male	Mysore	India	
	Abhishek	Surgeon	15	Male	Mysore	India	
	Apeksha	Surgeon	11	Female	Mysore	India	

### AT THE BACKEND:

### CONFIGURE ROUTE:

```
'GET /doctors/fetchExperiencedDoctors': 'DoctorController.fetchExperiencedDoctors',
```

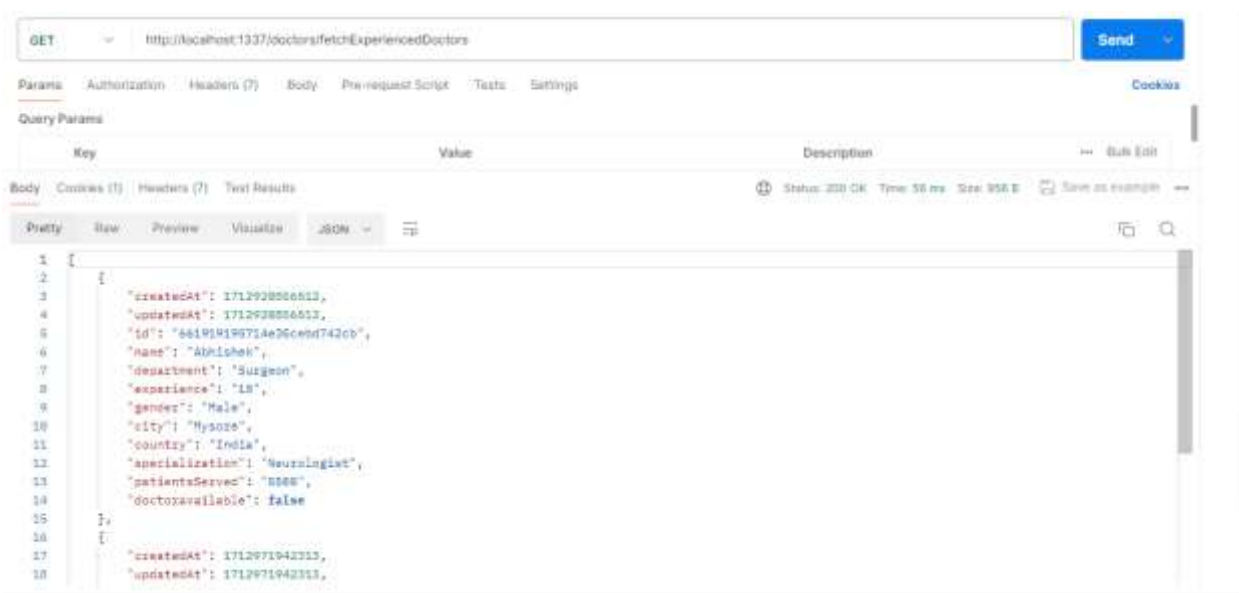
### CONTROLLER'S LOGIC:

```
async fetchExperiencedDoctors(req, res) {  
  try {  
    const doctors = await Doctor.find();  
    const experiencedDoctors = doctors.filter(doctor => doctor.experience >= 10);  
  
    return res.json(experiencedDoctors);  
  } catch (error) {  
    console.error('Error fetching experienced doctors:', error);  
    return res.serverError('Internal server error'); // Handle error gracefully  
  }  
},
```

## AT THE FRONTEND:

```
const fetchExperiencedDoctors = async () => {
  try {
    // Make a GET request to your backend API endpoint to fetch experienced doctors
    const response = await
axios.get('http://localhost:1337/doctors/fetchExperiencedDoctors');
    setDoctors(response.data); // Update the doctors state with the fetched experienced
doctors
  } catch (error) {
    console.error('Error fetching experienced doctors:', error);
  }
};
```

## TESTING THROUGH POSTMAN:



# CRUD OPERATIONS (FOR PATIENT'S PAGE)

## CREATE OPERATION

**ADD PATIENT**

Name	Disease
Sandhya	Malaria
Family Doctor	Gender
Shanthi	Female
City	Country
Bangalore	India
Description	
Sweating, Pain in abdomen and muscles, Chills, Fatigue	
<input checked="" type="checkbox"/> Severe	

**ADD +**

As soon as you enter the Patients data in the form, the data gets stored in the database and gets updated in the Table

```
_id: ObjectId('561a111e94dd66377ddc9a5c')
name: "Sandhya"
disease: "Malaria"
gender: "Female"
familydoctor: "Shanthi"
city: "Bangalore"
country: "India"
description: "Sweating, Pain in abdomen and muscles, Chills, Fatigue"
severe: true
createdAt: 1712984356588
updatedAt: 1712984356588
```

Unity LifeLine

Inserted Successfully

Dashboard

Doctors

Patients

PATIENTS

Patients Name >

Search by patientName

Submit

Severe patients

Add Patient

	Pallavi	Malaria	Pruthvi	Female	Mysore	India		
	Anusha	Fever	Bhuvana	Female	Mysore	India		
	Ananya	Cold	Sindhu	Female	Banglore	India		
	Pranav	Fever	Bhuvana	Male	Banglore	India		
	Sandhya	Malaria	Shanthi	Female	Banglore	India		

## READ OPERATION

**Unity LifeLine**

**PATIENTS**

Profile	Name	Disease	Family Doctor	Gender	City	Country	Actions
	Abhi	Dengue	Sindhu	Male	Mysore	India	
	Pallavi	Malaria	Pruthvi	Female	Mysore	India	

**localhost:27017**

**My Queries** **patients**

**Documents** **Aggregations** **Schemas** **Indexes** **Validation**

Type a query (field: "value") or **Generate query**

**EXPLORE** **EXPORT DATA** **UPDATE** **DELETE**

```
{
  "_id": ObjectId("561a111e94dd66377ddc9a5c"),
  "name": "Sandhya",
  "disease": "Malaria",
  "gender": "Female",
  "familydoctor": "Shanthi",
  "city": "Bangalore",
  "country": "India",
  "description": "Sweating, Pain in abdomen and muscles, Chills, Fatigue",
  "severe": true,
  "createdAt": 1712984356588,
  "updatedAt": 1712984356588
}
```

## UPDATE OPERATION

Unity LifeLine

PATIENTS

Patients Name ▾ Search by patientName

SUBMIT Severe patients Add Patient

Profile	Name	Disease	Family Doctor	Gender	City	Country	Actions
	Abhi	Dengue	Sindhu	Male	Mysore	India	
	Pallavi	Malaria	Pruthvi	Female	Mysore	India	

Updating Family Doctor of Patient Abhi from Sindhu to Pallavi

Unity LifeLine

UPDATE PATIENT

Name: Abhi Disease: Dengue

Family Doctor: Pallavi Gender: Male

City: Mysore Country: India

Description: Platelets Count is reduced

☒ Severe

UPDATE

Unity LifeLine

Updated Successfully

PATIENTS

Patients Name ▾ Search by patientName

SUBMIT Severe patients Add Patient

Profile	Name	Disease	Family Doctor	Gender	City	Country	Actions
	Abhi	Dengue	Pallavi	Male	Mysore	India	
	Pallavi	Malaria	Pruthvi	Female	Mysore	India	
	Anusha	Fever	Bhuvana	Female	Mysore	India	
	Ananya	Cold	Sindhu	Female	Banglore	India	

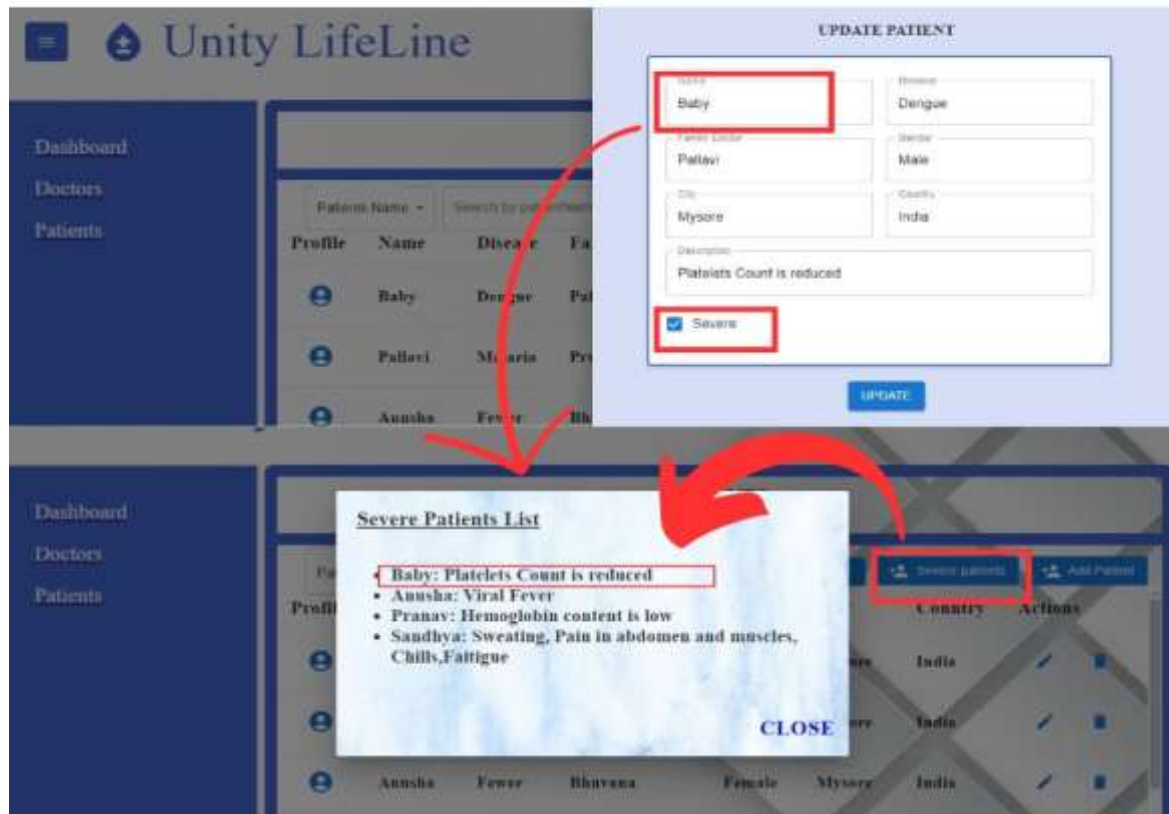


## DELETE OPERATION



The backend and frontend logic remains consistent between the doctor's page and the patient's page. The CRUD operations logic remains unchanged.

## SEVERE PATIENTS LIST



### AT THE BACKEND:

### CONFIGURE ROUTE:

```
'GET /patients/getSeverePatients' : 'PatientsController.getSeverePatients',
```

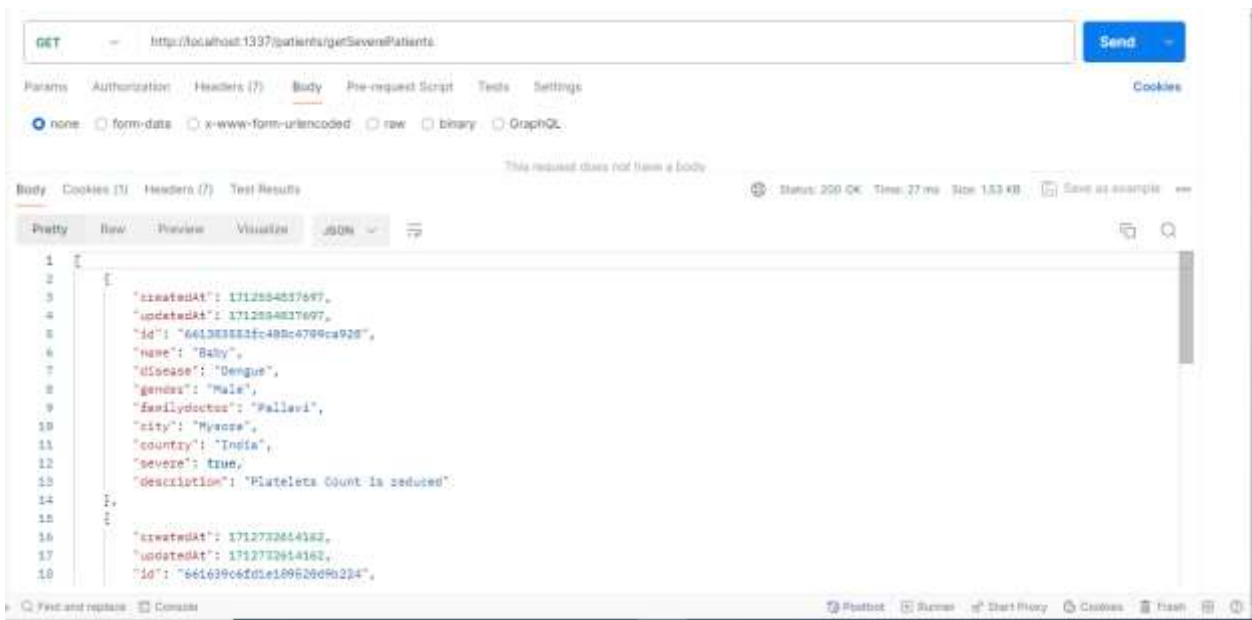
### CONTROLLER'S LOGIC:

```
async getSeverePatients(req, res) {
  try {
    const severePatients = await Patients.find({ severe: true }); // Assuming
    isSevere is a field indicating severity
    return res.json(severePatients);
  } catch (error) {
    return res.serverError(error);
  }
},
```

## FRONTEND LOGIC:

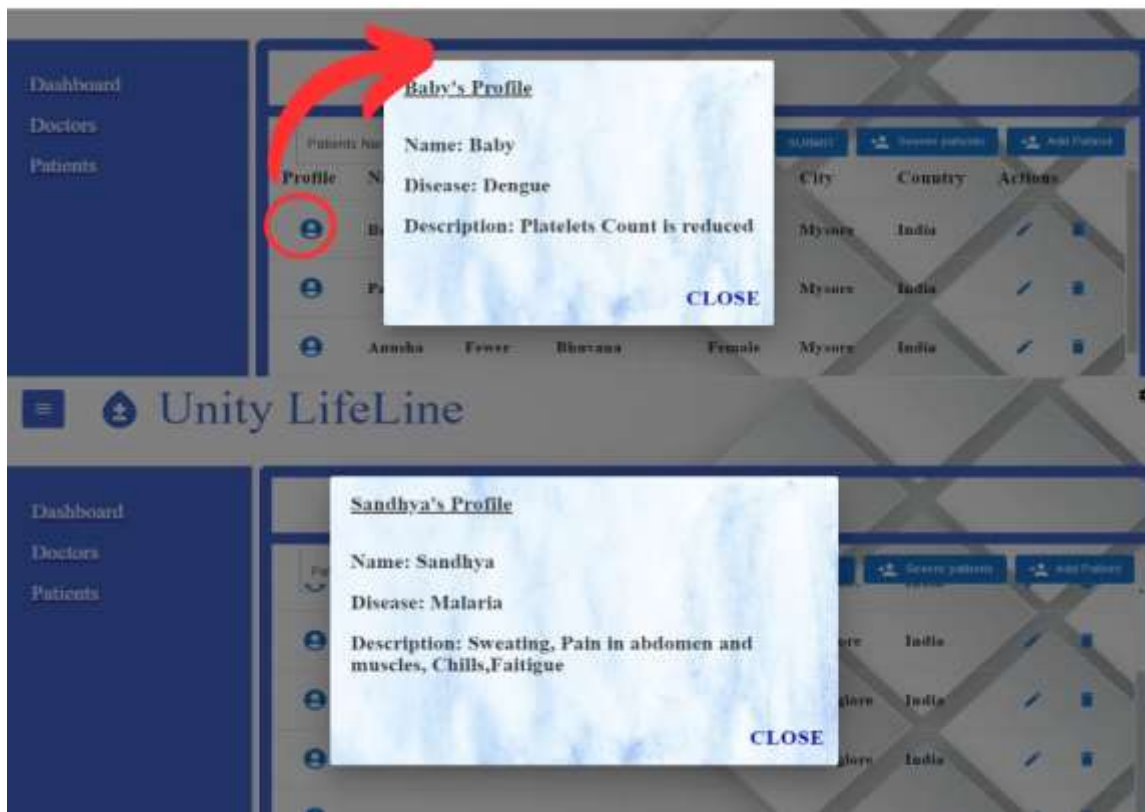
```
const handleSeverePatientsClick = async () => {
  try {
    // Make GET request to your backend API endpoint to fetch severe patients
    const response = await
axios.get('http://localhost:1337/patients/getSeverePatients');
    setShowSeverePopup(true); // Show the severe condition popup with severe patients
data
    setSeverePatients(response.data); // Set severe patients data
  } catch (error) {
    console.error('Error fetching severe patients:', error);
  }
};
```

## TESTING THROUGH POSTMAN:





## VIEW PATIENT'S DETAILS



On Click on Particular Patient's Profile, the details of the Patient will be displayed.

### AT THE FRONTEND:

```
const handleProfileClick = async (patient) => {
  setSelectedPatient(patient);
};
```

```
Dialog open={!selectedPatient} onClose={handleCloseProfileDialog}>
  <DialogContent>
    {selectedPatient && (
      <>
        <p>`Name: ${selectedPatient.name}`</p>
        <p>`Disease: ${selectedPatient.disease}`</p>
        <p>`Description: ${selectedPatient.description}`</p>
      </>
    )}
  </DialogContent>
</Dialog>
```

## FILTERING DETAILS

The top screenshot shows the 'PATIENTS' table with the 'Family Doctor' filter set to 'Bhuvana'. The table lists three patients: Anusha, Pranav, and Bhuvana, all with the disease 'Fewer' or 'Malaria'.

Profile	Name	Disease	Family Doctor	Gender	City	Country	Actions
	Anusha	Fewer	Bhuvana	Female	Mysore	India	
	Pranav	Fewer	Bhuvana	Male	Banglore	India	
	Bhuvana	Malaria	Bhuvana	Female	Mysore	India	

The bottom screenshot shows the 'PATIENTS' table with the 'Disease' filter set to 'Malaria'. The table lists two patients: Pallavi and Sandhya, both with the disease 'Malaria'.

Profile	Name	Disease	Family Doctor	Gender	City	Country	Actions
	Pallavi	Malaria	Pruthvi	Female	Mysore	India	
	Sandhya	Malaria	Shanthi	Female	Banglore	India	

The Patient's data would be filtered out based on our Requirements. If you want list all the Patients who would be treated by the Family Doctor Bhuvana that would be listed.

The same filtering is possible with respect to the disease and the name of the Patient's as well.

### AT THE BACKEND:

### CONFIGURE ROUTES:

```
'GET /patients/getByFilter': 'PatientsController.getByFilter'
```

### CONTROLLER'S LOGIC:

```
async getByFilter(req, res) {
  try {
    const filterType = req.query.filterType; // Extract the filter type from query
    parameters
    let filterValue = req.query.filterValue; // Extract the filter value from query
    parameters
  }
}
```

```

    // Determine the field based on the filter type
    let filterField;
    switch (filterType) {
      case 'doctorName':
        filterField = 'familydoctor';
        break;
      case 'patientName':
        filterField = 'name';
        break;
      case 'disease':
        filterField = 'disease';
        break;
      default:
        return res.badRequest('Invalid filter type');
    }

    // Query the database to find patients based on the specified filter
    const patients = await Patients.find({ [filterField]: filterValue });

    // Return the list of filtered patients
    return res.ok(patients);
  } catch (error) {
    return res.serverError(error);
  }
},

```

## AT THE FRONTEND:

```

const handleSubmitFilter = async () => {
  try {
    const endpoint = 'getByFilter';
    let queryParams = '';

    // Check if filterValue is not empty before adding it to queryParams
    if (filterValue) {
      queryParams = `filterType=${filterType}&filterValue=${filterValue}`;
    }

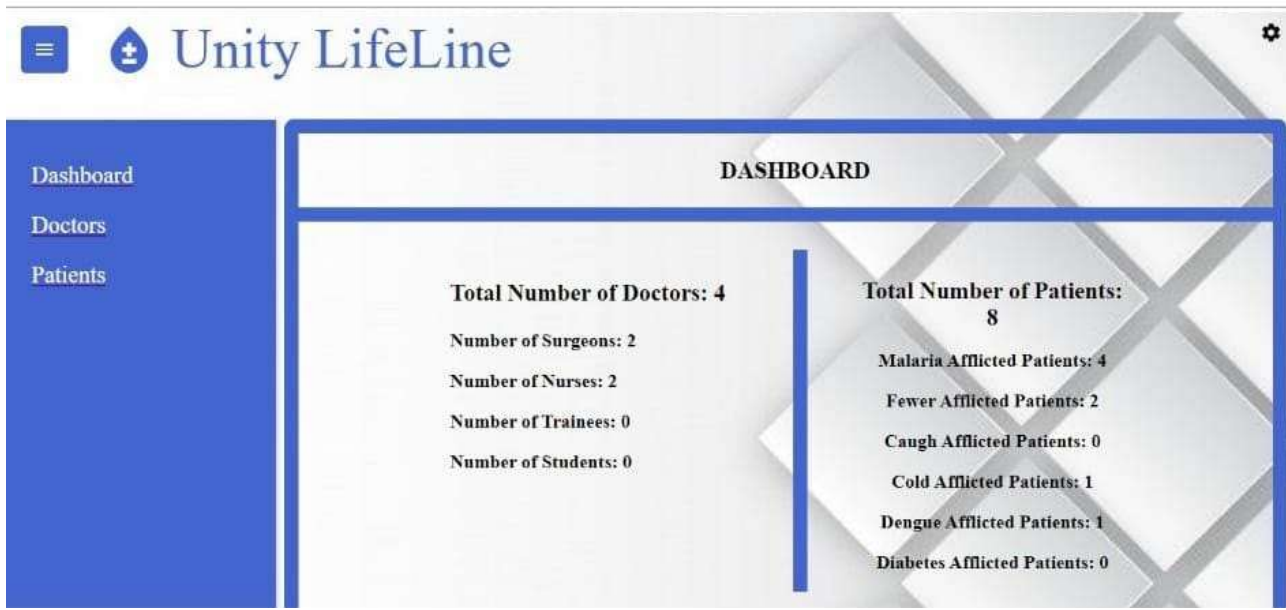
    const response = await
    axios.get(`http://localhost:1337/patients/${endpoint}?${queryParams}`);
    setpatients(response.data);

  } catch (error) {
    console.error('Error fetching patients by filter:', error);
  }
};

```

# DASHBOARD

The dashboard displays the total number of Doctors and Patients available which is calculated at their respective pages



## AT THE BACKEND:

### CONFIGURE ROUTES:

```
'GET /doctors/getCountByDepartment': 'DoctorController.getCountByDepartment',  
'GET /patients/getCountByDisease': 'PatientsController.getCountByDisease',
```

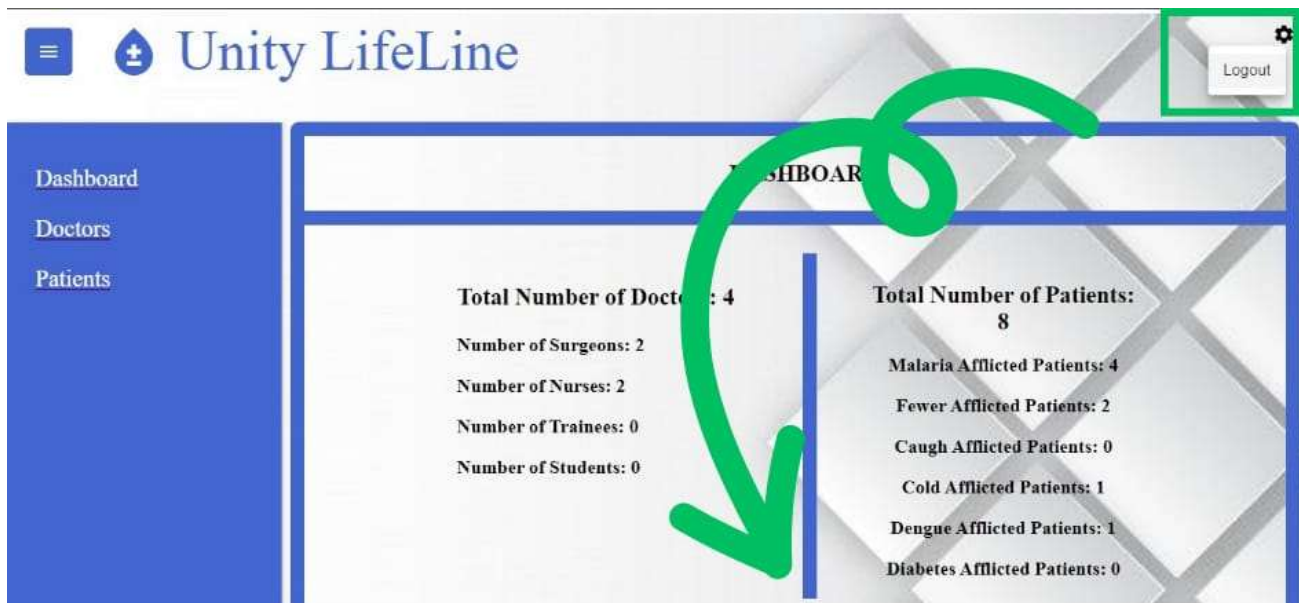
### CONTROLLER'S LOGIC:

```
//FOR DOCTOR:  
getCountByDepartment :async function(req, res) {  
  const { department } = req.query;  
  try {  
    const count = await Doctor.count({ department });  
    res.json({ count });  
  } catch (error) {  
    console.error('Error fetching count by department:', error);  
    res.status(500).json({ error: 'Internal server error' });  
  }  
},
```

```
//FOR PATIENT'S  
getCountByDisease :async function(req, res) {  
  const { disease } = req.query;  
  try {  
    const count = await Patients.count({ disease});  
    res.json({ count });  
  } catch (error) {  
    console.error('Error fetching count by department:', error);  
  }  
}
```

```
res.status(500).json({ error: 'Internal server error' });  
}  
},
```

Now, at the top right corner when you click on logout, you would be directed to the Sign In page.



The Sign In form is a white box with a purple lock icon at the top. It contains the following elements:

- Sign In** (Title)
- Email \*
- Password \*
- ☐ Remember Me
- 
- [Forgot Password?](#)
- [Do you have an Account? Signup](#)

On Click on the Logout button you would be brought back to the sign On page

-Thank you

