Project Specifications

Product Owner Name: Anirudh Belwadi
Product Owner Andrew ID: abelwadi

Product Backlog

Doctor Module

Authentication: OAuth for secure login

Manage the patients and their records: Search, add, edit or delete any record

Notification System: Send screening notification to patient with a verified pre-prepared template of questions

Screening Report: View the report generated from the screening

Patient Module

Authentication: OAuth for secure login

Details of the patient: To be captured while signing up

Give access to a provider: The patient will be able to search and give data access to a provider

Enable Notification channel: Patient will be able to activate a channel (chat on the portal with a bot or phone call) to receive notification for screening

Report viewing: View the report that was generated from the screening and verify it before being sent to the provider.

Call screening Module

Voice to Text Interaction: AI generated voice calls will be made to the patient

Data processed to AI reporting server: AI generates valid follow up questions to gather all the information required from the template of questions

Speech to Text: All the responses of the interaction will be transcribed and sent to report generation module

Chat screening Module

Chat interface: A bot will interact with the patient to gather all the information required in the template of screening questions

Data processed to AI reporting server: AI generates valid follow up questions to gather all the information required from the template of questions

Reporting of text transcription: All the responses of the interaction will be sent to report generation module

AI report generation module

NLP Analysis: AI will process generated transcripts from the screening to detect complaints, symptoms and also possibly provide probable diagnosis

Report Generation: Standardized report for screening will be generated from the analysis

Report sent to backend service: Generated report will be sent securely to the backend service to tie it with the specific patient.

Backend & Data Management Module

Database Management: Store both patient and doctors' information in a SQL database

APIs for smooth interaction: RESTful APIs for collecting and giving information across different modules in the system

First Sprint Backlog

1. Simple Authentication to onboard both patients and doctors
2. Set up required data to be collected from both types of users
3. Interface for doctors to view all the patient records and also dummy screening data.
4. Send notification function on the doctor's end to be enabled
5. Develop the chat interface for the patient to interact and dummy responses can be sent from the backend

Task Allocation

Task	**Team Member**
UI Mockups	Anirudh
Data Models, Forms, URLs	Jeremy, Lidia
UI - Doctor - Auth, Signup, Profile	Jeremy
UI - Doctor - View patient records	Jeremy
UI - Doctor - Send screening notification and view dummy reports	Jeremy
UI - Patient - Auth, Signup, Profile	Lidia
UI - Patient - Chat interface for screening	Lidia
Actions - Doctor - Auth, signup and profile	Anirudh
Actions - Doctor - Sending patient record information	Anirudh
Actions - Doctor - Send notification to patient as received from the doctor	Anirudh
Actions - Patient - Auth, signup and profile	Anirudh
Actions - Patient - Send logged in patient information	Anirudh
Actions - Patient - Chatbot interaction - sending dummy follow ups and storing the responses sent by the patient	Anirudh

UI - Comprises of creation of HTML templates, CSS, scripts of JavaScript and jQuery

Actions - Comprise of basically writing views.py but not restricted to it. We are generally calling it actions to take care of the backend logic for data processing

Data Models, Forms, URLs are self explanatory.

Data Model Specification

Patient (extends User Model)

- Name (First and Last)
- Email
- Address
- DOB
- Gender
- Phone No
- Blood Pressure
- Blood Sugar Level
- Weight
- Height
- Allergies (Multi Valued)

Medications

- Patient (FK)
- Name
- Frequency

Health Conditions

- Patient (FK)
- Name
- Severity
- Date of Diagnosis

Doctor (extends User Model)

- Name (First and Last)
- Email
- Phone
- Institution
- Speciality
- Patients Many to Many relation with Patient

Screening Request

- Request Date
- Due Date
- Patient (FK)
- Doctor (FK)
- Conversation JSON record
- Completion Timestamp

models.py code:

```
from django.db import models
from django.contrib.auth.models import User
from django.contrib.postgres.fields import ArrayField

class Patient(models.Model):
        user = models.OneToOneField(User, ondelete=models.PROTECT)
        first_name = models.CharField(max_length=50)
        last_name = models.CharField(max_length=50)
        email = models.CharField(max_length=50)
        address = models.CharField(max_length=255)
        date_of_birth = models.DateField()
        gender = models.CharField(max_length=10, choices=[('Male', 'Male'), ('Female', 'Female')])
        phone_number = models.CharField(max_length=15)
        blood_pressure = models.CharField(max_length=10, blank=True, null=True)
        blood_sugar_level = models.CharField(max_length=10, blank=True, null=True)
        weight = models.FloatField(blank=True, null=True)
        height = models.FloatField(blank=True, null=True)
        allergies = ArrayField(models.CharField(max_length=255), blank=True, null=True)

        def __str__(self):
                return self.user.username

class Medication(models.Model):
        patient = models.ForeignKey(Patient, on_delete=models.CASCADE, related_name="medications")
        name = models.CharField(max_length=255)
        frequency = models.CharField(max_length=255)

        def __str__(self):
                return f"{self.name} ({self.frequency})"


class HealthCondition(models.Model):
        patient = models.ForeignKey(Patient, on_delete=models.CASCADE)
        name = models.CharField(max_length=200)
        severity = models.CharField(max_length=50)
        diagnosis_date = models.DateField()

        def __str__(self):
                return f"{self.name} ({self.severity})"

class Doctor(models.Model):
        user = models.OneToOneField(User, ondelete=models.PROTECT)
        first_name = models.CharField(max_length=50)
        last_name = models.CharField(max_length=50)
        email = models.CharField(max_length=50)
        phone_number = models.CharField(max_length=15)
        institution = models.CharField(max_length=255)
        specialty = models.CharField(max_length=255)
        patients = models.ManyToManyField(Patient, ondelete=models.PROTECT)

class ScreeningRequest(models.Model):
        request_date = models.DateField(auto_now_add=True)
        due_date = models.DateField()
        patient = models.ForeignKey(Patient, on_delete=models.CASCADE)
```

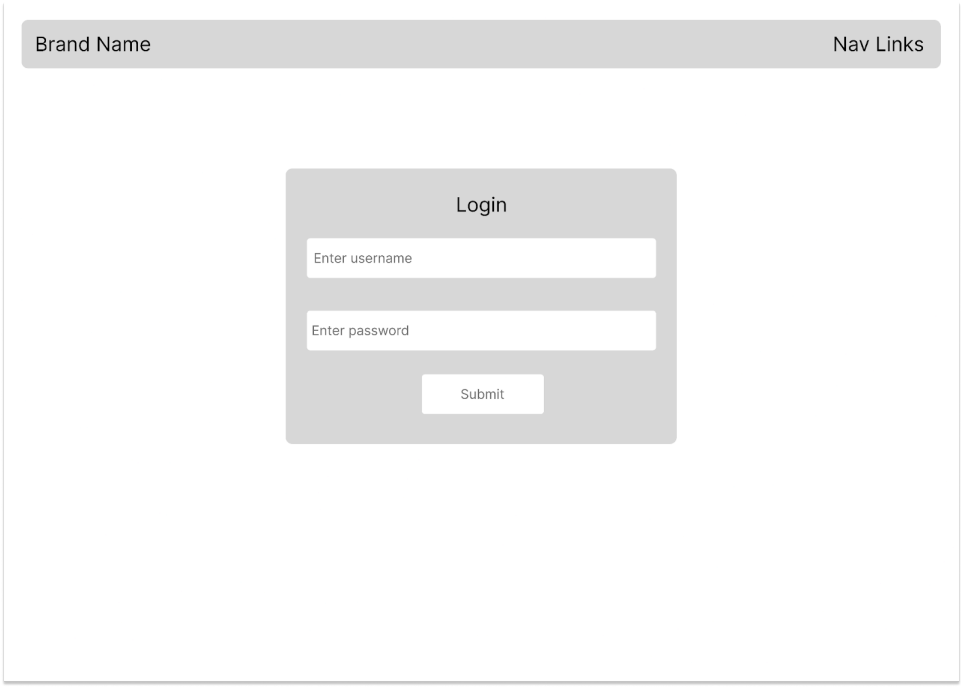
```
        doctor = models.ForeignKey(Doctor, on_delete=models.CASCADE)
        conversation = models.JSONField(blank=True, null=True)
        completion_timestamp = models.DateTimeField(null=True, blank=True)

        def __str__(self):
                return f"Screening Request ({self.patient.user.username} -
{self.doctor.user.username})"
```

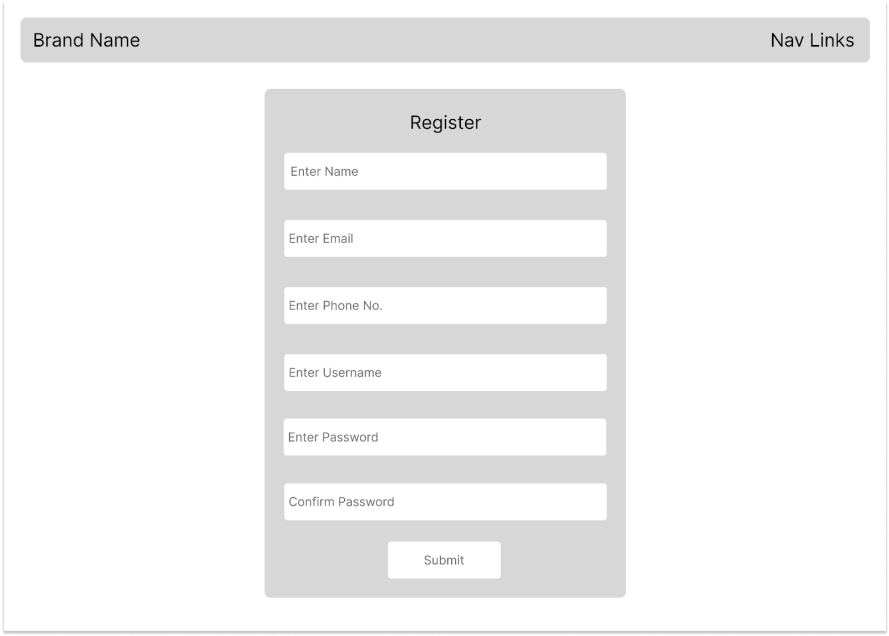
Wireframes & Mockups

Link to Figma Mockup:
https://www.figma.com/design/E9Jpiyvvsea2fwDNXb4OD7/Mockups-for?node-id=0-1&t=dX9TXWh3aKh7Hb3X-1

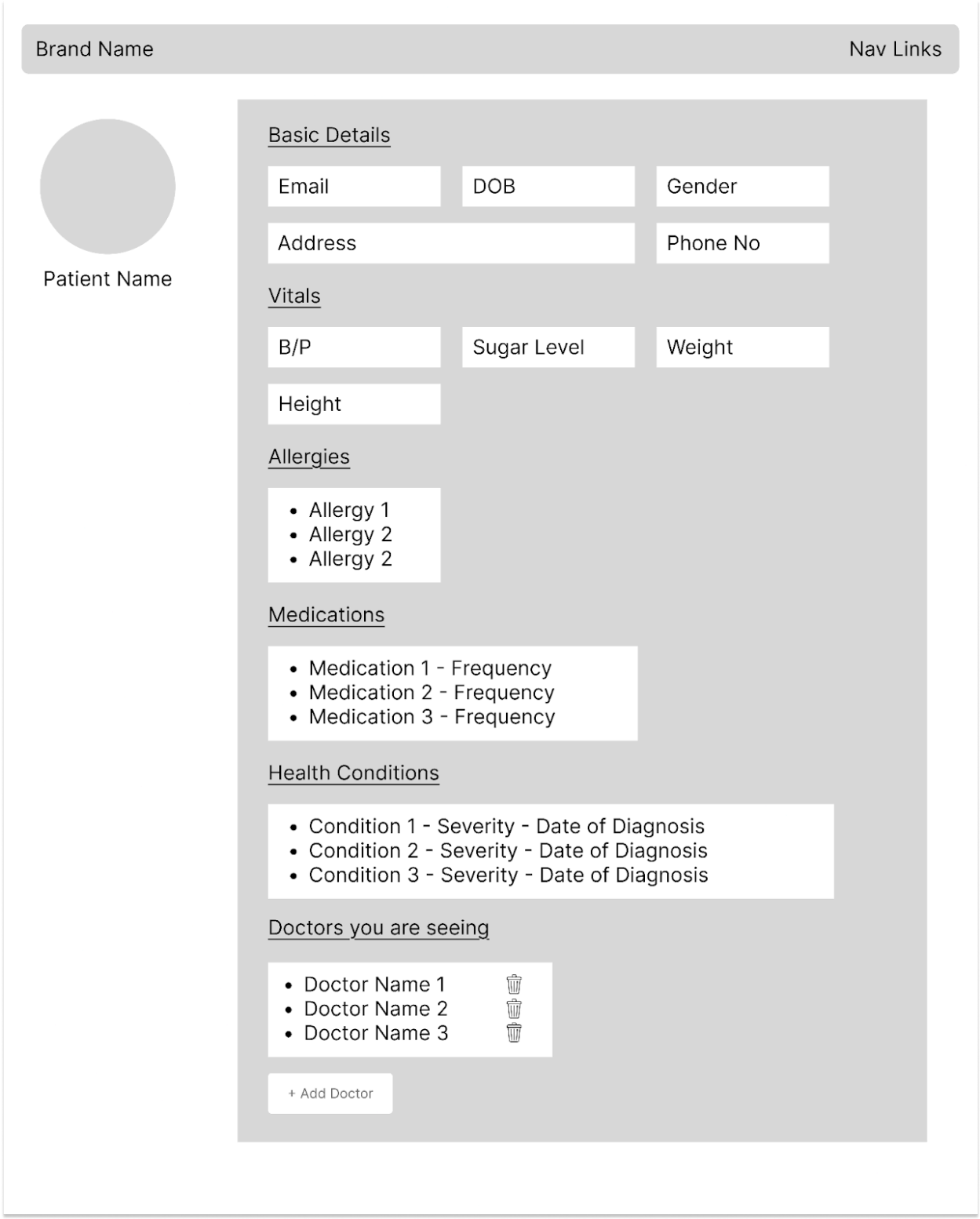
Patient/Doctor Login



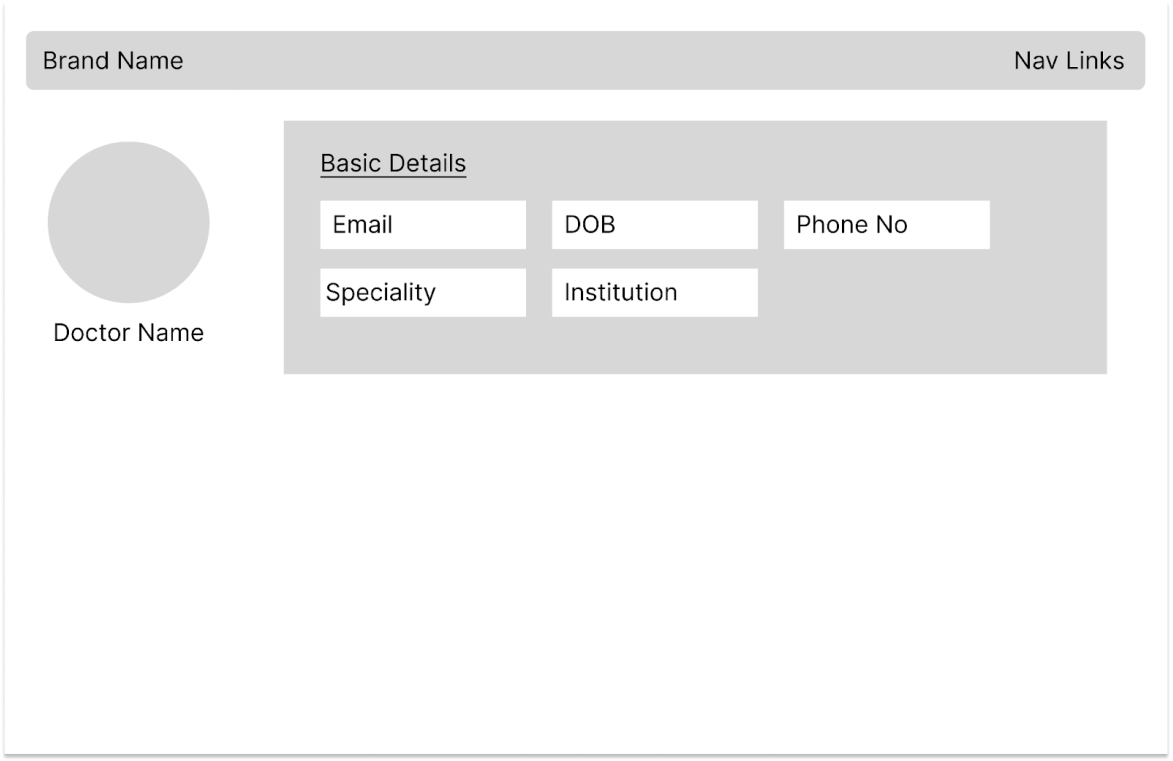
Patient/Doctor Signup



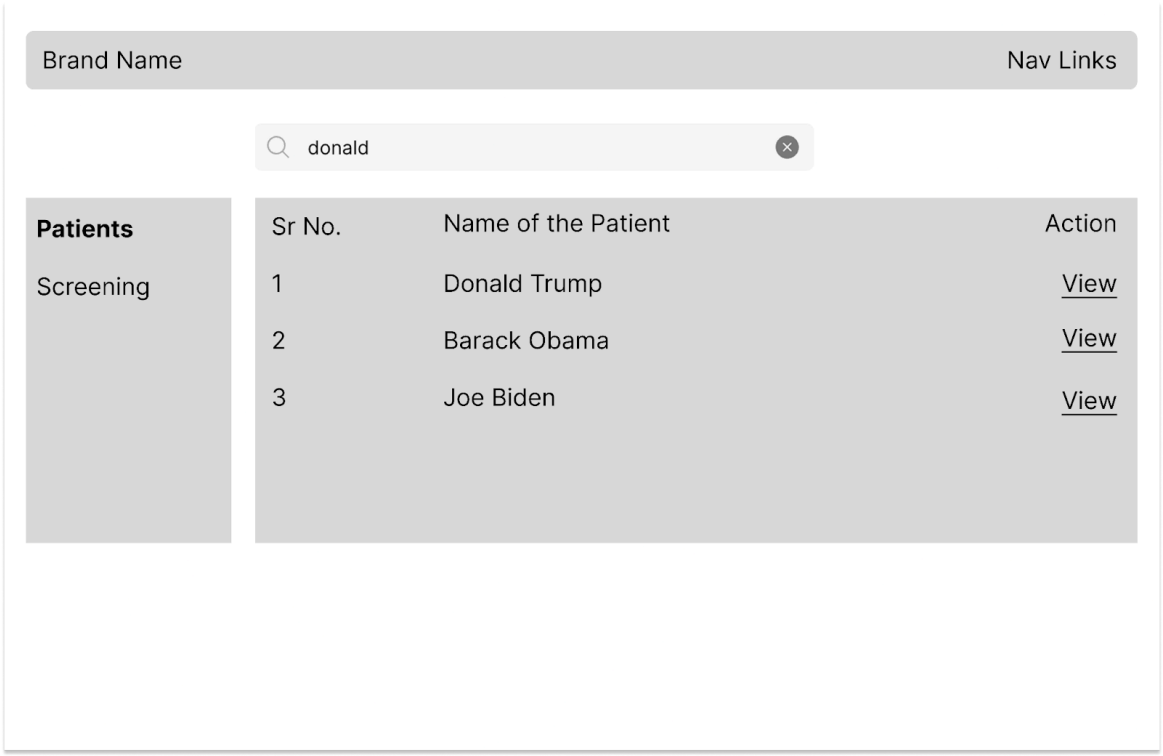
Patient Profile



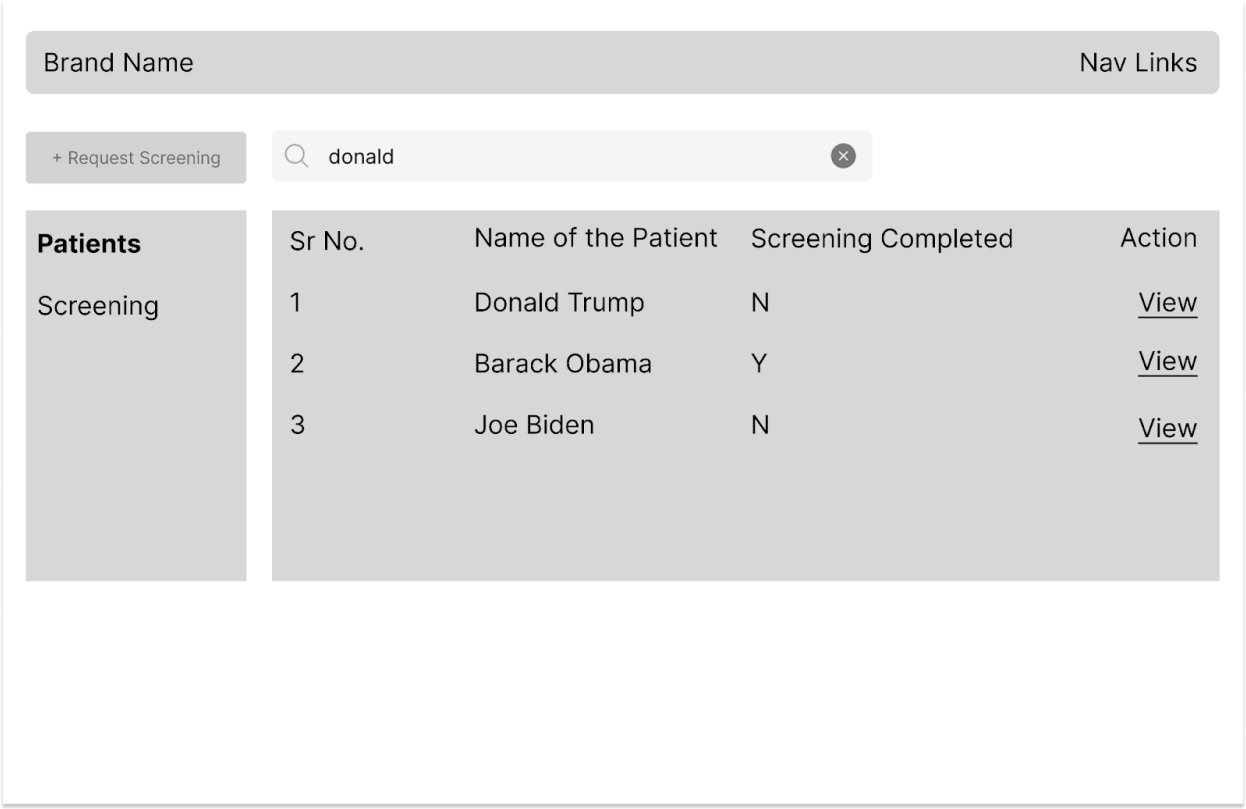
Doctor - Profile



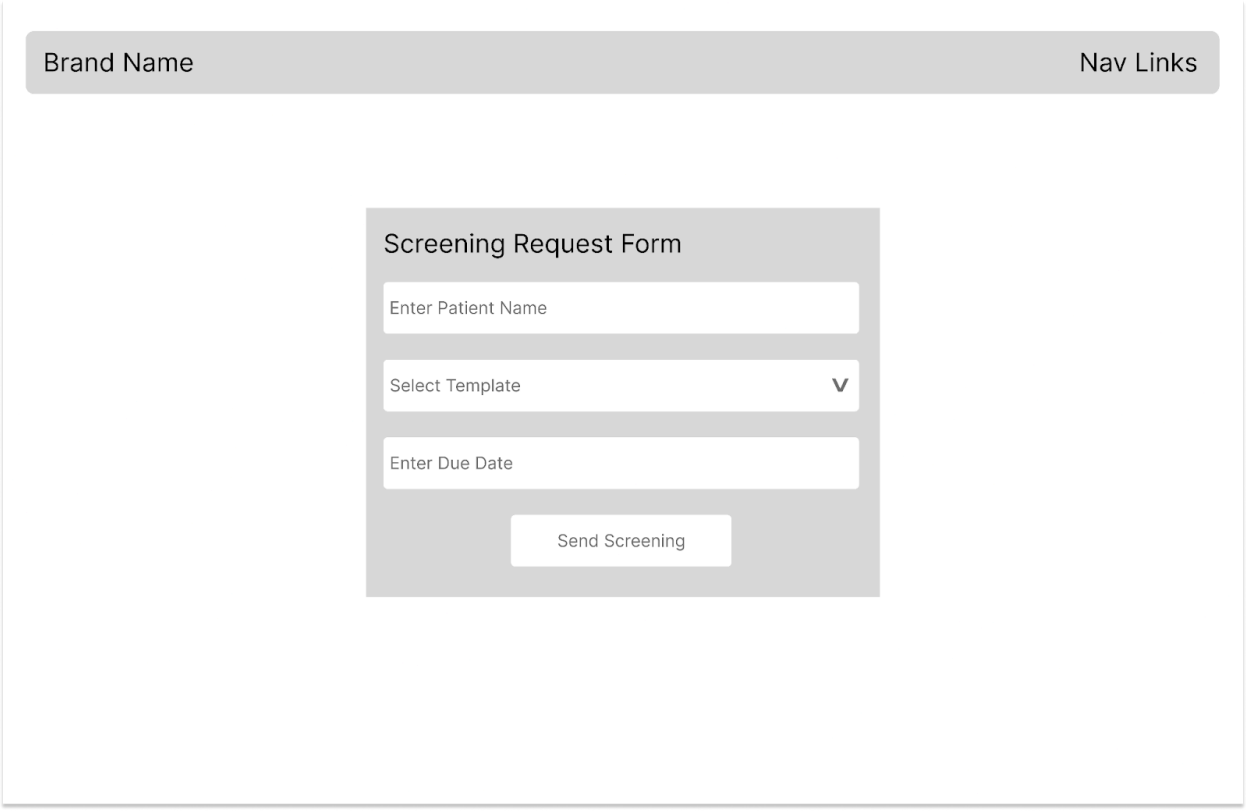
Doctor - Dashboard - View Patients (View button will take to Patient Profile Page)



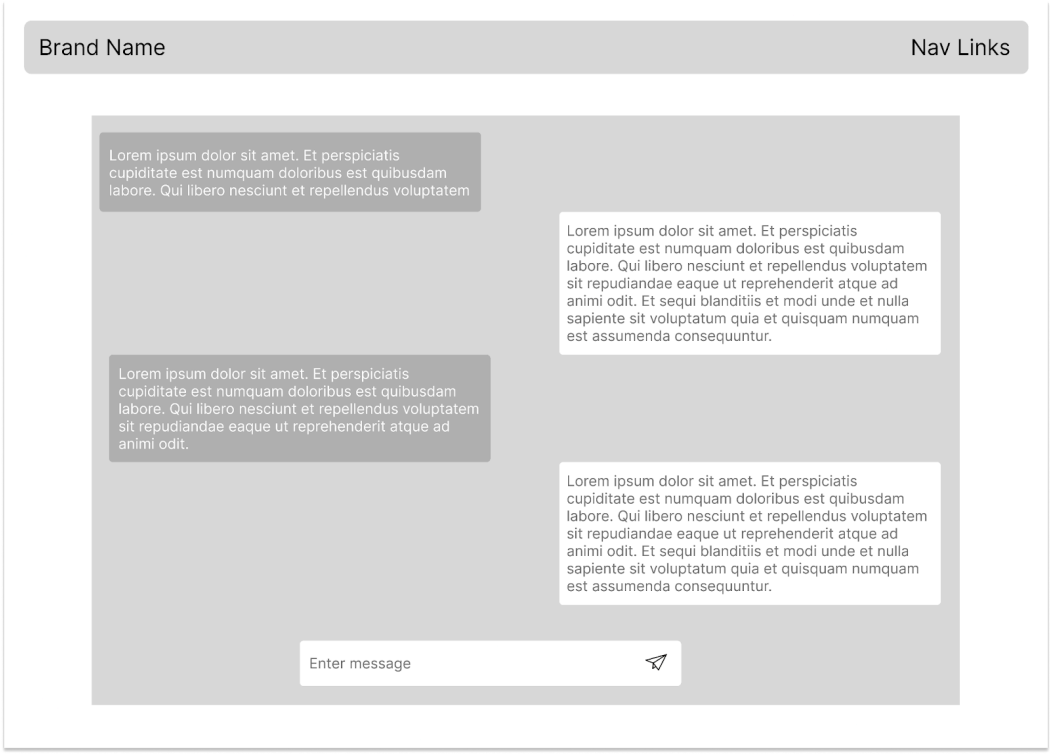
Doctor - Dashboard - Screenings (View button will take to View Screening Page)



Doctor - Request Screening



Patient - Screening Chat Interface



Doctor - View Screening Report (View button will take to Patient Profile Page) (For first sprint we have kept is minimal to show only the conversation, which will be transformed to AI summarization, in the future sprints)