HEALTH CARE DATA HANDLER

A PROJECT REPORT

Submitted by

VAGISHWARAN M (211420104292)

VIGNESH S (211420104302)

VISHNU SANKAR S P (211420104308)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MARCH 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report HEALTH CARE DATA HANDLER is the bonafide work of **VAGISHWARAN M (211420104292), VIGNESH S (211420104302), VISHNU SANKAR S P (211420104308)**, who carried out the project work under my supervision.

Signature of the HOD with date	**Signature of the Supervisor with date**
Dr. L. JABASHEELA M.E., Ph.D.,	**Dr. M. KRISHNAMOORTHY M.E., MBA., Ph.D.,**
PROFESSOR AND HEAD,	**ASSOCIATE PROFESSOR,**
Department of Computer Science and Engineering, Panimalar Engineering College, Chennai - 600123	Department of Computer Science and Engineering, Panimalar Engineering College, Chennai - 600123

Certified that the above candidates were examined in the End Semester Project Viva- Voce Examination held on..........................

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **VAGISHWARAN M (211420104292), VIGNESH S (211420104302), VISHNU SANKAR S P (211420104308),** hereby declare that this project report titled **HEALTH CARE DATA HANDLER** , under the guidance of **Dr.M.KRISHNAMOORTHY M.E., MBA., Ph.D.,** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

VAGISHWARAN M

VIGNESH S

VISHNU SANKAR S P

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**., for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project

We want to express our deep gratitude to our Directors,**Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of the project.

We would like to express our sincere thanks to **Dr. V. SUBEDHA M.Tech., Ph.D.,** and **Dr. M. KRISHNAMOORTHY M.E., MBA., Ph.D.,** and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

VAGISHWARAN M (211420104292)

VISHNUSANKAR S P (211420104308)

VIGNESH S (211420104302)

ABSTRACT

The demand for efficient and accurate medical record-keeping has been on the rise in recent years. With the increasing number of patients and the complexity of medical treatments, healthcare providers have been facing significant challenges in managing and organizing patient data effectively. To address these challenges, an automatic medical form filling application has been developed. NFC technology allows the transfer of data wirelessly between two devices in close proximity. By using NFC-enabled computers, medical professionals can initiate the automatic medical data filling process quickly and easily.The benefits of this innovative application are numerous. First, it reduces the time and effort required for manual data entry, which enables healthcare providers to focus more on patient care. Second, it minimizes errors in medical records, which can significantly contribute to better patient care outcomes. Accurate medical records can help healthcare providers make informed decisions about patient treatment plans, which can improve patient outcomes and reduce healthcare costs. Furthermore, the application can help healthcare providers stay up-to-date with the latest medical standards and protocols. It can also streamline administrative tasks, such as scheduling appointments, sending reminders, and processing insurance claims. By reducing the burden of administrative tasks, healthcare providers can focus more on patient care, which can lead to better patient outcomes.

In conclusion, the automatic medical form filling application has the potential to revolutionize the medical industry by improving the accuracy of medical records and reducing the burden of administrative tasks. By using NFC technology to automate the process of filling out medical forms, healthcare providers can save time and effort, while also improving patient care outcomes.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The demand for efficient and accurate medical record-keeping has been on the rise due to the increasing complexity of medical treatments and the growing number of patients. The traditional manual data entry process for medical forms is time-consuming, labor-intensive, and prone to errors, which can result in delayed treatment, misdiagnosis, and medical errors. The healthcare industry has been looking for ways to streamline this process, and an automatic medical form filling application has been developed to address these challenges. This application aims to automate the process of filling out medical forms using Near Field Communication (NFC) technology. The application allows for the transfer of data between devices without the need for wires or an internet connection.

NFC technology ensures secure and reliable data transfer, reducing the risk of data breaches.

The automatic medical form filling application integrates with the patient device’s local storage,hence ensuring that the information remains secure and protected. This system can access the saved patient data and initiate writing it to an NFC tag. The application's automatic form filling feature is designed to minimize errors and improve the accuracy of medical records. By reducing the burden of administrative tasks, healthcare providers can focus more on patient care, leading to better patient outcomes.

This innovative application has the potential to revolutionize the medical industry by reducing the burden of administrative tasks and improving the accuracy of medical records. The application can lead to better patient care outcomes by minimizing errors in medical records.

1.2 PROBLEM DEFINITION

- Manual Data Entry Problems: Manual entry of medical forms is slow, error-prone, and can lead to serious issues like delayed treatments and misdiagnosis.

- Challenges for Healthcare Providers: Organizing and managing patient data efficiently is tough due to reliance on manual processes and poor integration with electronic medical record (EMR) systems.

- Impact on Patient Care: Inefficiencies in record-keeping can negatively affect the quality of care patients receive.

- Need for Better Systems: With more complex medical treatments and an increasing number of patients, accurate and efficient record-keeping is more important than ever.

- Technology as a Solution: Using artificial intelligence (AI), machine learning (ML), and natural language processing (NLP) can automate data entry, reduce errors, and improve efficiency.

- Blockchain for Security: Blockchain technology can enhance the security and sharing of patient data across different healthcare providers.

- Moving Forward: Healthcare providers, technology developers, and policymakers need to work together to adopt these technologies and improve medical record-keeping and patient care.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The patient data is typically entered into the hospital's electronic health record (EHR) system manually , which is a digital filing system used to store and manage patient health information. There are several methods used to enter patient data into the EHR system, including:

● Manual Data Entry: This is the most common method of entering patient data into the EHR system. Healthcare professionals, such as nurses or medical assistants, typically input patient data into the EHR system during the patient's visit. They use a computer or a tablet with a keyboard or a touchscreen to enter information such as the patient's name, date of birth, medical history, medications, allergies, and vital signs.

● Barcode Scanning: Another method used to enter patient data into the EHR system is through barcode scanning. Patient information is stored in a barcode, which can be scanned by a handheld scanner or a computer. This method is commonly used to enter medications, and other relevant information into the EHR system.

● Patient Portals: Some hospitals allow patients to enter their own data into the EHR system through a patient portal. Patients can log into the portal using their own secure login credentials and enter their personal information, medical history, medications, and other relevant information.

Issues in Existing System

- EHR systems face challenges such as user errors, technical issues, data security and privacy concerns, interoperability challenges, cost, and workflow integration.
- Healthcare organizations must address these issues to ensure patient care is not compromised.
- Measures to address these issues include appropriate training for healthcare professionals, regular maintenance and technical support, compliance with data security and privacy regulations, data sharing between different systems and providers, evaluating the costs and benefits of EHR systems, and careful integration into existing workflows.
- Failure to address these issues can result in errors in patient care and treatment, disruption of patient care, breaches of patient data, and increased costs for healthcare organizations.
- Overcoming these challenges can improve the efficiency and accuracy of medical record-keeping, ultimately leading to better patient care.

2.2 PROPOSED SYSTEM

The introduction of an automatic medical form filling application that utilizes NFC (Near Field Communication) technology represents a significant step forward in addressing the challenges associated with manual data entry in healthcare settings. This innovative approach not only streamlines the process of patient medical data sharing but also introduces a new level of efficiency and accuracy in maintaining electronic health records (EHRs). Below are expanded points on how this application transforms healthcare data management and addresses existing issues:

Enhanced Efficiency

Rapid Data Transfer: The use of NFC technology for data sharing significantly reduces the time required to input patient information into EHR systems, making the process quicker and more efficient.

Streamlined Workflow: Healthcare professionals can access patient data almost instantaneously by tapping the NFC-enabled device on a reader, streamlining the workflow and allowing more time for patient care.

Improved Accuracy

Reduction of Human Errors: By automating the data entry process, the risk of errors commonly associated with manual input is greatly minimized. This leads to more reliable and accurate medical records, which are crucial for diagnosis and treatment planning.

Consistency in Data Entry: The application ensures that data entered is uniform and consistent, further improving the quality of patient records.

Seamless Integration

Compatibility with Existing EHR Systems: The application is designed to integrate seamlessly with existing electronic health record systems, ensuring that new data is accurately merged with existing patient records without disrupting the integrity of the data.

Secure Data Exchange: By using NFC and secure data transmission protocols, the application ensures that all patient information remains secure and protected against unauthorized access.

User-Centric Design

Simplicity for Patients: Patients are required to enter their medical data into a mobile application once, after which their data can be securely stored and easily transmitted through a heka tag. This process reduces the burden on patients to repeatedly provide the same information.

Convenience for Healthcare Providers: Medical professionals can access up-to-date patient data with a simple tap, using an NFC tag reader, through a web application designed for ease of use and maximum efficiency.

Data Management and Privacy

Local Storage of Data: The initial storage of medical data on the patient's mobile device in local storage before writing to the heka tag ensures that patients have control over their personal information.

Compliance and Security: The application is built with a focus on compliance with healthcare regulations and standards for data security, ensuring that patient privacy is maintained and data is protected against breaches.

Future Implications

Potential for Wider Adoption: The success of this system could encourage wider adoption of NFC technology in healthcare, further revolutionizing how patient data is handled and shared.

Integration with Other Technologies: There's potential for integration with other technologies such as AI for predictive analytics and blockchain for enhanced security, opening up new avenues for improving patient care and outcomes.

By addressing the major issues of time consumption, error proneness, and integration challenges associated with traditional EHR systems, this automatic medical form filling application paves the way for a more efficient, secure, and patient-centered approach to healthcare data management.

2.3 DEVELOPMENT ENVIRONMENT

2.3.1 SOFTWARE REQUIREMENT

- Windows/Mac OS
- HTML
- CSS
- Javascript
- React Native
- VS code
- Browser

2.3.2 HARDWARE REQUIREMENT

- Processor: Minimum 1 GHz
- NFC tag reader
- Memory (RAM): 4 GB
- Hard Drive: 32 GB
- Internet Connection.CHAPTER 3

SYSTEM DESIGN

3.1 UML DIAGRAMS

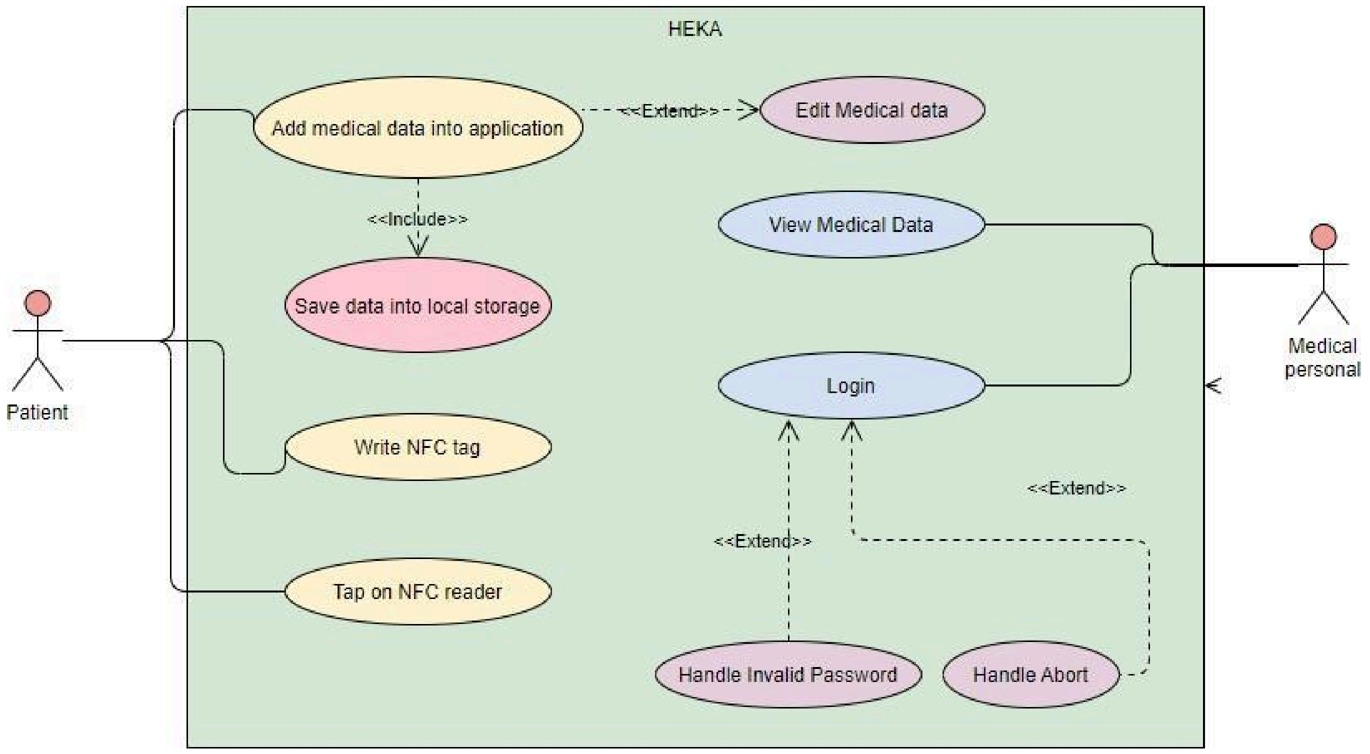


Fig 3.1.1 Use case diagram for Heka

This use case diagram refers to activities done by User(Patients) and Doctors(Medical Personal) and their corresponding use cases

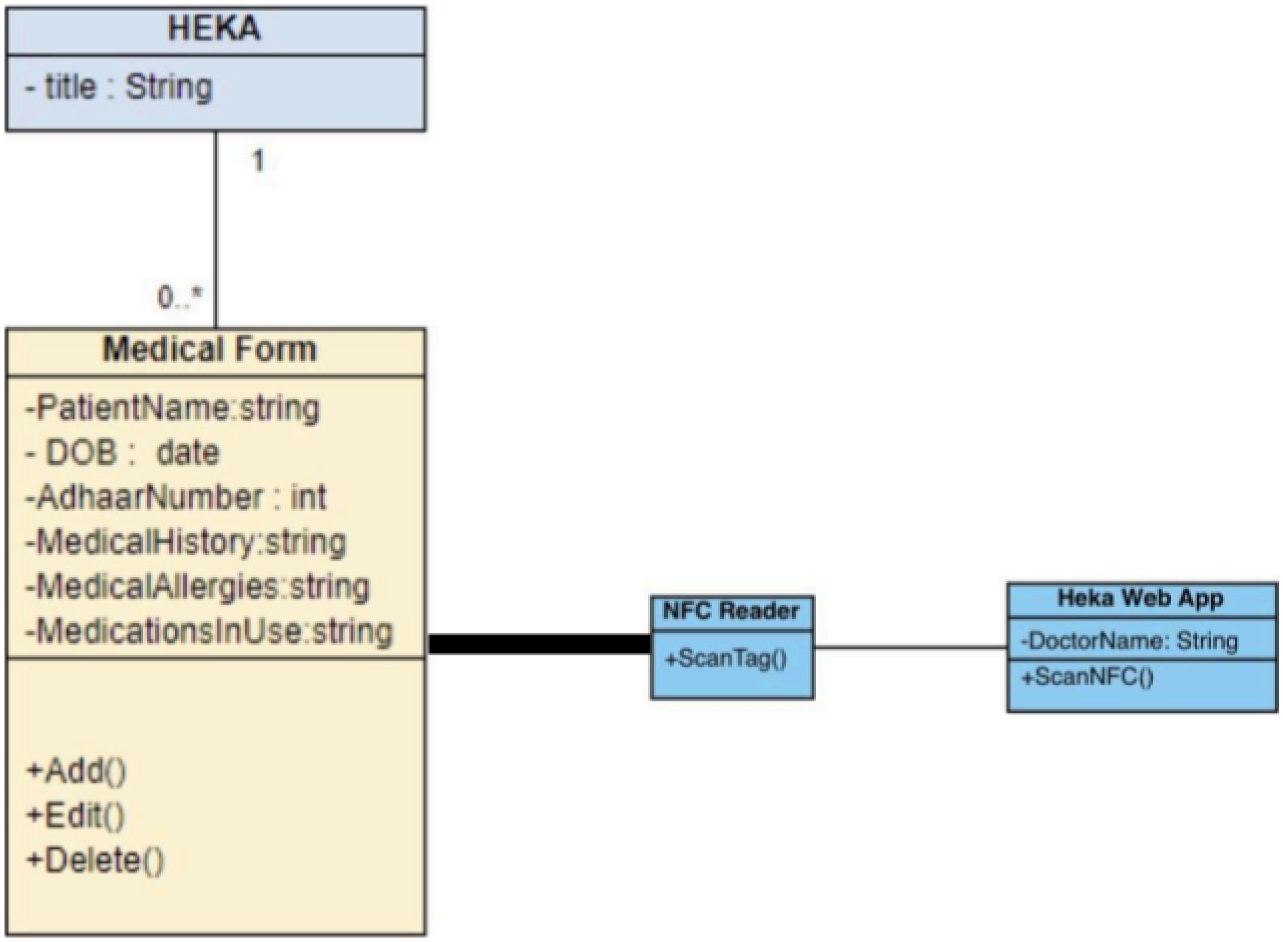


Fig 3.1.2 Class diagram for Heka

The class diagram refers to relationships between different classes.

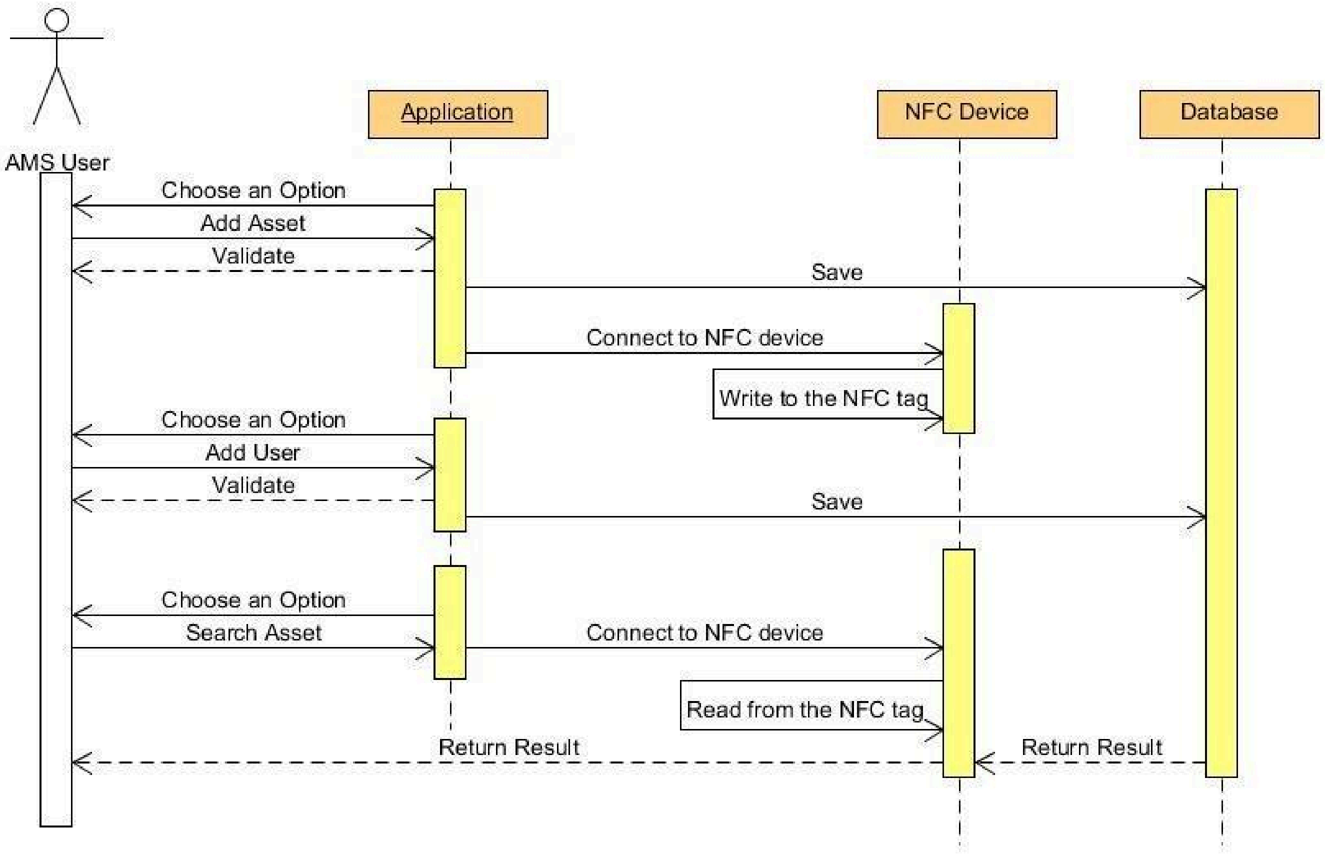


Fig 3.1.3 Sequence diagram for Heka

The sequence diagram of Heka of patients using the app to transfer medical files to doctors.

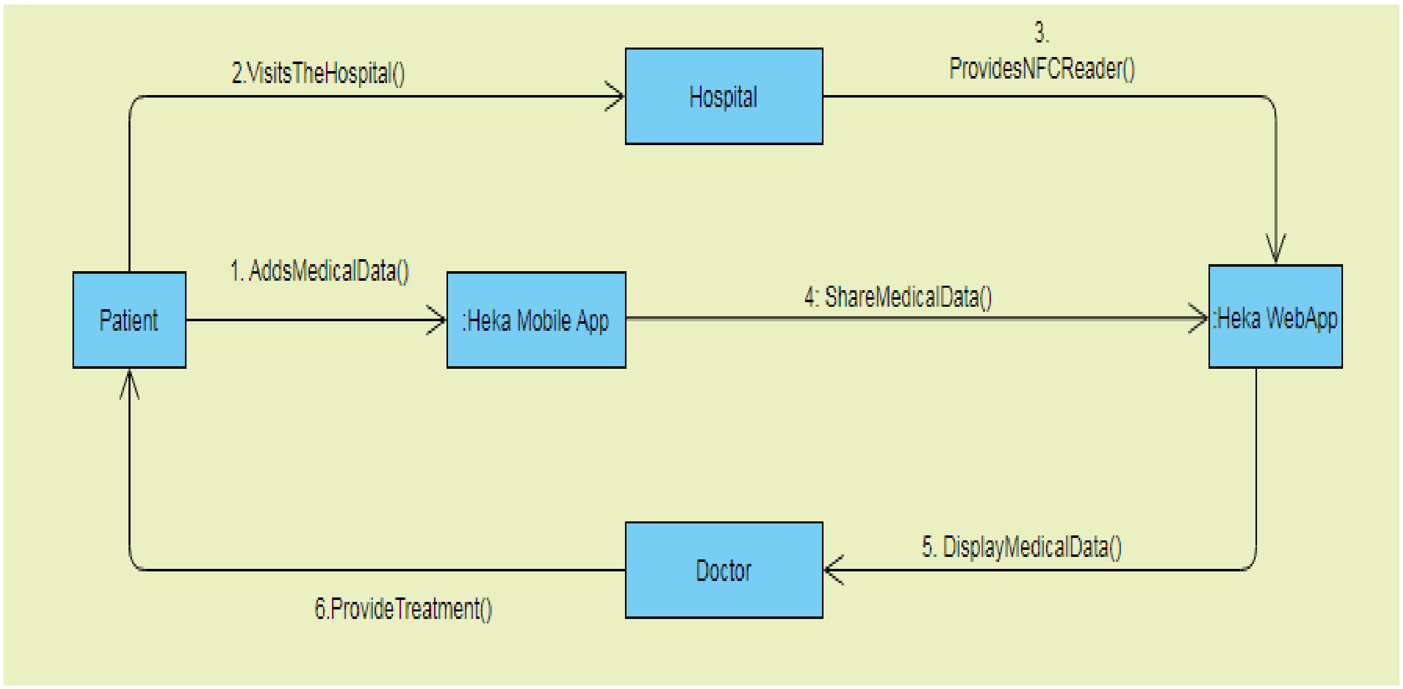


Fig 3.1.4 Collaboration diagram for Heka

The collaboration diagram of Heka shows the sequence of activities of adding data and sharing with doctor

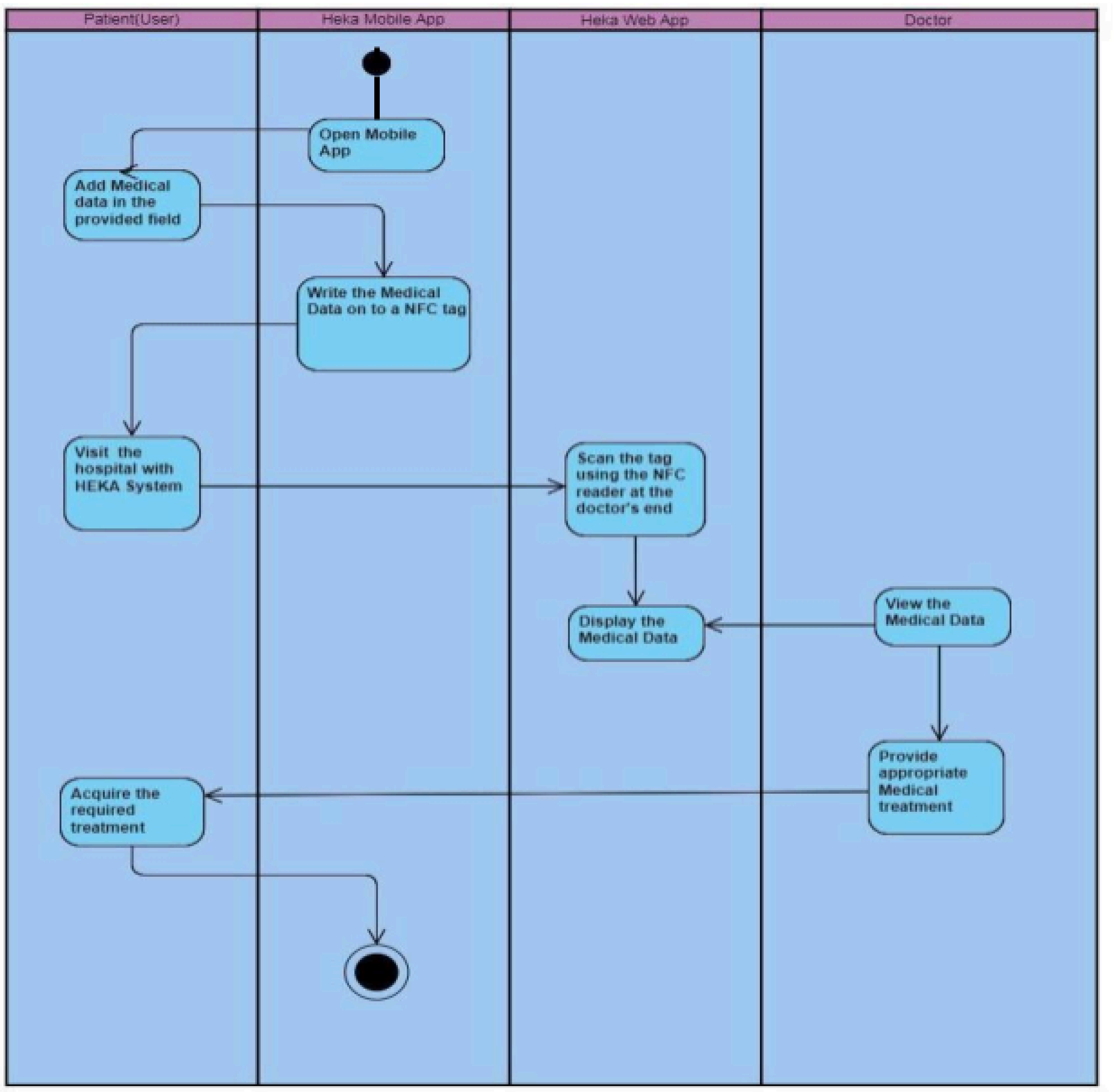

Patient(User)
Heka Mobile App
Heka Web App
Doctor
Open Mobile App
Add Medical data in the provided field
Write the Medical Data on to a NFC tag
Visit the hospital with HEKA System
Scan the tag using the NFC reader at the doctor's end
Display the Medical Data
View the Medical Data
Provide appropriate Medical treatment
Acquire the required treatment

Fig 3.1.5 Activity diagram for Heka

The activity diagram of Heka of patients sharing medical data with doctors.

3.2 DATA DICTIONARY

This is normally represented as the data about data. It is also termed as metadata some times which gives the data about the data stored in the database. It defines each data term encountered during the analysis and design of a new system. Data elements can describe files or the processes. Following are some rules, which defines the construction of data dictionary entries:

1. Words should be defined to understand what they need and not the variable need by which they may be described in the program.

2. Each word must be unique. We cannot have two definitions of the same client.

3. Aliases or synonyms are allowed when two or more entries show the same meaning.

4. A self-defining word should not be decomposed. It means that the reduction of any information into subparts should be done only if it is really required, that is it is not easy to understand directly.

Data dictionary includes information such as the number of records in a file, the frequency a process will run, security factors like pass word which the user must enter to get excess information.

3.2.1 SIGNUP TABLE

COLUMN NAME	DATA TYPE	DESCRIPTION	CONSTRAINT
NAME	VARCHAR	NAME OF THE USER	NOT NULL
PASSWORD	VARCHAR	PASSWORD OF THE USER	NOT NULL
REPEAT PASSWORD	VARCHAR	REPEAT THE PASSWORD OF THE USER	NOT NULL

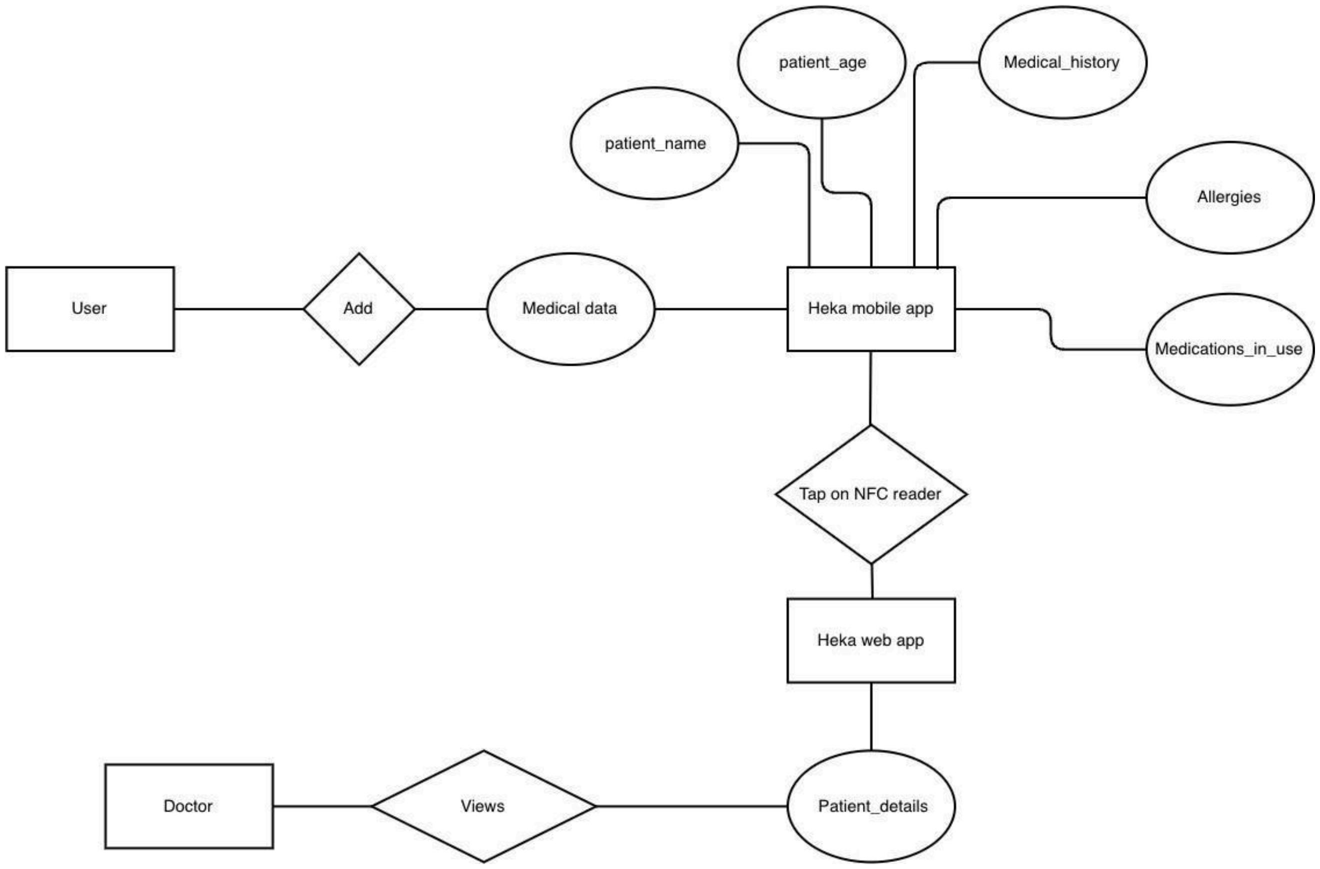
3.2.1 Signup table for Heka

3.2.2 SIGN IN TABLE

COLUMN NAME	DATA TYPE	DESCRIPTION	CONSTRAINT
NAME	VARCHAR	NAME OF THE USER	NOT NULL
PASSWORD	VARCHAR	PASSWORD OF THE USER	NOT NULL

3.2.2 Sign in table for Heka

3.3 ER DIAGRAM



The E-R diagram gives the entities that are Users, app, Doctor and their attributes Patient Medical data.

3.4 DATA FLOW DIAGRAM

0 LEVEL DFD

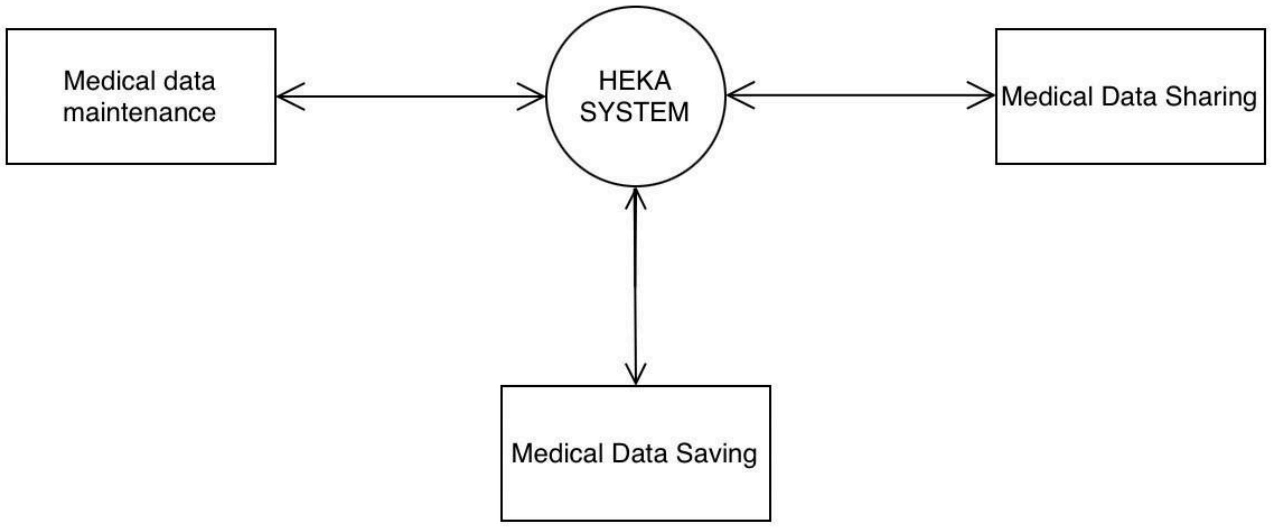


Fig 3.3.1 Data Flow diagram level 0

The zero level data flow diagram of Heka System shows the various capabilities of the system.

SECOND LEVEL DFD

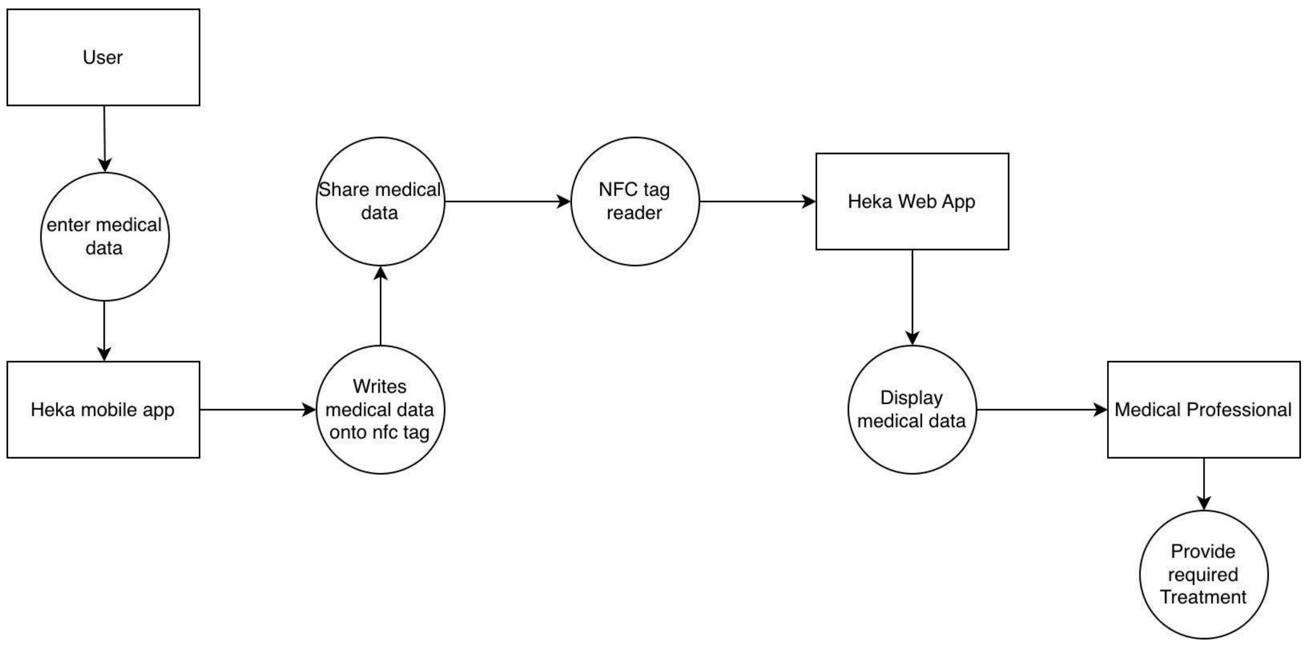


Fig 3.3.3 Data Flow diagram level 2

The second level of the data flow diagram of Heka showing the flow of data and the result.

CHAPTER 4

SYSTEM ARCHITECTURE

4.1 ARCHITECTURE OVERVIEW

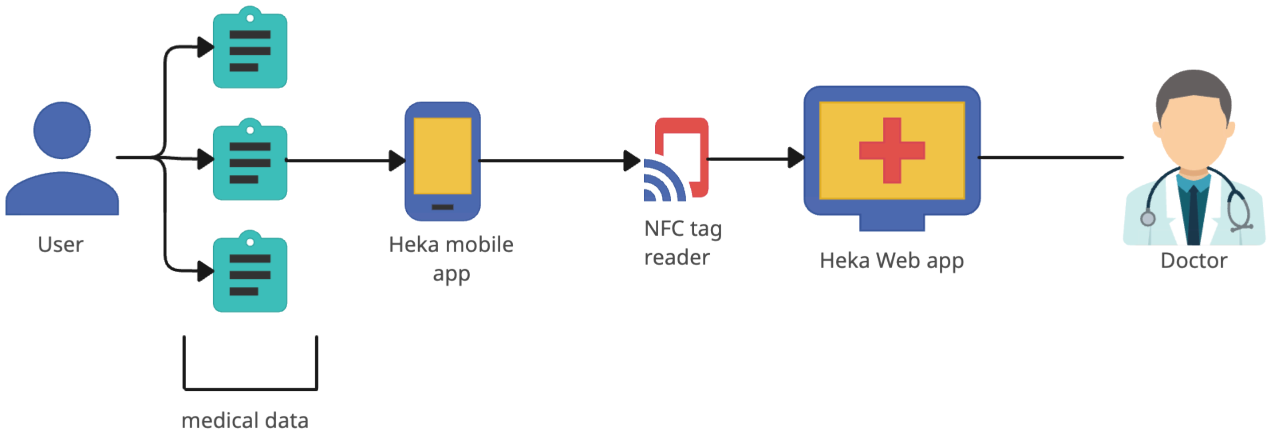


Fig 4.1 Architecture diagram for Heka System

Figure 4.1 consists of User who enters medical data into the Heka mobile application, the mobile app will write the data onto an NFC tag, and when tapped on a Heka web app in the subscribed hospital, the doctor can view it and provide required consultation.

4.2 MODULE DESCRIPTION

Heka 3 consists of modules. They are

- Add Medical Data
- Write tag
- View medical data

Add Medical Data module

- Can enter user medical data
- Can alter existing saved data

Write tag module

- Can write medical data onto nfc tag View medical data module
- View patient medical data when tapped on NFC reader.

CHAPTER 5

SYSTEM IMPLEMENTATION

FRONT-END CODING BASE.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>HekaHealth</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1>HekaHealth</h1>
  </header>
  <main>
    <div class="card">
                                                      <div class="card-header">
                                                        <h2>Medical Data</h2>
        <button id="clear-btn">Clear</button>
      </div>
      <div class="card-body">
        <div id="patient-info">
```

```
          <h3>Patient Information</h3>
          <div class="info">
            <p><strong>Name:</strong> <span id="name"></span></p>
            <p><strong>Date of Birth:</strong> <span id="dob"></span></p>
            <p><strong>Gender:</strong> <span id="gender"></span></p>
            <p><strong>Blood Type:</strong> <span
            id="blood-type"></span></p>
            <p><strong>Insurance Provider:</strong> <span
id="insurance-provider"></span></p>
            <p><strong>Insurance Policy Number:</strong> <span
id="insurance-policy-number"></span></p>
          </div>
        </div>
        <div id="medical-history">
          <h3>Medical History</h3>
          <table>
            <thead>
              <tr>
                <th>Date</th>
                <th>Diagnosis</th>
                <th>Prescription</th>
                <th>Doctor</th>
              </tr>
            </thead>
            <tbody id="history-table">
              <tr>
                <td colspan="4">No medical history available.</td>
                                </tr>
                              </tbody>
          </table>
        </div>
        <div id="vitals">
          <h3>Vitals</h3>
          <div class="info">
            <p><strong>Temperature:</strong> <span id="temperature"></span>
&#8457;</p>
            <p><strong>Blood Pressure:</strong> <span
id="blood-pressure"></span> mmHg</p>
            <p><strong>Heart Rate:</strong> <span id="heart-rate"></span>
bpm</p>
            <p><strong>Oxygen Saturation:</strong> <span
id="oxygen-saturation"></span>%</p>
            <p><strong>Respiration Rate:</strong> <span
id="respiration-rate"></span> bpm</p>
          </div>
        </div>
        <div id="allergies">
          <h3>Allergies</h3>
          <ul id="allergies-list">
            <li>No allergies reported.</li>
          </ul>
        </div>
        <div id="medications">
          <h3>Current Medications</h3>
          <ul id="medications-list">
            <li>No medications reported.</li>
          </ul>
        </div>
```

```
    <div id="immunizations">
      <h3>Immunizations</h3>
      <ul id="immunizations-list">
        <li>No immunizations
```

CONTACT.HTML

```
{% extends 'core/base.html' %}

{% block title %}Contact Us{% endblock title %}

{% block content %}

                                                          <h1>Contact Us</h1>

                                                        {% endblock content %}
```

FRONTPAGE.HTML

```
{% extends 'core/base.html' %}

{% block title %}Home Page{% endblock title %}

      <!DOCTYPE html>
      <html>
        <head>
          <title>Heka</title>
      <link rel="stylesheet" href="{% static 'css/styles.css' %}">
    </head>
    <body>
      <header>
        <h1>Heka Health</h1>
      </header>
      <main>
        <h2>Welcome!</h2>
        <p>Use the NFC reader to access patient medical records.</p>
        <form>
          <label for="nfc-reader">Tap the NFC reader to scan the patient's
  data:</label>
          <input type="text" id="nfc-reader" name="nfc-reader" placeholder="Tap
  to scan">
          <button type="submit">Submit</button>
        </form>
      </main>
      <footer>
        <p>&copy; 2023 HekaHealth</p>
      </footer>
    </body>
  </html>
  {% endfor %}
</div>
{% endblock content %}
```

Code to read a nfc tag

```
import NfcManager, { NfcEvents } from 'react-native-nfc-manager';
 const [hasNfc, setHasNFC ] = useState(null); //to check if device supports nfc useEffect(()
=> {
    const checkIsSupported = async () => {
      const deviceIsSupported = await NfcManager.isSupported()

      setHasNFC(deviceIsSupported) if (deviceIsSupported) {
        await NfcManager.start()
      }
    }

    checkIsSupported()
  }, [])
if (hasNfc === null) return null;

  if (!hasNfc) { return (
      <View style={styles.sectionContainer}>
        <Text>NFC not supported</Text>
      </View>
    )
  }

  return (
    <SafeAreaView style={styles.sectionContainer}>
      <Text>Hello world</Text>
      <TouchableOpacity style={[styles.btn, styles.btnScan]} onPress={readTag}>
        <Text style={{ color: "white" }}>Scan Tag</Text>
      </TouchableOpacity>
      <TouchableOpacity style={[styles.btn, styles.btnCancel]} onPress={cancelReadTag}>
        <Text style={{ color: "white" }}>Cancel Scan</Text>
      </TouchableOpacity>
    </SafeAreaView>
  );
```

To write medical data onto an NFC tag

```
const writeNFC = async() => { let result = false;

  try {
    await NfcManager.requestTechnology(NfcTech.Ndef);
    const bytes = Ndef.encodeMessage([Ndef.uriRecord('https://blog.logrocket.com/')]); if
    (bytes) {
      await NfcManager.ndefHandler
        .writeNdefMessage(bytes); result = true;
    }
  } catch (ex) { console.warn(ex);
  } finally { NfcManager.cancelTechnologyRequest();
  }

  return result;
 }
<TouchableOpacity style={[styles.btn, styles.btnWrite]} onPress={writeNFC}>
      <Text style={{ color: "white" }}>Write Tag</Text>
 </TouchableOpacity>
```

React native code for heka mobile app

```
import React, { useState } from 'react';
import { View, Text, TextInput, TouchableOpacity, Alert, StyleSheet } from
'react-native'; import NfcManager, { Ndef } from 'react-native-nfc-manager';


export default function Heka() {
```

```
const [name, setName] = useState(''); const [age, setAge] = useState('');
const [gender, setGender] = useState('');
const [bloodType, setBloodType] = useState('');
const [medicalHistory, setMedicalHistory] = useState(''); const [medicationsInUse,
setMedicationsInUse] = useState(''); const [allergies, setAllergies] = useState('');

const handleWriteNfc = async () => { try {
  await NfcManager.requestTechnology(NfcManager.tech.Ndef, { alertMessage: 'write
    medical data onto NFC tag.',
  });
  const bytes = Ndef.encodeMessage([ Ndef.textRecord(name), Ndef.textRecord(age),
    Ndef.textRecord(gender), Ndef.textRecord(bloodType),
    Ndef.textRecord(medicalHistory), Ndef.textRecord(medicationsInUse),
    Ndef.textRecord(allergies),
  ]);
  await NfcManager.writeNdefMessage(bytes);
  Alert.alert('Success', 'Medical data has been written onto NFC tag.');
 } catch (err) { console.warn(err);
  Alert.alert('Error', 'Unable to write medical data onto NFC tag.');
  }
};

return (
  <View style={styles.container}>
    <Text style={styles.title}>Medical Data Form</Text>
    <TextInput style={styles.input} placeholder="Name" value={name}
      onChangeText={(value) => setName(value)}
    />
    <TextInput style={styles.input} placeholder="Age" value={age}
      onChangeText={(value) => setAge(value)}
    />
    <TextInput style={styles.input} placeholder="Gender" value={gender}
      onChangeText={(value) => setGender(value)}
    />
    <TextInput style={styles.input} placeholder="Blood Type"
```

```
      value={bloodType}
      onChangeText={(value) => setBloodType(value)}
    />
    <TextInput style={styles.input}
      placeholder="Medical History" value={medicalHistory}
      onChangeText={(value) => setMedicalHistory(value)}
    />
    <TextInput style={styles.input}
      placeholder="Medications in Use" value={medicationsInUse}
      onChangeText={(value) => setMedicationsInUse(value)}
    />
    <TextInput style={styles.input} placeholder="Allergies" value={allergies}
      onChangeText={(value) => setAllergies(value)}
    />
    <TouchableOpacity style={styles.button} onPress={handleWriteNfc}>
      <Text style={styles.buttonText}>HekaText>
    </TouchableOpacity>
  </View>
 );
}
```

```
const styles = StyleSheet.create({ container: {
    flex: 1,
    justifyContent: 'center', alignItems: 'center', padding: 20,
  },
  title: { fontSize: 24,
    fontWeight: 'bold', marginBottom: 20,
  },
  input: { height: 40,
    width: '100%', borderColor: 'gray', borderWidth: 1,
    marginBottom: 10,
    padding: 10,
```

MAIN.CSS

```
/* Global styles */
* {
  box-sizing: border-box; margin: 0;
  padding: 0;
}

body {
  background-color: #f8f8f8; font-family: Arial, sans-serif;
}

.container {
  max-width: 800px; margin: 0 auto; padding: 40px; background-color: #fff; border-radius:
  10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
  font-size: 36px; font-weight: bold;
  margin-bottom: 20px; color: #444;
}

h2 {
  font-size: 24px; font-weight: bold;
  margin-bottom: 10px; color: #444;
}

button {
  background-color: #ff69b4; border: none;
  color: #fff;
  padding: 10px 20px; border-radius: 5px; cursor: pointer; transition: all 0.3s ease;
}

button:hover {
  background-color: #ff4d9e;
}

input[type="text"], input[type="date"], input[type="tel"] { width: 100%; padding: 10px;
  margin-bottom: 20px; border-radius: 5px; border: none;
  background-color: #f1f1f1;
}

/* Specific styles for the patient data form */
.patient-form {
  max-width: 500px; margin: 0 auto;
}

.patient-form h2 { margin-top: 40px;
}

.patient-form button { margin-top: 20px;
}

.patient-form label { display: block; font-size: 16px; font-weight: bold;
```

```
  margin-bottom: 5px; color: #444;
}

.patient-form input[type="text"] { font-size: 16px;
}

.patient-form input[type="date"] { font-size: 16px;
}

.patient-form input[type="tel"] { font-size: 16px;
}

/* Specific styles for the medical records table */
.medical-records { margin-top: 40px; width: 100%;
  border-collapse: collapse;
}

.medical-records td,
.medical-records th { border: 1px solid #ddd; padding: 10px;
}

.medical-records th { background-color: #ff69b4; color: #fff;
  font-weight: bold; text-align: left;
}

.medical-records tr:nth-child(even) { background-color: #f2f2f2;
}
```

```
.medical-records tr:hover { background-color: #ddd;
}
```

BACK-END CODING MODEL.PY

```
from django.shortcuts import get_object_or_404, render, redirect from django.urls import
reverse
from .models import Patient from .forms import PatientForm

def home(request):
    patients = Patient.objects.all()
    return render(request, 'app/home.html', {'patients': patients})

def patient_detail(request, pk):
    patient = get_object_or_404(Patient, pk=pk)
    return render(request, 'app/patient_detail.html', {'patient': patient})

def patient_create(request):
    if request.method == 'POST':
        form = PatientForm(request.POST) if form.is_valid():
            patient = form.save()
            return redirect('patient_detail', pk=patient.pk)
    else:
        form = PatientForm()
    return render(request, 'app/patient_form.html', {'form': form})

def patient_update(request, pk):
    patient = get_object_or_404(Patient, pk=pk) if request.method == 'POST':
        form = PatientForm(request.POST, instance=patient) if form.is_valid():
            form.save()
            return redirect('patient_detail', pk=patient.pk)
    else:
        form = PatientForm(instance=patient)
    return render(request, 'app/patient_form.html', {'form': form})
```

```
def patient_delete(request, pk):
    patient = get_object_or_404(Patient, pk=pk) if request.method == 'POST':
        patient.delete()
        return redirect('home')
    return render(request, 'app/patient_confirm_delete.html', {'patient': patient})
```

URLS.PY

```
from django.urls import path

from . import views


urlpatterns = [

    path('', views.home, name='home'),

    path('patient/<int:pk>/', views.patient_detail, name='patient_detail'),

    path('patient/new/', views.patient_create, name='patient_create'),

                              path('patient/<int:pk>/update/',
views.patient_update, name='patient_update'),

    path('patient/<int:pk>/delete/', views.patient_delete, name='patient_delete'),

]
```

VIEWS.PY

```
    from django.shortcuts import render

    from .models import Patient, MedicalRecord


    def home(request):

        if request.method == 'POST':

            # Get patient data from NFC reader

            # and save it to the database
            patient = Patient()

            patient.name = request.POST.get('name')

            patient.date_of_birth = request.POST.get('date_of_birth')

            patient.phone_number = request.POST.get('phone_number')

            patient.save()


            # Get medical record data from NFC

            reader # and save it to the database

            medical_record = MedicalRecord()

            medical_record.patient = patient

            medical_record.date = request.POST.get('date')

            medical_record.description =

            request.POST.get('description') medical_record.save()


        # Retrieve all patients and medical records from the database

        patients = Patient.objects.all()

        medical_records = MedicalRecord.objects.all()


        # Pass data to the template for rendering
```

```
    context = {
        'patients':
        patients,
        'medical_records': medical_records
    }

    return render(request, 'home.html', context)
```

UTILITIES.PY

```
import re
from datetime import datetime


def validate_date(date_string):
    """
     Validates a date string in the format 'MM/DD/YYYY' and returns a boolean
value indicating
    whether or not the date is valid.
    """
    try:
        datetime.strptime(date_string,
        '%m/%d/%Y') return True
    except ValueError:
        return False


def validate_phone_number(phone_number):
    """
        Validates a phone number string and returns a boolean value indicating whether
or not the
    phone number is valid.
    """
    pattern = re.compile(r'^\d{10}$')
    return pattern.match(phone_number) is not None


def format_date(date_string):
    """
    Converts a date string in the format 'MM/DD/YYYY' to a datetime object.
    """
    return datetime.strptime(date_string, '%m/%d/%Y')


def format_phone_number(phone_number):
    """
    Formats a phone number string to the format '(XXX) XXX-XXXX'.
    """
```

```
            return    '({})    {}-{}'.format(phone_number[:3],
phone_number[3:6], phone_number[6:])


    def calculate_age(date_of_birth):
        """
        Calculates the age of a person given their date of birth.
        """
        today = datetime.today()
        birth_date = datetime.strptime(date_of_birth, '%m/%d/%Y')
            age   =   today.year  -  birth_date.year  -  ((today.month,  today.day)  <
(birth_date.month, birth_date.day))
        return age


    def generate_unique_id(prefix=", suffix=", length=8):
        """
        Generates a unique ID string with a specified prefix, suffix, and length.
        """
        timestamp = datetime.now().strftime('%Y%m%d%H%M%S')
          random_str = ".join(random.choices(string.ascii_uppercase + string.digits, k=length))
        return '{}{}{}'.format(prefix, timestamp + random_str, suffix
```

CHAPTER 6

SYSTEM

TESTING

6.1 TEST CASES & REPORTS

TEST CASE ID	**TESTCASE/ ACTION TO BE PERFORMED**	**EXPECTED RESULT**	**ACTUAL RESULT**	**PASS/ FAIL**
1.	Selecting “Write data”button	Write data onto nfc tag	Write data onto nfc tag	Pass
2.	Selecting “Edit data”	Edit data and Write data onto nfc tag	Edit data and Write data onto nfc tag	Pass
3.	Tap tag on NFC reader	share medical data	share medical data	Pass
4.	open Heka web app	Display shared medical data	Display shared medical data	Pass

CHAPTER 7

7.1 CONCLUSION

In conclusion, our automatic medical form filling application offers a revolutionary solution to the challenges faced in manual medical data entry. With the use of NFC technology, the process of data sharing is accelerated, reducing the time and effort required to fill out medical forms.

By eliminating human errors associated with manual data entry, we ensure that medical records are more accurate and reliable. Our application seamlessly integrates with existing electronic medical record systems, providing enhanced security and protection of sensitive medical information. By simplifying the process of data sharing, we empower medical professionals to provide better patient care and improve overall healthcare outcomes.

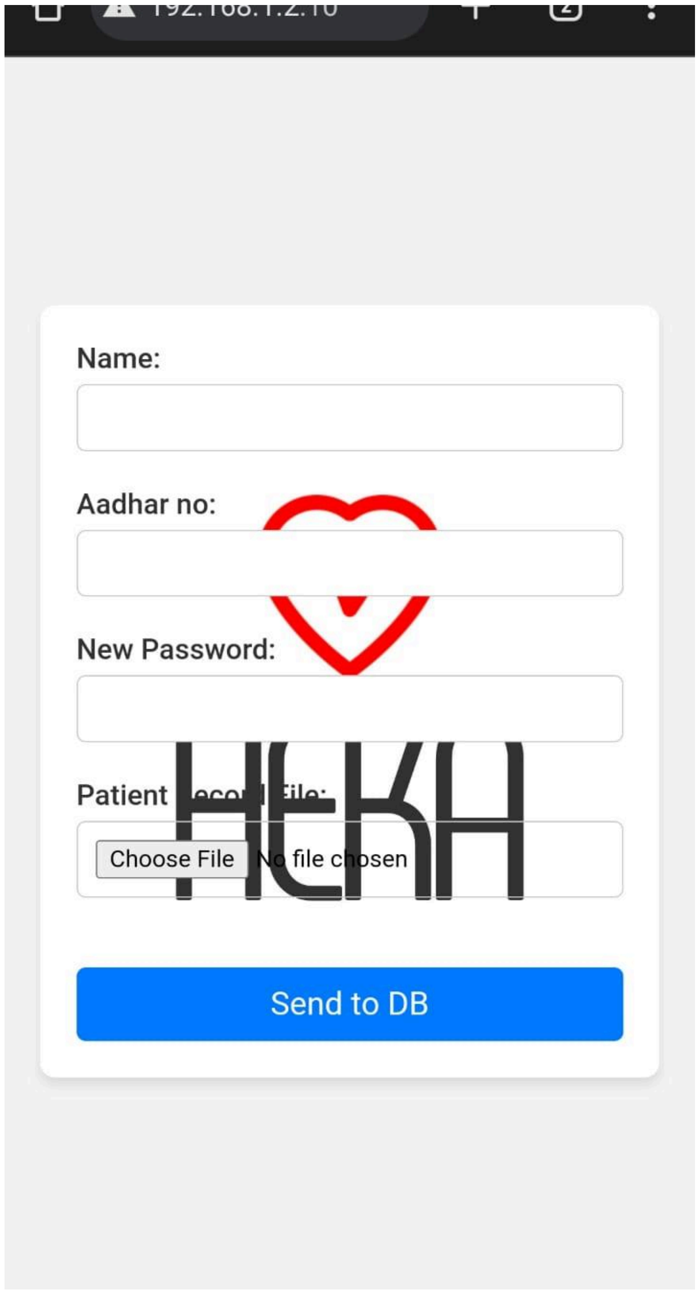
7.2 FUTURE ENHANCEMENTS

1. Make a system that updates the medical records on the EHR everytime.
2. Provide storage by using blockchain technology.
3. Use fingerprint and face id for more security.
4. Integrate with patient legal ID such as adhaar card or other such proofs.
5. Enable access across all platforms by using servers.

The above mentioned are the future enhancements that can be done to make this project much more dynamic.

CHAPTER 8

APPENDICES



192.168.1.2:10

Name:

Aadhar no:

New Password:

Patient Record File:

Choose File No file chosen

Send to DB

Fig 8.1 :Heka mobile application form filling

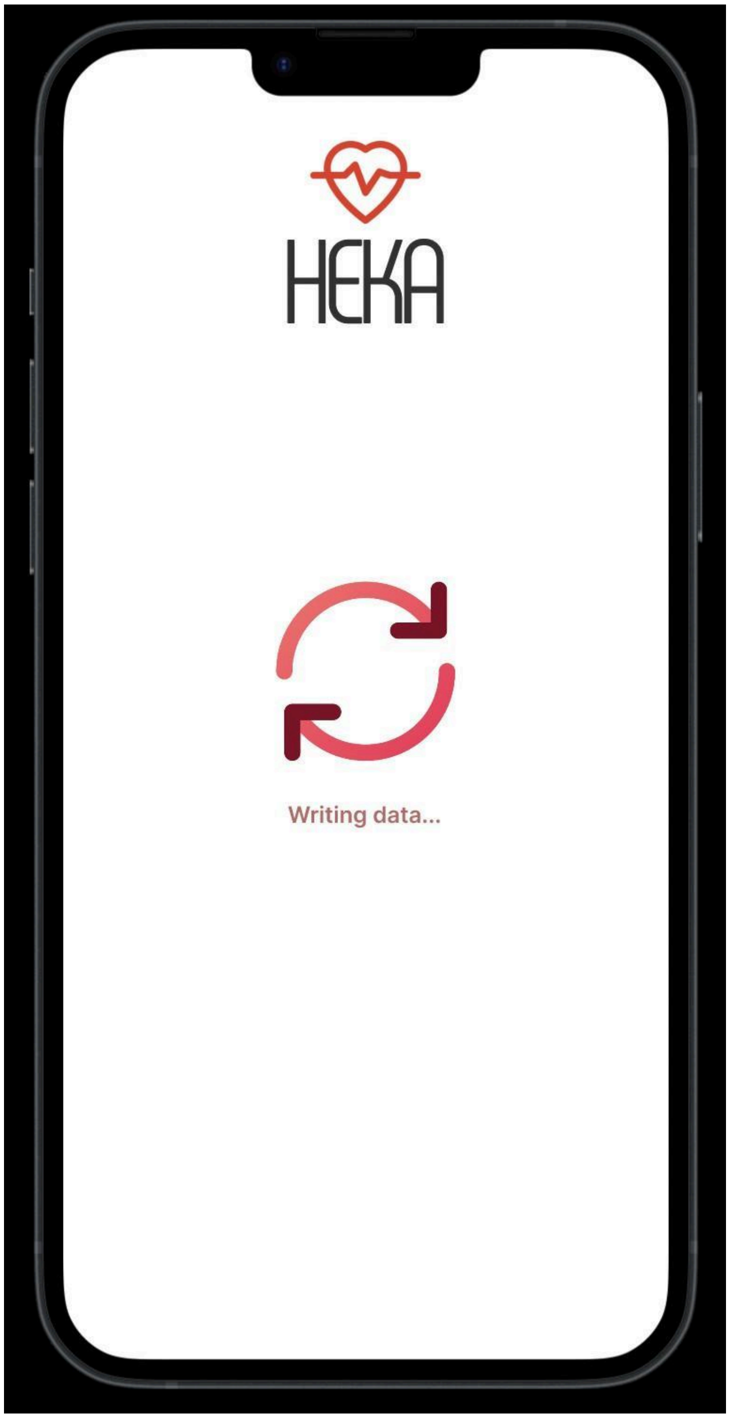
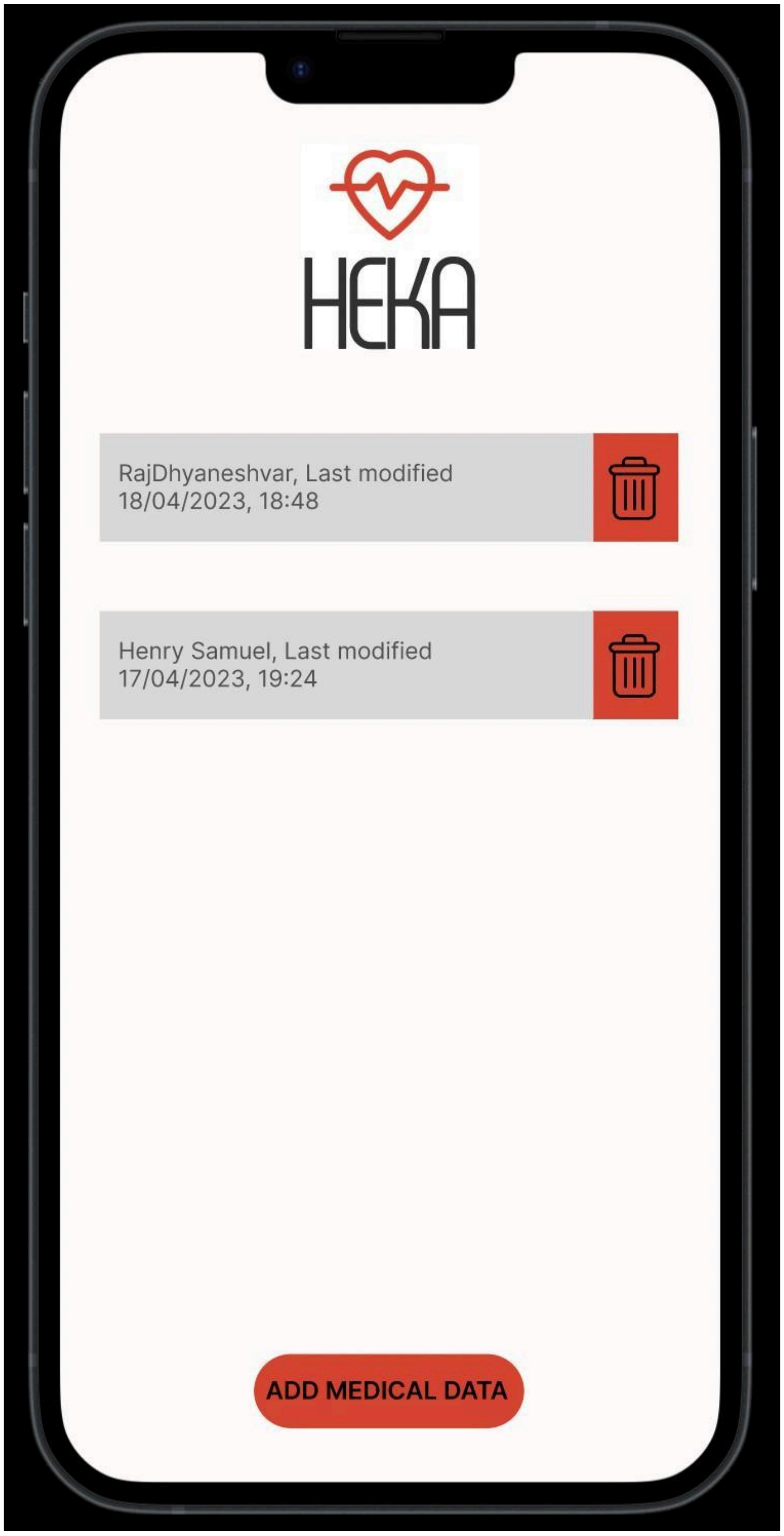


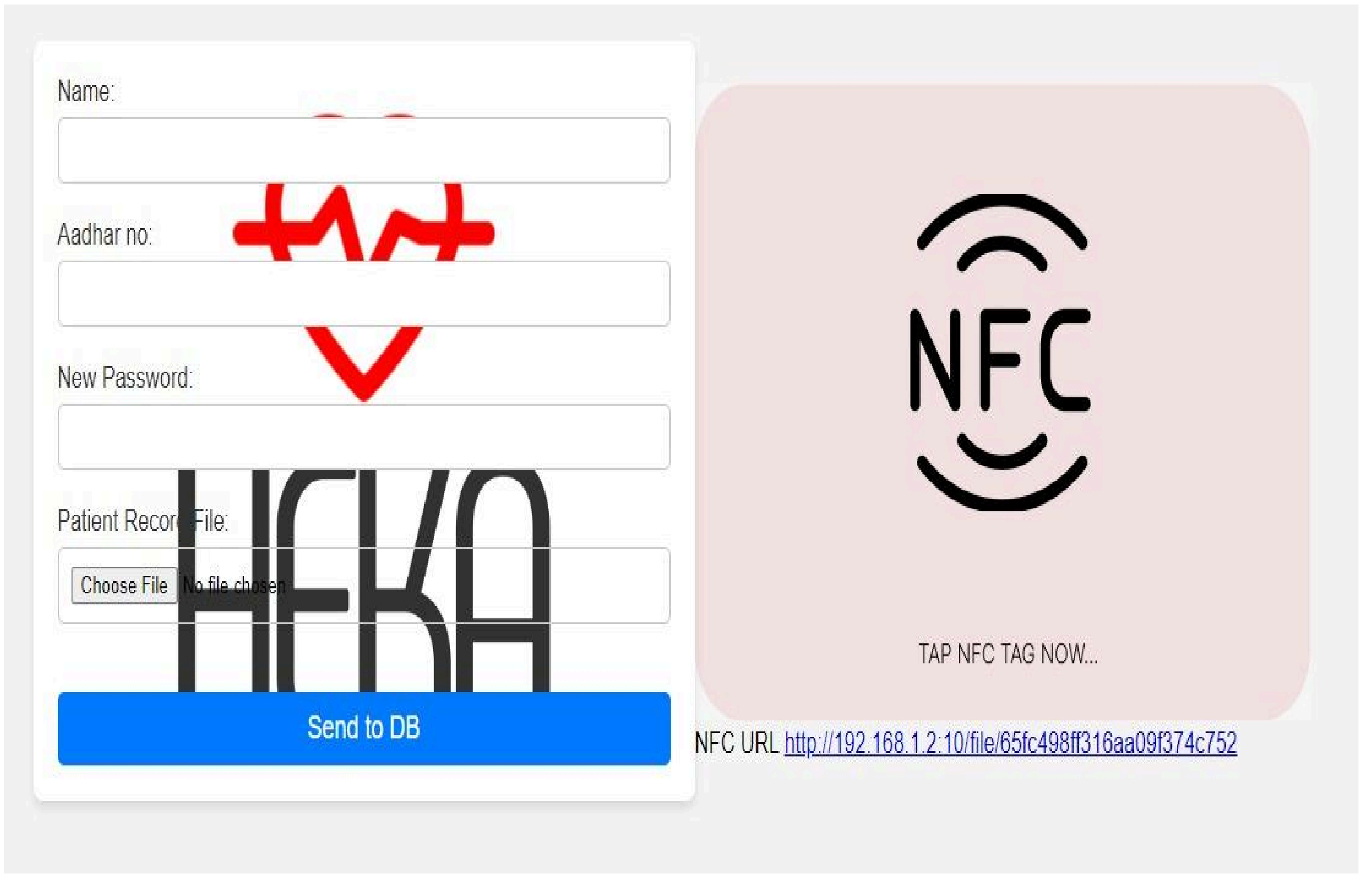
Fig 8.2:Heka Mobile application Data Writing(loading page)



RajDhyaneshvar, Last modified 18/04/2023, 18:48

Henry Samuel, Last modified 17/04/2023, 19:24

Fig 8.3:Heka Mobile application saved medical forms page



Name:

Aadhar no:

New Password:

Patient Record File:

Choose File No file chosen

NFC URL http://192.168.1.2:10/file/65fc498ff316aa09f374c752

Fig 8.4: Heka Web App login page

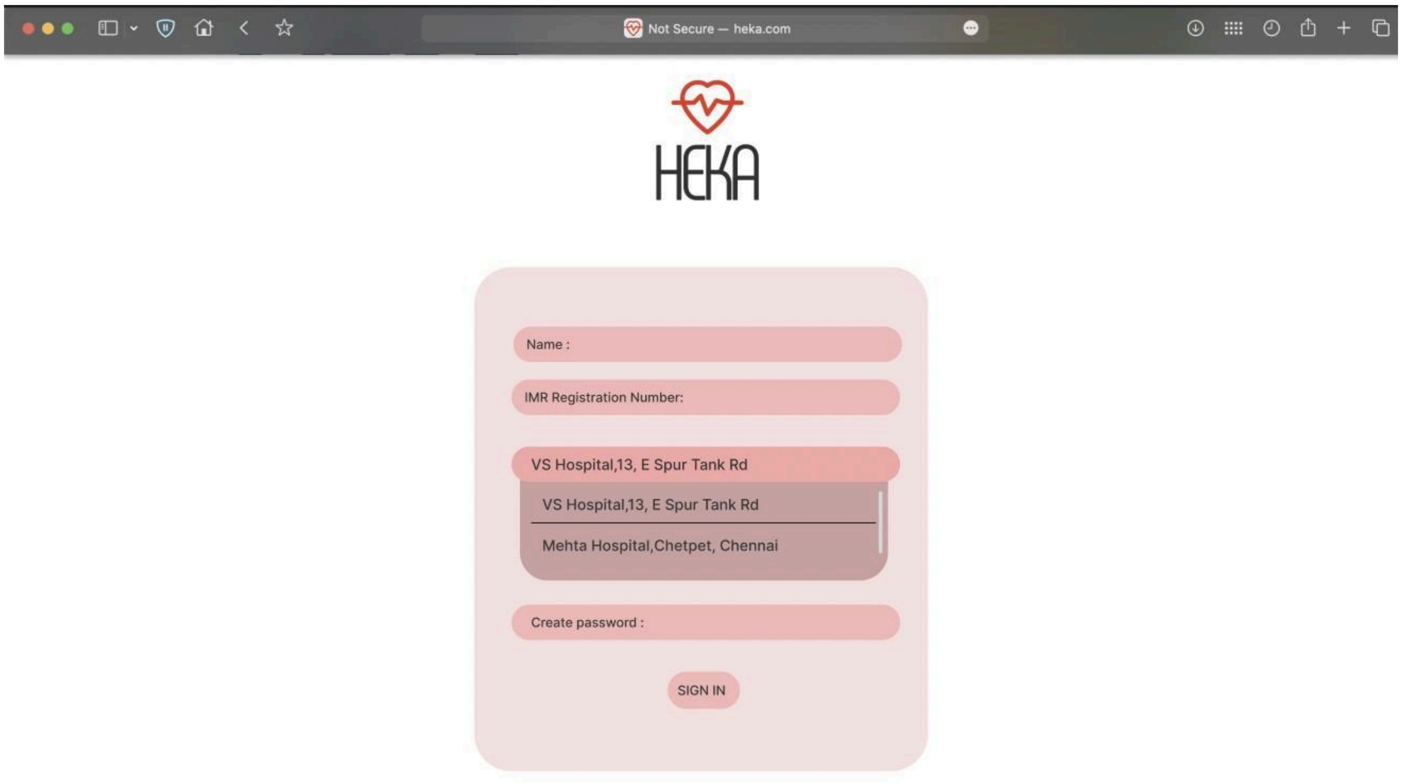

Not Secure — heka.com
HEKA
Name :
IMR Registration Number:
VS Hospital,13, E Spur Tank Rd
VS Hospital,13, E Spur Tank Rd
Mehta Hospital,Chetpet, Chennai
Create password :
SIGN IN

Fig 8.5: Heka Web App Sign Up page

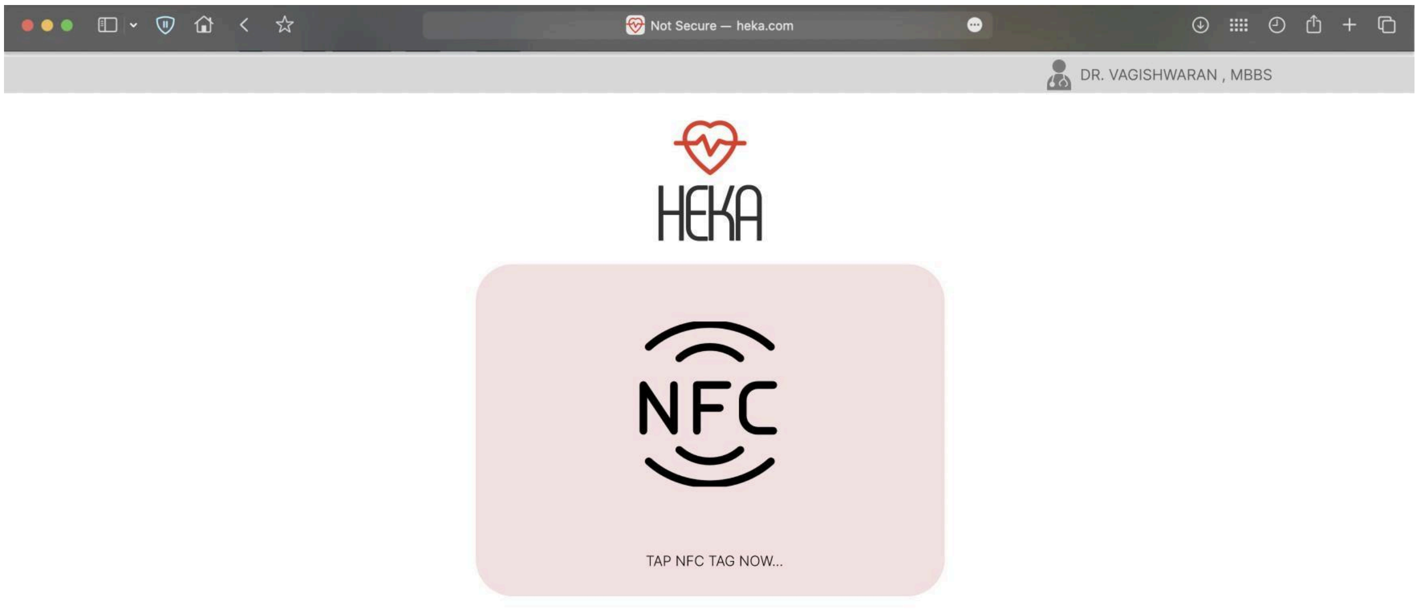
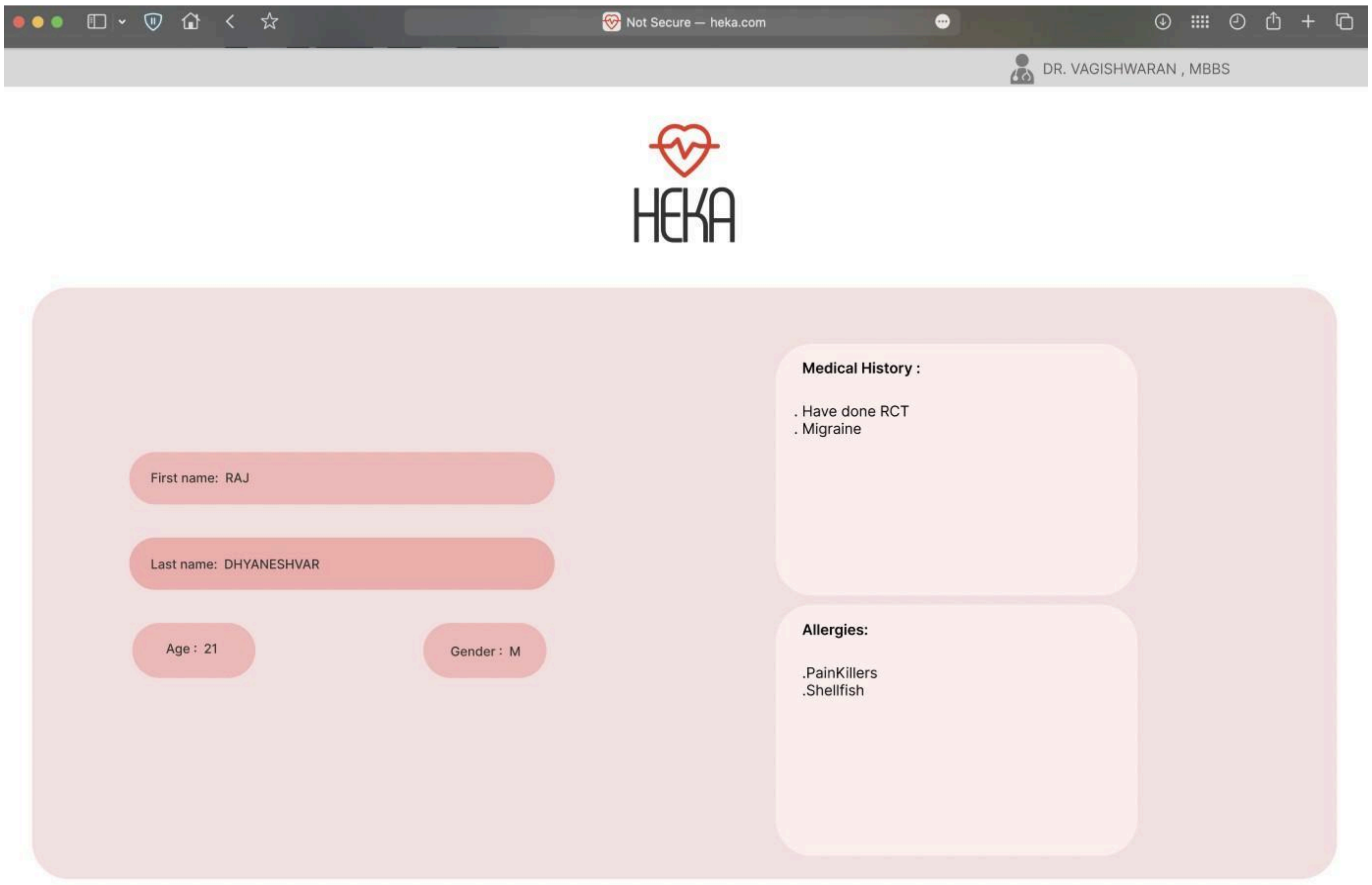


Fig 8.6:Heka Web app NFC receiving page



First name: RAJ

Last name: DHYANESHVAR

Age : 21

Gender : M

Medical History :

. Have done RCT
. Migraine

Allergies:

.PainKillers
.Shellfish

Fig 8.7:Heka Web app Medical data display page

CHAPTER 9

REFERENCES

www.w3schools.com

www.reddit.com

www.codegrepper.com

www.geeksforgeeks.org

www.stackoverflow.com

[1] W. Burns, Who launches taskforce to fight counterfeit drugs, Bulletin of the World Health Organization 84 (2006) 689–690.

[2] J. Sambira, Counterfeit drugs raise africa's temperature, Africa Renewal 27 (1) (2013) 5–7.

[3] WHO, Substandard, spurious, falsely labeled, falsified and counterfeit (ssffc) medical products (2016).

[4] H. Cheung, S. Choi, Implementation issues in rfid-based anti-counterfeiting systems, Computers in Industry 62 (7) (2011) 708–718.

[5] S. Choi, C. Poon, An rfid-based anti-counterfeiting system, IAENG International Journal of Computer Science (2008).

[6] M. Wang, Z. Yan, A survey on security in d2d communications, Mobile Networks and Applications 22 (2) (2017) 195–208.

[7] A. Bodhani, New ways to pay, Engineering & Technology 8 (7) (2013) 32–35.

[8] N. M. Smith, C. Cahill, Continuous multi-factor authentication, uS Patent 9,705,869 (Jul. 11 2017).

[9] Q. Z. Sheng, S. Zeadally, A. Mitrokotsa, Z. Maamar, Rfid technology, systems, and applications, Journal of Network and Computer Applications 34 (ARTICLE) (2011) 797–798.

[10] D. He, S. Zeadally, Authentication protocol for an ambient assisted living system, IEEE Communications Magazine 53 (1) (2015) 71–77.

[11] D. He, S. Zeadally, N. Kumar, J.-H. Lee, Anonymous authentication for wireless body area networks with provable security, IEEE Systems Journal 11 (4) (2016) 2590–2601.

[12] D. He, N. Kumar, N. Chilamkurti, A secure temporal-credential-based mutual authentication and key agreement scheme with pseudo identity for wireless sensor networks, Information Sciences 321 (2015) 263–277.