Streamlining Healthcare Data Management: An In-Depth Exploration of a Spring Boot Based Embedded Health Record Keeper Application

Author 1: **Suriya Kumari A M**, Post Graduate Student, Embedded Systems , Hindusthan College of Engineering and Technology,Coimbatore,

Co Author 2: **Ms Nandhini B, M.E., Ph.D..,** Assistant Professor, Department of Electrical and Electronics Engineering, Hindusthan College of Engineering and Technology, Coimbatore,

Co Author 3**: Dr. K Mathan M.E. Ph.D**., Professor, Department of Electrical and Electronics Engineering, Hindusthan College of Engineering and Technology, Coimbatore.

ABSTRACT

This journal explores the development of the Health Record-Keeping System, designed to address the complexities of modern healthcare record management. By integrating advanced technologies such as RFID and fingerprint security with PostgreSQL databases and Spring Boot, the system provides a secure and user-friendly platform for managing an individual's comprehensive medical history from birth onward. The primary aim is to enhance data accessibility, ensure security, and empower users while meeting the needs of healthcare facilities and patients. Over the course of a person's life, healthcare systems must manage extensive and secure medical records. In order to overcome these issues, this article presents the Health Recordkeeper Application, a unique system that combines embedded system implementations like RFID and fingerprint security with PostgreSQL databases utilising Spring Boot. The principal aim of this programme is to facilitate the maintenance of an individual's entire medical history, including information about vaccinations, diagnostic tests, medication records, and other related facts, from the time of birth to the present. Within healthcare facilities, the system allows users with unique ID cards that support RFID technology to securely add and view medical records.

CHAPTER 1 INTRODUCTION

In today's digital era, the Health Recordkeeper application emerges as a vital solution, aiming to revolutionize the management of individual medical records. This application seeks to provide a consolidated and secure platform for users to maintain a comprehensive medical history from birth to the present day. By offering an intuitive interface for data entry and retrieval and integrating advanced security measures like RFID and fingerprint authentication, the Health Recordkeeper ensures accessibility and protection of sensitive medical information. One of the biggest challenges facing modern healthcare systems is maintaining secure, comprehensive medical records. The resilience and accessibility required to preserve a patient's whole medical history from the time of birth throughout their lifetime are sometimes lacking in traditional techniques. The necessity for an inventive solution is highlighted by fragmented records, accessibility problems, and security concerns.

1.1 PROBLEM STATEMENT

The Health Recordkeeper Application's goals are to transform healthcare record management by bringing in a safe, user-friendly framework. Enabling people to easily preserve and retrieve their extensive medical histories is the major goal. Utilising cutting-edge technologies like embedded systems like RFID and fingerprint security, PostgreSQL databases for data storage, and Spring Boot for reliable application development, the application aims to handle the following important aspects.

CHAPTER 2 LITERATURE REVIEW

1.A. Gharat and G. Phadke, "*Smart Medical Health Card for Hospital Management,*" 2022 2nd Asian Conference on Innovation in Technology (ASIANCON), Ravet, India, 2022.

The use of information technology has become a common place in healthcare. The increase in the number of patients in need of ongoing care is a responsibility for medical professionals. Today the patient monitoring system and wireless medical equipment are used to monitor patients, but nevertheless patients have to stay inside the wireless coverage area. This real-time information presents an ongoing challenge to the emergency community. In these emergencies as unconscious and unaccompanied patients, giving emergency physicians or doctor an accurate and updated medical history of a patient can be the difference between his/her life and death[1]. In this smart health card project patient we are tracking and monitoring of patient history. The system monitors patient’s vital signs such as past history, blood group, past medicine history bills. The patient data is stored in one RFID card. In this smart card project we are using RFID technology. This RFID technology is currently very useful in health care system due to its increased performance, lower cost, high efficiency and easy-to-use skills. Physicians can track all information about their patients using a dedicated system or model. Also, they can ask using an established web server. The integration of smart card system used in existing automation system should be important part of the hospital management as far as it benefits both the hospital management and patients.

2. G. Paolini, D. Masotti and A. Costanzo, "*RFID Reader and Wearable Tags for Smart Health Applications,*" 2021 XXXIVth General Assembly and Scientific Symposium of the International Union of Radio Science (URSI GASS), Rome, Italy, 2021.

This work presents an RFID system working in the 2.4 GHz band able track and tridimensional locate people moving in indoor environments equipped with active tags realized on wearable and flexible textile materials. Thanks to the beam steering capabilities of the custom-designed reader both in the azimuth and in the elevation plane, it is also possible to detect potential falls that could happen to elderly or impaired people living alone and send an alarm to the family, the caregivers, or the medical team to point out the problem. Moreover, a series of antennas on denim substrates has been realized and exploited in combination with the circuitry of the tag to enhance the comfort and psychological acceptability of the technology for the users. For what concerns 2-D localization capabilities, an average error of about 20 cm has been detected in a room larger than 30 m2, whereas an error of only 8.5 cm has been noticed along the vertical axis, widely sufficient to correctly detect the upright status of tagged people.

3. Korkmaz, Ilker & Atay, Coskun & Kyparisis, George. (2010).

A mobile patient monitoring system using RFID. In the last decade, Radio Frequency Identification (RFID) has become popular in so many fields from military to industry applications. RFID tags have been embedded into many various products especially in logistics sector. A tag stores individual information of its attached object and an RFID reader communicates with the tag in radio frequencies to identify the object. This object to be monitored may also be a human. In our work, RFID technology is applied in health care systems. The system supports wireless mobile communication between the RFID tags and readers. Each patient available in the system is inherently mobile and wears a bracelet integrated with a unique tag, and the readers are mobile PDA devices each including a wireless RFID reader card. The proposed application can be used to identify and monitor the patients.

4. Rahman, Mohammad. (2018). ***Using RFID Technology for Managing Patient Medical File*****.**

American Journal of Computer Science and Information Technology. 06. 10.21767/2349-3917.100024. Radio Frequency Identification (RFID) technology usingradio frequencies for identifying objects in inventory, tracingcars in parking areas and identifying books in libraries. RFIDtechnology also can be used in hospital medical record formanaging patient medical Files. RFID is a very populartechnology for identifying an item or object from nearestdistance. RFID also can be integrated with Computer basedmedical record file tracking application system. RFIDtechnology can be used to quickly and easily tracking andidentifying a patient medical record file. RFID can replacethe existing barcode system of identifying a patientfile. Thisresearch was done for managing patient medical filse ingovt. hospitals in Saudi Arabia.

5. **R. S. Sangwan, R. G. Qiu and D. Jessen, "*Using RFID tags for tracking patients, charts and medical equipment within an integrated health delivery network*," Proceedings. 2005 IEEE Networking, Sensing and Control, 2005., Tucson, AZ, USA, 2005.**

Tracking patients, charts and equipment in hospitals and across integrated health delivery networks is mostly done manually on white boards or manual entry within health information systems. Some health delivery networks have adopted systems using a combination of infrared and radio frequency (RF) technology to help manage the tracking process. This paper proposes an approach that can improve the operational efficiency of a health delivery network by automating this process through the use of RF identification (RFID) tags. RFID tags are low power communication devices that can be embedded in a patient's ID bracelet, inside a patient chart and medical equipment, and can help track their location and status.

CHAPTER 3 EXISTING SYSTEM

3.1 INTRODUCTION

In assessing the landscape of healthcare record management systems, several existing solutions provide parallels to the Health Recordkeeper Application. Systems like [mention specific examples or types of systems], known for their emphasis on [mention key features such as security, usability, or integration], have paved the way in revolutionizing healthcare data management. These systems highlight the industry's ongoing efforts to address challenges such as [mention common challenges like data security, accessibility, or interoperability] through innovative technological solutions.

3.2 DISADVANTANGES

Despite advancements in healthcare record management systems, several existing solutions exhibit notable drawbacks. For instance, some systems encounter challenges related to data fragmentation, where medical records remain siloed across different departments or healthcare facilities. This fragmentation can hinder healthcare providers' ability to access comprehensive patient information swiftly, potentially impacting treatment decisions and patient care. Moreover, interoperability issues are prevalent among existing systems, particularly those using older technologies or proprietary formats. These issues often arise from incompatible data standards and protocols, making it difficult to seamlessly exchange patient information between different healthcare providers or systems. As a result, patients may experience delays in receiving coordinated care or encounter discrepancies in their medical records across healthcare settings.

CHAPTER 4 METHODOLOGY

4.1 INTRODUCTION

Healthcare record management is a multifaceted domain, requiring a nuanced and comprehensive methodology to ensure the development of a system that aligns with the evolving needs of patients and healthcare professionals. The goal was to create a Health Record-Keeping System that not only centralizes medical data but also empowers individuals to actively manage their health records from birth onwards. The methodology was structured to encompass critical phases, from requirements gathering to continuous improvement, with a keen focus on security, usability, and adaptability to future healthcare trends.

TECHNOLOGICAL FRAMEWORK

In order to guarantee effectiveness, security, and scalability, the Health Recordkeeper Application was created utilising a contemporary technology stack. The principal technologies employed comprise:

Spring Boot: Selected for its enterprise-level application interoperability, robustness, and quick application development capabilities. Because of its modular design, development was expedited and several capabilities could be implemented more easily.

Databases built with PostgreSQL: Selected for its scalability, extensibility, and ability to handle intricate datastructures. For healthcare records, PostgreSQL guarantees effective data storage and retrieval while preserving data integrity.

Embedded Systems: Integrated to strengthen the security of the application (RFID and Fingerprint Security). A further degree of biometric security is added via fingerprint authentication, while RFID technology allows safe user identification and access to medical records.

4.2 SYSTEM ARCHITECTURE

The application uses a tiered architecture that consists of:

Presentation Layer: Employing responsive web design concepts, this layer implements user interfaces for administrators and patients, allowing access from a variety of devices.

Application Layer: Uses Spring Boot to handle database interactions, authentication protocols, and business logic for the application.

Data Layer: Safely stores and manages medical records using PostgreSQL databases.

4.3 DEVELOPMENT PROCESS

An agile and iterative technique was used throughout the development process to enable ongoing feedback, modification, and enhancement.

Design and Prototyping: Before beginning a full-scale development project, wireframes and prototypes are created to show the application's functionality, user flows, and layout.

Development Process: The process was developed using an agile, iterative technique that allowed for constant improvement, modification, and feedback. Prerequisites Collaborating with healthcare professionals and prospective users, gathering requirements for security and determining necessary features.

Authorization and Authentication: To enable patients to view their records, the system employs strong authentication procedures with RFID-enabled unique IDs. Patients and administrators have several access levels, guaranteeing limited access and role-based right modification.

Installation and Upkeep: Installing the programme on a safe server, continuously checking its functionality, applying security updates, and offering assistance to users.

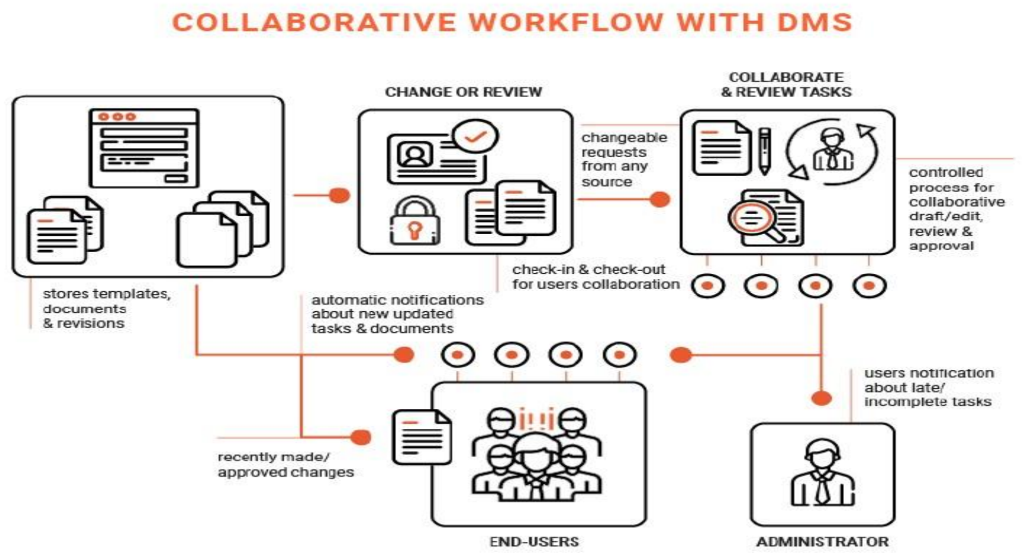

COLLABORATIVE WORKFLOW WITH DMS
CHANGE OR REVIEW
COLLABORATE & REVIEW TASKS
changeable requests from any source
controlled process for collaborative draft/edit, review & approval
check-in & check-out for users collaboration
stores templates, documents & revisions
automatic notifications about new updated tasks & documents
users notification about late/ incomplete tasks
recently made/ approved changes
END-USERS
ADMINISTRATOR

Fig 3.1 workflow with documents

CHAPTER 4 APPLICATION DESIGN AND FEATURES

4.1 USER ROLES AND ACCESS CONTROL

Patient Access: People are given special IDs that are RFID-enabled so they can safely access their medical records. In order to maintain data integrity and compliance, patients are only allowed to access records; they are not allowed to alter or remove them.

Administrator Privileges: With all-encompassing access permissions, administrators can oversee system functionality, manage user accounts, and carry out any necessary changes or alterations.

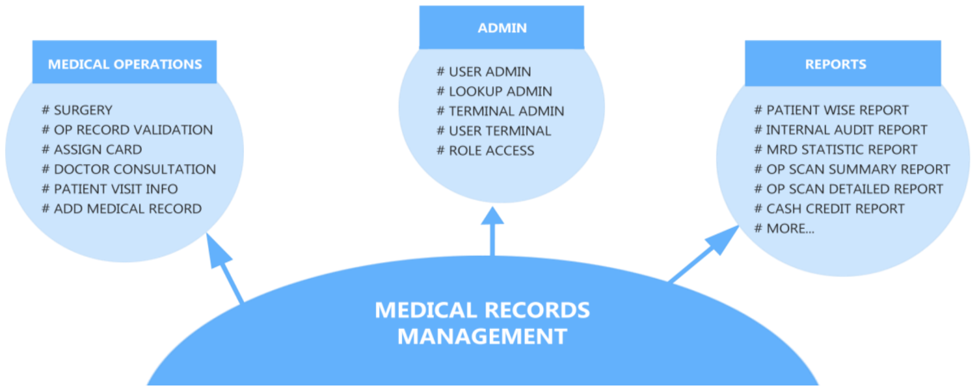


Fig 4.1 Medical Records Management

4.2 FUNCTIONALITIES

Medical Record Management: Presented chronologically from birth to the present, users can examine a complete medical history including immunisations, diagnostic tests, prescription drugs, and therapies.

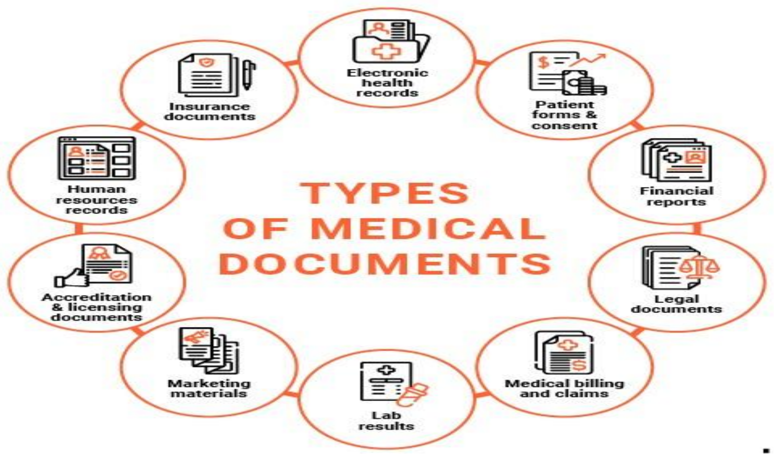


Fig 4.2 Types of Medical Documents

Addition of Records: Patients are able to upload fresh medical records, including information on current treatments, prescription drugs, and any operations they have had, to their profile.

Restrictions on Editing: In order to preserve a verifiable and accurate medical history, patients are prohibited from making changes to or erasing already-existing information.

4.3 USER INTERFACE

Intuitive Dashboard: Easily Navigate to specific categories such as vaccinations, diagnoses, and prescriptions with the application's intuitive dashboard, which showcases an overview of medical records.

Responsive Design: The UI is made to be easily accessed from a variety of devices, such as smartphones, tablets, laptops, and desktop computers.

4.4 SECURITY MEASURES

Fingerprint Authentication: Adding an extra degree of protection to biometric security via fingerprint authentication improves data security and thwarts unwanted access.

RFID Integration: By enabling secure user identification through seamless integration, RFID technology guarantees that only persons with the proper authorization can access their particular medical records.

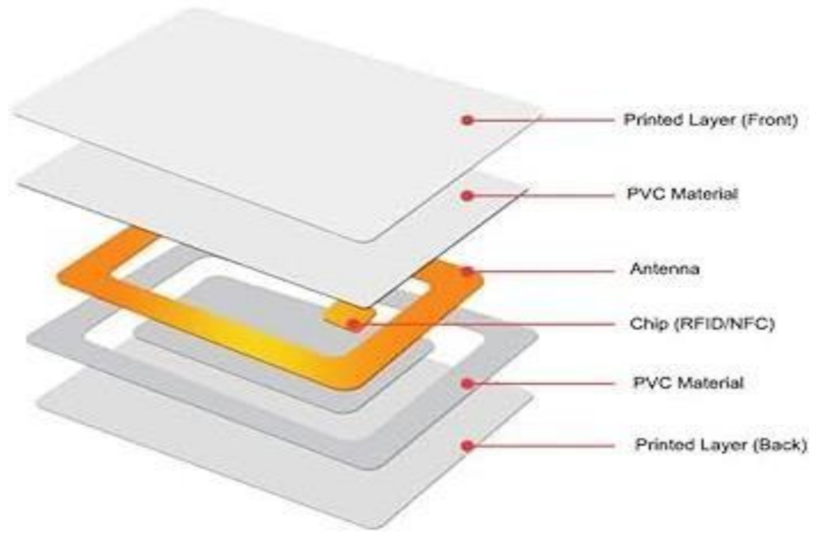


Fig 4.3 RFID Card

4.5 AUDIT TRAIL

All user activity, including as logins, access attempts, and modifications (if approved by administrators), are recorded by the programme, which keeps an audit trail. Accountability and traceability of system interactions are guaranteed by this trail. Document security is a critical aspect of any system dealing with sensitive information, especially in the context of healthcare record management. In the Health Record-Keeping System, the approach to document security is comprehensive, incorporating multiple layers of protection to ensure the confidentiality, integrity, and accessibility of medical records. This section provides an in-depth elaboration on the document security measures implemented within the system.

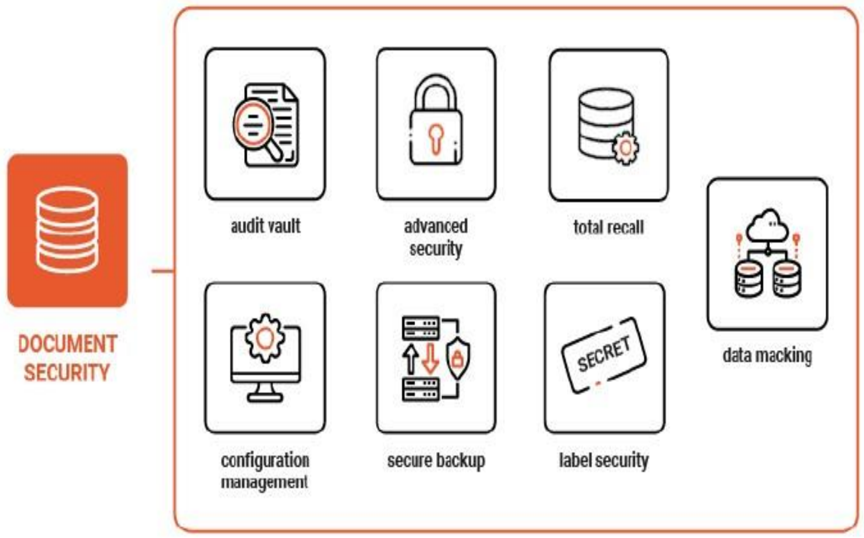


Fig 4.4 Document Security

4.6 USER EXPERIENCE FOCUS

With a focus on the user experience, the programme was put through a rigorous usability testing process to make sure that it was easy to use, had clear navigation, and was simple enough for administrators and patients alike.

4.7 FLOW DIAGRAM OF HEALTH RECORD KEEPER SYSTEM

Health administrators, who have elevated access rights, can oversee the system, manage user accounts, and address any issues that may arise.Administrators play a crucial role in maintaining the overall integrity and functionality of the system. They have the authority to intervene in case of user-related concerns or system anomalies.

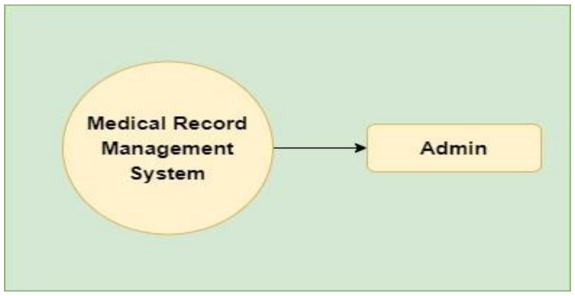


Fig 4.5 Flow Diagram of Admin

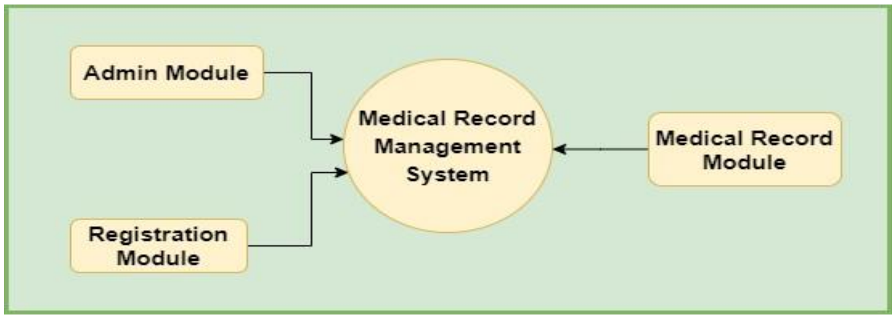


Fig 4.6 Flow Diagram of Admin Module

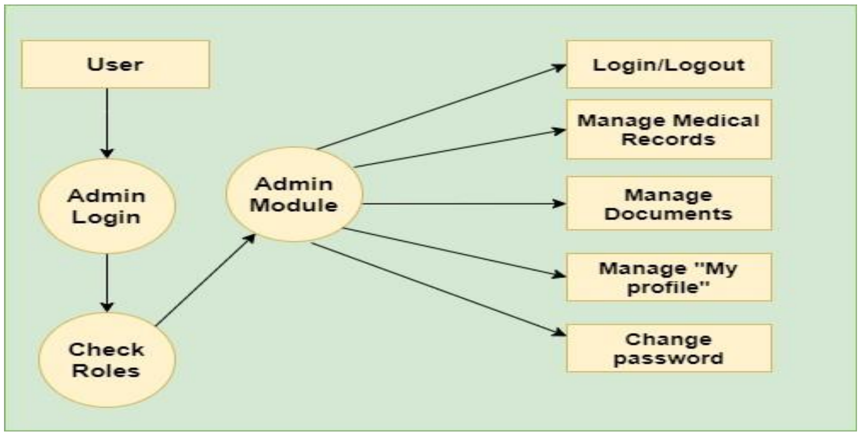


Fig 4.7 Flow Diagram of Admin Module with roles

TOOLS AND TECHNOLOGIES USED

Front-End: JSP, Html, CSS, JS.

Server-side: Spring Boot.

Back-end: Postgres SQL, Hibernate.

Server: Tomcat 8.5.

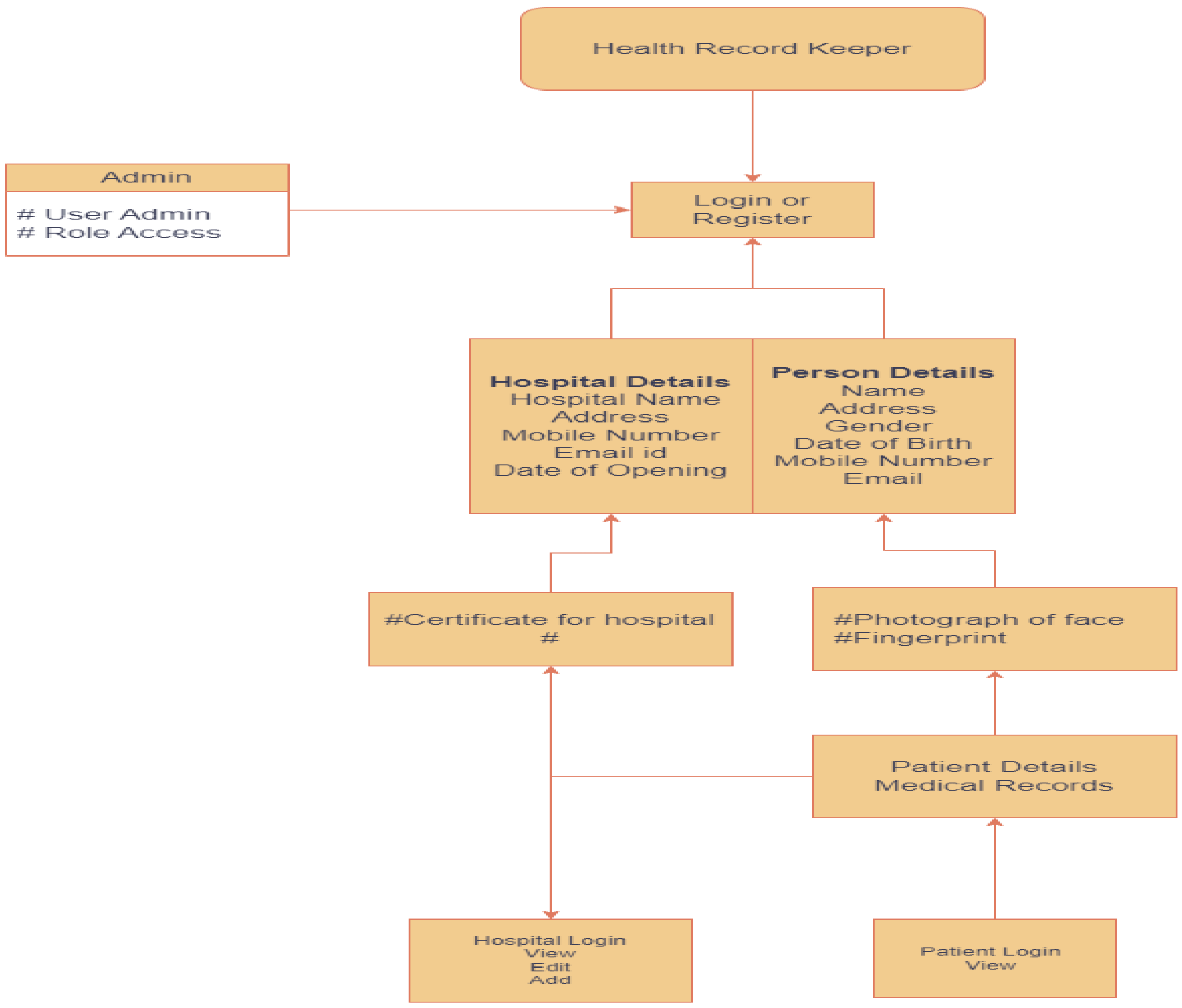


Fig 4.8 Flow Diagram of Health Record Keeper Application

CHAPTER 5 IMPLEMENTATION DETAILS

5.1 User Registration and Unique ID Generation

Users undergo a registration process to create accounts and access the system. Personal information such as name and contact details is collected, followed by identity verification (e.g., email or mobile verification). Upon verification, user accounts are created, and each user is issued a unique RFID-enabled ID card for authentication within healthcare facilities. Additionally, a numeric identifier may be assigned for remote access.

5.2 Security Protocols

- **Encryption Mechanisms**: Protect sensitive information using industry-standard encryption, ensuring data remains unreadable without decryption keys.

- **Data Integrity Checks**: Use hashing algorithms to detect and prevent unauthorized modifications.
- **Secure Data Transfer**: Employ HTTPS to encrypt data in transit, safeguarding against interception.
- **RFID Authentication**: Unique RFID-enabled IDs ensure only authorized users access records.
- **Fingerprint Authentication**: Biometric verification adds an extra layer of security, particularly for administrators.
- **Audit Trail**: Logs significant actions, allowing administrators to monitor and trace user activities..

5.3 Database Structure

The database schema is designed to support various medical record formats, categorizing data like diagnoses, treatments, and prescriptions. Using a relational structure enables efficient searching and retrieval based on criteria such as date or healthcare provider.

5.4 Integration of Embedded Systems

- **RFID Integration**: Enables secure user authentication and access control.
- **Fingerprint Authentication**: Adds biometric verification for enhanced security.

5.5 User Interface and Application Deployment

The application features a responsive design for accessibility across devices. It is deployed on a secure server with ongoing monitoring for updates and performance optimization.

SOURCE CODE

- **User Entity**: This entity typically stores information about users who interact with the system. It might include fields like username, password, email, etc., depending on your authentication and user management requirements.
- **MedicalRecord Entity**: This entity represents individual medical records for each user. It could include fields such as recordId, userId (to associate records with users), diagnosis, treatments, vaccines, prescriptionDrugs, etc.
- **Role Entity**: If you have role-based access control (RBAC), this entity defines various roles users can have (e.g., ADMIN, PATIENT, etc.).
- **AuditLog Entity**: This entity tracks important actions within the application for auditing purposes. It might include fields like logId, userId, action, timestamp, etc.
- **Settings Entity**: If your application includes configurable settings (e.g., security settings, system preferences), you might have a Settings entity to store and manage these.
- **Feedback Entity**: If you want to capture user feedback or suggestions, a Feedback entity might include fields like feedbackId, userId, feedbackMessage, timestamp, etc.

ENTITY STRUCTURE

```
@Entity
```

```
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false, unique = true)
    private String username;
    @Column(nullable = false)
    private String password;
    // Other fields like email, roles (if not using separate role entity)
    // Getters and setters
}
@Entity
@Table(name = "medical_records")
public class MedicalRecord {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long recordId;
    @ManyToOne
    @JoinColumn(name = "user_id", nullable = false)
    private User user;
    @Column(nullable = false)
    private String diagnosis;
    @Column(nullable = false)
    private String treatments;
    // Other fields like vaccines, prescriptionDrugs
    // Getters and setters
}
@Entity
@Table(name = "roles")
public class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long roleId;
    @Column(nullable = false, unique = true)
    private String roleName;
    // Getters and setters
}
@Entity
@Table(name = "audit_logs")
public class AuditLog {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long logId;
    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;
    @Column(nullable = false)
    private String action;
    @Column(nullable = false)
    private LocalDateTime timestamp;
    // Getters and setters
}
```

REPOSITORY CLASS

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.Optional;
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByUsername(String username);
    Optional<User> findByEmail(String email);
    boolean existsByUsername(String username);
    boolean existsByEmail(String email);
}
```

METHODS:

- findByUsername(String username): Finds a user by their username. Returns an Optional<User> to handle the case where the user might not be found.
- findByEmail(String email): Finds a user by their email. Also returns an Optional<User>.
- existsByUsername(String username): Checks if a user with the given username exists. Returns a boolean.
- existsByEmail(String email): Checks if a user with the given email exists. Returns a boolean.

SERVICE CLASS

```
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.Optional;

@Service

public class UserService {

    @Autowired

    private UserRepository userRepository;

    @Transactional(readOnly = true)

    public Optional<User> findByUsername(String username) {

        return userRepository.findByUsername(username);

    }

    @Transactional(readOnly = true)

    public Optional<User> findByEmail(String email) {

        return userRepository.findByEmail(email);

    }

    @Transactional

    public User saveUser(User user) {

        return userRepository.save(user);
    }

    @Transactional(readOnly = true)

    public boolean existsByUsername(String username) {

        return userRepository.existsByUsername(username);

    }

    @Transactional(readOnly = true)

    public boolean existsByEmail(String email) {

        return userRepository.existsByEmail(email);

    }
```

- **Annotations**:
  - @Service: Indicates that this is a service class.
  - @Transactional: Ensures that the methods run within a transaction. The readOnly attribute is used for read-only operations to optimize performance.

- **Methods**:
  - findByUsername(String username): Uses the repository to find a user by username.
  - findByEmail(String email): Uses the repository to find a user by email.
  - saveUser(User user): Saves a user entity to the database.
  - existsByUsername(String username): Checks if a user with the given username exists.
  - existsByEmail(String email): Checks if a user with the given email exists.

USER CONTROLLER

```
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.*;

import java.util.Optional;

@RestController

@RequestMapping("/api/users")

public class UserController {
```

```
@Autowired
private UserService userService;
@GetMapping("/{id}")
public ResponseEntity<User> getUserById(@PathVariable Long id) {
    Optional<User> user = userService.findById(id);
    return user.map(ResponseEntity::ok)
            .orElseGet(() -> ResponseEntity.notFound().build());
}
@GetMapping("/username/{username}")
public ResponseEntity<User> getUserByUsername(@PathVariable String username) {
    Optional<User> user = userService.findByUsername(username);
    return user.map(ResponseEntity::ok)
            .orElseGet(() -> ResponseEntity.notFound().build());
}
@GetMapping("/email/{email}")
public ResponseEntity<User> getUserByEmail(@PathVariable String email) {
    Optional<User> user = userService.findByEmail(email);
    return user.map(ResponseEntity::ok)
            .orElseGet(() -> ResponseEntity.notFound().build());
}
@PostMapping
public ResponseEntity<User> createUser(@RequestBody User user) {
    if (userService.existsByUsername(user.getUsername()) || userService.existsByEmail(user.getEmail())) {
        return ResponseEntity.badRequest().build();
        }
        User savedUser = userService.saveUser(user);
        return ResponseEntity.ok(savedUser);
    }
    @PutMapping("/{id}")
    public ResponseEntity<User> updateUser(@PathVariable Long id, @RequestBody User user) {
        Optional<User> existingUser = userService.findById(id);
        if (!existingUser.isPresent()) {
            return ResponseEntity.notFound().build();
        }
        user.setId(id);
        User updatedUser = userService.saveUser(user);
        return ResponseEntity.ok(updatedUser);

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteUser(@PathVariable Long id) {
        if (!userService.existsById(id)) {
            return ResponseEntity.notFound().build();
        }
        userService.deleteUserById(id);
        return ResponseEntity.noContent().build();
    }
}
```

PROJECT STRUCTURE

user-management-app/

```
├── src/
│   ├── components/
│   │   ├── UserForm.js
│   │   ├── UserList.js
│   │   ├── UserDetail.js
│   │   └── Navbar.js
│   ├── pages/
│   │   ├── HomePage.js
│   │   ├── UserPage.js
│   └── App.js
│   └── index.js
```

Axios Configuration

```
import axios from 'axios';
const api = axios.create({
  baseURL: 'http://localhost:8080/api/users'
});
export default api;
```

NAVBAR.js

```
import React from 'react';
import { Link } from 'react-router-dom';
const Navbar = () => (
  <nav>
    <Link to="/">Home</Link>
    <Link to="/users">Users</Link>
  </nav>
);
export default Navbar;
```

USERLIST.js

```
import React, { useEffect, useState } from 'react';
import { Link } from 'react-router-dom';
import api from '../api';
const UserList = () => {
  const [users, setUsers] = useState([]);
  useEffect(() => {
    api.get('/')
      .then(response => setUsers(response.data))
      .catch(error => console.error(error));
  }, []);

  return (
    <div>
      <h2>User List</h2>
      <ul>
        {users.map(user => (
          <li key={user.id}>
            <Link to={`/users/${user.id}`}>{user.username}</Link>
          </li>
        ))}
      </ul>
      <Link to="/users/new">Create New User</Link>
    </div>
  );
};
export default UserList;
```

USER DETAIL .js

```
import React, { useEffect, useState } from 'react';
import { useParams, useHistory } from 'react-router-dom';
import api from '../api';
const UserDetail = () => {
  const { id } = useParams();
  const history = useHistory();
  const [user, setUser] = useState(null);
  useEffect(() => {
    api.get(`/${id}`)
```

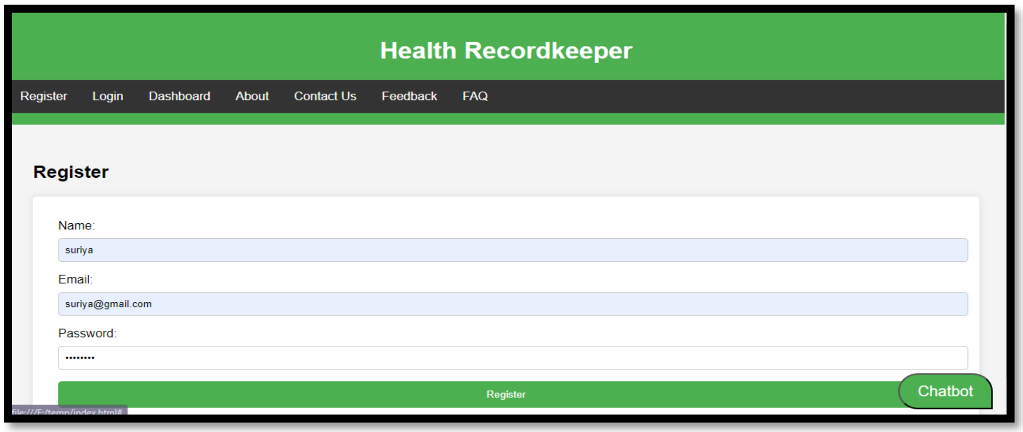
```
      .then(response => setUser(response.data))
      .catch(error => console.error(error));
  }, [id]);
  const deleteUser = () => {
    api.delete(`/${id}`)
      .then(() => history.push('/users'))
      .catch(error => console.error(error));
  };
  if (!user) return <div>Loading...</div>;
  return (
    <div>
      <h2>User Detail</h2>
      <p>Username: {user.username}</p>
      <p>Email: {user.email}</p>
      <button onClick={deleteUser}>Delete User</button>
    </div>
  );
};
export default UserDetail;
```

USERFORM.js

```
import React, { useState, useEffect } from 'react';
import { useHistory, useParams } from 'react-router-dom';
import api from '../api';
const UserForm = () => {
  const { id } = useParams();
  const history = useHistory();
  const [user, setUser] = useState({ username: '', email: '' });
  useEffect(() => {
    if (id) {
      api.get(`/${id}`)
        .then(response => setUser(response.data))
        .catch(error => console.error(error));
    }
  }, [id]);
  const handleChange = e => {
    const { name, value } = e.target;
    setUser(prevState => ({ ...prevState, [name]: value }));
  };
  const handleSubmit = e => {
    e.preventDefault();
    const request = id ? api.put(`/${id}`, user) : api.post('/', user);
    request
      .then(() => history.push('/users'))
      .catch(error => console.error(error));
  };
  return (
    <div>
      <h2>{id ? 'Edit User' : 'Create User'}</h2>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Username</label>
          <input name="username" value={user.username} onChange={handleChange} />
        </div>
        <div>
          <label>Email</label>
          <input name="email" value={user.email} onChange={handleChange} />
        </div>
        <button type="submit">{id ? 'Update' : 'Create'}</button>
      </form>
    </div>
  );
};
export default UserForm
```

HOMEPAGE.js

```
import React from 'react';
const HomePage = () => (
  <div>
    <h1>Welcome to the User Management App</h1>
  </div>
);
export default HomePage;
```



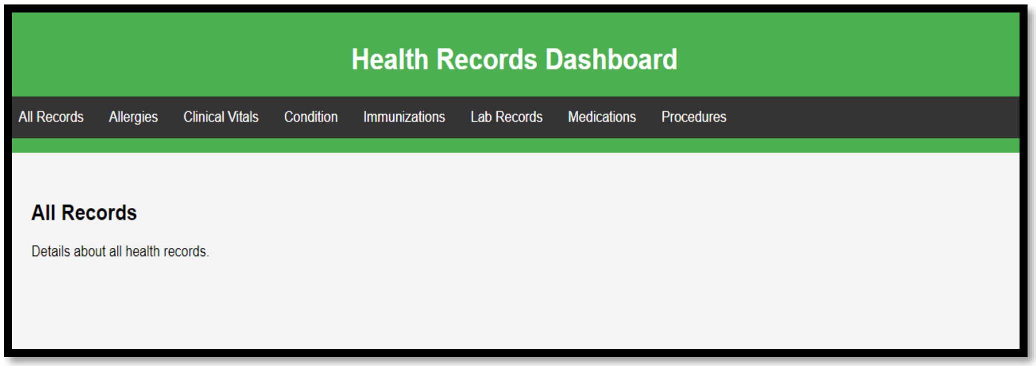
Register

Name: suriya

Email: suriya@gmail.com

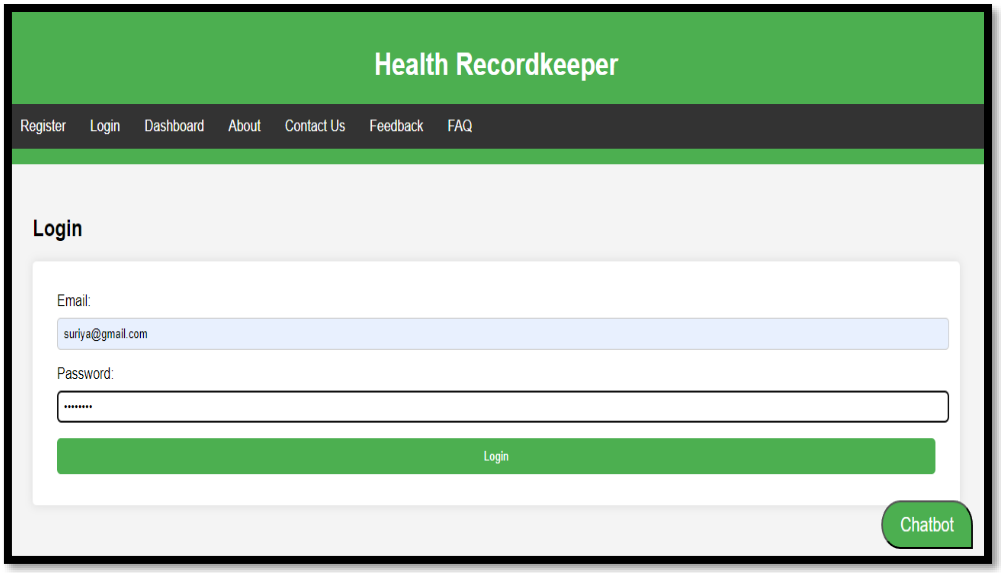
Password: ••••••••

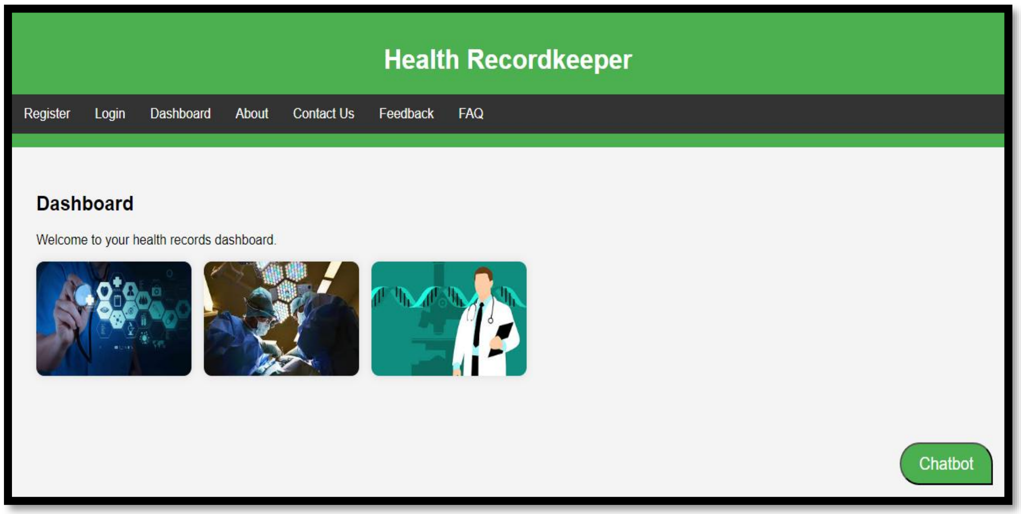
Register

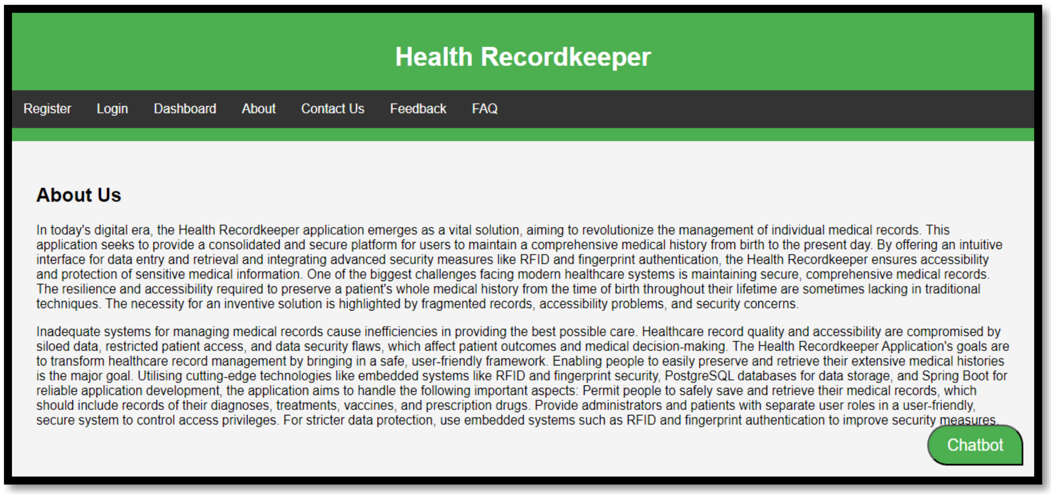


All Records

Details about all health records.



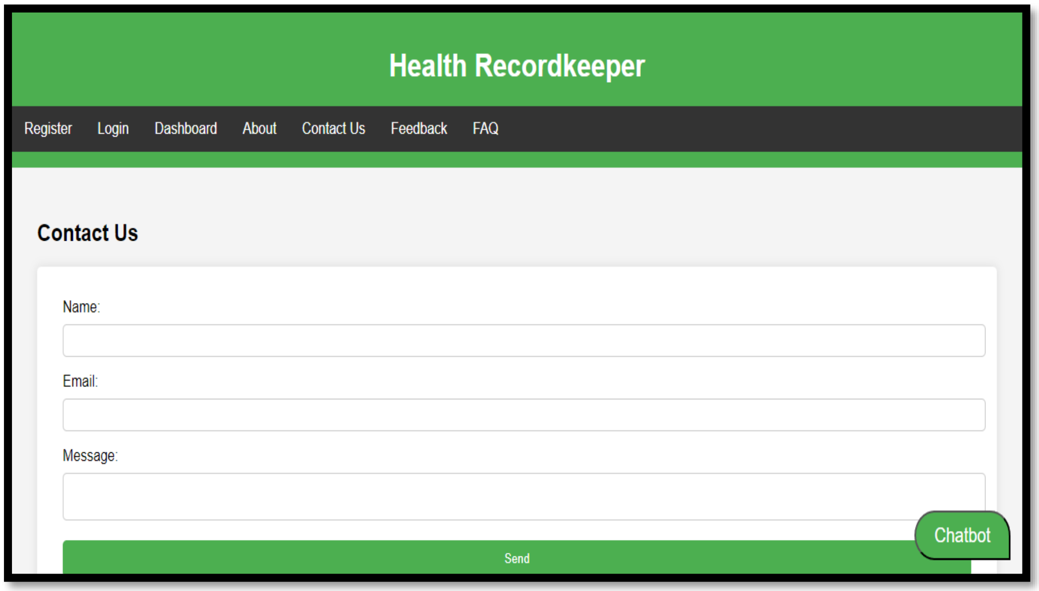


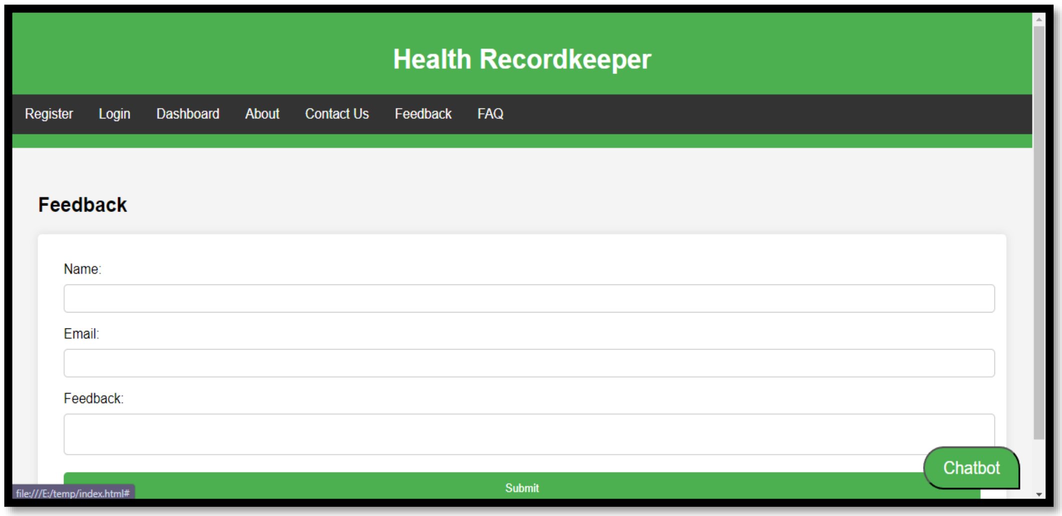


About Us

In today's digital era, the Health Recordkeeper application emerges as a vital solution, aiming to revolutionize the management of individual medical records. This application seeks to provide a consolidated and secure platform for users to maintain a comprehensive medical history from birth to the present day. By offering an intuitive interface for data entry and retrieval and integrating advanced security measures like RFID and fingerprint authentication, the Health Recordkeeper ensures accessibility and protection of sensitive medical information. One of the biggest challenges facing modern healthcare systems is maintaining secure, comprehensive medical records. The resilience and accessibility required to preserve a patient's whole medical history from the time of birth throughout their lifetime are sometimes lacking in traditional techniques. The necessity for an inventive solution is highlighted by fragmented records, accessibility problems, and security concerns.

Inadequate systems for managing medical records cause inefficiencies in providing the best possible care. Healthcare record quality and accessibility are compromised by siloed data, restricted patient access, and data security flaws, which affect patient outcomes and medical decision-making. The Health Recordkeeper Application's goals are to transform healthcare record management by bringing in a safe, user-friendly framework. Enabling people to easily preserve and retrieve their extensive medical histories is the major goal. Utilising cutting-edge technologies like embedded systems like RFID and fingerprint security, PostgreSQL databases for data storage, and Spring Boot for reliable application development, the application aims to handle the following important aspects: Permit people to safely save and retrieve their medical records, which should include records of their diagnoses, treatments, vaccines, and prescription drugs. Provide administrators and patients with separate user roles in a user-friendly, secure system to control access privileges. For stricter data protection, use embedded systems such as RFID and fingerprint authentication to improve security measures.




Health Recordkeeper
Register Login Dashboard About Contact Us Feedback FAQ
Feedback
Name:
Email:
Feedback:
Submit
Chatbot
file:///E:/temp/index.html#

CHAPTER 6 RESULTS AND DISCUSSION

6.1 SYSTEM PERFORMANCE EVALUATION

The application demonstrated commendable response times, ensuring swift access to medical records for users. Accessibility across multiple devices contributed to a seamless user experience. Initial tests indicate the application's potential for scalability, accommodating a growing user base and expanding medical records without compromising performance.

6.2 SECURITY ASSESSMENT

The integration of embedded systems, including RFID and fingerprint authentication, significantly enhanced the application's security. Unauthorized access attempts were effectively mitigated.

6.3 USER EXPERIENCE FEEDBACK

User feedback highlighted the application's intuitive design and eases of navigation, allowing users to swiftly access and views their medical histories without complexity. Users expressed confidence in the security measures implemented, particularly appreciating the biometric authentication features for safeguarding their sensitive health information.

6.4 CHALLENGES AND LIMITATIONS

1. User Adaptation to RFID Technology: The introduction of RFID technology for user authentication may pose challenges for individuals unfamiliar with this technology.

2. Initial System Familiarization: Both patients and administrators may face a learning curve during the initial implementation, adapting to the new system's features and functionalities.

3. Limited Update and Deletion Access for Patients: Patients are restricted from updating or deleting records, which may be inconvenient in cases where amendments are necessary.

4. Integration Challenges with Existing Healthcare Systems: Integration with legacy healthcare systems may present compatibility and interoperability challenges.

5. Data Privacy and Regulatory Compliance: Adhering to stringent data privacy regulations, such as HIPAA, poses challenges in terms of securing and managing health information.

6.5 FUTURE DIRECTIONS

The Health Record-Keeping System, while already demonstrating significant advancements in healthcare management, presents a promising avenue for future development and expansion. The following elaboration outlines the potential future scope of the system, including planned enhancements and areas for further exploration.

CHAPTER 7 CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The Health Recordkeeper Application signifies a pivotal advancement in healthcare record management, emphasizing user control, security, and accessibility. The application's implementation underscores the importance of patient-centric approaches and innovative technologies in enhancing healthcare outcomes.

7.2 IMPLICATIONS IN HEALTHCARE MANAGEMENT

Empowering individuals to actively manage their medical records promotes patient engagement, facilitates informed decision-making, and improves healthcare delivery. The application's security measures set a benchmark for data protection and confidentiality, setting the stage for future developments in healthcare cybersecurity.

7.3 FUTURE DIRECTIONS AND RECOMMENDATIONS:

Continual refinement of the application is essential, focusing on performance optimization, additional functionalities, and further enhancing user experience based on ongoing feedback. Collaborations with healthcare providers and regulatory bodies can further enrich the application's capabilities and compliance with evolving healthcare standards. The Health Record-Keeping System, while already demonstrating significant advancements in healthcare management, presents a promising avenue for future development and expansion. The following elaboration outlines the potential future scope of the system, including planned enhancements and areas for further exploration.

7.4 CLOSING STATEMENT

The Health Recordkeeper Application stands as a testament to the transformative potential of technology in healthcare record management. Its user-centric design, robust security protocols, and intuitive functionalities lay the groundwork for a future where individuals have secure and comprehensive control over their medical information.

REFERENCES

[1]. G. S. K, R. Kalpana, M. SK, K. S and P. PS, "Health Record Management Using Blockchain Technology," 2022 8th International Conference on Smart Structures and Systems (ICSSS), Chennai, India, 2022.

[2]. A. F. R, A. A. B, J. K and L. P, "Electronic Medical Record Management System Using

Recommendation Algorithm," 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT), Kannur, India, 2022.

[3]. S. Souiki, M. Hadjila, D. Moussaoui, S. Ferdi and S. Rais, "M-Health Application for Managing a Patient's Medical Record based on the Cloud: Design and Implementation," 2020 2nd International Workshop on Human-Centric Smart Environments for Health and Well-being (IHSH), Boumerdes, Algeria, 2021.

[4]. E. Halim, G. N. Thomas, D. R. Hidayat and D. Gea, "“Smart Healthcare” a Medical Record System for Effective Health Services," 2020 International Conference on Information Management and Technology (ICIMTech), Bandung, Indonesia, 2020, pp. 806-811.

[5]. P. Y. S. Lakshman, D. P. Kumar, N. Ramesh and K. Pranathi, "Medical Records Management Using Cloud Technology," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2021.

[6]. W. -K. Yang, J. -S. Chen and Y. -S. Chen, "An Electronic Medical Record Management System Based on Smart Contracts," 2019 Twelfth International Conference on Ubi-Media Computing (Ubi-Media), Bali, Indonesia, 2019.