

E-HEALTH MANAGEMENT SYSTEM



A PROJECT REPORT

Submitted by

DHARANI D V (2303811710422028)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)**

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **“E-HEALTH MANAGEMENT SYSTEM”** is the bonafide work of **DHARANI D V (2303811710422028)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

CGB1201-JAVA PROGRAMMING
Mr. M. SARAVANAN, M.E.,
SUPERVISOR
ASSISTANT PROFESSOR

SIGNATURE

Mr. M. Saravanan, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 02.12.2024.

CGB1201-JAVA PROGRAMMING
Mr. MANI ARMANAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

INTERNAL EXAMINER

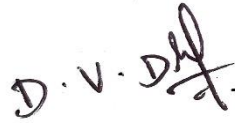
CGB1201-JAVA PROGRAMMING
Dr.R.SETHAMILSELVI, M.E., Ph.D.,
EXTERNAL EXAMINER
PROFESSOR
8138-SCE, TRICHY.

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**E-HEALTH MANAGEMENT SYSTEM**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201-JAVA PROGRAMMING**.

Signature

A handwritten signature in black ink, appearing to read 'D.V. D.V.' with a stylized flourish at the end.

Dharani D V

Place: Samayapuram

Date: 02.12.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director. **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E., Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mr. M. SARAVANAN, M.E.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The E-Health Management System is a comprehensive software solution designed to enhance the management of healthcare operations, improving patient care, administrative efficiency, and data security. The system is built around several core modules that address key healthcare management needs, such as user authentication, patient record management, appointment scheduling, and real-time feedback. Each module is carefully designed to cater to different user roles (admin, staff, and patients), ensuring secure, role-based access to system functionalities. The User Management Module ensures secure login and role-based access control, enabling admins to manage all aspects of the system, staff to handle patient and appointment data, and patients to manage their personal details and book appointments. The Patient Management Module stores and organizes patient information, providing easy retrieval and viewing of patient records. The Appointment Management Module facilitates appointment scheduling and management, ensuring that patients can book, modify, and view their appointments with healthcare providers. The Validation and Error Handling Module performs real-time data validation to ensure integrity and provides user-friendly error messages and feedback. The Reporting and Feedback Module gives users immediate feedback on their actions, confirming successful operations or indicating errors with actionable solutions. By integrating these modules, the system ensures a cohesive user experience and reliable operation.

1 ABSTRACT WITH POs AND PSOs MAPPING

2 CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The E-Health Management System is a comprehensive software solution designed to enhance the management of healthcare operations, improving patient care, administrative efficiency, and data security. The system is built around several core modules that address key healthcare management needs, such as user authentication, patient record management, appointment scheduling, and real-time feedback. Each module is carefully designed to cater to different user roles (admin, staff, and patients), ensuring secure, role-based access to system functionalities.	PO1 -3 PO2 -3 PO3 -3 PO4 -3 PO5 -3 PO6 -3 PO7 -3 PO8 -3 PO9 -3 PO10 -3 PO11-3 PO12 -3	PSO1 -3 PSO2 -3 PSO3 -3

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	viii
1	INTRODUCTION	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
2	PROJECT METHODOLOGY	4
	2.1 Proposed Work	4
	2.2 Block Diagram	5
3	MODULE DESCRIPTION	6
	3.1 User Management Module	6
	3.2 Patient Management Module	6
	3.3 Appointment Management Module	7
	3.4 User Interface Module (GUI)	7
	3.5 Input Validation and Error Handling Module	8
4	CONCLUSION & FUTURE SCOPE	9
	4.1 Conclusion	9
	4.2 Future Scope	9
	REFERENCES	10
	APPENDIX A (SOURCE CODE)	11
	APPENDIX B (SCREENSHOTS)	18

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE

The objective of the **E-Health Management System** is to create a comprehensive, user-friendly, and secure platform that revolutionizes the management of healthcare services. The system aims to streamline patient records by maintaining accurate, centralized, and up-to-date data, including demographics, medical history, and treatment details, accessible only to authorized personnel. It facilitates seamless appointment scheduling, allowing patients to book, view, and manage appointments with healthcare providers efficiently, thus reducing waiting times and improving time management. By fostering enhanced communication between patients, healthcare providers, and administrative staff, the system ensures better coordination and improved service delivery.

1.2 OVERVIEW

The E-Health Management System is a comprehensive Java-based application that employs a modular approach to deliver an all-encompassing solution for healthcare administration. The system is divided into key modules: User Management, Patient Management, Appointment Scheduling, Validation and Error Handling, and Reporting. Each module works cohesively to ensure seamless operation. User authentication and account creation are managed securely through password validation and role-based access. The patient management module allows efficient storage and retrieval of patient information, while the appointment module facilitates easy scheduling with proper date validation.

The application is designed with a user-friendly graphical interface using Java Swing, enabling dynamic interaction for tasks such as viewing patient records, booking appointments, and generating feedback. Data is managed through in-memory structures like ArrayLists, ensuring efficient operations while leaving room for future database integration. Through centralized validation and error handling mechanisms, the system prevents inconsistencies and ensures smooth workflows. The system is a scalable, intuitive, and reliable platform tailored to meet the dynamic needs of modern healthcare facilities.

1.3 JAVA PROGRAMMING CONCEPTS

1.3.1 OBJECT-ORIENTED PROGRAMMING (OOP)

Object-Oriented Programming (OOP) is the foundation of the system's design. Key principles such as Encapsulation, Inheritance, Polymorphism, and Abstraction were employed to structure the application. These principles allowed the creation of reusable, modular, and maintainable code. Encapsulation ensured that sensitive data, such as patient information, was securely managed, while inheritance allowed for efficient code reuse, particularly in the different user roles like Admin, Staff, and Patient.

1.3.2 EXCEPTION HANDLING

Java's robust exception handling mechanism was essential to prevent the application from crashing due to unexpected errors. The try-catch blocks were used to handle errors gracefully, such as invalid user input, database connection issues, or incorrect appointment dates. Custom error messages guided the users to fix issues, ensuring a smooth and uninterrupted experience. This concept was crucial for maintaining the integrity and stability of the system.

1.3.3 COLLECTION FRAMEWORK

The Java Collections Framework was used to manage large datasets efficiently, such as patient records and appointment details. ArrayList, HashMap, and HashSet were used for dynamic data storage and retrieval. The HashMap helped in managing role-based access control, enabling quick lookups of user roles and corresponding permissions, while ArrayList was used to store and manipulate collections of patient or appointment data.

1.3.4 FILE HANDLING (I/O)

Java's File I/O functionality was used to read and write data to text files or databases, ensuring that important data such as user profiles and appointment histories were stored persistently. BufferedReader and FileWriter were used for efficient reading and writing, while serialization was employed for saving and retrieving complex objects like patient records. This concept was key for enabling data persistence in the application.

1.3.5 MULTITHREADING

Java's multithreading capabilities were employed to allow multiple tasks to run concurrently, ensuring the system remained responsive. Tasks such as appointment scheduling, sending notifications, and background data processing could run in parallel without blocking the main user interface. This was crucial for maintaining a smooth user experience, especially in an environment where real-time updates and responsiveness are essential.

CHAPTER 2

PROJECT METHODOLOGY

2.1 PROPOSED WORK

The proposed E-Health Management System is designed to enhance healthcare management through the use of technology, offering a user-friendly, secure, and efficient platform for managing patient records, appointments, and other healthcare services. The system aims to streamline the administrative and clinical workflows for healthcare providers, ensuring that medical information is organized, accessible, and easily managed.

The system will be developed with several core objectives:

2.1.1 User Management: The platform will include robust user management features to ensure secure access based on different roles, such as Admin, Staff, and Patients. Each user will have customized access to the system depending on their role, ensuring that sensitive medical data is only accessible to authorized personnel.

2.1.2 Patient Information Management: A centralized database will store patient information, including personal details, medical history, and treatment plans. This information will be accessible to both medical staff and the patients themselves, allowing for better collaboration and improved patient care.

2.2.3 Appointment Scheduling: The system will enable patients to book appointments online, and healthcare providers can view, manage, and update appointments easily. The scheduling system will also include automated reminders to help minimize missed appointments and optimize the clinic's schedule.

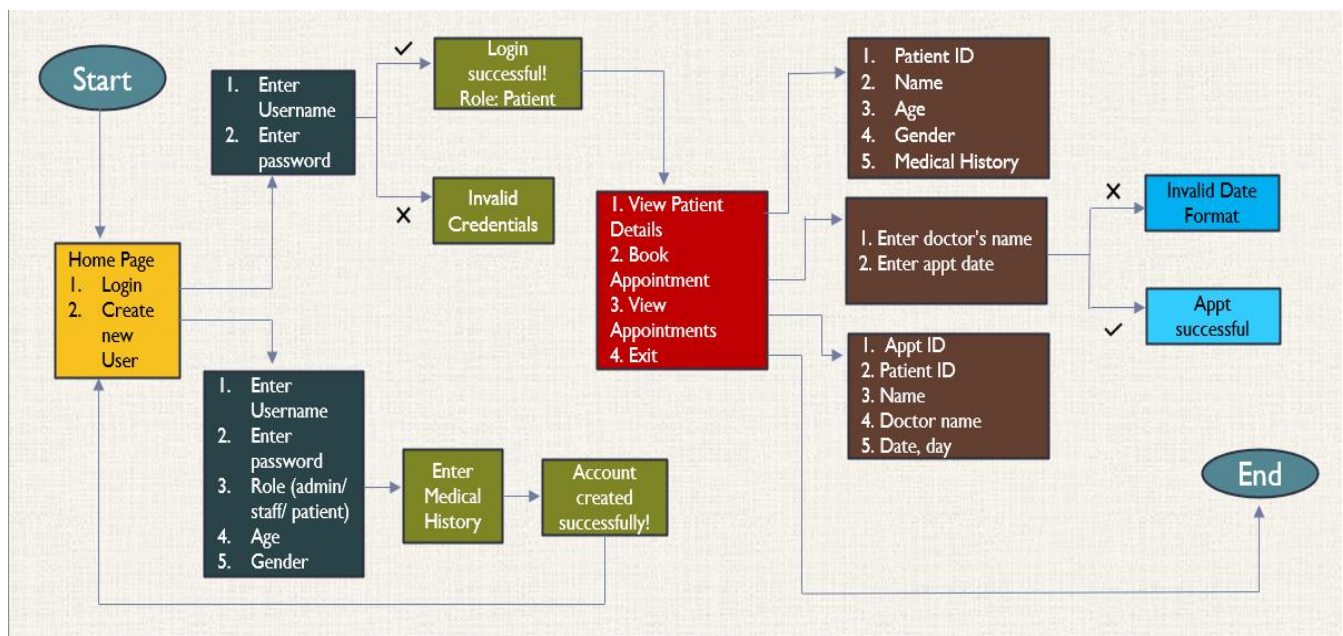
2.2.4 Security and Data Privacy: Ensuring the privacy and security of sensitive data will be a top priority. The system will implement encryption protocols, user authentication, and secure data transmission to protect patient and healthcare provider information.

2.2.5 Reporting and Analytics: The platform will feature reporting tools to track patient visits, treatment outcomes, and staff performance. These reports will help healthcare administrators make informed decisions about resource allocation and service improvements.

2.2.6 Error Handling and Validation: The system will incorporate extensive error handling and validation mechanisms to ensure smooth operation. Input fields will be validated to prevent data entry errors, and appropriate feedback will be provided to users in case of any issues.

By integrating these functionalities, the E-Health Management System aims to create a seamless, secure, and efficient environment for managing healthcare services. The project's goal is to improve healthcare delivery by reducing administrative burden, minimizing errors, and enhancing patient and staff experiences.

2.2 BLOCK DIAGRAM



CHAPTER 3

MODULE DESCRIPTION

The E-Health Management System consists of various modules, each designed to address specific aspects of the application, ensuring efficient healthcare management. Below is an overview of each module, highlighting its unique features.

3.1 USER MANAGEMENT MODULE

This module is the backbone of the system, enabling secure access and efficient role-based management of users. The **login functionality** ensures the system verifies user credentials securely, utilizing robust password authentication mechanisms. Through **role-based access**, it differentiates between Admin, Staff, and Patient roles, providing varying levels of system access tailored to their responsibilities. The **account creation** process allows new users to register by entering necessary details, including username, password, and role, with optional fields for additional information. It incorporates error-handling measures to ensure valid input and maintain unique usernames. Additionally, the module emphasizes **reporting and feedback**, displaying success messages for actions like account creation and login, while clearly communicating errors such as invalid credentials or duplicate usernames.

3.2 PATIENT MANAGEMENT MODULE

This module handles the efficient management of patient information, ensuring seamless integration of personal and medical details into the system. Through its **patient records management**, it facilitates the addition of patient data during account creation, including critical details such as age, gender, and medical history. The module enables **viewing and searching patient details**, allowing authorized users like Admin and Staff to access comprehensive patient information quickly. A **search functionality** is implemented to locate patient records by name, providing instant retrieval of relevant data.

3.3 APPOINTMENT MANAGEMENT MODULE

This module simplifies the process of booking and managing appointments, creating a seamless experience for both patients and healthcare providers. It supports **appointment booking**, enabling patients to select a preferred doctor and schedule appointments by specifying a suitable date. The module ensures proper **date validation**, using reliable formats to prevent scheduling errors. Additionally, **viewing appointments** is role-specific, with Admin and Staff having access to all scheduled appointments, while patients can view only their own. By dynamically managing appointment records and ensuring real-time updates, this module ensures smooth communication between patients and providers, significantly enhancing healthcare efficiency.

- 3.3.1 Appointment booking is facilitated by selecting a doctor and specifying a date.
- 3.3.2 Date format validation ensures appointments are scheduled correctly.
- 3.3.3 Admin and staff can view all appointments, while patients can access their own appointment details.
- 3.3.4 ArrayList is used to dynamically manage appointment records, with date validation handled through SimpleDateFormat.

3.4 USER INTERFACE MODULE (GUI)

This module offers an intuitive and interactive graphical interface for users to engage with the system effectively. Designed using **Java Swing**, it provides visually appealing and functional interfaces for operations such as login, account creation, and role-specific tasks. Through **dynamic input dialogs**, users can perform actions like booking appointments and adding patient details seamlessly. **Navigation between features** is facilitated through menu-driven options tailored to the user's role, ensuring a streamlined workflow. This module prioritizes user experience by presenting clear feedback and ensuring all functionalities are easily accessible, making the system user-friendly and efficient for all roles.

3.5 INPUT VALIDATION AND ERROR HANDLING MODULE

This module is integral to maintaining the stability, security, and accuracy of the system, addressing errors proactively and ensuring consistent data integrity. Through **centralized data validation**, all system inputs such as user roles, patient details, and appointment dates are rigorously verified to prevent inaccuracies. It incorporates **error prevention mechanisms**, restricting invalid actions based on user roles, thereby ensuring that only authorized operations are executed. Logical workflows are integrated seamlessly, enabling users to perform permitted actions while mitigating the risk of unintended errors.

By validating data at every step and providing clear feedback through error messages or success confirmations, this module plays a crucial role in enhancing user experience and maintaining the overall reliability of the system. Its integration with core application logic ensures a smooth and secure operation, making it an indispensable part of the E-Health Management System.

3.5.1 Password validation is used during login to ensure security.

3.5.2 Input validation is performed for patient details, appointment dates, and role-specific actions.

3.5.3 Confirmation messages are displayed for successful account creation and appointment bookings.

3.5.4 Error messages are shown for failed operations, such as incorrect login credentials or invalid date formats

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The E-Health Management System provides a robust solution to streamline healthcare operations, ensuring secure, efficient, and user-friendly management of essential healthcare processes. By integrating modules like User Management, Patient Record Handling, Appointment Scheduling, and Data Validation, the system offers a seamless experience for administrators, healthcare professionals, and patients alike. With role-based access, each user can securely interact with the system according to their needs, ensuring that sensitive data is protected and workflows are efficiently managed. The system's centralized approach to data validation and error handling enhances its reliability and stability, minimizing human error and ensuring data integrity throughout. These features guarantee that users are limited to performing actions permitted for their roles, thus reducing the risk of invalid data or incorrect actions. As healthcare demands grow, the scalability of this system ensures it remains effective in managing larger datasets and more complex workflows.

4.2 FUTURE SCOPE

The future of the E-Health Management System offers significant opportunities for innovation and growth. Key advancements could include the integration of cloud computing, allowing for real-time synchronization across multiple locations and devices. Another promising avenue for the system's future development is the incorporation of Artificial Intelligence (AI) and machine learning algorithms. These technologies could support predictive analytics for better health management, such as anticipating patient needs or diagnosing conditions more accurately. On the security front, advanced encryption protocols and multi-factor authentication could bolster the protection of patient data, ensuring the system remains resilient against cyber threats. As technology continues to evolve, the E-Health Management System could expand to encompass additional modules like medical billing, inventory management, and patient monitoring, further streamlining healthcare management and improving patient outcomes.

REFERENCES

WEBSITES:

1. Oracle Corporation. *Java SE Documentation*. Retrieved from <https://docs.oracle.com/javase/>
2. Sun Microsystems. *Java Swing Tutorial*. Retrieved from <https://docs.oracle.com/javase/tutorial/uiswing/>
3. Stack Overflow. *Discussions on Appointment Scheduling and Java Swing Implementation*. Retrieved from <https://stackoverflow.com>
4. GeeksforGeeks. *Java Concepts and Practical Implementation Guides*. Retrieved from <https://www.geeksforgeeks.org/>
5. Journal of Medical Informatics. *Role of Automation in Modern Healthcare Management*, Vol. 45, Issue 3, 2020.
6. Java Code Examples. *In-memory Data Management Techniques*. Retrieved from <https://javacodeexamples.com/>
7. Kumar, R. *E-Health Systems: Development and Implementation Strategies*. Tech Publishers, 2021.

YOUTUBE LINKS:

1. Java for Beginners - Java Brains
 - URL: <https://www.youtube.com/user/koushks>
 - Offers Java tutorials from the basics to advanced concepts. The channel provides detailed guides on Java programming, including working with objects and classes, which are crucial for building an EPMS.

APPENDIX A (SOURCE CODE)

```
import javax.swing.*;
import java.awt.*;
import java.util.List;
import java.util.ArrayList;
import java.util.Date;
import java.text.SimpleDateFormat;

// 1. Patient Class
class Patient {
    private int id;
    private String name;
    private int age;
    private String gender;
    private String medicalHistory;

    public Patient(int id, String name, int age, String gender, String medicalHistory) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
        this.medicalHistory = medicalHistory;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getMedicalHistory() {
        return medicalHistory;
    }

    @Override
    public String toString() {
        return "Patient ID: " + id + ", Name: " + name + ", Age: " + age + ", Gender: " + gender
+
        ", Medical History: " + medicalHistory;
    }
}

// 2. Appointment Class
class Appointment {
```

```

private int id;
private int patientId;
private String patientName;
private String doctorName;
private Date appointmentDate;

public Appointment(int id, int patientId, String patientName, String doctorName, Date
appointmentDate) {
    this.id = id;
    this.patientId = patientId;
    this.patientName = patientName;
    this.doctorName = doctorName;
    this.appointmentDate = appointmentDate;
}

public int getPatientId() {
    return patientId;
}

@Override
public String toString() {
    return "Appointment ID: " + id + ", Patient ID: " + patientId + ", Patient Name: " +
patientName +
        ", Doctor: " + doctorName + ", Date: " + appointmentDate;
}
}

// 3. Appointment Manager
class AppointmentManager {
    List<Appointment> appointments = new ArrayList<>();
    private int nextId = 1;

    public void bookAppointment(Patient patient, String doctorName, Date date) {
        appointments.add(new Appointment(nextId++, patient.getId(), patient.getName(),
doctorName, date));
        System.out.println("Appointment booked successfully!");
    }

    public void listAppointments() {
        if (appointments.isEmpty()) {
            System.out.println("No appointments available.");
        } else {
            appointments.forEach(System.out::println);
        }
    }
}

// 4. Patient Manager Class
class PatientManager {
    private List<Patient> patients = new ArrayList<>();

```

```

private int nextId = 1;

public Patient addPatient(String name, int age, String gender, String medicalHistory) {
    Patient newPatient = new Patient(nextId++, name, age, gender, medicalHistory);
    patients.add(newPatient);
    return newPatient;
}

public void listPatients() {
    if (patients.isEmpty()) {
        System.out.println("No patients available.");
    } else {
        patients.forEach(System.out::println);
    }
}

public Patient findPatientByName(String name) {
    return patients.stream().filter(p ->
p.getName().equalsIgnoreCase(name)).findFirst().orElse(null);
}
}

```

// 5. User Authentication Class

```

class User {
    private String username;
    private String password;
    private String role;

    public User(String username, String password, String role) {
        this.username = username;
        this.password = password;
        this.role = role;
    }

    public String getUsername() {
        return username;
    }

    public boolean authenticate(String password) {
        return this.password.equals(password);
    }

    public String getRole() {
        return role;
    }
}

class UserManager {
    private List<User> users = new ArrayList<>();
}

```

```

public UserManager() {
    // Adding default users
    users.add(new User("admin", "admin123", "admin"));
    users.add(new User("staff", "staff123", "staff"));
    users.add(new User("patient1", "patient123", "patient"));
}

public User login(String username, String password) {
    return users.stream()
        .filter(user -> user.getUsername().equals(username) &&
user.authenticate(password))
        .findFirst()
        .orElse(null);
}

public void addUser(String username, String password, String role) {
    users.add(new User(username, password, role));
}
}

// 6. Main Class with GUI
public class EHealthSystemGUI {
    private static UserManager userManager = new UserManager();
    private static PatientManager patientManager = new PatientManager();
    private static AppointmentManager appointmentManager = new AppointmentManager();
    private static User loggedInUser = null;

    public static void main(String[] args) {
        JFrame frame = new JFrame("E-Health Management System");
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new BorderLayout());

        // Initial panel for login or account creation
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3, 1));

        JLabel welcomeLabel = new JLabel("Welcome to the E-Health Management System",
SwingConstants.CENTER);
        panel.add(welcomeLabel);

        JButton loginButton = new JButton("Login");
        JButton createAccountButton = new JButton("Create New Account");

        panel.add(loginButton);
        panel.add(createAccountButton);

        frame.add(panel, BorderLayout.CENTER);
        frame.setVisible(true);
    }
}

```



```

// Add action listeners
loginButton.addActionListener(e -> showLoginDialog(frame));
createAccountButton.addActionListener(e -> showCreateAccountDialog(frame));
}

private static void showLoginDialog(JFrame frame) {
    JPanel loginPanel = new JPanel(new GridLayout(3, 2));

    JLabel usernameLabel = new JLabel("Username:");
    JTextField usernameField = new JTextField();
    JLabel passwordLabel = new JLabel("Password:");
    JPasswordField passwordField = new JPasswordField();

    loginPanel.add(usernameLabel);
    loginPanel.add(usernameField);
    loginPanel.add(passwordLabel);
    loginPanel.add(passwordField);

    int option = JOptionPane.showConfirmDialog(frame, loginPanel, "Login",
JOptionPane.OK_CANCEL_OPTION);
    if (option == JOptionPane.OK_OPTION) {
        String username = usernameField.getText();
        String password = new String(passwordField.getPassword());
        loggedInUser = userManager.login(username, password);

        if (loggedInUser != null) {
            JOptionPane.showMessageDialog(frame, "Login Successful! Role: " +
loggedInUser.getRole());
            startEHealthSystem(frame);
        } else {
            JOptionPane.showMessageDialog(frame, "Invalid credentials. Try again or create a
new user.");
        }
    }
}

private static void showCreateAccountDialog(JFrame frame) {
    JPanel createAccountPanel = new JPanel(new GridLayout(6, 2));

    JLabel usernameLabel = new JLabel("Username:");
    JTextField usernameField = new JTextField();
    JLabel passwordLabel = new JLabel("Password:");
    JPasswordField passwordField = new JPasswordField();
    JLabel roleLabel = new JLabel("Role (admin/staff/patient):");
    JTextField roleField = new JTextField();
    JLabel ageLabel = new JLabel("Age:");
    JTextField ageField = new JTextField();
    JLabel genderLabel = new JLabel("Gender:");
    JTextField genderField = new JTextField();

```

```

createAccountPanel.add(usernameLabel);
createAccountPanel.add(usernameField);
createAccountPanel.add(passwordLabel);
createAccountPanel.add(passwordField);
createAccountPanel.add(roleLabel);
createAccountPanel.add(roleField);
createAccountPanel.add(ageLabel);
createAccountPanel.add(ageField);
createAccountPanel.add(genderLabel);
createAccountPanel.add(genderField);

int option = JOptionPane.showConfirmDialog(frame, createAccountPanel, "Create
Account", JOptionPane.OK_CANCEL_OPTION);
if (option == JOptionPane.OK_OPTION) {
    String username = usernameField.getText();
    String password = new String(passwordField.getPassword());
    String role = roleField.getText();
    String ageText = ageField.getText();
    String gender = genderField.getText();

    try {
        if (role.equalsIgnoreCase("patient")) {
            int age = Integer.parseInt(ageText);
            String medicalHistory = JOptionPane.showInputDialog(frame, "Enter your
medical history:");
            patientManager.addPatient(username, age, gender, medicalHistory);
        }
        userManager.addUser(username, password, role);
        JOptionPane.showMessageDialog(frame, "Account created successfully!");
    } catch (Exception e) {
        JOptionPane.showMessageDialog(frame, "Invalid details provided.");
    }
}

private static void startEHealthSystem(JFrame frame) {
    // Define options for role-specific menu
    String[] options = "patient".equals(loggedInUser.getRole())
        ? new String[]{"View Patient Details", "Book Appointment", "View
Appointments", "Exit"}
        : new String[]{"View All Patients", "View All Appointments", "Exit"};

    int choice = JOptionPane.showOptionDialog(frame, "Choose an action:", "Menu",
        JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE, null,
options, options[0]);

    if (choice == 0) {
        if ("patient".equals(loggedInUser.getRole())) {
            Patient loggedInPatient =
patientManager.findPatientByName(loggedInUser.getUsername());

```

```

        JOptionPane.showMessageDialog(frame, loggedInPatient != null ?
loggedInPatient.toString() : "Patient not found");
    } else {
        patientManager.listPatients();
    }
} else if (choice == 1) {
    if ("patient".equals(loggedInUser.getRole())) {
        // Book an appointment
        String doctorName = JOptionPane.showInputDialog(frame, "Enter doctor's
name:");
        String dateStr = JOptionPane.showInputDialog(frame, "Enter appointment date
(yyyy-MM-dd):");
        try {
            Date date = new SimpleDateFormat("yyyy-MM-dd").parse(dateStr);
            Patient loggedInPatient =
patientManager.findPatientByName(loggedInUser.getUsername());
            appointmentManager.bookAppointment(loggedInPatient, doctorName, date);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(frame, "Invalid date format.");
        }
    } else {
        appointmentManager.listAppointments();
    }
} else if (choice == 2) {
    // View appointments based on user role
    if ("patient".equals(loggedInUser.getRole())) {
        Patient loggedInPatient =
patientManager.findPatientByName(loggedInUser.getUsername());
        if (loggedInPatient != null) {
            // Filter and show appointments for this patient
            appointmentManager.appointments.stream()
                .filter(a -> a.getPatientId() == loggedInPatient.getId())
                .forEach(a -> JOptionPane.showMessageDialog(frame, a.toString()));
        } else {
            JOptionPane.showMessageDialog(frame, "No appointments found for this
patient.");
        }
    } else {
        appointmentManager.listAppointments(); // Staff/admin can see all appointments
    }
} else {
    System.exit(0);
}
}
}

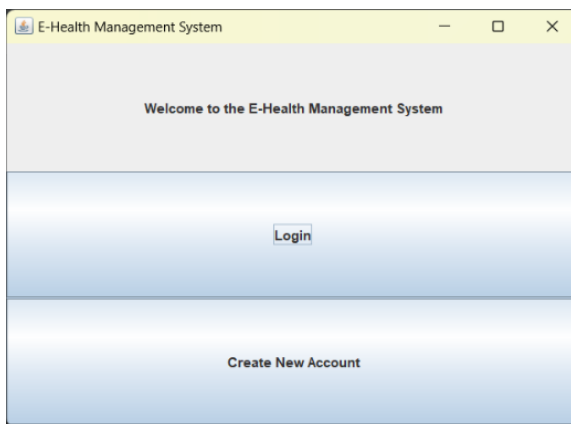
```

APPENDIX B (SCREENSHOTS)

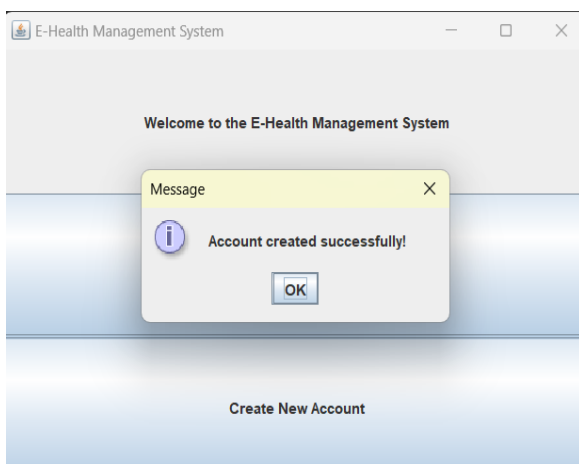
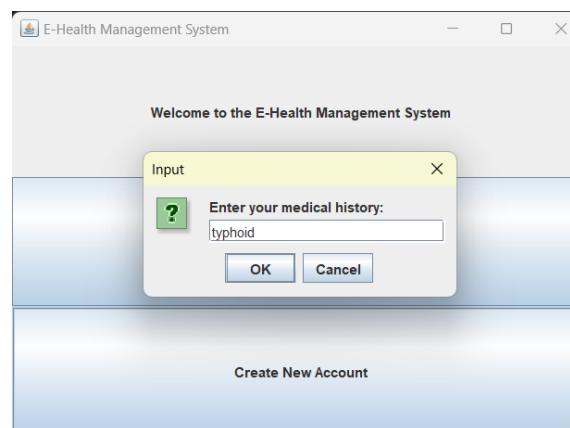
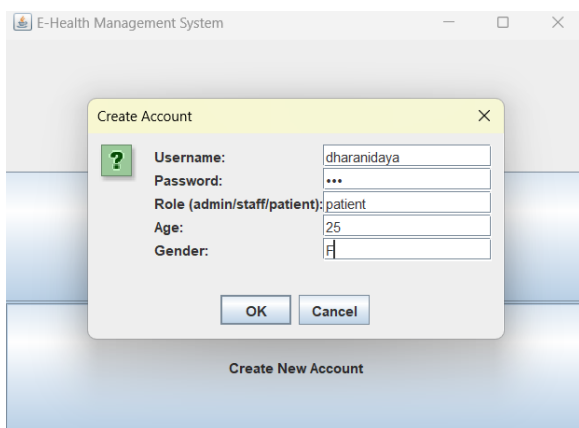
1. E- Health Management System Home page

1.1 Login

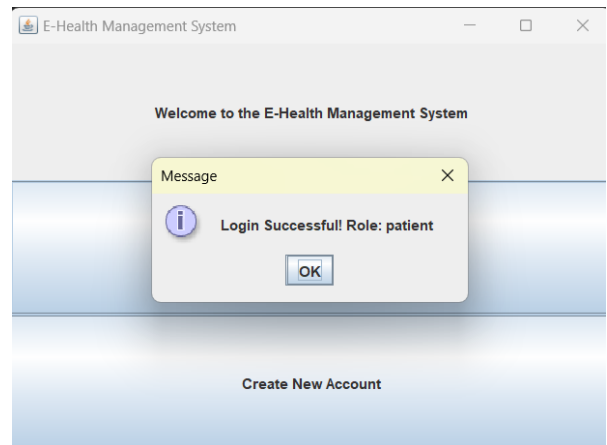
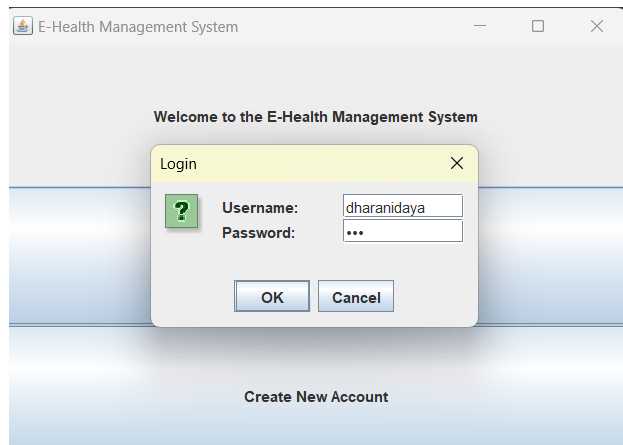
1.2 Create new Account



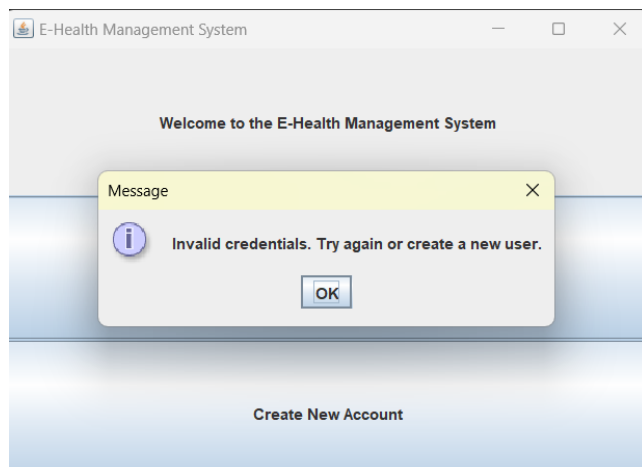
1.2 Create new Account



1.1 Login

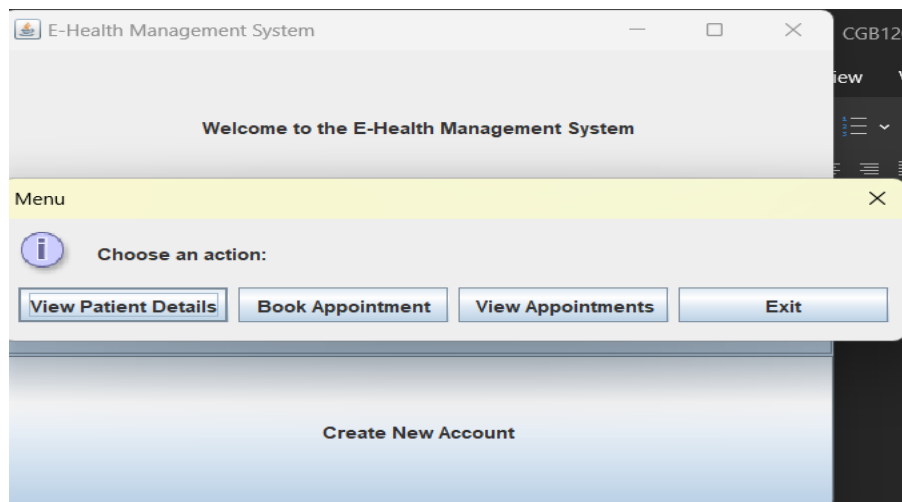


(If entered wrong username or password)

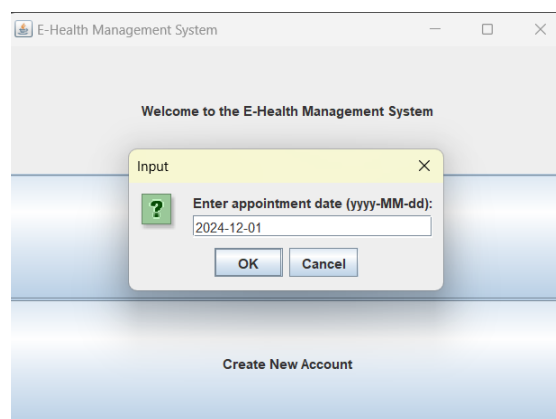
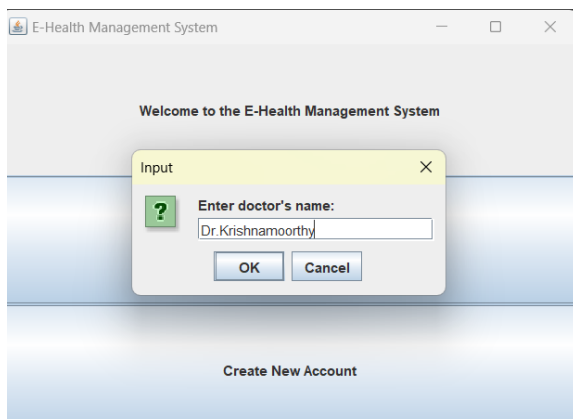


2.1 Menu after Login

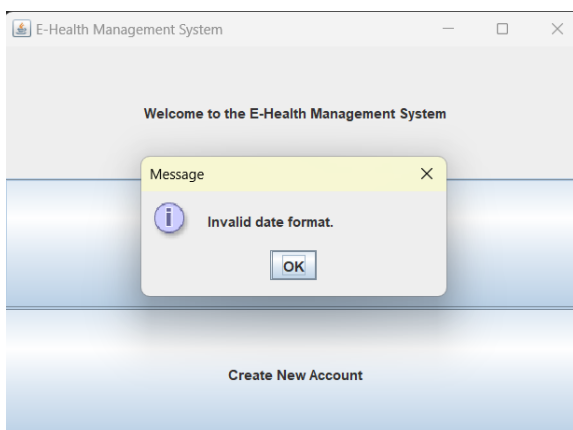
- 2.1.1 View Patient Details
- 2.1.2 Book Appointment
- 2.1.3 View Appointment
- 2.1.4 Exit



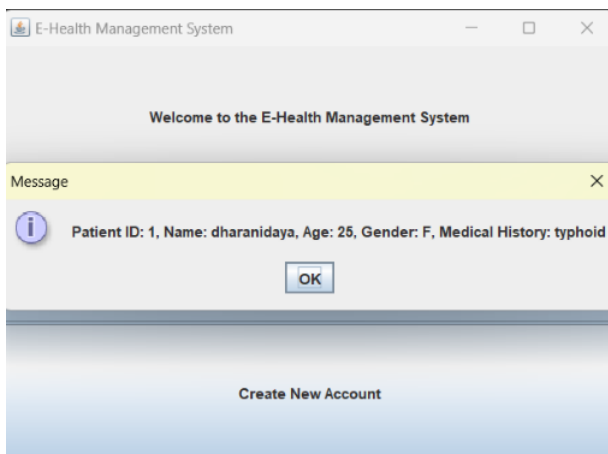
2.1.2 Book Appointment



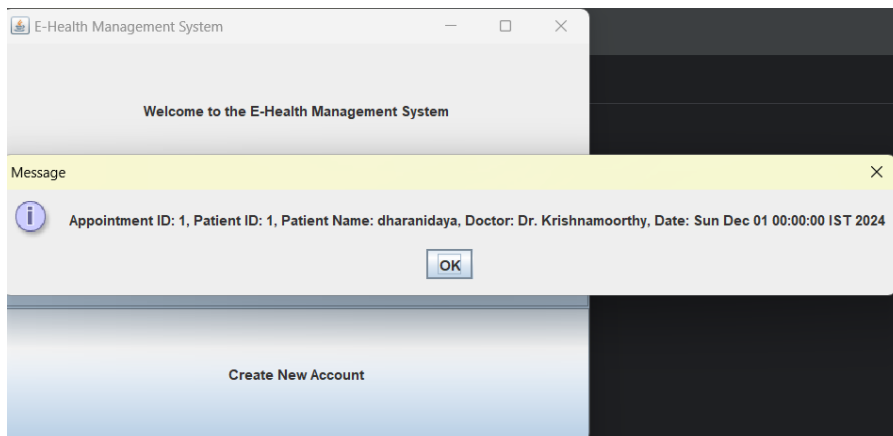
(If entered wrong Date Format)



2.1.1 View Details



2.1.3 View Appointment



2.1.4 Exit

