Remote Patient Monitoring System: Project Requirement Report

Abstract :

The Remote Patient Monitoring System (RPMS) is a Java-based application designed to enhance healthcare delivery by enabling remote monitoring of patients' health, facilitating communication between stakeholders, and automating critical healthcare processes. Leveraging object-oriented programming (OOP) principles, the system integrates a MySQL database, JavaFX for the user interface, and external APIs for notifications (e.g., Twilio for SMS, Gmail SMTP for email). This report outlines the projects requirements, its significance in addressing healthcare challenges, and the reasons it stands as a robust and impactful solution. The RPMS demonstrates the practical application of OOP concepts such as inheritance, polymorphism, encapsulation, and abstraction, while providing a scalable and user-centric platform for healthcare management.

1. Introduction :

The Remote Patient Monitoring System (RPMS) addresses the growing need for efficient, secure, and accessible healthcare solutions in an era where remote healthcare is increasingly vital. Traditional healthcare systems often face challenges such as fragmented data, manual errors, delayed communication, and limited scalability. The RPMS mitigates these issues by providing a centralized platform where patients, doctors, and administrators can interact seamlessly. Built using Java with JavaFX for the front-end and MySQL for data persistence, the system employs OOP principles to ensure modularity, maintainability, and extensibility. This report details the functional and non-functional requirements of the RPMS, its significance in improving healthcare delivery, and why it is a valuable project for academic and practical purposes.

2. Project Requirements:

The RPMS is designed to meet specific functional and non-functional requirements, ensuring it addresses the needs of all stakeholders while maintaining reliability and performance. Below are the key requirements:

2.1 Functional Requirements:

• User Management: The system must support three user rolespatients, doctors, and administratorswith distinct functionalities: – Patients can view prescriptions, feedback, vital signs, request appointments, and communicate via chat or video.
– Doctors can view patient vitals, prescribe medications, provide feedback, manage appointments, and engage in communication.
– Administrators can add/remove users and view system logs.
• Data Management: A MySQL database must store user profiles, appointments, vital signs, feedback, prescriptions, chat messages, and video consultation details.
• Appointment Scheduling: Patients can request appointments, and doctors can approve or cancel them, with automated reminders sent via email.

• Health Monitoring: The system must allow patients to upload vital signs (via CSV files) and enable doctors to view trends using JavaFX charts. • Communication: Secure chat and video consultation features must facilitate doctor - patient interaction, with messages stored in the database. • Emergency Alerts: The system must detect abnormal vital signs (e.g., heart rate > 100 bpm) and send alerts via email or SMS using Twilio.
• Reporting: Doctors and patients can generate health reports summarizing vital signs, feedback, and prescriptions, saved as text files.

2.2 Non-Functional Requirements :

• Performance: The system must handle database queries efficiently, with response times under 2 seconds for most operations.
• Scalability: The design must support adding new users and features without significant refactoring, leveraging OOP principles.
• Security: User data must be protected with role-based access control, and communication (e.g., email, SMS) must use secure protocols.
• Usability: The JavaFX interface must be intuitive, with clear navigation between login, registration, and role-specific dashboards.
• Reliability: The system must handle exceptions gracefully (e.g., database failures, invalid inputs) and ensure data integrity with transactions.

3. Significance of the RPMS:
The RPMS addresses critical challenges in healthcare delivery, making it a significant contribution to both technology and society. Its importance can be evaluated across several dimensions:

• Improved Access to Care: By enabling remote monitoring and communication, the RPMS ensures patients can manage their health without frequent hospital visits, which is particularly beneficial for those with chronic conditions or mobility issues. For instance, patients can upload vital signs and receive timely feedback from doctors, reducing delays in care. • Enhanced Efficiency: Automating processes like appointment scheduling, reminders, and vital sign monitoring reduces the administrative burden on healthcare providers. Doctors can quickly access patient data through the centralized database, improving decision-making during consultations.
• Error Reduction: The system enforces data integrity through database constraints and input validation (e.g., ensuring phone numbers are in E.164 format for SMS). This minimizes manual errors, such as incorrect prescriptions or scheduling conflicts.
• Real-Time Alerts: The emergency alert system, which triggers notifications for abnormal vital signs, ensures timely interventions. For example, if a patients oxygen level drops below 95%, doctors are alerted via email or SMS, potentially saving lives in critical situations.
• Secure Communication: The integration of chat and video consultation features, backed by a secure database, ensures that sensitive patient-doctor interactions are protected. Role-based access control further safeguards data privacy.

• Data-Driven Insights: The health trends visualization (using JavaFX charts) allows doctors to identify patterns in patient vital signs, facilitating proactive care. For example, a rising trend in blood pressure can prompt early intervention.

4. Why the RPMS is a Good Project :

The RPMS stands out as a commendable project for several reasons, reflecting its technical merit, practical impact, and alignment with OOP principles:

• Application of OOP Principles: The project effectively demonstrates OOP concepts: – Inheritance: The User class hierarchy (Patient, Doctor, Administrator) allows for code reuse and role-specific functionality. – Polymorphism: Interfaces like UserOperations, Notifiable, and Alertable enable flexible behavior across classes (e.g., EmailNotification and SMSNotification implement Notifiable). – Encapsulation: Private fields with public getters/setters (e.g., in VitalSign, Appointment) protect data integrity. – Abstraction: Abstract classes like EmergencyAlert and interfaces provide a blueprint for related functionalities, enhancing modularity.

• Technical Excellence: The project integrates multiple technologiesJavaFX for the GUI, MySQL for data persistence, and external APIs (Twilio, Gmail SMTP)showcasing the ability to build a full-stack application. The use of design patterns (e.g., Singleton in DataBase) ensures efficient resource management.

Github link:
https://github.com/Ailya-Shah/OOP-RHMS