Health Logger: Managing Patient Details and Vitals Recording

The Health Logger application is aimed at managing patient details and recording patient vitals for medical practitioners. It includes functionalities for patient management and vitals management, with features such as adding patients, listing patients, searching patients, updating patient details, adding vitals records, listing vitals records, displaying vitals in graphs, identifying patients with critical vitals, and exporting health logs.

Project Agenda:

The project involves developing a prototype of the Health Logger application within a timeframe of 15 working days. The key deliverables include a specification document outlining application capabilities, appearance, and user interactions, along with the use of Java EE concepts, frontend and backend technologies, and the MySQL database.

Tasks to Perform:

Tasks include developing the application using Java EE, working with Java Server Pages (JSP) for the frontend, implementing Servlets for the backend, and performing CRUD operations using JDBC or Hibernate for MySQL. Additionally, operations such as patient management and vitals management are to be implemented, with various functionalities detailed in the project agenda.

Flow and Features:

The application flow involves documenting the flow of the application and preparing a flow chart, listing core technologies and algorithms, and coding to display the welcome screen. The application features include options for managing patients and managing vitals, with functionalities for authentication,

adding patient details, listing patients, updating patient details, adding vitals records, listing vitals records, displaying vitals in graphs, identifying patients with critical vitals, and exporting health logs.

Conclusion:

In conclusion, the Health Logger application prototype aims to address the needs of medical practitioners in managing patient details and recording patient vitals efficiently. By utilizing Java EE concepts, Servlets, JSP, JDBC or Hibernate, MySQL, and other relevant technologies, the application provides a user-friendly interface for medical professionals to access and manage patient information effectively. The implementation of various features and functionalities outlined in the project agenda ensures the application meets the requirements of James Tech Pvt. Ltd. and provides a solid foundation for further development and refinement.

Algorithm for Health Logger Application

here's a high-level algorithm for the Health Logger application:

1. **Initialize Application:**
    - Start the application.
    - Display the welcome screen with the title "Health Logger: Managing Patient Details and Vitals Recording".
2. **User Authentication:**
    - Prompt the user to log in using predefined admin credentials.
    - Validate the entered credentials against the database records.
    - If authentication is successful, proceed to the dashboard. Otherwise, display an error message.
3. **Dashboard:**

- Display options for managing patients and managing vitals.
- Allow the user to select an option.

4. **Patient Management:**
    - If the user selects patient management:
        - Provide options to add, list, search, update, and navigate to patient vitals.
        - Implement functionality for each option using appropriate HTTP requests and servlets.
        - Ensure proper validation of user input and handle any errors gracefully.
5. **Vitals Management:**
    - If the user selects vitals management:
        - Provide options to add, list, visualize, identify critical patients, and export vitals.
        - Implement functionality for each option using appropriate HTTP requests and servlets.
        - Utilize chart.js library to visualize vitals data in graphs.
        - Apply threshold values to identify patients with critical vitals.
        - Implement export functionality to generate CSV files of patient vitals.
6. **Exception Handling:**
    - Implement proper exception handling mechanisms to handle errors gracefully.
    - Display meaningful error messages to the user in case of invalid inputs or system failures.
7. **Navigation:**
    - Implement a navigation bar or menu for easy access to different features and options.

- Ensure seamless navigation between different sections of the application.

8. **Source Code Optimization:**
    - Utilize appropriate Java concepts such as collections and searching techniques for optimizing source code.
    - Implement efficient algorithms for sorting and searching patient and vitals data.
9. **Testing:**
    - Perform thorough testing of the application to ensure functionality and usability.
    - Test various scenarios, including valid and invalid inputs, edge cases, and error conditions.
10. **Deployment:**
    - Once testing is complete, deploy the application on a server with Apache Tomcat or similar web server.
11. **Documentation:**
    - Document the application flow, features, and technical details for reference and future development.
12. **Presentation:**
    - Prepare a presentation showcasing the prototype of the Health Logger application, including its features, functionalities, and potential benefits.
13. **Feedback and Iteration:**
    - Gather feedback from stakeholders and users.
    - Iterate on the application based on feedback and make necessary improvements.

This algorithm outlines the key steps involved in developing the Health Logger application prototype. Each step should be implemented carefully to ensure the functionality, usability, and reliability of the application.