Karthikeyan Sivasubramanian

Project 1: Context Monitoring App

Abstract:

A context refers to the state of an environment or information about the situation of an entity, including the user and the application, which is relevant for decision-making and processing. Context-aware applications are those that interact with their environment to make real-time decisions based on gathered information. These systems offer significant advantages, such as personalized user experiences and proactive support functionalities. The use of context-aware applications in healthcare, utilizing a network of sensors, allows for a comprehensive monitoring system that is fast and reliable for processing health information. This project aims to design and implement a context-aware Android application named ***HEALTHYWE*** that monitors vital signs, such as heart rate and respiration rate, using built-in sensors in smartphones. In addition to capturing vital information, the application records symptoms from the user and saves them in a local database. Through this application, the project seeks to provide a convenient and user-friendly means of monitoring and storing the health information and a detailed understanding of the context-based mechanisms and databases.

Introduction:

Context-aware applications play a crucial role in the healthcare sector by sensing and processing vital physiological data to make informed decisions. With continuous monitoring setups, these applications can assist in the early diagnosis of potential health issues, enabling proactive treatments. Health-based context-aware applications have significantly increased in number due to their growing popularity and the ease with which sensors can be integrated into smaller devices such as watches, mobile phones, and PCs. This project presents an application called HEALTHYWE that uses health-context information to monitor heart rate and respiratory rate from the body. The primary objectives of this application are as follows:

- Design a user-friendly and interactive application with the required UI elements that senses heart rate and respiratory rate using built-in sensors.
- Design a pop-up action menu to receive the intensity of 10 different types of symptoms as input from the user.
- Create a local SQL-based database to record and update the readings of vital signs along with symptom ratings received from the user.

Technical Approach:

The approach to designing the application is divided into three main parts:

1. Setting up the User Interface (UI):

The UI for the app includes several buttons to trigger functionalities for reading the two vital signs, updating values to the database, and navigating back to the previous page. Widgets to open a pop-up menu for symptom selection are also provided. Additionally, an image view is used to display the app logo, and text views are created for displaying heart rate, respiratory rate, symptom names, app title, and other miscellaneous text. To receive intensity input for each symptom, a rating bar with star ratings from 1 to 5 is included.

2. Adding heart rate and respiratory rate functionalities:

A backend function to monitor and compute the values of the vital signs is linked to the corresponding UI elements separately:

Heart rate sensing:

Heart rate sensing is done using the built-in camera. With the flashlight on, a 45-second video of the index finger pressed against the lens is recorded. The variation in red coloration is used to calculate the heart rate. This method requires access to the flash and camera for recording the video, as well as access to external storage to retrieve the URI of the media.

Respiratory rate sensing:

Respiratory rate is measured using the built-in accelerometer or gyroscope by placing the mobile phone flat on the chest and evaluating the acceleration along the x, y, and z axes. Access to retrieve the accelerometer readings require in app permissions.

3. Implementing the database and other miscellaneous functionalities:

An SQL-based database using RoomDB (named symptom_ratings) is created with id (int), symptom_name (string), and ratings (float) as attributes, with id serving as the primary key. The database includes methods for inserting, updating, deleting elements, resetting the auto-increment to reset the ids, and retrieving all the data. The database setup is connected to the submit button to store data in the table. Additionally, backend methods are implemented to handle user input for button clicks (e.g., back button, selecting symptoms via the pop-up menu, and entering ratings), completing the overall implementation.

Design Choices:

Platform: Android: Emulator v 2024.1.2 (Koala);

TargetSDK: API level 33; Target device: Nexus 6P

UI/UX Design

The app is split into three different activities: Main activity displaying the app name and logo along with elements to compute and display the heart rate and respiratory rate along with a button to move to the symptom uploading. The second activity involved selection of 10 different symptoms. This activity offers a pop-up menu button to select symptoms and a back button to navigate back. The third activity is required to upload the ratings for the selected symptom and submit the rating to the local database and a back button to select other symptoms. Layout: Constrained Layout; Widgets: TextView(4), Buttons(7), Rating bar(1), ImageView(1);

Functionalities

Symptom selection (Pop-up action), heart rate and respiratory rate sensing, back navigation, rating upload, and database upload and querying.

Database Choice

For storing the 10 symptoms and two vital signs information, a table or SQL based local database is designed using RoomDB. The database stores id, symptom_name and rating values of type int, string and float as the types respectively. The database functions include addRating() to add new symptom, insertRating to update existing symptom, getRating to get a symptom value, getAllRatings() to get all symptoms value, deleteRatingsForSymptoms() to delete specific symptoms, clearAllRatings() to clear the table, resetAutoIncrement() to reset the id attribute upon deletion.

Sensors

Camera and accelerometer are the sensors used to monitor the heart and respiratory rate. The heart rate can be detected using the variation in red color pixels from the finger as recorded by the camera and respiratory rate through the breathing pattern as measured by the accelerometer.

Implications:

Real-time data monitoring: The app measures heart rate and respiratory rate using real-time data, which can be useful in various environments.

Additional symptom input: The app allows users to input different health issues as symptoms, providing a comprehensive health report that can be helpful for diagnosis and treatment.

Local database usage: The vital signs and symptom ratings are stored in a local database, allowing efficient access to the data through offline means. This local setup ensures the security of private health data and offers an effective way to query the information.

Limitations:

Sensor dependency: The application heavily relies on the camera and accelerometer. Improper calibration and sensor quality can affect measurements.

Energy inefficient method: Using the camera to record video with the flash and the accelerometer to read data for 45 seconds consumes significant energy.

Limited context: The data from the user and the vital signs are limited, making it difficult to perform a comprehensive health analysis. Moreover, user input can sometimes be unreliable, skewing the analysis and predictions.

Local database dependence: As the amount of data increases, the local database, using internal memory, can become heavily occupied, leading to difficulties in storing large amounts of information.

Suggestions for future work:

Cloud integration: Cloud integration allows for long-term storage and resolves memory constraints. Several advanced algorithms can be run in the cloud for better health analytics, improving overall performance.

Integrating additional context: The app could be extended to consider more contextual data beyond heart rate and respiratory rate for a more advanced health analysis system. Personalization features such as a diet tracker and exercise tips could provide more health insights.

Conversion to a dynamic system: Incorporating feedback mechanisms, such as health warnings or reminders based on vital sign thresholds, can make the system more proactive.

Battery optimization: Reducing energy consumption by optimizing sensor usage and processing algorithms can be beneficial.

Links:

Youtube video link:-

https://www.youtube.com/watch?v=_A9EDUjSxJQ