File: `database_manager.py`

Description

This file defines the `DatabaseManager` class, which is responsible for managing the SQLite database used to store and retrieve blood pressure data. The class provides methods to create the database table, add new blood pressure entries, and fetch entries filtered by a specified number of days.

Classes

`DatabaseManager`

This class manages the interaction with an SQLite database for storing and retrieving blood pressure data.

Methods

- **`__init__(self, db_path='db/blood_pressure.db')`**
  Initializes the `DatabaseManager` instance and establishes a connection to the SQLite database.
  - The default database path is `db/blood_pressure.db`.
  - Calls `create_table()` to ensure the database table is created.
- **`create_table(self)`**
  Creates the `blood_pressure` table in the database if it does not already exist.
  - The table has columns: `id` (primary key), `timestamp` (datetime of the entry), `sys` (systolic blood pressure), `dia` (diastolic blood pressure), and `pulse` (heart rate).
- **`add_entry(self, sys, dia, pulse)`**
  Adds a new blood pressure entry to the database.
  - The `timestamp` is automatically set to the current date and time.
  - The systolic (`sys`), diastolic (`dia`), and pulse (`pulse`) values are provided as parameters.
  - The data is inserted into the `blood_pressure` table.
- **`fetch_filtered_data(self, days)`**
  Fetches blood pressure data from the database, filtered by the number of days specified.
  - `days`: The number of days of data to retrieve (entries newer than this threshold).
  - The method returns a list of tuples containing the `timestamp`, `sys`, `dia`, and `pulse` values of the fetched entries.

Database Schema

The SQLite database contains a single table `blood_pressure` with the following schema: `-` `id` (INTEGER, Primary Key): A unique identifier for each blood pressure entry. - **`timestamp`** (TEXT): The timestamp when the entry was created (formatted as `YYYY-MM-DD HH:MM:SS`). - **`sys`** (INTEGER): The systolic blood pressure value. - **`dia`** (INTEGER): The diastolic blood pressure value. - **`pulse`** (INTEGER): The pulse rate.

Usage

1. **Database Initialization**: When the `DatabaseManager` is instantiated, it connects to the SQLite database (or creates it if it doesn't exist) and ensures the `blood_pressure` table is created.
2. **Adding Entries**: You can add new blood pressure entries by calling `add_entry()` with the systolic, diastolic, and pulse values. The current timestamp is automatically assigned.
3. **Fetching Data**: To fetch blood pressure entries within a specified range of days, use `fetch_filtered_data()`, passing the number of days to filter by. The method returns all matching entries within the last `n` days.

Dependencies

- **sqlite3**: Used for database management (creating tables, inserting data, and fetching entries).
- **datetime**: Used for handling timestamps and calculating date ranges for filtering.