File: `evaluation_dialog.py`

Description

This file defines the `EvaluationDialog` class, which provides a user interface for evaluating blood pressure data. It displays a table of blood pressure entries and their corresponding categories based on systolic and diastolic values. It allows users to filter data by days and export the results to a PDF. The dialog includes a legend explaining the color-coded blood pressure categories.
The program makes use of QTs capability to render HTML. This way consistent styling of the interface, as well as the PDFs generation is given.

Classes

`EvaluationDialog`

This class represents a dialog for evaluating and displaying blood pressure data. The dialog shows a table of entries with color-coded rows indicating the blood pressure category. It provides an option to export the data as a PDF.

Constants

- `COLOR_OPTIMAL`: Represents the color for optimal blood pressure (green).
- `COLOR_NORMAL`: Represents normal blood pressure (slightly darker green).
- `COLOR_HOCHNORMAL`: Represents high-normal blood pressure (yellow).
- `COLOR_HYPERTONIE_GRAD_1`: Represents hypertension grade 1 (light red).
- `COLOR_HYPERTONIE_GRAD_2`: Represents hypertension grade 2 (darker red).
- `COLOR_HYPERTONIE_GRAD_3`: Represents hypertension grade 3 (dark red).
- `COLOR_ISOLIERTE_HYPERTONIE`: Represents isolated systolic hypertension (same as grade 3).
- `COLOR_DEFAULT`: Default color for rows with no category (white).

HTML Legend: HTML string that provides a legend for the color coding of blood pressure categories.

Methods

- `__init__(self, db_manager, parent=None)`
  Initializes the dialog window.
  - Sets the window title to "Auswertung."
  - Displays the HTML legend explaining blood pressure categories.
  - Adds a combo box to filter entries by days.
  - Adds a table to display blood pressure data.
  - Includes a button to export the data as a PDF.
- `display_data(self)`
  Fetches and displays the blood pressure data based on the selected day

filter from the combo box. Populates the table with the entries.

- **`insert_row(self, entry)`**
  Inserts a new row in the table with the blood pressure entry data.
  - Colors the row based on the blood pressure category.
- **`apply_row_color(self, row_position, color)`**
  Applies a background color to the row based on the blood pressure category.
- **`get_row_color(self, sys, dia)`**
  Determines the color for a row based on systolic and diastolic blood pressure values.
- **`export_pdf(self)`**
  Exports the blood pressure data and the legend to a PDF file (`evaluation_report.pdf`).
  - Generates the HTML content for the report.
  - Uses the `QPrinter` class to create the PDF.
- **`generate_html_content(self)`**
  Generates the HTML content for the evaluation report.
  - Loads CSS from an external file (`res/styles.css`).
  - Builds the HTML content for the legend and the table of blood pressure entries.
  - Adds the data from the selected days filter to the table.

Dependencies

- **PyQt5 Modules**:
  - `QDialog`, `QVBoxLayout`, `QLabel`, `QComboBox`, `QTableWidget`, `QTableWidgetItem`, `QPushButton`: Used to create the dialog, layout, and UI components.
  - `QColor`, `QTextDocument`, `QPrinter`: Used for color handling and generating PDF exports.
- **Custom Modules**:
  - `BloodPressureEntry`: Represents a blood pressure entry, used to structure the data.
  - `DaysOption`: Provides day filtering options.
  - `db_manager`: Used for database operations, fetching blood pressure entries.

Key Features

1. **Blood Pressure Categories**
   Blood pressure entries are color-coded based on predefined categories (e.g., Optimal, Normal, Hypertonie Grad 1). The categories are explained in an HTML legend.

2. **Data Filtering**
   Users can select a number of days from the combo box to filter the displayed blood pressure entries.

3. **PDF Export**
   The data, including the color-coded table and legend, can be exported to a PDF file.

4. **Interactive Table**
   The table dynamically updates when the user changes the day filter. Each entry is displayed w