Design patterns:

The HeartWave project is a data-driven application that focuses on measuring and displaying performance metrics and analysis in a well-organized and user-friendly manner.

Implemented using Qt, the project incorporates the Observer pattern. The Observer pattern is a behavioral design pattern that establishes a one-to-many dependency between objects, allowing automatic updates and notifications whenever an object changes its state. In our case, buttons and battery spinbox within the User UI emit signals when user presses the buttons and modifies battery levels. The MainWindow listens to these signals by connecting their slots to the signals. When a signal is emitted, the corresponding connected slot will be automatically executed to allow these objects to communicate with each other. For example, when user presses the sensor button, the button sends a signal, which will be handled by activateSensor() function in MainWindow class automatically. Consequently, user will see that the HR contact symbol is on the screen, and the session begins outputting the HRV graph and metrics.

Moreover, the Menu class exhibits some characteristics of the Composite pattern. This pattern composes objects in a tree structure to represent part-whole hierarchies. Although separate Leaf and Composite classes are absent, the Menu class accommodates both simple menu items and other menus within its subMenu list. This structure enables a tree-like organization, a common feature of the Composite pattern. Each menu object may contain multiple submenu objects, with methods available for adding, removing, and retrieving child menu items.

In summary, while the project does not predominantly depend on design patterns, it does demonstrate the presence of both the Observer and Composite patterns.