Both the RouteHandler and the Manager incorporate fault tolerance. The RouteHandler publishes unfiltered data to the MqttBroker and the Manager subscribes to the unfiltered data and runs it through the PipeAndFilter to be be filtered according to the user's desired parameters. Because both the RouteHandler and the Manager rely heavily on the Mqtt broker, the fault tolerance for both components pertained to reconnecting to the broker in the case that either component would become disconnected.

The fault tolerance mechanism is the same for both the RouteHandler and the Manager. They use a method in the Python paho library called on_disconnect which runs every time the client is disconnected. The method is runs a loop in which it tries to reconnect to the broker. Once it has been reconnected, the loop breaks and the program continues to run as normal.

The mechanism has a positive on the system because it only runs when a disconnection has occurred. This means that it doesn't slow the program down while the program is running normally.

The mechanism still allows the components to subscribe to and publish multiple topics at once so it does not get in the way of scalability. It does not affect the throughput in anyway since it is not used for controlling the flow of data and preventing an overload of data. The throughput would only be affected if the connection with the broker is lost. When everything is working as it should be and the connection is working, the latency is not affected. It also does not affect the availability when everything is working as it should be. Some functionality of the system is not available if a disconnection occurs, but the on_disconnect method works to fix that. Because both of the components in the backend that are connected to the broker utilize the same method with the same logic, there is consistency in how the system deals with the same kind of fault.