Project Report: Clustering and Heuristic Optimization Application

1. Project Objective

This project is a software solution aimed at solving the clustering problem using algorithms such as K-means, DBSCAN, and heuristic optimization methods like Hill Climbing and Simulated Annealing. The user can load data, apply clustering algorithms, and improve the solution using heuristic methods. The project is designed with a modular structure, making it expandable and maintainable.

2. Project File Structure

The file structure of the project is as follows:

Root Directory:

- main.py
- qt_design.py
- undo_redo.py
- utils.py
- **controllers:**
  - clustering_controller.py
  - commands.py
  - edit_controller.py
  - file_controller.py
  - heuristic_controller.py
  - __init__.py
- **models:**
  - data_loader.py
  - solution.py
  - __init__.py
- **services:**
  - clustering_service.py
  - heuristic_service.py

__init__.py

Data Files:

Exported_Data

Saved_Data

Doxyjen Files:

html

latex

3. Class Structure and Functional Modularity

The project is designed with a modular structure where each functionality is handled by separate classes. Each class serves a specific purpose, ensuring that the project remains maintainable and extendable. Below is a description of the primary files and their functions:

- main.py: Controls the main flow of the application, handling the startup and overall management of the program.
- qt_design.py: Contains the graphical user interface (GUI) design using PyQt or a similar GUI library.
- undo_redo.py: Manages undo and redo operations. This file is responsible for implementing these functions using object-oriented programming principles.
- utils.py: Contains utility functions used across various parts of the application. These may include data preprocessing, mathematical calculations, or general helper functions.

- controllers:
  - clustering_controller.py: Manages the execution of clustering algorithms. It triggers appropriate clustering operations based on user interactions.
  - commands.py: Manages and executes various commands in the system. These commands include operations like loading data, running clustering algorithms, and others.
  - edit_controller.py: Manages data editing operations. This file controls actions like clearing data, editing, or undoing changes to clustering and heuristic solutions.

  - file_controller.py: Handles file operations such as loading, saving, and exporting data.
  - heuristic_controller.py: Manages the execution of heuristic optimization techniques like Hill Climbing and Simulated Annealing to improve clustering results.
  - __init__.py: Marks the `controllers` directory as a Python package, enabling modular import of the files in the directory.

- models:
  - data_loader.py: Responsible for loading data from external files and making it ready for processing. It handles data parsing and preprocessing.
  - solution.py: Contains logic to process the clustering results and represents the solutions, including storing cluster labels and other relevant data.
  - __init__.py: Marks the `models` directory as a Python package, facilitating the import of the necessary modules within the project.

- services:
  - clustering_service.py: Contains the core clustering algorithms and handles their execution. It interacts with libraries like scikit-learn for clustering.
  - heuristic_service.py: Implements heuristic optimization methods to improve clustering results, including techniques like Hill Climbing and Simulated Annealing.
  - __init__.py: Marks the `services` directory as a Python package, enabling the structured use of service classes across the project.

4. Doxygen Compatibility

The project is documented to be compatible with **Doxygen**. Each class and method is detailed with comments that explain their functionality. These comments describe the parameters the methods take, the return values, and the operations they perform. This ensures that the project remains understandable and maintainable, and provides a foundation for generating documentation automatically using Doxygen.

5. Conclusion

This project is designed using object-oriented programming principles, ensuring that it is modular, maintainable, and easily extendable. The Python files are organized in such a way that each file serves a specific role, making it easier to manage the codebase.

Additionally, Doxygen-compatible comments have been included to ensure proper documentation of the project, making it suitable for future expansion and easy maintenance.