**Documentation for “Barbie” Project**

**Title:** Barbie Database Manager

**Description:** A Qt-based application for managing a Barbie doll database. It allows users to load data from a CSV file, add new entries, delete existing entries, and sort the database.

**Purpose of the project:** The Barbie Database Manager project aims to provide a user-friendly application for managing and organizing information about Barbie dolls. This application is designed to help users efficiently handle their Barbie collections by offering functionalities for loading, adding, deleting, and sorting Barbie data.

**Main features:**

1. Data Loading and Saving

* Load Data from CSV: Allows users to load existing Barbie doll data from a CSV file.
* Save Data to CSV: Enables users to save the current state of the database to a CSV file for future use.

1. Data Management

* Add New Barbie Entry: Provides an interface for users to input new Barbie doll information and add it to the database.
* Delete Barbie Entry: Allows users to select and delete an existing Barbie entry from the database.

1. Data Organization

* Sort Data: Users can sort the Barbie entries based on various columns such as series, model, year, sales, price, and accessories.

1. User Interaction and Interface

* Graphical User Interface (GUI): An intuitive and easy-to-navigate interface built with Qt Designer.
* Interactive Table: A table that displays the Barbie data, allowing for easy viewing and interaction.

**Main functions:**

1. File Handling Functions

* loadDataFromCSV(const QString &filePath):
  + Opens a CSV file from the specified file path.
  + Reads the contents and populates the QTableWidget with the data.
* saveDataToCSV(const QString &filePath):
* Iterates through the QTableWidget and writes the data to a CSV file at the specified file path.

1. Data Management Functions

* addBarbie():
* Reads input from various UI fields (e.g., QLineEdit, QSpinBox, QDoubleSpinBox).
* Creates a new row in the QTableWidget and populates it with the input data.
* deleteBarbie():
* Deletes the currently selected row in the QTableWidget.

1. Sorting Functions

* sortTable():
  + Sorts the data in the QTableWidget based on the selected column in the QComboBox.
  + Utilizes QTableWidget’s sortItems(int column) method to perform the sorting.

**Setup instructions**

Prerequisites

* Qt framework installed (Qt Creator recommended)
* C++ compiler

Installation Steps

1. Clone the repository

|  |  |
| --- | --- |
| 1  2 | git clone https://github.com/yourusername/barbie-database-manager.git  cd barbie-database-manager |

1. Open the project in Qt Creator

* Open Qt Creator.
* Click on "File" -> "Open File or Project".
* Navigate to the cloned repository and open the .pro file.

1. Build the project

* Click on the "Build" button or press Ctrl+B.

1. Run the project

* Click on the "Run" button or press Ctrl+R.

**Usage Instructions**

Loading Data

* Launch the application.
* Click on the "Load Data" button.
* Browse and select the CSV file containing Barbie data.

Adding a New Barbie

* Enter the details in the respective fields:
  + Series
  + Model
  + Year
  + Sales
  + Price
  + Accessories
* Click the "Add" button.

Deleting a Barbie

* Select the row you want to delete.
* Click the "Delete" button.

Sorting the Data

* Select the column by which you want to sort the data from the dropdown.
* Click the "Sort" button.

**Code explanation**

Main components:

* Window “Barbie-history”: Displays the information about “History of the creation of Barbie dolls”
* Window “Database”: The main interface class that handles user interactions and displays the data
* Window “Main window”: Main window that has links to other windows
* Window “What kind of Barbie are you?”: Window with test

**Additional information**

File structure:

* The UI Design files:
  + ‘aboutbarbie.ui’
  + ‘databasewindow.ui’
  + ‘mainwindow.ui’
  + ‘testnabarbie.ui’
* Header files:
  + ‘aboutbarbie.h’
  + ‘databasewindow.h’
  + ‘testnabarbie.h’
  + ‘mainwindow.h’
* Implementation files:
  + ‘aboutbarbie.cpp’
  + ‘databasewindow.cpp’
  + ‘testnabarbie.cpp’
  + ‘mainwindow.cpp’
* ‘main.cpp’: Entry point of the application