```python
# controller of disk database program
#it is the graphical interface
# this will be the overall control of the calory program
# it creates a meun, reads in the configuration and
# drives the program


import config_disk_db as CD       # get the configuration classs

import disk_db as DB




import os
import sys
import platform
from loguru import logger

# to import from subdir, create __init__.py in subdir and then you can import
# in this case we import module database_connect.py in subdir ui_files

from ui_files.database_connect import Ui_db_dialog

from PySide6.QtWidgets import (QApplication,
                               QFileDialog,
                               QDialog,
                               QLabel,
                                QMainWindow,
                                QMenu,
                                QPushButton,
                                QVBoxLayout,
                                QWidget)
from PySide6.QtGui import QAction, QIcon
from PySide6.QtUiTools import QUiLoader



loader = QUiLoader()
basedir = os.path.dirname(__file__)


class database_dialog(QDialog,Ui_db_dialog):
    def __init__(self):
        super().__init__()
        self.setupUi(self)





class CalMain(QMainWindow):
    def __init__(self,config_file = None):
        super().__init__()





        self.setWindowTitle("Database Control")
        myLabel = QLabel("Disk Database program vs 1.0")
        myCloseButton = QPushButton("Close")
        myCloseButton.clicked.connect(self.close_app)
        layout= QVBoxLayout()
        layout.addWidget(myLabel)
        layout.addWidget(myCloseButton)


        widget = QWidget()
        widget.setLayout(layout)



        self.setCentralWidget(widget)
        self.show()

        self.config_file = config_file

        #setup menu
        menu = self.menuBar()

        file_menu = menu.addMenu("&Action")


   # Create a " config" action
        config_action = QAction( " Config File", self)
        config_action.setShortcut("Ctrl+F")
        config_action.setStatusTip("change config file")
        config_action.triggered.connect(self.SetupConfigNew)
        file_menu.addAction(config_action)

   # Create a " connect db" action
        db_action = QAction( " Connect Database", self)
        db_action.setStatusTip("connect database ")
        db_action.triggered.connect(self.connect_db)
        file_menu.addAction(db_action)

        #instantiate configuration

        self.SetupConfig()

        self.SetupLogger()




#        self.log_level = self.CM.log_level
#        self.ingred_table = self.CM.ingred_table


    #instantiate the disk_db class


    def connect_db(self):
        db_name     = self.CD.db_name
        db_user     = self.CD.db_user
        db_system   = self.CD.db_address
        db_pwd      = self.CD.db_pwd


        return

    def close_app(self):
        """Closes the program"""

        logger.info("closing down")
        self.close()


    def SetupConfig(self):
        mysystem = platform.system()

        # get the config filename
        # here we do a filedialog
        if(self.config_file == None or not os.path.isfile(self.config_file)):
            self.config_file , filter = QFileDialog.getOpenFileName(self,
                                        self.tr("Open Config file"), "~", self.tr("*.json"))


        logger.info("config file %s" % self.config_file)



        self.CD = CD.MyConfig(self.config_file)
        self.log_level = self.CD.log_level
        self.log_output = self.CD.log_output
        #reset self.config_file, so we can change it through the menu
        #self.config_file = None


    def SetupConfigNew(self):
        """ gest called when one ants to read in a new config file"""
        mysystem = platform.system()

        # get the config filename
        # here we do a filedialog
        self.config_file = None
        if(self.config_file == None or not os.path.isfile(self.config_file)):
            self.config_file , filter = QFileDialog.getOpenFileName(self,
                                        self.tr("Open Config file"), "~", self.tr("*.json"))


        logger.info("config file %s" % self.config_file)



        self.CD = CD.MyConfig(self.config_file)
        self.log_level = self.CD.log_level
        self.log_output = self.CD.log_output
        #reset self.config_file, so we can change it through the menu
        #self.config_file = None

```

```python


    def SetupLogger(self):


        logger.remove(0)
        #now we add color to the terminal output
        logger.add(sys.stdout,
                colorize = True,format="<green>{time}</green>   {function}  {line}   {level}    <level>{message}</level>" ,
                level = "DEBUG")



        fmt =  "{time} - {name}-  {function} -{line}- {level}   - {message}"
        logger.add('info.log', format = fmt , level = 'INFO',rotation="1 day")


        # set the colors of the different levels
        logger.level("INFO",color ='<black>')
        logger.level("WARNING",color='<green>')
        logger.level("ERROR",color='<red>')
        logger.level("DEBUG",color = '<blue>')

        return





if __name__ == "__main__":
    app = QApplication([])
    config_file = '/Users/klein/git/diskdb/config/config_disk_db.json'
    window = CalMain(config_file = config_file )
    window.show()
    app.exec()
```