**Question-5**

**Design and Code a simple clone app of Tesla Powerhub software based on Chromium Embedded Framework? Link**

**attached for reference - https://www.tesla.com/support/energy/tesla-software/powerhub.**

#include <iostream>

#include "include/cef\_app.h"

#include "include/cef\_browser.h"

#include "include/cef\_command\_line.h"

#include "include/cef\_frame.h"

#include "include/cef\_runnable.h"

#include "include/cef\_scheme.h"

#include "include/cef\_waitable\_event.h"

#include "include/wrapper/cef\_closure\_task.h"

#include "include/wrapper/cef\_helpers.h"

#include "include/wrapper/cef\_message\_router.h"

class TeslaPowerhubApp : public CefApp, public CefBrowserProcessHandler {

public:

TeslaPowerhubApp() {}

virtual CefRefPtr<CefBrowserProcessHandler>GetBrowserProcessHandler() OVERRIDE { return this; }

virtual void OnContextInitialized() OVERRIDE {

CefRefPtr<CefCommandLine>command\_line = CefCommandLine::GetGlobalCommandLine();

CefRefPtr<CefBrowser> browser = CefBrowserHost::CreateBrowserSync(

CefWindowInfo(), CefBrowserSettings(), "https://www.tesla.com/support/energy/tesla-software/powerhub", NULL);

}

IMPLEMENT\_REFCOUNTING(TeslaPowerhubApp);

};

int main(int argc, char\* argv[]) {

CefMainArgsmain\_args(argc, argv);

CefRefPtr<TeslaPowerhubApp>app(new TeslaPowerhubApp());

return CefExecuteProcess(main\_args, app.get(), NULL);

}

Compiling and executing a CEF-based application involves a bit more setup compared to a typical C or C++ program due to the dependency on the Chromium Embedded Framework. Here's a general guide on how to compile and execute the program:

**Step 1: Set Up CEF**

1.Download the CEF binary distribution from the official website: <https://cefbuilds.com/>.

2.Extract the downloaded archive to a directory on your system.

**Step 2: Set Up Development Environment**

1.Make sure you have a C++ compiler and CMake installed on your system.

2.Set up your project directory structure and create source files as per the example code provided earlier.

### Step 3: Configure CMakeLists.txt

Create a CMakeLists.txt file in the root directory of your project to define the build configuration.

Here's a basic example:

cmake\_minimum\_required(VERSION 3.10)

project(tesla\_powerhub\_clone)

set(CMAKE\_CXX\_STANDARD 17)

# Find CEF

set(CEF\_ROOT\_DIR "/path/to/cef\_binary")

find\_package(CEF REQUIRED)

include\_directories(${CEF\_INCLUDE\_DIRS})

add\_executable(${PROJECT\_NAME} src/main.cpp src/application.cpp src/browser\_window.cpp src/browser\_client.cpp src/resource\_handler.cpp)

target\_link\_libraries(${PROJECT\_NAME} ${CEF\_LIBRARIES})

Replace **/path/to/cef\_binary** with the actual path to the CEF binary distribution directory.

### Step 4: Build the Project

### Open a terminal or command prompt and navigate to your project directory. Then, run the following commands:

mkdir build

cd build

cmake ..

make

This will generate the build files and compile your application.

### Step 5: Execute the Program

Once the compilation is successful, you can execute the program by running:

./tesla\_powerhub\_clone

Make sure you have the necessary CEF libraries available in your system's library path or alongside your executable.