Common Gateway Interface With Tiny Database Engine

Names:

- Omnia Ahmed Bakr Sec (1)
- Aya Abo-Elela Khaled Sec (1)
- Omnia Mohamed Abd-Elaal Sec (1)
- Abdullah Mohamed Mahmoud Abu-Braik Sec (2)
- Omar Abd-Elnasser Ahmed Sec (2)

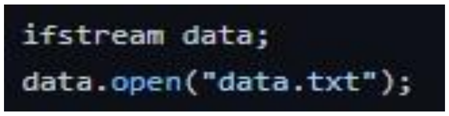
4th IS

Dr/ Ibrahim El-Semman

1. **We create text file data.txt and fill it with data contains table of student with data:**
   - ID
   - FName
   - LName
   - GPA
2. **We create project in visual studio and use C++.**

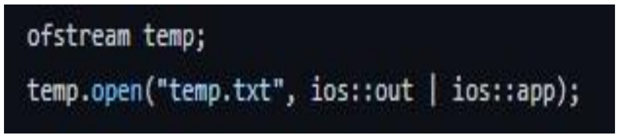
3. **We use file structure to deal with a file:**

   - **Open file :**



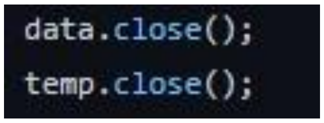
```
ifstream data;
data.open("data.txt");
```

   - **Open & Create New file:**



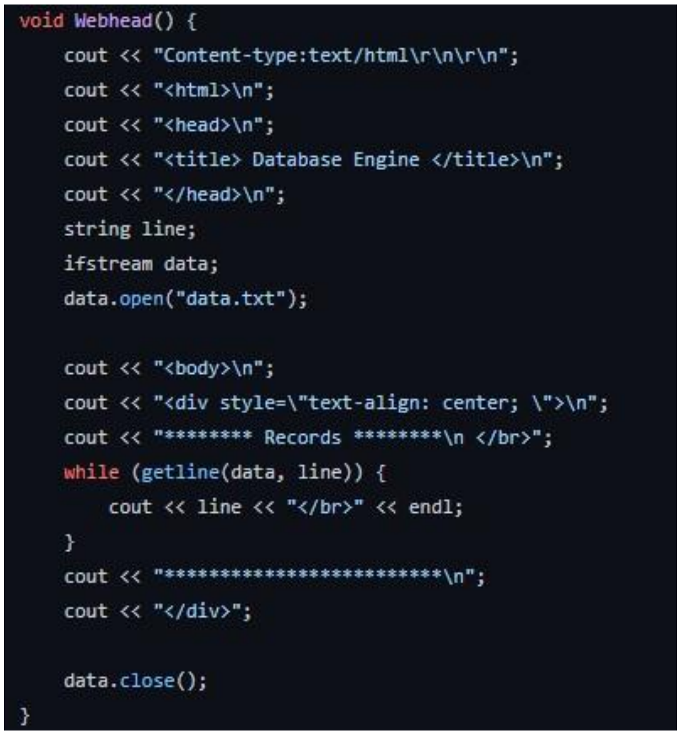
```
ofstream temp;
temp.open("temp.txt", ios::out | ios::app);
```

   - **Close file:**



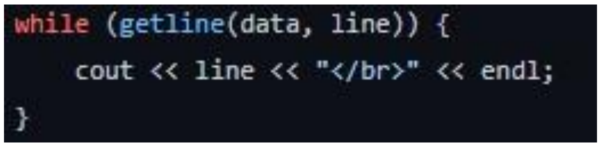
```
data.close();
temp.close();
```

4. We create a Webhead() function to execute HTML in web page:



```
void Webhead() {
    cout << "Content-type:text/html\r\n\r\n";
    cout << "<html>\n";
    cout << "<head>\n";
    cout << "<title> Database Engine </title>\n";
    cout << "</head>\n";
    string line;
    ifstream data;
    data.open("data.txt");

    cout << "<body>\n";
    cout << "<div style=\"text-align: center; \">\n";
    cout << "******** Records ********\n </br>";
    while (getline(data, line)) {
        cout << line << "</br>" << endl;
    }
    cout << "*************************\n";
    cout << "</div>";

    data.close();
}
```

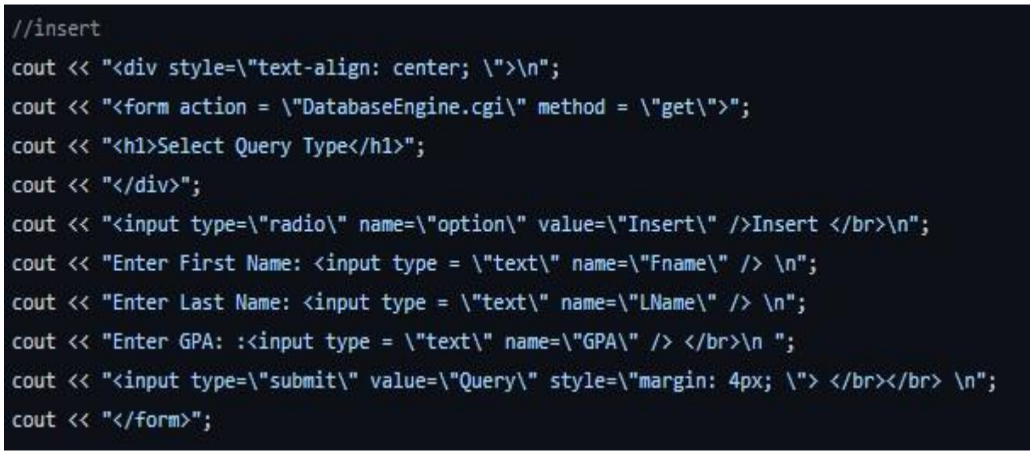
5. To show records of file we used While loop to get lines of file and show it in web page:



```
while (getline(data, line)) {
    cout << line << "</br>" << endl;
}
```

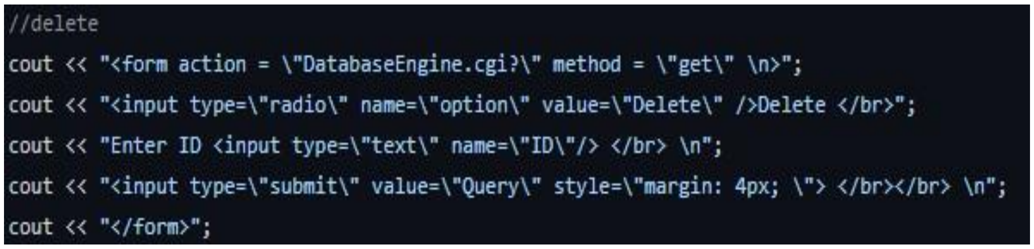
6. We have Three types of Query we create them in web page by using three forms:

- **Insert:**



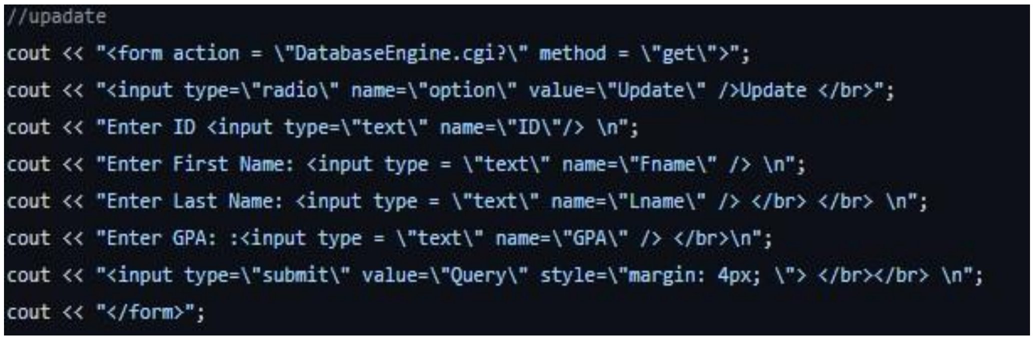
```
//insert
cout << "<div style=\"text-align: center; \">\n";
cout << "<form action = \"DatabaseEngine.cgi\" method = \"get\">";
cout << "<h1>Select Query Type</h1>";
cout << "</div>";
cout << "<input type=\"radio\" name=\"option\" value=\"Insert\" />Insert </br>\n";
cout << "Enter First Name: <input type = \"text\" name=\"Fname\" /> \n";
cout << "Enter Last Name: <input type = \"text\" name=\"LName\" /> \n";
cout << "Enter GPA: :<input type = \"text\" name=\"GPA\" /> </br>\n ";
cout << "<input type=\"submit\" value=\"Query\" style=\"margin: 4px; \"> </br></br> \n";
cout << "</form>";
```

- **Delete:**



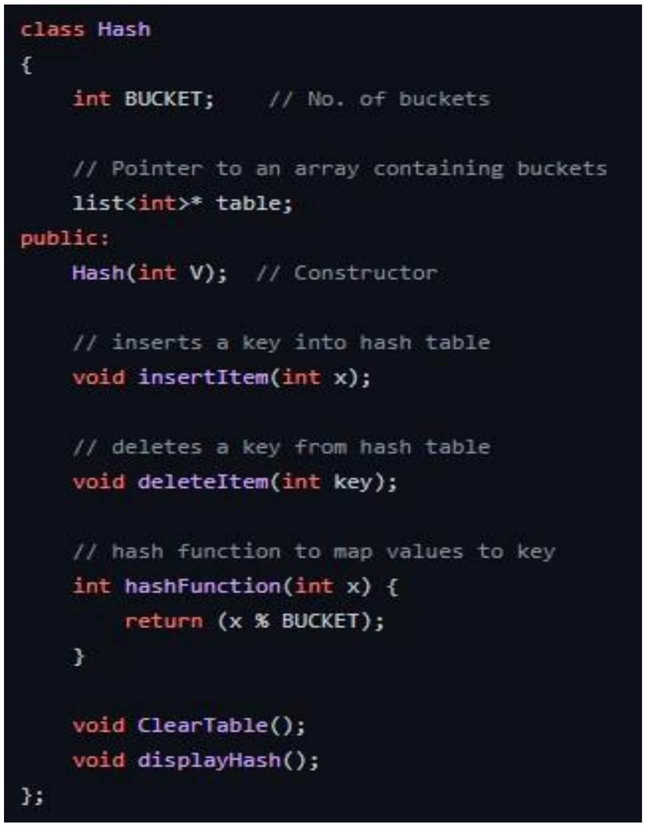
```
//delete
cout << "<form action = \"DatabaseEngine.cgi?\" method = \"get\" \n>";
cout << "<input type=\"radio\" name=\"option\" value=\"Delete\" />Delete </br>";
cout << "Enter ID <input type=\"text\" name=\"ID\"/> </br> \n";
cout << "<input type=\"submit\" value=\"Query\" style=\"margin: 4px; \"> </br></br> \n";
cout << "</form>";
```

- **Update:**



```
//upadate
cout << "<form action = \"DatabaseEngine.cgi?\" method = \"get\">";
cout << "<input type=\"radio\" name=\"option\" value=\"Update\" />Update </br>";
cout << "Enter ID <input type=\"text\" name=\"ID\"/> \n";
cout << "Enter First Name: <input type = \"text\" name=\"Fname\" /> \n";
cout << "Enter Last Name: <input type = \"text\" name=\"Lname\" /> </br> </br> \n";
cout << "Enter GPA: :<input type = \"text\" name=\"GPA\" /> </br>\n";
cout << "<input type=\"submit\" value=\"Query\" style=\"margin: 4px; \"> </br></br> \n";
cout << "</form>";
```

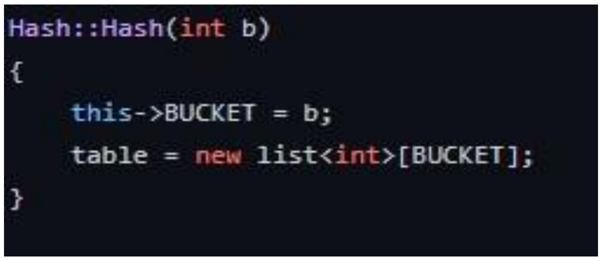
7. **We create a Hash class to make a hash table of our data:**



```cpp
class Hash
{
    int BUCKET;    // No. of buckets

    // Pointer to an array containing buckets
    list<int>* table;
public:
    Hash(int V);  // Constructor

    // inserts a key into hash table
    void insertItem(int x);

    // deletes a key from hash table
    void deleteItem(int key);

    // hash function to map values to key
    int hashFunction(int x) {
        return (x % BUCKET);
    }

    void ClearTable();
    void displayHash();
};
```

, then we create constructor:

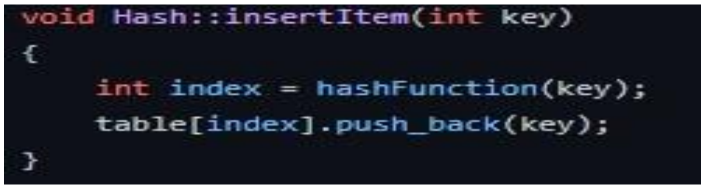
- **Hash():**



```cpp
Hash::Hash(int b)
{
    this->BUCKET = b;
    table = new list<int>[BUCKET];
}
```

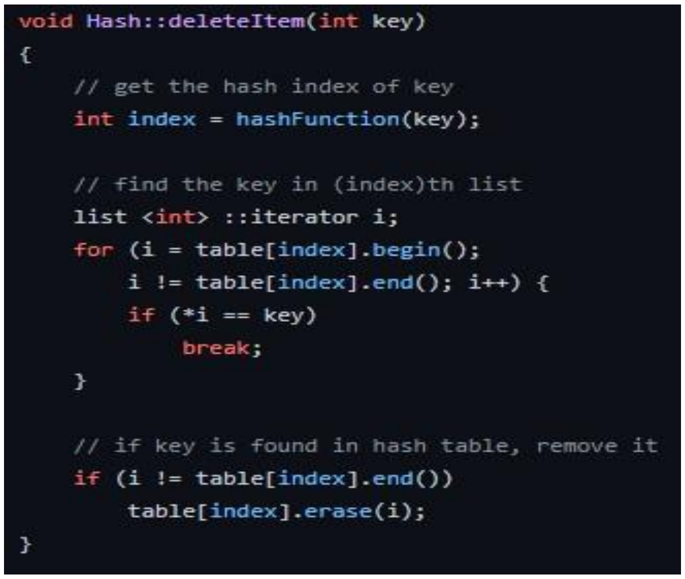
, and three methods:

- **insertItem():**



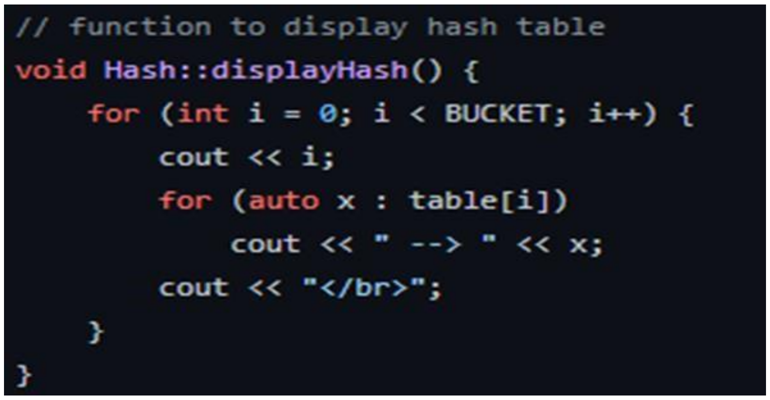
```cpp
void Hash::insertItem(int key)
{
    int index = hashFunction(key);
    table[index].push_back(key);
}
```

- **deleteItem():**



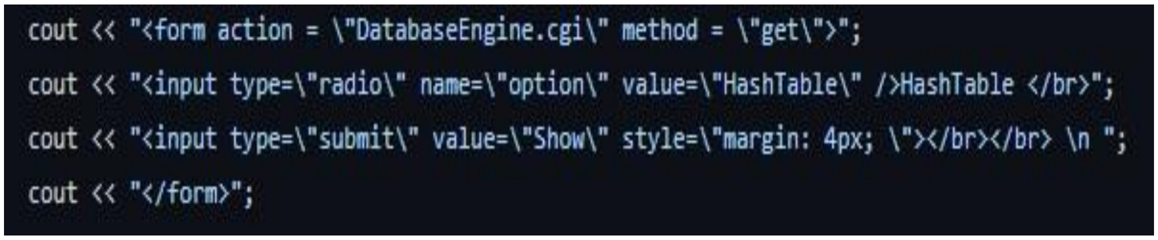
```
void Hash::deleteItem(int key)
{
    // get the hash index of key
    int index = hashFunction(key);

    // find the key in (index)th list
    list <int> ::iterator i;
    for (i = table[index].begin();
        i != table[index].end(); i++) {
        if (*i == key)
            break;
    }

    // if key is found in hash table, remove it
    if (i != table[index].end())
        table[index].erase(i);
}
```

- **displayHash():**



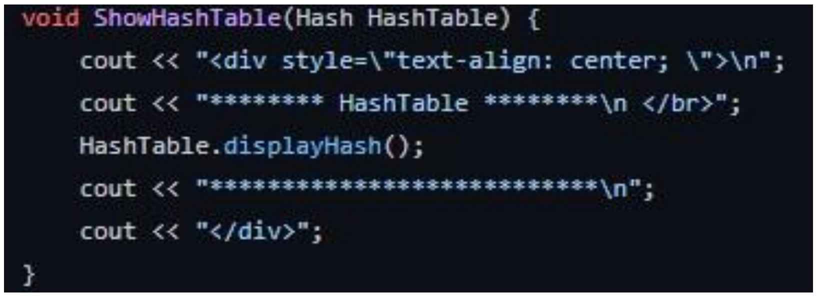
```
// function to display hash table
void Hash::displayHash() {
    for (int i = 0; i < BUCKET; i++) {
        cout << i;
        for (auto x : table[i])
            cout << " --> " << x;
        cout << "</br>";
    }
}
```

8. We create another Form for Hashing :



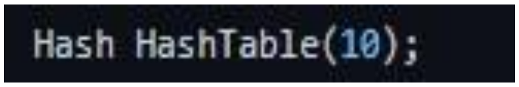
```
cout << "<form action = \"DatabaseEngine.cgi\" method = \"get\">";
cout << "<input type=\"radio\" name=\"option\" value=\"HashTable\" />HashTable </br>";
cout << "<input type=\"submit\" value=\"Show\" style=\"margin: 4px; \"></br></br> \n ";
cout << "</form>";
```

9. To show Hashing table we create ShowHashTable() function and used within it displayHash() function:



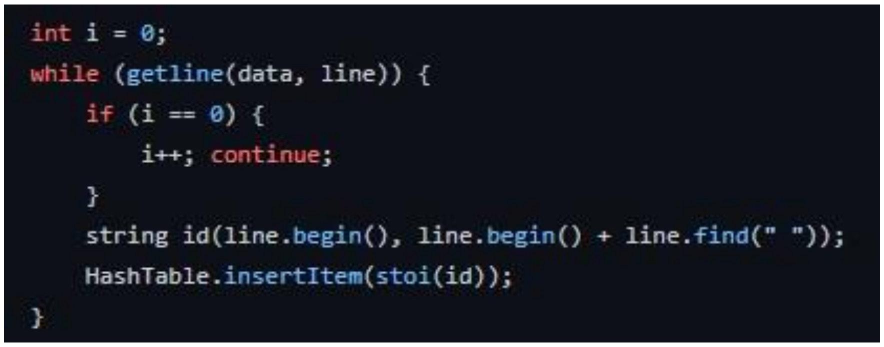
```
void ShowHashTable(Hash HashTable) {
    cout << "<div style=\"text-align: center; \">\n";
    cout << "********* HashTable ********\n </br>";
    HashTable.displayHash();
    cout << "****************************\n";
    cout << "</div>";
}
```

10. We create Hash Table object from class Hash and passing 10 buckets to constructor:



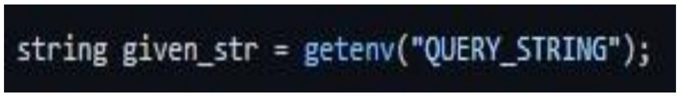
```
Hash HashTable(10);
```

11. At the Beginning of the program, we insert data from file into Hash Table created in previous step:



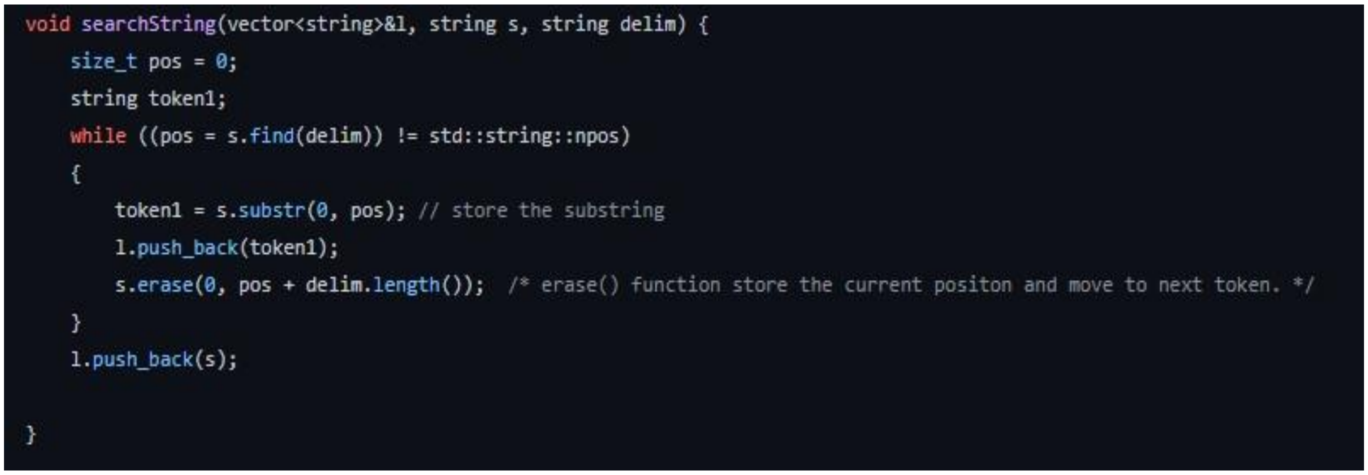
```
int i = 0;
while (getline(data, line)) {
    if (i == 0) {
        i++; continue;
    }
    string id(line.begin(), line.begin() + line.find(" "));
    HashTable.insertItem(stoi(id));
}
```

12. When user select Type of query from web page we call getenv() function to access QUERY_STRING from URL:



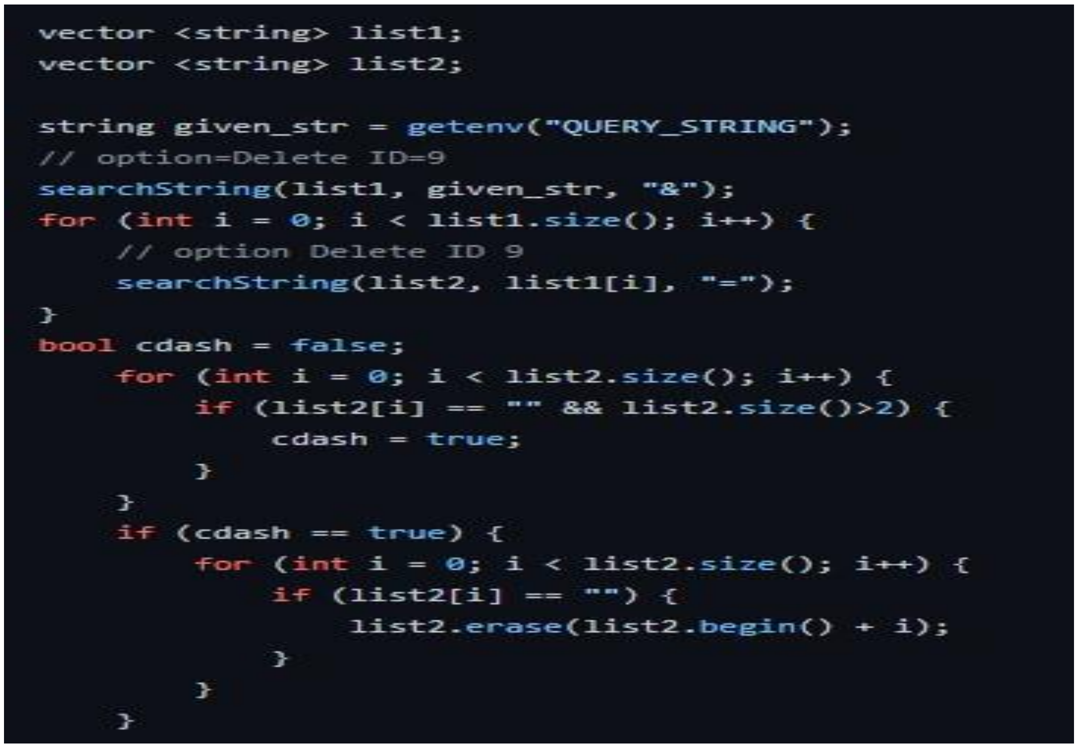
```
string given_str = getenv("QUERY_STRING");
```

13. Then we create searchString() function to store QUERY_STRING in vector:



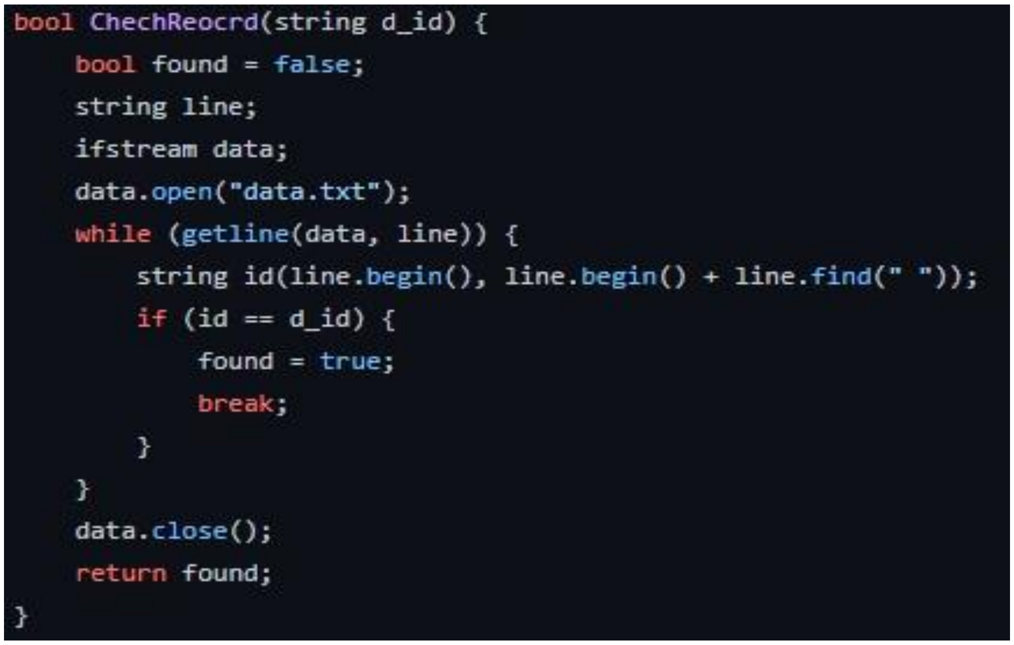
```
void searchString(vector<string>&l, string s, string delim) {
    size_t pos = 0;
    string token1;
    while ((pos = s.find(delim)) != std::string::npos)
    {
        token1 = s.substr(0, pos); // store the substring
        l.push_back(token1);
        s.erase(0, pos + delim.length());  /* erase() function store the current positon and move to next token. */
    }
    l.push_back(s);

}
```

14. We used a function that in previous step to split QUERY_STRING based on delimiter into two vectors and remove an empty value from URL:



```
vector <string> list1;
vector <string> list2;

string given_str = getenv("QUERY_STRING");
// option=Delete ID=9
searchString(list1, given_str, "&");
for (int i = 0; i < list1.size(); i++) {
    // option Delete ID 9
    searchString(list2, list1[i], "=");
}
bool cdash = false;
    for (int i = 0; i < list2.size(); i++) {
        if (list2[i] == "" && list2.size()>2) {
            cdash = true;
        }
    }
    if (cdash == true) {
        for (int i = 0; i < list2.size(); i++) {
            if (list2[i] == "") {
                list2.erase(list2.begin() + i);
            }
        }
    }
```

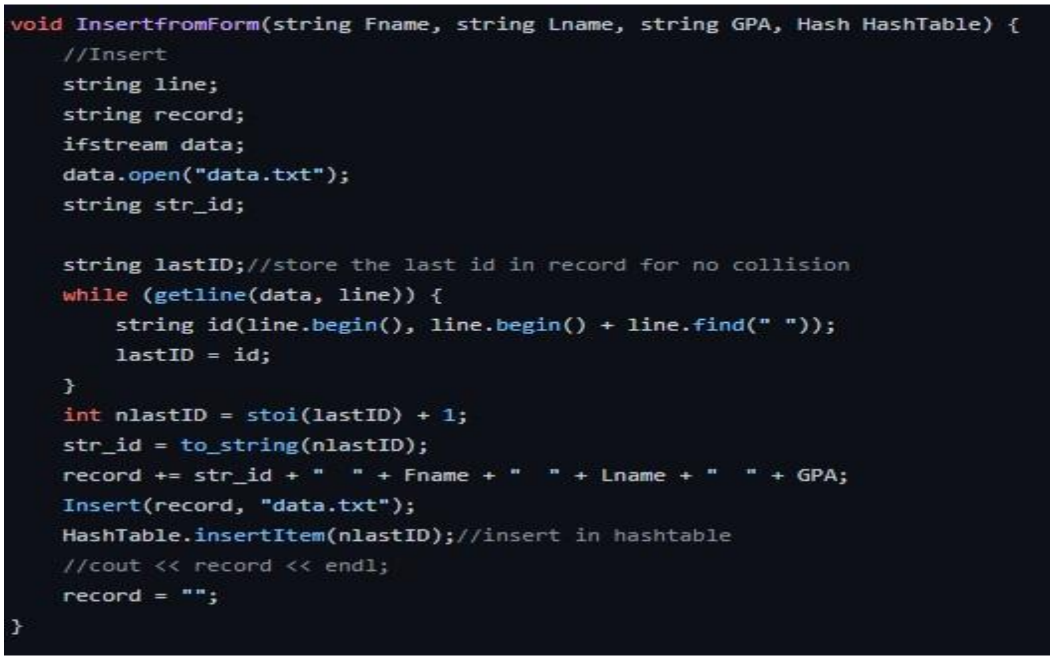
15. **We create CheckRecord() function to Check if record is exist or not:**



```
bool ChechReocrd(string d_id) {
    bool found = false;
    string line;
    ifstream data;
    data.open("data.txt");
    while (getline(data, line)) {
        string id(line.begin(), line.begin() + line.find(" "));
        if (id == d_id) {
            found = true;
            break;
        }
    }
    data.close();
    return found;
}
```

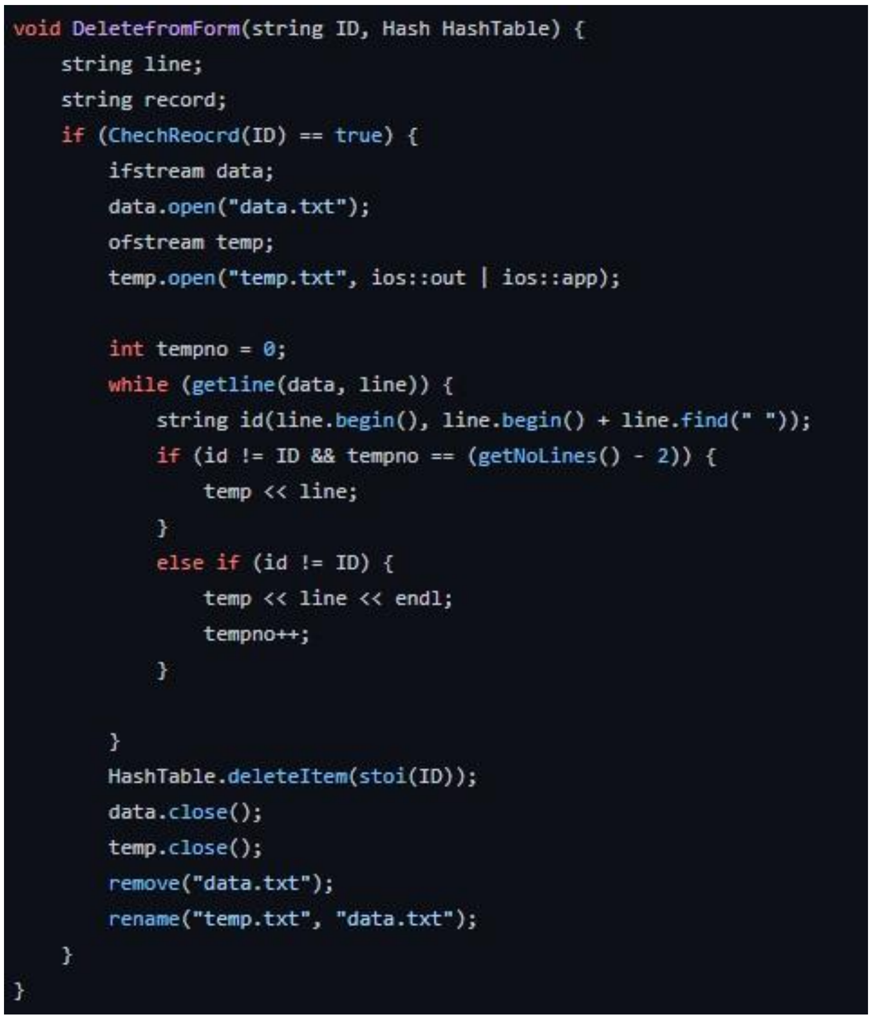
16. **Then we create three function to call them after checked type of query:**

- **InsertfromForm():**



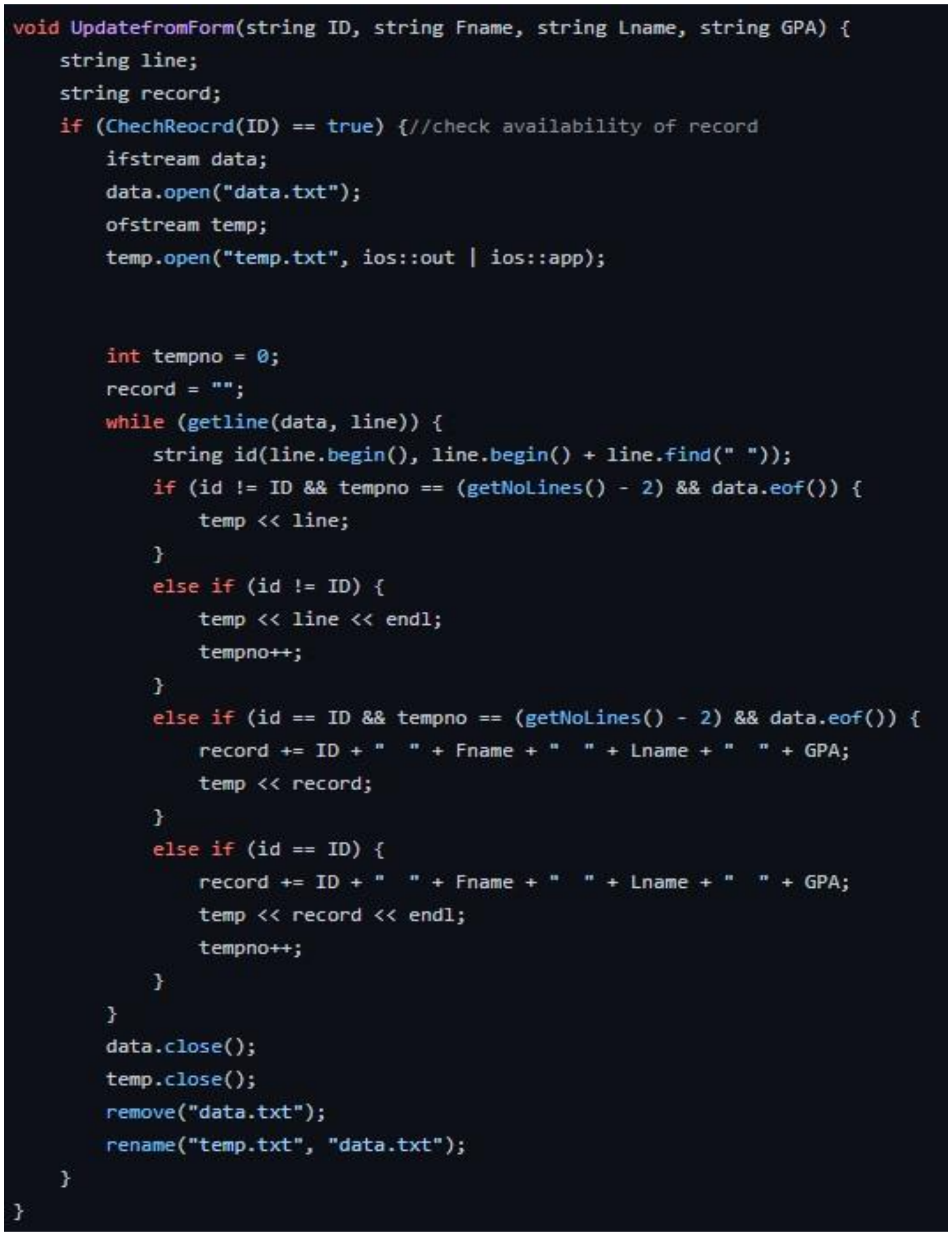
```
void InsertfromForm(string Fname, string Lname, string GPA, Hash HashTable) {
    //Insert
    string line;
    string record;
    ifstream data;
    data.open("data.txt");
    string str_id;

    string lastID;//store the last id in record for no collision
    while (getline(data, line)) {
        string id(line.begin(), line.begin() + line.find(" "));
        lastID = id;
    }
    int nlastID = stoi(lastID) + 1;
    str_id = to_string(nlastID);
    record += str_id + "   " + Fname + "   " + Lname + "   " + GPA;
    Insert(record, "data.txt");
    HashTable.insertItem(nlastID);//insert in hashtable
    //cout << record << endl;
    record = "";
}
```

- **DeletefromForm():**



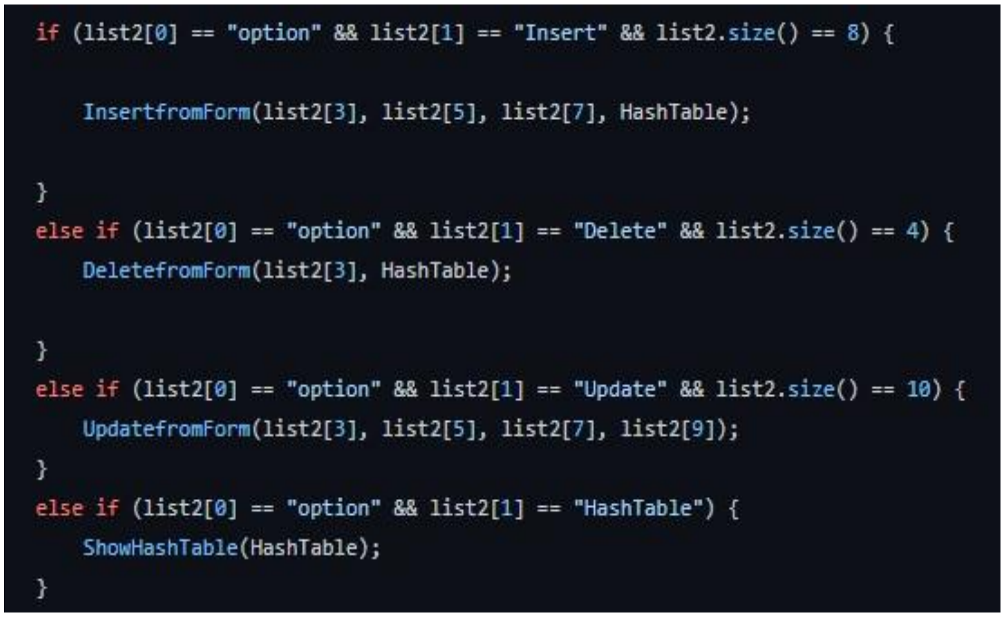
```
void DeletefromForm(string ID, Hash HashTable) {
    string line;
    string record;
    if (ChechReocrd(ID) == true) {
        ifstream data;
        data.open("data.txt");
        ofstream temp;
        temp.open("temp.txt", ios::out | ios::app);

        int tempno = 0;
        while (getline(data, line)) {
            string id(line.begin(), line.begin() + line.find(" "));
            if (id != ID && tempno == (getNoLines() - 2)) {
                temp << line;
            }
            else if (id != ID) {
                temp << line << endl;
                tempno++;
            }

        }
        HashTable.deleteItem(stoi(ID));
        data.close();
        temp.close();
        remove("data.txt");
        rename("temp.txt", "data.txt");
    }
}
```

• UpdatefromForm():



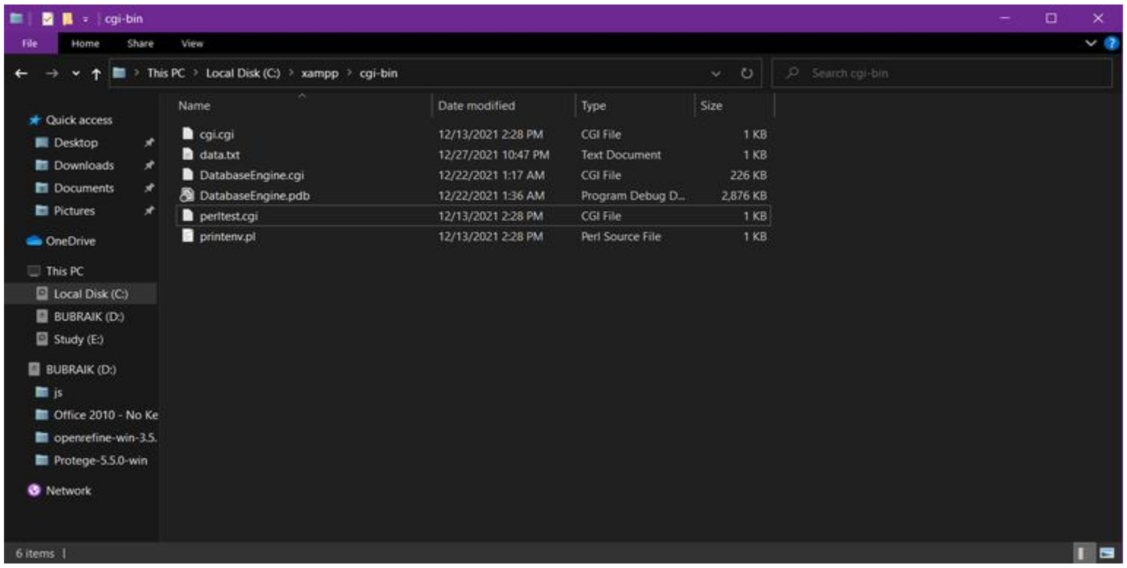
```cpp
void UpdatefromForm(string ID, string Fname, string Lname, string GPA) {
    string line;
    string record;
    if (ChechReocrd(ID) == true) {//check availability of record
        ifstream data;
        data.open("data.txt");
        ofstream temp;
        temp.open("temp.txt", ios::out | ios::app);


        int tempno = 0;
        record = "";
        while (getline(data, line)) {
            string id(line.begin(), line.begin() + line.find(" "));
            if (id != ID && tempno == (getNoLines() - 2) && data.eof()) {
                temp << line;
            }
            else if (id != ID) {
                temp << line << endl;
                tempno++;
            }
            else if (id == ID && tempno == (getNoLines() - 2) && data.eof()) {
                record += ID + "   " + Fname + "   " + Lname + "   " + GPA;
                temp << record;
            }
            else if (id == ID) {
                record += ID + "   " + Fname + "   " + Lname + "   " + GPA;
                temp << record << endl;
                tempno++;
            }
        }
        data.close();
        temp.close();
        remove("data.txt");
        rename("temp.txt", "data.txt");
    }
}
```

17. Last step we check type of query from vector which stored parameters of QUERY_STRING then we call a function based on the type of query:



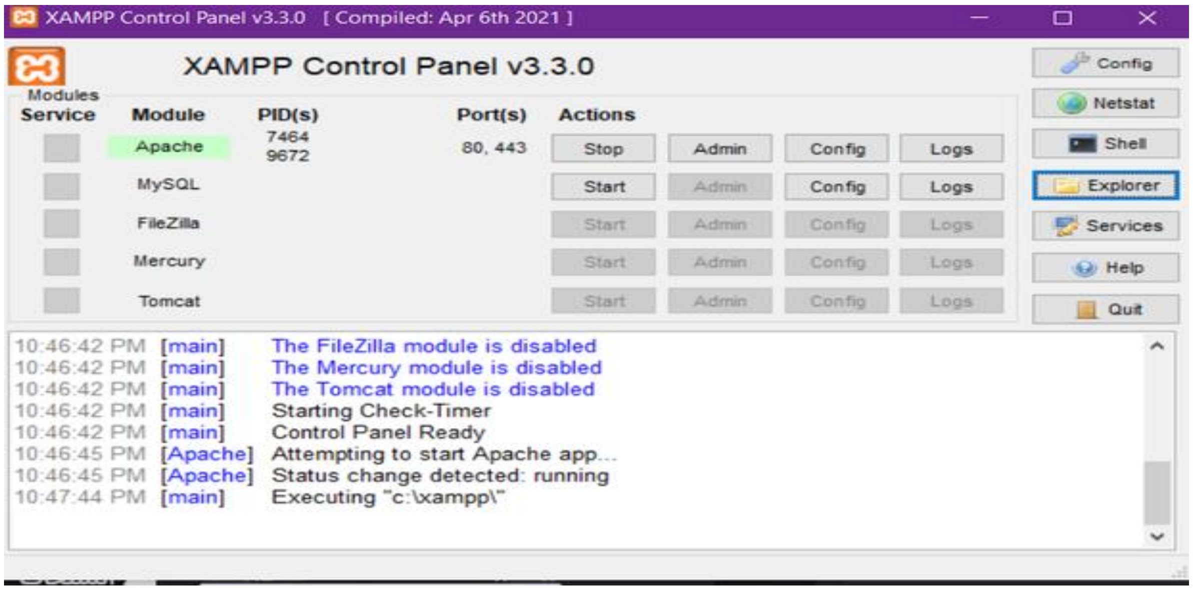
```
if (list2[0] == "option" && list2[1] == "Insert" && list2.size() == 8) {

    InsertfromForm(list2[3], list2[5], list2[7], HashTable);

}
else if (list2[0] == "option" && list2[1] == "Delete" && list2.size() == 4) {
    DeletefromForm(list2[3], HashTable);

}
else if (list2[0] == "option" && list2[1] == "Update" && list2.size() == 10) {
    UpdatefromForm(list2[3], list2[5], list2[7], list2[9]);
}
else if (list2[0] == "option" && list2[1] == "HashTable") {
    ShowHashTable(HashTable);
}
```

18. To show web page on browser, get into xampp folder and select the cgi- bin folder and paste you CGI folder there:



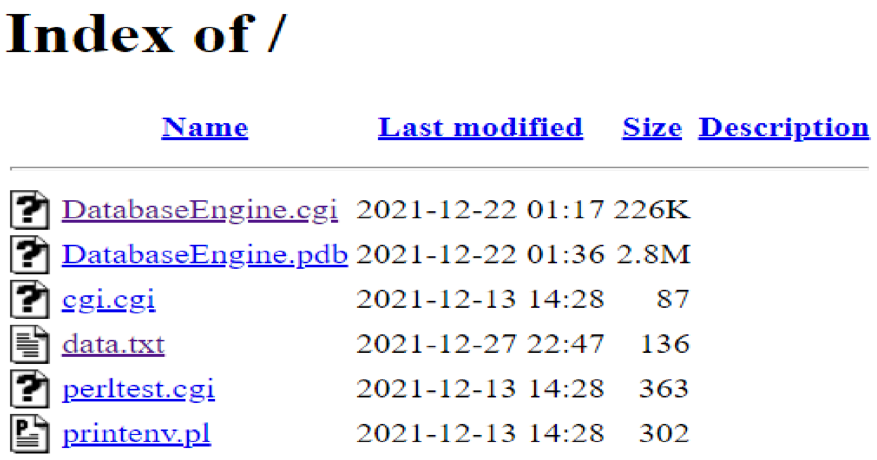
Name	Date modified	Type	Size
cgi.cgi	12/13/2021 2:28 PM	CGI File	1 KB
data.txt	12/27/2021 10:47 PM	Text Document	1 KB
DatabaseEngine.cgi	12/22/2021 1:17 AM	CGI File	226 KB
DatabaseEngine.pdb	12/22/2021 1:36 AM	Program Debug D...	2,876 KB
perltest.cgi	12/13/2021 2:28 PM	CGI File	1 KB
printenv.pl	12/13/2021 2:28 PM	Perl Source File	1 KB

- After that, click admin



```
10:46:42 PM  [main]    The FileZilla module is disabled
10:46:42 PM  [main]    The Mercury module is disabled
10:46:42 PM  [main]    The Tomcat module is disabled
10:46:42 PM  [main]    Starting Check-Timer
10:46:42 PM  [main]    Control Panel Ready
10:46:45 PM  [Apache]  Attempting to start Apache app...
10:46:45 PM  [Apache]  Status change detected: running
10:47:44 PM  [main]    Executing "c:\xampp\"
```

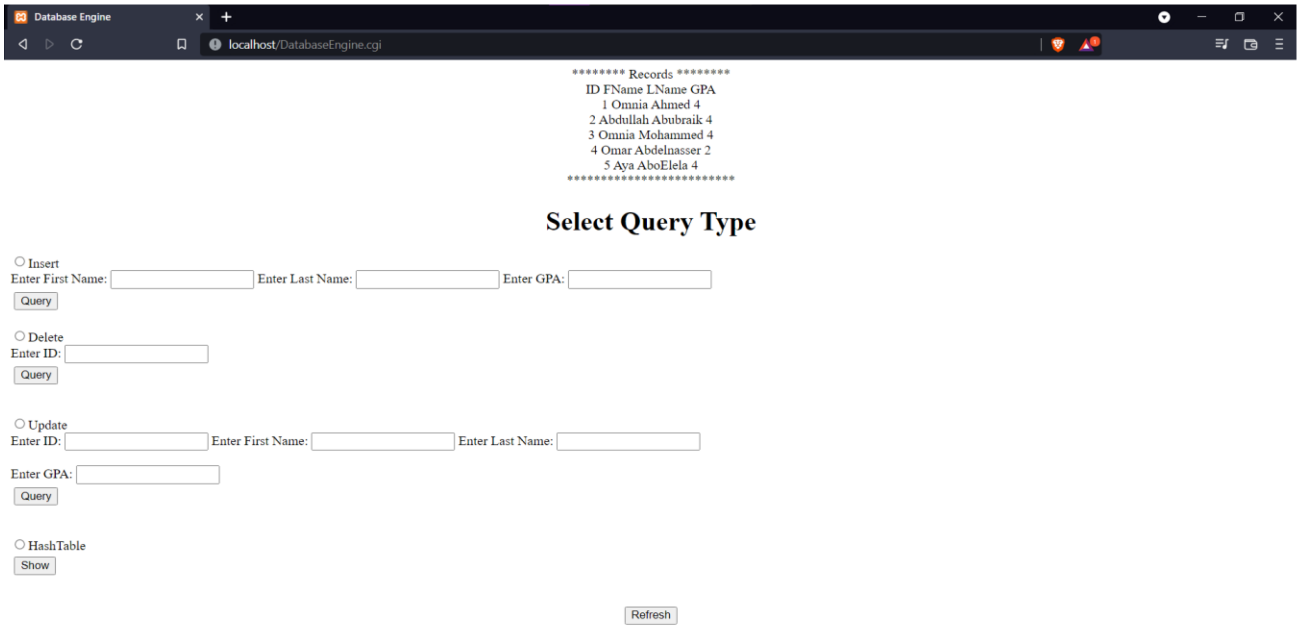
- Choose your CGI folder



Index of /

Name	Last modified	Size	Description
DatabaseEngine.cgi	2021-12-22 01:17	226K	
DatabaseEngine.pdb	2021-12-22 01:36	2.8M	
cgi.cgi	2021-12-13 14:28	87	
data.txt	2021-12-27 22:47	136	
perltest.cgi	2021-12-13 14:28	363	
printenv.pl	2021-12-13 14:28	302	

- **Finally, The Web Page Will Open.**



```
******** Records ********
ID FName LName GPA
1 Omnia Ahmed 4
2 Abdullah Abubraik 4
3 Omnia Mohammed 4
4 Omar Abdelnasser 2
5 Aya AboElela 4
*************************
```

Select Query Type

○ Insert
Enter First Name: Enter Last Name: Enter GPA:
Query

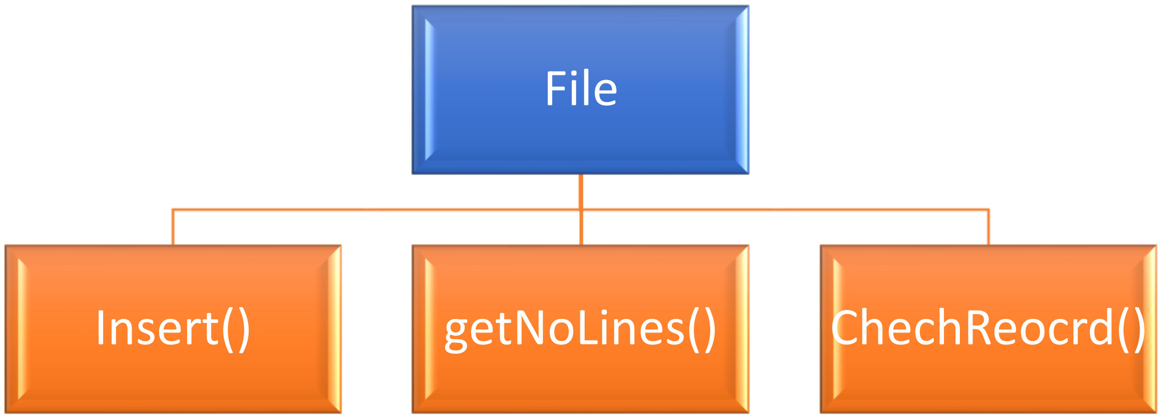
○ Delete
Enter ID:
Query

○ Update
Enter ID: Enter First Name: Enter Last Name:
Enter GPA:
Query

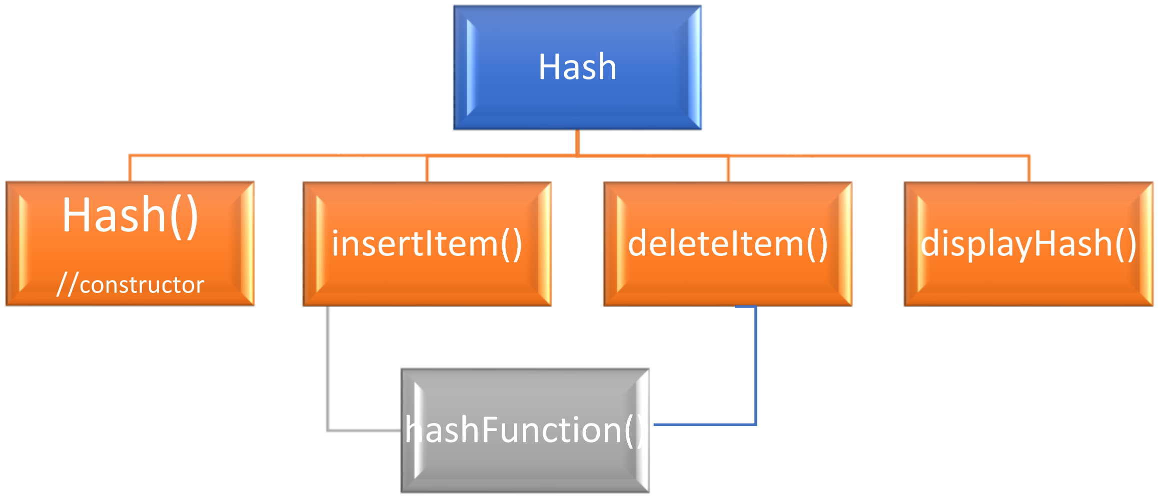
○ HashTable
Show

Refresh

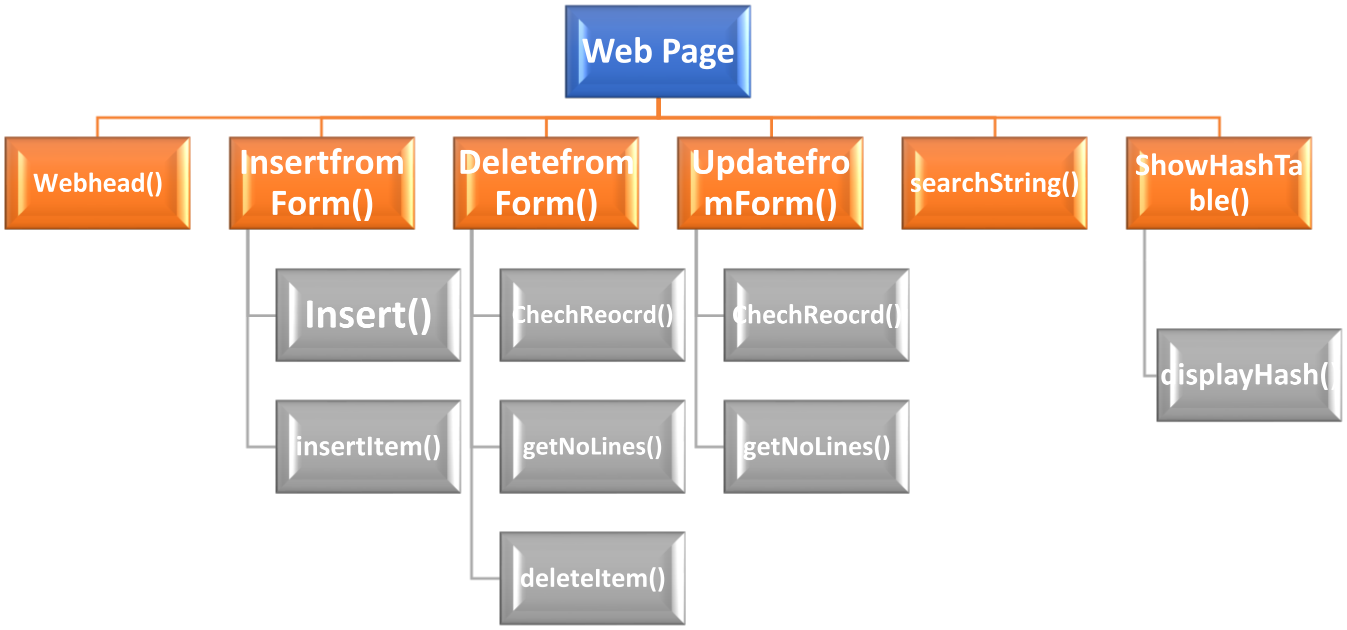
➢ Functions used in file:

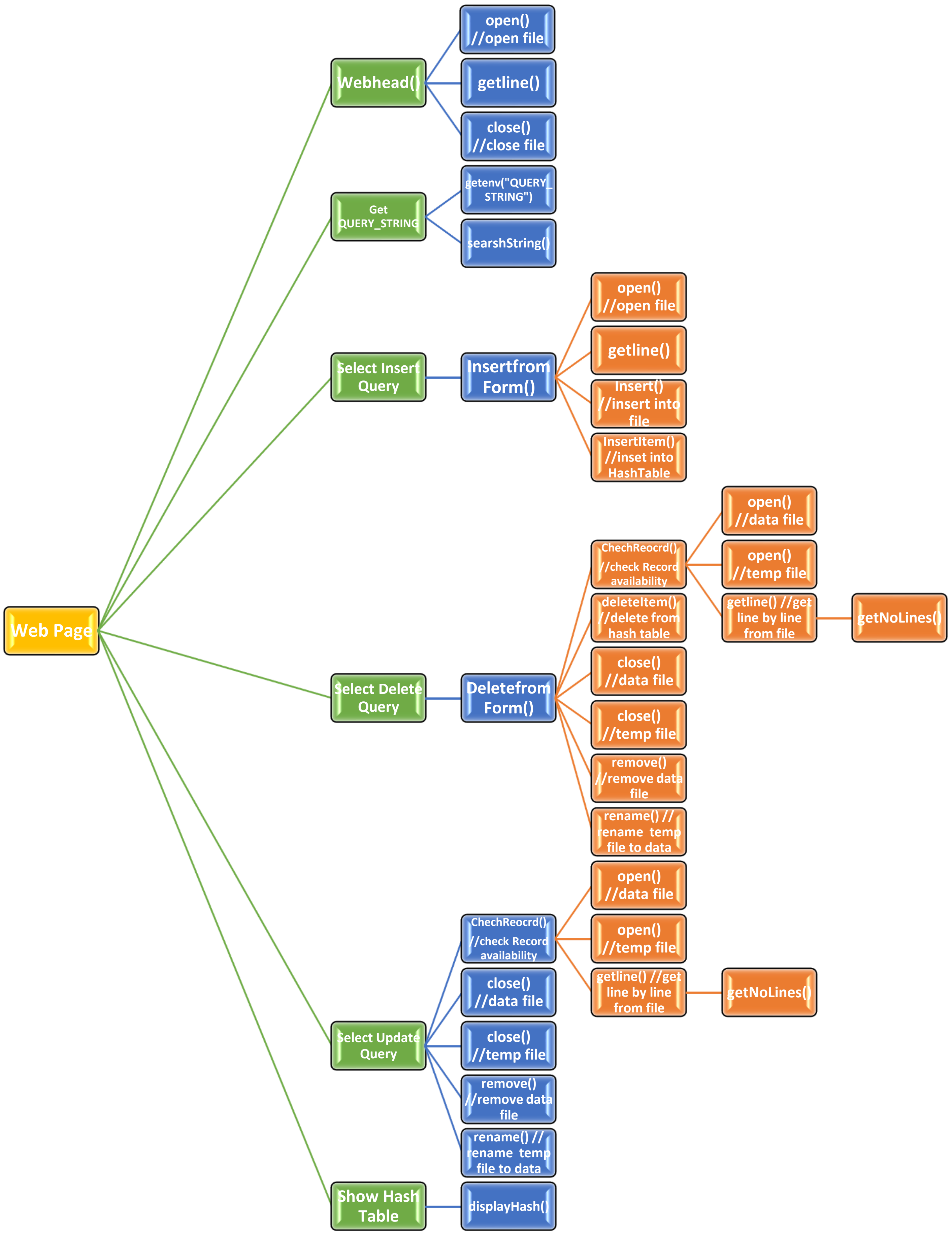


➢ Functions used in Hash:



➢ Functions in Web Page:




Web Page
Webhead()
open() //open file
getline()
close() //close file
Get QUERY_STRING
getenv("QUERY_STRING")
searshString()
Select Insert Query
Insertfrom Form()
open() //open file
getline()
Insert() //insert into file
InsertItem() //inset into HashTable
Select Delete Query
Deletefrom Form()
ChechReocrd() //check Record availability
open() //data file
open() //temp file
getline() //get line by line from file
getNoLines()
deleteItem() //delete from hash table
close() //data file
close() //temp file
remove() //remove data file
rename() // rename temp file to data
Select Update Query
ChechReocrd() //check Record availability
open() //data file
open() //temp file
getline() //get line by line from file
getNoLines()
close() //data file
close() //temp file
remove() //remove data file
rename() // rename temp file to data
Show Hash Table
displayHash()

Requirements:

1. XAMPP sever
2. Visual Studio
3. C++
4. Any Browser