**Search Engine**

**Members**: Taha Sohail (19L-2241), Hussnain (19L-2260)

**Taha’s Code:**

The code is properly commented so kindly refer if something is not clear.

🡪**Functions of “Helper.h”:**

**void login(HashTable\*& table);**

This function is to login already existing users. I differentiates between admin and normal user and directs them to their respective settings menu and makes relative entry in “History.txt”. Exceptions such as char input in int has been handled. This function searches hash table to find username and password.

**void newUser(HashTable\*& table);**

This function creates a new user by taking input of the credentials, the username must be unique, then inputting them into a file. All exceptions such as character input in passcode<int>, similar username and both passcode not similar have been handled. This function makes an entry in hash table.

**void forgot(HashTable\*& table);**

If a user has forgotten his password he/she will enter username to recover password. This is done by searching the non-empty locations of table. This function is not efficient in terms of time complexity but has practical uses.

**void mainMenu(HashTable\*& table);**

This function creates a menu in which user can select to login, make new account or recover forgotten password. Exceptions have been handled.

**void loadScr(HashTable\*& table);**

This function is called by mainMenu() and is responsible for updating hash table on runtime in case of new users.

**bool checkSimilarity(int pass1, int pass2);**

A function to check if both passcodes are similar.

**void loginSystem();**

Created a hash table and calls the mainMenu() function. The function itself is called in the start page in “Main.cpp”.

**bool check(int arr1[], int arr2[]);**

A function to check if both arrays are similar.

**void adminSettings(HashTable\*& table, Admin& admin);**

This function prints and calls the relative setting if a user has logged into the application. As all above functions exceptions have been handled.

**🡪Functions of Admin.h and Admin.cpp**:

Proper comments have been added in code.

**class User {**

**string username;**

**int passcode;**

**string full\_name;**

**int access\_lvl[4]; // [admin access(0), read access(1), write access(1), status(1)]= normal user**

**// [admin access(1), read access(1), write access(1), status(1)]= admins**

**public:**

**User();**

**string getUsername(); //------**

**int getPasscode(); // Getter functions**

**string getFullName(); //**

**int\* getAccessLvl(); //--**

**void setUsername(string str); //------**

**void setPasscode(int pass); // Setter functions**

**void setFullName(string str); //**

**void setAccessLvl(int arr[4]); //--**

**User& operator=(User& obj);**

**bool operator!=(User& obj);**

**friend ostream& operator<<(ostream& out, const User& obj);**

**};**

**The hash table is custom coded.**

**struct Node { // Node for hash table**

**int key;**

**User value;**

**Node();**

**Node(int key);**

**Node(int p\_key, User p\_value);**

**};**

**class HashTable { // Hash table with linear probing ( modifications to class code )**

**int capacity;**

**int table\_size;**

**Node\*\* table;**

**Node\* dummy; // To store in place of deleted value**

**public:**

**HashTable();**

**int hashFunc(int p\_key);**

**int sizeOfTable();**

**bool isEmpty();**

**void printTable();**

**void insert(int p\_key, User p\_value);**

**User remove(int p\_key);**

**User search(int p\_key);**

**User forgotten(string p\_username);**

**};**

**class Admin : protected User {**

**public:**

**Admin(User p\_user); // Constructor**

**void viewHistory(); // Only admin can look at search history**

**//void changeAccess(User& p\_user); // Alterate the access of a user**

**};**

**Hussnain’s Code:**

**Dictionary APP**

void insertWord(unordered\_map<string, Meaning>& umap) :

This function will insert data from file to the hash table.

void deleteWord(unordered\_map<string, Meaning>& umap):

This function can delete the word. Only Admin is responsible for that.

void updateWord(unordered\_map<string, Meaning>& umap) :

This function will change the meaning, synonyms and antonyms of the word. Only Admin is responsible for that.

void displayWord():

This function will display all the words in the dictionary on the screen.

void composeWord(unordered\_map<string, Meaning>& umap):

This function will insert a new word in the dictionary with its meaning synonyms and antonyms.

void searchWord(unordered\_map<string, Meaning> umap):

This function can search a word’s meaning, synonyms and antonyms from the dictionary.

**Blog APP**

void insertBlog(unordered\_map<string, string>& umap)

This function will insert data from file to the hash table.

void displayBlog():

This function will display all the articles in the Blog app on the screen.

void deleteBlog(unordered\_map<string, string>& umap):

This function can delete the article. Only Admin is responsible for that.

void updateBlog(unordered\_map<string, string>& umap)

This function will change the text in the article. Only Admin is responsible for that.

void composeBlog()

This function will insert a new article in the Blog app.

void searchBlog(unordered\_map<string, string> umap)

This function can search an article from the blog app.