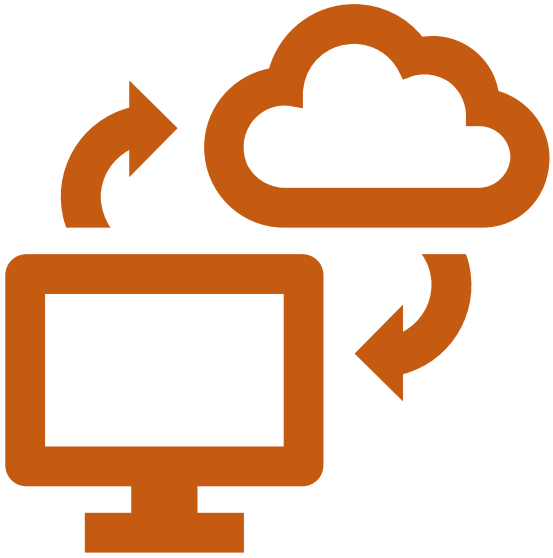
REPORT ON
DICTIONARY



Group ID: 7
Couse name: CS163 – Data Structures
Class name: 22CTT1
Date: 22/8/2023

Group member:

1) **Nguyen Bach Truong Giang**
2) **Le Van Cuong**
3) **Lam Tu Nhi**
4) **Do Huynh Diem Uyen**

Table of Contents

1) Abstract:

In an increasingly globalizing world, the ability to efficiently communicate with people from different cultures is highly important. Therefore, it is of convenience to have a portable digital dictionary so that language barriers can be easily overcome. And this is what this report is about: detailed information about one such cross-platform application.

This dictionary can translate from English to English, Vietnamese to English and vice versa. It can also give explanations for slang and emojis in both mentioned languages. In addition, it provides users with flexible features enabling them to customize their dictionary based on the original given dataset. This will greatly improve users' experience and make their learning process more personalized.

2) Introduction:

According to The British Council, English is now the world's most widely spoken language. It shows that the ability to use English, either natively or as a second language, will broaden individuals' opportunity when entering the global market. However, in a Ho Chi Minh Communist Youth Union's article, in 2018, only 48,3% of Vietnamese students met the English level required to graduate.

With that being said, one of the obstacles when learning a new language is the vocabulary. Because of this, the dictionary has always been a loyal friend to language learners. And it is even more beneficial if people can access it anywhere and anytime.

Because of all mentioned above, this project was born. With an aim to provide an accessible, user-friendly, and fun means to upgrade the learning experience, our dictionary comes along with several interesting functions that can transform how we view language-learning.

As fanatics of linguistics ourselves, we started off with what we know best: dictionaries currently available on the market. Our group pinpointed the primary problems we have found during our time using these tools and made it our goal to solve them. One of the biggest irks we have with digital dictionaries is the unintuitive engineering. With prominent dictionaries - such as Oxford or Cambridge, it is uncommon to see much-needed functions like edit mode and adding new words incorporated. However, these are crucial to the learning process, especially for linguistics since languages are ever-changing.

Moreover, from the perspective of more-than-10-year English learners, the application's design does affect our motivation. With the inherent tediousness that comes with language learning in general, an intriguing interface is bound to be a surefire way to catch the attention of learners, either beginners or veterans.

In short, our optimal goal is to make learning more enjoyable for everyone with technology.

3) Group Information:

Group 7's members:

- Lê Văn Cường - backend developer, data receiver
- Nguyễn Bạch Trường Giang - frontend co-developer, report writer, data receiver
- Lâm Tú Nhi - interface designer, frontend co-developer, tester, report brainstormer, data receiver
- Đỗ Huỳnh Diễm Uyên - background designer, frontend co-developer

Each member's tasks:

1. Lê Văn Cường:
    - Build backend function
    - Find and format dataset
    - Merge backend function to GUI
    - Add sound system
    - Favorite & History GUI building

2. Nguyễn Bạch Trường Giang:
    - Build frontend foundation
    - Develop welcome window
    - Co-build dictionary window’s layout base
    - Write report

3. Lâm Tú Nhi:
    - Design layout
    - Co-build dictionary window's layout base
    - Implement and fine-tune interface
    - Write report (introduction)
    - Main tester

4. Đỗ Huỳnh Diễm Uyên:
    - Design background for login window
    - Implement and fine-tune interface

4) Data Storage:

Let’s consider for only one data set.

For each word, there are two elements: word and definition.

We store each pair in a separate file, and use the word as key. In other word, we map each word to a number, and store the definition in its respective file.

To get the number fast, we can use a ternary search tree. This data structure is similar to both a trie and a binary search tree. At the node that mark the end of a word, we store the number there.

5) Project Architecture:

There are only 4 folder that play a meaningful role in our program, most other is just unused data. They are:

- databank: Store each data set
- coreData: Store important information
- debugLog: Not that important for user, but invaluable for developer in case thing go wrong or just for checking behavior
- sound: Store sound, pretty self-explanatory.

As I have mentioned (4), the core data structure that was used is a ternary search tree. This can be considered to be a fusion of trie and binary search tree. This is faster than a binary searching tree and more space-efficient than a trie in string searching problem, yet slower than a trie and uses more space than a binary search tree. As such, this is the perfect balance between speed and space, in line with our development philosophies in which we accept a acceptable slower speed in exchange for using less resources.

Still, the difference in speed of trie makes it a more favorable choice in most cases for string searching.

However, most is not all.

Through our research, a ternary search tree (TST) has one benefit over trie that led to our decision. Like a binary search tree, the only thing that we need to care about is which element is bigger. Hence, for calculation, the value of each node is not important. For trie, however this is not the case. Trie is limited by the size of the alphabet, some methods like converting to codepoint may help Trie to be able to store UTF-8 characters and more. But for TST, every character can be stored, as long as it can be stored in char, which they all do.

Std::string is a powerful string container, at the early stage of our development process, wstring seems to be a better choice. While wstring is not as well supported as string, it can store Unicode characters while std::string can't. We are wrong, std::string can store Unicode characters too, and more, if the char we need to store needs more than 8 bit, std::string can handle it. As such, using string with TST creates a both general, stable, and efficient way to deal with string.

Hence, we choose you, TST.

And for why do we decide to store definition in file but not RAM, it is to reduce space usage. 2GB of raw data only cost 30MB of RAM to hold. It's suitable for most device, even the very low-end one.

6) Implementation Details:

a) Core part:

```
struct TSTNode {
    char key;
    int val;


    TSTNode *lo,*mid,*hi;


    TSTNode (char _key) {
        lo=mid=hi=nullptr;
        key=_key;
        //if (key==' ') key='_';
        val=0;
    }


    TSTNode () {
        lo=mid=hi=nullptr;
        key='a';
        val=0;
    }


    TSTNode* get(string cur,int idx);
    TSTNode* get(string cur);
```

```
    TSTNode* getAlways(string cur);
    TSTNode* getAlways(string cur,int idx);
    TSTNode* getAlwaysDFS();


    TSTNode* insert(string cur);
    TSTNode* insert(string cur,int idx);


    void loadFromFile(ifstream &in);
    void loadFromFile(string path);
    void saveToFile(ofstream &out);
    void saveToFile(string path);


    void clear();
};


struct TST {
    TSTNode* pRoot;


    TST() {
        pRoot=new TSTNode();
    }


    TSTNode* get(string cur,int idx);
    TSTNode* get(string cur);
    TSTNode* getAlways(string cur,int idx);
    TSTNode* getAlways(string cur);


    TSTNode* insert(string cur);
    TSTNode* insert(string cur,int idx);


    void loadFromFile(ifstream &in);
    void loadFromFile(string path);
    void saveToFile(ofstream &out);
    void saveToFile(string path);


    void clear();
};
```

Here we see the TST data structure, it support both an update and insert query. Each character, or sub-character is used as key, and when the string end, the index of the file is store at val.

There are three pointer point to three other TSTNode, hence the name ternary.

We denote S to be the sum of string length.

- To save the file, we save the data in its DFS inorder order. Time complexity: O(S)
- To load the file, we just load the DFS order we have run. Time complexity: O(S).
- Insertion: Avarage O(K*log(S)) K is the length of the string.
- Get: Avarage O(K*log(S)) K is the length of the string.
- Delete (Insert null): Avarage O(K*log(S)) K is the length of the string.

b) GUI part:

i. Login Window:

```
class FormLogin : public wxFrame
{
public:
    FormLogin(const wxString& title);

    // Destructor
    virtual ~FormLogin();

private:
    wxPanel* panel;
    wxTextCtrl* m_usernameEntry;
    wxTextCtrl* m_passwordEntry;
    wxButton* m_buttonLogin;
    wxButton* m_buttonSignUp;
    wxMessageDialog *dlg;
    wxImage m_image;
    wxBitmap m_scaledBg;
    wxStaticText* new_username;
    wxStaticText* new_password;
    wxTextCtrl* m_newUsernameEntry;
    wxTextCtrl* m_newPasswordEntry;
    wxDialog* signUpdlg;
    wxDialog* retrievedlg;
    wxString username;
    wxString password;
    wxString retrievedPass;
    wxString retrieveUser;
    wxButton* m_btnStart;
        wxString GetPasswordFromUsername(const wxString& username);

private:
    void OnLogin(wxCommandEvent& event);
    void OnImagePanelPaint(wxPaintEvent&);
    void CreateScaledBg();
    void OnForgetPassword(wxMouseEvent& event);
```

```
    void OnSignUp(wxCommandEvent& event);

  private:
    DECLARE_EVENT_TABLE()

    enum
        •  {
      BUTTON_CreateAcc = wxID_HIGHEST +3,
      BUTTON_RetrievePass = wxID_HIGHEST+4,
      BUTTON_SignUp = wxID_HIGHEST + 2,
      BUTTON_Login = wxID_HIGHEST + 1,
        };
};
#endif
```

This is the class for the GUI mode of login window. In this class, there are several functions with explanation of their usage below:

- OnLogin: When clicking the Login button, this function will check whether the username and password are correct. If indeed they are, the dictionary window will be opened and the login window will be closed.
- OnImagePanelPaint: Painting the background image of this window
- CreateScaledBg: Create a scaled background image
- OnForgetPassword: When clicking the "Forget your password? line, there will be a dialog for the users to retrieve their password from username.
- OnSignUp: If this person is a new user and has no account, they can click this button to sign up and have their own account to use this dictionary

ii. Dictionary Window:

```
class MyFrame : public wxFrame
{
public:
   MyFrame();
   virtual ~MyFrame();
   void OnBookCtrl(wxBookCtrlBaseEvent& event);
   void OnShowImages(wxCommandEvent& event);

   enum GameMode {
      GuessWord,
      GuessDefinition
   };

   void OnNotebook(wxNotebookEvent& event) { OnBookCtrl(event); }

   wxBookCtrlBase *GetCurrentBook() const { return m_bookCtrl; }
   void RecreateBook();
   void RecreateBook(int idx);
   wxPanel *CreateNewPage() const;
   void AddFlagStrIfFlagPresent(wxString & flagStr, long flags, long flag, const wxString& flagName) const;

   enum BookType
```

```
    {
        Type_Notebook,
        Type_Max
    } m_type;


    wxSound m_tabChangeSound;
    wxPanel *m_panel; // Panel containing notebook and other controls
    wxBookCtrlBase *m_bookCtrl;
    wxStaticText *m_text;
    bool m_chkShowImages;
    wxBoxSizer *m_sizerFrame;
    wxBookCtrlBase::Images m_images;
    wxBitmapButton* m_reset;
    wxBitmapButton* m_logout;

    wxDECLARE_EVENT_TABLE();
};

enum ID_COMMANDS
{
    ID_BOOK_NOTEBOOK,
    ID_BOOK_MAX,
};

//insert page
#define DICTIONARY        "Dictionary"

//add new word
#define ADD_NEW_WORD          "Add new word"

//favourite list
#define FAVOURITE_LIST                "Favourite list"

//game
#define GAME                "Game"
#endif /* treebook_test_hpp */
```

This is the class for dictionary window, where there are 4 pages in form of an wxNotebook (belong to wxWidgets) with different functions:

- OnBookCtrl: handle event related to book control
- OnShowImages: handle events related to displaying images
- OnNotebook: an event handler that invokes OnBookCtrl for wxAuiNotebook events.
- GetCurrentBook: return the current book control
- CreateNewPage: create new page for book control

```
wxPanel *DictionaryPage(wxBookCtrlBase *parent):
```

This is the function to create Dictionary page, where the common functions of a dictionary are operated (normal lookup, reverse lookup, edit meaning, add to favorite list)

```
wxPanel *CreateAddPage(wxBookCtrlBase *parent):
```

This is the function to create a page where users can view their lookup history.

wxPanel *FavoriteList(wxBookCtrlBase *parent):
This is the function to create a page where users can view their favorite vocabulary list and adjust the list to their preference.

wxWindow* CreateGamePage(wxBookCtrlBase* parent):
This is the function to create a page where users can learn new words. There are two game modes: guess word and guess meaning. There will be a random word/definition shown on the screen and users can choose one in four answers generated by our system.

7) Technical Problems And Solutions:

- Heavy RAM Usage of Trie and definition loading

During the early stage of development, we have always worried about the heavy RAM usage. As loading the whole data set into the RAM causes the program to take more space. In addition, it does cause the loading process to be noticeable.

➔ Solution: Use modified Ternary Search Tree for data mapping and store the infomation in seperate files. As Ternary Search Tree is a data structure similar to both a trie and a binary search tree, it inherited the space-efficiency of binary search tree and also the time-effectiveness of trie on string-searching problem. While it is slower than a trie, the different isn't noticeable, at least for our data set.

- Multiple platforms and IDEs:

Our group consists of four members and each person uses a different IDE namely Visual Studio Code, Visual Studio, XCode and CodeBlock. There are also two platforms which are Window and MacOS. This variance has led to many issues, including different available libraries and GUI features. wxWidgets also changes the final interface based on the platforms, which makes it hard to navigate the final layout in a group project.

➔ Solution: Compromise on differences and choose the most common platform in our class which is Window to focus on because of time and manpower constraints. However, we always strive for the best possible result on both platforms.

- wxWidgets

This is cross-platform GUI library. However, the installation is not easy. It took us almost 3 days to successfully compile its first simplest sample. The manual script is also unclear, and we had to learn everything from scratch, which took more time and effort especially in a time-limited project.

➔ Solution: We used online forums like Stack Overflow or wxWidgets Discussion Forums to learn things we do not understand and fix bugs. Also, we took advantage of ChatGPT to understand complicated concept and to have suggestions for personal challenging bugs.

- GitHub

Specifically, using GitHub has been a huge challenge for our group. It is not because we do not know how to use it, but the point lies in GitHub Desktop. It frequently said it could not find the local repository at the given directory despite no changes having been made. It also took a considerable amount of time to clone again or

pull due to huge dataset. Sometimes it duplicated files without no specific reasons or kept files that we had deleted.

➔ Solution: Being patience is the key. Although furiousness and irritation were inevitable, we tried to calm each other down and redid everything after reading suggestions and advice from experts.

- Definition to word in general

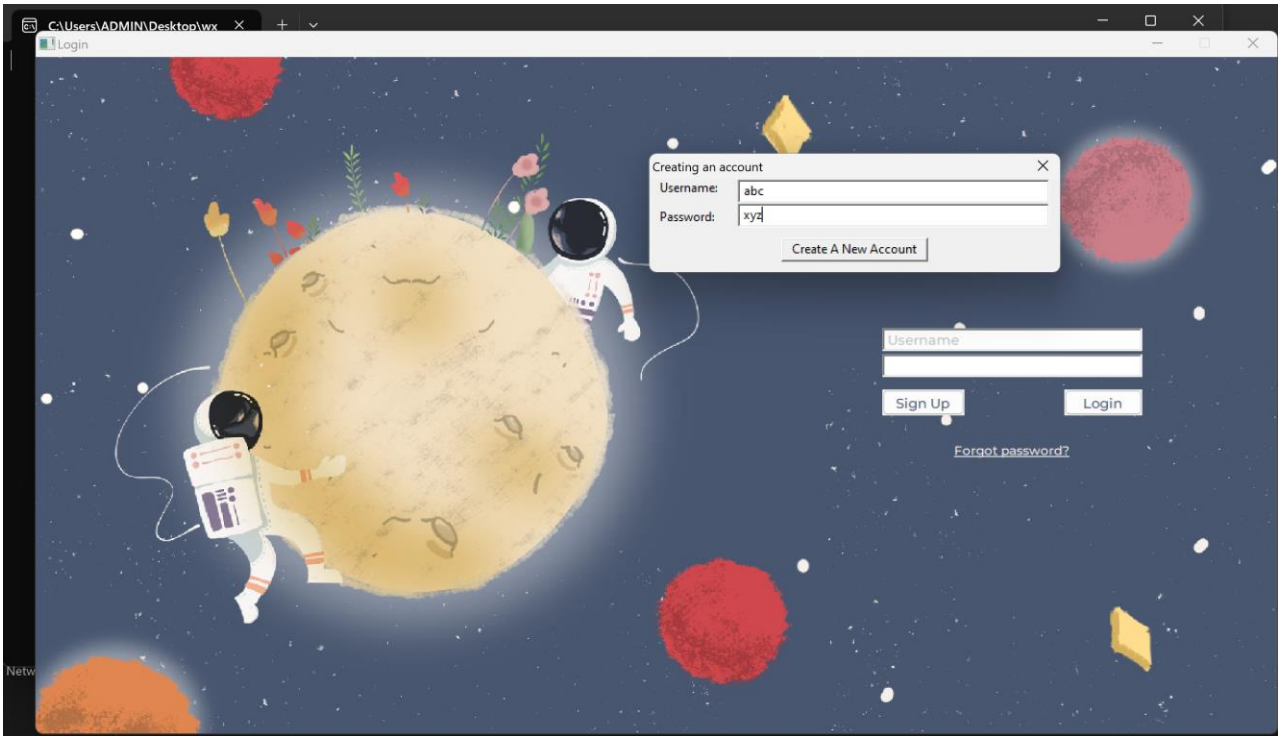
We have tried many ways for this, from Z-function, cosine similarity but this is still the slowest part of our program, the lowest complexity is O(nk) for k to be the sum of words of the string and n be the number of words we search.

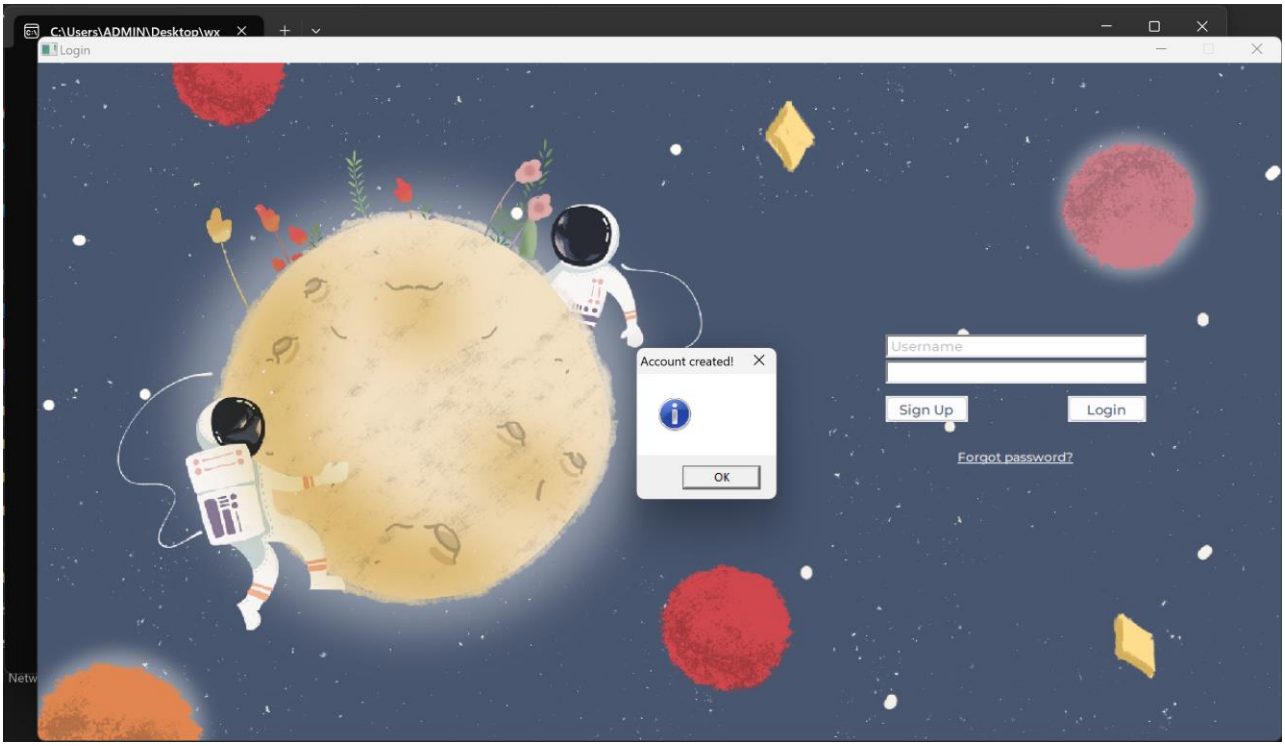
➔ Solution: We ditch every complicated solution we have and return to the simplest way, brute force through all files. But we also apply multi-threading to our program, now the search time is divided by the number of threads we can utilize.

This work beautifully as this does not drain resources and does not need a good single core. Low-end users can use it (a bit slow of course) and for people who want to scale up, they don't need to get a good CPU, just connect some old CPU together and we're done.

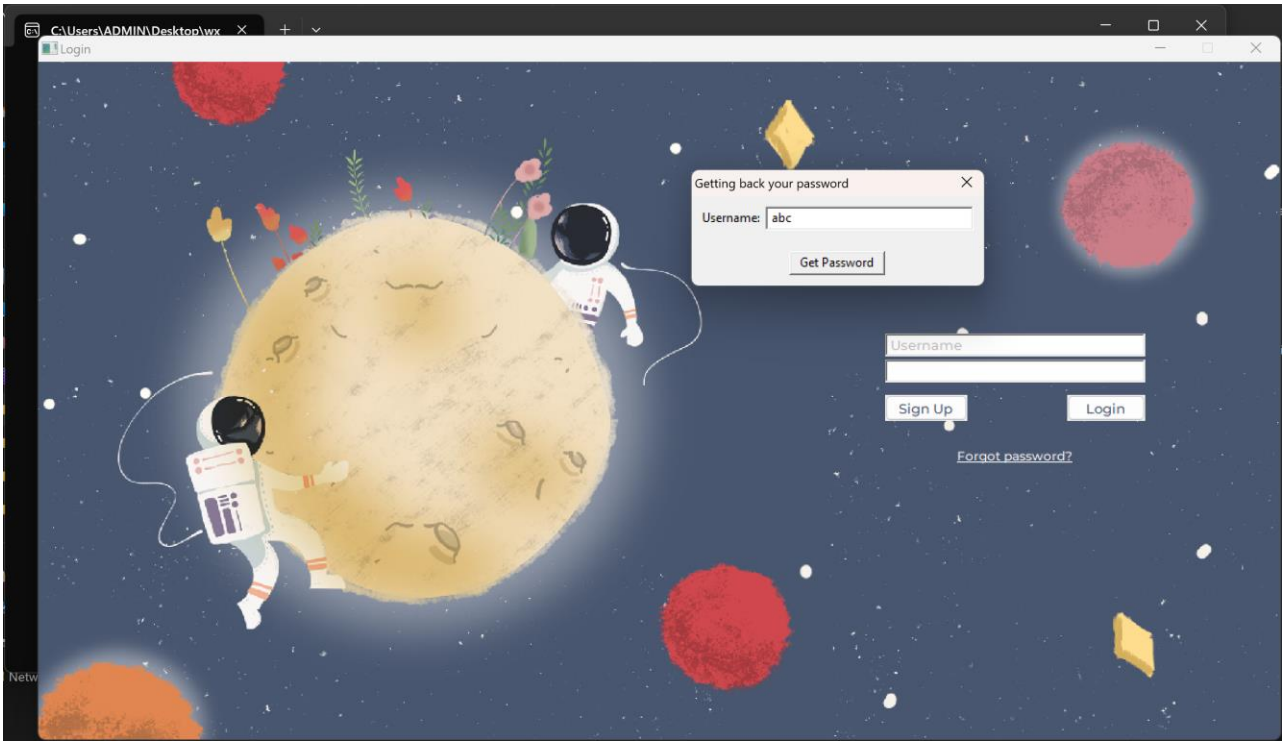
8) Feature Demonstrations:

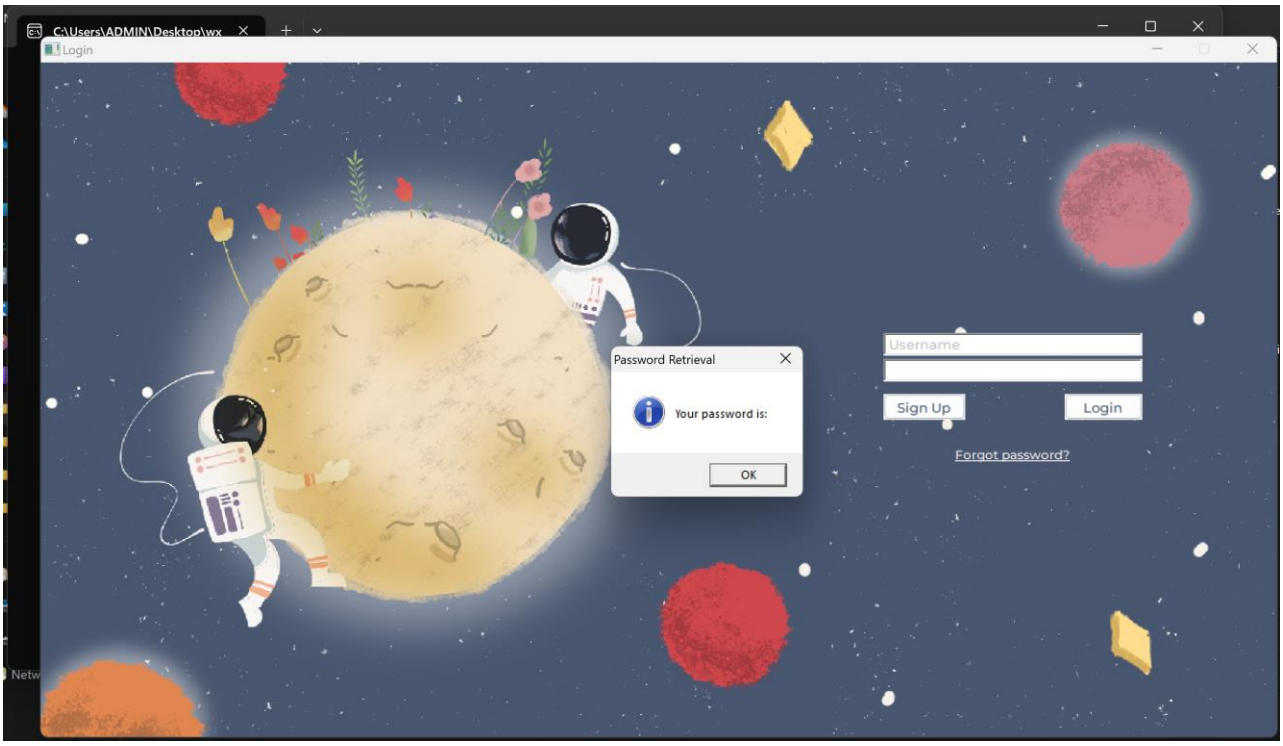
a) Login Window:



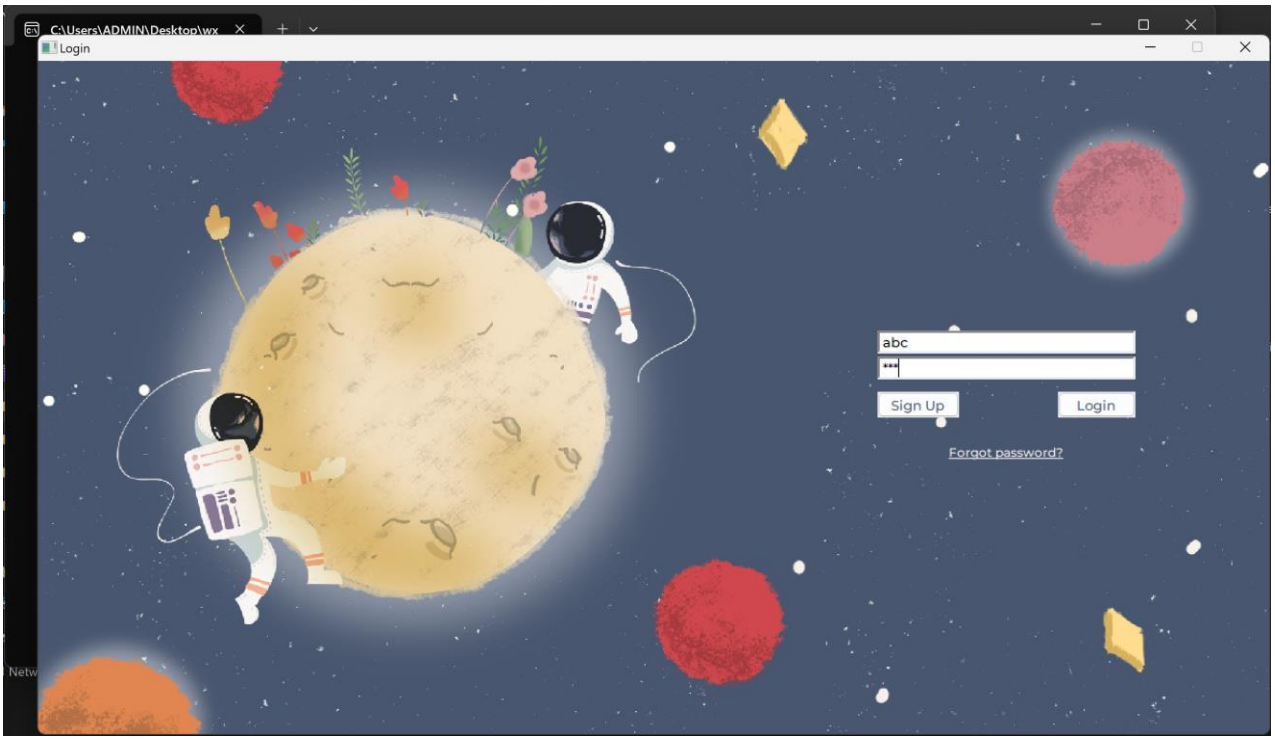


To sign up, click the “Sign Up” button and there will be a dialog appearing as in the picture. Simply type in your username and password of your preference and click “Create A New Account”. If this username and password are unique, there will be a message informing you of successful signing up. Click “OK” to continue with your login process.





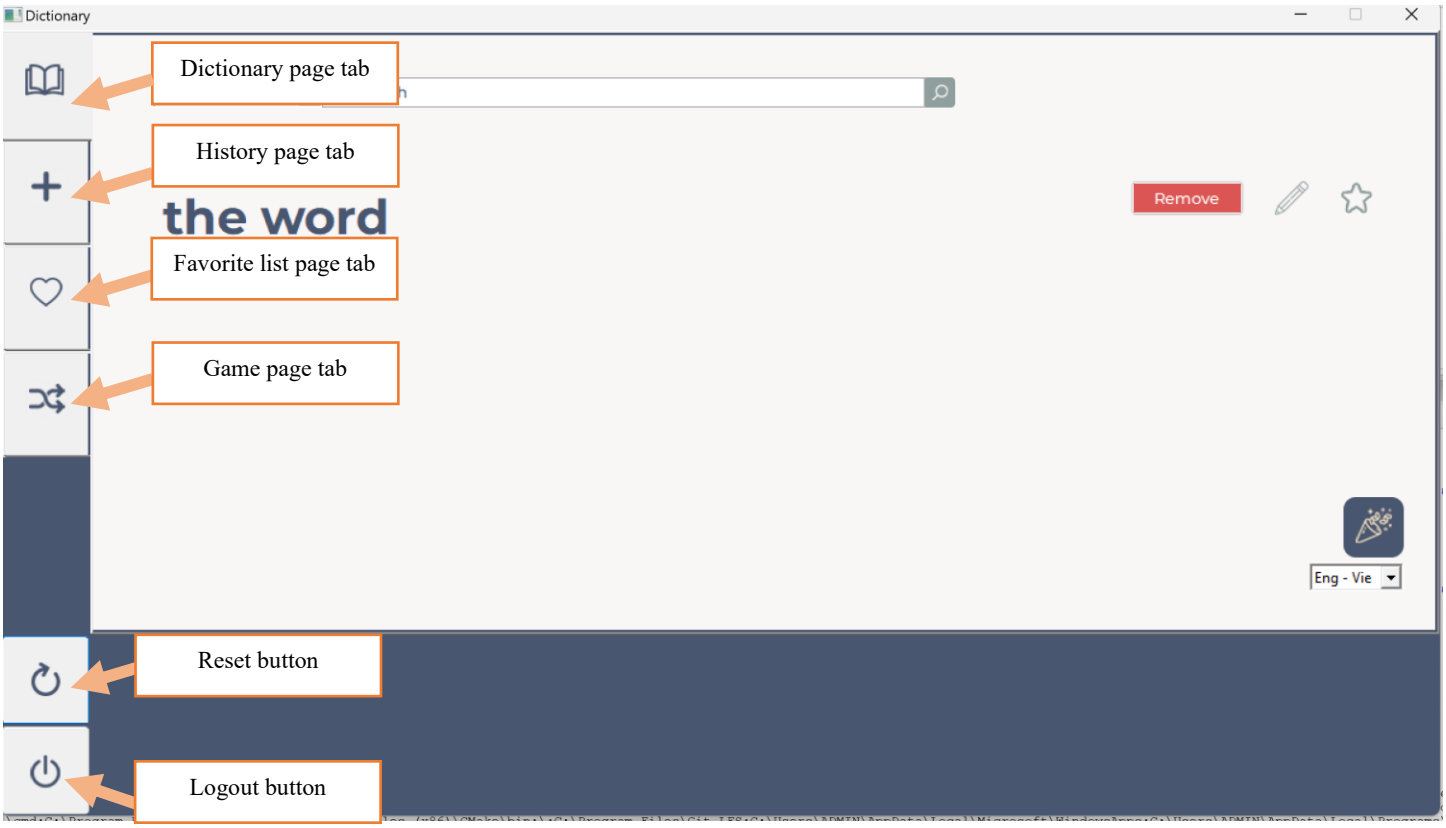
If you forget your password, click the “Forgot password?” line and there will be a dialog showing up. You just need to type in your username and click “Get Password”. If your username exists, the password will be shown to you through a message box. Click “OK” to close the box and continue with your login process.



To login, type your username and password in the two long white rectangular boxes as in the picture above. Then, just click “Login”. If your username and password are valid, we will show you the dictionary window and you can use it as instructed in the b, c, d, e parts below.

b) Main layout of Dictionary:

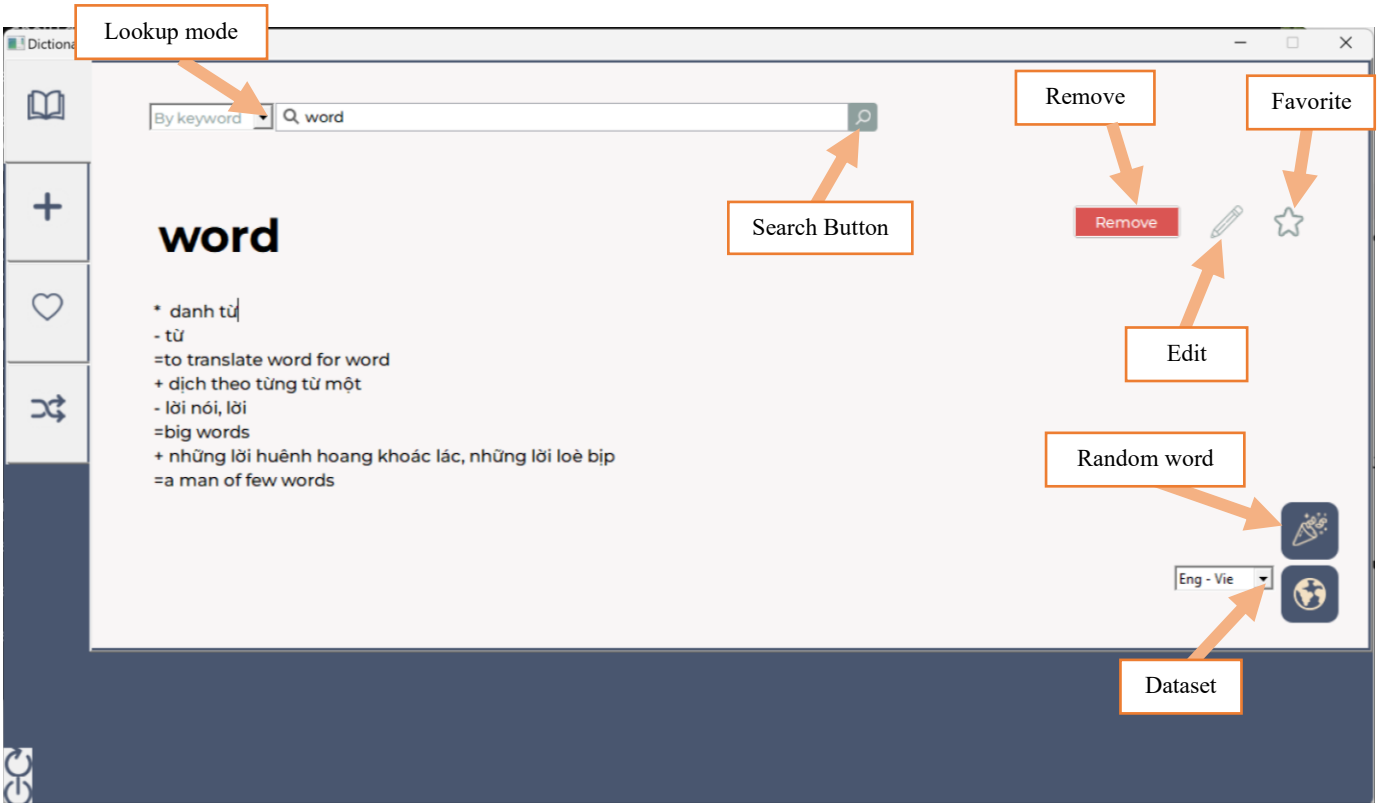
When you successfully log in to our application, this will be the page that firstly appears on your computer:



On the right hand side, you can see 4 tabs. Each tab represents a page with a specific purpose. The tab with the book icon is for dictionary page, which is also the one showing on your screen right now. The second one with the plus icon is for history page, a page where you can view your lookup history in each dataset. The third one with a heart icon is for favorite list page, where you can work with your vocabulary favorite list. And finally, the one with two intertwined arrows is for game page, which we will discuss later in part f of this features demonstration. When you want to change to another tab, simply click the corresponding tab and you will be moved to where you want.

At the bottom left corner, there are 2 buttons. The upper one is for resetting your dataset to the original set. This means any changes you have made so far will be discarded. The lower one is for logging out. It will bring you back to the login window. Please note that these 2 buttons can always be accessed from any pages.

c) Dictionary Page:



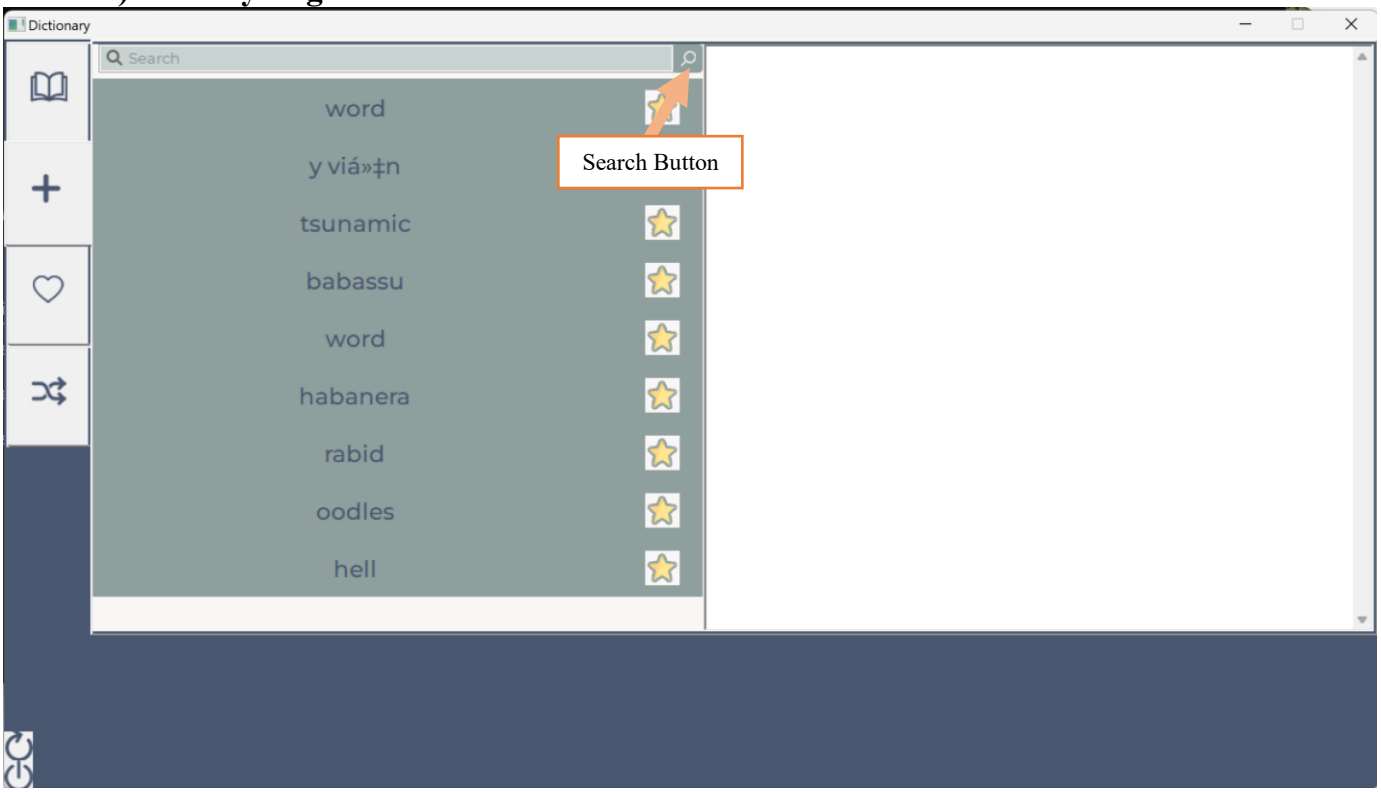
You first start with choosing the dataset by clicking on the arrow in the box on the bottom right corner and choose one option from the list. Then, you choose a lookup mode by clicking on the arrow in the box on the top left corner and choose “By word” or “By definition”. “By word” is the normal lookup: word to definition and “By definition” is the reverse lookup: from definition to word.

After that, you can start looking up words. You type the keyword into the search bar and click the magnifying glass button to search.

On the top right corner of the meaning box, you can see a combination of 3 buttons. The first one is to remove the currently shown word from the dataset. The second one is to edit the meaning of the currently shown word. If you click that button, the word and meaning box will become editable. If you only change the meaning and click “Save”, it will only change the meaning of the current word. If you change the word too, our application will automatically create a new word with the corresponding meaning for you. The third button is a star button. It will show you the favorite state of that word. If the star is yellow, the word has been added to the favorite list. If it is white, it has not been added to the list. You can remove or add it to the list by clicking that button.

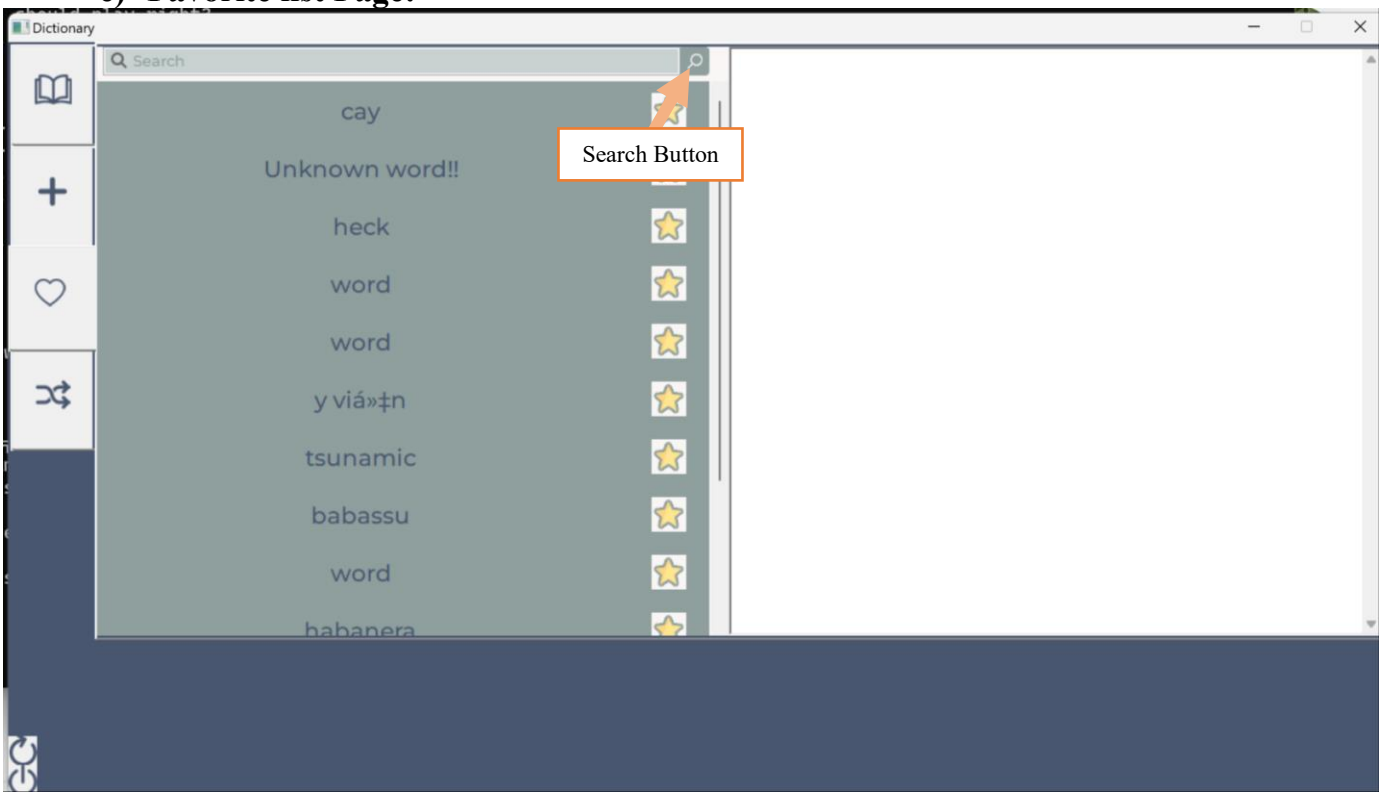
You can also view a random word in our dataset by clicking the confetti cannon on the bottom right corner.

d) History Page:



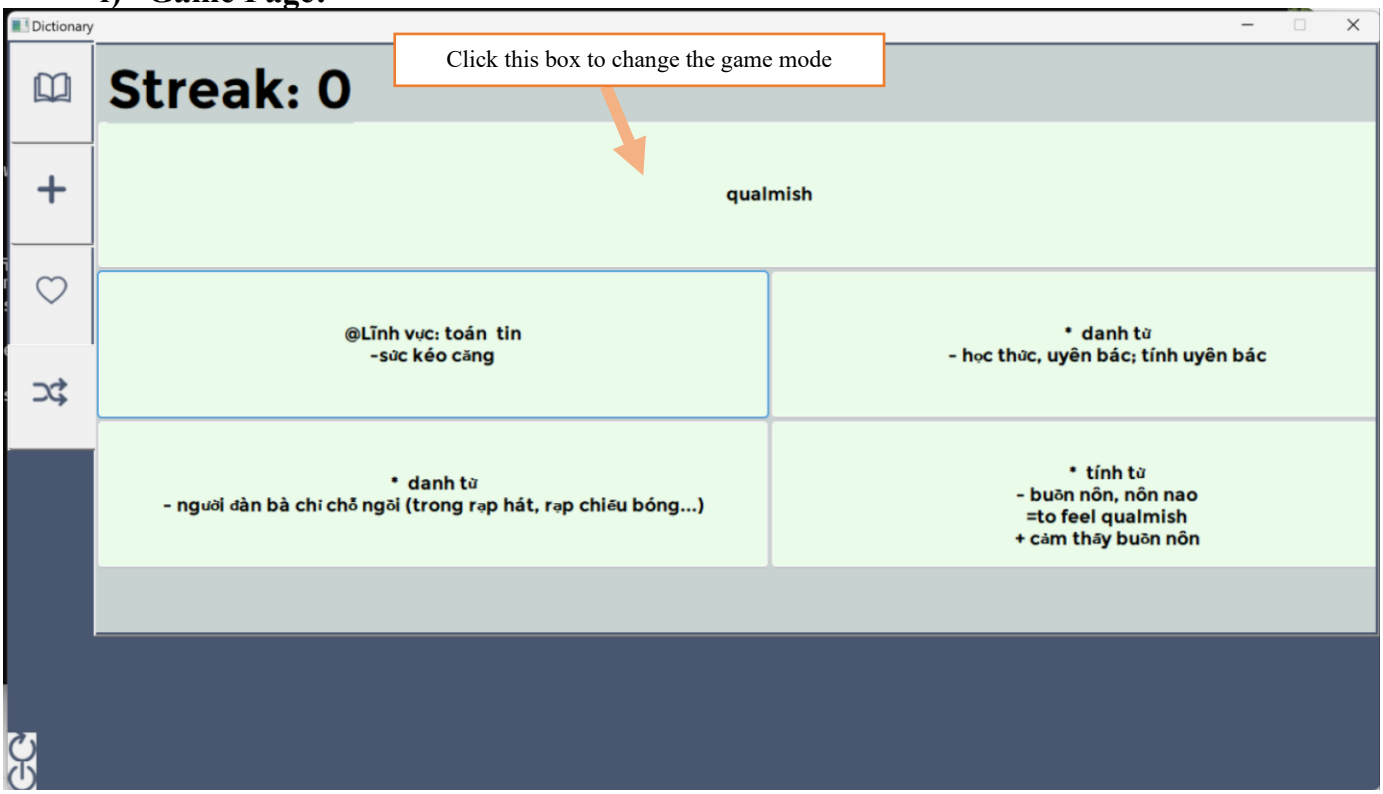
Here, you can view all the words that you have looked up in the dictionary page. You can check if a word is there or not with the search function at the top.

e) Favorite list Page:



In this page, you can search for a word in your list by simply tying that word in the search bar and click the magnifying glass button. If that word has been added, it will show up below the search bar. You can remove that word from the list by clicking the yellow star on the right.

f) Game Page:



On the top left corner, you can see your streak. It will count the number of time you continuously choose the correct answer. If you choose the wrong one, it will be reset to 0. Below the streak is the question. In the picture above, we are in guess-definition mode. It means that you need to guess the meaning of the given word. Choose any of the four boxes under the word.

You can change the mode to guess-word by clicking the question box. It will give you a definition and you can choose one out of 4 words given in the answer boxes.

9) Conclusion:

This project has provided users with not only common but new features to personalize their learning experience. Also, a colorful and carefully designed interface is presented to help evoke the passion and interest. However, there is still room for development such as a more time-efficient dataset loading and more interactive widgets.

10) References:

- Somenath, M. (2006). English Dictionary. *CodeProject*. https://www.codeproject.com/Articles/13730/English-Dictionary?fbclid=IwAR0HrcuqhufGjvS5mgxT6QYBjYi3kMelg0kRWzZFLpfVOR91e9FL77B08zo_aem_AVObwtxEZ7zjJtOSikKkCQFTcCVplcY9EB5jY8QWB-4yFkjdQo4r73-Wi9EHWI_pRvE
- *WXWidgets: Documentation*. (n.d.). https://docs.wxwidgets.org/3.2/
- Imron. (2014, December 1). *Wxwidgets form login part 2. (Parent window is disabled after child window login success)*. Imron02. https://imron02.wordpress.com/2014/12/01/wxwidgets-form-login-part-2-parent-window-is-disabled-after-child-window-login-success/