Mehran University of Engineering & Technology, Jamshoro

PROJECT REPORT

DIGITAL NOTEBOOK

Department	Computer Systems Engineering
Class	24CS Section 2
Submitted By	Mokash Kumar
	Naveed
	Muhammad Tarique
	Muhammad Azaan
Submitted To	Sir Fawad Mangi
Subject	Computer Programming
Course Code	CS-151

Contents

1. Introduction:

In this modern and digitally-driven age, the skill to manage and keep data in an organized manner becomes an absolute necessity. Traditionally, note-taking with paper often seems quite inflexible, unavailable, and incapable of handling the current-day pace and heavy flow of information. The solution is a digitized replacement of traditional functions, which will prove of extreme usefulness while such must-have features like searching, encrypting, and saving data on-the-go are made accessible to devices.

The **Digital Notebook** is an application designed for efficient and secure note management. This system allows users to create, view, edit, and delete notes with features like password protection, sticky notes, encryption, recycle bin. Its user-friendly design and functionality make it suitable for academic, professional, and personal use whether it is study notes, grocery items list, tasks, or personal notes.

2. Problem Statement

Traditional methods of note-taking face challenges like:

- Difficulty in managing large amounts of data.
- Lack of data privacy and security.
- Inability to search, encrypt, or store notes efficiently.

The Digital Notebook Management System addresses these issues with a secure, efficient, and feature-rich solution.

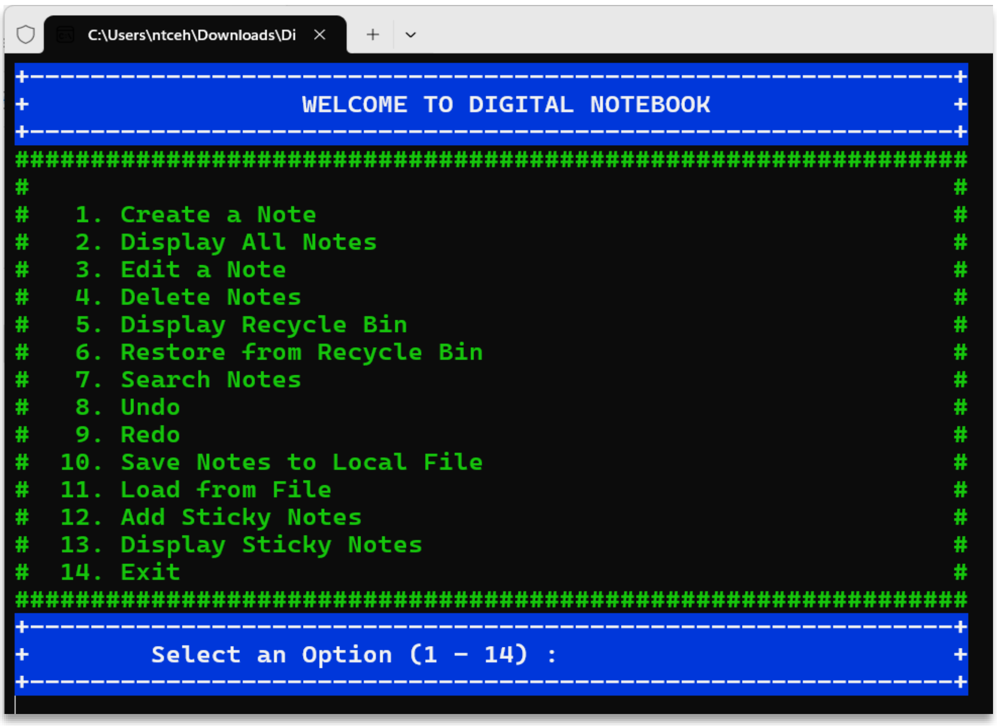
3. Research Objectives:

1. **Development of a Feature-Rich Note-Taking Application**
    - To design and implement a digital notebook that allows users to create, edit, and organize their notes with ease.
2. **Integration of Advanced Functionalities**
    - To provide functionalities such as encryption, priority tagging, undo/redo capabilities, and sticky notes to enhance user experience.
3. **Data Security and Privacy**
    - To ensure data confidentiality through encryption mechanisms and optional password protection for sensitive notes.
4. **Enhanced Accessibility and Searchability**
    - To implement a robust search function that enables users to quickly retrieve notes based on categories, tags, or keywords.
5. **Seamless User Experience**
    - To develop an intuitive and interactive interface for easy navigation and efficient note management.
6. **Sustainability and Portability**

- To ensure the application supports file-based storage, allowing users to save and retrieve notes across sessions and devices.

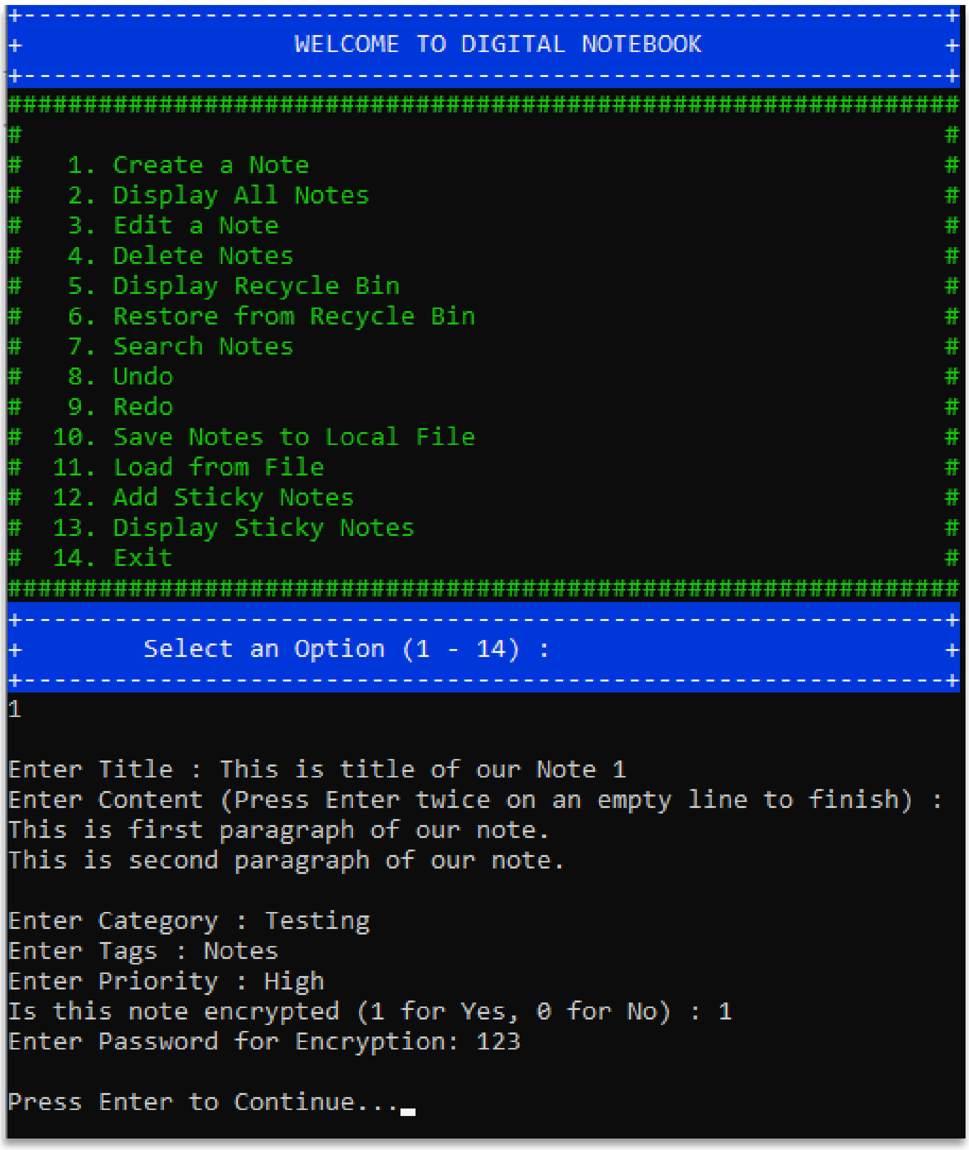
4. System Overview

The system offers the following features:



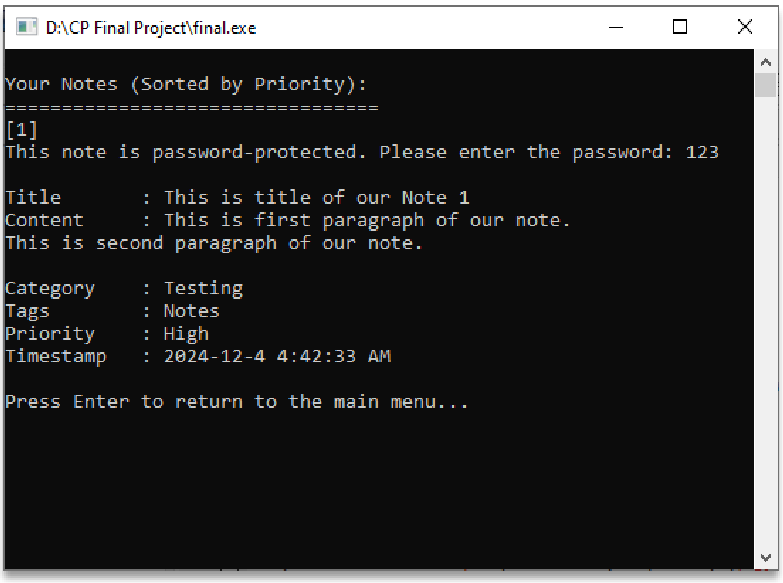
```
+---------------------------------------------------------------+
+                  WELCOME TO DIGITAL NOTEBOOK                  +
+---------------------------------------------------------------+
################################################################
#                                                              #
#   1.  Create a Note                                          #
#   2.  Display All Notes                                      #
#   3.  Edit a Note                                            #
#   4.  Delete Notes                                           #
#   5.  Display Recycle Bin                                    #
#   6.  Restore from Recycle Bin                               #
#   7.  Search Notes                                           #
#   8.  Undo                                                   #
#   9.  Redo                                                   #
#  10.  Save Notes to Local File                               #
#  11.  Load from File                                         #
#  12.  Add Sticky Notes                                       #
#  13.  Display Sticky Notes                                   #
#  14.  Exit                                                   #
################################################################
+---------------------------------------------------------------+
+         Select an Option (1 - 14) :                           +
+---------------------------------------------------------------+
```

- **Create Notes**: Add detailed notes with title, content, category, tags, and priority.
- **Encryption**: Ensure privacy by encrypting sensitive notes.



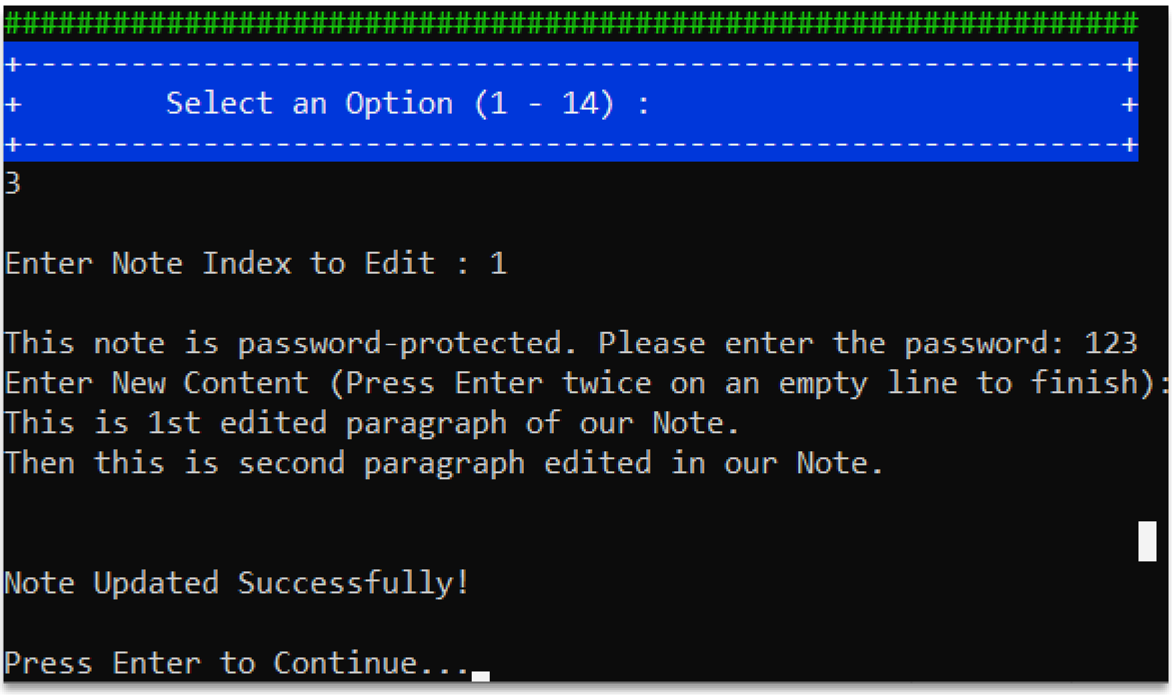
```
+---------------------------------------------------------------+
+                  WELCOME TO DIGITAL NOTEBOOK                  +
+---------------------------------------------------------------+
################################################################
#                                                              #
#   1. Create a Note                                           #
#   2. Display All Notes                                       #
#   3. Edit a Note                                             #
#   4. Delete Notes                                            #
#   5. Display Recycle Bin                                     #
#   6. Restore from Recycle Bin                                #
#   7. Search Notes                                            #
#   8. Undo                                                    #
#   9. Redo                                                    #
#  10. Save Notes to Local File                                #
#  11. Load from File                                          #
#  12. Add Sticky Notes                                        #
#  13. Display Sticky Notes                                    #
#  14. Exit                                                    #
################################################################
+---------------------------------------------------------------+
+        Select an Option (1 - 14) :                            +
+---------------------------------------------------------------+
1

Enter Title : This is title of our Note 1
Enter Content (Press Enter twice on an empty line to finish) :
This is first paragraph of our note.
This is second paragraph of our note.

Enter Category : Testing
Enter Tags : Notes
Enter Priority : High
Is this note encrypted (1 for Yes, 0 for No) : 1
Enter Password for Encryption: 123

Press Enter to Continue...
```

- **Display Notes:** This enables us to display our notes and if they are encrypted with password then it first asks for it.



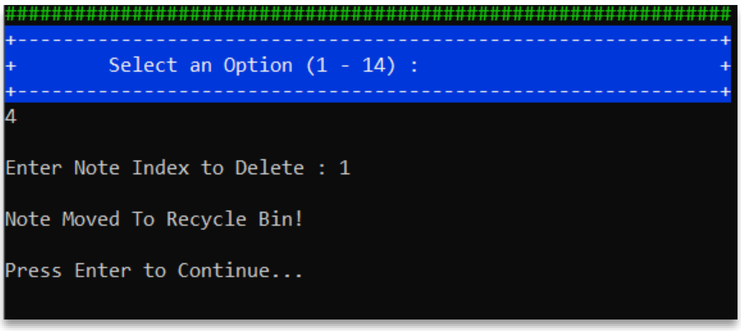
```
D:\CP Final Project\final.exe

Your Notes (Sorted by Priority):
=================================
[1]
This note is password-protected. Please enter the password: 123

Title        : This is title of our Note 1
Content      : This is first paragraph of our note.
This is second paragraph of our note.

Category     : Testing
Tags         : Notes
Priority     : High
Timestamp    : 2024-12-4 4:42:33 AM

Press Enter to return to the main menu...
```

- **Edit Note:** We can edit content of our note.



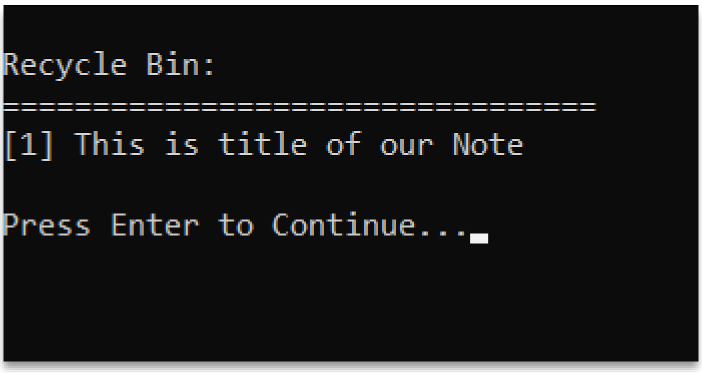
```
##################################################################
+----------------------------------------------------------------+
+        Select an Option (1 - 14) :                             +
+----------------------------------------------------------------+
3

Enter Note Index to Edit : 1

This note is password-protected. Please enter the password: 123
Enter New Content (Press Enter twice on an empty line to finish):
This is 1st edited paragraph of our Note.
Then this is second paragraph edited in our Note.


Note Updated Successfully!

Press Enter to Continue...
```

- **Delete Notes**: We can delete any note by Index Number from File.

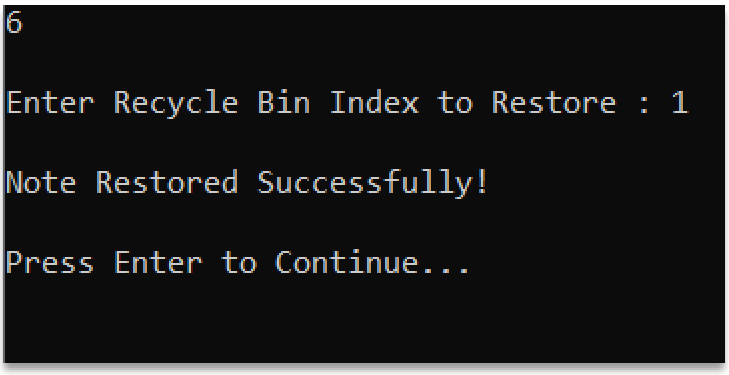


```
##################################################################
+----------------------------------------------------------------+
+        Select an Option (1 - 14) :                             +
+----------------------------------------------------------------+
4

Enter Note Index to Delete : 1

Note Moved To Recycle Bin!

Press Enter to Continue...
```

- **Recycle Bin**: Safeguard deleted notes for future restoration and helps us to recover deleted files.

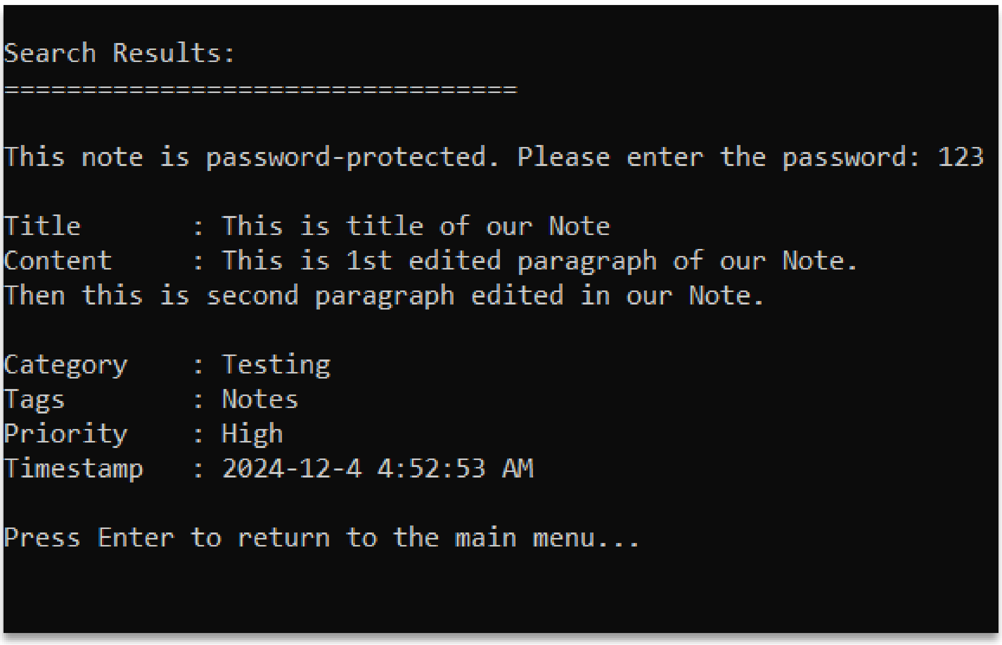


```
Recycle Bin:
==================================
[1] This is title of our Note

Press Enter to Continue...
```



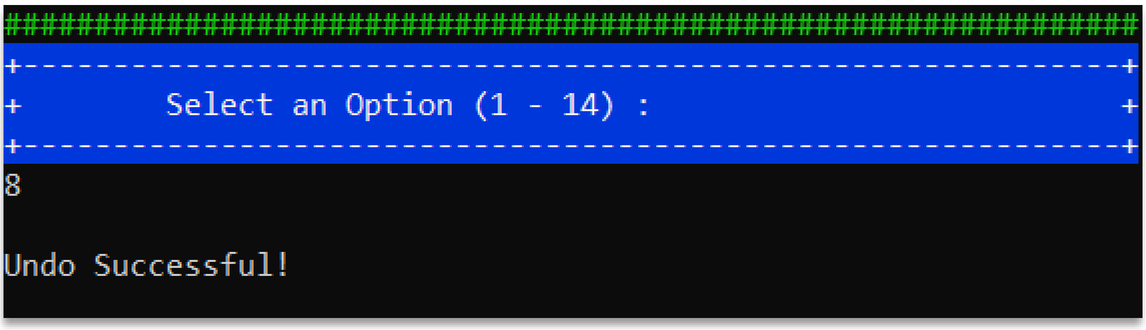
```
6

Enter Recycle Bin Index to Restore : 1

Note Restored Successfully!

Press Enter to Continue...
```

- **Search Notes:** This feature helps you to find note by matching keywords in note content.

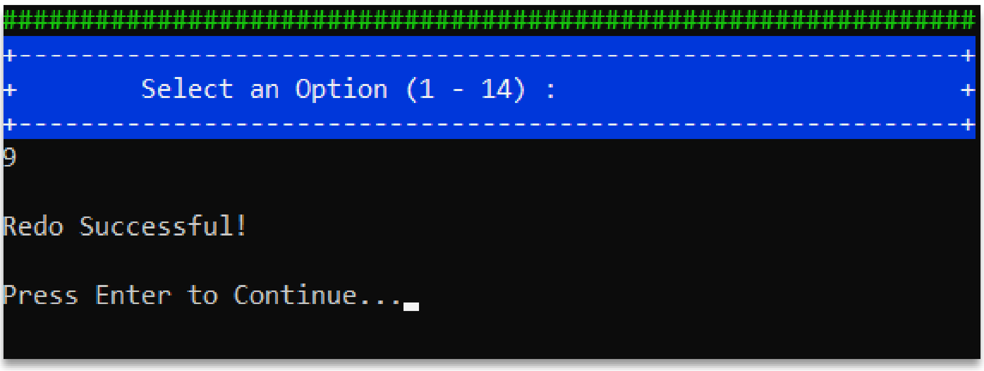


```
Search Results:
=================================

This note is password-protected. Please enter the password: 123

Title        : This is title of our Note
Content      : This is 1st edited paragraph of our Note.
Then this is second paragraph edited in our Note.

Category     : Testing
Tags         : Notes
Priority     : High
Timestamp    : 2024-12-4 4:52:53 AM

Press Enter to return to the main menu...
```

- **Undo/Redo**: Navigate through changes for reliability.

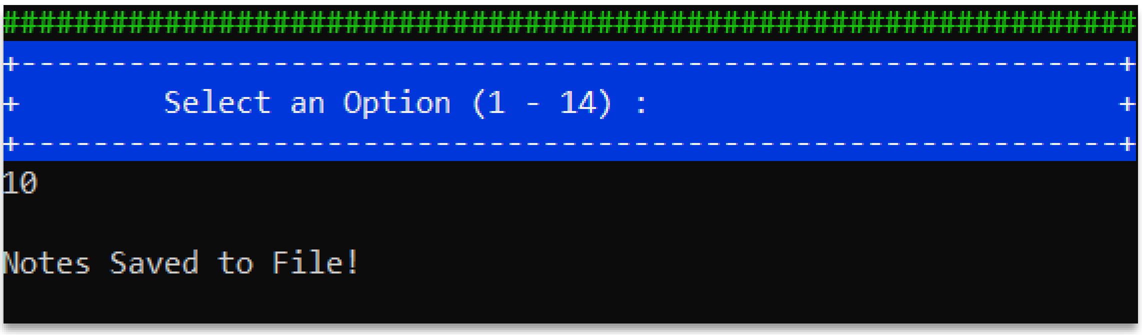


```
#########################################################################
+-----------------------------------------------------------------------+
+          Select an Option (1 - 14) :                                  +
+-----------------------------------------------------------------------+
8

Undo Successful!
```

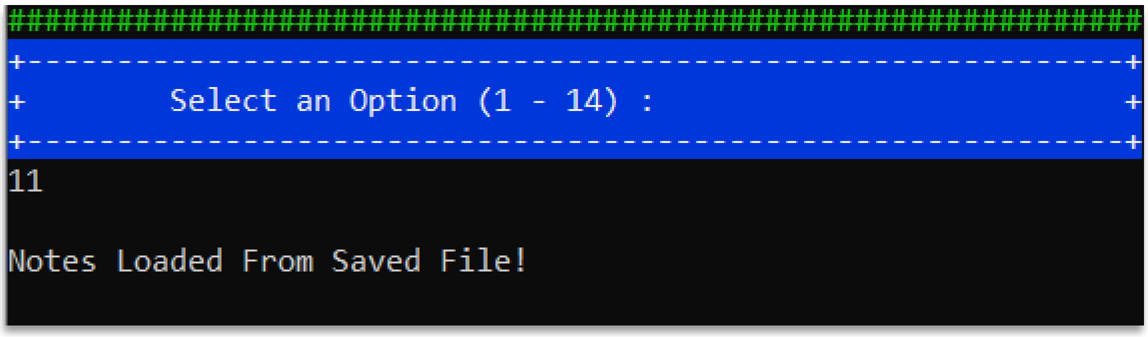


```
#################################################################
+---------------------------------------------------------------+
+          Select an Option (1 - 14) :                          +
+---------------------------------------------------------------+
9

Redo Successful!

Press Enter to Continue...
```

- **Save file and load File:** It helps us to store our notes in local storage and when we need those notes we can easily load them from saved file. It helps users to save data for long term and minimizes risk of loss of data.

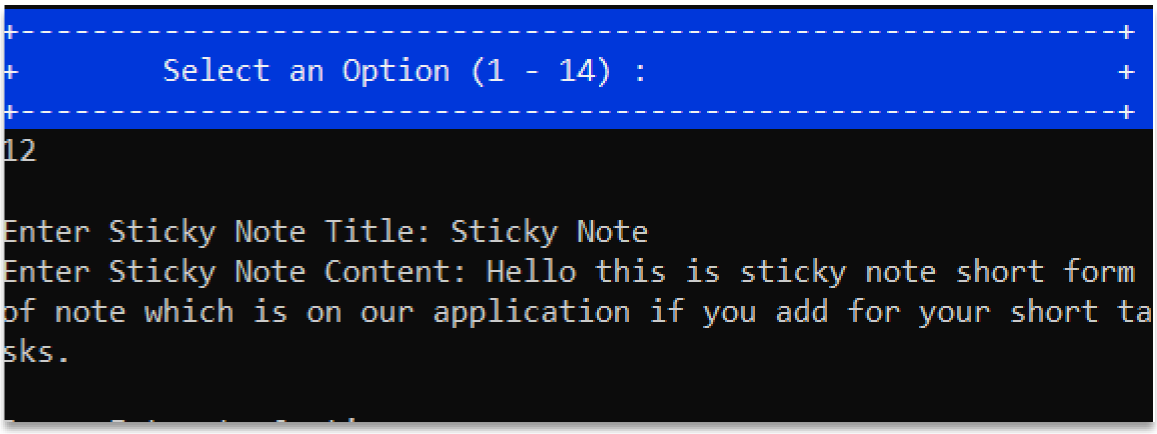


```
###############################################################
+-------------------------------------------------------------+
+        Select an Option (1 - 14) :                          +
+-------------------------------------------------------------+
10

Notes Saved to File!
```

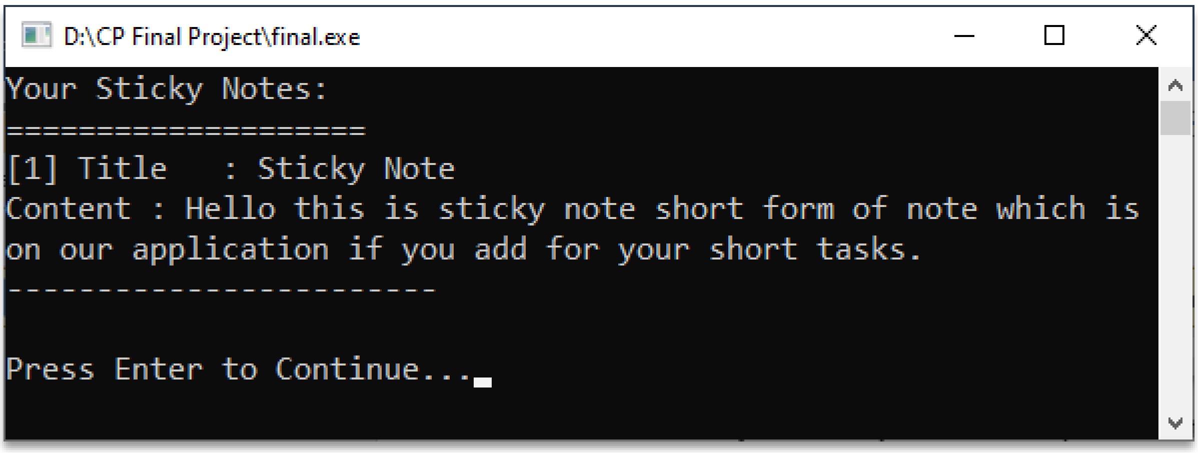


```
###############################################################
+-------------------------------------------------------------+
+        Select an Option (1 - 14) :                          +
+-------------------------------------------------------------+
11

Notes Loaded From Saved File!
```

- **Sticky Notes**: Quick and simple note-taking for reminders.



```
+-------------------------------------------------------------+
+        Select an Option (1 - 14) :                          +
+-------------------------------------------------------------+
12

Enter Sticky Note Title: Sticky Note
Enter Sticky Note Content: Hello this is sticky note short form
of note which is on our application if you add for your short ta
sks.
```



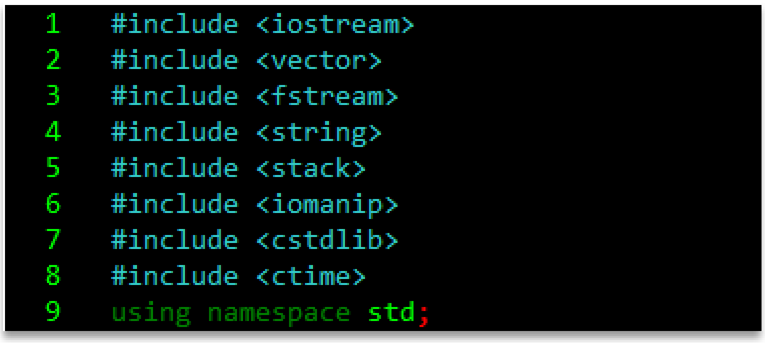
D:\CP Final Project\final.exe

```
Your Sticky Notes:
====================
[1] Title   : Sticky Note
Content : Hello this is sticky note short form of note which is
on our application if you add for your short tasks.
------------------------


Press Enter to Continue...
```

5. Code Analysis

1. Libraries Used

Here are the libraries or header files we have used in our project:

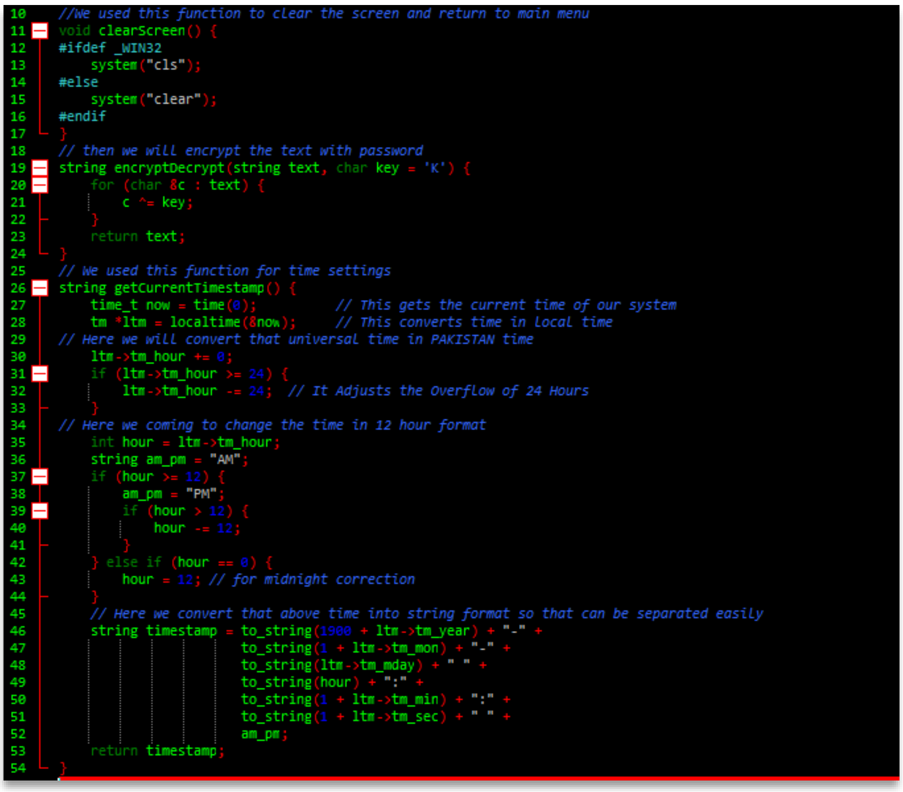


```cpp
#include <iostream>
#include <vector>
#include <fstream>
#include <string>
#include <stack>
#include <iomanip>
#include <cstdlib>
#include <ctime>
using namespace std;
```

- **`#include <iostream>`:** This is used for standard input and output.
- **`#include <vector>`:** It helps us to manage dynamic arrays (e.g., notes, sticky notes).
- **`#include <fstream>`:** Fstream used for for file handling (saving/loading notes).
- **`#include <string>`:** Used to handle strings efficiently.
- **`#include <stack>`:** To implement undo/redo operations.
- **`#include <iomanip>`:** It is used for formatted input/output.
- **`#include <cstdlib>`:** Actually this is used for system commands like (**`cls/clear`**).
- **`#include <ctime>`:** Now it is used For handling timestamps like while we save a note it notes timestamps.

2. Key Functionalities

a. Utility Functions



```cpp
//We used this function to clear the screen and return to main menu
void clearScreen() {
#ifdef _WIN32
    system("cls");
#else
    system("clear");
#endif
}
// then we will encrypt the text with password
string encryptDecrypt(string text, char key = 'K') {
    for (char &c : text) {
        c ^= key;
    }
    return text;
}
// We used this function for time settings
string getCurrentTimestamp() {
    time_t now = time(0);              // This gets the current time of our system
    tm *ltm = localtime(&now);         // This converts time in local time
// Here we will convert that universal time in PAKISTAN time
    ltm->tm_hour += 0;
    if (ltm->tm_hour >= 24) {
        ltm->tm_hour -= 24;  // It Adjusts the Overflow of 24 Hours
    }
// Here we coming to change the time in 12 hour format
    int hour = ltm->tm_hour;
    string am_pm = "AM";
    if (hour >= 12) {
        am_pm = "PM";
        if (hour > 12) {
            hour -= 12;
        }
    } else if (hour == 0) {
        hour = 12; // for midnight correction
    }
    // Here we convert that above time into string format so that can be separated easily
    string timestamp = to_string(1900 + ltm->tm_year) + "-" +
                       to_string(1 + ltm->tm_mon) + "-" +
                       to_string(ltm->tm_mday) + " " +
                       to_string(hour) + ":" +
                       to_string(1 + ltm->tm_min) + ":" +
                       to_string(1 + ltm->tm_sec) + " " +
                       am_pm;
    return timestamp;
}
```

- **`clearScreen`:**

  - It clears the terminal screen based on the operating system.

- **`encryptDecrypt`:**

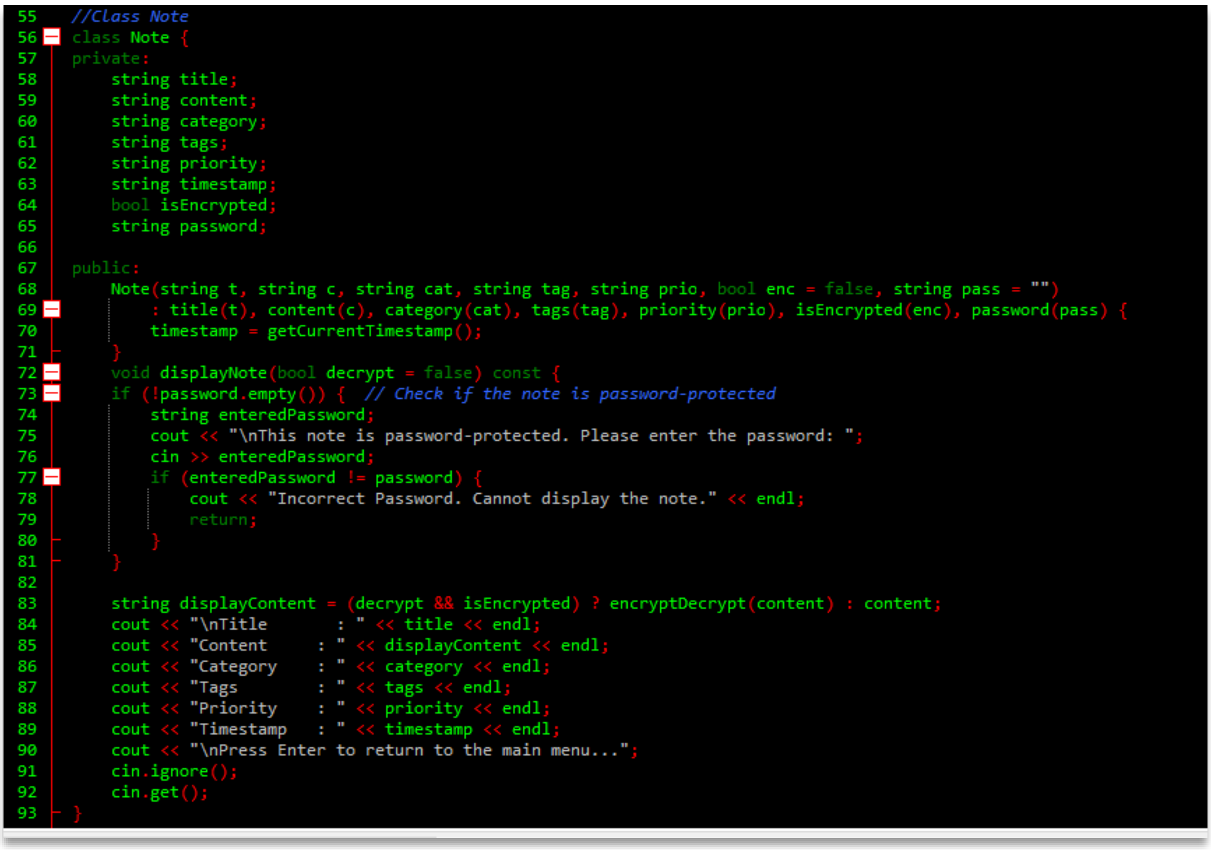
- It implements XOR-based encryption/decryption for notes while we access it either viewing or saved notes or editing our notes.
- Here we have used default encryption key is `'K'`.

- **`getCurrentTimestamp`:**

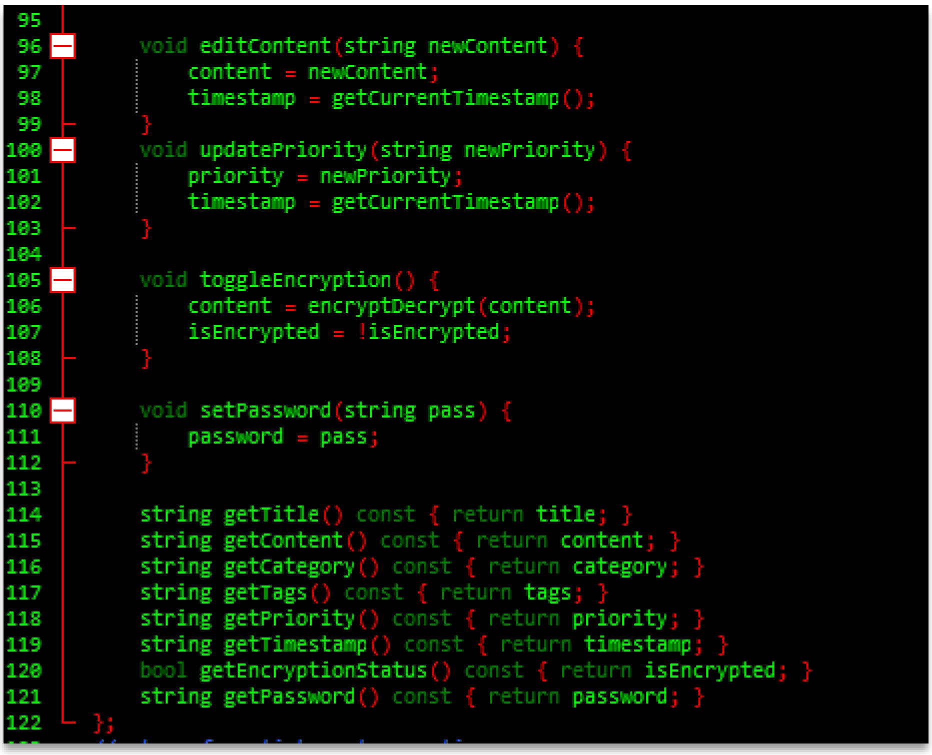
  - Fetches and formats the current system time, adjusted for Pakistan's timezone and converts to a 12-hour format with AM/PM.

*b. Class: **`Note`***

It is responsible for functionality of individual notes:



```cpp
//Class Note
class Note {
private:
    string title;
    string content;
    string category;
    string tags;
    string priority;
    string timestamp;
    bool isEncrypted;
    string password;

public:
    Note(string t, string c, string cat, string tag, string prio, bool enc = false, string pass = "")
        : title(t), content(c), category(cat), tags(tag), priority(prio), isEncrypted(enc), password(pass) {
        timestamp = getCurrentTimestamp();
    }
    void displayNote(bool decrypt = false) const {
    if (!password.empty()) {  // Check if the note is password-protected
        string enteredPassword;
        cout << "\nThis note is password-protected. Please enter the password: ";
        cin >> enteredPassword;
        if (enteredPassword != password) {
            cout << "Incorrect Password. Cannot display the note." << endl;
            return;
        }
    }

    string displayContent = (decrypt && isEncrypted) ? encryptDecrypt(content) : content;
    cout << "\nTitle       : " << title << endl;
    cout << "Content     : " << displayContent << endl;
    cout << "Category    : " << category << endl;
    cout << "Tags        : " << tags << endl;
    cout << "Priority    : " << priority << endl;
    cout << "Timestamp   : " << timestamp << endl;
    cout << "\nPress Enter to return to the main menu...";
    cin.ignore();
    cin.get();
}
```


```

    void editContent(string newContent) {
        content = newContent;
        timestamp = getCurrentTimestamp();
    }
    void updatePriority(string newPriority) {
        priority = newPriority;
        timestamp = getCurrentTimestamp();
    }

    void toggleEncryption() {
        content = encryptDecrypt(content);
        isEncrypted = !isEncrypted;
    }

    void setPassword(string pass) {
        password = pass;
    }

    string getTitle() const { return title; }
    string getContent() const { return content; }
    string getCategory() const { return category; }
    string getTags() const { return tags; }
    string getPriority() const { return priority; }
    string getTimestamp() const { return timestamp; }
    bool getEncryptionStatus() const { return isEncrypted; }
    string getPassword() const { return password; }
};
```

- **Attributes**:

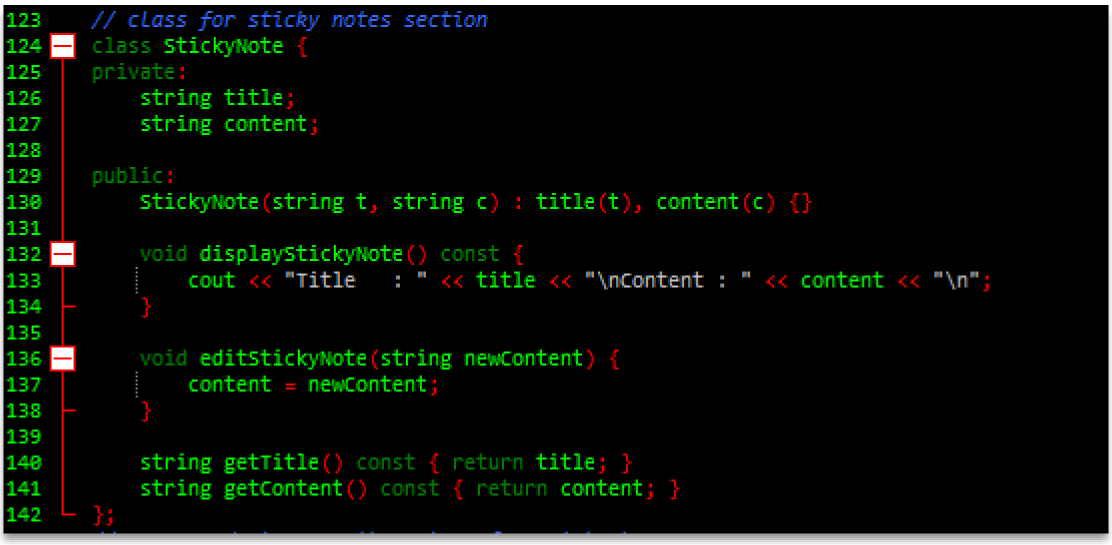
  - Here are the some attributes we have in class Note : **`title`**, **`content`**, **`category`**, **`tags`**, **`priority`**, **`timestamp`**, **`isEncrypted`**, **`password`**.

- **Methods**:

  - **`displayNote`:** Shows note details (decrypts if needed).
  - **`editContent`:** Updates content and timestamp.
  - **`updatePriority`:** Changes note priority.
  - **`toggleEncryption`:** Encrypts/Decrypts content.
  - **`setPassword`:** Assigns a password for encryption.

- Provides getter methods for all attributes.

c. Class: **StickyNote**



```cpp
// class for sticky notes section
class StickyNote {
private:
    string title;
    string content;

public:
    StickyNote(string t, string c) : title(t), content(c) {}

    void displayStickyNote() const {
        cout << "Title   : " << title << "\nContent : " << content << "\n";
    }

    void editStickyNote(string newContent) {
        content = newContent;
    }

    string getTitle() const { return title; }
    string getContent() const { return content; }
};
```

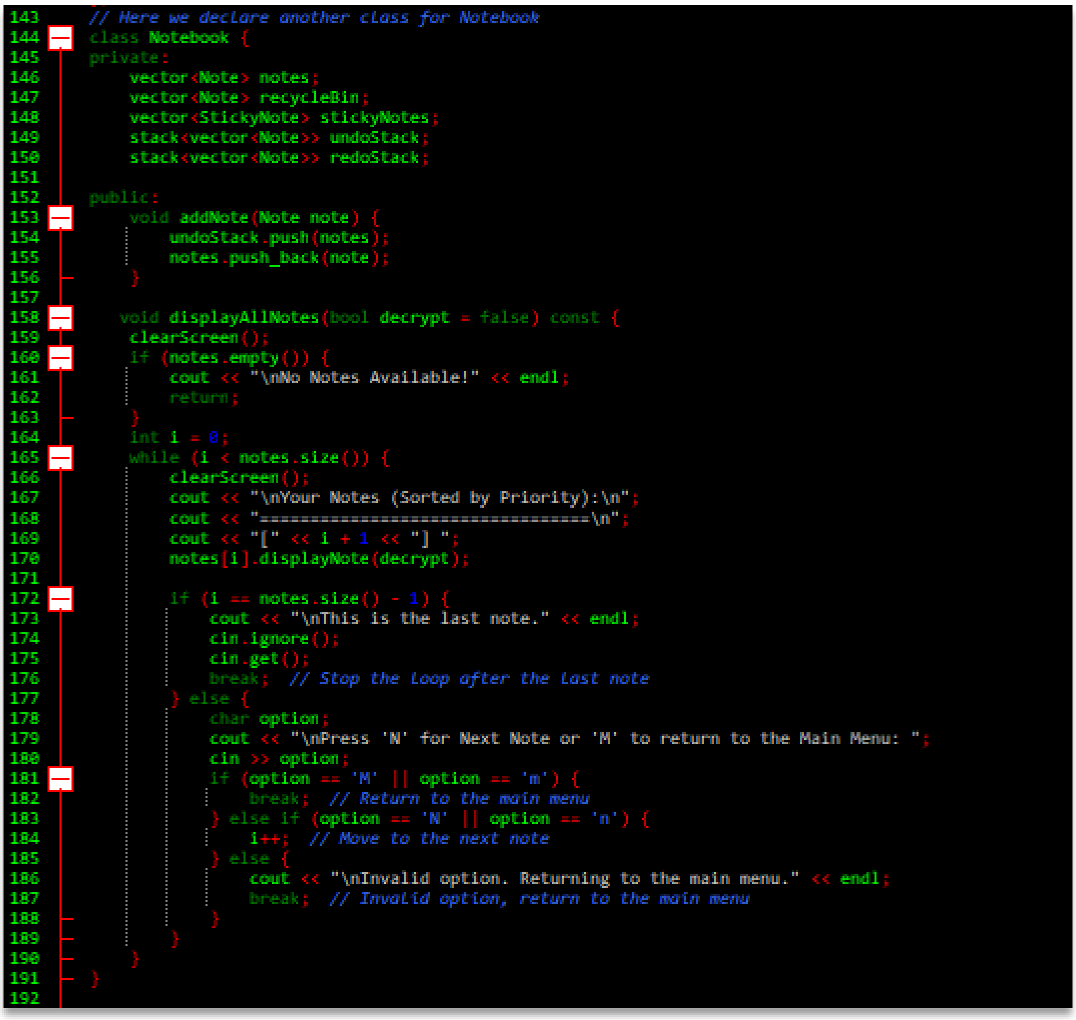
Handles simple, short notes without encryption.

- **Attributes**: `title`, `content`.
- **Methods**:

  - `displayStickyNote`: Displays sticky note details.
  - `editStickyNote`: Updates content.

d. Class: **Notebook**

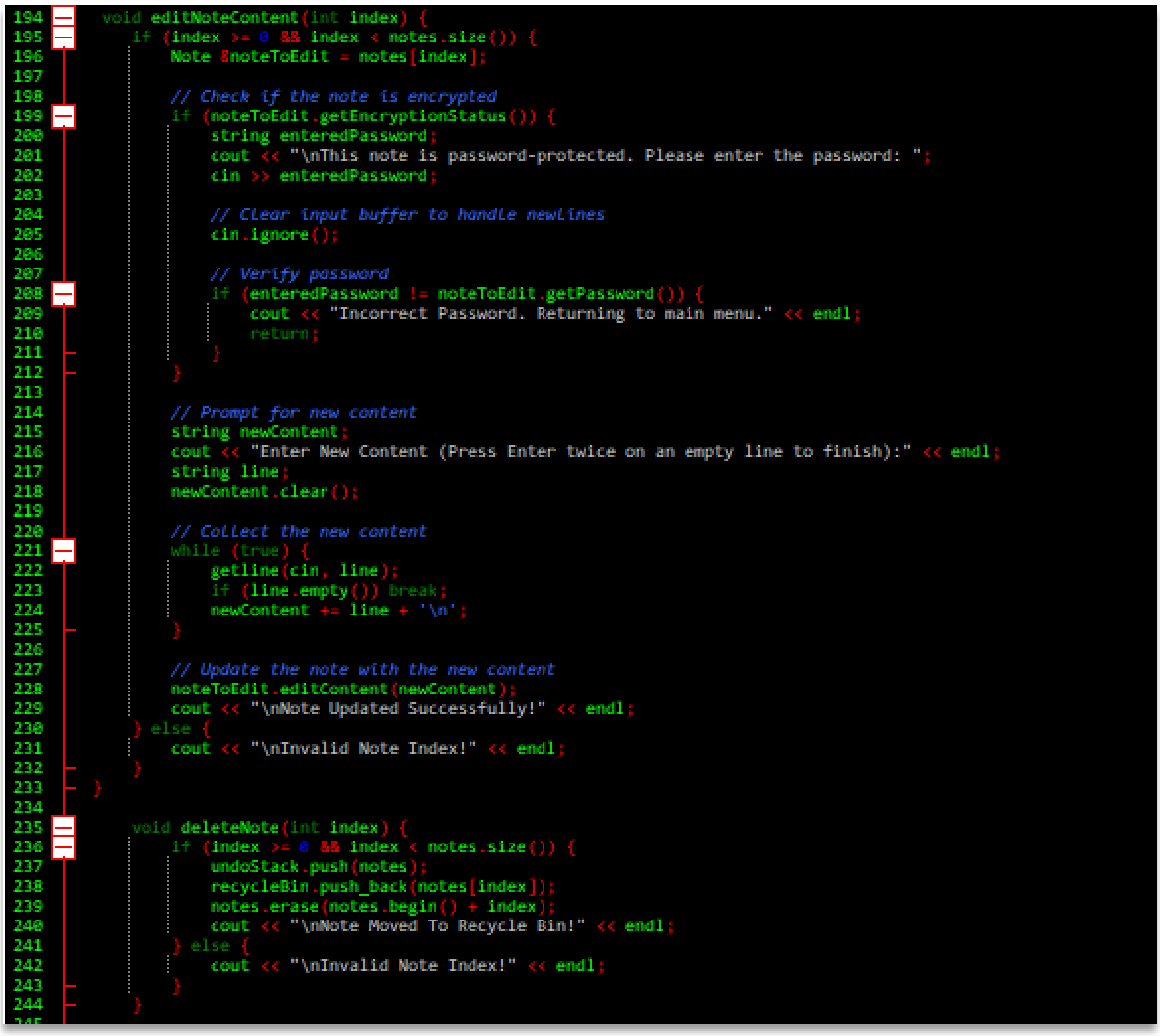
Manages a collection of `Note` and `StickyNote` objects and includes advanced functionalities:



```cpp
// Here we declare another class for Notebook
class Notebook {
private:
    vector<Note> notes;
    vector<Note> recycleBin;
    vector<StickyNote> stickyNotes;
    stack<vector<Note>> undoStack;
    stack<vector<Note>> redoStack;

public:
    void addNote(Note note) {
        undoStack.push(notes);
        notes.push_back(note);
    }

   void displayAllNotes(bool decrypt = false) const {
    clearScreen();
    if (notes.empty()) {
        cout << "\nNo Notes Available!" << endl;
        return;
    }
    int i = 0;
    while (i < notes.size()) {
        clearScreen();
        cout << "\nYour Notes (Sorted by Priority):\n";
        cout << "=================================\n";
        cout << "[" << i + 1 << "] ";
        notes[i].displayNote(decrypt);

        if (i == notes.size() - 1) {
            cout << "\nThis is the last note." << endl;
            cin.ignore();
            cin.get();
            break;  // Stop the Loop after the Last note
        } else {
            char option;
            cout << "\nPress 'N' for Next Note or 'M' to return to the Main Menu: ";
            cin >> option;
            if (option == 'M' || option == 'm') {
                break;  // Return to the main menu
            } else if (option == 'N' || option == 'n') {
                i++;  // Move to the next note
            } else {
                cout << "\nInvalid option. Returning to the main menu." << endl;
                break;  // Invalid option, return to the main menu
            }
        }
    }
}
```

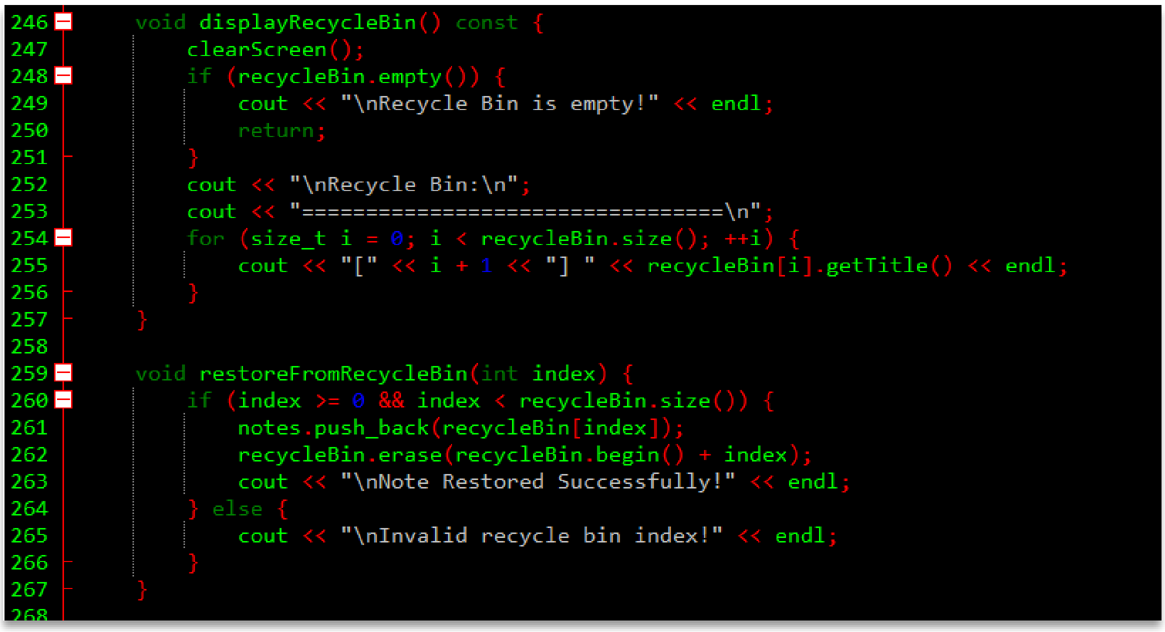
- **Attributes**:
  - `notes`, `recycleBin`, `stickyNotes`: Manage the notes.
  - `undoStack`, `redoStack`: Support undo/redo operations.

- **Methods**:
  - **Add/Edit/Delete**:
    - `addNote`, `editNoteContent`, `deleteNote`.



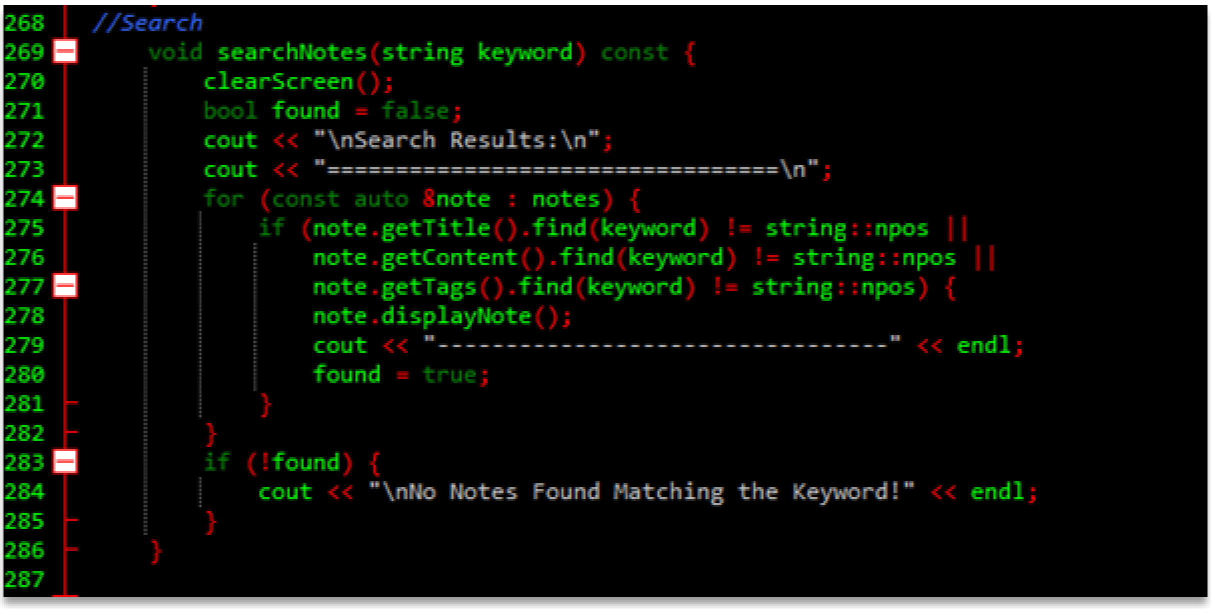
```
    void editNoteContent(int index) {
        if (index >= 0 && index < notes.size()) {
            Note &noteToEdit = notes[index];

            // Check if the note is encrypted
            if (noteToEdit.getEncryptionStatus()) {
                string enteredPassword;
                cout << "\nThis note is password-protected. Please enter the password: ";
                cin >> enteredPassword;

                // Clear input buffer to handle newlines
                cin.ignore();

                // Verify password
                if (enteredPassword != noteToEdit.getPassword()) {
                    cout << "Incorrect Password. Returning to main menu." << endl;
                    return;
                }
            }

            // Prompt for new content
            string newContent;
            cout << "Enter New Content (Press Enter twice on an empty line to finish):" << endl;
            string line;
            newContent.clear();

            // Collect the new content
            while (true) {
                getline(cin, line);
                if (line.empty()) break;
                newContent += line + '\n';
            }

            // Update the note with the new content
            noteToEdit.editContent(newContent);
            cout << "\nNote Updated Successfully!" << endl;
        } else {
            cout << "\nInvalid Note Index!" << endl;
        }
    }

        void deleteNote(int index) {
            if (index >= 0 && index < notes.size()) {
                undoStack.push(notes);
                recycleBin.push_back(notes[index]);
                notes.erase(notes.begin() + index);
                cout << "\nNote Moved To Recycle Bin!" << endl;
            } else {
                cout << "\nInvalid Note Index!" << endl;
            }
        }
```

- **Recycle Bin**:
    - displayRecycleBin, restoreFromRecycleBin.



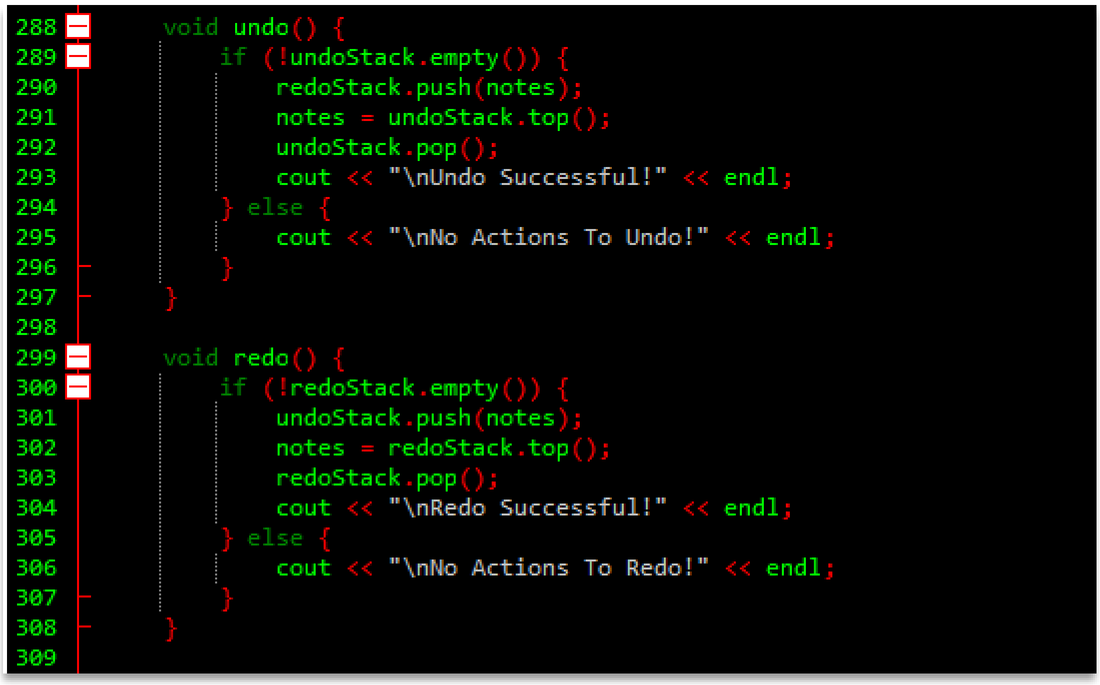
```
    void displayRecycleBin() const {
        clearScreen();
        if (recycleBin.empty()) {
            cout << "\nRecycle Bin is empty!" << endl;
            return;
        }
        cout << "\nRecycle Bin:\n";
        cout << "=================================\n";
        for (size_t i = 0; i < recycleBin.size(); ++i) {
            cout << "[" << i + 1 << "] " << recycleBin[i].getTitle() << endl;
        }
    }

    void restoreFromRecycleBin(int index) {
        if (index >= 0 && index < recycleBin.size()) {
            notes.push_back(recycleBin[index]);
            recycleBin.erase(recycleBin.begin() + index);
            cout << "\nNote Restored Successfully!" << endl;
        } else {
            cout << "\nInvalid recycle bin index!" << endl;
        }
    }
```

- **Search**:
    - `searchNotes` searches across `title`, `content`, `tags`.



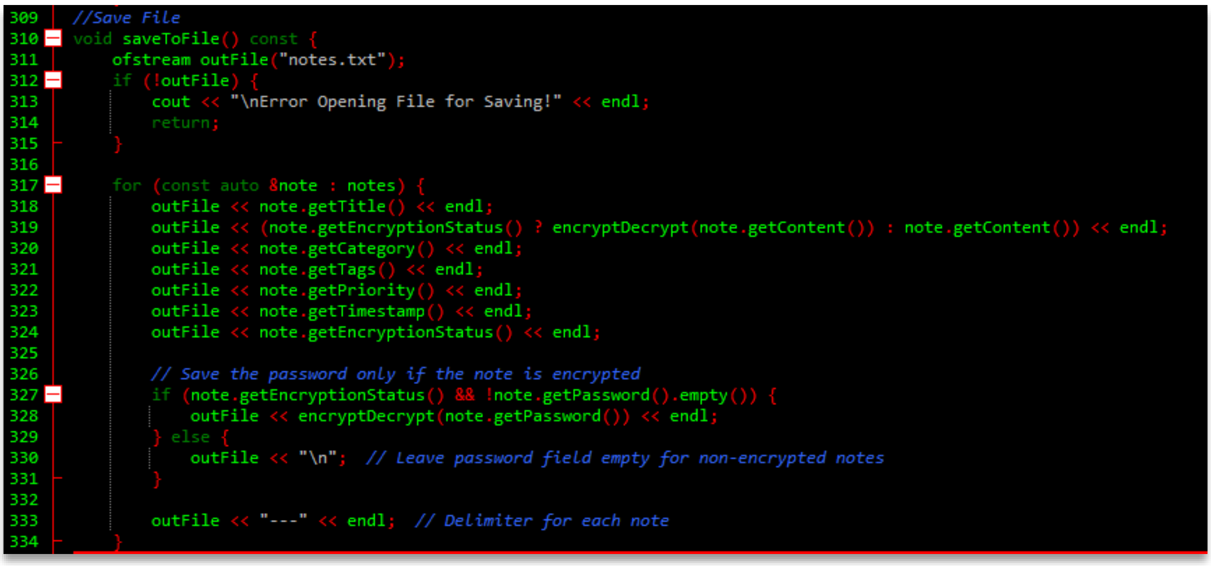
```
    //Search
        void searchNotes(string keyword) const {
            clearScreen();
            bool found = false;
            cout << "\nSearch Results:\n";
            cout << "=================================\n";
            for (const auto &note : notes) {
                if (note.getTitle().find(keyword) != string::npos ||
                    note.getContent().find(keyword) != string::npos ||
                    note.getTags().find(keyword) != string::npos) {
                    note.displayNote();
                    cout << "----------------------------------" << endl;
                    found = true;
                }
            }
            if (!found) {
                cout << "\nNo Notes Found Matching the Keyword!" << endl;
            }
        }
```

- **Undo/Redo**:
    - `undo`, `redo` using stacks for state management.

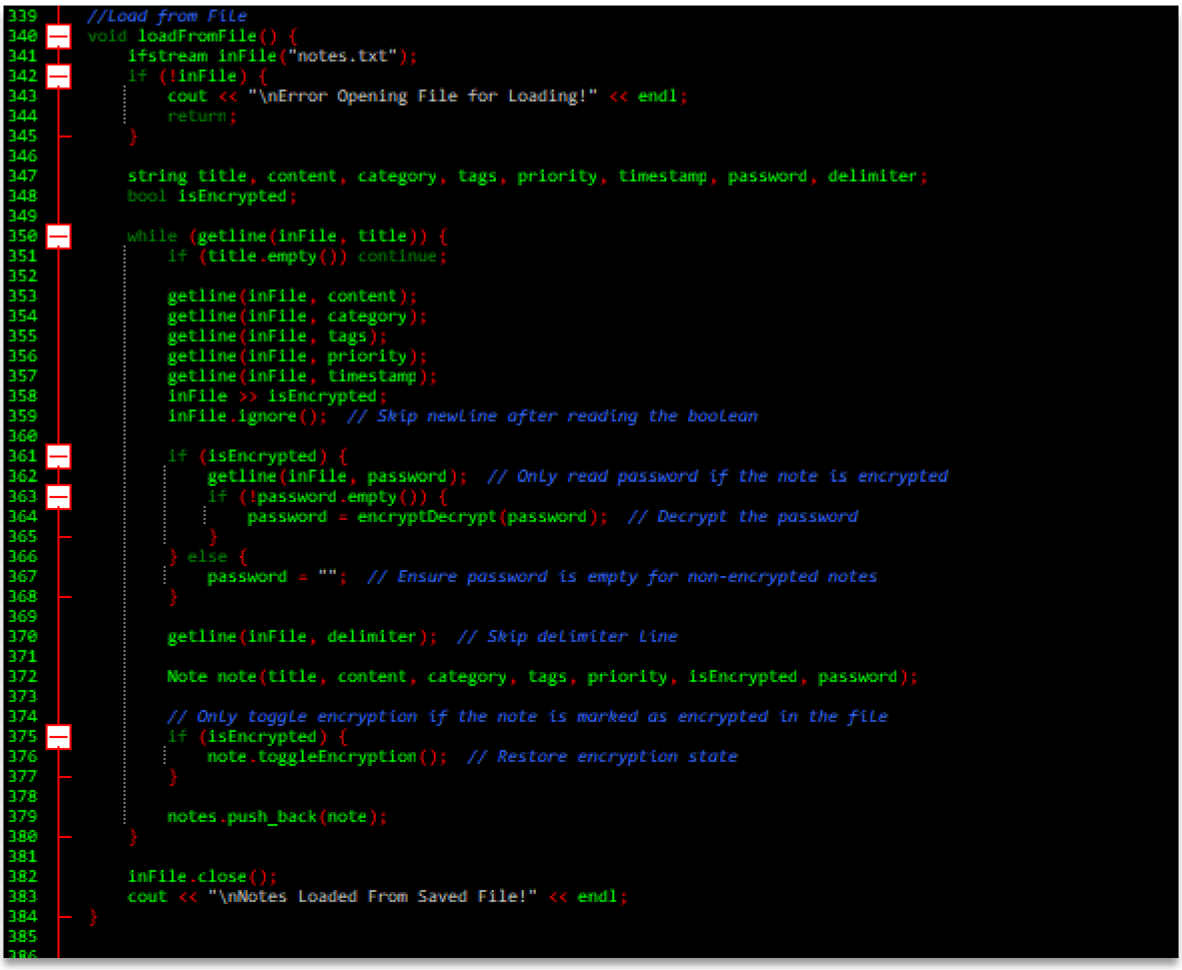


```
        void undo() {
            if (!undoStack.empty()) {
                redoStack.push(notes);
                notes = undoStack.top();
                undoStack.pop();
                cout << "\nUndo Successful!" << endl;
            } else {
                cout << "\nNo Actions To Undo!" << endl;
            }
        }

        void redo() {
            if (!redoStack.empty()) {
                undoStack.push(notes);
                notes = redoStack.top();
                redoStack.pop();
                cout << "\nRedo Successful!" << endl;
            } else {
                cout << "\nNo Actions To Redo!" << endl;
            }
        }
```

- **File Operations**:
    - `saveToFile`, `loadFromFile`.

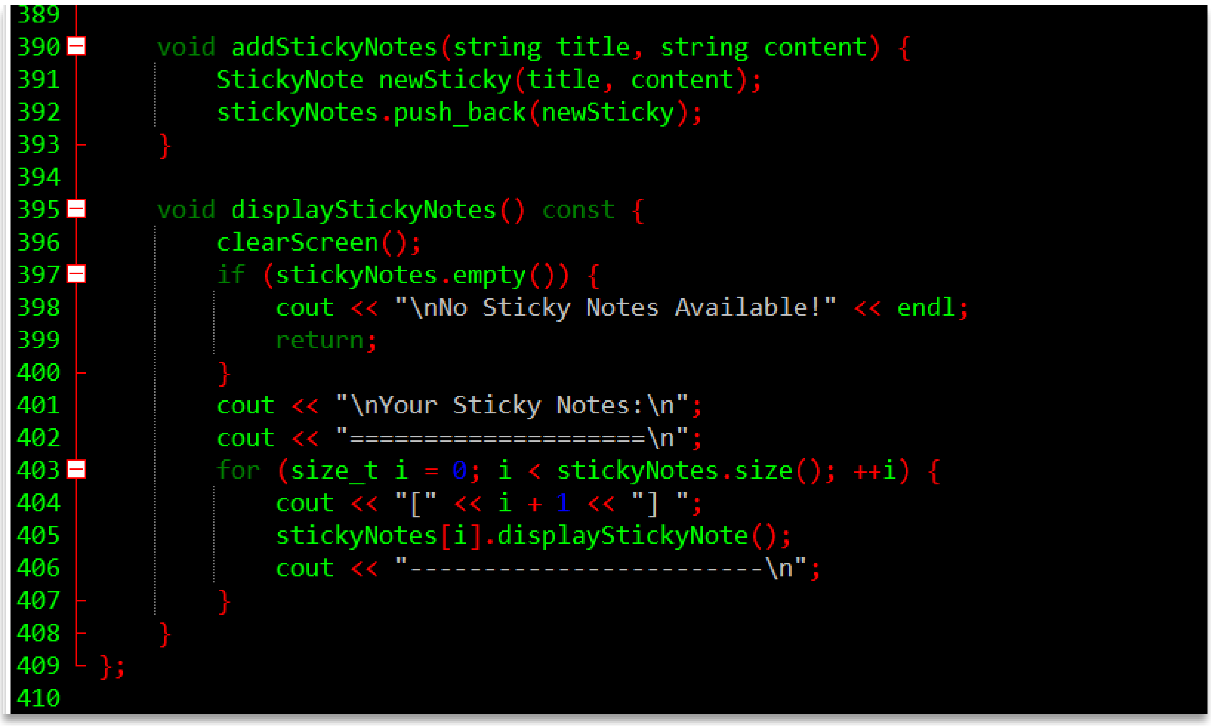


```cpp
//Save File
void saveToFile() const {
    ofstream outFile("notes.txt");
    if (!outFile) {
        cout << "\nError Opening File for Saving!" << endl;
        return;
    }

    for (const auto &note : notes) {
        outFile << note.getTitle() << endl;
        outFile << (note.getEncryptionStatus() ? encryptDecrypt(note.getContent()) : note.getContent()) << endl;
        outFile << note.getCategory() << endl;
        outFile << note.getTags() << endl;
        outFile << note.getPriority() << endl;
        outFile << note.getTimestamp() << endl;
        outFile << note.getEncryptionStatus() << endl;

        // Save the password only if the note is encrypted
        if (note.getEncryptionStatus() && !note.getPassword().empty()) {
            outFile << encryptDecrypt(note.getPassword()) << endl;
        } else {
            outFile << "\n";  // Leave password field empty for non-encrypted notes
        }

        outFile << "---" << endl;  // Delimiter for each note
    }
```



```cpp
//Load from File
void loadFromFile() {
    ifstream inFile("notes.txt");
    if (!inFile) {
        cout << "\nError Opening File for Loading!" << endl;
        return;
    }

    string title, content, category, tags, priority, timestamp, password, delimiter;
    bool isEncrypted;

    while (getline(inFile, title)) {
        if (title.empty()) continue;

        getline(inFile, content);
        getline(inFile, category);
        getline(inFile, tags);
        getline(inFile, priority);
        getline(inFile, timestamp);
        inFile >> isEncrypted;
        inFile.ignore();  // Skip newline after reading the boolean

        if (isEncrypted) {
            getline(inFile, password);  // Only read password if the note is encrypted
            if (!password.empty()) {
                password = encryptDecrypt(password);  // Decrypt the password
            }
        } else {
            password = "";  // Ensure password is empty for non-encrypted notes
        }

        getline(inFile, delimiter);  // Skip delimiter line

        Note note(title, content, category, tags, priority, isEncrypted, password);

        // Only toggle encryption if the note is marked as encrypted in the file
        if (isEncrypted) {
            note.toggleEncryption();  // Restore encryption state
        }

        notes.push_back(note);
    }

    inFile.close();
    cout << "\nNotes Loaded From Saved File!" << endl;
}
```

- **Sticky Notes**:
    - `addStickyNotes`, `displayStickyNotes`.

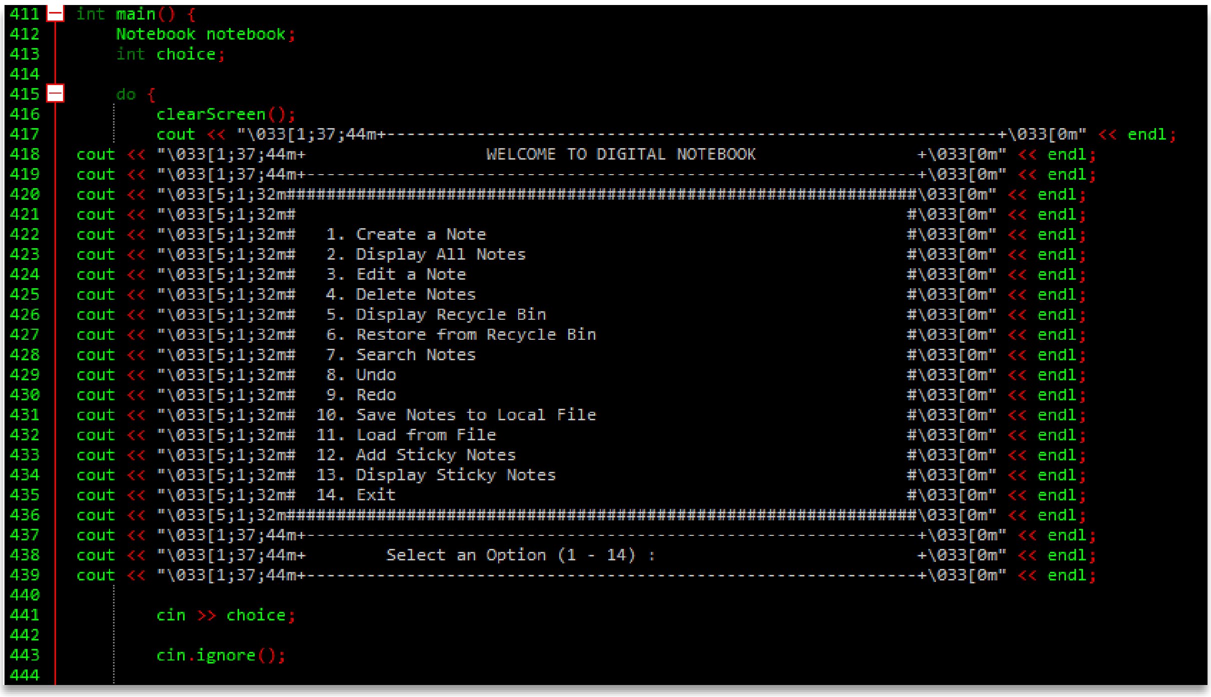


```cpp

    void addStickyNotes(string title, string content) {
        StickyNote newSticky(title, content);
        stickyNotes.push_back(newSticky);
    }

    void displayStickyNotes() const {
        clearScreen();
        if (stickyNotes.empty()) {
            cout << "\nNo Sticky Notes Available!" << endl;
            return;
        }
        cout << "\nYour Sticky Notes:\n";
        cout << "====================\n";
        for (size_t i = 0; i < stickyNotes.size(); ++i) {
            cout << "[" << i + 1 << "] ";
            stickyNotes[i].displayStickyNote();
            cout << "------------------------\n";
        }
    }
};

```

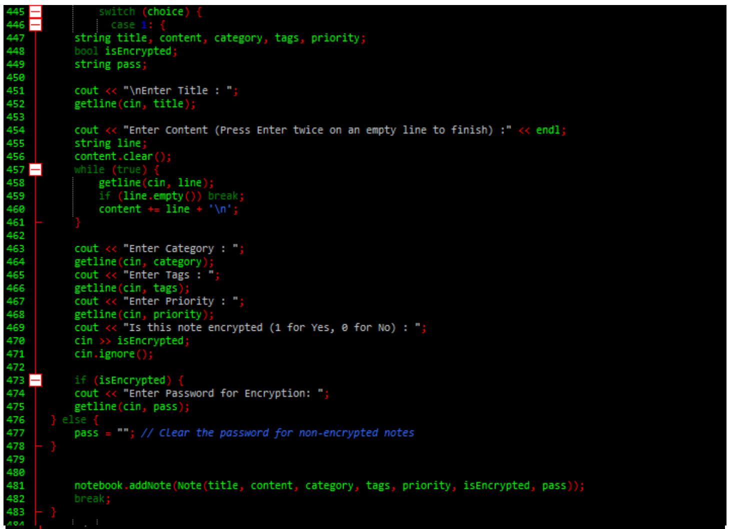
---

e. `main` Function

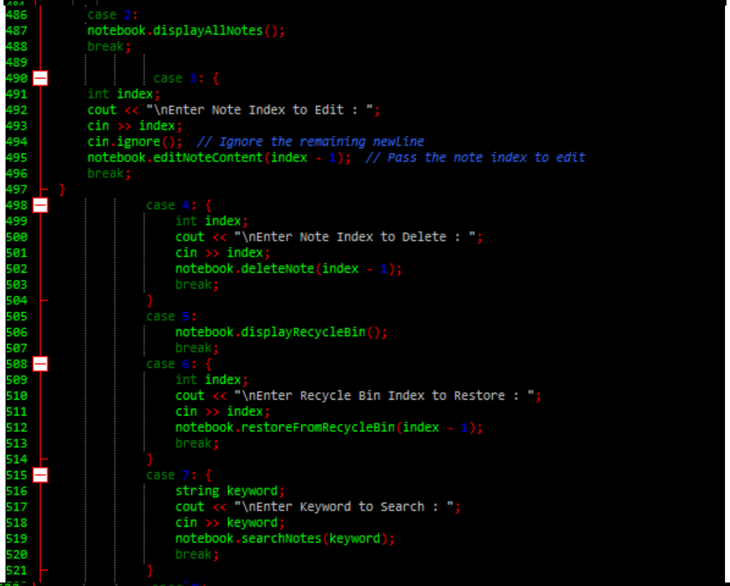
Implements the menu-driven interface:



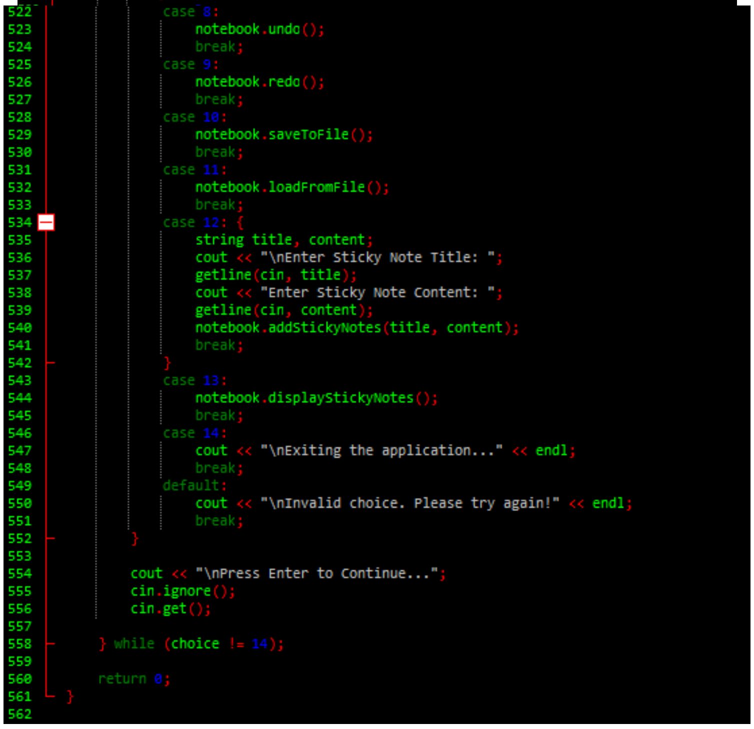
```cpp
int main() {
    Notebook notebook;
    int choice;

    do {
        clearScreen();
        cout << "\033[1;37;44m+-------------------------------------------------------------+\033[0m" << endl;
cout << "\033[1;37;44m+                  WELCOME TO DIGITAL NOTEBOOK                +\033[0m" << endl;
cout << "\033[1;37;44m+-------------------------------------------------------------+\033[0m" << endl;
cout << "\033[5;1;32m###############################################################\033[0m" << endl;
cout << "\033[5;1;32m#                                                              #\033[0m" << endl;
cout << "\033[5;1;32m#   1. Create a Note                                           #\033[0m" << endl;
cout << "\033[5;1;32m#   2. Display All Notes                                       #\033[0m" << endl;
cout << "\033[5;1;32m#   3. Edit a Note                                             #\033[0m" << endl;
cout << "\033[5;1;32m#   4. Delete Notes                                            #\033[0m" << endl;
cout << "\033[5;1;32m#   5. Display Recycle Bin                                     #\033[0m" << endl;
cout << "\033[5;1;32m#   6. Restore from Recycle Bin                                #\033[0m" << endl;
cout << "\033[5;1;32m#   7. Search Notes                                            #\033[0m" << endl;
cout << "\033[5;1;32m#   8. Undo                                                    #\033[0m" << endl;
cout << "\033[5;1;32m#   9. Redo                                                    #\033[0m" << endl;
cout << "\033[5;1;32m#  10. Save Notes to Local File                                #\033[0m" << endl;
cout << "\033[5;1;32m#  11. Load from File                                          #\033[0m" << endl;
cout << "\033[5;1;32m#  12. Add Sticky Notes                                        #\033[0m" << endl;
cout << "\033[5;1;32m#  13. Display Sticky Notes                                    #\033[0m" << endl;
cout << "\033[5;1;32m#  14. Exit                                                    #\033[0m" << endl;
cout << "\033[5;1;32m###############################################################\033[0m" << endl;
cout << "\033[1;37;44m+-------------------------------------------------------------+\033[0m" << endl;
cout << "\033[1;37;44m+        Select an Option (1 - 14) :                          +\033[0m" << endl;
cout << "\033[1;37;44m+-------------------------------------------------------------+\033[0m" << endl;

        cin >> choice;

        cin.ignore();

```



```cpp
            switch (choice) {
              case 1: {
        string title, content, category, tags, priority;
        bool isEncrypted;
        string pass;

        cout << "\nEnter Title : ";
        getline(cin, title);

        cout << "Enter Content (Press Enter twice on an empty line to finish) :" << endl;
        string line;
        content.clear();
        while (true) {
            getline(cin, line);
            if (line.empty()) break;
            content += line + '\n';
        }

        cout << "Enter Category : ";
        getline(cin, category);
        cout << "Enter Tags : ";
        getline(cin, tags);
        cout << "Enter Priority : ";
        getline(cin, priority);
        cout << "Is this note encrypted (1 for Yes, 0 for No) : ";
        cin >> isEncrypted;
        cin.ignore();

        if (isEncrypted) {
        cout << "Enter Password for Encryption: ";
        getline(cin, pass);
    } else {
        pass = ""; // Clear the password for non-encrypted notes
    }


        notebook.addNote(Note(title, content, category, tags, priority, isEncrypted, pass));
        break;
    }
```



```cpp
         case 2:
         notebook.displayAllNotes();
         break;

                    case 3: {
         int index;
         cout << "\nEnter Note Index to Edit : ";
         cin >> index;
         cin.ignore();  // Ignore the remaining newline
         notebook.editNoteContent(index - 1);  // Pass the note index to edit
         break;
     }
                   case 4: {
                       int index;
                       cout << "\nEnter Note Index to Delete : ";
                       cin >> index;
                       notebook.deleteNote(index - 1);
                       break;
                   }
                   case 5:
                       notebook.displayRecycleBin();
                       break;
                   case 6: {
                       int index;
                       cout << "\nEnter Recycle Bin Index to Restore : ";
                       cin >> index;
                       notebook.restoreFromRecycleBin(index - 1);
                       break;
                   }
                   case 7: {
                       string keyword;
                       cout << "\nEnter Keyword to Search : ";
                       cin >> keyword;
                       notebook.searchNotes(keyword);
                       break;
                   }
```



```cpp
                case 8:
                    notebook.undo();
                    break;
                case 9:
                    notebook.redo();
                    break;
                case 10:
                    notebook.saveToFile();
                    break;
                case 11:
                    notebook.loadFromFile();
                    break;
                case 12: {
                    string title, content;
                    cout << "\nEnter Sticky Note Title: ";
                    getline(cin, title);
                    cout << "Enter Sticky Note Content: ";
                    getline(cin, content);
                    notebook.addStickyNotes(title, content);
                    break;
                }
                case 13:
                    notebook.displayStickyNotes();
                    break;
                case 14:
                    cout << "\nExiting the application..." << endl;
                    break;
                default:
                    cout << "\nInvalid choice. Please try again!" << endl;
                    break;
            }

            cout << "\nPress Enter to Continue...";
            cin.ignore();
            cin.get();

        } while (choice != 14);

        return 0;
    }
```

- **Menu Options**:
    - Create, display, edit, delete, and search notes.
    - Manage recycle bin and sticky notes.
    - Undo/Redo actions.
    - Save/Load notes to/from a file.
    - Exit the application.
- **Structure**:
    - Loops until the user selects the exit option.
    - Uses appropriate methods from the `Notebook` class for each functionality.

3. Noteworthy Elements

a. Data Encryption

- XOR-based encryption ensures basic security for note content and passwords.

b. Undo/Redo

- Implemented using two stacks:
    - **`undoStack`** for saving the state before an action.
    - **`redoStack`** for restoring the state when an undo is reversed.

c. File Handling

- **Save**:
    - Serializes notes to a text file.
    - Handles encrypted and non-encrypted content appropriately.
- **Load**:
    - Deserializes notes from a text file.
    - Restores encryption state if applicable.

d. Timestamp Management

- Notes include a timestamp, updated after edits, ensuring proper record-keeping.

e. Responsive Design

- User prompts and navigation ensure an intuitive interface.
- Provides feedback for invalid inputs or actions.

6. Technologies Used

- C++: Core programming language.
- Stack and Vector: Efficient data handling for undo/redo and note management.

7. Conclusion

Our project Digital Notebook effectively combines note-taking functionality with security and user convenience. Its friendly design ensures efficient management of personal or professional data of users with features like note editing, undo, redo, edit recycle bin and password protection that ensures security and convince use.

8. Future Enhancements

Some of our future work we will work on contain these features:

- Adding a graphical user interface (GUI) for better interaction.
- Implementing cloud storage integration.
- Advanced encryption algorithms for enhanced security.
- Reminders
- Version Control

9. Team:

- Mokash Kumar: Follow on Github and LinkedIn.
- Muhammad Tarique: Follow on Github and LinkedIn.
- Naveed: Follow on Github and LinkedIn.
- Muhammad Azaan - 24CS54