Spekify Developer Documentation

Requirements to build and correctly run the program: -

- Must use a windows system (because of the use of <windows.h> header file)
- Should have “jazz.txt”, “jazz adjectives.txt”, “indiepop.txt”, “indie pop adjectives.txt”, “hiphop.txt”, and “hiphopadjectives.txt” in the same folder as the build file.

Data structures used in the code: -

- The program uses binary trees which are made based on the conditions as desired by the user.
- The program also makes extensive use of strings and arrays in functions that need them.

Structs used in the program:-

- Struct album data – For storing album data like year, name, artist name, int noartist (number of artists), rating (double), genre, moods, and int nomoods (number of moods). Artist names and moods are stored in a string of words; hence they are of the char ** type.
- Struct tree – It is used to build the tree, it contains albumdata album, and two tree pointers, one for the left node, and one for the right node.

Functions used and their description: -

Some common functions that are used extensively throughout the program: -

- Menucatch function – This function flushes the standard input, and attempts to scan an int type value. If the scan if successful (in case user entered an integer), it returns the integer. In case of an unsuccessful scan (as in case if user entered a character), it returns -1. The function is intended to avoid scanning errors in case user enters something other than an integer.
- Clear function – This function is used to clear the screen before every menu, or a tree is printed.
- Logo function – This function prints the “SPEKIFY” on every screen. Usually, it is called immediately after the clear function so that the first thing on the screen is the logo.
- Invalidchoice function – This function prints an error message (invalid choice).
- Int Err is used extensively throughout the program, which is used to check for invalid values entered by the user, and to display a message.

A simple pattern of functions is used to print and manage menus as follows: -

1. Imagine a simple menu
   1.1. Choice 1
   1.2. Choice 2
   1.3. Choice 3
   1.4. Exit.

   This menu will be handled by 3 functions:-

   1.1. Void menu() – The function that handles the printing of the menu, and invalid inputs.
   1.2. Void pmenu() – The function that prints the menu.

1.3. Int menuchoice(int x) – The function that handles the choice entered by the user.

The 3 functions will be interlinked as follows:-

The first function will clear the screen, print the logo, print the menu, and print an error message if err is 1. It will continue the loop until the value returned by the menuchoice option is equal to the value of the exit condition specified in the menu.

```
Void menu()

{

Do

{

Clear screen; (clear();)

Print logo; (logo();)

Print menu; (pmenu();)

If(err is 1) -> print invalid choice message, and set err to 0;

}while(menuchoice(menucatch())!=4); //menu exit condition

Return;

}


//Void pmenu only serves to print the menu.

Void pmenu()
{

//Just prints the menu;

Return;

}


//This function is used to actually make an action based on the choice inputted by the user.

//For the parameter, the above mentioned function int menuchoice() is called. Which returns with either the value entered by the user, or -1 in case of unsuccessful scans.

Int menuchoice(int x)
{

//We switch-case based on x to make an apt choice.

Switch(x)

{

Case 1:
```

//Executes choice 1

Break;

Case 2:

//Executes choice 2

Break;

Case 3:

//Executes choice 3

Break;

Case 4:

//Does nothing

Break;

Default:

Set err to 1 (so that when the control is returned back to the menu function, an error message can be printed)

}

Return x;

//So that when 4 is returned, the menu function can break the loop

}

A trilogy of such 3 functions is used to execute every menu in the application. It guarantees printing an error message, taking the suitable choice, and even handling the cases where no integer could be scanned.

Functions that are part of such trilogy are:-

- Void mainmenu, void pmainmenu, int main2sub(int c).
- Void theme, void ptheme, int pictheme(int x).
- Void vdb, void pvdb, int vdbchoice(int x).
- Void vsdb, void pvsdb, int vsdbchoice(int x).
- Void vedb, void pevdb, int vedbchoice(int x).
- Void yoyr, void pyoyr, int yoyrchoice(int x).
- Void ysortby(int y1, int y2, int x) //here y1 and y2 as parameters are the initial and final years of the range, and x is how you want the list sorted, void pysortby(), int ysortbychoice(int y1, int y2, int x)
- Void dosearch(char* name) //here name is what the user wants to search, void pdosearch, int dosearchchoice.

- Void loadit, void ploadit, int loaditchoice(char* name, int x) //Here name is the name of profile the user wishes to load, and x is how he wants to sort it.
- Void suggest, void psuggest, int suggestchoice. //This one is continued in next point
- Void suggestit, int pickadjec void sugg are continuation of the suggestion option where the user is asked to pick genre, adjectives, and is suggested an album.

Other more important functions include: -

- Void filenames (char *** filenames, char * x, int * size) – This function takes address of a string of arrays, a string, and address of an integer as parameters. It stores individual words from the string into the array of strings passed by address, and the size of that array in the integer passed by address.
- Void breakdown – Same thing as void filenames, except words are separated by '|' and not by space.
- Treeptr buildtree(char *x, int type, int sptype, int sp1, int sp2, char* sp3) – This function is used to build a tree. The char*x is the filename from which the tree will be built. The int type represents the type of tree which will be built. The value of int type is passed to the addnode function which then decides which data to compare to decide if the node will be added on the left or the right. The int sptype is used in case user wants a special type of tree (such as only a particular artist, or in a particular year range). The int sp1 and int sp2 denotes value of special parameters (such as year range) in case user wants a special tree. Char* sp3 is used to pass a special artist name, or album name, in case user wishes to make a search based tree, or of albums by only a particular artist. Only a single treeptr function is used to handle the building of every kind of tree in the program.
- Treeptr addnode(treeptr root, albumdata data, int type) – It adds a node to a tree with a particular album data. Int type decides which data to compare to decide at which node position will the album data be added.
- Void printree – This function prints "Nothing found" if the tree is empty, or further calls the printtree function to print the actual tree using inorder traversal
- Void printtree – This function prints the tree using inorder traversal.
- Compare name – Since the album name is stored in the form "Artist name – Album name (year)", a special compare name function is used to compare names of 2 albums, which compares the name after the '-', and returns a suitable integer value.
- Compare artist – Same as compare name but for comparing artist name.
- Compare genre – Compares genre case insensitively.
- Treeptr delete – This function deletes a tree.
- Customscan function – It is used to input a profile name of exactly the necessary size.

Functions that use filehandling: -

- Buildtree function uses file handling to read the data from a file.
- Createit function uses file handling to create a profile.