Developer Documentation

Explanation of the Solution

The **Event Management System** is a console-based application implemented in C. It uses linked lists to manage event records dynamically and binary files for storage. The program utilizes several libraries to create simple functions for efficient implementation.

Modules Overview

Single Program Module

1. **Authentication:** Handles user login and registration using binary file storage.
2. **Event Management**: Manages operations for adding, editing, deleting, and displaying events
3. **File Management**: Provides functionality for saving and loading data to/from binary files.
4. **User Interface** : Offers a console-based interface to interact with the user.

Data Structures

Event

```
typedef struct event{

        int year;

        int month;

        int day;

        char type[20];

        char category[20];

        char name[20];

        struct event* next;

}event;
```

Fields:

1. year: Event year (YYYY) as a single integer.
2. month: Event month (MM) as a single integer.
3. day: Event day (DD) as a single integer.

4. type: Event type (e.g. birthday, anniversary) as a character array (a string).
5. category: Event category (e.g. family, friend) as a character array (a string).
6. name: The name of the event as a character array (a string).
7. next: Pointer to the next event in the linked list.

Credentials

```
typedef struct credentials{

        char username[20];

        char password[20];

}credentials;
```

Fields:

1. username: The user's username.
2. password: The user's password.

Algorithms

Adding an Event:

1. Create a new event at the end of the linked list by allocating memory for it.
2. Input the event details received from the user.
3. End the linked list with a NULL.

Editing an Event:

1. Traverse the linked list to locate the event by name.
2. Modify the event details if found in the linked list.

Deleting an Event:

1. Traverse the linked list to locate the event by its name.
2. Adjust pointers to bypass and free the memory allocated for the specified event in order to remove it.

Displaying Events:

1. Traverse the linked list and filter based on the user's input (category or date).

Saving and Loading Data:

1. Use binary file operations to read/write the linked list into/from events.bin.

Functions and Interfaces

Authentication Functions

```
void save_credentials(credentials* cred)
```

1. **Input**: Credentials (username and password).
2. **Output**: Saves credentials in credentials.bin.

```
int authenticate()
```

1. **Input**: No input.
2. **Output**: Returns 1 if credentials match, returns 0 otherwise.

Event Management Functions

```
void add_event(event** head)
```

1. **Input**: Linked list head (linked list of event details).
2. **Output**: New head of the linked list (New head formed when an event is added).

```
void edit_event(event* head)
```

1. **Input**: Linked list head (linked list of event details, name of event is used to edit an existing event).
2. **Output**: Changes details of the specified event.

```
void delete_event(event** head)
```

1. **Input**: Linked list head (linked list of event details, name of event is used to delete an existing event).
2. **Output**: Updated head of the linked list.

```
void display_by_category(event* head)
```

1. **Input**: Linked list head (linked list of event details, linked list is traversed in search of a match to the category from the user).
2. **Output**: Prints events that match the category received from the user.

```
void display_by_date(event* head)
```

1. **Input**: Linked list head (linked list of event details, linked list is traversed in search of a match to the date from the user).
2. **Output**: Prints events that match the date received from the user.

File Management Functions

```
void load_database(event** head)
```

1. **Input**: Linked list head (pointer to a pointer of linked list of event details).
2. **Output**: Loaded linked list head.

```
void save_database(event* head)
```

1. **Input**: Linked list head (linked list of event details).
2. **Output**: Writes the linked list to the events.bin file.

```
int load_credentials(credentials* cred)
```

1. **Input**: Credentials (username and password).
2. **Output**: Returns 1 if credentials exist and have been read , 0 credentials do not exist or are unable to be read .

Memory Management Functions

```
void free_events(event* head)
```

1. **Input**: Linked list head.
2. **Output**: Frees all allocated memory.