Recipe Organizer App. (Full Stack Developer)

Project Overview: Develop a recipe organizer application where users can create a personal cookbook, add recipes, and categorize them. Each user should see only their own recipes after logging in. The app should feature an appealing interface with options to upload pictures and instructions for each recipe.

Key Features:

User signup and login system.

Add, delete, and edit recipes.

Organize recipes into categories.

Feature to search through one's own recipes.

Technical Specifications:

Local SQLite database to store user and recipe data.

React or Angular for the frontend to manage state effectively.

Effective use of contextBridge to manage data securely.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define MAX_RECIPES 100

#define MAX_TITLE_LEN 50

#define MAX_INGREDIENTS_LEN 200

#define MAX_INSTRUCTIONS_LEN 500


// Define a structure to store recipe information

typedef struct {

    char title[MAX_TITLE_LEN];

    char ingredients[MAX_INGREDIENTS_LEN];

    char instructions[MAX_INSTRUCTIONS_LEN];

} Recipe;


// Array to store all recipes and a counter for the number of recipes
```

```c
Recipe recipeBook[MAX_RECIPES];
int recipeCount = 0;

// Function to add a new recipe
void addRecipe() {
    if (recipeCount >= MAX_RECIPES) {
        printf("Recipe book is full!\n");
        return;
    }

    Recipe newRecipe;

    // Get the title of the recipe
    printf("Enter the title of the recipe: ");
    scanf(" %[^\n]", newRecipe.title);

    // Get the ingredients of the recipe
    printf("Enter the ingredients (comma separated): ");
    scanf(" %[^\n]", newRecipe.ingredients);

    // Get the instructions for the recipe
    printf("Enter the instructions: ");
    scanf(" %[^\n]", newRecipe.instructions);

    // Store the new recipe in the recipe book
    recipeBook[recipeCount++] = newRecipe;
    printf("Recipe added!\n");
}

// Function to display all recipes
void listRecipes() {
    if (recipeCount == 0) {
        printf("No recipes available.\n");
        return;
    }

    // Loop through the recipes and print their details
    for (int i = 0; i < recipeCount; i++) {
        printf("\nRecipe %d:\n", i + 1);
        printf("Title: %s\n", recipeBook[i].title);
        printf("Ingredients: %s\n", recipeBook[i].ingredients);
        printf("Instructions: %s\n", recipeBook[i].instructions);
    }
}

// Function to display the menu and handle user choices
void menu() {
    int choice;

    do {
        printf("\nRecipe Organizer\n");
        printf("1. Add a Recipe\n");
        printf("2. View Recipes\n");
        printf("3. Exit\n");
        printf("Choose an option: ");
        scanf("%d", &choice);
        getchar(); // Clear the newline character left by scanf

        switch (choice) {
            case 1:
                addRecipe();
                break;
            case 2:
                listRecipes();
                break;
            case 3:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid option. Please try again.\n");
        }
    } while (choice != 3);
}

// Main function to start the program
int main() {
    printf("Welcome to the Recipe Organizer!\n");
    menu();
    return 0;
}
```