```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include <time.h>

#include <conio.h> // For password masking on Windows


// User structure

struct User {

   char name[50];

   char id[20];

   char email[50];

   char password[50];

};


// Product structure

struct Product {

   char name[50];

   float price;

   int quantity;

   char category[50];

   float discount;

   float rating;

};


// Order structure
```

```
struct Order {
    char userEmail[50];
    char productName[50];
    int quantity;
    float totalPrice;
    char date[20];
};

// Cart item structure
struct CartItem {
    char productName[50];
    int quantity;
};

// Feedback structure
struct Feedback {
    char userEmail[50];
    char comment[200];
};

// Global variables
struct User currentUser;
int isAdmin = 0;

// Data storage
struct User users[100];
```

```
int userCount = 0;

struct Product products[100];
int productCount = 0;

struct Order orders[100];
int orderCount = 0;

struct Feedback feedbacks[100];
int feedbackCount = 0;

// Function prototypes
void registerUser();
int loginUser();
void addProduct();
void viewProducts();
void searchProduct();
void editProduct();
void deleteProduct();
void sortProductsByPrice();
void sortProductsByQuantity();
void addToCart(struct CartItem *cart, int *cartSize);
void removeFromCart(struct CartItem *cart, int *cartSize);
void viewCart(struct CartItem *cart, int cartSize);
void checkout(struct CartItem *cart, int *cartSize);
void viewOrderHistory();
void updateUserInfo();

void deleteAccount();

void viewAllUsers();

void adminDashboard();

void logout();

int emailValidation(char *email);

int passwordValidation(char *password);

void getPasswordWithMask(char *password);

void applyDiscount();

void rateProduct();

void submitFeedback();

void viewAllFeedbacks();

int isEmailUnique(char *email);

void saveDataToFile();

void loadDataFromFile();

void displayMainMenu();

void displayAdminMenu();

void displayUserMenu();

void generateReceipt(float total);

void viewProductCategories();

void filterProductsByCategory();

void changePassword();


int main() {

    int choice;

    int loggedIn = 0;
```

```
struct CartItem cart[100];
int cartSize = 0;

loadDataFromFile(); // Load data from files at startup

while (1) {
    if (!loggedIn) {
        displayMainMenu();
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                registerUser();
                break;
            case 2:
                loggedIn = loginUser();
                break;
            case 3:
                printf("Exiting...\n");
                saveDataToFile();
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    } else {
        if (isAdmin) {
```

```
        displayAdminMenu();
        scanf("%d", &choice);

        switch (choice) {
            case 1: addProduct(); break;
            case 2: viewProducts(); break;
            case 3: searchProduct(); break;
            case 4: editProduct(); break;
            case 5: deleteProduct(); break;
            case 6: sortProductsByPrice(); break;
            case 7: sortProductsByQuantity(); break;
            case 8: viewAllUsers(); break;
            case 9: adminDashboard(); break;
            case 10: applyDiscount(); break;
            case 11: viewAllFeedbacks(); break;
            case 12: viewProductCategories(); break;
            case 13: filterProductsByCategory(); break;
            case 14: logout(); loggedIn = 0; break;
            case 15: printf("Exiting...\n"); saveDataToFile(); exit(0);
            default: printf("Invalid choice! Please try again.\n");
        }
    } else {
        displayUserMenu();
        scanf("%d", &choice);

        switch (choice) {
}

                case 1: viewProducts(); break;
                case 2: searchProduct(); break;
                case 3: sortProductsByPrice(); break;
                case 4: sortProductsByQuantity(); break;
                case 5: addToCart(cart, &cartSize); break;
                case 6: removeFromCart(cart, &cartSize); break;
                case 7: viewCart(cart, cartSize); break;
                case 8: checkout(cart, &cartSize); break;
                case 9: viewOrderHistory(); break;
                case 10: updateUserInfo(); break;
                case 11: deleteAccount(); loggedIn = 0; break;
                case 12: rateProduct(); break;
                case 13: submitFeedback(); break;
                case 14: viewProductCategories(); break;
                case 15: filterProductsByCategory(); break;
                case 16: changePassword(); break;
                case 17: logout(); loggedIn = 0; break;
                case 18: printf("Exiting...\n"); saveDataToFile(); exit(0);
                default: printf("Invalid choice! Please try again.\n");
            }
        }
    }
  }

  return 0;
}
```

```
void displayMainMenu() {
    printf("\n--- Grocery Management System ---\n");
    printf("1. Register\n");
    printf("2. Login\n");
    printf("3. Exit\n");
    printf("Enter your choice: ");
}

void displayAdminMenu() {
    printf("\n--- Admin Menu ---\n");
    printf("1. Add Product\n");
    printf("2. View Products\n");
    printf("3. Search Product\n");
    printf("4. Edit Product\n");
    printf("5. Delete Product\n");
    printf("6. Sort Products by Price\n");
    printf("7. Sort Products by Quantity\n");
    printf("8. View All Users\n");
    printf("9. Admin Dashboard\n");
    printf("10. Apply Discount\n");
    printf("11. View All Feedbacks\n");
    printf("12. View Product Categories\n");
    printf("13. Filter Products by Category\n");
    printf("14. Logout\n");
    printf("15. Exit\n");
    printf("Enter your choice: ");
}

void displayUserMenu() {
    printf("\n--- User Menu ---\n");
    printf("1. View Products\n");
    printf("2. Search Product\n");
    printf("3. Sort Products by Price\n");
    printf("4. Sort Products by Quantity\n");
    printf("5. Add to Cart\n");
    printf("6. Remove from Cart\n");
    printf("7. View Cart\n");
    printf("8. Checkout\n");
    printf("9. View Order History\n");
    printf("10. Update User Info\n");
    printf("11. Delete Account\n");
    printf("12. Rate Product\n");
    printf("13. Submit Feedback\n");
    printf("14. View Product Categories\n");
    printf("15. Filter Products by Category\n");
    printf("16. Change Password\n");
    printf("17. Logout\n");
    printf("18. Exit\n");
    printf("Enter your choice: ");
}
```

```
void registerUser() {
    struct User newUser;
    printf("Enter Name: ");
    scanf("%s", newUser.name);
    printf("Enter ID: ");
    scanf("%s", newUser.id);

    do {
        printf("Enter Email: ");
        scanf("%s", newUser.email);
        if (!emailValidation(newUser.email)) {
            printf("Invalid email format. Please try again.\n");
        } else if (!isEmailUnique(newUser.email)) {
            printf("Email already exists. Please use a different email.\n");
        }
    } while (!emailValidation(newUser.email) || !isEmailUnique(newUser.email));

    do {
        printf("Enter Password (must contain uppercase, lowercase and number): ");
        getPasswordWithMask(newUser.password);
        if (!passwordValidation(newUser.password)) {
            printf("Password must contain at least one uppercase letter, one lowercase letter, and
one number.\n");
        }
    } while (!passwordValidation(newUser.password));
    users[userCount] = newUser;

    userCount++;

    printf("Registration successful!\n");

    saveDataToFile();

}


int loginUser() {

    char email[50], password[50];

    printf("Enter Email: ");

    scanf("%s", email);

    printf("Enter Password: ");

    getPasswordWithMask(password);


    for (int i = 0; i < userCount; i++) {

        if (strcmp(users[i].email, email) == 0 && strcmp(users[i].password, password) == 0) {

            currentUser = users[i];

            printf("Login successful! Welcome, %s.\n", users[i].name);

            if (strcmp(users[i].email, "admin@example.com") == 0 ||

                strcmp(users[i].email, "saifullah@example.com") == 0 ||

                strcmp(users[i].email, "sabina@example.com") == 0 ||

                strcmp(users[i].email, "farzana@example.com") == 0) {

                isAdmin = 1;

            }

            return 1;

        }

    }
    printf("Invalid email or password.\n");
    return 0;
}

void getPasswordWithMask(char *password) {
    int i = 0;
    char ch;

    while (1) {
        ch = getch();

        if (ch == 13) { // Enter key
            password[i] = '\0';
            break;
        } else if (ch == 8) { // Backspace
            if (i > 0) {
                i--;
                printf("\b \b");
            }
        } else {
            password[i] = ch;
            i++;
            printf("*");
        }
    }
    printf("\n");
```

```
}

int emailValidation(char *email) {
    int atFound = 0, dotFound = 0;
    for (int i = 0; email[i] != '\0'; i++) {
        if (email[i] == '@') atFound = 1;
        if (atFound && email[i] == '.') dotFound = 1;
    }
    return atFound && dotFound;
}

int isEmailUnique(char *email) {
    for (int i = 0; i < userCount; i++) {
        if (strcmp(users[i].email, email) == 0) {
            return 0;
        }
    }
    return 1;
}

int passwordValidation(char *password) {
    int hasUpper = 0, hasLower = 0, hasDigit = 0;

    for (int i = 0; password[i] != '\0'; i++) {
        if (isupper(password[i])) hasUpper = 1;
        if (islower(password[i])) hasLower = 1;
        if (isdigit(password[i])) hasDigit = 1;
    }

    return hasUpper && hasLower && hasDigit;
}

void addProduct() {
    struct Product newProduct;
    printf("Enter Product Name: ");
    scanf("%s", newProduct.name);
    printf("Enter Price: ");
    scanf("%f", &newProduct.price);
    printf("Enter Quantity: ");
    scanf("%d", &newProduct.quantity);
    printf("Enter Category: ");
    scanf("%s", newProduct.category);
    newProduct.discount = 0.0;
    newProduct.rating = 0.0;

    products[productCount] = newProduct;
    productCount++;
    printf("Product added successfully!\n");
    saveDataToFile();
}

void viewProducts() {
    printf("\n--- All Products ---\n");
    for (int i = 0; i < productCount; i++) {
        printf("Name: %s, Price: $%.2f, Quantity: %d, Category: %s, Discount: %.2f%%,
Rating: %.1f\n",
               products[i].name, products[i].price, products[i].quantity, products[i].category,
               products[i].discount, products[i].rating);
    }
}

void searchProduct() {
    char productName[50];
    printf("Enter Product Name: ");
    scanf("%s", productName);

    int found = 0;
    for (int i = 0; i < productCount; i++) {
        if (strcmp(products[i].name, productName) == 0) {
            printf("Product Found: Name: %s, Price: $%.2f, Quantity: %d, Category: %s,
Discount: %.2f%%, Rating: %.1f\n",
                   products[i].name, products[i].price, products[i].quantity, products[i].category,
                   products[i].discount, products[i].rating);
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Product not found.\n");
    }
}

void editProduct() {
    char productName[50];
    printf("Enter Product Name to Edit: ");
    scanf("%s", productName);

    int found = 0;
    for (int i = 0; i < productCount; i++) {
        if (strcmp(products[i].name, productName) == 0) {
            found = 1;
            printf("Enter New Price: ");
            scanf("%f", &products[i].price);
            printf("Enter New Quantity: ");
            scanf("%d", &products[i].quantity);
            printf("Enter New Category: ");
            scanf("%s", products[i].category);
            printf("Product updated successfully!\n");
            saveDataToFile();
            break;
        }
    }
    if (!found) {
        printf("Product not found.\n");
    }
```

```
}


void deleteProduct() {
    char productName[50];
    printf("Enter Product Name to Delete: ");
    scanf("%s", productName);


    int found = 0;
    for (int i = 0; i < productCount; i++) {
        if (strcmp(products[i].name, productName) == 0) {
            found = 1;
            for (int j = i; j < productCount - 1; j++) {
                products[j] = products[j + 1];
            }
            productCount--;
            printf("Product deleted successfully!\n");
            saveDataToFile();
            break;
        }
    }
    if (!found) {
        printf("Product not found.\n");
    }
}


void sortProductsByPrice() {
    for (int i = 0; i < productCount - 1; i++) {
        for (int j = 0; j < productCount - i - 1; j++) {
            if (products[j].price > products[j + 1].price) {
                struct Product temp = products[j];
                products[j] = products[j + 1];
                products[j + 1] = temp;
            }
        }
    }

    printf("\n--- Products Sorted by Price ---\n");
    viewProducts();
}

void sortProductsByQuantity() {
    for (int i = 0; i < productCount - 1; i++) {
        for (int j = 0; j < productCount - i - 1; j++) {
            if (products[j].quantity > products[j + 1].quantity) {
                struct Product temp = products[j];
                products[j] = products[j + 1];
                products[j + 1] = temp;
            }
        }
    }

    printf("\n--- Products Sorted by Quantity ---\n");
    viewProducts();
}

void addToCart(struct CartItem *cart, int *cartSize) {
    char productName[50];
    int quantity;
    printf("Enter Product Name: ");
    scanf("%s", productName);
    printf("Enter Quantity: ");
    scanf("%d", &quantity);

    int found = 0;
    for (int i = 0; i < productCount; i++) {
        if (strcmp(products[i].name, productName) == 0) {
            found = 1;
            if (products[i].quantity >= quantity) {
                strcpy(cart[*cartSize].productName, productName);
                cart[*cartSize].quantity = quantity;
                (*cartSize)++;
                printf("Product added to cart!\n");
            } else {
                printf("Insufficient stock!\n");
            }
            break;
        }
    }
    if (!found) {
        printf("Product not found.\n");
    }
}

void removeFromCart(struct CartItem *cart, int *cartSize) {
    char productName[50];
    printf("Enter Product Name to Remove: ");
    scanf("%s", productName);

    int found = 0;
    for (int i = 0; i < *cartSize; i++) {
        if (strcmp(cart[i].productName, productName) == 0) {
            found = 1;
            for (int j = i; j < *cartSize - 1; j++) {
                cart[j] = cart[j + 1];
            }
            (*cartSize)--;
            printf("Product removed from cart!\n");
            break;
        }
    }
    if (!found) {
        printf("Product not found in cart.\n");
    }
}
```

```
void viewCart(struct CartItem *cart, int cartSize) {
    if (cartSize == 0) {
        printf("Your cart is empty.\n");
        return;
    }

    printf("\n--- Your Cart ---\n");
    float total = 0;
    for (int i = 0; i < cartSize; i++) {
        for (int j = 0; j < productCount; j++) {
            if (strcmp(cart[i].productName, products[j].name) == 0) {
                float itemPrice = products[j].price * (1 - products[j].discount / 100);
                printf("%d. Product: %s, Quantity: %d, Price: $%.2f each, Total: $%.2f\n",
                       i+1, cart[i].productName, cart[i].quantity, itemPrice, itemPrice * cart[i].quantity);
                total += itemPrice * cart[i].quantity;
                break;
            }
        }
    }
    printf("Total: $%.2f\n", total);
}

void checkout(struct CartItem *cart, int *cartSize) {
    if (*cartSize == 0) {
        printf("Your cart is empty!\n");
        return;
    }

    float total = 0;
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    char date[20];
    sprintf(date, "%02d-%02d-%04d", tm.tm_mday, tm.tm_mon + 1, tm.tm_year + 1900);

    for (int i = 0; i < *cartSize; i++) {
        for (int j = 0; j < productCount; j++) {
            if (strcmp(cart[i].productName, products[j].name) == 0) {
                float discountedPrice = products[j].price * (1 - products[j].discount / 100);
                total += discountedPrice * cart[i].quantity;
                products[j].quantity -= cart[i].quantity;

                // Add to orders
                strcpy(orders[orderCount].userEmail, currentUser.email);
                strcpy(orders[orderCount].productName, cart[i].productName);
                orders[orderCount].quantity = cart[i].quantity;
                orders[orderCount].totalPrice = discountedPrice * cart[i].quantity;
                strcpy(orders[orderCount].date, date);
                orderCount++;
                break;
            }
        }
    }

    generateReceipt(total);
    *cartSize = 0;
    printf("Checkout successful! Thank you for your purchase.\n");
    saveDataToFile();
}

void generateReceipt(float total) {
    printf("\n=== RECEIPT ===\n");
    printf("Customer: %s\n", currentUser.name);
    printf("Email: %s\n", currentUser.email);
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Date: %02d-%02d-%04d\n", tm.tm_mday, tm.tm_mon + 1, tm.tm_year + 1900);
    printf("Total: $%.2f\n", total);
    printf("Thank you for shopping with us!\n");
    printf("===============\n");
}

void viewOrderHistory() {
    printf("\n--- Order History ---\n");
    int found = 0;
    for (int i = 0; i < orderCount; i++) {
        if (strcmp(orders[i].userEmail, currentUser.email) == 0) {
            printf("Product: %s, Quantity: %d, Total: $%.2f, Date: %s\n",
                orders[i].productName, orders[i].quantity, orders[i].totalPrice, orders[i].date);
            found = 1;
        }
    }
    if (!found) {
        printf("No orders found.\n");
    }
}

void updateUserInfo() {
    char newName[50], newId[20];
    printf("Enter new name: ");
    scanf("%s", newName);
    printf("Enter new ID: ");
    scanf("%s", newId);

    for (int i = 0; i < userCount; i++) {
        if (strcmp(users[i].email, currentUser.email) == 0) {
            strcpy(users[i].name, newName);
            strcpy(users[i].id, newId);
            strcpy(currentUser.name, newName);
            strcpy(currentUser.id, newId);
            printf("User info updated successfully!\n");
            saveDataToFile();
            return;
        }
    }
    printf("Error updating user info.\n");
}

void changePassword() {
    char currentPassword[50], newPassword[50];
    printf("Enter current password: ");
    getPasswordWithMask(currentPassword);

    if (strcmp(currentUser.password, currentPassword) != 0) {
        printf("Incorrect current password.\n");
        return;
    }

    do {
        printf("Enter new password (must contain uppercase, lowercase and number): ");
        getPasswordWithMask(newPassword);
        if (!passwordValidation(newPassword)) {
            printf("Password must contain at least one uppercase letter, one lowercase letter, and
one number.\n");
        }
    } while (!passwordValidation(newPassword));

    for (int i = 0; i < userCount; i++) {
        if (strcmp(users[i].email, currentUser.email) == 0) {
            strcpy(users[i].password, newPassword);
            strcpy(currentUser.password, newPassword);
            printf("Password changed successfully!\n");
            saveDataToFile();
            return;
        }
    }
}

void deleteAccount() {
    char confirm[5];
    printf("Are you sure you want to delete your account? (yes/no): ");
    scanf("%s", confirm);

    if (strcmp(confirm, "yes") != 0) {
        printf("Account deletion cancelled.\n");
        return;
    }

    for (int i = 0; i < userCount; i++) {
        if (strcmp(users[i].email, currentUser.email) == 0) {
            for (int j = i; j < userCount - 1; j++) {
                users[j] = users[j + 1];
            }
            userCount--;
            printf("Account deleted successfully!\n");
            saveDataToFile();
            return;
        }
    }
    printf("Error deleting account.\n");
}

void viewAllUsers() {
    printf("\n--- All Users ---\n");
    for (int i = 0; i < userCount; i++) {
        printf("Name: %s, ID: %s, Email: %s\n", users[i].name, users[i].id, users[i].email);
    }
}

void adminDashboard() {
    printf("\n--- Admin Dashboard ---\n");
    printf("Total Users: %d\n", userCount);
    printf("Total Products: %d\n", productCount);
    printf("Total Orders: %d\n", orderCount);
    printf("Total Feedbacks: %d\n", feedbackCount);

    float totalRevenue = 0;
    for (int i = 0; i < orderCount; i++) {
        totalRevenue += orders[i].totalPrice;
    }
    printf("Total Revenue: $%.2f\n", totalRevenue);
}
```

```
void logout() {
    strcpy(currentUser.name, "");
    strcpy(currentUser.email, "");
    strcpy(currentUser.id, "");
    strcpy(currentUser.password, "");
    isAdmin = 0;
    printf("Logged out successfully!\n");
}

void applyDiscount() {
    char productName[50];
    float discount;
    printf("Enter Product Name: ");
    scanf("%s", productName);
    printf("Enter Discount Percentage (0-100): ");
    scanf("%f", &discount);

    if (discount < 0 || discount > 100) {
        printf("Invalid discount percentage!\n");
        return;
    }

    int found = 0;
    for (int i = 0; i < productCount; i++) {
        if (strcmp(products[i].name, productName) == 0) {
            found = 1;
            products[i].discount = discount;
            printf("Discount applied successfully!\n");
            saveDataToFile();
            break;
        }
    }
    if (!found) {
        printf("Product not found.\n");
    }
}

void rateProduct() {
    char productName[50];
    float rating;
    printf("Enter Product Name: ");
    scanf("%s", productName);
    printf("Enter Rating (1-5): ");
    scanf("%f", &rating);

    if (rating < 1 || rating > 5) {
        printf("Invalid rating! Please enter a value between 1 and 5.\n");
        return;
    }

    int found = 0;
    for (int i = 0; i < productCount; i++) {
        if (strcmp(products[i].name, productName) == 0) {
            found = 1;
            // Simple average rating calculation
            if (products[i].rating == 0) {
                products[i].rating = rating;
            } else {
                products[i].rating = (products[i].rating + rating) / 2;
            }
            printf("Rating submitted successfully!\n");
            saveDataToFile();
            break;
        }
    }
    if (!found) {
        printf("Product not found.\n");
    }
}

void submitFeedback() {
    char comment[200];
    printf("Enter your feedback (max 200 characters): ");
    getchar(); // Clear buffer
    fgets(comment, 200, stdin);
    comment[strcspn(comment, "\n")] = 0; // Remove newline
    strcpy(feedbacks[feedbackCount].userEmail, currentUser.email);
    strcpy(feedbacks[feedbackCount].comment, comment);
    feedbackCount++;
    printf("Thank you for your feedback!\n");
    saveDataToFile();
}

void viewAllFeedbacks() {
    printf("\n--- All Feedbacks ---\n");
    for (int i = 0; i < feedbackCount; i++) {
        printf("User: %s\n", feedbacks[i].userEmail);
        printf("Feedback: %s\n", feedbacks[i].comment);
        printf("---------------------------\n");
    }
}

void viewProductCategories() {
    printf("\n--- Product Categories ---\n");
    char categories[100][50];
    int categoryCount = 0;

    for (int i = 0; i < productCount; i++) {
        int found = 0;
        for (int j = 0; j < categoryCount; j++) {
            if (strcmp(products[i].category, categories[j]) == 0) {
                found = 1;
                break;
            }
        }
        if (!found) {
            strcpy(categories[categoryCount], products[i].category);
            categoryCount++;
        }
    }

    for (int i = 0; i < categoryCount; i++) {
        printf("%d. %s\n", i+1, categories[i]);
    }
}

void filterProductsByCategory() {
    char category[50];
    printf("Enter Category Name: ");
    scanf("%s", category);

    printf("\n--- Products in Category: %s ---\n", category);
    int found = 0;
    for (int i = 0; i < productCount; i++) {
        if (strcmp(products[i].category, category) == 0) {
            printf("Name: %s, Price: $%.2f, Quantity: %d, Discount: %.2f%%, Rating: %.1f\n",
                   products[i].name, products[i].price, products[i].quantity,
                   products[i].discount, products[i].rating);
            found = 1;
        }
    }
    if (!found) {
        printf("No products found in this category.\n");
    }
}

void saveDataToFile() {
    FILE *file;

    // Save users
    file = fopen("users.dat", "wb");
    if (file != NULL) {
        fwrite(&userCount, sizeof(int), 1, file);
        fwrite(users, sizeof(struct User), userCount, file);
        fclose(file);
    }

    // Save products
    file = fopen("products.dat", "wb");
    if (file != NULL) {
        fwrite(&productCount, sizeof(int), 1, file);
        fwrite(products, sizeof(struct Product), productCount, file);
        fclose(file);
    }
    // Save orders
    file = fopen("orders.dat", "wb");
    if (file != NULL) {
        fwrite(&orderCount, sizeof(int), 1, file);
        fwrite(orders, sizeof(struct Order), orderCount, file);
        fclose(file);
    }

    // Save feedbacks
    file = fopen("feedbacks.dat", "wb");
    if (file != NULL) {
        fwrite(&feedbackCount, sizeof(int), 1, file);
        fwrite(feedbacks, sizeof(struct Feedback), feedbackCount, file);
        fclose(file);
    }
}

void loadDataFromFile() {
    FILE *file;

    // Load users
    file = fopen("users.dat", "rb");
    if (file != NULL) {
        fread(&userCount, sizeof(int), 1, file);
        fread(users, sizeof(struct User), userCount, file);
        fclose(file);
    } else {
        // Add default admin users if file doesn't exist
        strcpy(users[userCount].name, "Admin");
        strcpy(users[userCount].id, "0000000000000000");
        strcpy(users[userCount].email, "admin@example.com");
        strcpy(users[userCount].password, "Admin123");
        userCount++;

        strcpy(users[userCount].name, "Mohammad Saifullah Mansoor");
        strcpy(users[userCount].id, "241-35-408");
        strcpy(users[userCount].email, "saifullah@example.com");
        strcpy(users[userCount].password, "Saif123");
        userCount++;

        strcpy(users[userCount].name, "Sabina Easmin Meem");
        strcpy(users[userCount].id, "241-35-039");
        strcpy(users[userCount].email, "sabina@example.com");
        strcpy(users[userCount].password, "Sabina123");
        userCount++;

        strcpy(users[userCount].name, "Farzana Nopur");
        strcpy(users[userCount].id, "241-35-142");
        strcpy(users[userCount].email, "farzana@example.com");
        strcpy(users[userCount].password, "Farzana123");
        userCount++;
```

```
}


// Load products
file = fopen("products.dat", "rb");
if (file != NULL) {
    fread(&productCount, sizeof(int), 1, file);
    fread(products, sizeof(struct Product), productCount, file);
    fclose(file);
} else {
    // Add default products if file doesn't exist
    strcpy(products[productCount].name, "Apple");
    products[productCount].price = 1.50;
    products[productCount].quantity = 100;
    strcpy(products[productCount].category, "Fruits");
    products[productCount].discount = 0.0;
    products[productCount].rating = 0.0;
    productCount++;


    strcpy(products[productCount].name, "Milk");
    products[productCount].price = 2.00;
    products[productCount].quantity = 50;
    strcpy(products[productCount].category, "Dairy");
    products[productCount].discount = 0.0;
    products[productCount].rating = 0.0;
    productCount++;
```

```
        strcpy(products[productCount].name, "Bread");
        products[productCount].price = 1.00;
        products[productCount].quantity = 200;
        strcpy(products[productCount].category, "Bakery");
        products[productCount].discount = 0.0;
        products[productCount].rating = 0.0;
        productCount++;
    }

    // Load orders
    file = fopen("orders.dat", "rb");
    if (file != NULL) {
        fread(&orderCount, sizeof(int), 1, file);
        fread(orders, sizeof(struct Order), orderCount, file);
        fclose(file);
    }

    // Load feedbacks
    file = fopen("feedbacks.dat", "rb");
    if (file != NULL) {
        fread(&feedbackCount, sizeof(int), 1, file);
        fread(feedbacks, sizeof(struct Feedback), feedbackCount, file);
        fclose(file);
    }
}
```