1. Account Structure:

We define a struct for Account to hold basic details like account number, name, balance, and a pointer to the next account in the li nked list.

```
typedef struct Account {
   int account_number;
   char name[50];
   float balance;
   struct Account* next;
} Account;
```

2. Transaction Structure and Queue:

We define a struct for Transaction to record account transactions (deposit/withdrawal). The queue structure is used to manage the transaction list.

```
typedef struct Transaction {
   int account_number;
   float amount;
   char type[10]; // Deposit or Withdraw
   struct Transaction* next;
} Transaction;

typedef struct {
   Transaction* front;
   Transaction* rear;
} TransactionQueue;
```

3. Create Account Function:

This function creates a new account and adds it to the linked list of accounts

```
void createAccount(Account** head) {
   Account* newAccount = (Account*) malloc(sizeof(Account));
   printf("Enter Account Number: ");
   scanf("%d", &newAccount->account_number);
   printf("Enter Name: ");
   scanf("%s", newAccount->name);
   newAccount->balance = 0.0;
   newAccount->next = *head;
   *head = newAccount;
}
```

A new Account is created and added to the front of the linked list.

4. Display Account Function:

This function finds and displays the details of a specific account.

```
void displayAccount(Account* head, int account_number) {
   Account* temp = head;
   while(temp != NULL && temp->account_number != account_number) {
      temp = temp->next;
   }
   if(temp != NULL) {
      printf("Account Number: %d\n", temp->account_number);
      printf("Name: %s\n", temp->name);
      printf("Balance: %.2f\n", temp->balance);
   } else {
      printf("Account not found!\n");
   }
}
```

The function traverses the linked list to find the account with the given account number and prints its details.

5. Delete Account Function:

This function deletes an account from the linked list.

The function looks for the account and updates the pointers to remove it from the list, then frees the memory.

```
void deleteAccount(Account** head, int account_number) {
    Account* temp = *head;
    Account* prev = NULL;

    if(temp != NULL && temp->account_number == account_number) {
        *head = temp->next;
        free(temp);
        printf("Account deleted successfully.\n");
        return;
    }
    while(temp != NULL && temp->account_number != account_number) {
        prev = temp;
        temp = temp->next;
    }
    if(temp == NULL) {
        printf("Account not found!\n");
        return;
    }
    prev->next = temp->next;
    free(temp);
    printf("Account deleted successfully.\n");
}
```

6. Transaction Queue Functions:

Functions to add transactions to the queue and display them.

The enqueue function adds a new transaction to the end of the queue.

The displayTransactions function prints all transactions in the queue.

```
void enqueue(TransactionQueue* q, int account_number, float amount, const char* type) {
    Transaction* newTransaction = (Transaction*) malloc(sizeof(Transaction));
    newTransaction->account_number = account_number;
    newTransaction->amount = amount;
    strcpy(newTransaction->type, type);
    newTransaction->next = NULL;

    if(q->rear == NULL) {
        q->front = q->rear = newTransaction;
    } else {
        q->rear->next = newTransaction;
        q->rear = newTransaction;
    }
}

void displayTransactions(TransactionQueue* q) {
    Transaction* temp = q->front;
    while(temp != NULL) {
        printf("Account Number: %d, Amount: %.2f, Type: %s\n", temp->account_number, temp->amount, temp->type);
        temp = temp->next;
    }
}
```

7. Deposit Function:

This function updates the account balance and records the transaction in the queue.

This function deposits money into an account and logs the transaction.

```
void deposit(Account* head, int account_number, float amount, TransactionQueue* queue) {
   Account* temp = head;
   while(temp != NULL && temp->account_number != account_number) {
      temp = temp->next;
   }
   if(temp != NULL) {
      temp->balance += amount;
      printf("Deposit Successful. New Balance: %.2f\n", temp->balance);
      enqueue(queue, account_number, amount, "Deposit");
   } else {
      printf("Account not found!\n");
   }
}
```

8. Withdraw Function:

This function withdraws money from an account if the balance is sufficient and logs the transaction.

This function checks if the account has enough balance, updates it, and records the transaction

```
void withdraw(Account* head, int account_number, float amount, TransactionQueue* queue) {
   Account* temp = head;
   while(temp != NULL && temp->account_number != account_number) {
      temp = temp->next;
   }
   if(temp != NULL) {
      if(temp->balance >= amount) {
         temp->balance -= amount;
         printf("Withdrawal Successful. New Balance: %.2f\n", temp->balance);
         enqueue(queue, account_number, amount, "Withdraw");
      } else {
         printf("Insufficient Balance!\n");
      }
   } else {
      printf("Account not found!\n");
   }
}
```

9. Main Function:

Handles user input to interact with the system.

The main function provides a menu for the user to create, display, delete accounts, and handle deposits, withdrawals, and transaction display. It uses a switch
case to handle user inputs and call the relevant functions.

```
int main() {
   Account* head = NULL;
   TransactionQueue queue = {NULL, NULL};

   int choice, account_number;
   float amount;

   while(1) {
      printf("\n1. Create Account\n2. Display Account\n3. Delete Account\n4. Deposit\n5. Withdraw\n6. Display Transactions\n7. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch(choice) {
         case 1:
            createAccount(&head);
```

```
            break;
        case 2:
            printf("Enter Account Number: ");
            scanf("%d", &account_number);
            displayAccount(head, account_number);
            break;
        case 3:
            printf("Enter Account Number: ");
            scanf("%d", &account_number);
            deleteAccount(&head, account_number);
            break;
        case 4:
            printf("Enter Account Number: ");
            scanf("%d", &account_number);
            printf("Enter Amount: ");
            scanf("%f", &amount);
            deposit(head, account_number, amount, &queue);
            break;
        case 5:
            printf("Enter Account Number: ");
            scanf("%d", &account_number);
            printf("Enter Amount: ");
            scanf("%f", &amount);
            withdraw(head, account_number, amount, &queue);
            break;
        case 6:
            displayTransactions(&queue);
            break;
        case 7:
            exit(0);
        default:
            printf("Invalid choice!\n");
    }
  }

  return 0;
}
```