Work-flow of “The Cloud resource billing and Management System”

- **Starting the Program**: When you run the program, it displays a menu of options.

- **Adding a New Customer**: You choose to add a new customer. The program asks for the customer's name, user ID, and how many minutes they used cloud resources. It then calculates the bill and saves this information in a file.

- **Viewing All Customers**: If you want to see all the customers, you choose the view option, and the program shows you a list of all customers and their details.

- **Updating a Customer’s Usage**: If a customer’s usage changes, you can update their record. The program will recalculate their bill and save the updated record to a new file.

- **Checking a Customer’s Bill**: If you want to know how much a particular customer owes, you can search for their ID, and the program will show you their total bill.

- **Finding a Specific Customer**: You can search for a customer by their ID and view their details.

- **Removing a customer**: If a customer is no longer needed, you can delete their record, and the program will update the list accordingly.

Detailed Workflow

1. **Initialization**:
    - **Global Variables**:
        - `customer_data customers[MAX_CUSTOMERS]`: An array to store up to 100 customer records.
        - `static int g_customer_count = 0`: A counter to keep track of the number of customer records currently stored.

2. **Displaying Menu Choices**:
    - **Function**: `static inline void display_choices()`
    - **Purpose**: Shows the options to the user (Add, View, Modify, etc.).
    - **Technical Detail**: Uses `printf` to print a menu to the console, where the user can choose an action by entering a number.

3. **Adding a Record**:
    - **Function**: `static void add_record()`
    - **Purpose**: Collects customer details, calculates the total bill, and saves the record to a file.
    - **Steps**:
        - Opens the file `cloudcustomerdata.txt` in append mode.
        - Checks if there's space for new records (`g_customer_count < MAX_CUSTOMERS`).
        - Reads user input (name, user ID, usage) and calculates the total bill based on fixed rates.
        - Appends the new record to the file and updates the global record list.
        - Closes the file.

4. **Viewing Records**:
    - **Function**: `static void view_records()`
    - **Purpose**: Displays all customer records currently stored in the program.
    - **Steps**:
        - Prints out the details of each customer record from the global array using `printf`.

5. **Modifying a Record**:
    - **Function**: `static void modify_record(uchar1_t user_id[])`
    - **Purpose**: Updates the usage for a specific customer and saves the updated records to a new file.
    - **Steps**:
        - Opens a new file `modifiedcloudcustomerdata.txt`.
        - Searches for the record with the specified `user_id`.
        - Updates the usage and recalculates the total bill.
        - Writes the updated record to the new file.
        - Closes the file.

6. **Viewing Payment**:
    - **Function**: `static void view_payment(uchar1_t user_id[])`
    - **Purpose**: Displays the total bill for a specific customer.
    - **Steps**:
        - Searches for the record with the specified `user_id`.
        - Prints the total bill for the customer if found; otherwise, displays a not-found message.

7. **Searching for a Record**:
    - **Function**: `static void search_record(uchar1_t user_id[])`
    - **Purpose**: Finds and displays details of a specific customer record.
    - **Steps**:
        - Searches for the record with the specified `user_id`.
        - Prints the record's details if found; otherwise, shows a not-found message.

8. **Deleting a Record**:
    - **Function**: `static void delete_record(uchar1_t user_id[])`
    - **Purpose**: Removes a specific customer record from the global list.
    - **Steps**:
        - Searches for the record with the specified `user_id`.
        - Removes the record and shifts subsequent records up to fill the gap.
        - Updates the global customer count.