Algorithm for searching expenses:

1. Start the `searchExpenses` method.
2. Read the length of the expenses list and store it in a variable `leng`.
3. Print a prompt asking the user to enter the expense they want to search.
4. Read the expense value entered by the user and store it in a variable `expense`.
5. Initialize a boolean variable `found` as `false`.
6. Iterate through the expenses list using a loop with index `i` from 0 to `leng-1`.
   - If `arrayList.get(i)` is equal to `expense`, then do the following:
     - Print "Expense found at index `i`".
     - Set `found` to `true`.
     - Break the loop.
7. If `found` is still `false` after the loop, then do the following:
   - Print "Expense not found".
8. End the `searchExpenses` method.

Algorithm for sorting expenses:

1. Start with the `arrayList` containing the expenses to be sorted.
2. Get the size of the `arrayList` and store it in `arrLength`.
3. Use a nested loop structure:
   - Outer loop: Iterate from `i = 0` to `arrLength - 1`.

- Inner loop: Iterate from `j = 0` to `arrLength - i - 1`.

4. Inside the inner loop, compare `arrayList[j]` with `arrayList[j + 1]`.

5. If `arrayList[j]` is greater than `arrayList[j + 1]`, swap the elements at positions `j` and `j + 1` in `arrayList`.

6. Repeat steps 4 and 5 until the inner loop completes for each value of `j`.

7. Repeat steps 3-6 until the outer loop completes for each value of `i`.

8. After the sorting is complete, print the sorted `arrayList` to display the expenses in ascending order.

GIT Link: https://github.com/sathwikraju/Full-Stack---The-Desk-Application-.git