```
({
    loadContacts : function (component) {
        var pageNumber = component.get("v.pageNumber");
        var pageSize   = component.get("v.pageSize");

        var countAction = component.get("c.getTotalContacts");
        countAction.setCallback(this, function(res){
            var total = res.getReturnValue();
            component.set("v.totalRecords", total);
            var totalPages = Math.ceil(total / pageSize);
            component.set("v.totalPages", totalPages);
            var pages = [];
            for (let i = 1; i <= totalPages; i++) {
                pages.push(i);
            }
            component.set("v.pageList", pages);
        });
        $A.enqueueAction(countAction);

        var dataAction = component.get("c.getContacts");
        dataAction.setParams({ pageNumber : pageNumber, pageSize : pageSize });
        dataAction.setCallback(this, function(res){
            component.set("v.contacts", res.getReturnValue());
        });
        $A.enqueueAction(dataAction);
    }
})
```

Overview of Helper File

The helper file contains reusable functions that perform the actual logic of querying server-side data and preparing pagination-related information for the component.

Step 1: Get Total Record Count

First, it calls the Apex method getTotalContacts to determine the total number of Contact records available. This value is stored in v.totalRecords.

Step 2: Calculate Total Pages

Using the total number of records and the page size, it calculates how many pages are required. This is done with Math.ceil(total / pageSize) to round up.

Step 3: Build Page List

It builds an array of page numbers from 1 to totalPages so that the component can render clickable page buttons.

Step 4: Get Contacts for Current Page

Finally, it calls the Apex method getContacts with the pageNumber and pageSize to retrieve the correct subset of contact records. These records are assigned to v.contacts so that the datatable displays the correct data.