* Project Folder Creation
* Server application
  + npm install -g json-server
    - C:\Users\{your\_user\_name}\AppData\Roaming\npm
  + Update to package.json
    - "start": "json-server ./db2.json --port 4000"
  + Chrome JSON Formatter Plugin
    - <https://chrome.google.com/webstore/detail/json-formatter/bcjindcccaagfpapjjmafapmmgkkhgoa?hl=en>
* JSON Server Capabilities
  + Usage of postman
* Client Application - Create
  + npx create-react-app client --template typescript
  + npm start
  + Cleanup activities
    - Src folder
      * Kept App.tsx and index.tsx
      * Removed the references (to logo, css)
  + Created the expense-tracker component
    - Referenced this component from App component
* Installation of dependencies
  + npm install react-bootstrap bootstrap axios
* TypeScript - Demo
  + Installation
    - Option 1
      * npm install -g typescript
      * tsc hello.ts
      * node hello.js
    - Option 2
      * npm install ts-node
      * ts-node hello.ts
  + Declare Type , Interfaces
    - Passing them to functions
* Client App - Fetch Data
  + Creation of models and services folder
  + Models
    - Create type IExpenseItem
  + Services
    - Create getAllExpenseItems
      * Dependent on axios
      * Promise Usage - await (async)
      * Usage of IExpenseItem[]
  + ExpenseTracker
    - Usage of useEffect
    - Calling the service method
    - Fetching the data
* Displaying Expense Items - Success Scenario
  + Create a separate component
    - Expense-items - To manage the display expense items
    - Passed the IExpenseModel[] from the parent component
  + ExpenseTracker
    - STate management for expenseItems
  + CSS usage in App.tsx
* Display Expense Items - Error scenario
  + useState for adding for “error”
  + Made use try-catch block
    - Setting the error object in catch block
  + Update the rendering code in expense-tracker
    - Make use of Alert
    - Made use of error object for conditional display purposes
* “Loading Message” -
  + Usage of useState (loading)
  + Update the render method
    - Rendered the Spinner component
  + Simulating the delay
    - "start-delay": "json-server --watch ./db2.json --port 4001 --delay 4000"
* Display of Expense Items - within Table
  + Usage of Table component (from react-bootstrap)
  + Usage of serial number
    - By considering index (that is passed as a parameter)
* Expense By Payee - Component
  + Creation of a new component
    - ExpenseByPayees
  + Usage of table to render the content
  + Usage of the following utility methods
    - getAllPayeeNames
    - getTotalExpenseByPayee
* Expense Addition (UI) - Part I
  + Create a new separate “ExpenseCreator”
    - Addition of a new button and support for Dialog
  + Usage of “ExpenseCreator” within “ExpenseTracker” component
* Expense Addition (UI) - Part II
  + Addition of Form within the Modal Dialog
  + Cleanup activities
    - Change the controlID
  + For the payeeName
    - Representation through Dropdown/Select component
* Expense Addition (UI) - Part III
  + Dynamically populating the payee-names in the Dropdown/Select component
  + Refactoring
    - Moving the getAllPayeeNames to expense-utils module
  + ExpenseCreator
    - Brought reference to expenseItems (IExpenseItem[]) from the parent (expense-tracker)
* Handling event for ‘Add Expense’
  + Addition of form-level event
  + Usage of useRef for all three properties (expenseDescription, payeename, price)
  + handleClose() at the end
* Type Additions/Corrections
  + Creation of a new item ‘IExpenseCreateItem’
    - Supply the POST Http Call
  + Usage of “as string” while retrieving data from ref objects
* Support for POST HTTP method
  + Usage of axios.put to call the POST method to create the item
  + Usage of the postMethod within the handleAddExpense method
  + Verify if “id” with being returned from POST along with value
* Refresh parent from child
  + Create new method refreshParent (in expense-tracker)
    - To change the state by calling setExpenseItems() with the new expense item
  + Child component to call the method refreshParent signaling to refresh the parent