Assignment 2

Exercise 1. Deploy Transformation Logic to a Server

A REST-based API was created in order to enable clients to submit requests for expression simplification. JSON requests are sent from the client to the server, and JSON responses are sent back from the server to the client. In the design of the Node.js server built in this assignment, techniques such as encapsulation and data flow analysis were studied in which control flow and data flow were analyzed.

Encapsulation: A module contains all of the resources needed to function. How to find these encapsulated functions? One option is to use data flow analysis.

DFA: Identify the highest abstraction points for data both as input and output, and decompose information flow into three modules; input, transform, and output.

Exercise 2. Apply Structural Design Patterns

Facade Pattern

We have applied this methodology to help abstract our expression simplification algorithm. The Boolean Expression Simplifier application contains logic that is quite difficult to understand/walk through for someone who may not be experienced on the topic of boolean algebra. Applying the Facade pattern makes the complex sub system easier to use. A simple interface was constructed for a set of commonly used methods in the algorithm for simplifying the input expression.The Facade interface contains the method simplifyExpression(), which simplify the input boolean expression and list the steps involved in the simplification, respectively. This Facade interface calls another class which invokes a chain of other methods required to perform the requested action. However, the user does not see the complexity of method calls since they are in the children classes of the Facade, which are not visible to the user.

Remote Proxy

The Remote Proxy structural pattern was used in the design of this application in order to facilitate access to information. The remote proxy provides a local representative for an object that resides in a different address space. This is particularly useful in this application because there may be cases in which the information being processed does not change very often. When a user inputs a boolean expression in order to send for the simplification process, the application must send the input string to the server and perform a simplification algorithm on it. However, if a particular expression has been already simplified in the past, the Remote Proxy pattern will retrieve the simplified result from the cache and display it to the front end user. In such a case, the proxy provides a local representative for the object that resides in the server side.

Exercise 3. Apply Additional Design Patterns

Singleton

The Singleton is an important behavioural design pattern that ensures that a class only has one instance and provides access to it. This is important in this application due to the use of the Remote Proxy. When boolean expressions are simplified for the first time, the result is stored in the proxy. Then, when the boolean expression needs to be simplified again, the algorithm searches the proxy to check if the result is stored, and if it is, it displays that result to the user on the front end. A Singleton pattern is important here to ensure that only 1 access point to the proxy object exists so that multiple search algorithms are not performed. There must be only 1 access to the proxy as having multiple access points will only create redundancy and overhead; there is only one source of truth in this case.

Iterator Design Pattern

The Iterator design pattern is used to sequentially access elements of a collection using uniform interface. This is used in this web application during the creation of the history list on the front end because the traversal of an array data structure is simplified. The simplified boolean expressions displayed on the client are stored into an array data structure. Then, when a user requests to view the history of simplified expressions, a list view is dynamically generated by iterating through the array of simplified expressions and creating a list on the client UI. For each expression within the array, a new list element is generated. A while loop is used in order to iterate through the collection of expression items, and in each iteration, a new list element, <li> element, is generated and appended to a unordered element, <ul>. The iterator stops the traversal process after it has reached the last element within the array.

Exercise4. Document Design Decisions (submit as A2design.pdf)