Module 3 Activity 5: Data Access

Diya Liza Varghese

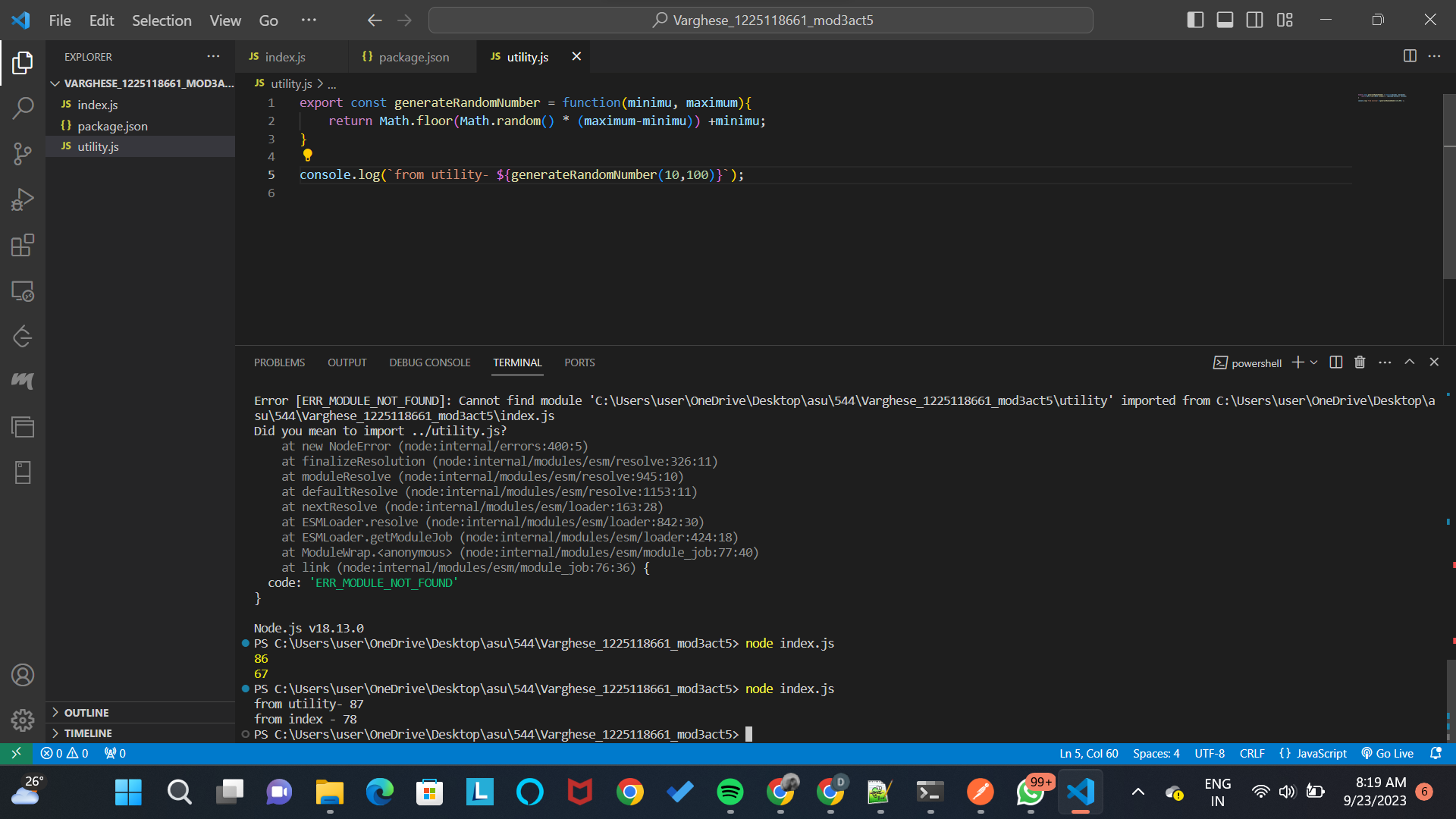
Information Technology, Polytechnic School, Arizona State University

IFT 544: Middleware Prog & Database Sec (2023 Fall)

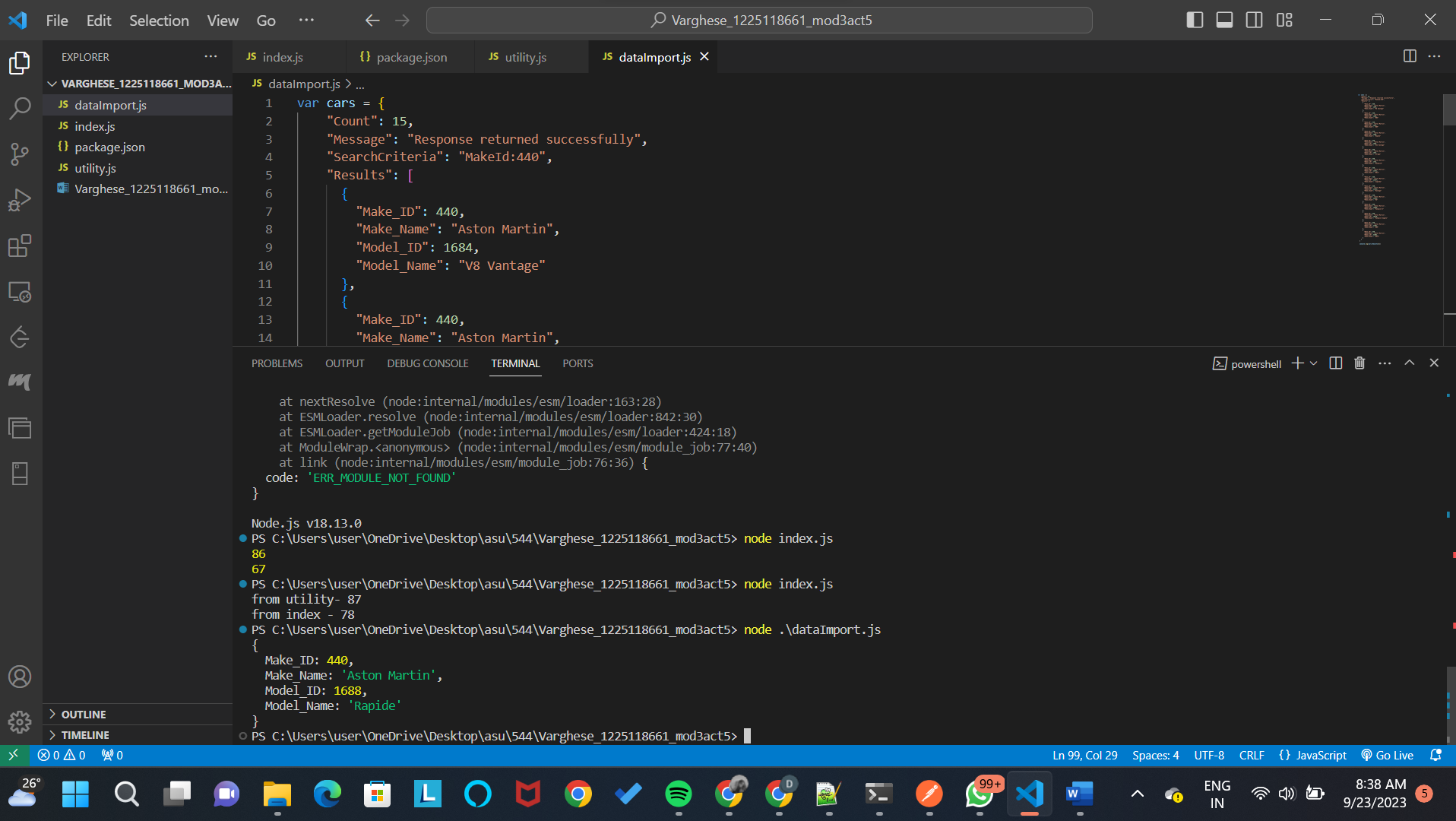
Dr. Dinesh Sthapit

September 24, 2023

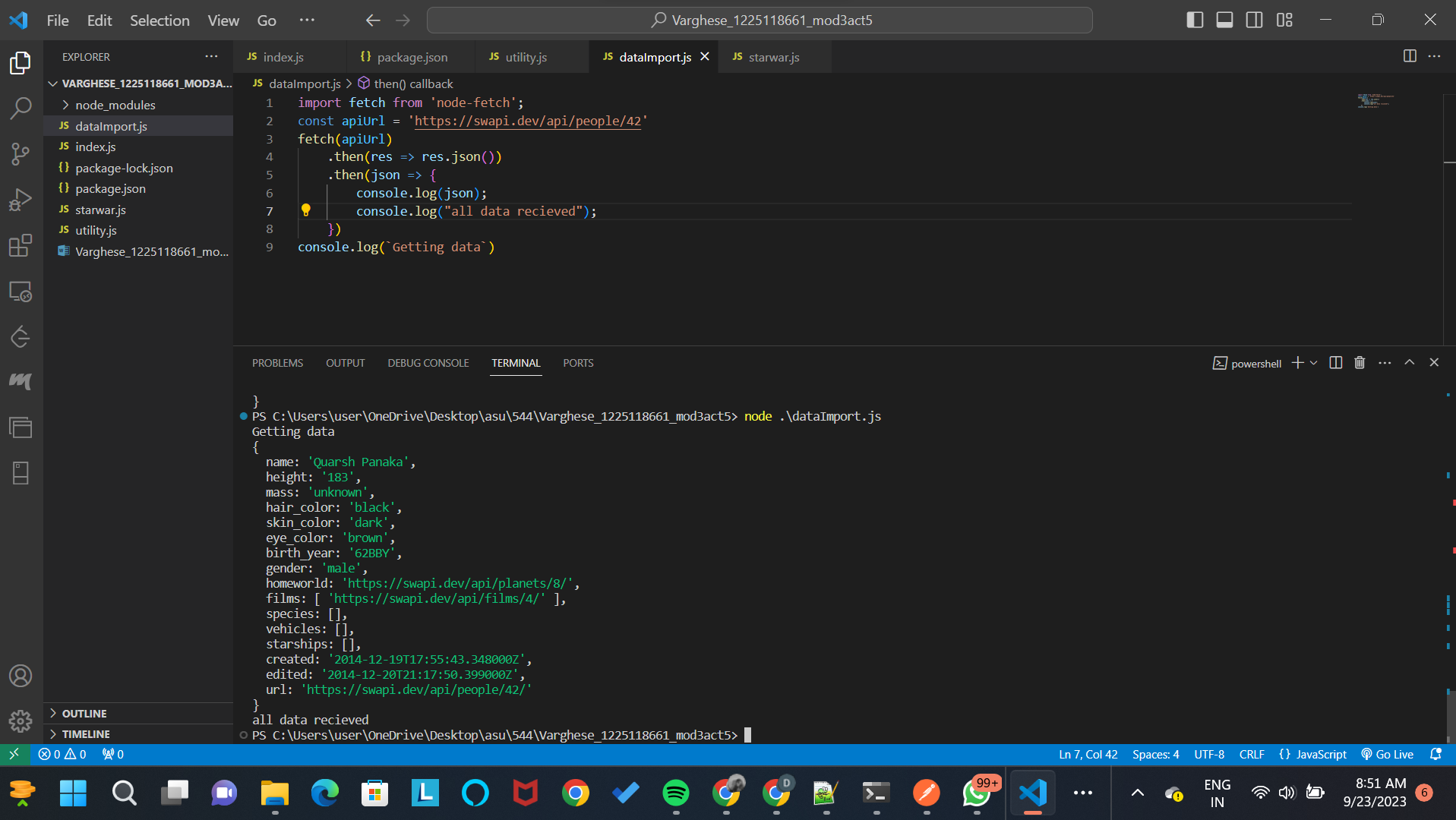
Reusable components to create composite components:-1



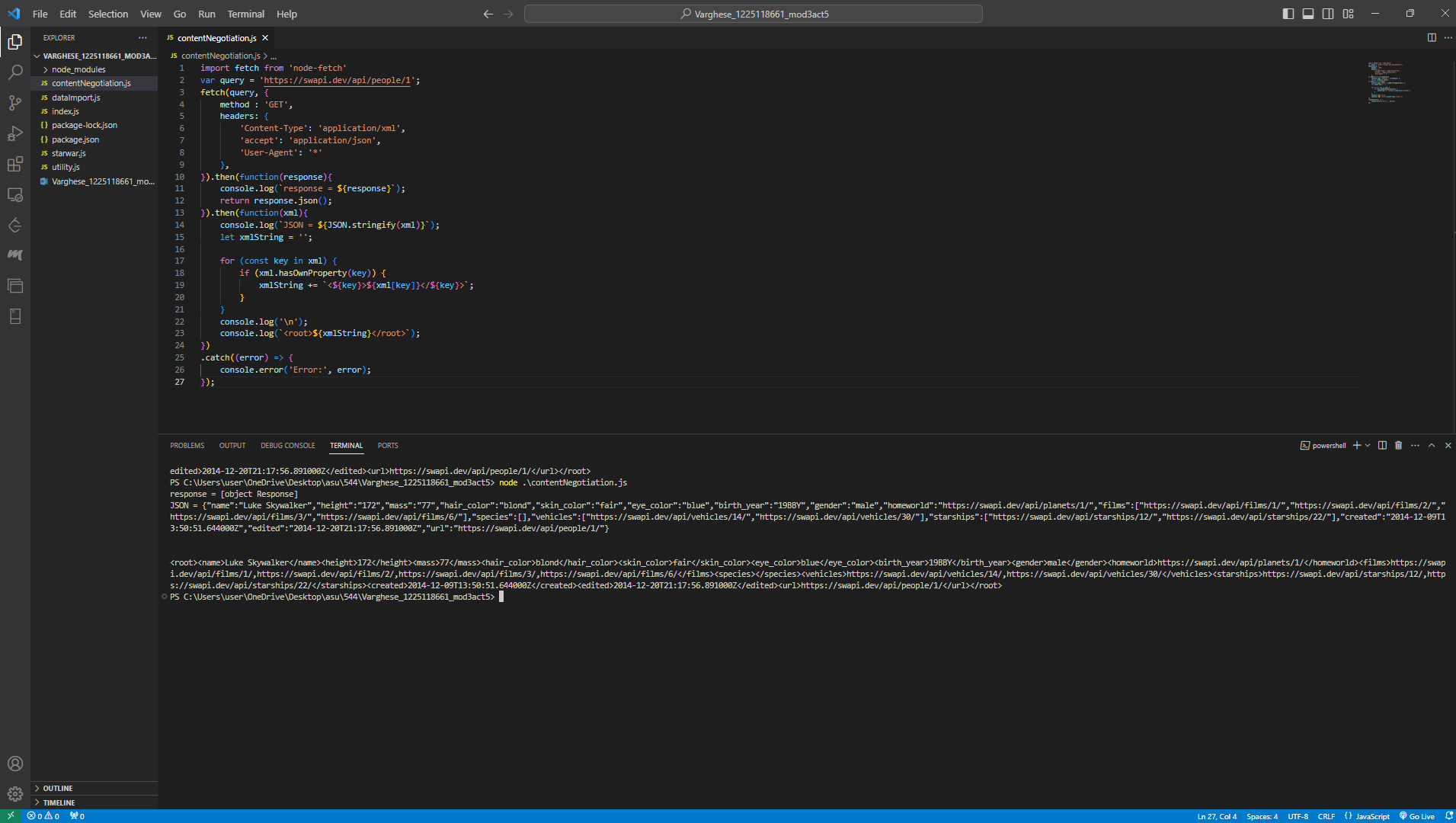
'index.js' and 'utility.js' are two JavaScript files that make up the code. The 'generateRandomNumber' function from 'utility.js' is imported into 'index.js' before being used to create a random number and log a message to the console using the minimum and maximum values supplied in the imported function. The 'generateRandomNumber' function, which accepts a minimum and maximum number and generates a random integer within that range, is defined in the 'utility.js' file. Additionally, the script prints a message to the console displaying the random number produced by the utility module. In essence, the code shows how to create and display random integers in two different files using JavaScript's import/export feature and modularization.

Using Importing External Modules (third-party modules).

The code given defines the JavaScript object 'cars', which has data about several automobile types. It contains information like the total number of car models, a message stating whether the response was successful, search parameters, and an array of car model objects under the "Results" field. The fourth automobile model object (index 3) from the "Results" array is logged to the console using the 'console.log(cars.Results[3])' line. In this instance, it receives information for the fourth vehicle model, an Aston Martin called "Rapide" with the "Model\_ID" of 1688 and the "Model\_Name" of "Rapide."

Data Formats and Data Exchange

The provided code makes use of the 'node-fetch' module to send an HTTP GET request to the Star Wars API (SWAPI) in order to get details about a particular character, identifiable by the distinct ID 42. The 'fetch' function from the 'node-fetch' library is first imported into the script, and the 'apiUrl' variable is set as the endpoint URL for the SWAPI character request. It then starts the HTTP GET request to the designated 'apiUrl'. The API answer is then handled by promise-based '.then()' methods in the code. When the response is received, it decodes the response body as JSON and writes the gleaned information to the console. Additionally, it sends out a message to confirm that all data was successfully received. Notably, the code also logs an alert message to the console that says, "Getting data," before the data is retrieved. The asynchronous nature of the retrieve request is highlighted by the sequence in which the code is executed. This causes the "all data received" message to be reported once the data is made accessible.

Content Negotiation

Using the 'node-fetch' package, the provided code demonstrates an HTTP GET call to the URL 'https://swapi.dev/api/people/1'. 'Content-Type' is set to 'application/xml' and 'accept' is set to 'application/json,' indicating that the code expects to receive JSON data but sets the 'Content-Type' header for XML in the request headers. The response from the API is processed in a promise chain. The response object is initially logged to the console. The response body is then attempted to be converted to JSON using the'response.json()' function. However, since the 'Content-Type' header suggests XML even though the API is supposed to provide JSON, this translation may fail.

To get around this, the code logs the response's JSON representation before manually creating an XML string by iterating through the properties of the JSON data. A root element encloses the XML string, creating valid XML.

In conclusion, the code illustrates a situation in which the 'Content-Type' (XML) given in the request headers differs with the expected return format (JSON). While processing the JSON answer, it also shows how to manually convert JSON to XML; however, if the API actually returns XML data, this conversion might not be appropriate. It also has error handling so that any mistakes made while performing the fetch operation can be recorded.