1. Abstract

This application was constructed using **JavaScript (Node.js)**, a widely used, dynamically typed scripting language suited for web development and general-purpose scripting. The program performs simple sentiment analysis by assigning a score to every word in a review and then generating an overall star rating for the review text. The completed application accepts input via the command line and outputs a running sentiment score and final star rating.

2. Approach

The implementation applied several core JavaScript and Node.js techniques. For data storage, the application used a **plain JavaScript object as a dictionary** to map words to sentiment scores from the socialsent.csv file. This structure provides efficient lookup performance during review analysis.

The program follows a modular design with three primary functions:

- buildSocialSentimentTable(filename) reads and parses the sentiment file line-by-line using the Node.js fs module, constructing a lookup table of word–score pairs.
- getSocialSentimentScore(filename) reads the review file and splits its content into words, using regular expressions to clean up punctuation. It then looks up each word in the sentiment table and accumulates the total score, printing each word’s score and the running total in the format [word: score, accumulated_score].
- getStarRating(score) uses basic conditional logic to map the final sentiment score to a rating between 1 and 5 stars.

The program uses Node.js's process.argv to receive the review file name via the command line. If no file is specified, it defaults to review.txt. File input/output is handled through the built-in fs module. Console output allows users to view the word-by-word scoring progression live.

3. What I Learned About JavaScript

a. Unique/Special Features

JavaScript’s dynamic typing and flexible data structures (like plain objects and arrays) make it especially easy to manipulate and transform data on the fly. One notable aspect is JavaScript's lack of enforced types, which offers rapid prototyping but requires careful runtime validation. Additionally, JavaScript’s regex capabilities for string parsing are powerful for tasks like tokenizing sentences. Using fs.readFileSync() and process.argv[] enables seamless command-line interaction.

b. Features I Liked

The concise and lightweight syntax in JavaScript makes it ideal for scripting-style applications. The use of objects for dictionary-like access, the lack of boilerplate setup code (e.g., no Main() or class declarations required), and the ability to handle file input and output with only a few lines of code all contributed to fast development. I also appreciated JavaScript's compatibility with VS Code and the excellent ecosystem of extensions for linting, auto-formatting, and debugging.

c. Features I Found Challenging

JavaScript’s asynchronous nature (though not used here) can be tricky when working with file I/O if not using the synchronous versions like readFileSync. Additionally, the lack of built-in integer or float types (everything is just Number) can lead to unexpected behavior in edge cases, especially with floating-point precision. Unlike languages like C# or Java, JavaScript also lacks native support for strong typing or built-in dictionary/Map structures with full functionality in plain syntax, requiring developers to be more vigilant.

4. Notes on Tools Used

This assignment was developed locally using **Visual Studio Code** on **Windows**, and executed using **Node.js (LTS version)**. To test and evaluate the application, I used the following commands in the terminal:

```
node socialSentimentAnalyzer.js bad.txt
node socialSentimentAnalyzer.js good.txt
node socialSentimentAnalyzer.js
```

(uses default review.txt)

All input and output files (socialsent.csv, review files) were placed in the same directory as the script for ease of testing. The lightweight, file-based setup made the application easy to debug and extend.