**Group Name:** Javaverse Team

**Group Members:**   
Jann Dominguez

Efrain Rivero

Manuel Albisu-Bouza

**Project Title:** Implementing an Autocomplete System Using a Trie Data Structure in Java

**Project Overview:**

The goal of this project is to design and implement an efficient autocomplete system similar to those used in search engines or text editing software. This system will suggest possible word completions based on the user's input. The primary data structure used will be a Trie, which offers efficient storage and retrieval of words and prefixes, making it ideal for an autocomplete system.

**Project Components:**

**TrieNode Class:** This class represents each node in the Trie. Each TrieNode will contain an array of child nodes for each letter of the English alphabet and a boolean flag indicating the end of a word.

**Trie Class:** This class provides the primary operations of the Trie. It includes methods for inserting words into the Trie and retrieving word completions based on a given prefix.

**AutocompleteSystem Class:** This class is responsible for user interaction. It accepts user input, generates possible word completions using the Trie, and displays these suggestions to the user.

**File Handling:** If an external dictionary file is used to populate the Trie, Java's FileReader and BufferedReader classes will be used for file reading.

**Error Handling:** Possible errors, such as invalid user input or file reading issues, will be handled using Java's exception handling mechanisms.

**Performance Testing:** The system's performance will be evaluated by measuring the time taken to generate autocomplete suggestions. This will be done using Java's System.nanoTime() function or similar methods.

**Project Plan:**

**Phase 1:** Design and implement the TrieNode and Trie classes. Test these components by inserting some sample words into the Trie and retrieving them based on prefixes.

**Phase 2:** Design and implement the AutocompleteSystem class. This includes the user interface for accepting input and displaying output. Test this component with the Trie from Phase 1.

**Phase 3:** If using an external dictionary, implement file reading to populate the Trie with words from this dictionary. Handle any possible file reading errors.

**Phase 4:** Perform comprehensive testing of the entire system. This includes testing with different dictionary sizes and different user inputs. Also measure and evaluate the system's performance.

**Team Members and Contributions:**

**Efrain Rivero:** Responsible for the TrieNode and Trie classes, including testing these components.

**Jann Dominguez:** Responsible for the AutocompleteSystem class, including testing this component.

**Manuel Albisu-Bouza:** Responsible for file handling and performance testing.

**Expected Outcome:**

At the end of this project, we expect to have a fully functional autocomplete system that can suggest possible word completions based on user input. The system should demonstrate the efficient storage and retrieval of words and prefixes using a Trie data structure.