**Business Requirements Document (BRD)**

**Project Title : Text Editor with Dictionary.**

**Project Overview**

This project is a C++-based text editor that supports file handling, dictionary integration, and an indexed mini-database for efficiently managing words.

The editor will have features such as:

* CRUD operations on dictionary words (Add, Search, Update, Delete).
* Index file storage for fast lookups and word retrieval.
* Spell-checking functionality to detect and suggest corrections.
* Synonym replacement system to enhance vocabulary.
* Basic text editor operations (Open, Edit, Save text files

**Problem Statement**

Writers, students, and professionals often struggle with spelling mistakes, repetitive words, and slow word lookup while working with text documents. Traditional text editors lack built-in word management, spell-check, and synonym replacement features. Additionally, manually searching a large dictionary file is slw, making real-time corrections impractical.

**Solution**

*This C++-based text editor will:*

* Provide a dictionary-powered text editing experience with real-time spell-check.
* Use an index file for fast lookups (instead of linear searching).
* Allow adding, updating, and deleting words in the dictionary.
* Suggest synonyms to improve writing quality.
* Save text files with changes and allow reloading/editing.

**Key Features & Functionalities**

**Core Features**

* Text File Handling – Open, edit, and save text files.
* Dictionary CRUD Operations – Manage words efficiently with indexing.
* Indexed Storage for Fast Lookups – Use a separate index file to speed up word retrieval.
* Spell Check – Detect and suggest correct words from the dictionary.
* Synonym Replacement – Find synonyms and suggest alternative words.

**Advanced Features**

* Auto-Save Feature – Periodically save work automatically.
* Multi-Language Support – Expand dictionary for multiple languages.
* Word Frequency Analysis – Identify commonly used words and suggest replacements.
* Dictionary Import/Export – Allow importing external dictionaries in CSV format.

**Technical Requirements**

Programming Language: C++

File Handling: fstream (for reading/writing text & dictionary files)

Data Structures:

unordered\_map for fast word storage

vector for synonym lists

map for indexed word lookups

Libraries:

<fstream> – File operations

<unordered\_map> – Dictionary storage

<sstream> – Parsing text files

<algorithm> – Searching & sorting

**Functional Requirements**

* Handle large text files (up to 100MB).
* Quickly search words using an index file (O(1) lookup).
* Support at least 50,000 words in the dictionary.
* Ensure fast spell-checking within a document (<1 second per 1,000 words).

**Non-Functional Requirements**

* Performance: Must open and process large files efficiently.
* Scalability: Should support adding external dictionary files.
* Security: Prevent corruption of dictionary and index files.

**Data Storage & Indexing Strategy**

Dictionary File (dictionary.txt)

Stores words and their synonyms in a structured format:

word:synonym1,synonym2,synonym3

example:sample,instance,illustration

beautiful:attractive,gorgeous,stunning

Index File (index.txt)

Stores words with byte offsets to enable fast lookup:

apple:0

banana:22

example:48

This allows direct file access instead of scanning the entire dictionary.

**Expected Benefits**

* Faster word lookup using index-based storage.
* Real-time spell checking improves writing efficiency.
* Synonym replacement enhances vocabulary.
* Efficient CRUD operations allow easy dictionary expansion.

**Future Enhancements**

* GUI Implementation – Convert CLI editor into a full-featured GUI-based text editor.
* Machine Learning for Auto-Corrections – Suggest better word replacements.
* Grammar Checking Feature – Extend beyond spell check.
* Would you like an extended implementation guide or a specific feature breakdown?