💻 VocabCLI Project Presentation

Group 18

Atharva Shah (60)

Anay Deshpande (18)

Summary

🔹 VocabularyCLI is a lightweight Command Line Interface that allows users to look up word definitions, examples, synonyms and antonyms directly via the command line. Powered with several

utility based commands our CLI offers rapid and robust Knowledge Base capabilities like Flashcards, Tagging, Word Management, Graph Reporting, Bulk import and export of word lists and is a definitive software for linguaphiles.

🔹 This application boasts a simple and intuitive interface that is easy to use and is a must have for anyone who wants to expand their vocabulary and improve their language skills. The app also offers advanced Text Classification and Processing via the use of Natural Language Processing and Machine Learning algorithms which will be discussed in detail in the "Scope and Features" section.

🔹 The CLI will be offered with eye-catching Panels, Tables, Animated Symbols, Emojis, Interactive Menus, Spinners, Colored fonts and other rich features that will make the user experience more enjoyable and interactive. The CLI will also be offered with a comprehensive User Manual and a detailed Documentation that will help users get started with the CLI and use it to its full potential.

Objectives & Need

Objectives:

- To provide a lightweight Command Line Interface for users to look up word definitions, examples, synonyms and antonyms directly via the command line.
- To offer rapid and robust Knowledge Base capabilities such as Flashcards, Tagging, Word Management, Graph Reporting, Bulk import and export of word lists.
- To provide advanced Text Classification and Processing via the use of Natural Language Processing and Machine Learning algorithms.
- To provide a simple and intuitive interface that is easy to use and is a must-have for anyone who wants to expand their vocabulary and improve their language skills.
- To provide a comprehensive and reliable vocabulary tool that can be used by students, educators, and language enthusiasts.
- To ensure the application is optimized for performance and can handle a large volume of data and user requests.

Needs:

- The need for a lightweight Command Line Interface that allows users to easily look up word definitions, examples, synonyms and antonyms directly via the command line.
- The need for rapid and robust Knowledge Base capabilities such as Flashcards, Tagging, Word Management, Graph Reporting, Bulk import and export of word lists.
- The need for advanced Text Classification and Processing via the use of Natural Language Processing and Machine Learning algorithms.

- The need for a user-friendly and intuitive interface that is easy to use and helps users expand their vocabulary and improve their language skills.
- The need for accurate and up-to-date definitions, synonyms, and antonyms for a wide range of words and phrases.
- The need for a tool that can help users learn and retain new vocabulary effectively through features such as flashcards and word management.

What Purpose does this Project fulfill?

The VocabCLI (Vocabulary Builder Command Line Interface) fulfills the purpose of providing a convenient and efficient way for users to look up word definitions, synonyms, antonyms, examples, and other information related to words and language. It provides a lightweight and user-friendly command line interface that can be accessed easily by anyone who wants to expand their vocabulary and improve their language skills.

In addition to its core functionality of word lookup and definition, VocabCLI also provides a range of features to help users manage their vocabulary, such as Flashcards, Tagging, Word Management, Graph Reporting, Bulk import and export of word lists. These features allow users to organize and track their progress in learning new words and phrases.

Moreover, VocabCLI incorporates advanced Natural Language Processing and Machine Learning algorithms to provide text classification and processing capabilities, enabling users to analyze and understand complex language structures and patterns. This can be particularly useful for language learners and educators who want to analyze and understand texts in greater detail.

VocabCLI is designed to be a versatile and user-friendly command line interface that allows users to easily access a wealth of information about words and language. It can be used by a wide range of users, from language learners and educators to writers and journalists who need to look up words and phrases quickly and efficiently and enhances their language skills and improve their understanding of language.

Some of the use cases for VocabCLI include:

- **Language learning:** VocabCLI can be used by language learners who want to expand their vocabulary and improve their language skills. The Flashcards feature allows users to quiz themselves on new words and phrases, while the Word Management feature allows them to track their progress in learning new words and phrases.
- **Writing and journalism:** Writers and journalists can use VocabCLI to quickly look up words and phrases and check their definitions and usage in context. This can help them write more effectively and accurately.

- **Research:** Researchers and academics can use VocabCLI to analyze language patterns and structures in texts, as well as to look up definitions and synonyms for specific words or phrases.
- **Teaching:** Educators can use VocabCLI as a tool for teaching language and vocabulary to their students. The Graph Reporting feature can help them track their students' progress and identify areas where they need to focus more attention.

Overall, VocabCLI fulfills the purpose of providing a convenient and efficient way for users to access a wide range of information about words and language, while also providing advanced features such as Natural Language Processing and Machine Learning to help users analyze and understand language in greater detail.

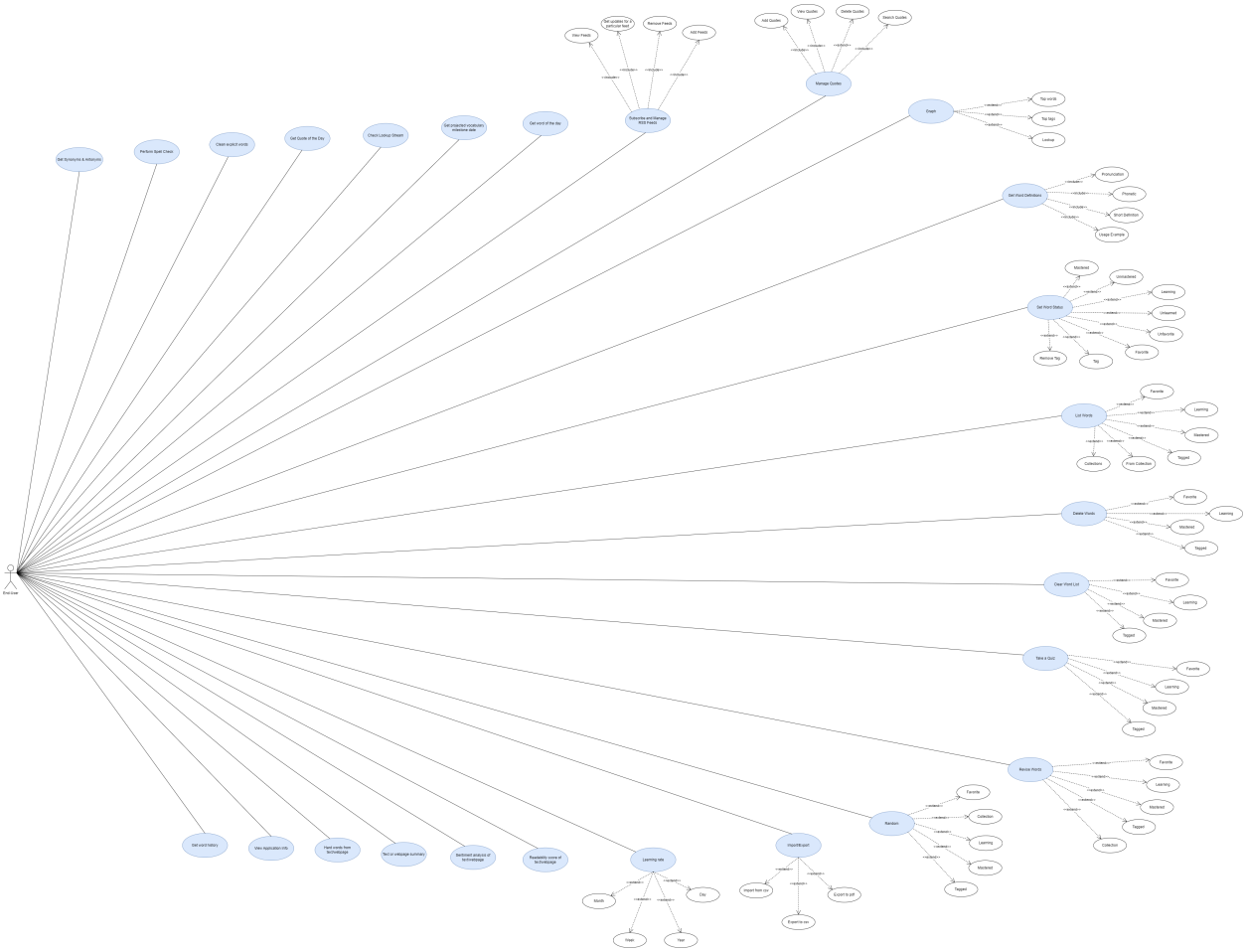
Features 🎯

- Look up word definitions, examples, homonyms, synonyms and antonyms
- Generate flashcards for words
- Tag words, set words as favorites, set words as learning, set words as mastered
- Generate Graph Reports
- Import and Export word lists in PDF and CSV formats
- Show word lookup history and word lookup statistics
- Perform Text Classification and Processing and Summarize Web Articles
- Delete words from the database and clear attributes associated with the words
- Revise words and ready-made word collections
- Quiz Mode
- TTS (Text to Speech) Mode and Accessibility Mode
- Paraphrase Text and Generate Word Clouds
- Detect Plagiarism and Generate Similarity Scores
- Determine Readability Index and Filter out offensive words
- Generate Word Frequency Distributions
- Save User Quotes and Provide Quotes of the Day
- Determine Word Sentiment and Generate Sentiment Analysis Reports
- And many other

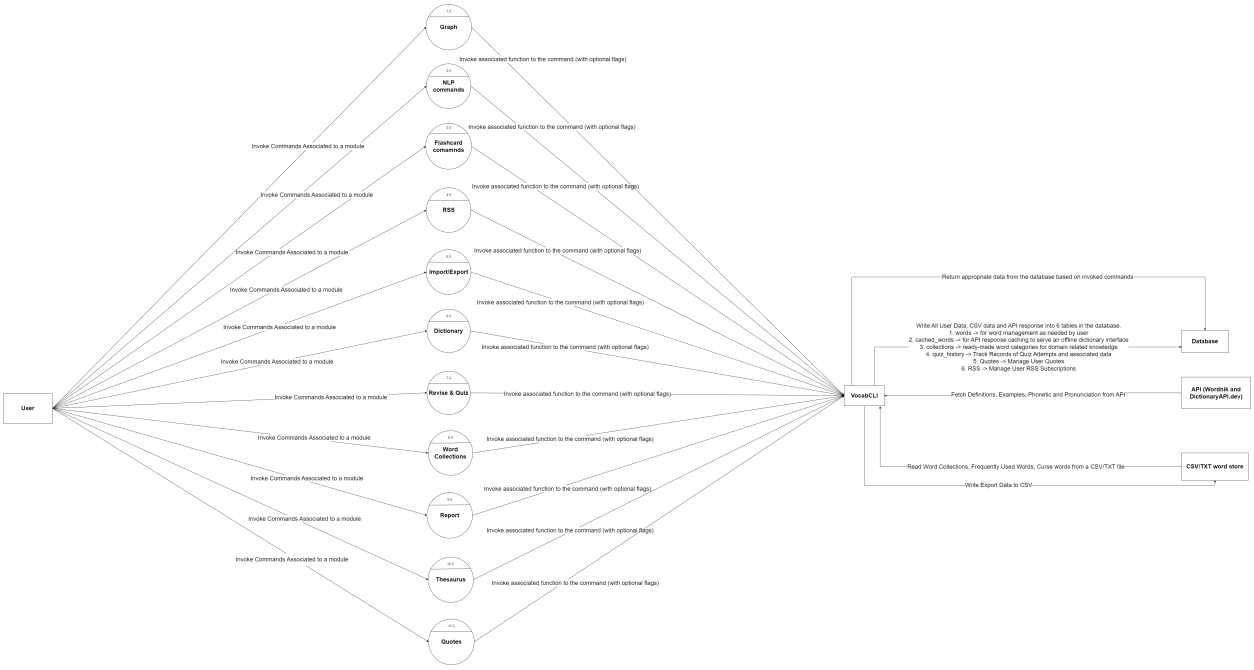
Technical Details

- **Primary Development Language**: Python 3.10
- **Database Management System**: SQLite3
- **API**: DictionaryAPI (https://api.dictionaryapi.dev/api/v2/entries/en/hello)
- **Deployment**: PyPi (Python Package Index)
- **Libraries**: Rich, Typer, Matplotlib, Seaborn, NLTK, SpaCy, Pandas, Numpy, Requests, Beautiful Soup, PyDictionary
- **Testing**: PyTest (Unit Testing), PyTest-Cov (Code Coverage), PyTest-Benchmark (Benchmarking)
- **Documentation**: Sphinx, ReadTheDocs

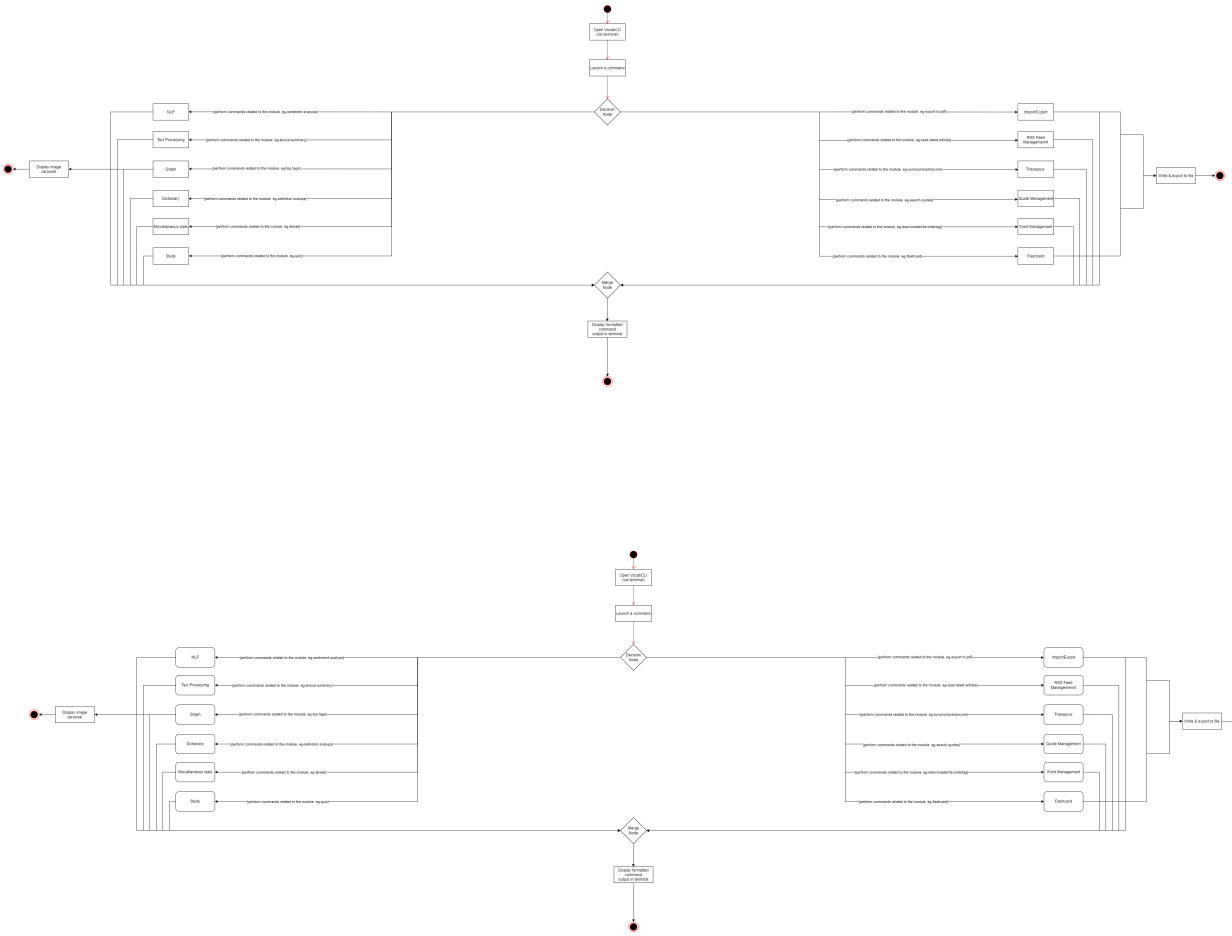
UML Diagrams



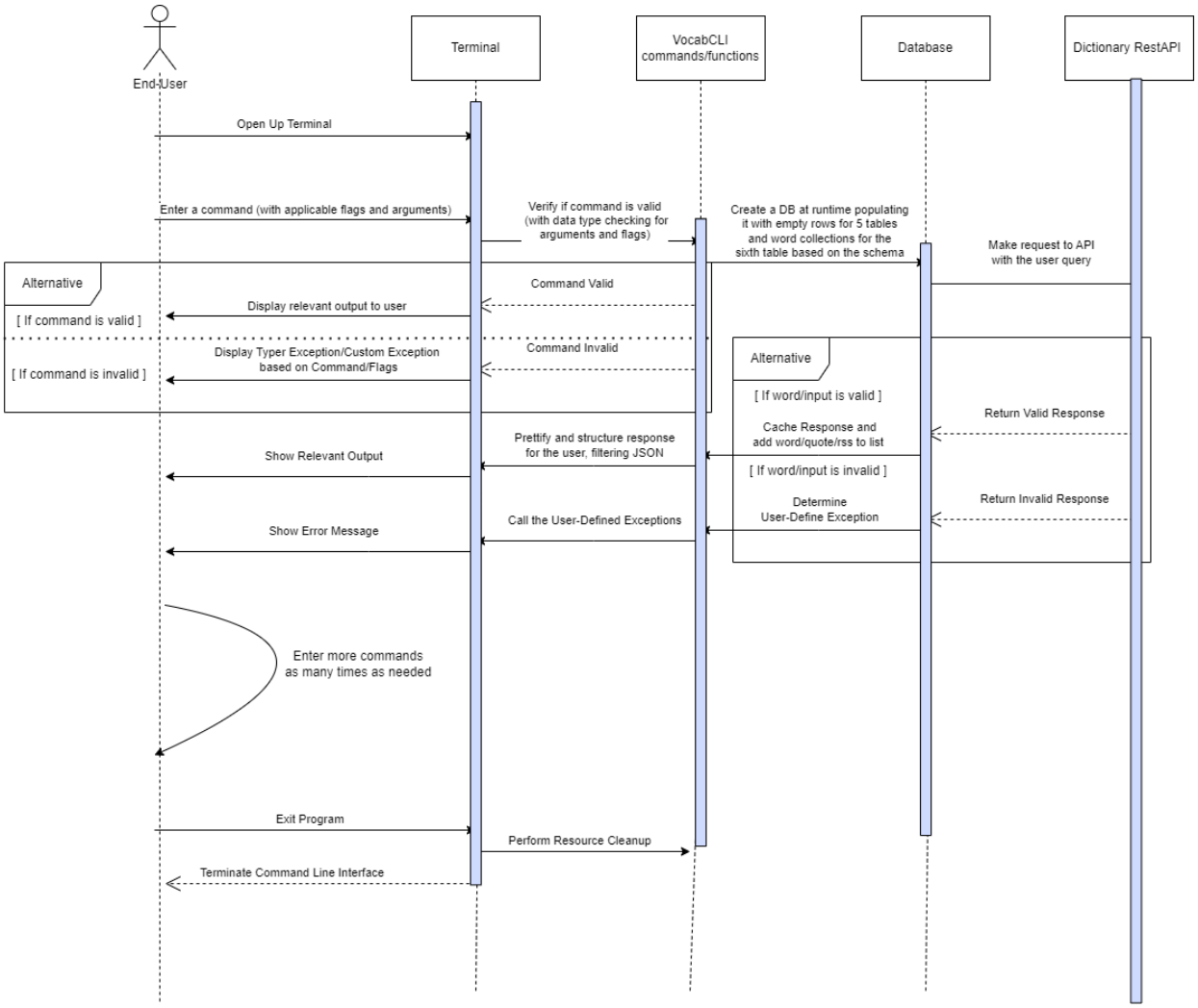
Use Case Diagram



Data Flow Diagram



Activity Diagram



Sequence Diagram

AGILE USER STORY

USER STORY ID	PRIORITY	AS A	I WANT TO	SO THAT I CAN	STATUS
1	Low	Administrator	Add new words to Collection	Maintain relevance of word store	Completed
2	Medium	Administrator	Collect usage statistics	See downloads, activity and analyze it to improve software quality	Completed
3	High	User	Use dictionary module	Get definition of words with all the details	Completed
4	Medium	User	Use list command	Get a list of words and filter with various criterias	Completed
5	Low	User	Use graph module	User statistics in graph format	Completed
6	Low	User	Use flashcards	Get quick learning bytes to improve my Vocabulary	Completed
7	Medium	User	Generate report	Download pdf/ graph report of my app	Completed
8	High	User	Use thesaurus	Get synonyms & antonyms of a word	Completed
9	Medium	User	Use quotes	Add quotes and list them	Completed
10	High	User	Use RSS	Subscribe to RSS feeds, get a list of them	Completed
11	High	User	Use NLP	Summarize articles, extract hardwords, check readability	Completed

USER STORY ID	PRIORITY	AS A	I WANT TO	SO THAT I CAN	STATUS
12	Medium	User	Study	Revise words and play a quiz	Completed

TASK AND CONTRIBUTION MATRIX

Atharva Shah	Anay Deshpande
Configuring Pytest for automated testing and constructing a robust testing architecture.	Utilizing type hinting and specifying optional arguments to enhance code readability and maintainability.
Installing Sphinx and MkDocs for the purpose of generating clear, concise and structured documentation that can be exported to HTML and published on the web.	Establishing a connection between the Driver CLI Program and backend function calls.
Developing a simplistic website designed to promote CLI and increase audience growth.	Generating coverage reports to ensure that all commands and functions are tested.
Writing all code related to NLP functions for the backend.	Enhancing code quality through refactoring and identifying bugs by generating issues/tickets.
Crafting all backend code and visualization for Graphs integrated with GUI slider.	Developing and executing all Word Management functions.
Designing and creating 50% of test cases.	Enhancing the output of the terminal using Rich library and Typer libraries.
Implementing all import and export features in addition to customizable flashcard image generation.	Creating docstrings for each function to improve code documentation and clarity.
Incorporating animation, progress bars and spinners for asynchronous tasks.	Designing and implementing test cases for 50% of the codebase to ensure code reliability and functionality.
Implementing a Caching System to enable offline usage for word retrieval.	Implementing error handling and exception handling to ensure the application does not crash or behave unexpectedly when errors occur.
Collaborating with Dictionary API to obtain definition, usage, and examples from numerous endpoints.	Writing integration tests to ensure that all components of the application work together seamlessly.
Implementing machine learning algorithms to enable advanced text analysis and classification capabilities.	Designing and implementing a user interface for the application that is intuitive, user-friendly, and visually appealing.
Implementing a feature to allow users to track their progress over time	Developing and implementing a feature to allow users to create and manage their own word lists.
Developing and integrating a feature to allow users to compete with each other in vocabulary-building challenges, such as quizzes	Working on the application website design and development collaboratively.

PROJECT MEETING LOG

Meeting Date	Atharva Shah	Anay Deshpande	Topic of Discussion
2022-11-20	✅	✅	Project Planning. Discussed Technology to be used. Generated a rough roadmap for the next 3 months
2022-11-27	✅	✅	Deciding Libraries, Databases and Codebase Architecture
2022-12-04	✅	✅	Working on Driver Program for the CLI
2022-12-11	✅	✅	Working on the Vocabulary Builder Module
2022-12-18	✅	✅	Working on Word Management Module
2022-12-25	✅	✅	Working on Graph Module
2023-01-01	✅	✅	Working on NLP Module
2023-01-08	✅	✅	Working on Miscellaneous Module
2023-01-15	✅	✅	Adding Caching System
2023-01-22	✅	✅	Setting up Testing and Documentation. Adding Animations, State Loaders and Progress Bars.
2023-01-29	✅	✅	Beautifying output and using Terminal Themed Styling using Rich
2023-02-05	✅	✅	UML Diagrams Brainstorming
2023-02-12	✅	✅	PyPi Package Development
2023-02-19	✅	✅	Testing and Documentation Review.
2023-02-26	✅	✅	Generating Project Report

TESTING METHODOLOGY

We have implemented a comprehensive automated testing strategy for the VocabularyCLI application using PyTest, PyTest-Cov and PyTest-Benchmark modules.

- **PyTest** has been used for unit testing the classes and functions of the application, which helps to ensure that the code is functioning as intended and meets the desired requirements.
- **PyTest-Cov** has been used to generate code coverage reports, which provide insights into how much of the code has been tested and helps to identify any areas of the application that may need additional testing.
- **PyTest**-Benchmark has been used for benchmarking, which allows you to measure the performance of the application and identify any areas where optimizations may be required.

Our automated testing approach using PyTest, PyTest-Cov and PyTest-Benchmark has helped to ensure that the VocabularyCLI application is functioning as intended, meets the desired requirements, and performs optimally.

The first step in any automated testing approach is to identify all the possible test cases and write test cases to cover all aspects of the application. We have followed this approach by writing tests for all possible test cases to ensure that there are no bugs in the VocabularyCLI application. This is a good practice as it helps to ensure that the application is functioning as intended and meets the desired requirements.

PyTest is a flexible and powerful testing framework for Python that supports various types of testing such as functional testing, integration testing, and unit testing. It provides a rich set of features that make it easy to write comprehensive test cases, including the use of fixtures, parametrization, and assertions.

In addition to functional testing, we have also done load testing with a database of more than 4000 entries to check the performance of the application. Load testing involves testing the application under various load conditions to ensure that it can handle the expected number of users and requests without any performance degradation. This is an important step in ensuring that the application can handle the expected load without any performance issues.

PyTest-Benchmark has been used to measure the performance of the application and identify any areas where optimizations may be required. By using PyTest-Benchmark, we have measured the response times of the VocabularyCLI application under various load conditions and identify any areas where the performance needs to be optimized.

Finally, we have manually tested certain functionality of the application like Graph and NLP. Manual testing involves testing the application manually to ensure that it is functioning as intended and meets the desired requirements. This is an important step in ensuring that the application is user-friendly and meets the needs of the end-users.

Overall, the combination of PyTest and manual testing has helped to ensure that the VocabularyCLI application is functioning as intended, meets the desired requirements, and is user-friendly.

We have implemented a comprehensive automated testing strategy for the VocabularyCLI application using PyTest, PyTest-Cov and PyTest-Benchmark modules.

PyTest has been used for unit testing the classes and functions of the application, which helps to ensure that the code is functioning as intended and meets the desired requirements.

PyTest-Cov has been used to generate code coverage reports, which provide insights into how much of the code has been tested and helps to identify any areas of the application that may need additional testing.

PyTest-Benchmark has been used for benchmarking, which allows you to measure the performance of the application and identify any areas where optimizations may be required.

Our automated testing approach using PyTest, PyTest-Cov and PyTest-Benchmark has helped to ensure that the VocabularyCLI application is functioning as intended, meets the desired requirements, and performs optimally.

The first step in any automated testing approach is to identify all the possible test cases and write test cases to cover all aspects of the application. We have followed this approach by writing tests for all possible test cases to ensure that there are no bugs in the VocabularyCLI application. This is a good practice as it helps to ensure that the application is functioning as intended and meets the desired requirements.

PyTest is a flexible and powerful testing framework for Python that supports various types of testing such as functional testing, integration testing, and unit testing. It provides a rich set of features that make it easy to write comprehensive test cases, including the use of fixtures, parametrization, and assertions.

In addition to functional testing, we have also done load testing with a database of more than 4000 entries to check the performance of the application. Load testing involves testing the application under various load conditions to ensure that it can handle the expected number of users and requests without any performance degradation. This is an important step in ensuring that the application can handle the expected load without any performance issues.

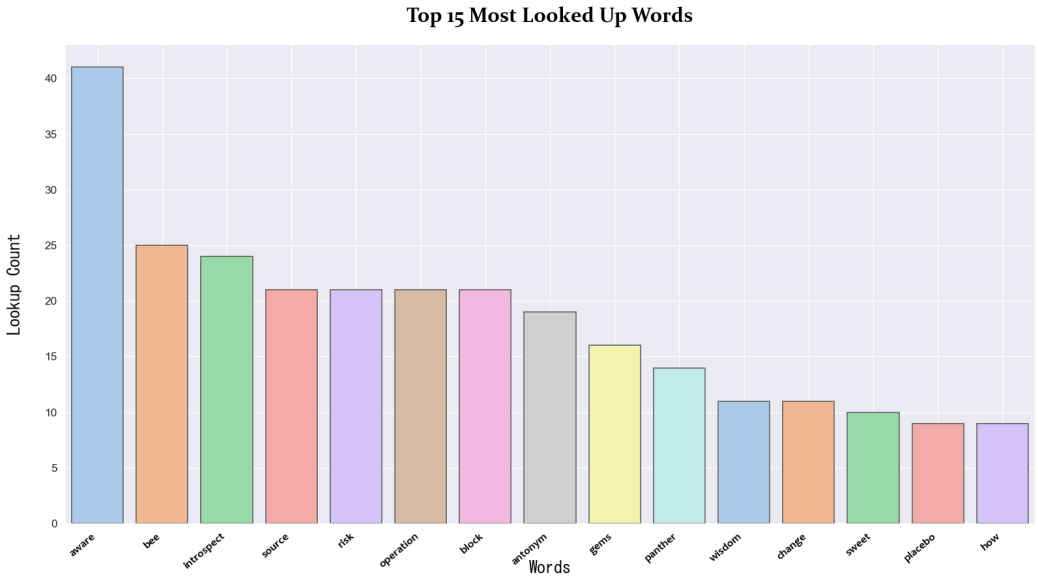
PyTest-Benchmark has been used to measure the performance of the application and identify any areas where optimizations may be required. By using PyTest-Benchmark, we have measured the response times of the VocabularyCLI application under various load conditions and identify any areas where the performance needs to be optimized.

Finally, we have manually tested certain functionality of the application like Graph and NLP. Manual testing involves testing the application manually to ensure that it is functioning as intended and meets the desired requirements. This is an important step in ensuring that the application is user-friendly and meets the needs of the end-users.

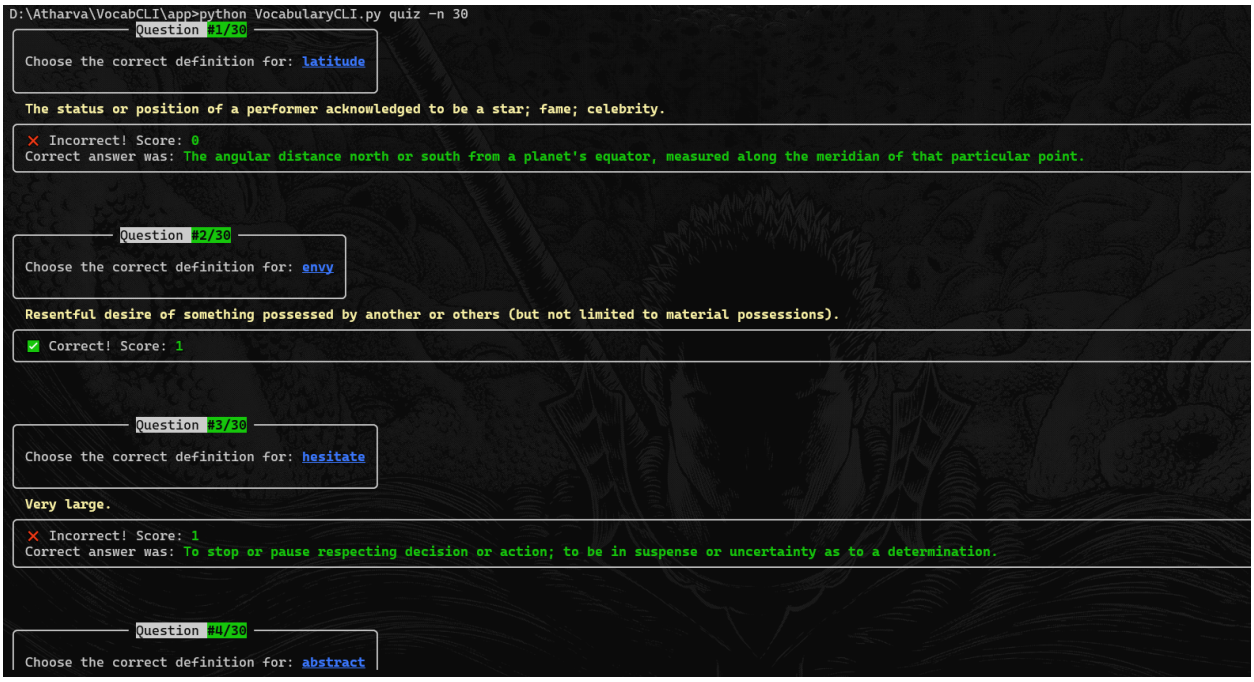
Overall, the combination of PyTest and manual testing has helped to ensure that the VocabularyCLI application is functioning as intended, meets the desired requirements, and is user-friendly.

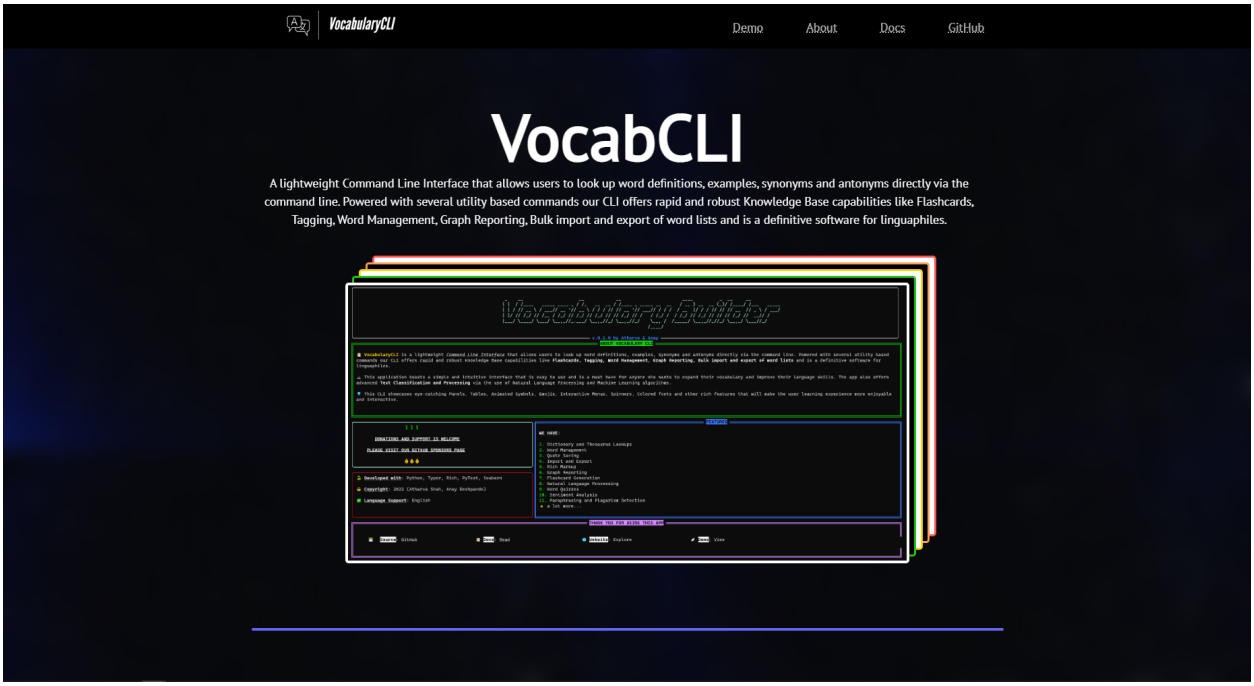
Screenshots

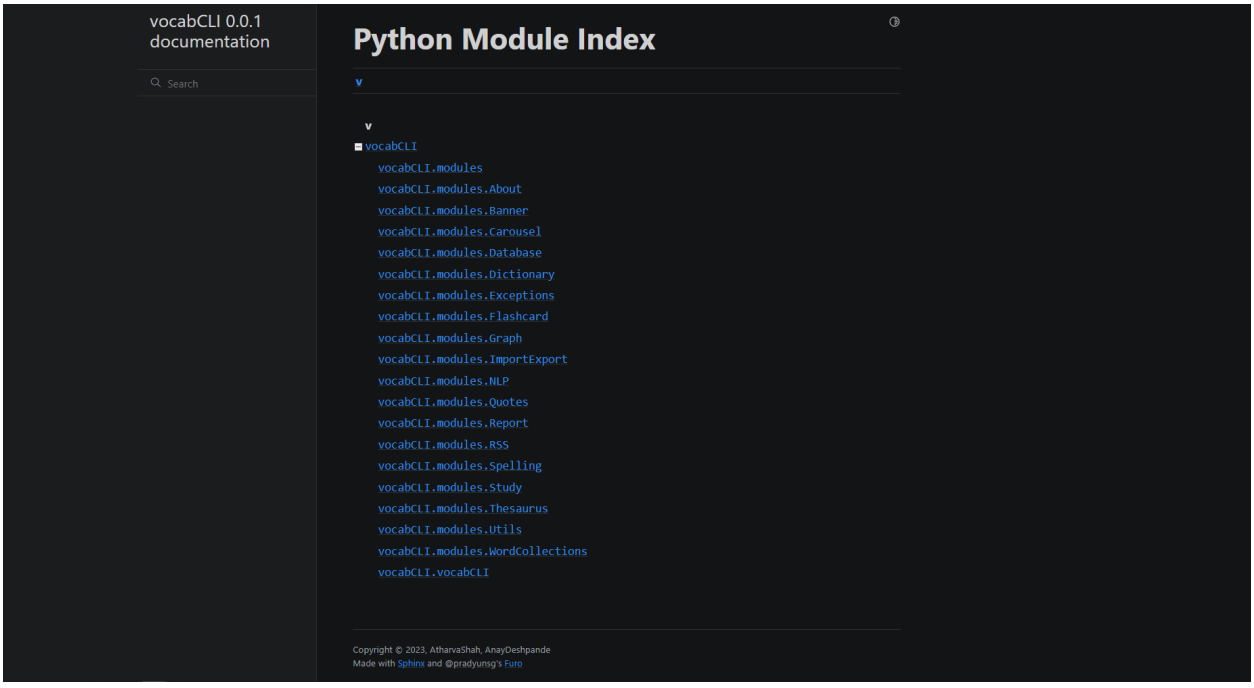
```
D:\Atharva\VocabCLI\app>python VocabularyCLI.py qotd
                Quote of the Day - 30 Dec '22

 Quote: "If I work as hard as I can, I wonder how much I can do in a day?"

 Author: Ezra Taft Benson


D:\Atharva\VocabCLI\app>python VocabularyCLI.py rate
                  Today's Learning Rate

 🚀 You have looked up 100% MORE words today compared to yesterday.
 Today: 1 words.
 Yesterday: 0 words.


D:\Atharva\VocabCLI\app>python VocabularyCLI.py streak
                      Streak

 🔥 Your longest word lookup streak is 7 days.
 Start Date: 10 December 2022
 End Date: 16 December 2022
```

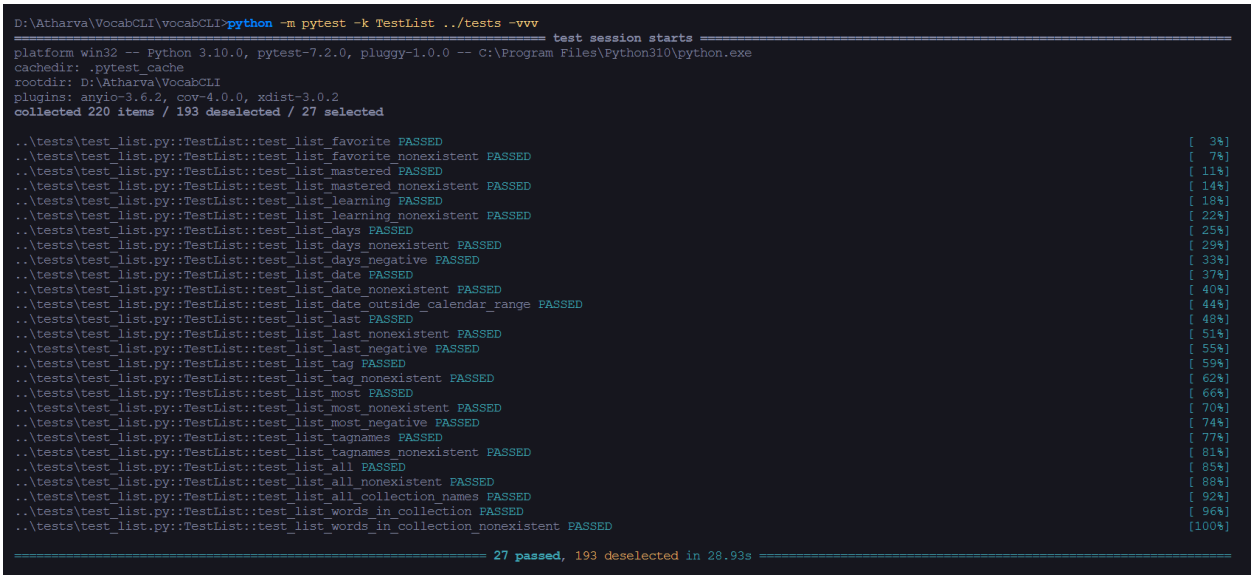



D:\Atharva\VocabCLI\app>python VocabularyCLI.py list -f
Favorite [8 word(s)]
bee introspect wisdom change aware butterfly python agave
D:\Atharva\VocabCLI\app>python VocabularyCLI.py list -m
Mastered [11 word(s)]
aware butterfly how envy agave pipe ternary risk operation block source
D:\Atharva\VocabCLI\app>python VocabularyCLI.py list -t diamonds
Words with tag diamonds [3 word(s)]
python trill dub
D:\Atharva\VocabCLI\app>python VocabularyCLI.py list -l
Learning [12 word(s)]
antique introspect wisdom effect gems antonym sweet the sound high mellow fruit


D:\Atharva\VocabCLI\app>python VocabularyCLI.py quiz -n 30
Question #1/30
Choose the correct definition for: latitude
The status or position of a performer acknowledged to be a star; fame; celebrity.
Incorrect! Score: 0
Correct answer was: The angular distance north or south from a planet's equator, measured along the meridian of that particular point.
Question #2/30
Choose the correct definition for: envy
Resentful desire of something possessed by another or others (but not limited to material possessions).
Correct! Score: 1
Question #3/30
Choose the correct definition for: hesitate
Very large.
Incorrect! Score: 1
Correct answer was: To stop or pause respecting decision or action; to be in suspense or uncertainty as to a determination.
Question #4/30
Choose the correct definition for: abstract


VocabularyCLI
Demo
About
Docs
GitHub
VocabCLI
A lightweight Command Line Interface that allows users to look up word definitions, examples, synonyms and antonyms directly via the command line. Powered with several utility based commands our CLI offers rapid and robust Knowledge Base capabilities like Flashcards, Tagging, Word Management, Graph Reporting, Bulk import and export of word lists and is a definitive software for linguaphiles.


vocabCLI 0.0.1 documentation
Search
Python Module Index
v
v
vocabCLI
vocabCLI.modules
vocabCLI.modules.About
vocabCLI.modules.Banner
vocabCLI.modules.Carousel
vocabCLI.modules.Database
vocabCLI.modules.Dictionary
vocabCLI.modules.Exceptions
vocabCLI.modules.Flashcard
vocabCLI.modules.Graph
vocabCLI.modules.ImportExport
vocabCLI.modules.NLP
vocabCLI.modules.Quotes
vocabCLI.modules.Report
vocabCLI.modules.RSS
vocabCLI.modules.Spelling
vocabCLI.modules.Study
vocabCLI.modules.Thesaurus
vocabCLI.modules.Utils
vocabCLI.modules.WordCollections
vocabCLI.vocabCLI
Copyright © 2023, AtharvaShah, AnayDeshpande
Made with Sphinx and @pradyunsg's Furo



References

General Blogs

- Managing Virtual Environment
- Python SQLite

Youtube

- PyTest
- Python OOP
- Python Type Hinting
- Python package creation PYPI

Projects made with Typer

- Typer ToDo CLI
- Typer Project Management CLI
- Please ToDo

Typer and Docs

- Generate CLI Docs with Typer
- Testing Typer CLI

- Using Rich with Typer
- Rich Main Documentation
- Open Source Rich Projects

For Designing Flashcards using Pillow

- Discord Custom Card 1
- Discord Custom Card 1
- Make ID Card
- Make Instagram Card

Rich Tables

- Docs for Tables

Graphs

- Beautiful Graphs
- Python Graph Gallery
- 50 Visualizations

NLP and Advanced Word Processing

Sentiment Analysis (Input Can be Webpage URL, Text File name, Console Text Input)

- https://importsem.com/evaluate-sentiment-analysis-in-bulk-with-spacy-and-python/
- https://medium.com/mlearning-ai/ai-in-the-real-world-2-sentiment-analysis-using-spacy-pipelines-b39a2618d7c1

Extract Difficult Words (Readability Index)

- https://stackoverflow.com/questions/54710198/getting-out-difficult-english-words-from-text-for-vocabulary-building-using-pyth
- https://www.geeksforgeeks.org/readability-index-pythonnlp/

Plagiarism Detector

- https://betterprogramming.pub/build-your-own-plagiarism-checker-with-python-and-machine-learning-7e81877fd970