Documentation for the “Punctuation Analysis Tool” application.

Authors: Everest Alonso, Patrik Homola, Kamil Pecela

Project developed for the course “Development of Information Systems”

Faculty of Mathematics, Physics and Informatics

Comenius University Bratislava

2023-2024

Table of Contents

[1 Introduction 3](#__RefHeading___Toc451_21973997)

[2 Implementation Plan 4](#__RefHeading___Toc383_21973997)

[2.1 Project Setup 4](#__RefHeading___Toc385_21973997)

[2.2 Dependencies 4](#__RefHeading___Toc387_21973997)

[2.3 GUI Design and Layout 4](#__RefHeading___Toc389_21973997)

[2.4 File Handling 4](#__RefHeading___Toc391_21973997)

[2.5 Punctuation Selection 4](#__RefHeading___Toc393_21973997)

[2.6 Logarithmic Scale 4](#__RefHeading___Toc395_21973997)

[2.7 Analysis Process 4](#__RefHeading___Toc397_21973997)

[2.8 Results Display 5](#__RefHeading___Toc399_21973997)

[2.9 "Analyze New Text" Button 5](#__RefHeading___Toc401_21973997)

[2.10 Error Handling 5](#__RefHeading___Toc403_21973997)

[2.11 Documentation 5](#__RefHeading___Toc405_21973997)

[2.12 Testing 5](#__RefHeading___Toc407_21973997)

[2.13 User Interface Polish 5](#__RefHeading___Toc409_21973997)

[2.14 Deployment 5](#__RefHeading___Toc411_21973997)

[2.15 User Manual 5](#__RefHeading___Toc413_21973997)

[3 Architecture and Design Overview 6](#__RefHeading___Toc415_21973997)

[3.1 High-Level Architecture: 6](#__RefHeading___Toc417_21973997)

[3.2 File Handling and Model: 6](#__RefHeading___Toc419_21973997)

[3.3 User Interface Design (View): 6](#__RefHeading___Toc421_21973997)

[3.4 Graphical Representation (View): 6](#__RefHeading___Toc423_21973997)

[3.5 User Interaction (Controller): 6](#__RefHeading___Toc425_21973997)

[3.6 Error Handling: 7](#__RefHeading___Toc427_21973997)

[3.7 Global Variables: 7](#__RefHeading___Toc429_21973997)

[4 Testing scenarios 8](#__RefHeading___Toc431_21973997)

[4.1 File Upload and Encoding 8](#__RefHeading___Toc433_21973997)

[4.2 Punctuation Selection 8](#__RefHeading___Toc435_21973997)

[4.3 Logarithmic Scale 8](#__RefHeading___Toc437_21973997)

[4.4 Analysis Process 8](#__RefHeading___Toc439_21973997)

[4.5 Results Display 9](#__RefHeading___Toc441_21973997)

[4.6 Error Handling 9](#__RefHeading___Toc443_21973997)

[4.7 User Interface 9](#__RefHeading___Toc445_21973997)

[4.8 General Application Flow 9](#__RefHeading___Toc447_21973997)

# Introduction

This document was created within the subject Development of Information Systems in 2023/2024 and serves as a complete design of the application “Punctuation Analysis Tool”. It contains information necessary for the explanation and understanding the functionality as well as the method of system implementation. This document refers to a requirements catalog. The following chapters will present the implementation plan, system architecture description, and test scenarios.

# Implementation Plan

## Project Setup

* Create a new project directory.
* Set up version control.
* Initialize a virtual environment for the project.

## Dependencies

* Install required dependencies (Tkinter, Matplotlib).

## GUI Design and Layout

* Design the initial page with widgets for file selection, author, title, and punctuation mark buttons.
* Implement the "Go" button for initiating the analysis.
* Implement the disclaimer display.

## File Handling

* Implement functionality to handle file uploads.
* Ensure that only \*.txt files with utf-8 encoding are accepted.
* Extract author and title information from the uploaded text.

## Punctuation Selection

* Create buttons for each punctuation mark.
* Allow users to select a specific punctuation mark for analysis.

## Logarithmic Scale

* Implement checkboxes for logarithmic X and Y axes.
* Incorporate these choices in the plot generation.

## Analysis Process

* Implement the analysis process.
* Transform the text into an array of words.
* Count the number of words between occurrences of chosen punctuation marks or any punctuation marks.

## Results Display

* Create a new page for displaying analysis results.
* Show statistics and a bar graph.
* Allow users to save results in image format to a chosen directory.

## "Analyze New Text" Button

* Implement a button on the result page to redirect users to the initial page for a new analysis.

## Error Handling

* Implement error handling for file reading and invalid input.

## Documentation

* Document the code using comments and docstrings.
* Create a README file explaining how to run the application.

## Testing

* Conduct unit testing for individual functions.
* Perform integration testing for the entire application.

## User Interface Polish

* Enhance the user interface for better user experience.
* Ensure consistency in design and responsiveness.

## Deployment

* Package the application for distribution.
* Create an executable file for users to run the application.

## User Manual

* Prepare a user manual explaining how to use the application.

# Architecture and Design Overview

## **High-Level Architecture:**

* The application follows a Model-View-Controller (MVC) architectural pattern.
* Model: Encapsulated in the count\_words\_between\_punctuation function, responsible for text processing and analysis.
* View: Implemented using Tkinter for the graphical user interface (GUI) elements.
* Controller: Orchestrated through the submit\_form function, handling user inputs and triggering the analysis process.

## **File Handling and Model:**

* The count\_words\_between\_punctuation function handles file reading, text processing, and word count analysis.
* It utilizes regular expressions to split the text into segments based on user-defined punctuation marks.
* The result is a dictionary (word\_counts) containing word frequencies based on the distance before the next punctuation mark.

## **User Interface Design (View):**

* The GUI is developed using Tkinter with two main frames: one for file handling and another for text input and analysis configuration.
* Widgets include labels, buttons, entry fields, and checkboxes for user interactions.
* The application features error message pop-ups using Tkinter's messagebox for a user-friendly experience.

## **Graphical Representation (View):**

* Matplotlib is employed for graphical representation.
* The show\_plot function generates a bar graph based on the analysis results, with the option for logarithmic scales on the X and Y axes.

## **User Interaction (Controller):**

* Users interact with the system by uploading a text file, selecting punctuation marks, and configuring logarithmic scales.
* The submit\_form function orchestrates the entire process, including error handling for file selection, punctuation input validation, and triggering the analysis.

## **Error Handling:**

* Robust error handling is implemented throughout the application.
* Users receive informative error messages in case of issues, such as file reading errors or invalid input for punctuation marks.

## **Global Variables:**

* Global variables (file\_path, fig\_canvas, canvas\_frame, log\_x\_axis, and log\_y\_axis) are used to maintain state and share information across functions.

# Testing scenarios

## File Upload and Encoding

* Scenario 1: User uploads a valid \*.txt file with utf-8 encoding.
  + Expected Result: File is accepted, and the system extracts author and title information.
* Scenario 2: User tries to upload a non-\*.txt file.
  + Expected Result: System displays an error message indicating that only \*.txt files are accepted.
* Scenario 3: User uploads a \*.txt file with a different encoding.
  + Expected Result: System displays an error message indicating that utf-8 encoding is required.

## Punctuation Selection

* Scenario 1: User chooses specific punctuation marks for analysis.
  + Expected Result: Analysis is conducted based on the chosen punctuation marks.
* Scenario 2: User doesn't select any punctuation mark.
  + Expected Result: Analysis is conducted for occurrences of any punctuation marks.

## Logarithmic Scale

* Scenario 1: User checks both logarithmic X and Y axes.
  + Expected Result: Graph is displayed with logarithmic scales on both axes.
* Scenario 2: User checks only logarithmic X axis.
  + Expected Result: Graph is displayed with a logarithmic scale on the X axis.

## Analysis Process

* Scenario 1: User initiates analysis with a valid file and punctuation marks.
  + Expected Result: The system successfully analyzes the text and generates statistics and a bar graph.
* Scenario 2: User initiates analysis without selecting a file.
  + Expected Result: System displays an error message prompting the user to select a file.

## Results Display

* Scenario 1: User saves analysis results as an image.
  + Expected Result: Results are saved in the specified directory in image format.
* Scenario 2: User clicks "Analyze New Text" on the results page.
  + Expected Result: User is redirected to the initial page for a new analysis.

## Error Handling

* Scenario 1: User encounters an error during file reading.
  + Expected Result: System displays an error message indicating the issue with file reading.
* Scenario 2: User enters invalid punctuation marks.
  + Expected Result: System displays an error message indicating the invalid input.

## User Interface

* Scenario 1: User interacts with the graphical user interface.
  + Expected Result: All widgets and buttons respond correctly to user interactions.
* Scenario 2: User clicks the "Go" button without selecting a file.
  + Expected Result: System displays an error message prompting the user to select a file.

## General Application Flow

* Scenario 1: User goes through the entire process from file upload to result display.
  + Expected Result: The application successfully conducts the analysis and presents the results.
* Scenario 2: User closes and reopens the application.
  + Expected Result: The application retains its state, and the user can resume or start a new analysis.