Software Requirements Specifications for Social Butterfly

Version <1.3> approved

Prepared by:

Group Name: Code Crafters

Juliana Unger	z23618477	junger2021@fau.edu
Matthew Henao	z23685608	henaom2022@fau.edu
Tarek Kayali		

Instructor: Safak Kayıkcı

Course: Principles Software Engineering

Section: 4010-002

Teaching assistant: Preston Billion Polak

Date: June 23, 2024

Social Butterfly
Revision Sheet

Version	Revision Date	Changes	Author
1.1	6/21	Changed the performance section and added more information	Juliana Unger
1.2	6/22	Looked and changed the potential drawbacks	Juliana Unger
1.3	6/23	Added more information to the security and maintainability sections	Juliana Unger

Table of Contents

Software Requirement Specifications (SRS)

Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the requirements for the Social Butterfly project. Social Butterfly is a Python-based tool designed to collect and analyze data from Twitter (X) and Reddit via their respective APIs. This document aims to provide a comprehensive description of the software's functionality, performance, and interface requirements to guide its development and ensure it meets the needs of its users.

1.2 Scope

The scope of the Social Butterfly project includes the development of a user-friendly tool that allows users to search, collect, and analyze social media posts and content from Twitter (X) and Reddit. The system will enable users to specify search parameters, retrieve relevant data, and generate reports to understand online conversations and community discussions better. The project will use the Tweepy and PRAW libraries for interacting with the Twitter and Reddit APIs.

1.3 Definitions, Acronyms, and Abbreviations

- **API**: Application Programming Interface
- **Twitter (X)**: A social media platform for microblogging
- **Reddit**: A social media platform and forum for community discussions
- **Tweepy**: A Python library for accessing the Twitter API
- **PRAW**: Python Reddit API Wrapper, a Python library for accessing the Reddit API
- **Social Butterfly**: The name of the project/tool being developed
- **IOS:** Mobile operating system developed by Apple.
- **Android:** MObile operating system based on a modified version of the LInux Kernel and other open-source software.

- **CPU:** central processing unit that is a component of a computer that performs the systems basic operations such as processing data and data exchanges with the system's memory.

1.4 References

- Twitter API Documentation: https://developer.twitter.com/en/docs
- Reddit API Documentation: https://www.reddit.com/dev/api/
- Tweepy Documentation: https://docs.tweepy.org/en/stable/
- PRAW Documentation: https://praw.readthedocs.io/en/stable/

1.5 Overview

This document is structured to provide a detailed understanding of the Social Butterfly project.

- **Overall Description**: An outline of the system's overall functionality, user characteristics, and constraints.
- **Specific Requirements**: Detailed requirements including external interface requirements, functional requirements, performance requirements, and design constraints.
- **User Roles and Permissions:** Definitions of different user roles and their permissions that come with that.
- **Use Cases**: Detailed user cases that describe how the users interact with the system.
- **Non-Functional Requirements:** specifications for security, maintainability, reliability, portability, usability, scalability, performance, and availability.

- **Design Constraints:** Constraints related to the API rate limits, terms of service compliance, usability, and scalability.

Specific Requirements

2.1.1 User Interfaces

- **Search Interface**:
  - **Description**: A simple and intuitive interface for users to input search parameters for both Twitter (X) and Reddit.
  - **Features**:
    - Text fields for keywords, hashtags, and user handles.
    - Dropdown menus for selecting subreddits and date ranges.
    - Options to filter by content type (e.g., tweets, posts, comments).
    - Submit button to initiate the search.
- **Report Generation Interface**:
  - **Description**: A user-friendly interface to generate and download reports based on the collected data.
  - **Features**:
    - Button to generate and download the report.
- **Dashboard**:
  - **Description**: A dashboard that displays real-time data visualizations and summaries.
  - **Features**:
    - Charts and graphs showing data trends, sentiment analysis, and other key metrics.
    - Summary statistics and insights.

2.1.2 Hardware Interfaces

- **Client Device**:
  - **Description**: The tool will run on standard desktop and laptop computers.
  - **Requirements**:
    - No special hardware requirements beyond a typical computing device capable of running Python.
- **Server/Cloud Infrastructure**:
  - **Description**: If deployed as a web application, it will interface with server or cloud infrastructure.

    - **Requirements**:
        - Sufficient capacity to handle data volume and processing requirements.

2.1.3 Software Interfaces

- **Twitter API (Tweepy)**:
    - **Description**: Interface with the Twitter API to fetch tweets.
    - **Requirements**:
        - Authentication using OAuth.
        - Ability to retrieve tweets based on user-defined parameters.
- **Reddit API (PRAW)**:
    - **Description**: Interface with the Reddit API to fetch posts and comments.
    - **Requirements**:
        - Authentication using OAuth.
        - Ability to retrieve posts and comments based on user-defined parameters.
- **Database**:
    - **Description**: Interface with a database system to store collected data.
    - **Requirements**:
        - Structured storage of data for easy access and processing.
- **Visualization Libraries**:
    - **Description**: Use libraries such as Matplotlib or Plotly for generating data visualizations. (Python Libraries)
    - **Requirements**:
        - Capability to create charts and graphs for data analysis. (pandas in my python)

2.1.4 Communications Protocols

- **HTTP/HTTPS**:
    - **Description**: Communication with the Twitter and Reddit APIs will use HTTP/HTTPS.
    - **Requirements**:
        - Ensure secure data transmission.
- **OAuth**:
    - **Description**: Authentication with the Twitter and Reddit APIs using OAuth.
    - **Requirements**:
        - Manage secure access tokens for data retrieval.

2.2 Functional Requirements

- **Data Retrieval**:
    - Ability to retrieve data from Twitter (X) and Reddit based on user-specified search parameters.
- **Data Storage**:
    - Store the retrieved data in a structured format within a database.
- **Data Analysis**:
    - Provide capabilities for analyzing the retrieved data, including sentiment analysis, frequency analysis, and trend identification.
- **Report Generation**:
    - Generate reports summarizing the analysis results in various formats (e.g., PDF, CSV).
- **User Authentication**:
    - Support user authentication to secure access to the tool's features and stored data.
- **Error Handling**:
    - Handle errors gracefully, providing informative error messages and logging issues for further investigation.

2.3 Non-Functional Requirements

2.3.1 Reliability: Ensure high reliability with a minimal failure rate.

- **Data Integrity:** Implements robots data retrieval and storage to prevent data loss or corruption.
- **Error Handling:** Incorporating an comprehensive error handling mechanism to manage and log errors which will ensure the system stability.
- **Automated Testing:** This will conduct an automated unit, integration, and system testing to detect and fix issues early in the development process.
- **Redundancy:** Using redundant components and failover strategies to maintain operations in case of component failure.

2.3.2 Availability : The system should be available 99.9% of the time, pending maintenance of any API services.

- **Uptime:** Designed the system to maximize uptime with minimal interruptions for maintenance or updates.
- **Monitoring:** Implementing continuous monitoring of the systems health and performance making it faster to detect issues and address issues quickly.

- **Recovery:** Creating a recovery plan to restore services quickly in the event of a major issue or failure.

2.3.3 Security: Implement strong security measures to protect user data and API keys. This will comply with Twitter and Reddit terms of service. Example: Privacy

- **Access Control:** Implementing intricate authentication and authorization mechanisms to ensure that only authorized users can access the system and its data.
- **Compliance:** To ensure compliance with relevant data protection regulations and industry standards.
- **Security Audits:** Performing regular security audits and vulnerability assessments to identify and manage potential threats to the system.
- **Data Encryption:** Using encryption for data at rest and in transit to safeguard all sensitive information.

2.3.4 Maintainability: Design for ease of maintenance with well-documented code.

- **Code Documentation:** Clear separation of concerns in the architecture to facilitate updates and bug fixes.
- **Modular Design:** Modular architecture to allow individual components to be uploaded or replaced without affecting the entire system.
- **Automated Deployment:** Which allows automated deployment pipelines to streamline updates and rollbacks.
- **Issue Tracking:** Using an issue tracking system, this can allow us to manage the bug reports and feature requests efficiently.

2.3.5 Portability

- **Cross Platform Support:** Ensure portability across different operating systems, primarily supporting Windows and macOS.
- **Dependency Management:** To manage external dependencies effectively to avoid compatibility issues when porting the application from different platforms, such as IOS or Android.
- **Platform Independence**: To use platform independence technologies and libraries to facilitate easy deployment of various operating systems.
- **Containerization:** Using containerization to package the application ensuring consistent behavior across different environments, like using a docker.

2.3.6 Performance

- **Fast Data Retrieval and processing**: This ensures that the system can retrieve and process data quickly to provide users with fast results.
  - Data retrieval will optimize API requests and responses to minimize latency.
  - Data Processing utilizes parallel processing and multithreading techniques to handle large volumes of data

- **Target Response Time:** Achieving a response time of less than 2 minutes for search queries and report generation tasks.
  - Search queries which will ensure that the system can process search parameters and retrieve proper data from Twitter(X) and Reddit.
  - Report generation which will design the report generation process, compile and format data efficiently.

  - **Caching Mechanism:** By implementing a caching mechanism to store frequently accessed data, this will reduce the need for repeated API calls and in turn improve response times
    - Data caching will search results for a configurable durations to speed up subsequent queries with similar parameters
    - Report caching will generate reports quickly for retrieval and download, which reduces the amount of time for re-generation.

  - **Resource Optimization:** Will optimize the use of system resources such as the memory and CPU which will make sure the data is handled correctly.
    - CPU utilization will distribute processing tasks evenly to make optimal use of available CPU cores which will in turn minimize the processing time.
    - Memory management which will implement efficient memory management techniques to handle large datasets without exhausting system resources.

  - **Database performance:** which ensures the database is optimized for high performance and that will have fast read and write operations for users.
    - Indexing which will cause fields to speed up data retrieval.
    - Normalization which the design of the database for efficient storage and retrieval of data which will minimize the redundancy and improve query performance.

- **Scalability:** designs the system to maintain performance levels that allow data and the number of users to increase.
    - High volume accounts which ensures the system can handle data retrieval from high traffic social media accounts and manage large datasets effectively.

2.4 Design Constraints

- **API Rate Limits**:
    - Comply with the rate limits imposed by the Twitter and Reddit APIs to avoid being blocked or restricted.
    - Rate limiting services which implement strategies such as requesting throttling and exponential backoff to manage and distribute API requests effectively
    - Continuously monitoring the number of API calls and implementing alerts to notify admins when approaching rate limits.

- **Compliance with Terms of Service**:
    - Comply with the terms of service of Twitter and Reddit regarding data usage and privacy.
    - Implement measures to protect user data and API keys including encryption.
    - Maintain detailed logs of data access and API usage to ensure accountability.

- **Scalability**:
    - Design to scale as the number of users and the volume of data increase.

- **Usability**:
    - Ensure the user interface is intuitive and easy to use, requiring minimal training for new users.