"Freestyle" Project - Implementation

> See: The "Freestyle" Project

Learning Objectives

1. Create a tool to address user needs and solve a business problem.
2. Practice implementing software according to a plan.
3. Practice investigating and leveraging third-party services and Python packages to speed development and enhance capabilities.
4. Practice incorporating quality control best practices into the development process.

Instructions

After defining the requirements and objectives of your proposed solution, and after investigating its technical feasibility, it's time to write (or continue writing) Python code to implement some or all of the application's desired functionality.

The implementation should adhere to the requirements below.

Documentation Requirements

Your project repository should contain a "README.md" file. The README file should provide instructions to help someone else install, setup, run, and test your program. This includes:

- instructions for creating and activating a virtual environment, most likely using Anaconda
- instructions for installing package dependencies, most likely using Pip
- instructions for obtaining and setting environment variables, as necessary

As you document for your application, strive to make it as easy as possible for someone else (or even your future self) to install it, use it, and understand what it is about.

Also, your codebase should be reasonably documented with docstrings and other comments as necessary, to help others (and your future self) understand the code.

Licensing Requirements

Choose a software license, and include a corresponding file called "LICENSE" or "LICENSE.md" in the root directory of your repository. If you need help choosing a license, ask an instructor for guidance.

Security Requirements

If your program requires sensitive information like secret passwords, API keys, or other credentials, those secret values should absolutely not be included in the source code or its revision history. For example, use environment variables in conjunction with a ".env" file and a ".gitignore" file.

Quality Requirements

Scan your application's codebase for duplication of terms, and refactor (using custom functions as necessary) to simplify the code and make it easier to maintain.

As desired, integrate your GitHub repository with a service like Code Climate to provide automated code quality checks.

Testing Requirements

Implement at least a few automated tests using the Pytest package.

As you think about ways to test your application, consider asking yourself questions like the following:

- As I was developing this application, what manual steps did I take to ensure it was functioning properly? Can I automate those manual processes?
- Is it possible for the application to receive user inputs that are unexpected or invalid? How should the application handle various invalid inputs?
- How should the application's component functions perform given various inputs, whether valid or invalid?
- Are there any functions or sections of the code which aren't easy to read or understand? Is there a way to use examples to communicate what is supposed to happen?
- If the application processes data from the Internet: Is there a way to test the application's functionality without making any additional / unnecessary network requests?
- If the application processes data from a CSV file or database: Is there a way to test the application's functionality without affecting the development or production environment datastores?

As desired, integrate your GitHub repository with a Continuous Integration service like Travis CI to automatically run your tests on the CI server whenever new code is pushed to the remote GitHub repository.

Dev Process Requirements

Iteratively develop your project using version control practices. Save new versions of your source code as you reach key milestones.

You are encouraged to use branch operations to develop logically-related features on a specific branch, then push that branch to GitHub in order to create a Pull Request, where you can further review your proposed changes and allow automated checks to run before eventually "merging" the code into the master branch.

If working in a group, each group member must make significant contributions to the application's source code! Any group member not committing significant portions of the code may be subject to deductions.

Submission Instructions

Submit the designated Google Form before the designated date, providing the URL to your GitHub repository, and any others (like a web application URL) as desired.

Evaluation

Project implementations will be evaluated according to the requirements set forth above, as summarized by the rubric below:

Category	Requirement	Weight
Uniqueness and Individuality	Exhibits creativity, and a unique set of functionality	15%
User Experience	Provides a simple, pleasant, and intuitive experience for the user, with clear usage instructions, and free of idiosyncrasies or errors	15%
Documentation	Contains a comprehensive README file; includes Python docstrings as necessary	20%
Licensing	Contains an appropriate LICENSE file	5%
Security	Excludes sensitive information and credentials; protects user data as necessary	10%
Quality	Simplified to remove or minimize code duplication	10%
Testing	Contains relevant automated tests	10%
Dev Process	Submitted via Git repository which reflects an incremental revision history, branch operations, a Pull Request workflow, and contributions from all team members	15%

This rubric is tentative, and may be subject to slight adjustments during the grading process.