DEVELOPER MANUAL NIGHTCORE MECH

SIMILABS 2022

A GUIDE TO THE SETUPS, TOOLS AND CODE NECESSARY TO DEVELOP AND MAINTAIN THE APPLICATION

DEVELOPMENT TEAM:

HANNO VISAGIE	**31594883**
HANO STRYDOM	**31597793**
MICHAEL ROSIN	**31704948**
LLEWELLYN ANTHONY	**32969694**
ANNIKA DU TOIT	**31842534**
SHENÉ BOSHOFF	**31775357**

23/11/2022

Version 1.0.0

VERSION HISTORY				
VERSION	**APPROVED BY**	**REVISION DATE**	**DESCRIPTION OF CHANGE**	**AUTHOR**
1.0.0		22/11/2022	First Draft OF Developer Manual	Development Team

Important Links:

SimiLabs Repository	https://github.com/ISE-Project-2022/SimiLabs_2022
SimiLabs Documentation	https://github.com/ISE-Project-2022/Documentation
GitBash	https://git-scm.com/downloads
Python	https://www.python.org/downloads/
Visual Studio	https://code.visualstudio.com/download
MySQL	https://dev.mysql.com/downloads/installer/

TABLE OF CONTENTS

1. INTRODUCTION

Several administrative duties are the responsibility of the Registrar at North-West University. Keeping track of university students' grades and a wide range of other supporting records and documentation are among these duties. In accordance with Gartner's definition of information governance, the university sees it as an all-encompassing framework that gives control over information and the procedures by which it is created, processed, and curated at the institution. The current demands from the client (the NWU Registrar), with Mr. Zander Janse van Rensburg serving as the project managing manager, requires our company to design and construct a modular workflow system that would assist academic lecturers in identifying and reporting cases of academic misconduct in accordance with the NWU SOPS.

To combat contract-cheating, the NWU Registrar must assess each instance separately and employ specialists to provide technical reports. If the technical reports do not self-evidently emphasize the severity of the plagiarism, external subject matter experts (SMEs) are asked to review the technical reports with an additional report that offers a deeper understanding of the suspected plagiarism. The technical need is to manually compare the allegedly plagiarized text in question with the original text used as evidence. The registrar also needs to identify authorship attribution, via the use of stylometry to generate reports. It is rather difficult to manually compare text, so the Registrar encouraged the NightcoreMech development team to develop a system that can automatically detect text comparisons and authorship attribution without the need to manually compare documents.

This manual serves the following purposes:
- Provides an overview of the development tools necessary to develop and maintain the application
- Provides an overview of how to set up the tools and environment in order to maintain the application
- Provides an overview of the SimiLabs 2022 repository
- The functionality of the application is clarified and what each section of code in the project files are responsible for.
- A discussion of the future of the SimiLabs plagiarism checker and possible features for a next release

2. TOOLS

The following tools will be necessary to develop and maintain the application:

2.1. DEVELOPER ENVIRONMENT: VS CODE

Visual Studio Code is a quick yet effective desktop source code editor that runs on Windows, macOS, and Linux. It contains support for JavaScript, TypeScript, and Node.js, as well as a robust ecosystem of extensions for additional languages and runtimes (including C++, C#, Java, Python, PHP, Go, and.NET).

Benefits of utilizing VS Code as an editor/development environment:

- Simplicity
- Extensibility
- Open-Source and free
- Enhanced performance and lightweight
- VS Code evolves constantly and offers a huge variety of amazing extensions and developer tools

2.2. PROGRAMMING LANGUAGE: PYTHON

Python is an interpreted, object-oriented, high-level, dynamically semantic programming language. It is particularly desirable for Rapid Application Development as well as for usage as a scripting or glue language to tie existing components together due to its high-level built-in data structures, dynamic typing, and dynamic binding. Python's simple syntax prioritizes readability and makes it simple to learn, which lowers the cost of development maintenance. Python's support for modules and packages promotes the modularity and reusability of code in programs. For all popular platforms, the Python interpreter and the comprehensive standard library are freely distributable and available in source or binary form.

Benefits of utilizing Python as a backend language (especially in the case of this project):

- One of the finest computer programming languages for creating websites and software, automating processes, and performing data analysis. Python is a general-purpose language that is used to build a wide range of programs and isn't tailored for any products. Python is one of the most popular programming languages used today due to its versatility and beginner-friendliness.
- Python excels in the areas of web app development, quick prototyping, machine learning, scripting, data science, math and statistical analysis as well as database programming. Which makes it especially suitable for this application.
- Open-Source and free
- Highly extensible and offers great performance
- Offers a ton of libraries that tailors different needs

2.3. WEB DEVELOPMENT FRAMEWORK: FLASK
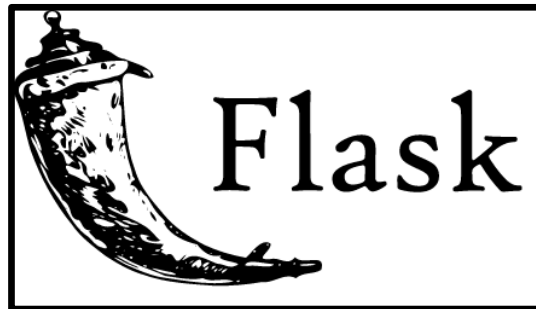
A Python package called Flask serves as a web framework that makes it simple to create web apps. Its core is compact and simple to extend; it's a micro-framework without an object relationship manager or similar capabilities. It offers several excellent features, including a template engine and URL routing. It is a web app framework for WSGI.

For the creation of Python web applications, the Web Server Gateway Interface (WSGI) has been the de facto standard. The WSGI specification describes a standard interface for web servers and online applications. Requests, response objects, and utility functions are all implemented by the WSGI toolkit known as Werkzeug. This enables a web frame to be built on it. Werkzeug serves as one of the foundations of the Flask framework.

A well-liked Python template engine is Jinja2. The web template system renders a dynamic web page by fusing a template with a particular data source. By doing this, Python variables can be sent to HTML templates.

Advantages of utilizing Flask as a web framework:

- It is scalable, allowing one to extend web apps tremendously.
- Flexibility
- Web developers can easily understand the microframework, which not only saves them time and effort but also gives them greater control over their code and the possibilities.
- It is lightweight, unlike the Django framework
- The framework has excellent documentation on how to develop web applications and use templates.

2.4. DATABASE: MYSQL

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle. MySQL can be utilized to manage a database and to manipulate data with the help of various SQL queries.

These queries consist of inserting records, updating records, deleting records, selecting records, creating, and dropping tables etc. It is fast, scalable, and an easy-to-use database management system in comparison with Microsoft SQL Server and the Oracle Database. MySQL supports many operating systems with many languages like PHP, PERL, C, C++, JAVA, etc. MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

2.5. BROWSER
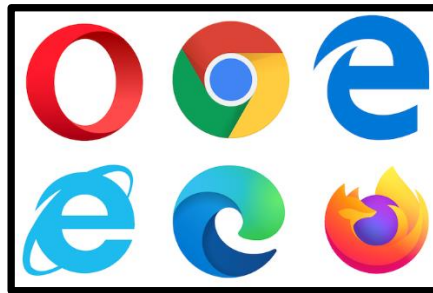
An application program known as a browser offers a way to view and engage with all the content on the World Wide Web. Web pages, movies, and photos are included as well. Prior to the invention of the Web, the term "browser" was used to refer generally to user interfaces that enable you explore (navigate through and read) text files online. Web browsers are widely used today to access the internet and are virtually considered a requirement for how many people go about their everyday lives. A Web browser is a client application that, on behalf of the browser user, sends HTTP (Hypertext Transfer Protocol) requests to Web servers all over the Internet. Most browsers are compatible with FTP and email, as well as multiple other applications such as e-commerce platforms and streaming sites. One will need a browser, such as Microsoft Edge, Google Chrome or Mozilla Firefox to use and deploy the application. The browser also serves as the interface a user will use to provide input to the application, as well as viewing output on a device's screen.

Client/server models are how web browsers operate. The browser that runs on the user's device and communicates with the Web server is known as the client, whereas the web server that communicates with the browser directly is known as the server-side. The information is subsequently interpreted and shown by the browser on the user's device. Web browsers typically consist of several cooperating components. The user interface (UI), which is the point at which the user interacts with the browser, is included in this. The rendering engine responds to requests from the browser engine and renders the requested web page by deciphering the HTML or XML data. Security and communication on the internet are handled through networking. The JavaScript code on a website is interpreted and executed using a JavaScript interpreter. Widgets like windows are created using the UI backend. A persistence layer that manages data like cookies, caches, and bookmarks is known as data persistence or storage.

6

3. SETUPS

3.1. DEVELOPER ENVIRONMENT: VS CODE

1. Download the Visual Studio Code installer for Windows:

   https://code.visualstudio.com/.

2. Once it is downloaded, run the installer (VSCodeUserSetup-{version}.exe). This will only take a minute.

3. By default, VS Code is installed under
   C:\Users\{Username}\AppData\Local\Programs\Microsoft VS Code.

4. Useful/necessary extensions to add to editor (via Visual Studio Marketplace or within the editor itself):

   - Path Intellisense
     - Visual Studio Code plugin that autocompletes filenames.
   - Docker
     - The Docker extension makes it easy to build, manage, and deploy containerized applications from Visual Studio Code. It also provides one-click debugging of Node.js, Python, and .NET Core inside a container. Dockerizing the application is also a good idea for the next release.
   - Github Copilot
     - GitHub Copilot uses OpenAI Codex to suggest code and entire functions in real-time right from your editor. Trained on billions of lines of public code, GitHub Copilot turns natural language prompts including comments and method names into coding suggestions across dozens of languages. To sign up for Github Copilot in order to utilize the extension: https://docs.github.com/en/copilot/quickstart.
   - Gitignore
     - An extension for Visual Studio Code that assists you in working with .gitignore files. A very useful extension for source control. One can add a local .gitignore file by pulling .gitignore templates from the github/gitignore repository. It also provides language support for .gitignore files
   - Gitlens
     - GitLens supercharges Git inside VS Code and unlocks untapped knowledge within each repository. It helps you to visualize code

7

authorship at a glance via Git blame annotations and CodeLens, seamlessly navigate and explore Git repositories, gain valuable insights via rich visualizations and powerful comparison commands, and so much more.

- HTML CSS Support
  - HTML id and class attribute completion for Visual Studio Code.
  - Features:
    - Supports linked and embedded style sheets.
    - Supports template inheritance.
    - Supports additional style sheets.
    - Supports other HTML like languages.
    - Validates CSS selectors on demand.
- HTML Formatter
  - This extension wraps js-beautify to format your JS, CSS, HTML, JSON file.
- HTML Snippets
  - This extension adds rich language support for the HTML Markup to VS Code
- Intellicode
  - The Visual Studio IntelliCode extension provides AI-assisted development features for Python, TypeScript/JavaScript and Java developers in Visual Studio Code, with insights based on understanding your code context combined with machine learning.
- Isort
  - A Visual Studio Code extension that provides import sorting
- Jupyter
  - A Visual Studio Code extension that provides basic notebook support for language kernels that are supported in Jupyter Notebooks today. The Jupyter Extension includes the Jupyter Keymaps and the Jupyter Notebook Renderers extensions by default. The Jupyter Keymaps extension provides Jupyter-consistent keymaps and the Jupyter Notebook Renderers extension provides renderers for MIME types such

as latex, plotly, vega, and the like. Both of these extensions can be disabled or uninstalled.

- Jupyter Cell Tags
  - o Alllows one to execute Jupyter Notebook cells within VS Code
- Kabukichõ
  - o A techno-neon, autorobotic, VHS-degradaded, superatomic-AI color theme for Visual Studio Code.
- Live Server
  - o Allows one to immediately test a web app on a browser
- Material Theme Icons
  - o An icon theme for VS Code
- Prettier
  - o Prettier is an opinionated code formatter. It enforces a consistent style by parsing your code and re-printing it with its own rules that take the maximum line length into account, wrapping code when necessary.
- Pylance
  - o Pylance is an extension that works alongside Python in Visual Studio Code to provide performant language support.
- Python
  - o A Visual Studio Code extension with rich support for the Python language (for all actively supported versions of the language: >=3.7), including features such as IntelliSense (Pylance), linting, debugging, code navigation, code formatting, refactoring, variable explorer, test explorers etc.
- Rainbow Brackets
  - o Provide rainbow colors for the round brackets, the square brackets and the squiggly brackets. This extension provides to be very useful when debugging and searching for bracket closures.
- Shell Launcher
  - o Easily launch multiple shell configurations in the terminal, such as PowerShell, Bash, CMD etc.

3.2. PROGRAMMING LANGUAGE: PYTHON

The first step is to install Python on the machine, in order to be able to deploy the streaming application on a device. Open a web browser and navigate to the official Python website: https://www.python.org/downloads/windows/. Select the Download tab for Windows. Choose a Python 3 release, in this case, select a Python version such as 3.10.2. If one happens to be using a using a 32-bit installer, click the link to get the Windows x86 executable installer. Download the Windows x86-64 executable installer the Windows installation is a 64-bit version. Launch the Python installer after downloading it. Select the box next to "Install launcher for all users". Additionally, one can include the interpreter in the execution path by selecting the "Add Python 3.7 to path" check box. The figure below displays the window that will pop up once the installer file is being executed:
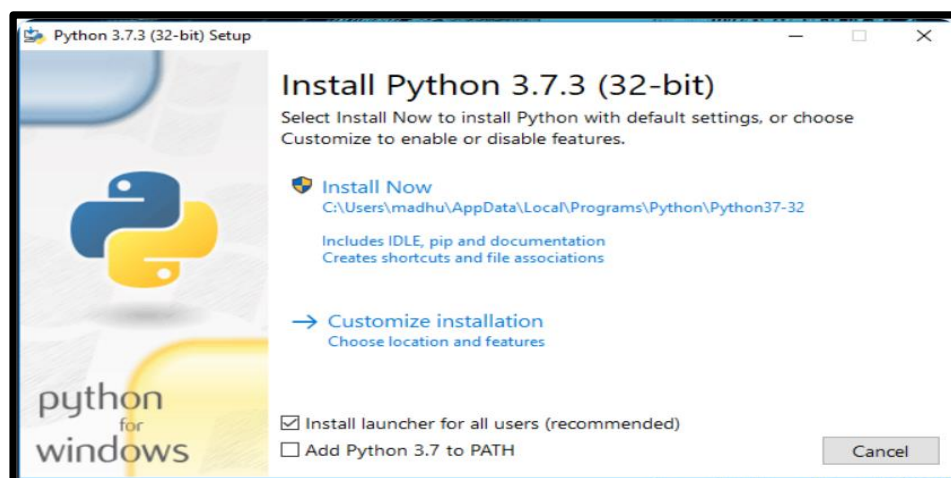
Python Installation

Choose the "Customize installation" option. By selecting the following check boxes, you can select the optional features for installation:

- Documentation
- pip
- tcl/tk and IDLE (to install tkinter and IDLE)
- Python test suite (to install the standard library test suite of Python)
- Install the global launcher for ".py" files. This makes it easier to launch Python on a device.
- Install for all users.

10
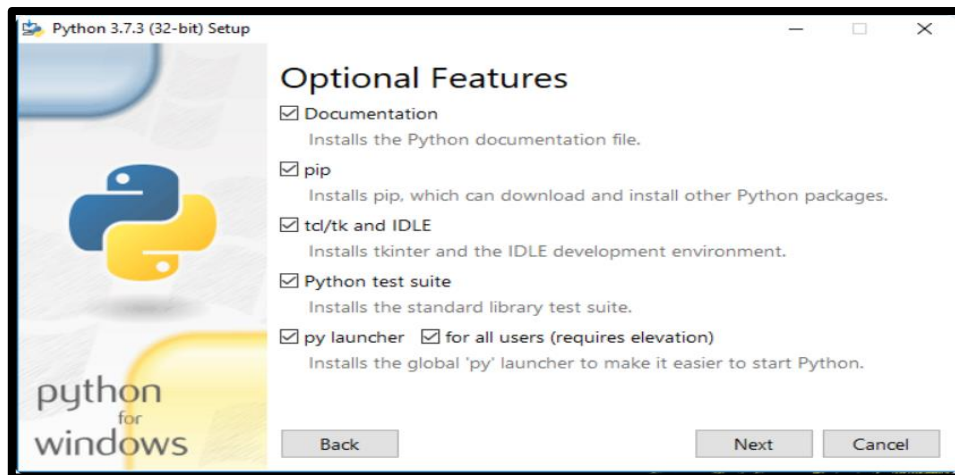
Python Optional Features

Click on Next. This will navigate the user to the Advanced Options page for the Python installation. Tick the boxes next to "Add Python to environment variables" and "Install for all users". One can alternatively choose the "Associate files with Python", "Make shortcuts for installed programs" and other sophisticated features. The Python installation directory shown in this step should be noted. It is required for the next action. Click Install to begin installation after making your selections for the Advanced settings. After the installation have been completed successfully, the figure below will pop up on the device. Click on Close to exit the installer.

Python Installation Success

If one configured the Advanced settings during the installation process and added Python to the system environment variables, this step can be skipped. In order to ensure that the Python system variable have been added to the system environment

variables for Windows, Navigate to the search glass at the bottom of the screen, next to the windows icon. Enter "environment" in the search bar and click on "Edit the system environment variables" option that pops up. Click on the "Environment Variables" button. Find the system's Python installation directory. Python will be installed in the following locations if you followed the instructions above exactly:

C:\Program Files (x86)\Python37-32: for a 32-bit installation

C:\Program Files\Python37-32: for a 64-bit installation

If one installed a different version of Python, the folder name might be different. Look for a folder with the word "Python" in the name. As indicated below, add the following entries to the PATH variable. Click on "New" and add the Python path variable:
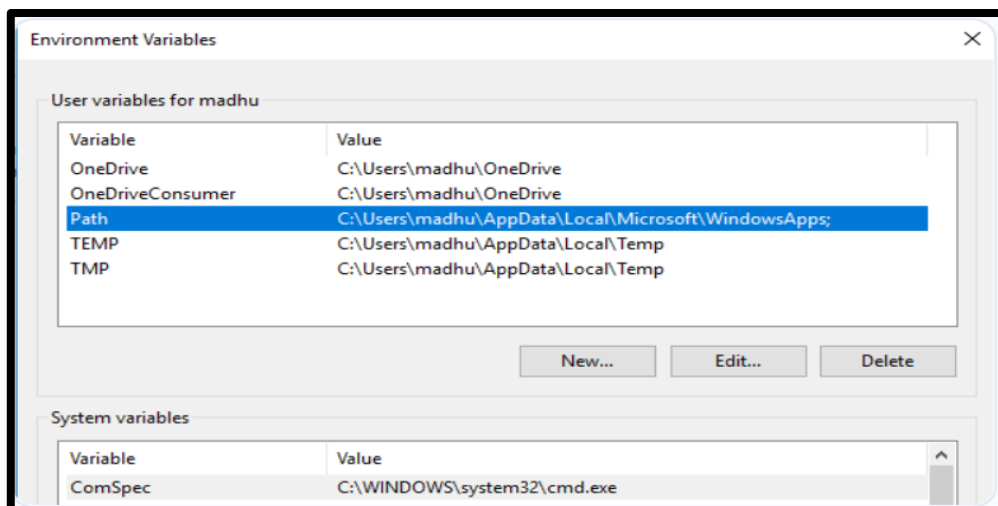
System Environment Variables
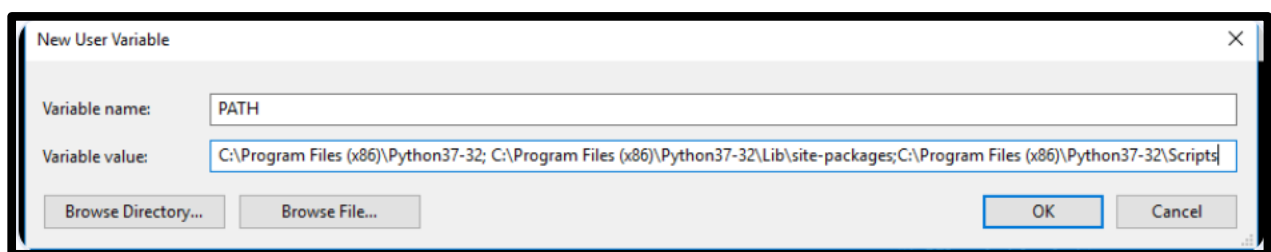
Python Environment Variable

After the Python environment variable have been added to the system, close all the windows by clicking on the OK button on each window. The next step is to ensure that Python have been successfully installed on the machine and that the operating system detects the Python environment variable. Open a terminal, and enter the following command into the terminal:

Python -V

As indicated in the figure below, Python have been successfully installed on the machine. If Python is not recognized as an internal or external command, it means that one of the steps have been configured incorrectly. One needs to redo the Python installation steps if that happens to be the case.
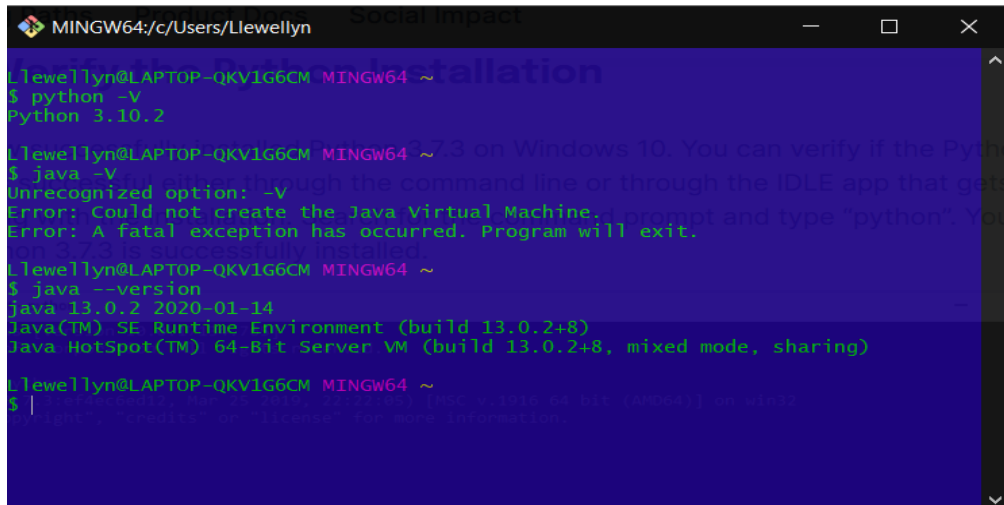

3.3. WEB DEVELOPMENT FRAMEWORK: FLASK

The following dependencies will be needed when setting up Flask:

- Werkzeug implements WSGI, the standard Python interface between applications and servers.

- Jinja is a template language that renders the pages your application serves.

- MarkupSafe comes with Jinja. It escapes untrusted input when rendering templates to avoid injection attacks.

- ItsDangerous securely signs data to ensure its integrity. This is used to protect Flask's session cookie.

- Click is a framework for writing command line applications. It provides the flask command and allows adding custom management commands.

Virtual Environments:

To handle the dependencies for your project in both development and production, use a virtual environment. Virtual environments manage multiple versions of Python libraries or even Python itself for multiple Python projects on a machine. A project's interoperability with newer versions of libraries used by another project may be broken. Each project has its own virtual environment, which is an independent collection of

Python libraries. The operating system's packages and other projects won't be impacted by the installation of packages for one project. The venv module for creating virtual environments is included with Python.

To create a virtual environment for your Python project, do the following:

1. Open a Bash shell and type the following:

   mkdir myproject

   cd myproject

   python3 -m venv venv

2. Next, you'll need to activate the virtual environment. Type the following into the Bash shell:

   . venv/bin/activate

   Your shell prompt will change to show the name of the activated environment.

Install Flask

Within the activated environment, use the following command to install Flask:

pip install Flask

If you haven't created or cloned the SimiLabs repository, run the following in a Bash Shell:

mkdir SimiLabs_2022

cd SimiLabs_2022

These commands will create the project folder and navigate inside the folder.

The project folder must contain all the following directories:
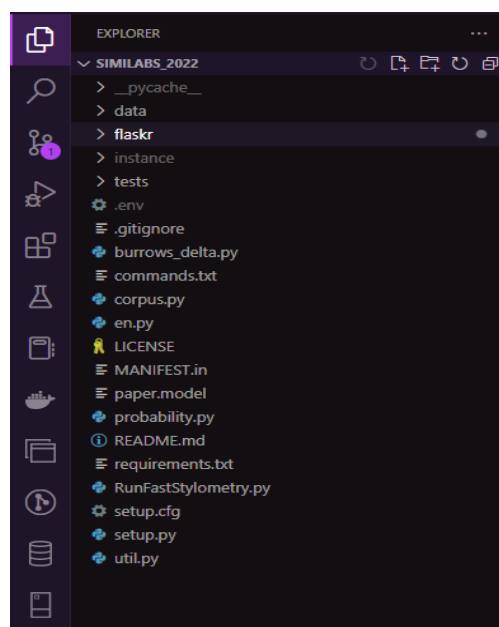
- Data: contains the test and training corpora for the stylometry functionality.
- Flaskr: contains the main application code and files.
- Tests: an optional directory containing test modules.
- If one created a virtual environment for the project, a venv folder will appear in the root directory where Flask and other dependencies are installed.
- The python files in the root directory are the stylometry files and is stored in the root directory since one cannot run a machine learning service alongside a be application service on the main thread.
- .env: contains environment variables for the project.
- .gitignore: a file that allows ne to untrack certain files in a project folder when utilizing source control.
- Commands.txt: contains some Bash commands the user can execute to quicklu run the Flask application.
- LICENSE: a license file issued by MIT verifying the Github repo of the project.
- MANIFEST.in: a file that specifies what additional files and directories should be included in your project
- ReadMe.md: contains a short developer manual on the project.
- Requirements.txt: contains all the Python libraries necessary to develop and maintain the application.

The flaskr directory must contain the following folders:
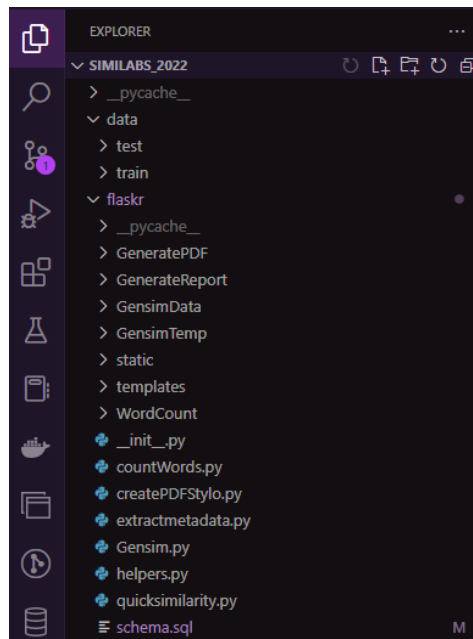
- GeneratePDF: saves PDFs to this folder in order to extract their textual content when utilizing the Quick Text functionality of the application.

- GenerateReport: generates reports with statistics when using the plagiarism checker.

- GensimData: contains the directories and data of the Etensive Text Comparison functionality of the application.

- GensimTemp: saves a document temporarily when compared to a corpora of documents to check for text similarities, as well as enabling the application to extract text from the document.

- Static: contains CSS, JS and any other documents that remains static on the web app.

- Templates: contains the HTML templates of the application

- WordCount: saves a word count of an alleged and comparison document utilizing the Quick Text functionality of the application, extracted into respective text files

- __init__.py: the file that will be executed when the application runs, also contains the most important Python library imports and URL routing.

- countWords.py: a module containing the code to extract wordcounts from an alleged and comparison document.

- createPDFStylo.py: a module that writes the statistics from the stylometry to a PDF.

- Extractmetadata.py: a module that allows the extraction of document metadata.

- Gensim.py: a module that allows one to check for text similarities between an alleged document and a corpus.
- Helpers.py: a module that allows one to highlight text similarities between an alleged document and a comparison document.
- QuickSimilarity.py: a module that generates the text similarity score on Quick Text documents
- Schema.sql: contains the DB design of the application

```
∨ GeneratePDF
  📄 Test.pdf
  📄 Test2.pdf
∨ GenerateReport
  📄 PlagiarismReport.pdf
  📄 styloReport.pdf
∨ GensimData
  > 12345678
  > 31597793
  > 31684203
  > 31775357
  > 32969694
∨ GensimTemp
  📄 Test3.docx
```

```
∨ static
  > css
  > fonts
  > images
  > js
  > pdf
  > vendor
  # baseReport.css
  # extensiveFooter.css
  # home.css
  # main.css
  # style.css
  # styloReport.css
  # text.css
  # tooltip.css
  # util.css
```
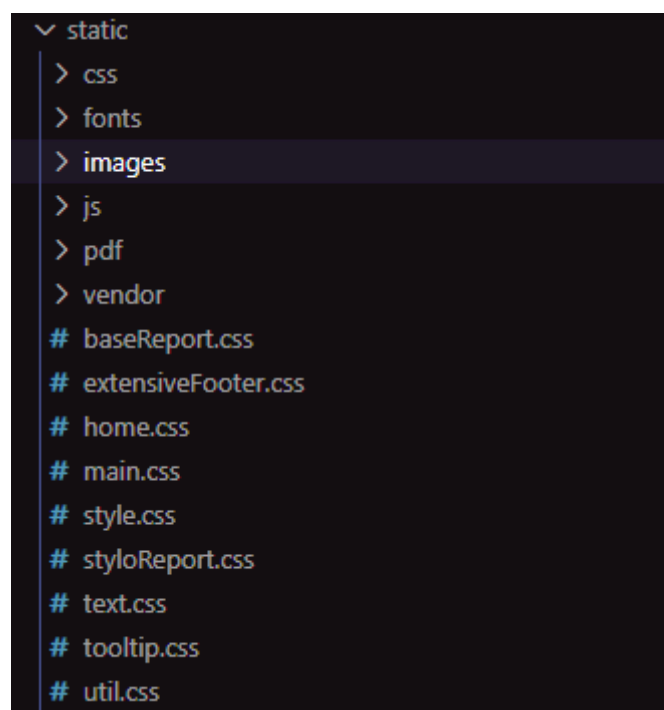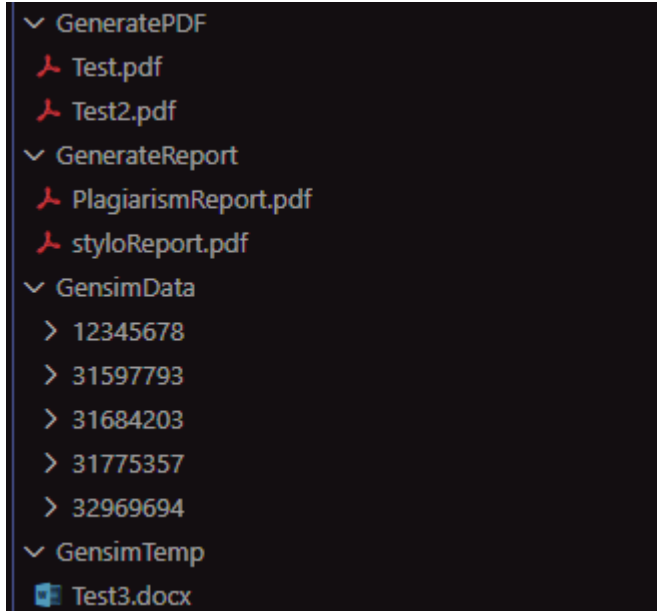
The static folder contains the following directories:

- Images: contains images generated by the application to insert into reports
- JS: contains JavaScript files that interacts with the browser for extra functionality or animations.
- PDF: contains static PDF files that is displayed on the Help page
- Vendor: contains CSS and JS files for the login page
- Additional CSS files: CSS files used throughout the project for the styling and positioning of content on the pages of the application.

```
∨ static
  > css
  > fonts
  ∨ images
    ∨ icons
      ★ favicon.ico
    ∨ ReportImages
      🖼 author_probability.png
      🖼 calibration.png
      🖼 delta_score.png
      🖼 operating_curve.png
      🖼 plot.png
    ∨ WordCloud
      🖼 wordcloud.png
      🖼 wordcloud2.png
    🖼 img-01.png
    🖼 SimiLabs.png
  ∨ js
    JS custom.js
    JS disablecheckbox.js
    JS disabledtext.js
    JS footer.js
    JS main.js
    JS materialize.js
    JS pdfobject.js
  ∨ pdf
    📄 SimiLabs_User_Manual.pdf
  ∨ vendor
    > animate
    > bootstrap
    > css-hamburgers
    > jquery
    > select2
    > tilt
```

```
∨ templates
  ∨ auth
    <> authLogin.html
    <> authRegister.html
  ∨ exceptions
    <> errors.html
  ∨ help
    <> help.html
  ∨ home
    <> home.html
  ∨ reports
    <> error.html
    <> extensiveReport.html
    <> quickReport.html
    <> report.html
    <> styloReport.html
  ∨ stylometry
    <> stylo.html
  ∨ text
    <> ExtensiveText.html
    <> QuickText.html
  <> baseNav.html
  <> baseNavHelp.html
  <> baseReport.html
  ∨ WordCount
    ☰ count.txt
    ☰ count2.txt
```

The templates directory contains the following folders:

- Auth: contains the login and register views
- Exceptions: contains a view that is displayed when the user receives an error message
- Help: contains the help view
- Home: contains the home view
- Reports: contains the views of the reports

- Stylometry: contains the stylometry view

- Text: contains the Quick and Extensive Text Views

- BaseNav HTML Files: provides base HTML templates that the other templates/views can extend upon (extend or replace page content)

3.4. DATABASE: MYSQL

Download the MySQL installer from https://dev.mysql.com/downloads/installer/

Select the community installer:
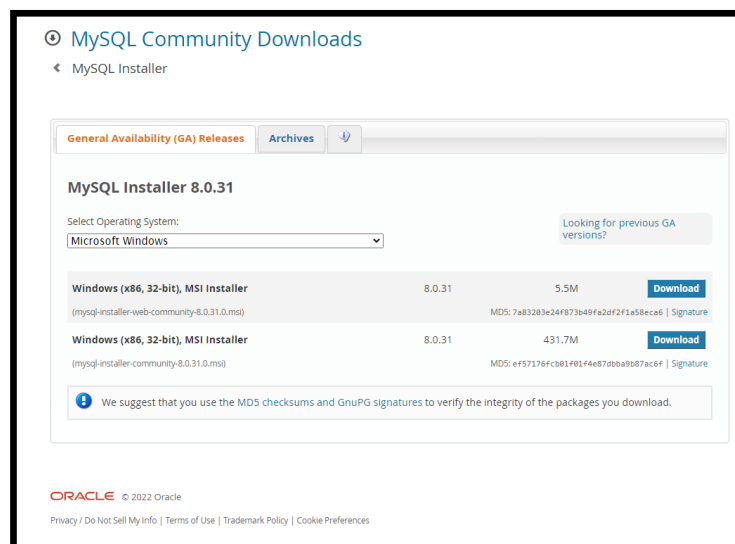
When clicking on the Download button for the community installer, you will be navigated to the following page. Click on "No thanks, just start my download". The MySQL Community Installer will be downloaded to your device.
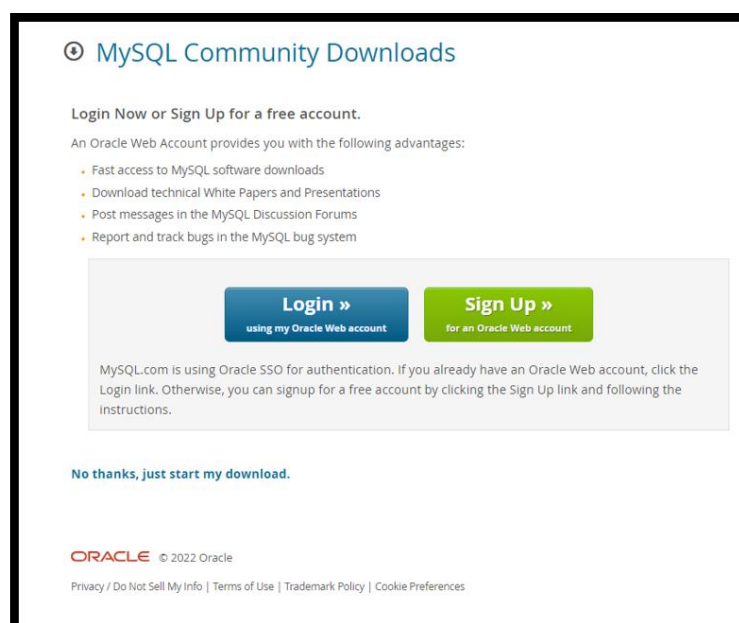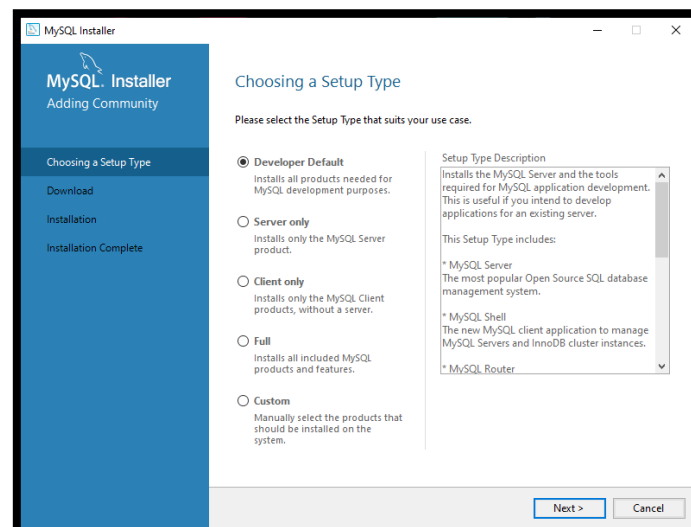
After finishing the download, open your File Explorer and double click on the installer .msi file to start the MySQL Installation process:
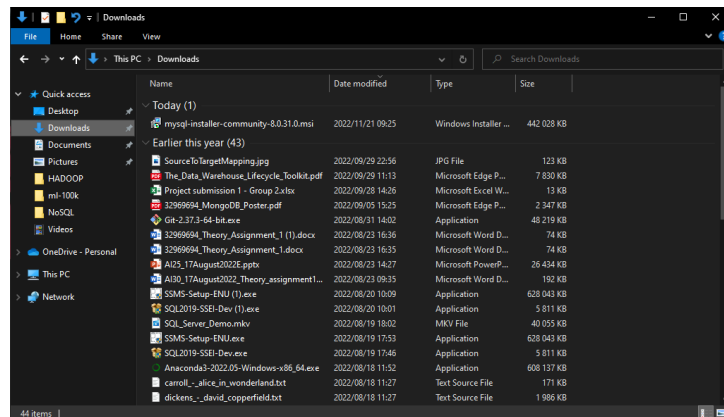
The following window will appear when running the installer. Select Developer Default:

You will be navigated to the Check Requirements tab. This contains the MySQL components that needs to be installed on your device in order to run and manage a MySQL database on a machine. MySQL Server is necessary to start a MySQL database server, MySQL Workbench as the GUI and DBMS to manage a MySQL database, optionally you can add a MySQL extension to Visual Studios that allows one to work with MySQL extensions in the IDE. MySQL Shell allows one to create and manage a MySQL database via using a shell. MySQL Router is a lightweight middleware that provides transparent routing between your application and any backend MySQL Servers. The MySQL connectors allows a MySQL database to communicate with other programming languages when performing CRUD and other database operations. Click on Execute to add the components to your device:

20
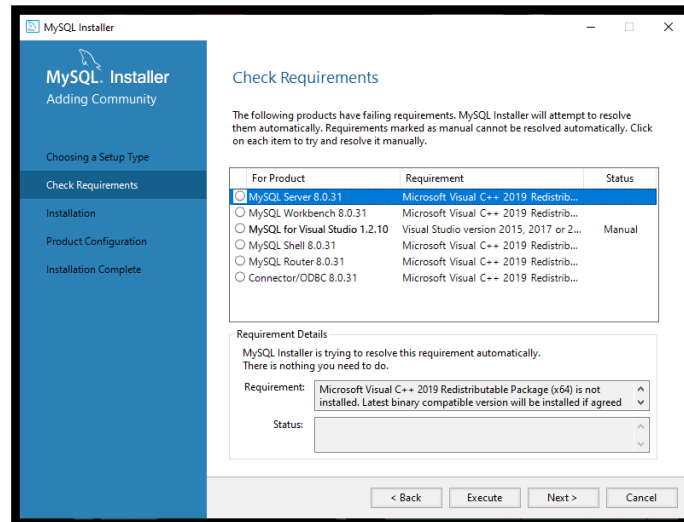
The next window will appear if the components have been successfully added:
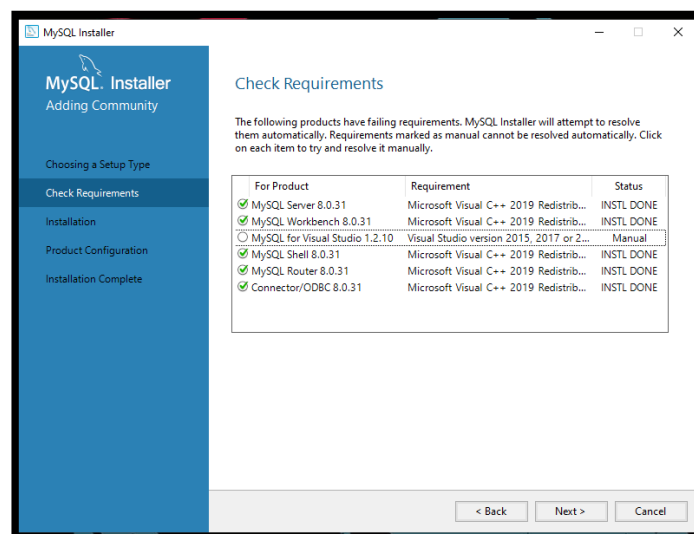
Click on Next. The following Window will appear, indicating the MySQL services and components that will be installed on your machine. It is especially important to ensure that the Python Connector is in the list of services, otherwise the application won't be able to use a MySQL database. Click on Execute:
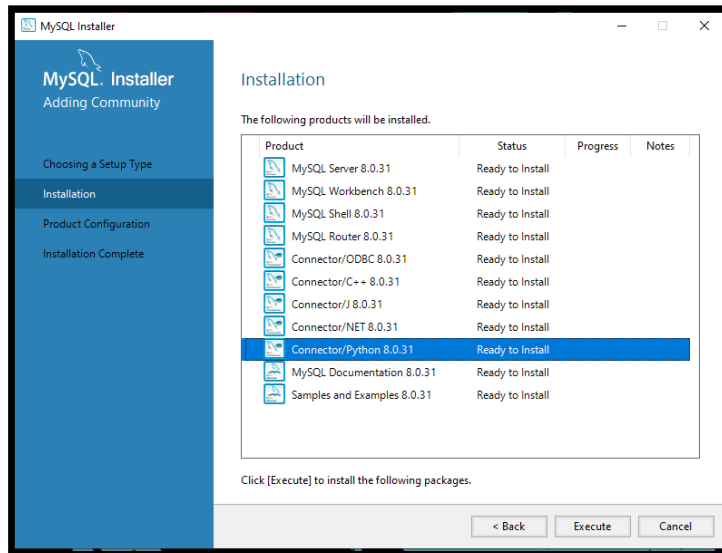
After the services have been verified, click on Next:
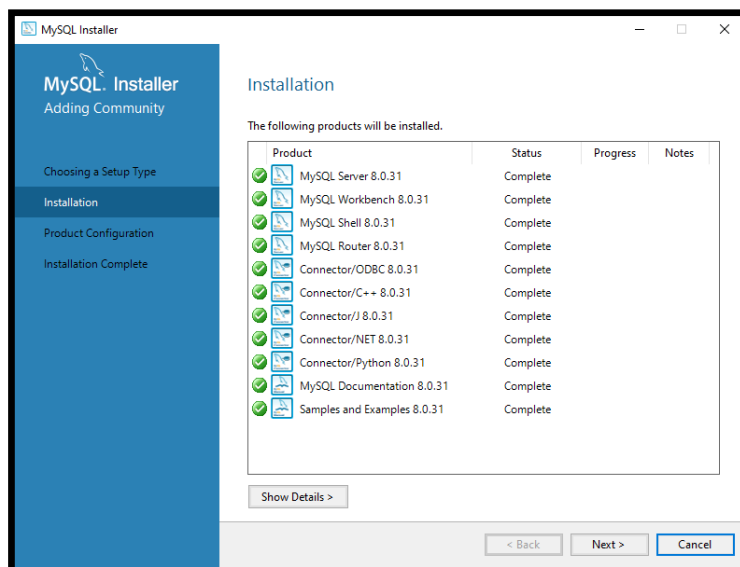
The following window will appear for Product Configurations. Click on Next:
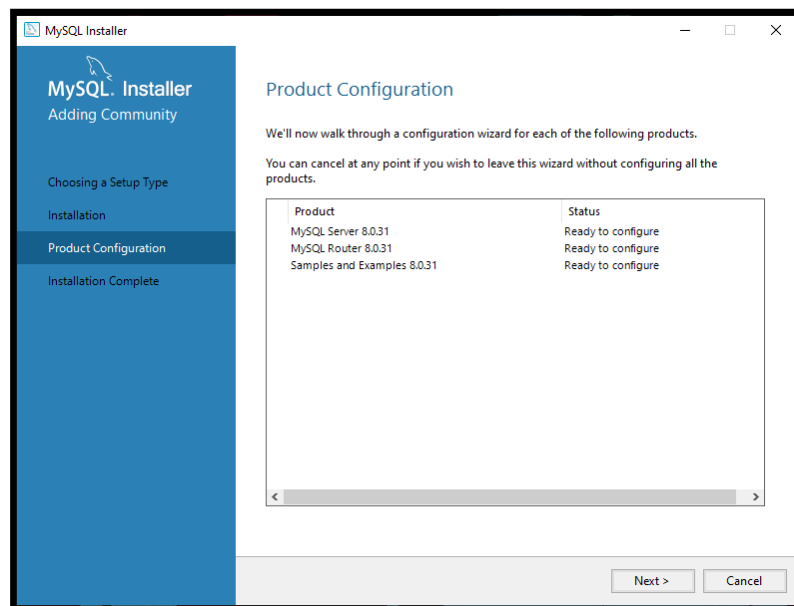
The Type and Networking tab specifies server configurations for your MySQL database and the ports your MySQL database will run on. Keep the default options and click on Next:
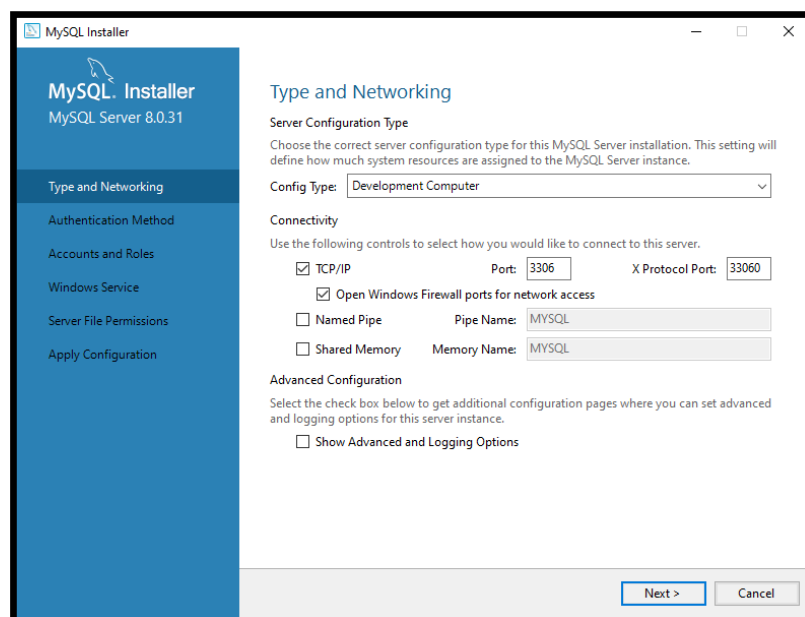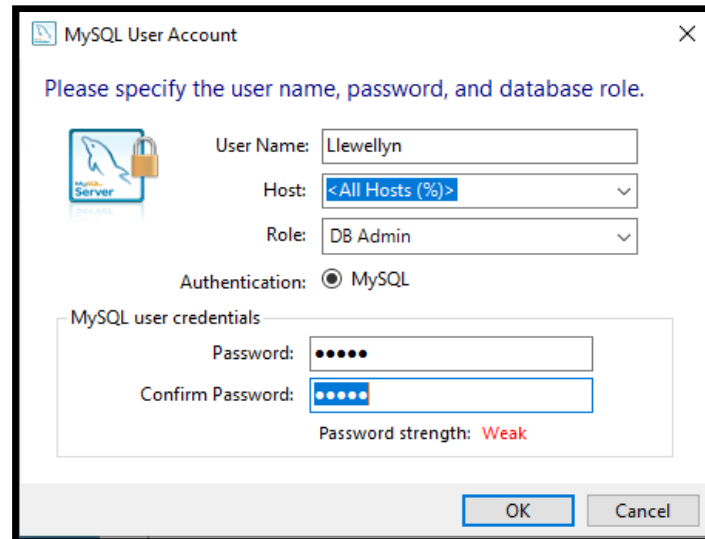
Next, the Authentication window will appear. This allows one to access a MySQL database using a user and a password. Keep the default options and click on Next:
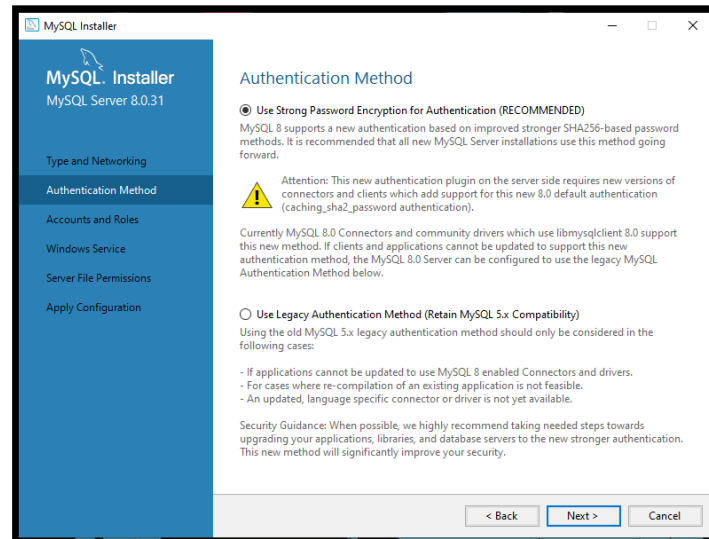
The Account and Roles window allows one to specify admin and other accounts that can be used on a machine to access and manage MySQL databases. For the root account, set the password and confirmation password to "admin". Next, click on the Add User button to add another default user.
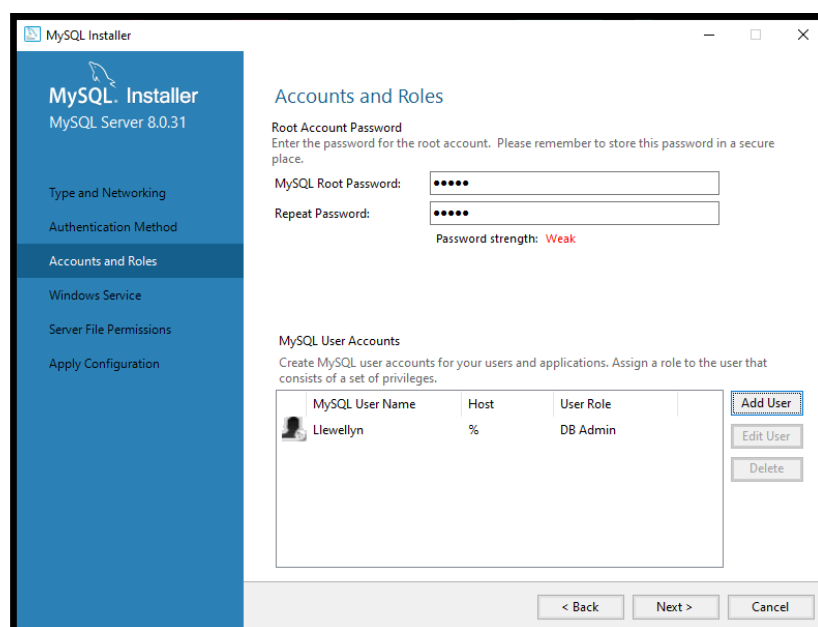
When adding another user, provide a username to assign roles to the specific user. Select Role as DB admin in order to grant all database permissions to the specified user. Once again, set the password and confirmation password to "admin":
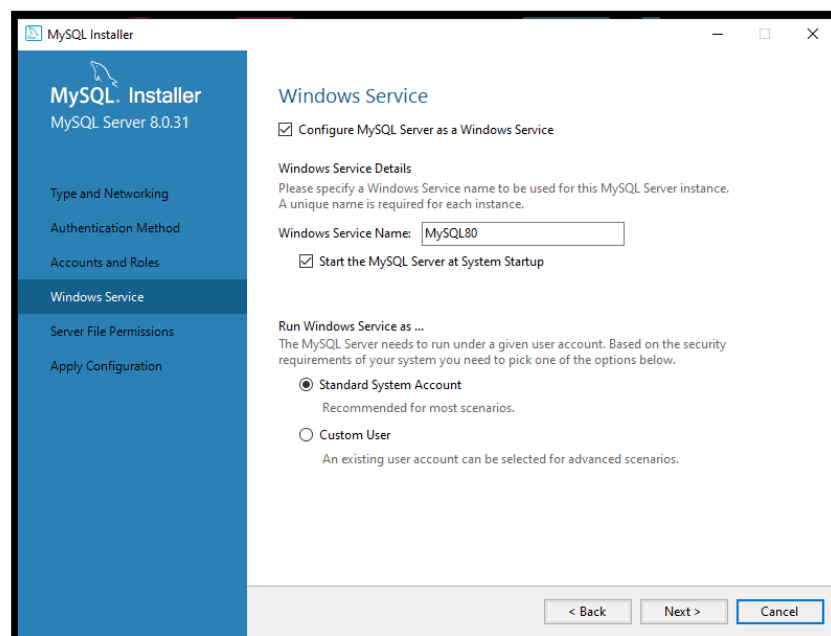
After finishing the Account and Roles setup, click Next to navigate to the next tab. The Windows Service allows one to assign a Windows service to the MySQL server. Keep the default settings and click on Next:

Server File Permissions allows one to specify where the data files of MySQL are uploaded, as well as allowing one to grant access to Windows Services or users to update MySQL data files. Keep the default settings and click on Next:

25
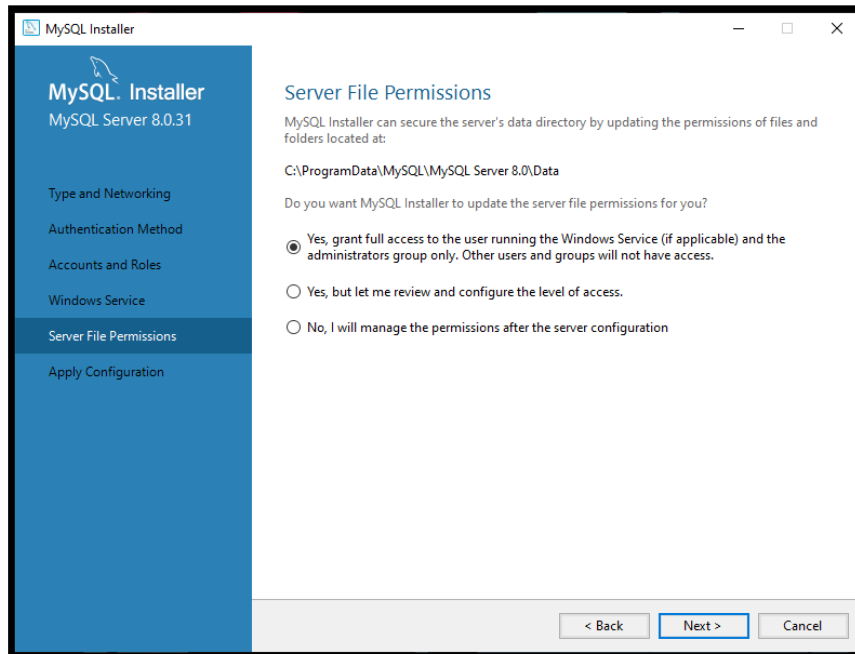
Next, you will be redirected to the final section of the MySQL installation. Verify that all the correct services have been added, and click on Execute to complete the installation of MySQL:
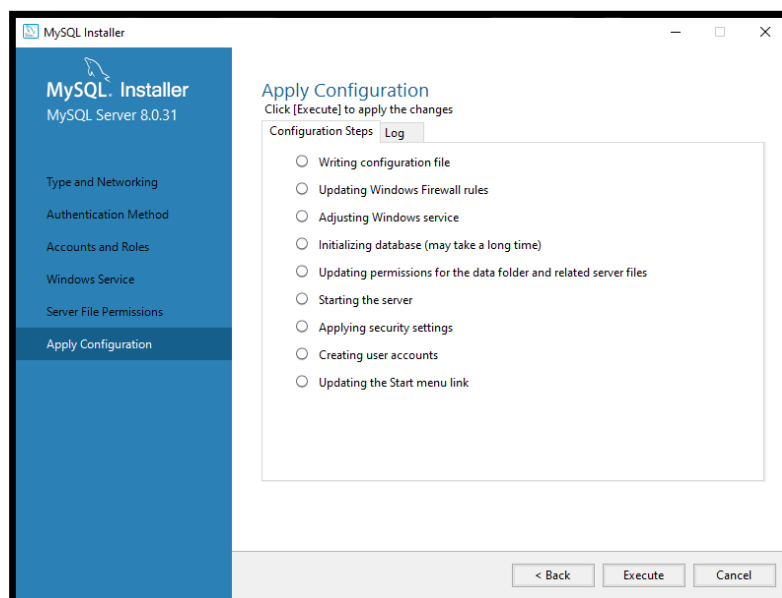
After the installation have been successfully completed, the MySQL Shell should start automatically. If this doesn't happen, it means one of the setup steps have been configured incorrectly. In that case, it might be best to restart the MySQL installation. Close the Shell if the installation has been successful.
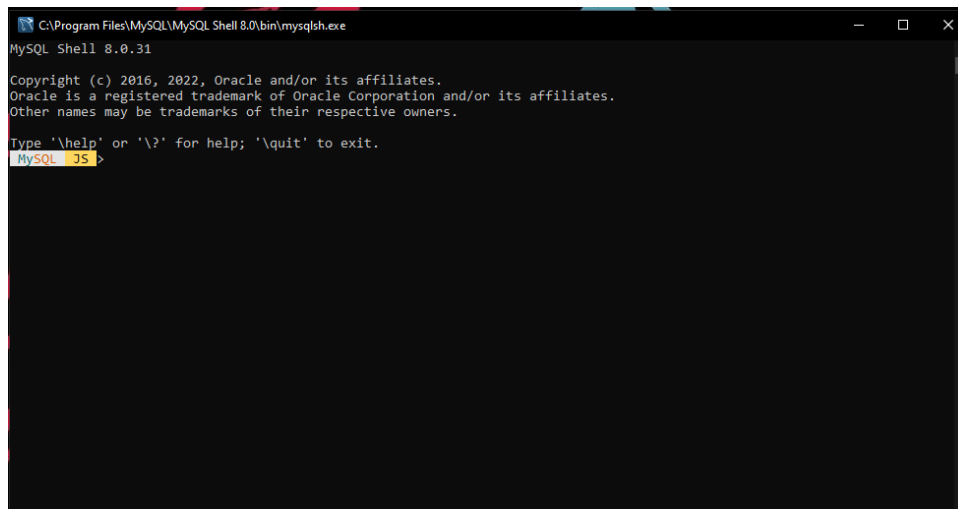
For the final part of the database setup: locate MySQL WorkBench on your device. Run the application, and the following window should appear:
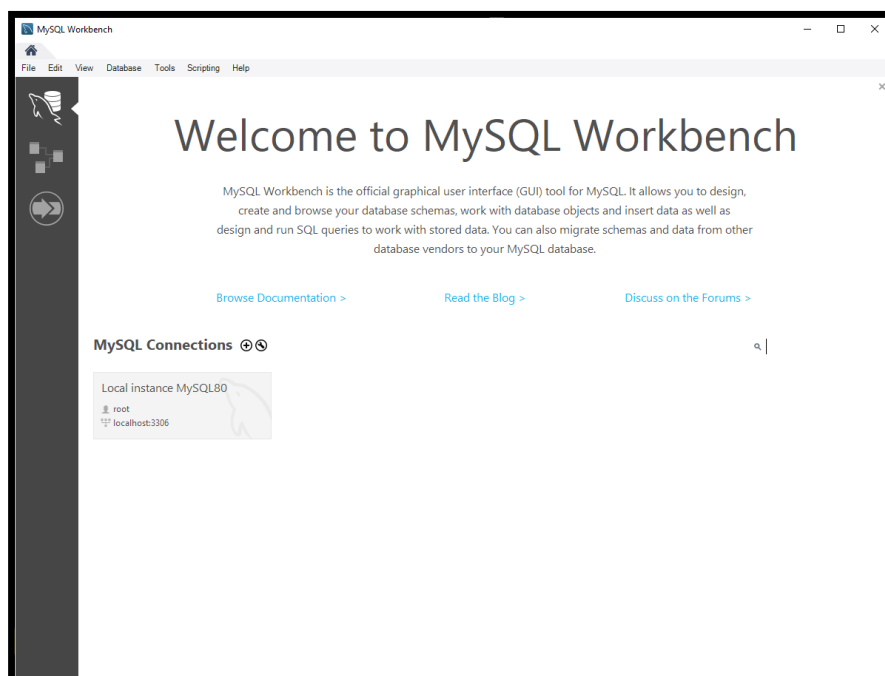
Click on the "plus" icon above the root instance of MySQL, and the following window will appear. It is necessary to create a SimiLabs 2022 server instance in order to host the MySQL database for the application. Apply the settings as indicated in the figure below, and test the connection:
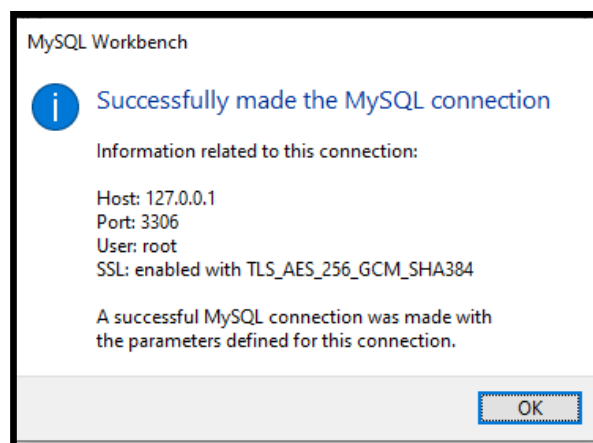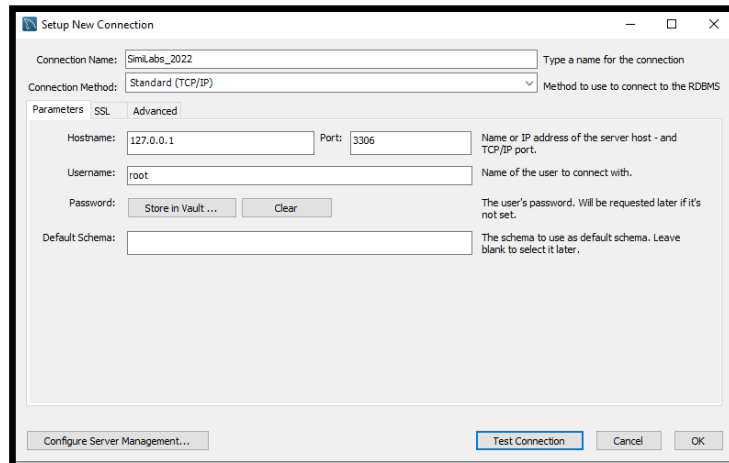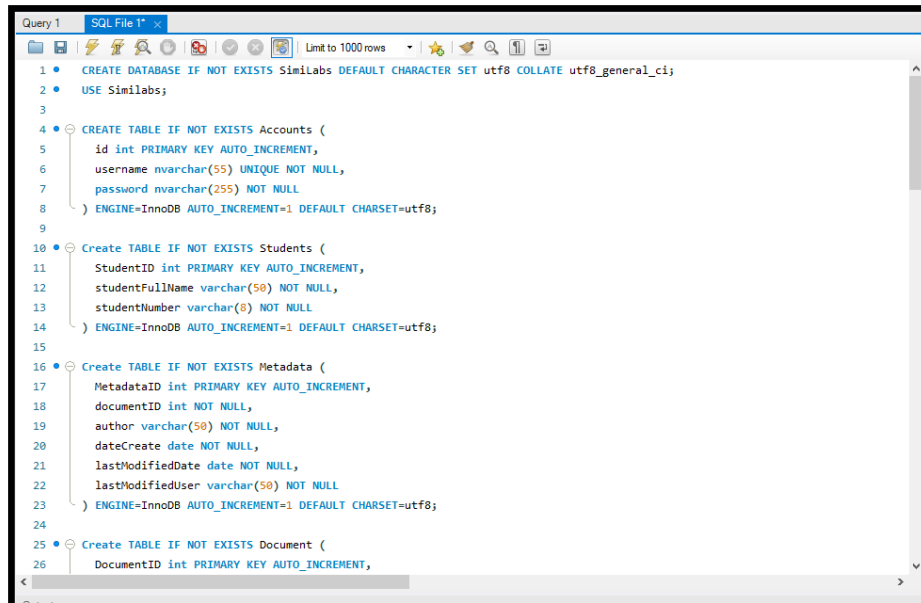
If the test connection window doesn't appear, one of the setup procedures has been configured improperly. It might be better to restart the MySQL installation in that case. Apply all the settings. Another server instance will appear in the MySQL WorkBench page: SimiLabs_2022. Double-Click on the server instance, enter your root password, and the following WorkBench environment will appear:
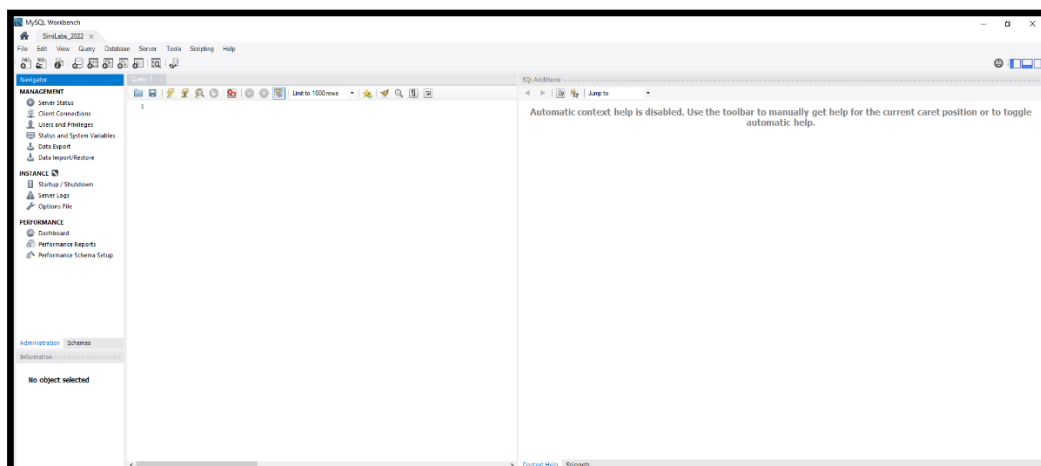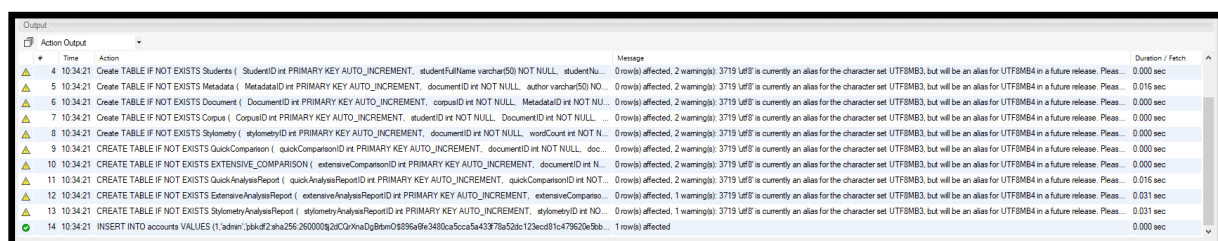
Select File → New Query Tab. Locate the Schema.sql file in your SimiLabs project, and copy the code from the SQL file into the new query tab:

Click on the lightning bolt icon to execute the query. If the window below doesn't appear without any green checkmarks, it means there was an issue executing the query, and there might have been syntax errors, invalid constraints, or logical errors in the query.

After the query have been successfully executed, navigate to Schemas in the navigator panel, refresh the panel, and your database and its tables should appear in the navigator pane and populated with data:
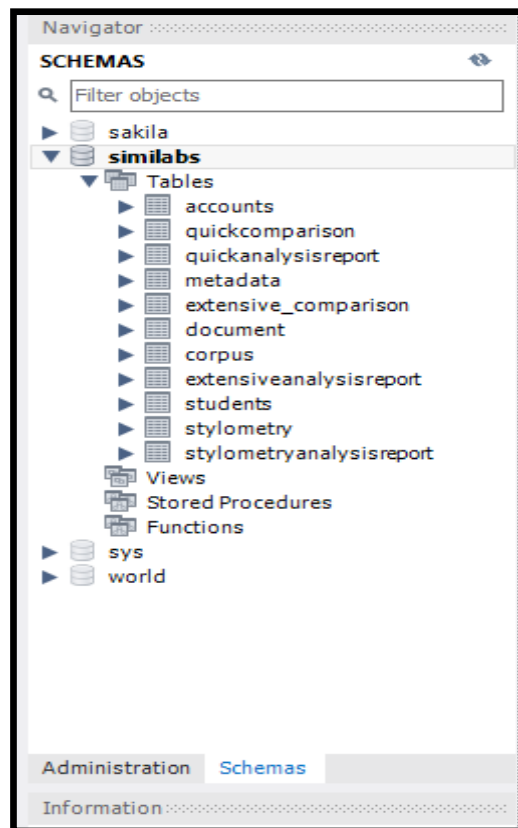
This concludes the setup of the MySQL database for the SimiLabs application.

3.5.   SIMILABS 2022 REPOSITORY

The Git for Windows distribution comes with Git Bash. Install Git for Windows the same way you would any other Windows program: https://gitforwindows.org/. Once the file has been downloaded, locate it and double-click it to launch Git Bash. Follow the installation instructions from the installer. By performing a right-click on any folder and choosing the Git Bash Here option from the context menu, you can launch Git Bash. To verify that Git Bash have been successfully installed, run the following command on any shell on Windows:

3.6. SIMILABS PROJECT SETUP

Within the environment file in your root directory, add the following code:

These environment variables store the details of the DB server and connections, as well as variables that stores the paths of documents and files in the project.

After including all relevant code and imports into the project, run the following commands in the root directory of the project using a Bash shell:

pip install .
or
cat requirements.txt | sed -e '/^\s*#.*$/d' -e '/^\s*$/d' | xargs -n 1 python -m pip install

The above commands essentially identify all the libraries and imports in your project and installs them to your system or virtual environment. Next, within the __init__.py file, add the following in order to allow the application to recognize the environment variables and other additional application setups, such as sockets and allowable file extensions for submissions:

```python
def create_app(test_config=None):
    try:
        # create and configure the app
        app = Flask(__name__, instance_relative_config=True)
        app.config.from_mapping(
            SECRET_KEY='dev'
        )

        # delete if broken
        app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024
        # Gets the current directory
        # path = os.getcwd()
        # UPLOAD_FOLDER = os.path.join(path, os.getenv('UPLOAD_DIR'))
        UPLOAD_IMG = os.getenv('UPLOAD_IMG')
        UPLOAD_PDF = os.getenv('UPLOAD_PDF')
        UPLOAD_REPORT = os.getenv('UPLOAD_REPORT')
        UPLOAD_STYLO = os.getenv('UPLOAD_STYLO')
        UPLOAD_EXTENSIVE = os.getenv('UPLOAD_EXTENSIVE')
        HELP_PAGE = os.getenv('HELP_PAGE')
        HELP_PDF = os.getenv('HELP_PDF')


        app.config['MYSQL_HOST'] = os.getenv('MYSQL_HOST')
        app.config['MYSQL_USER'] = os.getenv('MYSQL_USER')
        app.config['MYSQL_PASSWORD'] = os.getenv('MYSQL_PASSWORD')
        app.config['MYSQL_DB'] = os.getenv('MySQL_DB')


        mysql = MySQL(app)

        # insert if uploading to folder
        # if not os.path.isdir(UPLOAD_FOLDER):
        #     os.mkdir(UPLOAD_FOLDER)

        app.config['UPLOAD_IMG'] = UPLOAD_IMG
        app.config['UPLOAD_PDF'] = UPLOAD_PDF
        app.config['UPLOAD_REPORT'] = UPLOAD_REPORT
        app.config['UPLOAD_STYLO'] = UPLOAD_STYLO
        app.config['UPLOAD_EXTENSIVE'] = UPLOAD_EXTENSIVE
        app.config['HELP_PAGE'] = HELP_PAGE
        app.config['HELP_PDF'] = HELP_PDF

        ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif', 'docx'])
        app.config["TEMPLATES_AUTO_RELOAD"] = True
```

```
_init_.py 9+   ×

flaskr >  _init_.py >  create_app
87            ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif', 'docx'])
88            app.config["TEMPLATES_AUTO_RELOAD"] = True
89
90            def allowed_file(filename):
91                return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
92
93            if test_config is None:
94                # load the instance config, if it exists, when not testing
95                app.config.from_pyfile('config.py', silent=True)
96            else:
97                # load the test config if passed in
98                app.config.from_mapping(test_config)
99
100           # ensure the instance folder exists
101           try:
102               os.makedirs(app.instance_path)
103           except OSError:
104               pass
105
106           s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
107           try:
108               # doesn't even have to be reachable
109               s.bind((HOST,PORT))
110           except Exception:
111               return 'Error: server failed to startup ... could not find port number'
112           finally:
113               s.close()
114       except Exception as e:
115           return render_template("exceptions/errors.html", error = e)        sheneboshoff, 5 days ago
116
117       @app.after_request
118       def after_request(response):
119           try:
120               """Disable caching"""
121               response.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
122               response.headers["Expires"] = 0
123               response.headers["Pragma"] = "no-cache"
124               return response
125           except Exception as e:
126               return render_template("exceptions/errors.html", error = e)
127
```

After adding the above code, execute the following commands in your root directory using a Bash Shell:

---Setup commands for older version of FLASK---

export FLASK_APP=flaskr

export FLASK_ENV=development

flask run

---Setup commands for new version of FLASK---

export FLASK_APP=example

export FLASK_DEBUG=1

flask run
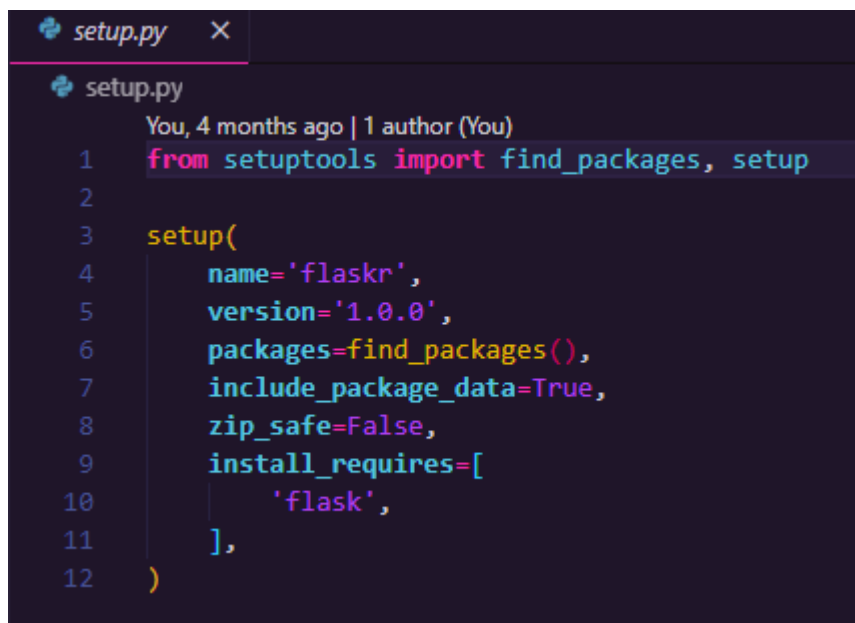
Make the project installable:

If you make the project installable, you can create a distribution file and install it in another environment in the same way that you installed Flask in the environment for your project. By doing this, deploying your project becomes as simple as installing a library, allowing you to handle everything with the regular Python tools.
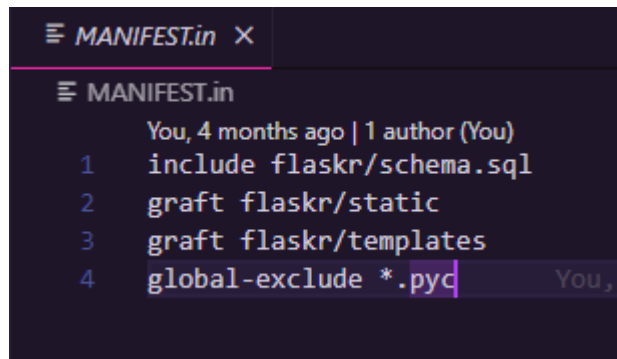
- Additionally, installing has advantages that may not be apparent from the instruction or to a novice Python user, such as:
- Because you're running from your project's directory, Python and Flask are the only ones who now understand how to use the flaskr package.
- Installing allows you to import it regardless of where you run.
- You can manage your project's dependencies just like other packages do, so pip install yourproject.whl installs them.
- Test tools can isolate your test environment from your development environment.

The setup.py file describes your project and the files that belong to it:

```python
from setuptools import find_packages, setup

setup(
    name='flaskr',
    version='1.0.0',
    packages=find_packages(),
    include_package_data=True,
    zip_safe=False,
    install_requires=[
        'flask',
    ],
)
```

The packages library instructs Python to include the specified package directories and the Python files they contain. You don't need to type in these folders because find packages() automatically identifies them. The include_package_data property is set to true to include more files, including those in the static and templates directories. To describe what this additional data is, Python requires a different file called MANIFEST.in:

```
MANIFEST.in ×

MANIFEST.in
      You, 4 months ago | 1 author (You)
  1   include flaskr/schema.sql
  2   graft flaskr/static
  3   graft flaskr/templates
  4   global-exclude *.pyc        You,
```

Use pip to install your project in the virtual environment, run this command in a Bash shell:

pip install -e .

This tells pip to find setup.py in the current directory and install it in editable or development mode. Editable mode means that as you make changes to your local code, you'll only need to re-install if you change the metadata about the project, such as its dependencies. You can observe that the project is now installed with pip list.

pip list --- Run this in the Bash shell

The following output should appear on your terminal:

Package	Version	Location
click	6.7	
Flask	1.0	
flaskr	1.0.0	/home/user/Projects/SimiLabs_2022
itsdangerous	0.24	
Jinja2	2.10	
MarkupSafe	1.0	
pip	9.0.3	
setuptools	39.0.1	
Werkzeug	0.14.1	
wheel	0.30.0	

Nothing changes from how you've been running the project so far. --app is still set to flaskr and flask run still runs the application, but you can call it from anywhere, not just the SimiLabs_2022 directory.

35

To make the app deployable, read the following from Flask's official documentation: https://flask.palletsprojects.com/en/2.2.x/tutorial/deploy/

Windows commands for setting up this project was not included. Look at the Windows command using Flask's official documentation and for additional information on the Flask framework and templates: https://flask.palletsprojects.com/en/2.2.x/

4. SIMILABS 2022 REPOSITORY: A WALKTHROUGH OF THE CODE

4.1.   REQUIREMENTS.TXT

Dependencies are simply external Python packages that your project depends on to perform its desired purpose. These dependencies are typically discovered when using Python and can be found on PyPI (Python Packages Index) or other repository management platforms like Nexus. A file called requirements.txt contains a list of every dependency needed by the SimiLabs project, and all Python projects in general. As was previously mentioned, it might also include dependencies of other dependencies. The entries on the list may or may not be pinned (either contains a package version or not). If a pin is utilized, you can indicate a particular package version (using ==), an upper or lower bound, or perhaps both, depending on the case. As mentioned in the previous section, you can install the requirements for the SimiLabs project by running the following commands in the root directory in a Bash Shell:

pip install .
or
cat requirements.txt | sed -e '/^\s*#.*$/d' -e '/^\s*$/d' | xargs -n 1 python -m pip install

A basic overview of dependencies in the requirements.txt file

Once all the dependencies are installed, you can see the precise version of each dependency installed in the project directory or virtual environment by running pip freeze:

4.2. SETUP.CFG AND SETUP.PY

The requirements file is helpful in most cases for development. However, you might require more than simply this file if you intend to make your package/project widely accessible (for example, on PyPI). The setup.py and setup.cfg files resolve this issue. On top of distutils, setuptools is a module that enables the creation and distribution of Python packages. It also has features that enables easier management of dependencies in a project.

Setuptools provides the flexibility to precisely achieve this. When you wish to distribute a package, you often require some metadata, such as the package name, version, dependencies, entry points, etc. The following figures demonstrates SimiLabs' setup.py and setup.cfg files:

- If a package is mostly for development purposes but you aren't planning on redistributing it, requirements.txt should be enough (even when the package is developed on multiple machines).
- If a package is developed only by yourself (i.e. on a single machine) but you are planning to redistribute it, then setup.py/setup.cfg should be enough.
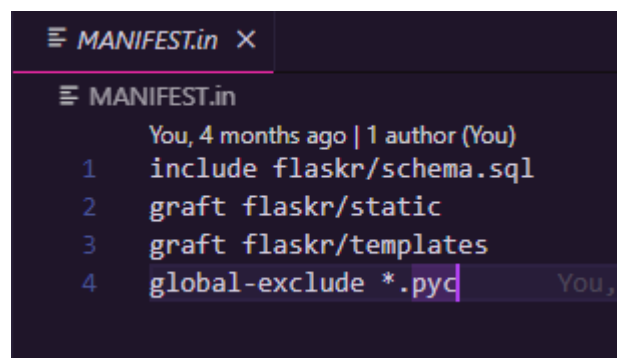
If a package is developed on multiple machines and you also need to redistribute it, you will need both the requirements.txt and setup.py/setup.cfg files.

4.3. MANIFEST.IN
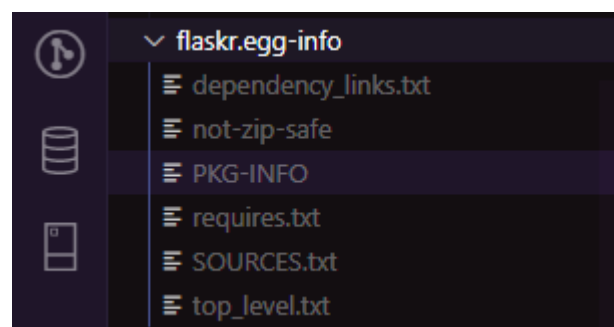
Only a small number of files are provided by default when creating a source distribution for your package. An authors/contributors file, a docs/ directory, or a directory containing data files used for testing may be extra files you decide to add to the source distribution. If your setup.py computes your project's long description by reading from both a README and a changelog file, you'll need to include both of those files in the sdist so that developers who build or install from the sdist get the correct result.

Writing a MANIFEST.in file at the project root allows you to add and remove files from the source distribution. After adding the above files to the sdist, the commands in MANIFEST.in (if such a file exists) are executed in order to add and remove further files to and from the sdist. Default files can even be removed from the sdist with the appropriate MANIFEST.in command.

After processing the MANIFEST.in file, setuptools removes the build/ directory as well as any directories named RCS, CVS, or .svn from the sdist, and it adds a PKG-INFO file and an *.egg-info directory to the root of a project. This behavior cannot be changed with a MANIFEST.in file.

4.4. LICENCE

- Our GitHub repository makes use of a MIT license
- The MIT license gives users express permission to reuse code for any purpose, sometimes even if code is part of proprietary software

4.5. FLASKR

4.5.1. GENERATEPDF

- In this folder, two PDF files will be stored: "Test" and "Test2"
- These documents are created when the user uploads the alleged document and the comparison document. It is necessary for the application to save the two PDFs, in order to extract the text in the 'compare()' function in "__Init__.py" and compare the text for plagiarism.

4.5.2. GENERATEREPORT

- In this folder, three PDF files are stored:
    1. PlagiarismReport.pdf
    2. PlagiarismReportExtensive.pdf
    3. StyloReport.pdf
- There reports are generated when the "Save" button is pressed in either the quickReport.html, extensiveReport.html or styloReport.html.
    - o The PlagiarismReport.pdf (for QuickText) is generated with the "WriteToPDF()" function inside "__Init__.py".

- The PlagiarismReportExtensive.pdf (for ExtensiveText) is generated with the "WriteToPDFExtensive()" function inside "__Init__.py".
- The StyloReport.pdf (for Stylometry) is generated with the "WriteToPDFStylo()" function inside "createPDFStylo.py".

4.5.3. GENSIMDATA

The GENSIMDATA folder is utilised to store all the results generated by the usage of genism in the extensive text comparison section of the application. An example of the file structure is as follow:

- GENSIMDATA:
  - 31594883
    - Corpus file
    - Data
  - 12345678
    - Corpus file
    - Data

When the user creates a new student by providing a student number, the above files will be created. The uploaded document supplied by the user, during the student creation or append function, will be converted and stored in a corpus text file which will be stored in the corpus file folder of the specified student. The name of the uploaded document will be stored in the Data folder in a document names text file for metadata purposes.

When the user wants to compare a corpus to an uploaded file, the specified student corpus text file is extracted for processing and a gensim dictionary is created along with a gensim corpus. The respective files are stored in the Data folder of the respective students along with an index for the gensim corpus to increase performance.

4.5.4. GENSIMTEMP

The GENSIMTEMP folder is used to temporarily store the documents uploaded by the user before it is processed by the various extensive text comparison functionality.

4.5.5. STATIC

4.5.5.1. IMAGES

Within this folder, there are three subfolders, and one lose image.

1. Icons
   - favicon.ico
     - This icon is used in authLogin.html and authregister.html

2. ReportImages
   - author_probability.png
     - This image is the image that displays on the StyloReport page and the generated PDF.
     - It shows the most likely author of the submitted document.

   - calibration.png
     - See the "**get_calibration_curve function:**
     - This image is the image that displays on the StyloReport page and the generated PDF.
     - It shows the relationship between the author probability and the burrows delta.

   - delta_score.png
     - See the **calculate_burrows_delta function:**
     - This image is the image that displays on the StyloReport page and the generated PDF.
     - It shows the burrows delta of all the documents.

- operating_curve.png
  - This image shows the relationship between the true positive rate and the false positive rate and displays a ROC curve that indicates how accurate the model is.
  - This image is the image that displays on the StyloReport page and the generated PDF.

- plot.png
  - This is an image of the plotted data. It shows all the documents and its authors plotted on a graph.
  - This graph is used to determine whether the author wrote all their submitted documents.
  - This image is displayed on the StyloReport page and the generated PDF

3. WordCloud
   - This folder contains two wordcloud images. These images are generated inside the 'compare()' function in "__Init__.py".
   - The images are generated using the two documents the user uploads in "QuickText".
   - The bigger the words are, the more they appeared in the document.

4. Lose image
   - SimiLabs.png
     - This is the logo of the application. It is displayed on the login screen.

4.5.5.2. JS

The JS directory contains the following JavaScript files:

- DisableCheckbox.js: disables the checkbox on the Extensive Text Form unless the Compare Corpus radiobutton is checked.
- DisabledText.js: disables the substring textbox on the Quick Text Form unless the substring radiobutton is checked.
- Footer.js: contains some JS styling for the footer on the pages
- Main.js: contains some JS validation for the login and register pages
- Materialize.js: contains some JS animations for the login and register pages, such as the SimiLabs Logo hover animation.

4.5.5.3. PDF

- This folder contains two PDFs
    - SimiLabs_User_Manual.pdf
        - This manual is for the day-to-day user to get a better understanding of how to navigate the application.
    - SimiLabs_Developer_Manual.pdf
        - This manual is for the developers and the project managers to get a better understanding of how the application works and how to navigate the file explorer, technologies, and code.

4.5.5.4. VENDOR

The Vendor directory contains additional CSS, jQuery and JS files which provides the animations and some styling for the Login and other HTML pages.

- Animate: a folder that contains a CSS file used for animations of the SimiLabs project
- Bootstrap: a folder that contains other CSS and JS files for HTML content templating such as styling and animations
- CSS-hamburgers provides CSS files for a hamburger menu animation
- jQuery: a folder that contains a jQuery script, complementing the files from the Bootstrap folder.
- select2: contains CSS and JS files that specifies styling and animations for HTML objects when selecting or clicking on them.
- tilt: provides a jQuery script for tilting animations for HTML objects on a page

4.5.5.5. CSS FILES

- This is the CSS files to style the HTML documents and make it more user friendly and appealing. You can see where these CSS files is used by looking at the top of the HTML files to see which files are imported.
- These are the following CSS files that can be found here.
    - baseReport.css
    - extensiveFooter.css
    - home.css
    - main.css
    - style.css
    - styloReport.css
    - text.css
    - tooltip.css
    - util.css

4.5.6. TEMPLATES

4.5.6.1. AUTH

- This folder contains two HTML files.
    - authLogin.html
        - It is divided into three parts: head, body and the scripts

The head contains all the links to the css files used to style the HTML

```html
<head>
    <title>Login - SimiLabs</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!--=============================================================-->
    <link rel="icon" type="image/png" href="../../static/images/icons/favicon.ico"/>
    <!--=============================================================-->
    <link rel="stylesheet" type="text/css" href="../../static/vendor/bootstrap/css/bootstrap.min.css">
    <!--=============================================================-->
    <link rel="stylesheet" type="text/css" href="../../static/fonts/font-awesome-4.7.0/css/font-awesome.min.css">
    <!--=============================================================-->
    <link rel="stylesheet" type="text/css" href="../../static/vendor/animate/animate.css">
    <!--=============================================================-->
    <link rel="stylesheet" type="text/css" href="../../static/vendor/css-hamburgers/hamburgers.min.css">
    <!--=============================================================-->
    <link rel="stylesheet" type="text/css" href="../../static/vendor/select2/select2.min.css">
    <!--=============================================================-->
    <link rel="stylesheet" type="text/css" href="../../static/util.css">
    <link rel="stylesheet" type="text/css" href="../../static/main.css">
    <!--=============================================================-->
</head>
```

The body is the physical body using HTML, that consists of divs and forms

```html
<div class="limiter">
    <div class="container-login100">
        <div class="wrap-login100">
            <div class="login100-pic js-tilt" data-tilt>
                <img src="../../static/images/SimiLabs.png" alt="IMG" style="border-radius: 100px solid black;">
            </div>

            <form class="login100-form validate-form" method="post">
                <span class="login100-form-title">
                    Member Login
                </span>

                <div class="wrap-input100 validate-input" data-validate = "Valid username is required">
                    <input class="input100" type="text" name="username" id="username" placeholder="Username">
                    <span class="focus-input100"></span>
                    <span class="symbol-input100">
                        <i class="fa fa-envelope" aria-hidden="true"></i>
                    </span>
                    <div class="msg"  style="color: red;text-align: center;">{{ msg }}</div>
                </div>

                <div class="wrap-input100 validate-input" data-validate = "Password is required">
                    <input class="input100" type="password" name="password" id="password" placeholder="Password">
                    <span class="focus-input100"></span>
                    <span class="symbol-input100">
                        <i class="fa fa-lock" aria-hidden="true"></i>
                    </span>
                    <div class="msg" style="color: red;text-align: center;">{{ msg }}</div>
                </div>


                <div class="container-login100-form-btn">
                    <button class="login100-form-btn">
                        Login
                    </button>
                </div>

                <div class="text-center p-t-136">
                    <a class="txt2" href="/auth/authRegister">
                        Create your Account
                        <i class="fa fa-long-arrow-right m-1-5" aria-hidden="true"></i>
                    </a>
                </div>
            </form>
        </div>
    </div>
</div>
```

The last part is at the bottom om the body, it is all the scripts that is needed for the HTML page to function and ensure safety.

```html
<!--===============================================================-->
    <script src="../../static/vendor/jquery/jquery-3.2.1.min.js"></script>
<!--===============================================================-->
    <script src="../../static/vendor/bootstrap/js/popper.js"></script>
    <script src="../../static/vendor/bootstrap/js/bootstrap.min.js"></script>
<!--===============================================================-->
    <script src="../../static/vendor/select2/select2.min.js"></script>
<!--===============================================================-->
    <script src="../../static/vendor/tilt/tilt.jquery.min.js"></script>
    <script >
        $('.js-tilt').tilt({
            scale: 1.1
        })
    </script>
<!--===============================================================-->
    <script src="../../static/js/main.js"></script>
```

- o authRegister.html
  - ▪ It is divided into three parts: head, body and the scripts

The head contains all the links to the css files used to style the HTML

```html
<head>
    <title>Register - SimiLabs</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
<!--===============================================================================================-->
    <link rel="icon" type="image/png" href="../../static/images/icons/favicon.ico"/>
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="../../static/vendor/bootstrap/css/bootstrap.min.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="../../static/fonts/font-awesome-4.7.0/css/font-awesome.min.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="../../static/vendor/animate/animate.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="../../static/vendor/css-hamburgers/hamburgers.min.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="../../static/vendor/select2/select2.min.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="../../static/util.css">
    <link rel="stylesheet" type="text/css" href="../../static/main.css">
<!--===============================================================================================-->
</head>
```

The body is the physical body using HTML, that consists of divs and forms

```html
<div class="limiter">
    <div class="container-login100">
        <div class="wrap-login100">
            <div class="login100-pic js-tilt" data-tilt>
                <img src="../../static/images/SimiLabs.png" alt="IMG" style="border-radius: 100px solid black;">
            </div>

            <form class="login100-form validate-form" method="post">
                <span class="login100-form-title">
                    Member Register
                </span>

                <div class="wrap-input100 validate-input" data-validate = "Valid username is required">
                    <input class="input100" type="text" name="username" id="username" placeholder="Username">
                    <span class="focus-input100"></span>
                    <span class="symbol-input100">
                        <i class="fa fa-envelope" aria-hidden="true"></i>
                    </span>
                    <div class="msg" style="color: red;text-align: center;">{{ msg }}</div>
                </div>

                <div class="wrap-input100 validate-input" data-validate = "Password is required">
                    <input class="input100" type="password" name="password" id="password" placeholder="Password">
                    <span class="focus-input100"></span>
                    <span class="symbol-input100">
                        <i class="fa fa-lock" aria-hidden="true"></i>
                    </span>
                    <div class="msg" style="color: red;text-align: center;">{{ msg }}</div>
                </div>

                <div class="wrap-input100 validate-input" data-validate = "Confirmation password is required">
                    <input class="input100" type="password" name="password2" id="password2" placeholder="Confirm Password">
                    <span class="focus-input100"></span>
                    <span class="symbol-input100">
                        <i class="fa fa-lock" aria-hidden="true"></i>
                    </span>
                    <div class="msg" style="color: red;text-align: center;">{{ msg }}</div>
                </div>

                <div class="container-login100-form-btn">
                    <button class="login100-form-btn">
                        Register
                    </button>
                </div>

                <div class="text-center p-t-136">
                    <a class="txt2" href="/auth/authLogin">
                        Login
                        <i class="fa fa-long-arrow-right m-l-5" aria-hidden="true"></i>
                    </a>
                </div>
            </form>
        </div>
    </div>
</div>
```

The last part is at the bottom om the body, it is all the scripts that is needed for the HTML page to function and ensure safety.

```html
<!--==========================================================-->
    <script src="../../static/vendor/jquery/jquery-3.2.1.min.js"></script>
<!--==========================================================-->
    <script src="../../static/vendor/bootstrap/js/popper.js"></script>
    <script src="../../static/vendor/bootstrap/js/bootstrap.min.js"></script>
<!--==========================================================-->
    <script src="../../static/vendor/select2/select2.min.js"></script>
<!--==========================================================-->
    <script src="../../static/vendor/tilt/tilt.jquery.min.js"></script>
    <script >
        $('.js-tilt').tilt({
            scale: 1.1
        })
    </script>
<!--==========================================================-->
    <script src="../../static/js/main.js"></script>
```

4.5.6.2. EXCEPTIONS

File: errors.html

HTML file for exception handling. Displays "Something went wrong" with the error message. The user will be able to navigate back to the home page with a hyperlink.

4.5.6.3. HELP

File: help.html

- It is divided into three parts: head, body and the scripts.

The head contains all the links to the CSS files used to style the HTML.

```html
<title>{% block title %} Help {% endblock %}</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lato">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="/static/home.css">
<script type="text/javascript" src="../../static/js/pdfobject.js"></script>
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
```

The body consists of styling.

```html
<style>
body {font-family: "Lato", sans-serif}
.mySlides {display: none}
</style>
```

48

The body is the physical body using HTML, that consists of divs and forms

```
{% block main %}

<br>
<br>
<br>

<h1> User Manual </h1>
<style>
    .container{padding: 30px}
</style>
<div class="container" style="border: 2px solid □black;background-color: ■rgb(197, 202, 200);">
    <embed src="../../static/pdf/SimiLabs_User_Manual.pdf" type="application/pdf" width="100%" height="700px">
</div>

<h1> Technical Manual </h1>
<style>
    .container{padding: 30px}
</style>
<div class="container" style="border: 2px solid □black;background-color: ■rgb(197, 202, 200);">
    <embed src="../../static/pdf/SimiLabs_User_Manual.pdf" type="application/pdf" width="100%" height="700px">
</div>

<div style="margin-bottom: 3em;"></div>

<script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script type="text/javascript" src="../../static/js/materialize.js"></script>
<script type="text/javascript" src="../../static/js/custom.js"></script>


{% endblock %}
```

4.5.6.4.   HOME

File: home.html

- The code that styles the home page.

- Consists of three parts, header, styling, and content.

The head contains all the links to the CSS files used to style the HTML.

```
<title>{% block title %} Home {% endblock %}</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lato">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="../../static/home.css">
<link rel="stylesheet" href="../../static/text.css">
<link rel="stylesheet" href="../../static/extensiveFooter.css">

<link rel="stylesheet" href="../../static/tooltip.css">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
<style>
```

The body consists of styling.

```css
body {font-family: "Lato", sans-serif}
.mySlides {display: none}

ol li {
    font-weight:700;
}

ol li dt dl span{
    font-weight: normal;
}
```

The Content of the page consists of divs.

```html
<!-- Page content -->
<br>
<br>
<br>
<h1> SimiLabs 2022</h1>
<br>

<div style="margin: auto;display:flex;background-color: rgb(197, 202, 200);width: 35%;border: 3px solid black;padding: 2em 2em 0em 2em;">
    <ol>
        <li> <span> Text Compare </span></li>
        <dt>- QuickText</dt>
        <dl>&nbsp &nbsp &nbsp - Compare alledged document and a comparison document for text similarities</dls>
        <dt>- ExtensiveText</dt>
        <dl>&nbsp &nbsp &nbsp - Compare alledged document to a corpus for text similarities</dl>

        <li>  <b> Stylometry </b></li>
        <dl>&nbsp &nbsp &nbsp - Compare alledged document to a corpus to identify authorship </dl>

        <li> <b> Reports </b></li>
        <dl>&nbsp &nbsp &nbsp - Display the database report results (Next Release)</dl>

        <li> <b> MORE </b></li>
        <dt>- Help</dt>
        <dl>  &nbsp &nbsp &nbsp - User Manual and the Technical Manual</dl>
        <dt>- Log Out</dt>
        <dl>  &nbsp &nbsp &nbsp - Log out of SimiLabs</dl>
    </ol>
</div>
<br>


<div style="margin-bottom: 6em;"></div>
<!-- End Page Content -->
```

4.5.6.5.   REPORTS

Reports are generated when the user chooses to save the results of the various analyses. 4 HTML files for the different reports:

- error.html: displays when there was an error creating the report.

```html
<div class="col">
    <div class="jumbotron">
        <h1 class="display-3">{{ error.code }}</h1>
        <p class="lead">{{ error.name }}</p>
```

```
        <hr class="my-4"/>
        <p>{{ error.description }}</p>
    </div>
</div>
```

- extensiveReport.html: creates a report for the extensive text comparison results.

```
{% extends 'baseReport.html' %}
{% block header %}
<title>{% block title %} Extensive Report {% endblock %}</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lato">
<link    rel="stylesheet"    href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="/static/home.css">
<style>
body {font-family: "Lato", sans-serif}
.mySlides {display: none}
</style>
{% endblock %}
{% block main %}
<!-- Page content -->
<br>
<br>
<br>
<h1> Extensive Text Report</h1>
<p> Choose a file to continue working on:</p>
<div style="display: flex;justify-content: center;align-items: center;width:
80%;margin-left: 10em;border: 3px solid black;padding: 2em 2em 2em 2em;">
    <ul>
    {% for item in theList %}
        <li>{{ item }}</li>
    {% endfor %}
    </ul>
    <!-- <p>{{theList}}</p> -->
```

```
</div>
<div style="justify-content: center;align-items: center;width: 80%;margin-left:
10em;border: 3px solid black;padding: 0em 2em 2em 2em;">
    <h2 style="text-align: center;">
        Description of results
    </h2>
    <p style="font-weight: 200 ">
        The application provides an LSI score and TD-IDF Score for each document
within the specified student corpus.<br>
         Both LSI score and TD-IDF scores are used to determine the similarity
between the documents. <br>
          The LSI calculation is based on the Latent Semantic Indexing (LSI)
algorithm and should be used for smaller size documents (Documents containing
less than 3 pages).<br>
        The TD-IDF calculation is based on the Term Frequency-Inverse Document
Frequency (TD-IDF) algorithm and should be used for larger size documents
(Documents containing more than 3 pages).<br>
        The scores range from -1 to 1 where -1 indicates that the documents are
completely different and 1 indicates that the documents are completely
similar.<br>
        The LSI calculation method focuses on the individual words in the corpus
whereas TD-IDF focuses on sentences and words.<br>
    </p>
</div>
<!-- End Page Content -->
{% endblock %}
```

- quickReport: creates a report for the quick text comparison results.

```
{% extends 'baseReport.html' %}
{% block header %}
<title>{% block title %} Quick Report {% endblock %}</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lato">
<link    rel="stylesheet"    href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
```

```html
<link                    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta/css/bootstrap.min.css" rel="stylesheet"/>
<!--Table css-->
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<!-- __ -->
<link rel="stylesheet" href="/static/home.css">
<link rel="stylesheet" href="../../static/tooltip.css">
<link                                                          rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css"
>
<style>
body {font-family: "Lato", sans-serif}
.mySlides {display: none}
.container {
    display: flex;
    /* flex-wrap: wrap; */
    justify-content: center;
    align-items: center;
  }
  img {
    width: 500px;
    height: 250px;
    margin: 10px;
  }
</style>
{% endblock %}
{% block main %}
<!-- Page content -->
<br>
<br>
<br>
<h1> Quick Text Report</h1>
<br>
<br>
<div class = "container" style="display: flex;justify-content: center;align-
items: center;">
```

53

```html
   <span  style="font-weight:  700;font-size:1.5em;margin-right:5em;  border:
none;" data-toggle="tooltip" title="{{JaccardTooltip}}"> Percentage similarity
Jaccard: <p style="color: {{colorJac}};">{{ similarityJac }}%</p> </span>
   <span  style="font-weight:  700;font-size:1.5em;float:right;  border:  none;"
data-toggle="tooltip"  title="{{CosineTooltip}}">Percentage  similarity  Cosine:
<p style="color: {{colorCos}};">{{ similarityCos }}%</p> </span>
</div>
<div class="w3-container">
  <table class="w3-table-all">
     <tr>
       <th width: 100px>
         Creator / Original Author
       </th>
       <th>
         Date Created
       </th>
       <th>
         Last Modified Author
       </th>
       <th>
         Date Modified
       </th>
     </tr>
     <tr class="{{rowColor}}">
       <td>{{ metadata[0] }}</td>
       <td>{{ metadata[1] }}</td>
       <td>{{ metadata[2] }}</td>
       <td>{{ metadata[3] }}</td>
     </tr>
   </table>
</div>
<br>
<div>
  <div>
    <h2 style=" float: left;margin-left: 8em"> Suspected Document  </h1>
  </div>
  <div>
```

```html
        <h2 style=" float: right; margin-right: 8em"> Comparison Document  </h1>
    </div>
</div>
<br>
<br>
<div style="display:flex;justify-content: center;align-items: center;">
    <p style="margin-right: 13em"> Wordcount: &nbsp &nbsp {{file1Length}} <p>
    <p style="margin-left: 30em;"> Wordcount: &nbsp &nbsp {{file2Length}} </p>
</div>
<!-- Suspect Doc -->
<div class="container-fluid">
        <div class="col-6" style="border: 3px solid black; margin: 5px 10px 5px 10px;float: left;background-color: white">
            <pre>{{ file1 | safe }}</pre>
        </div>
        <!-- Report -->
        <div class="col-6" style="border: 3px solid black; margin: 5px 10px 5px 10px; float: right;background-color: white">
            <pre>{{ file2 | safe }}</pre>
        </div>
</div>
<br>
<br>
<div class="container" style="border: 3px solid black;margin: 5px 200px 5px 180px; background-color: white;padding: 2em 2em 2em 2em;">
    <div>
        <h2 style="float: right;margin-right: 5em;font-weight: 700"> Suspect WordCloud </h2>
        <img src="../../static/images/WordCloud/wordcloud.png" alt="WordCloud.png">
    </div>
    <div>
        <h2 style="float: left;margin-left: 3em;font-weight: 700"> Comparison WordCloud </h2>
                <img        src="../../static/images/WordCloud/wordcloud2.png" alt="WordCloud2.png">
    </div>
</div>
```

55

```html
<br>
<br>
<script
src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.slim.min.js"></scri
pt>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></s
cript>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min
.js"></script>
<script>
  $(document).ready(function(){
    $('[data-toggle="tooltip"]').tooltip();
  });
</script>
<br>
<h2  style="float:  left;margin-left:  20em;font-weight:  700"> Words  Used
Comparison </h2>
<br>
<div class="container-fluid" style="height: 60vh;">
  <div  class="col-6"  style="border:  3px  solid  black;  margin:  5px  5px  5px
5px;float: left;background-color: white">
    <pre>{{ Test1 | safe }}</pre>
  </div>
  <!-- Report -->
  <div class="col-6" style="border: 3px solid black; margin: 5px 5px 5px 5px;
float: right;background-color: white">
    <pre>{{ Test2 | safe }}</pre>
  </div>
</div>
<!-- End Page Content -->
{% endblock %}
```

- styloReport.html: creates a report for the stylometry results.

```html
{% extends 'baseReport.html' %}
{% block header %}
```

```html
<title>{% block title %} Stylometry Report {% endblock %}</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lato">
<link   rel="stylesheet"   href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="/static/home.css">
<link rel="stylesheet" href="../../static/styloReport.css">
<link rel="stylesheet" href="../../static/tooltip.css">
<style>
body {font-family: "Lato", sans-serif;overflow-y: scroll;}
.mySlides {display: none}
table, th, td {
  border: 3px double black;
  border-collapse: collapse;
  padding: 15px;
}
</style>
{% endblock %}
{% block main %}
<!-- Page content -->
<br>
<br>
<br>
<h1> Stylometry Report</h1>
<div class = "container" style = "margin-bottom: 1em;margin: 5px 50px 10px
50px;">
  <table>
    <tr>
        <td style="font-weight: 700;">Burrow's Delta Score for Source
Document</th>
            <td><a   href="../../static/images/ReportImages/delta_score.png"
target="_blank">Click Here</a></td>
    </tr>
    <tr>
      <td style="font-weight: 700;">Author Probability</td>
```

```html
        <td><a  href="../../static/images/ReportImages/author_probability.png"
target="_blank">Click Here</a></td>
    </tr>
  </table>
</div>
<br>
<div class="container" style="border: 3px solid black;margin: 5px 50px 10px
50px;">
  <div>
    <span><p>ROC (Receiver Operating Characteristic) Curve</p></span>
          <img     src="../../static/images/ReportImages/operating_curve.png"
alt="operating_curve.png">
    <p>The Receiver Operator Characteristic (ROC) curve is a graphical plot
that illustrates the </p>
    <p>diagnostic ability of a binary classifier system as its discrimination
threshold is varied.</p>
    <p>Classifiers that give curves closer to the top-left corner indicate a
better performance. </p>
    <p>The closer the curve is to the 45 degree diagonal of the ROC space, the
less accurate the test is.</p>
    <p>An AUC score of 0.5 means that a classifier is performing badly, and a
1.0 score is a perfect score. </p>
    <p>The AUC score is on the bottom right corner of the graph.</p>
  </div>
  <div>
    <span><p>PCA-Derived Coordinates</p></span>
    <img src="../../static/images/ReportImages/plot.png" alt="plot.png">
    <p>The Principal Component Analysis (PCA) curve is a cluster of all the
authors and their documents.</p>
    <p>It is used to visualise the sylistic similarities between the documents
in the corpus. </p>
    <p>The closer the documents are to each other the more likely they are
written by the same author.</p>
  </div>
</div>
<!-- End Page Content -->
{% endblock %}
```

4.5.6.6. STYLOMETRY

HTML file for displaying the stylometry page.

```
{% extends 'baseNav.html' %}
{% block header %}
<title>{% block title %} Stylo {% endblock %}</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lato">
<link    rel="stylesheet"    href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="../../static/text.css">
<style>
body {font-family: "Lato", sans-serif}
.mySlides {display: none}
</style>
{% endblock %}
{% block main %}
<!-- Page content -->
<br>
<br>
<br>
<h1> Stylometry</h1>
<!-- <p> Choose a file to continue working on:</p> -->
<div class="form-center">
    <form method="post" action="/stylometry" enctype="multipart/form-data">
      <dl>
        <p>
          <p>Enter Student Number</p>
            <input  type="text"  name="studentname"  id="studentname"  required
autocomplete="off" class="input" style="margin-right: 190px;">
          <p>Upload Source Document</p>
              <input   type="file"   name="file1"   required   autocomplete="off"
class="input">
          <p>Upload Additional Document to Corpus (Optional) </p>
          <input type="file" name="comparison" autocomplete="off" class="input">
        </p>
```

```
        </dl>
        <p>
          <input type="submit" value="Submit">
        </p>
      </form>
    </div>
    <div style="margin-bottom: 13.5vh;"></div>
<!-- End Page Content -->
{% endblock %}
```

4.5.6.7.   TEXT

4.5.6.7.1.   QuickText HTML

- This HTML can be divided into two parts, the header, and the body.
    - The head contains all the links to the CSS files used to style the HTML

```
<title>{% block title %} Quick {% endblock %}</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lato">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="/static/text.css">

<style>
body{font-family: "Lato", sans-serif; height: 100%; margin: 0;}
.mySlides {display: none}
</style>
```

    - The body is the physical body using HTML, that consists of divs and forms that structures the HTML page

```
<!-- Page content -->
<br>
<br>
<br>

<h1> Quick Text Compare </h1>

<div class="form-center">
  <form method="post" action="/QuickText" enctype="multipart/form-data">
    <dl>
        <p>
            <p>Upload Source Document</p>
            <input type="file" name="file1" autocomplete="off" required class="input">
            <p>Upload Comparison Document</p>
            <input type="file" name="file2" autocomplete="off" required class="input">
        </p>
    </dl>

    <!-- Script to disable substring textbox unless the radiobutton is checked-->
    <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
    <script type="text/javascript" src="../../static/js/disabledtext.js"></script>

    <h5>Algorithm</h5>
            <input type="radio" name="algorithm" value="lines" checked> Lines<br>
            <input type="radio" name="algorithm" value="sentences"> Sentences<br>
            <input type="radio" name="algorithm" value="substrings" id="chkSubstring"> Substrings<br>
            <br>
            <h5>Length (for Substrings algorithm)</h5>
            <input type="text" name="length" id="txtSubstring" disabled="true">
            <br><br>
        <p>
            <input type="submit" value="Submit">
        </p>
</form>
</div>
</form>

<div style="margin-bottom: 5em;"></div>


<!-- End Page Content -->
```

4.5.6.7.2.   ExtensiveText.HTML

- This HTML can be divided into three parts, the header and the body and the scripts.

  - The head contains all the links to the css files used to style the HTML

```
<title>{% block title %} Extensive {% endblock %}</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lato">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="/static/text.css">
<link rel="stylesheet" href="/static/extensiveFooter.css">

<link rel="stylesheet" href="../../static/tooltip.css">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">


<style>
body {font-family: "Lato", sans-serif;}
.mySlides {display: none}
</style>
```

- o The body is the physical body using HTML, that consists of divs and forms that structures the HTML page

```html
<h1> Extensive Text Compare </h1>
<div class="form-center">
  <form method="post" action="/DeleteStudent" enctype="multipart/form-data">
    <h5> <u> Feedback </u> </h5>
    <p> {{ExtensiveFeedback}} </p>
  </form>
</div>

<div class="form-center">
  <form method="post" action="/ExtensiveText" enctype="multipart/form-data">
    <h1> <u> Manage Student Corpus </u> </h1>
    <h3> Student Number: </h3> <input type="text" id="stdNum" name="stdNum" placeholder="12345678" required>
    <br>
    <p>Upload Document</p>
    <input type="file" name="ExtensiveStudent" autocomplete="off" required class="input">
    <br>
    <br>
    <input type="radio" name="ExtensiveText" value="CreateStudent" checked> Create Student<br>
    <input type="radio" name="ExtensiveText" value="UpdateStudent"> Update Student<br>
    <input type="radio" name="ExtensiveText" value="CompareCorpus" id="ischecked"> Compare Corpus <br>
    <br>
    <input type="checkbox" id="extensiveBox" name="extensiveBox" value="extensiveBox" disabled>
    <label for="extensiveBox" data-toggle="tooltip" title="Check the box to add the document to the corpus."> Add to corpus</label><br>
    <br>
    <p>
      <input type="submit" value="Submit">
    </p>

  </form>
</div>

<div class="form-center">
  <form method="post" action="/DeleteStudent" enctype="multipart/form-data">
    <h1> <u> Delete Student</u> </h1>
    <h3> Student Number: </h3> <input type="text" id="dltStd" name="dltStd" placeholder="12345678" required>
    <br>
    <br>
    <input type="submit" value="Delete">
    <br>
    <br>

  </form>
</div>
```

- o All the scripts that are needed for the HTML page to function.

```html
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
<script type="text/javascript" src="../../static/js/disablecheckbox.js"></script>
<script src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"></script>
<script>
$(document).ready(function(){
  $('[data-toggle="tooltip"]').tooltip();
});
</script>
```

4.5.6.8. HTML INHERITANCE FILES

- Each page in the application will have the same basic layout around a different body.

- Instead of writing the entire HTML structure in each template, each template will extend a base template and override specific sections.

- In the application there are three base templates:
  - baseNav.html
    - The home screens of the different functionalities use this baseNav.
  - baseNavHelp.html
    - The help page uses this baseNavHelp.
  - baseReport.html
    - All the reports use this baseReport.

4.5.6.9. WORDCOUNT

- This folder contains two text files called: count.txt and count2.txt

- These text files contain words and the number of times they appear in a document that is called in "countwords.py"


```
1    the : 636
2    and : 540
3    to : 306
4    a : 282
5    of : 236
6    project : 220
7    in : 216
8    is : 213
9    that : 170
10   management : 130
```

4.5.7. INIT.PY

Help

The help method is used to provide the user with the user manual. Exception handling is incorporated to show an appropriate response to the error that occurred. This method gets the path to the HELP_PDF and return the pdf with the user manual on screen.

```
@app.route('/help')
def help():
    try:
        path = os.getenv('HELP_PDF')
        a = os.listdir(path)
        print(a)
        text = json.dumps(sorted(a))
        return render_template("help/help.html", contents = text)
    except Exception as e:
        return render_template("exceptions/errors.html", error = e)
```

Login:

The user enters his/her username and password into the fields. The username and password are then looked up in the mysql database from the Account table. If the username and password match to an existing username and corresponding password in the Account table, the user has successfully logged in. If the username doesn't exist in the table, the user will be informed that his/her credentials are invalid and needs to click on the register button to be redirected to the register form where the user can register. Once the user has successfully logged in, he/she will be redirected to the HOME page.

```
@app.route('/auth/authLogin',methods =['GET', 'POST'])
def authLogin():
    try:
        session.clear()
        msg = ''
        if request.method == 'POST' and 'username' in request.form and 'password' in request.form:
            username = request.form['username']
            password = request.form['password']
            cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
            cursor.execute('SELECT * FROM accounts WHERE username = %s', ([username]))
            account = cursor.fetchone()
            if account is None:
                msg = 'Ivalid credentials!'
            elif not check_password_hash(account['password'], password):
                msg = 'Invalid credentials!'
            if msg == '':
                session['loggedin'] = True
                session['id'] = account['id']
                session['username'] = account['username']
                msg = 'Logged in successfully!'
                return render_template('/home/home.html', msg = msg)
        return render_template('auth/authLogin.html', msg=msg)
    except Exception as e:
        return render_template("exceptions/errors.html", error = e)
```

Register:

The user enters his/her username and password into the fields. The username and password are then looked up in the mysql database from the Account table. If the username and password match to an existing username and corresponding password in the Account table, the user will be informed that his/her account already exists. If the two passwords don't match or if one of the fields are empty the user will be informed to fix his/her mistake(s). If the user doesn't exist, the username and password is inserted into the Account table. Once the users have successfully registered, he/she is redirected to the HOME page.

```python
@app.route('/auth/authRegister', methods=('GET', 'POST'))
def register():
    try:
        session.clear()
        msg = ''
        if request.method == 'POST' and 'username' in request.form and 'password' in request.form and 'password2' in request.form:
            username = request.form['username']
            password = request.form['password']
            password2 = request.form['password2']
            cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
            cursor.execute('SELECT * FROM accounts WHERE username = % s', ([username]))
            account = cursor.fetchone()
            if account:
                msg = 'Account already exists!'
            elif password != password2:
                msg = 'Passwords do not match!'
            elif not username or not password or not password2:
                msg = 'Please fill out the form!'
            else:
                cursor.execute('INSERT INTO accounts VALUES (NULL,% s, % s)', ([username], [generate_password_hash(password)]))
                mysql.connection.commit()
                cursor.execute('SELECT * FROM accounts WHERE username = %s', ([username]))
                account = cursor.fetchone()
                session['loggedin'] = True
                session['id'] = account['id']
                session['username'] = account['username']
                msg = 'Logged in successfully!'
                return render_template('/home/home.html', msg = msg)
        elif request.method == 'POST':
            msg = 'Please fill out the form !'
        return render_template('auth/authRegister.html', msg=msg)
    except Exception as e:
        return render_template("exceptions/errors.html", error = e)
```

Logout:

If the user clicks on the logout button, their session will be cleared to protect the API endpoints. Once the user's session has been cleared, he/she will be redirected to the Login page.

```python
@app.route('/logout')
def logout():
    try:
        # protecting your api endpoints
        for key in list(session.keys()):
            if session.key == "session":
                session["session"].clear()
            session.pop(key)
        # response.delete_cookie('session')
        session.clear()
        print(f"Session: {session['session']}")
        return redirect(url_for('authLogin'))
    except Exception as e:
        return render_template("exceptions/errors.html", error = e)
```

65

QuickText

The QuickText function in "__Init__.py" will request two files that the user uploaded. The function will then check which algorithm the user selected, and then highlight the matching sentences or substrings according to the algorithms.

The two documents will also be processed through two text similarity algorithms: Jaccard and Cosine. The function saves the text similarity score calculated from the Jaccard and the Cosine similarity functions, and the color of the bottom table row of the quickReport page. The color depends on the score calculated in the two text similarity algorithms. It will also save the document metadata in a list.

The similarity scores, table row color and the metadata values will then be passed to the report page, and it will also be saved to a PDF report.

Two wordclouds will also be generated within this function using the WordCloud api.

The two wordcount functions (described in 4.5.8) will then be called to retrieve the number of instances of the words in the respective documents.

The highlight function will go through the text and highlight the similarities based on the type of algorithm that was previously selected.

WriteToPDF() and WriteToPDFExtensive

See 4.5.9

Download

```python
@app.route('/GenerateReport/<path:filename>', methods=['GET', 'POST'])
def download(filename):
    try:
        # Appending app path to upload folder path within app root folder
        uploads = os.path.join(app.config['UPLOAD_REPORT'])
        # Returning file from appended path
        return send_from_directory(uploads, filename, as_attachment=True)
    except Exception as e:
        return render_template("exceptions/errors.html", error = e)
```

This function downloads a specified PDF report, depending on the report page the user visited. The file is saved to the "UPLOAD_REPORT" directory. When the user clicks on the save button, the report is saved locally to the device.

Stylometry:

```python
@app.route('/stylometry', methods=['GET','POST'])
def stylo():
```

Function to store the uploaded documents to the corpus and return results of the analysis.

```python
if request.method == 'POST':
    folder = "data/test"
    for filename in os.listdir(folder):
        file_path = os.path.join(folder, filename)
        try:
            if os.path.isfile(file_path) or os.path.islink(file_path):
                os.unlink(file_path)
            elif os.path.isdir(file_path):
                shutil.rmtree(file_path)
        except Exception as e:
            print('Failed to delete %s. Reason: %s' % (file_path, e))
```

If the application sends a POST method the test data folder will be selected. The file path will be created by combining the folder path name and the filename to ensure to uploaded document is stored in the correct path.

```python
if not request.files["file1"]:
    abort(400, "missing file")
try:
    name = request.form["studentname"]
    print(name)
    doc1 = request.files["file1"]
    doc1.encoding = 'utf-8'

    doc2 = request.files["comparison"]
    doc2.encoding = 'utf-8'
    print("IsThisEmpty: " + str(doc2))
```

If the user did not upload a file abort the function. Get the name of the uploaded document and store it in the "name" variable. Store the suspected document in "doc1" and the comparison document in "doc2".

67

```python
if doc1.filename.endswith('.docx'):
    filenamedoc = secure_filename(doc1.filename)
    file1 = docx2txt.process(doc1)
elif doc1.filename.endswith('.pdf'):
    pdfReader = PyPDF2.PdfFileReader(doc1)
    count = pdfReader.numPages
    file1 = ""
    for i in range(count):
        page = pdfReader.getPage(i)
        file1 += page.extractText()
    filenamePDF = secure_filename(doc1.filename)
else:
    filenametxt = secure_filename(doc1.filename)
    doc1.seek(0)
    file1 = doc1.read().decode("utf-8")
```

Check to see what type of file the suspected document is. If is of the types docx or pdf it is converted to a text file.

```python
if(str(doc2) == "<FileStorage: '' ('application/octet-stream')>"):
    print("This is empty")
else:
    if doc2.filename.endswith('.docx'):
        filenamedoc = secure_filename(doc2.filename)
        file2 = docx2txt.process(doc2)
    elif doc2.filename.endswith('.pdf'):
        pdfReader = PyPDF2.PdfFileReader(doc2)
        count = pdfReader.numPages
        file2 = ""
        for i in range(count):
            page = pdfReader.getPage(i)
            file2 += page.extractText()
        filenamePDF = secure_filename(doc2.filename)
    else:
        filenametxt = secure_filename(doc2.filename)
        doc2.seek(0)
        file2 = doc2.read().decode("utf-8")
    train_file = open("data/train/"+name+"_-_"+os.path.splitext(doc2.filename)[0]+".txt", "w", encoding="utf-8")
    train_file.write(file2)
    train_file.close()
```

Check to see what type of file the comparison document is. If it is of types docx or pdf it is converted to a text file. The comparison document is stored in the training corpus.

```python
#open text file
text_file = open("data/test/"+name+"_-_"+os.path.splitext(doc1.filename)[0]+".txt", "w", encoding="utf-8")

#write string to file
text_file.write(file1)

#close file
text_file.close()

fastStyle()
WriteToPDFStylo(name)
print(request.path)

return render_template('reports/styloReport.html')
```

Open the suspected document, write the text into it and close it. Call the fastStyle method to start the stylometric analysis. Write the results into a PDF report using the student number that was entered. Redirect the user to the stylometry report page.

ExtensiveText

For the user to successfully utilise the extensive text comparison functionality, the application must utilise the student number and uploaded document specified in the front-end view of the application. The uploaded file must be saved in the GensimTemp folder in order for the application to process the document. The data will be passed to the backend and will be utilised in the Gensim.py file.

```python
# Upload Folder
@app.route('/ExtensiveText', methods=['GET','POST'])
def upload_folder():
    try:
        if 'loggedin' not in session:
                return render_template('auth/authLogin.html')
        if request.method == 'POST':
            # Clear the GensimTemp folder
            folder = "./flaskr/GensimTemp"
            for filename in os.listdir(folder):
                file_path = os.path.join(folder, filename)
                try:
                    if os.path.isfile(file_path) or os.path.islink(file_path):
                        os.unlink(file_path)
                    elif os.path.isdir(file_path):
                        shutil.rmtree(file_path)
                except Exception as e:
                    print('Failed to delete %s. Reason: %s' % (file_path, e))

            stdNum = request.form['stdNum']
            doc1 = request.files["ExtensiveStudent"]
            doc1.save(os.path.join("./flaskr/GensimTemp/", doc1.filename))

            checked = request.form.getlist('extensiveBox')
            boolean = ""
            if(checked == ["extensiveBox"]):
                boolean = "true"
                print(boolean)

            ExtensiveFeedback = ""

            if request.form.get("ExtensiveText") == "CreateStudent":
                print("Create Student")
                CreateStudent(stdNum, doc1.filename)
                ExtensiveFeedback = "Student " + str(stdNum) + " is created"
            elif request.form.get("ExtensiveText") == "UpdateStudent":
                print("Compare Student")
                UpdateStudent(stdNum, doc1.filename)
                ExtensiveFeedback = "Student " + str(stdNum) + " is updated"
            elif request.form.get("ExtensiveText") == "CompareCorpus":
                print("Compare Corpus")
                extensiveList = CompareCorpus(stdNum, doc1.filename, boolean)
                WriteToPDFExtensive(extensiveList,stdNum, doc1.filename)
                return render_template('reports/extensiveReport.html',theList=extensiveList)

            print(stdNum)

        return render_template('text/extensiveText.html', ExtensiveFeedback = ExtensiveFeedback)
```

```
except Exception as e:
    return render_template("exceptions/errors.html", error = e)
```

DeleteStudent

A user might want to remove data relating to a student number. The user will enter the student number on the front-end side. The student number will be passed to the backend where it will be utilised in the Gensim.py file.

```
@app.route('/DeleteStudent', methods=['POST'])
def deleteStudent():
    ExtensiveFeedback = ""
    try:
        if 'loggedin' not in session:
            return render_template('auth/authLogin.html')
        if request.method == 'POST':
            stdNum = request.form['dltStd']
            DeleteStudent(stdNum)
            ExtensiveFeedback = "Student " + str(stdNum) + " is deleted"
        return render_template('text/extensiveText.html', ExtensiveFeedback = ExtensiveFeedback)
    except Exception as e:
        return render_template("exceptions/errors.html", error = e)
```

4.5.8. COUNTWORDS.PY

- This python file consists of two functions: countWords1() and countWords2()
    - The function checks whether the path exists using an if statement, if it does, the count.txt file will be deleted, because a new one will be created after this if statement.
    - The function will also loop through the doc that the function receives as a parameter. If the word does not exist, it will be added to the array and the count of that word will be incremented by one. If the word is already in the array, it will increment the count by one.
    - After this is done, a text file in the WordCount folder, called count.txt and count2.txt, will be opened and the array of words and the number of appearances will be written to the text file.

```python
def countWords1(doc):
    if os.path.exists('./flaskr/WordCount/count.txt'):
        os.remove('./flaskr/WordCount/count.txt')
    strings = ""
    d = dict()
    stringlist = doc.split()
    for word in stringlist:
        word = word.strip(string.punctuation)
        word = word.lower()
        if word not in d:
            d[word] = 1
        else:
            d[word] += 1
        strings += word+","
    stop_words = (" ")
    for word in stringlist:
        #Dont count words that are stop words
        if word in stop_words:
            continue
        if word in d:
            d[word] = d[word] + 1
        else:
            d[word] = 1
    counter = 0
    for key in sorted(d, key=d.get, reverse=True):
        with open('./flaskr/WordCount/count.txt', 'a', encoding="utf8") as f:
            f.write(key + " : " + str(d[key]) + "\n")
            counter = counter + 1
            # if(counter == 100 or str(key) == ""):
            #     break
```

4.5.9. CREATEPDFSTYLO.PY

- The createPDFStylo python file creates a PDF which can be downloaded. The PDF report contains the Burrows delta score, the authorship score, the ROC curve, and the PCA curve. These statistics and graphs contain descriptions.

- Makes use of the FPDF, datetime, and the os imports.

- The WriteToPDFStylo function is called by a student's university number

```python
from fpdf import FPDF
import datetime
import os

def WriteToPDFStylo(student_number):
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", size = 15)

    burrows_delta = "The Burrows delta score is calculated from the z-scores of every common word in the vocabulary, which represents the fingerprint of the document and can be across multi
    authorship_score = "The authorship score is a probability corresponding to the delta values from the Burrows delta matrix. A high value indicates a high probability thatthe author in th
    ROC_curve = "The Receiver Operator Characteristic (ROC) curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold i
    PCA_curve = "The Principal Component Analysis (PCA) curve is a cluster of all the authors and their documents. It is used to visualise the sylistic similarities between the documents in

    pdf.multi_cell(200, 10, txt = "Student number that was investigated: " + student_number, align = 'L')
    pdf.multi_cell(200, 10, txt = "Date the report was created: " + str(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")), align = 'L')

    pdf.multi_cell(200, 10, txt = " ", align = 'C')
    pdf.ln(200)

    pdf.multi_cell(200, 10, txt = "Burrows delta score", align = 'C')
    pdf.multi_cell(200, 10, txt = "", align = 'L')
    pdf.image(f"{os.getenv('UPLOAD_IMG')}SimiLabs_2022/flaskr/static/images/ReportImages/delta_score.png", w = 500, h = 50)
    pdf.multi_cell(200, 10, txt = burrows_delta, align = 'L')

    pdf.multi_cell(200, 10, txt = " ", align = 'C')
    pdf.multi_cell(200, 10, txt = "Authorship score", align = 'L')
    pdf.image(f"{os.getenv('UPLOAD_IMG')}SimiLabs_2022/flaskr/static/images/ReportImages/author_probability.png", w = 500, h = 50)
    pdf.multi_cell(200, 10, txt = authorship_score, align = 'L')

    pdf.multi_cell(200, 10, txt = " ", align = 'C')
    pdf.multi_cell(200, 10, txt = "Receiver Operator Characteristic (ROC) curve", align = 'L')
    pdf.image(f"{os.getenv('UPLOAD_IMG')}SimiLabs_2022/flaskr/static/images/ReportImages/operating_curve.png", w = 200, h = 100)
    pdf.multi_cell(200, 10, txt = ROC_curve, align = 'L')

    pdf.multi_cell(200, 10, txt = " ", align = 'C')
    pdf.multi_cell(200, 10, txt = "Principal Component Analysis (PCA) curve", align = 'L')
    pdf.image(f"{os.getenv('UPLOAD_IMG')}SimiLabs_2022/flaskr/static/images/ReportImages/plot.png", w = 200, h = 100)
    pdf.multi_cell(200, 10, txt = PCA_curve, align = 'L')

    pdf.output(dest="F", name=os.getenv('UPLOAD_REPORT') + "styloReport.pdf")
```

- pdf = FPDF() creates a PDF

- pdf.add_page() adds a page in the PDF

- pdf.set_font("Arial", size = 15) sets the font style and font size

- pdf.multi_cell creates and writes a line in the PDF

- pdf.ln(200) creates a break in the PDF

- pdf.image(f"{os.getenv('UPLOAD_IMG')}SimiLabs_2022/flaskr/static/imag es/ReportImages/delta_score.png", w = 500, h = 50) uploads an image to the PDF

- pdf.output(dest="F",   name=os.getenv('UPLOAD_REPORT')   + "styloReport.pdf") saves the PDF to the folder "UPLOAD_REPORT"

4.5.10.   EXTRACTMETADATA.PY

This module allows one to extract the metadata from document that was uploaded using the Quick text functionality of the application. The docx/python-docx, PyPDF2 libraries is needed to work with and extract text from the documents, and the datetime library is needed to format dates.

- The getMetaDataDoc() function receives a word document as a parameter. Metadata properties of a document such as the author, date of creation, last modified author as well as the last modified date are stored in a dictionary. This function returns the metadata dictionary.

- The getMetaDataPDF() function receives a PDF document as a parameter. The text and content of the document is extracted. The data variable contains a dictionary of metadata properties extracted from the PDF. One iterates through all the metadata properties, and extract the author, formatted creation date and last modified date. A last modified author is not extracted from the metadata, since the PDF library cannot extract that kind of metadata. The last modifid author is simply specified as "UNKNOWN". Finally, the function returns a list containing the metadata property values.

- The getRowColor() function returns a string in the form of a bootstrap CSS class, which returns a green table row highlight when the original author matches the last modified author. Otherwise, it returns a red table row highlight.

```
extractmetadata.py 6 ✕

flaskr > 🐍 extractmetadata.py > �‹› getRowColor
        Hano Strydom, 2 weeks ago | 2 authors (Plea_Banshee and others)
  1     import docx
  2     import PyPDF2
  3     from datetime import datetime
  4
  5     def getMetaDataDoc(doc):
  6         metadata = {}
  7         prop = doc.core_properties
  8         metadata["author"] = prop.author
  9         metadata["created"] = prop.created
 10         metadata["last_modified_by"] = prop.last_modified_by
 11         metadata["modified"] = prop.modified
 12         return metadata
 13
 14     def getRowColor(author, lastModifiedBy):
 15         if(author == lastModifiedBy):           Hano Strydom, 2 weeks ago • Beautified Table on QuickRepor
 16             return "w3-light-grey w3-hover-green"
 17         else:
 18             return "w3-light-grey w3-hover-red"
 19
 20     def getMetaDataPDF(pdf):
 21         author = ""
 22         created_date = ""
 23         last_modified_date = ""
 24         # pdfFile = PyPDF2.PdfFileReader(open(pdf,'rb'))
 25         pdfFile = PyPDF2.PdfReader(open(pdf,'rb'))
 26         data = pdfFile.getDocumentInfo()
 27         print(data)
 28         for metadata in data:
 29             if(metadata == '/Author'):
 30                 author = data[metadata]
 31             if(metadata == '/CreationDate'):
 32                 created_date = data[metadata]
 33                 created_dt = datetime.strptime(created_date.replace("'", ""), "D:%Y%m%d%H%M%S%z")
 34                 createdDate = created_dt.strftime("%Y-%m-%d %H:%M:%S")
 35             if(metadata == '/ModDate'):
 36                 last_modified_date = data[metadata]
 37                 last_dt = datetime.strptime(created_date.replace("'", ""), "D:%Y%m%d%H%M%S%z")
 38                 lastDate = last_dt.strftime("%Y-%m-%d %H:%M:%S")
 39         return [author, createdDate, "UNKNOWN", lastDate]
 40
 41
```

4.5.11. GENSIM.PY

In this section, the Gensim.py file will be discussed in detail in order to provide the user with a better understanding of what happens behind the scenes of the extensive text comparison functionality. The file will be discussed in a top to bottom fashion. Initially the following libraires and packages will be required to run this part of the application:

```
flaskr > 🐍 Gensim.py > ...
  1     import os
  2     import tkinter as tk
  3     from tkinter import filedialog
  4     import docx
  5     from gensim import corpora
  6     from gensim import models
  7     from gensim.utils import simple_preprocess
  8     from gensim.corpora import MmCorpus
  9     import logging
 10     from flask import Flask, render_template, request, redirect, url_for, flash
 11     import PyPDF2
 12     import shutil
 13
```

This part of the application utilises logging to provide further insight into what is happening when the application is running:

```
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.DEBUG)
```

The application has two functions to support the extraction of texts from either a pdf or docx document. The application will automatically detect which function to use when extracting texts from documents:

```python
def getText(filename):
    doc = docx.Document(filename)
    fullText = []
    for para in doc.paragraphs:
        fullText.append(para.text)
    return "".join(fullText)

def getPdfText(filename):
    pdfFileObj = open(filename, 'rb')
    pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
    count = pdfReader.numPages
    text = ""
    for i in range(count):
        pageObj = pdfReader.getPage(i)
        text += pageObj.extractText()
    return text
```

The main functionality of the extensive text comparison is divided between 4 functions, namely:

- CreateStudent
- UpdateStudent
- CompareCorpus
- DeleteStudent

CreateStudent

The CreateStudent function has two parameters, stdNum, and filename. The stdNum is the student number entered by the user and filename is the name of the file that was uploaded by the user.

The student number is used to create a path which will be used to store all the student data:

```python
def CreateStudent(stdNum, fileName):
    path = os.getenv('UPLOAD_EXTENSIVE')
    pathJoin = os.path.join(path, stdNum)

    if not os.path.exists(pathJoin):
        os.mkdir(pathJoin)
        os.mkdir(pathJoin + '/Corpus File')
        os.mkdir(pathJoin + '/Data')
    else:
        print("Directory already exists")
```

Once the path and directory have been created, the application will detect the file type of the uploaded document:

```python
47  if fileName.endswith('.docx'):
48      result = ''.join(line.strip() for line in getText("./flaskr/GensimTemp/" + fileName).splitlines())
49  elif fileName.endswith('.pdf'):
50      result = ''.join(line.strip() for line in getPdfText("./flaskr/GensimTemp/" + fileName).splitlines())
51      #print(getPdfText("./flaskr/GensimTemp/" + fileName))
52      #print(result)
53  else:
54      #print('File not supported')
55      return
```

The extracted text results from the uploaded document will then be written into the specified student corpus. The application checks whether the directory and files exists to determine if it should create new files or append the current ones:

```python
#Write to Corpus File
checkFile = os.path.isfile(pathJoin + '/Corpus File/' + stdNum + '_corpus.txt')
checkDoc = os.path.isfile(pathJoin + '/Data/' + stdNum + '_docNames.txt')
if checkFile == True and checkDoc == True:
    docCorpus = open(pathJoin + '/Corpus File/' + stdNum + '_corpus.txt', "a", encoding="utf-8")
    docCorpus.write(result + '\n')
    docCorpus.close()
    docNames = open(pathJoin + '/Data/' + stdNum + '_docNames.txt', "a", encoding="utf-8")
    docNames.write(fileName + '\n')
    docNames.close()
else:
    docCorpus = open(pathJoin + '/Corpus File/' + stdNum + '_corpus.txt', 'w', encoding="utf-8")
    docCorpus.write(result + '\n')
    docCorpus.close()
    docNames = open(pathJoin + '/Data/' + stdNum + '_docNames.txt', "w", encoding="utf-8")
    docNames.write(fileName + '\n')
    docNames.close()
```

Next we shift our focus to the UpdateStudent function. The UpdateStudent function has two parameters, stdNum, and filename. The stdNum is the student number entered by the user and filename is the name of the file that was uploaded by the user.

The student number will be used to check whether the student exists and to determine where the data should be stored:

The process is similar to that of the CreateStudent function but appends the specified student corpus instead of creating a new one:

```python
def UpdateStudent(stdNum, fileName):
    path = os.getenv('UPLOAD_EXTENSIVE')
    pathJoin = os.path.join(path, stdNum)
    if fileName.endswith('.docx'):
        result = ''.join(line.strip() for line in getText("./flaskr/GensimTemp/" + fileName).splitlines())
    elif fileName.endswith('.pdf'):
        result = ''.join(line.strip() for line in getPdfText("./flaskr/GensimTemp/" + fileName).splitlines())
    else:
        #print('File not supported')
        return
```

```python
#Write to Corpus File
if not os.path.exists(pathJoin):
    os.mkdir(pathJoin)
    os.mkdir(pathJoin + '/Corpus File')
    os.mkdir(pathJoin + '/Data')

checkFile =  os.path.isfile(pathJoin + '/Corpus File/' + stdNum + '_corpus.txt')
if checkFile == True:
    docCorpus = open(pathJoin + '/Corpus File/' + stdNum + '_corpus.txt', "a", encoding="utf-8")
    docCorpus.write(result + '\n')
    docCorpus.close()
    docNames = open(pathJoin + '/Data/' + stdNum + '_docNames.txt', "a", encoding="utf-8")
    docNames.write(fileName + '\n')
    docNames.close()
    #print('corpus updated')
else:
    docCorpus = open(pathJoin + '/Corpus File/' + stdNum + '_corpus.txt', 'w', encoding="utf-8")
    docCorpus.write(result + '\n')
    docCorpus.close()
    docNames = open(pathJoin + '/Data/' + stdNum + '_docNames.txt', "w", encoding="utf-8")
    docNames.write(fileName + '\n')
    docNames.close()
    #print('corpus updated')
```

Next we will look at the DeleteStudent function. The DeleteStudent function has one parameter, stdNum, which will be used to identify which folder to delete. The student number is provided by the user and subsequently the application will delete all folders and files relating to the specified student number:

```python
def DeleteStudent(stdNum):
    path = os.getenv('UPLOAD_EXTENSIVE')
    pathJoin = os.path.join(path, stdNum)
    shutil.rmtree(pathJoin)
    #print('Student deleted')
```

The final function will focus on is the CompareCorpus function. The CompareCorpus has three parameters, namely stdNum, filename, and bolean. The stdNum is the student number provided by the user. The student number is used to determine which corpus should be utlised during the comparison. The filename is the name of the file uploaded by the user which will be used for comparison to the corpus. The CompareCorpus function utilises the read_multiplefiles class to read all the individual documents stored within the corpus:

```python
class read_multiplefiles(object):
    def __init__(self, dir_path):
        self.dir_path = dir_path
    def __iter__(self):
        for filename in os.listdir(self.dir_path):
            for line in open(os.path.join(self.dir_path, filename), encoding="utf-8"):
                yield simple_preprocess(line)
```

The application starts by determining the path based on the student number and subsequently extracts the corpus texts and converts them into a gensim dictionary. A list of stop words is provided and will be removed when text is extracted from the corpus. Afterwards a gensim corpus is created based on the tokenised and processed text:

```python
def CompareCorpus(stdNum, fileName, boolean):
    # Path
    path = os.getenv('UPLOAD_EXTENSIVE')
    pathJoin = os.path.join(path, stdNum)
    dictionary = corpora.Dictionary(line.lower().split() for line in open(pathJoin + '/Corpus File/' + stdNum + '_corpus.txt', encoding="utf-8"))
    # remove stop words and words that appear only once
    stoplist = set('for a of the and to in'.split())
    stop_ids = [
        dictionary.token2id[stopword]
        for stopword in stoplist
        if stopword in dictionary.token2id
    ]
    once_ids = [tokenid for tokenid, docfreq in dictionary.dfs.items() if docfreq == 1]
    dictionary.filter_tokens(stop_ids + once_ids)  # remove stop words and words that appear only once
    dictionary.compactify()  # remove gaps in id sequence after words that were removed
    print(dictionary)
    # save your dictionary to disk
    dictionary.save(pathJoin + '/Data/' + stdNum + 'dictionary.dict')
    print("Dictionary saved")

    #creating a bag-of-words corpus from multiple files in the directory provided
    corpus = [dictionary.doc2bow(token, allow_update=True) for token in read_multiplefiles(pathJoin + '/Corpus File/')]
    corpora.MmCorpus.serialize(pathJoin + '/Data/' + stdNum + 'corpus.mm', corpus)
    # print("BoW corpus saved")
```

After the gensim corpus and dictionary is created, the different models can be created based on these files. First the LSI model for the submitted document is created:

```
lsi = models.LsiModel(corpus, id2word=dictionary, num_topics=2)
if fileName.endswith('.docx'):
    result = ''.join(line.strip() for line in getText("./flaskr/GensimTemp/" + fileName).splitlines())
elif fileName.endswith('.pdf'):
    result = ''.join(line.strip() for line in getPdfText("./flaskr/GensimTemp/" + fileName).splitlines())
else:
    #print('File not supported')
    return
```

The data is then converted into the LSI space:

```
vec_bow = dictionary.doc2bow(result.lower().split())
vec_lsi = lsi[vec_bow]  # convert the query to LSI space
```

Finally, the similarities are calculated:

```
from gensim import similarities
index = similarities.MatrixSimilarity(lsi[corpus])  # transform corpus to LSI space and index it

sims = index[vec_lsi]  # perform a similarity query against the corpus
print('-------LSI SIM-------')
print(list(enumerate(sims)))  # print (document_number, document_similarity) 2-tuples
```

The TF-IDF conversion processes utilise the gensim dictionary and corpus. A model is created for the corpus and uploaded document. The two models are compared to each other and the results are produced:

```
#TF-IDF model
tfidf = models.TfidfModel(corpus)
feauture_count = len(dictionary.token2id)
index2 = similarities.MatrixSimilarity(tfidf[corpus], num_features=feauture_count)
sim2 = index2[tfidf[vec_bow]]
resultlist = []
with open(pathJoin + '/Data/' + stdNum + '_docNames.txt') as f:
    docNames = f.readlines()
```

The results generated by the LSI and TD-IDF methods are formatted and passed through to the report page:

```
docNames = [x.strip() for x in docNames]
for x in range(len(docNames)):
    resultlist.append('TF-IDF model indicates the document is similar to ' + docNames[x] + 'with a score of: %.2f' % (sim2[x]))
    resultlist.append('LSI model indicates the document is similar to ' + docNames[x] + 'with a score of: %.2f' % (sims[x]))
```

The Boolean parameter is utilised to determine of the user wants to add the uploaded document to the corpus after comparison has completed. The user simply needs to check the check box and the document will be added to the corpus:

```python
if (boolean == "true"):
    checkFile = os.path.isfile(pathJoin + '/Corpus File/' + stdNum + '_corpus.txt')
    if checkFile == True:
        docCorpus = open(pathJoin + '/Corpus File/' + stdNum + '_corpus.txt', "a", encoding="utf-8")
        docCorpus.write(result + '\n')
        docCorpus.close()
        #print('Document added to corpus')
        docNames = open(pathJoin + '/Data/' + stdNum + '_docNames.txt', "a", encoding="utf-8")
        docNames.write(fileName + '\n')
        docNames.close()
    else:
        docCorpus = open(pathJoin + '/Corpus File/' + stdNum + '_corpus.txt', 'w', encoding="utf-8")
        docCorpus.write(result + '\n')
        docCorpus.close()
        #print('Document added to corpus')
        docNames = open(pathJoin + '/Data/' + stdNum + '_docNames.txt', "w", encoding="utf-8")
        docNames.write(fileName + '\n')
        docNames.close()
return resultlist
```

4.5.12. HELPERS.PY

```python
def lines(a, b):
    """Return lines in both a and b"""
    alist = set(a.split("\n"))
    blist = set(b.split("\n"))

    return list(alist & blist)


from nltk.tokenize import sent_tokenize

def sentences(a, b):
    """Return sentences in both a and b"""

    alist = set(sent_tokenize(a))
    blist = set(sent_tokenize(b))
    return list(alist & blist)


def substrings(a, b, n):
    """Return substrings of length n in both a and b"""
    alist = set()
    blist = set()

    for x in range(len(a) - n + 1):
        alist.add(a[x:x+n])
    for y in range(len(b) - n + 1):
        blist.add(b[y:y+n])

    return list(alist & blist)
```

- The lines function receives the parameters a and b (a is the text from the suspected plagiarism document and b is the text from the other submitted document)
- Alist splits the text of the parameter a into lines
- Blist splits the text of the parameter b into lines
- The function returns lines in both a and b
- The sentences function receives the parameters a and b (a is the text from the suspected plagiarism document and b is the text from the other submitted document)
- Alist splits the text of the parameter a into sentences using NLTK tokenize function

- Blist splits the text of the parameter b into sentences using NLTK tokenize function
- The function returns sentences in both a and b
- The substrings function receives the parameters a, b, and n (a is the text from the suspected plagiarism document and b is the text from the other submitted document, and n is the length of the substring)
- Alist is the words of a certain length n in document a
- Blist is the words of a certain length n in document b
- The function returns substrings of length n in both a and b

4.5.13. QUICKSIMILARITY.PY

This module provides the functionality to calculate the text similarity scores for the Quick Text documents. One needs to import the pandas library to use DataFrames (2D data structures) or matrixes to store data. The sklearn CountVectorizer and TfidVectoriser classes needs to be imported as well, which enables one to use vectors (single dimension arrays) to store data. Finally, one needs to import the sklearn cosine_similarity method that allows one to calculate the cosine similarity between two datasets using vectors or matrixes.

The calc_cosine_similarity function receives two parameters, which are the extracted text from an alleged and comparison document respectively. The datasets are converted into a vector format, and the cosine similarity is calculated based on the vector contents. If the vector contains a similarity score of less than 21%, the string "green" is returned by the function. If the vector contains a similarity score of more than 20% but less than 31%, the string "orange" is returned by the function. If the similarity score calculated doesn't fall within the previously specified ranges, the string "red" is returned by the function.

The jaccard_similarity() function calculates the similarity score of two documents based on the Jaccard algorithm. The function also receives two parameters: the extracted text from an alleged and comparison document. The similarity score is calculated based on set intersections and unions on the extracted text from the two documents. The function returns the string "green" if the similarity score is less than 21%. The string "orange" is returned by the function if the similarity score is greater

than 20% but less than 31%. The function returns the string "red" if the similarity score determined does not fall within the previously mentioned ranges.

```python
quicksimilarity.py 1 ×
flaskr > quicksimilarity.py > ...
        Hano Strydom, 2 weeks ago | 2 authors (Plea_Banshee and others)
1       import pandas as pd        Plea_Banshee, 2 weeks ago • QuickText similarities modified using the cosine …
2       from sklearn.feature_extraction.text import CountVectorizer
3       from sklearn.feature_extraction.text import TfidfVectorizer
4       from sklearn.metrics.pairwise import cosine_similarity
5
6       # Cosine Similarity is a method of calculating the similarity of two vectors by
7       # taking the dot product and dividing it by the magnitudes of each vector
8       # It is a method where text is transformed into numbers and stored in a vector
9       # https://en.wikipedia.org/wiki/Cosine_similarity
10
11      count_vect = CountVectorizer()
12      vectorizer = TfidfVectorizer()
13
14      def calc_cosine_similarity(x,y):
15          color = ""
16          corpus = [x,y]
17          X_train_counts = count_vect.fit_transform(corpus)
18          # creates a vector containing the two documents' word-to-numeric conversions
19          pd.DataFrame(X_train_counts.toarray(),columns=count_vect.get_feature_names(),index=['Document 1','Document 2'])
20
21          # Term Frequency Inverse Document Frequency (TFIDF).
22          # Technique to measure how unique a certain word is relative to every other word in a document.
23          # This is calculated on a scale from 0-1 with the most common words approaching 0 and the most unique words approaching 1.
24          trsfm=vectorizer.fit_transform(corpus)
25          pd.DataFrame(trsfm.toarray(),columns=vectorizer.get_feature_names(),index=['Document 1','Document 2'])
26          # this returns a matrix in the form of [[trsfm[0:1], trsfm]]. Calculates the cosine_similarity (trsfm)
27          val = cosine_similarity(trsfm[0:1], trsfm)
28          # print(f"Value: {val}")
29          if val[0][1] >= 0.00 and val[0][1] <= 0.20:
30              color = "green"
31          elif val[0][1] > 0.20 and val[0][1] <= 0.30:
32              color = "orange"
33          else:
34              color = "red"
35
36          # returns a tuple of the similarity score and the color
37          return val[0][1],color
38
39      def jaccard_similarity(x,y):
40          # returns the jaccard similarity between two lists
41          color = ""
42          intersection_cardinality = len(set.intersection(*[set(x), set(y)]))
43          union_cardinality = len(set.union(*[set(x), set(y)]))
44          val = intersection_cardinality/float(union_cardinality)
45          # determine color of text on frontend based on the sililarity score
46          if val >= 0.00 and val <= 0.20:
47              color = "green"
48          elif val > 0.20 and val <= 0.30:
49              color = "orange"
50          else:
51              color = "red"
52
53          # returns a tuple of the similarity score and the color
54          return val,color
55
```

4.5.14.   SCHEMA.SQL

```sql
CREATE DATABASE IF NOT EXISTS SimiLabs DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
USE Similabs;

--DROP TABLE IF EXISTS accounts;

CREATE TABLE IF NOT EXISTS Accounts (
  id int PRIMARY KEY AUTO_INCREMENT,
  username nvarchar(55) UNIQUE NOT NULL,
  password nvarchar(255) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

Create TABLE IF NOT EXISTS Students (
  StudentID int PRIMARY KEY AUTO_INCREMENT,
  studentFullName varchar(50) NOT NULL,
  studentNumber varchar(8) NOT NULL,
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;


Create TABLE IF NOT EXISTS Corpus (
  CorpusID int PRIMARY KEY AUTO_INCREMENT,
  studentID int NOT NULL,
  DocumentID int NOT NULL,
  corpusName varchar(50) NOT NULL,
  corpusDescription varchar(255) NOT NULL,
  FOREIGN KEY (studentID) REFERENCES students(StudentID)
  FOREIGN KEY (documentID) REFERENCES Document(DocumentID)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;


Create TABLE IF NOT EXISTS Document (
  DocumentID int PRIMARY KEY AUTO_INCREMENT,
  corpusID int NOT NULL,
  MetadataID int NOT NULL,
  documentName varchar(50) NOT NULL,
  FOREIGN KEY (corpusID) REFERENCES Corpus(CorpusID)
  FOREIGN KEY (MetadataID) REFERENCES Metadata(MetadataID)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;


Create TABLE IF NOT EXISTS Metadata (
  MetadataID int PRIMARY KEY AUTO_INCREMENT,
  documentID int NOT NULL,
  author varchar(50) NOT NULL,
  dateCreate date NOT NULL,
```

- CREATE DATABASE IF NOT EXISTS SimiLabs DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci: Creates the Similabs database

- USE Similabs: Specify we are using the Similabs database

- CREATE TABLE IF NOT EXISTS Accounts (
  id int PRIMARY KEY AUTO_INCREMENT,
  username nvarchar(55) UNIQUE NOT NULL,
  password nvarchar(255) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8:
  Creates the Accounts table

- INSERT INTO accounts VALUES (1,'admin','pbkdf2:sha256:260000$j2dCQrXnaDgBrbmO$896a6fe3480ca5cca5a433f78a52dc123ecd81c479620e5bb179de0cce115fef'): Inserts the login values into the ACCOUNTS table

- Other tables that were created can be taken further.

4.6.   ENVIRONMENT FILE

Environment variables are declared in this file. The MySQL variables are set up on every developer's local system, so there is no need to change the variables every time the project is worked on. Every other variable has to be changed to specify paths on the developer's local system.

```
MYSQL_HOST=localhost
MYSQL_USER=root
MYSQL_PASSWORD=admin
MYSQL_DB=SimiLabs
UPLOAD_IMG="D:/Uni 2022/ISE/Project/github similabs/"
UPLOAD_PDF="D:/Uni 2022/ISE/Project/github similabs/SimiLabs_2022/flaskr/GeneratePDF/"
UPLOAD_REPORT = "D:/Uni 2022/ISE/Project/github similabs/SimiLabs_2022/flaskr/GenerateReport/"
UPLOAD_EXTENSIVE = "D:/Uni 2022/ISE/Project/github similabs/SimiLabs_2022/flaskr/GensimData/"
HELP_PAGE="D:/Uni 2022/ISE/Project/github similabs/flaskr/templates/help/"
HELP_PDF = "D:/Uni 2022/ISE/Project/github similabs/flaskr/static/pdf/"
```

4.7.   STYLOMETRY FILES: ROOT DIRECTORY

Burrows delta python file:

The Burrows delta python file makes use of the following imports:

import operator

from collections import Counter

import numpy as np

import pandas as pd

from faststylometry import Corpus

get_top_tokens function:

Receives the corpus and the vocabulary size as the parameters. The function identifies the n highest-ranking tokens in the corpus.

param corpus: The corpus to look for the commonest words in. The corpus must be tokenised.

param vocab_size: The n most common words in the corpus will be used.

return: A list of the n most common tokens in the corpus.

set_top_tokens function:

Receives the corpus and the top tokens which are retrieved from the get_top_tokens function as the parameters. The function uses a predefined set of tokens to calculate Burrows' Delta on the corpus. You would do this if the corpus were a test corpus, and your model has been trained on a training corpus.

param corpus: the corpus you want to set the top tokens for.

param top_tokens: the top tokens which you have calculated elsewhere, e.g., from a training corpus.

get_token_counts function:

Receives the corpus as the parameter. The function calculates the number of times each of the top tokens occurs in the corpus and stores this data in two data frames in the corpus.

param corpus: The corpus to run the operation on and store the result in.

get_token_counts_by_author function:

Receives the corpus as the parameter. The function groups the token counts across works by the same author, so we have a token count for each author.

param corpus: The corpus to run the operation on and store the result in.

get_token_counts_by_author_and_book function:

Receives the corpus as the parameter. The function groups the token counts across works by the same author and document, so we have a token count for each document.

param corpus: The corpus to run the operation on and store the result in.

get_token_proportions function:

Receives the corpus as the parameter. The function calculates the fraction of words in each author's subcorpus which are equal to each word in our vocabulary.

param corpus: The corpus to run the operation on and store the result in.

get_author_z_scores function:

Receives the corpus and the training corpus as the parameters. The function calculates the Z-score relating the test corpus to each author's subcorpus in the training corpus.

param test_corpus: The corpus to run the operation on and store the result in.

param training_corpus: The training corpus with a few authors' subcorpora, which will be compared to the test corpus.

return: A dataframe of Z-scores for each author in the training corpus.

calculate_difference_from_train_corpus function:

Receives the test corpus and train corpus as the parameters. The function calculates the difference between the Z-scores of the test corpus and the Z-scores of each author in the training corpus.

param test_corpus: The corpus to run the operation on and store the result in.

param train_corpus: The training corpus with a number of authors' subcorpora, which will be compared to the test corpus.

calculate_burrows_delta function:

Receives the train corpus, test corpus, and vocabulary size as the parameters. The function calculates the Burrows' Delta statistic for the test corpus vs every author's subcorpus in the training corpus. This function makes use of each function described in this python file.

param train_corpus: A corpus of known authors, which we will use as a benchmark to compare to the test corpus by an unknown author.

param test_corpus: The corpus by an unknown author.

param vocab_size: We will take the top n tokens from the training corpus and use as the vocabulary for the model. Normally 50-100 make sensible values.

return: A DataFrame of Burrows' Delta values for each author in the training corpus.

corpus python file:

init function:

Receives the authors, documents, and tokens as parameters. The function creates a new corpus object, which is empty by default, but can be initialised optionally with a set of documents.

param authors: A list of strings representing author names. If a set of documents are being used to initialise the corpus, len(authors) must equal len(books) and len(tokens)

param books: A list of strings representing document titles.

param tokens: A list of lists representing tokens in the documents.

add_book function:

Receives the author, document, and the content of the document as parameters. The function adds a single document to the corpus. This can only be done if the corpus has not been initialised with documents in the constructor.

param author: The author's name as string.

param book: The document title as string.

param text: The document content as string.

tokenise function:

Receives the tokenise as a parameter. The function tokenises all the documents in the corpus using the custom tokenisation function.

param tokenise: a tokenise function which takes a str and returns a list of tokens. It is language-specific and should remove pronouns.

split function:

Receives the segment length as the parameter. The function Split the documents in the corpus into smaller documents of a specified maximum length. So, if segment_length = 1000 and you have a document of length 1500, it will be split into two documents of lengths 1000 and 500 respectively, with the same author and title attributes. This Corpus object is not modified but a new instance will be constructed.

param segment_length: The maximum length of documents in the new corpus.

return: The new corpus with split documents.

en python file:

The en python file makes use of the following imports:

- import re
- import nltk

The Python file creates a list of stopwords which is removed from the text.

tokenise_remove_pronouns_en function:

Receives a text as the parameter. The function tokenises a sentence according to Englsh rules, and remove all pronouns.

param text: the original sentence.

return: all non-pronoun tokens.

probability python file:

The probability python file makes use of the following imports:

import numpy as np

import pandas as pd

from sklearn.base import BaseEstimator

from sklearn.linear_model import LogisticRegression

from faststylometry.burrows_delta import calculate_burrows_delta

from faststylometry.corpus import Corpus

every_item_but_one function:

Receives a list to process and the index to be excluded from the list as the parameters. The function returns every item in the list except for the one at index idx.

param l: a list to process.

param idx: the index to be excluded from the list.

return: the list l minus the item at index idx.

get_calibration_curve function:

Receives the corpus as the parameter. The function calculates the probability calibration curve of the Burrows' Delta calculation on the train corpus, using a cross-validation technique.

param corpus: the corpus for which we desire a probability calibration curve.

return: two arrays, an array of ground truths (0: different authors, 1: same author), and of Burrows' delta values. These can be used to calibrate a model such as logistic regression, or to generate a ROC curve.

calibrate function:

Receives the corpus as a parameter. The function makes use of the get_calibration_curve function to calibrate the machine learning probability model which uses LogisticRegression.

predict_proba function:

Receives the train corpus and the test corpus as the parameters. The function returns the probability that the test corpus is by the same author as the training corpus.

param train_corpus: The corpus to serve as a baseline for the word frequency calculations.

param test_corpus: The corpus to compare it with.

return: The probability according to the calibrated model, that the test corpus was by the same author as the train corpus.

util python file

The util python file makes use of the following imports:

import os

import re

from faststylometry.corpus import Corpus

load_corpus_from_folder function:

Receives the folder and pattern as the parameters. The function iterates over the .txt files in a folder and adds their contents to the corpus. Assumes that files are of the format [author]_-_[title].txt.

param folder: the folder on our local system to iterate over.

param pattern: an optional filter to apply to the filenames.

return: a corpus of the files found.

RunFastStylometry.py:

Main script to execute stylometric analysis.

Necessary imports (pip install):

- Faststylometry

```python
from faststylometry import Corpus
from faststylometry import load_corpus_from_folder
from faststylometry import tokenise_remove_pronouns_en
from faststylometry import calculate_burrows_delta
from faststylometry import predict_proba, calibrate, get_calibration_curve
```

Import all necessary classes and functions from the following Python scripts:

- corpus.py
- util.py
- en.py
- burrows_delta.py
- probability.py

```
train_corpus = load_corpus_from_folder("data/train")
train_corpus.tokenise(tokenise_remove_pronouns_en)
```

Load the data from the train folder into the training corpus and tokenise the corpus by removing pronouns.

```
test_corpus = load_corpus_from_folder("data/test", pattern=None)
test_corpus.tokenise(tokenise_remove_pronouns_en)
```

Load the data from the test folder into the test corpus and tokenise the corpus by removing the pronouns.

```
import dataframe_image as dfi
calculate_burrows_delta(train_corpus, test_corpus, vocab_size = 50)
dfi.export(calculate_burrows_delta(train_corpus, test_corpus, vocab_size = 50),
'./flaskr/static/images/ReportImages/delta_score.png')
```

Import dataframe_image to save the dataframe as an image. Call the calculate_burrows_delta method and pass the training corpus, test corpus and vocabulary size as parameters to calculate the Burrows' delta for every author. The vocabulary size should be between 50 and 200 for the most accurate results. Export the returned dataframe to an image and store it in the ReportImages folder.

```
calibrate(train_corpus)
```

Call the calibrate method and pass the training corpus as a parameter.

```
predict_proba(train_corpus, test_corpus)
dfi.export(predict_proba(train_corpus,                         test_corpus),
'./flaskr/static/images/ReportImages/author_probability.png')
```

Call the predict_proba method and pass the training corpus and test corpus as parameters to predict the probability of authorship for every text. Export the returned dataframe to an image and store it in the ReportImages folder.

```python
import numpy as np
x_values = np.arange(0, 3, 0.1)

import matplotlib
matplotlib.use('agg')
import matplotlib.pyplot as plt
```

Import numpy and matplotlib to sketch diagrams. Give a range for the x values. Choose 'agg' as the backend for a noninteractive diagram.

```python
plt.plot(x_values,      train_corpus.probability_model.predict_proba(np.reshape(x_values,      (-1, 1)))[:,1])
plt.xlabel("Burrows delta")
plt.ylabel("Probability of same author")
plt.title("Calibration curve of the Burrows Delta probability model\nUsing Logistic Regression with correction for class imbalance")
plt.savefig('./flaskr/static/images/ReportImages/calibration.png', bbox_inches='tight')
```

Plot the values of the training corpus probabilities and store the diagram in the ReportImages folder.

```python
ground_truths, deltas = get_calibration_curve(train_corpus)
```

Calculate the ground truths and deltas of the training corpus.

```python
probabilities = train_corpus.probability_model.predict_proba(np.reshape(deltas, (-1, 1)))[:,1]
```

Calculate the probabilities of the training corpus.

```python
from sklearn.metrics import roc_curve, auc
```

Import sklearn to plot the ROC Curve.

```python
fpr, tpr, thresholds = roc_curve(ground_truths, probabilities)
```

Calculate fpr, tpr and thresholds from the ground truths and probabilities.

```
roc_auc = auc(fpr, tpr)
```

Calculate the ROC and AUC Curve.

```python
plt.figure()
plt.plot(fpr, tpr, color='darkorange',lw=1, label='ROC curve (area = %0.4f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating curve of the Burrows\' Delta classifier\noperating on entire books')
plt.legend(loc="lower right")
plt.savefig('./flaskr/static/images/ReportImages/operating_curve.png', bbox_inches='tight')
```

Plot the ROC Curve and store the image in the reportimages folder.

```python
from sklearn.decomposition import PCA
import re
import pandas as pd
```

Import sklearn, re, and pandas.

```python
test_corpus = load_corpus_from_folder("data/train")
test_corpus.tokenise(tokenise_remove_pronouns_en)
```

Reload the training corpus as the test corpus, re-tokenise it and segment it.

```python
split_test_corpus = test_corpus.split(80000)
```

Split each document into segments of 80000 words. This variable can be changed to increase or decrease accuracy. It is only useful for documents that are more than 100000 words, to increase the accuracy of the model and to compare different parts of the same document with each other.

```python
df_delta = calculate_burrows_delta(train_corpus, split_test_corpus)
```

Calculate the Burrows' delta using the training corpus and the test corpus that was split into segments.

```python
df_z_scores = split_test_corpus.df_author_z_scores
```

Calculate the z scores of the split test corpus.

```
pca_model = PCA(n_components=2)
```
Calculate the PCA model.

```
pca_matrix = pca_model.fit_transform(df_z_scores)
```
Calculate the PCA matrix.

```
authors = df_z_scores.index.map(lambda x : re.sub(" - .+", "", x))
```
Calculate the z scores of all authors.

```
df_pca_by_author = pd.DataFrame(pca_matrix)
df_pca_by_author["author"] = authors
```
Calculate the PCA matrix of all authors.

```
plt.figure(figsize=(15,15))
for author, pca_coordinates in df_pca_by_author.groupby("author"):
        plt.scatter(*zip(*pca_coordinates.drop("author", axis=1).to_numpy()), label=author)
plt.plot(*zip(*pca_coordinates.drop("author", axis=1).to_numpy()), alpha=0.5)
for i in range(len(pca_matrix)):
    plt.text(pca_matrix[i][0], pca_matrix[i][1]," " + df_z_scores.index[i], alpha=0.5)
plt.xlabel("Burrows Delta")
plt.ylabel('Probability of same author')
plt.legend()
plt.savefig('./flaskr/static/images/ReportImages/plot.png', bbox_inches='tight')
plt.title("Representation using PCA of works in training corpus")
```
Plot the PCA model and store the image in the ReportImages folder.

5. SIMILABS 2022 NEXT RELEASE AND FEATURES

There is a lot of possibilities and extended features that can be implemented for the next SimiLabs plagiarism checker release:

5.1. Migrating Frameworks from Flask to Django

Potential issues that a detected when using the Flask framework to develop the application:

- Increased likelihood of security hazards
- Typically, have a slower MVP (Model View Presenter) development
- More complex tech stack than most other web development frameworks
- Higher maintenance costs for more complex systems
- More complicated maintenance for larger implementations
- Does not have as much community support as other frameworks such as Django
- The Flask framework does not handle huge loads/traffic/requests well
- Difficult to migrate from Flask to other web frameworks
- Flask does not provide full stack templates
- More difficult to write CRUD operations for a database
- Flask apps are not great for production

It would be a good idea to migrate to an alternative Python web framework, such as Django.

Django is a free, open-source, Python-based web development framework that makes it easier to build coherent and clean websites or webapps that are powered by databases. It lessens a lot of the difficulties that a website developer must deal with. It is employed to create simple yet clever web apps. Django simultaneously enables developers to produce web applications in response to shifting business requirements. It handles all the hassle associated with web development because it is created by professionals, allowing you to concentrate on creating the application rather than learning the fundamentals.

Some of the advantages of developing web applications with Django:

- Django is backwards compatible, making it easier to work with older versions and templates of web content.

- Django have huge community support and well-defined documentation

- Django is DevOps compatible, which makes it extremely useful when using Agile methodologies to develop webapps

- Django offers extreme reusability of code, and this can keep webapps short and simple
- Django provides great infrastructure and allows the easy implementation of CRUD operations for databases
- Django provides excellent security features
- Easier and faster to develop webapps
- Provides neat HTML templates and views for applications

5.2. Quick Text

1. Search Functionality

On the Quick Text Report page, a search functionality can be implemented when the content of the alleged and comparison document is displayed on the screen. The alleged and comparison document containers can contain built-in search textboxes that highlights and navigates to a specific search term respectively.

2. Academic Writing Improvement functionality

One can use the statistics from the Quick Text, Extensive Text and Stylometry reports to determine whether a student improved their academic writing over a period via comparing their overall academic assignments. By determining wordcounts, vocabulary, and explicit writing styles, one can determine whether a student improved their own academic writing.

3. Extended Scrollbar Functionality

When comparing the two document's content, and the user selected the line-by-line algorithm for the Quick Text Comparison, allow the two pages' content to scroll down at the same time at the same level to make it easier for the user to navigate and compare the text similarities.

4. Auto detection of text similarity algorithm

The application should be able to detect which algorithm to use (Jaccard or Cosine) based on the number of words in both texts.

6.1   Additional Future Extensive Text

This feature allows the user to select a corpus in a folder format saved on a localhost. The user can also select the suspected document to test against the corpus for the percentage of text comparison.

Imports

The following imports is necessary for this feature:

- import gensim.models as gsm
- from os import listdir
- from os.path import join
- from gensim.models.doc2vec import TaggedDocument
- from tkinter import filedialog
- from tkinter.filedialog import askopenfilename
- import os
- import docx2txt
- from docx2txt import process
- import genism

Convert the word corpus to txt

In this method, the folder full of word documents will be converted to a folder of txt documents. This method will create a new folder named "Corpus" and save the word documents as txt documents within this folder. If the folder already exists, it is ignored, and the new txt documents is just added to the corpus.

class DocIterator(object)

This class is used to tag the file names to display them with the results. It consists of the "__init__()" and "__iter__()" methods.

Training the model

With this, a model is trained according to the words within the corpus. The code below will determine how the model will be trained and this will effect the results.

model = gensim.models.doc2vec.Doc2Vec(vector_size=50,min_count=2,epochs=40)

model.build_vocab(it)

model.train(it, total_examples=len(doc), epochs=30)

print(model.epochs)

def createModelFolder()
This method creates a separate folder within the word corpus. The model will be saved within this folder in order to avoid errors when calculations are performed.

Get suspected document()
This method will ask the user to select the directory of the suspected document to compare with the corpus. This will extract the text from the document and compare it with the corpus text.

Display the Similarity between documents

This part will create a vector with the word and calculate the similarity of each document to the suspected document.
new_test = open(join(path), 'r', encoding = 'iso-8859-1', errors = 'ignore').read().split()
inferred_docvec = m.infer_vector(new_test)
print('%s:\n %s' % (model, m.dv.most_similar(positive=[inferred_docvec], topn=length)))

Keep in mind
- This feature only works with word documents.
- When selecting the corpus folder, select the folder with word documents, not the corpus with txt documents.
- The suspected word document will be added to the corpus.
- The word documents may not be empty.
- The results of the comparison percentage are not 100% accurate.

6.2   Stylometry

- Add feature to select specific students and documents for POC plot. Having a lot of students in the corpus will make the plot unreadable.
- Add feature to change stopwords. The current stopwords are pronouns which are removed from the corpus. You can change it to common words (a, the, an,

etc.) to see the difference it makes to the results. You can give the user the option to choose which stopwords to remove, or to not remove any words.

- Segmenting the corpus with the specified words can be changed. Currently the documents are split at 80000 words, but this can be changed according to the average length of documents or to compare sections of the same document.
- Allow metadata content of the documents to be displayed using tooltips when hovering over the images.

6.3 Extensive Text Comparison Recommendations

- Additional models can be added in conjunction with the LSI and TD-IDF models to increase the accuracy of results produced. See Gensim website for additional info.
- The initial corpus comparison functionality can be extended to compare a document to various other student corpora
- An additional div can be introduced which can contain a list of all the current student numbers contained in the application

6.4 Prettifying the user interface

It would be a good idea to make the user interface more user-friendly and add additional styling to the pages in order to improve the user's experience.

6.5 Corpus Builder

Allow the user to build their own corpora and selecting their own documents to add to a corpus. It will also be more useful to allow users to view and modify corpora content on the application itself

6.6 Statistics Validity

Provide resources or tools within the application that can verify the validity of the statistics on the reports.