



A TECHNICAL REPORT ON

CARTOONIFY WEB APPLICATION USING PYTHON

(COE 505: SOFTWARE ENGINEERING)

PREPARED BY

GROUP C

DEPARTMENT OF COMPUTER ENGINEERING

SUBMITTED TO:
ENGR. OMITOLA

JANUARY 2023

TABLE OF CONTENTS

TABLE OF CONTENTS	i
LIST OF TABLES AND FIGURES	iii
LIST OF ACRONYMS	iv
ABSTRACT	v
CHAPTER ONE	1
1.0 INTRODUCTION	1
CHAPTER TWO	3
2.0 LITERATURE REVIEW	3
2.1 Why Python?	4
CHAPTER THREE	5
3.0 METHODOLOGY	5
3.1 Application Workflow	5
3.1.1 Flowchart	7
3.2 Backend	8
3.2.1 Python	8
3.2.2 Flask Framework	8
3.2.3 Database	9
3.3 Frontend	10
3.4 Packaging/Compiling	10
3.5 Requirements	11
3.5.1 What are the specified requirements?	11
3.5.2 Table of Requirements	12

3.6 Deployment	13
3.7 Testing	14
3.8 Validation and Verification Checks	15
3.9 Documentation	15
3.10 Code	16
CHAPTER FOUR.....	18
4.0 RESULTS AND DISCUSSIONS	18
4.1 Load Test Results	19
4.2 Database Schema	22
4.3 Functionality Test Results	23
CHAPTER FIVE.....	25
5.0 RECOMMENDATIONS	25
5.1 CONCLUSION	26
REFERENCES.....	27
APPENDIX	28

LIST OF TABLES AND FIGURES

Figure 1. 1: Application Homepage	2
Figure 1. 2: Application Output.....	2
Figure 3. 1: Application Flowchart	7
Figure 4. 1: Locust Test Monitoring Dashboard	19
Figure 4. 2: Locust Test Start Page.....	20
Figure 4. 3: Locust Test Total Requests per second.....	20
Figure 4. 4: Locust Test Response times	21
Figure 4. 5: Number of users against time	21
Figure 4. 6: Example of sample database details	22
Figure 4. 7: Database Model Schema.....	23
Figure 4. 8: Pytest with Github Actions Test Results.....	23
Figure 4. 9: Local Tests Results	24
Table 3. 1: Requirements Description	12

LIST OF ACRONYMS

HTML: HyperText Markup Language.....	1
CSS: Cascading Style Sheets	1
SQL: Structured Query Language	1
ORM: Object-relational mapping.....	4
JPEG: Joint Photographic Experts Group	24
PNG: Portable Network Graphics	24
GIF: Graphics Interchange Format.....	24
BMP: Bitmap Image file	24

ABSTRACT

This report presents a web application that uses Python, Flask, SQLAlchemy, and SQLite to handle user authentication and manage a database of uploaded images. The application features a login and registration page and allows users to upload and cartoonify images. The use of Python and relevant libraries enables efficient and secure handling of user authentication and image management.

Keywords: Python, Flask, SQLAlchemy, SQLite, Web application

CHAPTER ONE

1.0 INTRODUCTION

In this report, we present a web application called “Cartoonify”, developed using Python, HTML, CSS, JavaScript, Postgres and the Flask web framework. The application handles user authentication and utilizes an SQLite or Postgres database to store user information. In addition, the application employs the SQLAlchemy library to facilitate interaction with the database.

Upon accessing the application, users are presented with a login page. If the user does not have an existing account, they have the option to register by providing a unique username and password.

Upon successful login or registration, the user is redirected to the main page of the application.

On the main page, the user could upload an image of their choice. Upon selecting an image and clicking the "Cartoonify" button, the application processes the image and generates a cartoonified version of the original. The resulting image is then displayed to the user.

This application serves as a simple and intuitive platform for creating cartoonified versions of images. Its implementation using Flask, SQLAlchemy, and SQLite enables efficient and secure handling of user authentication and image management.

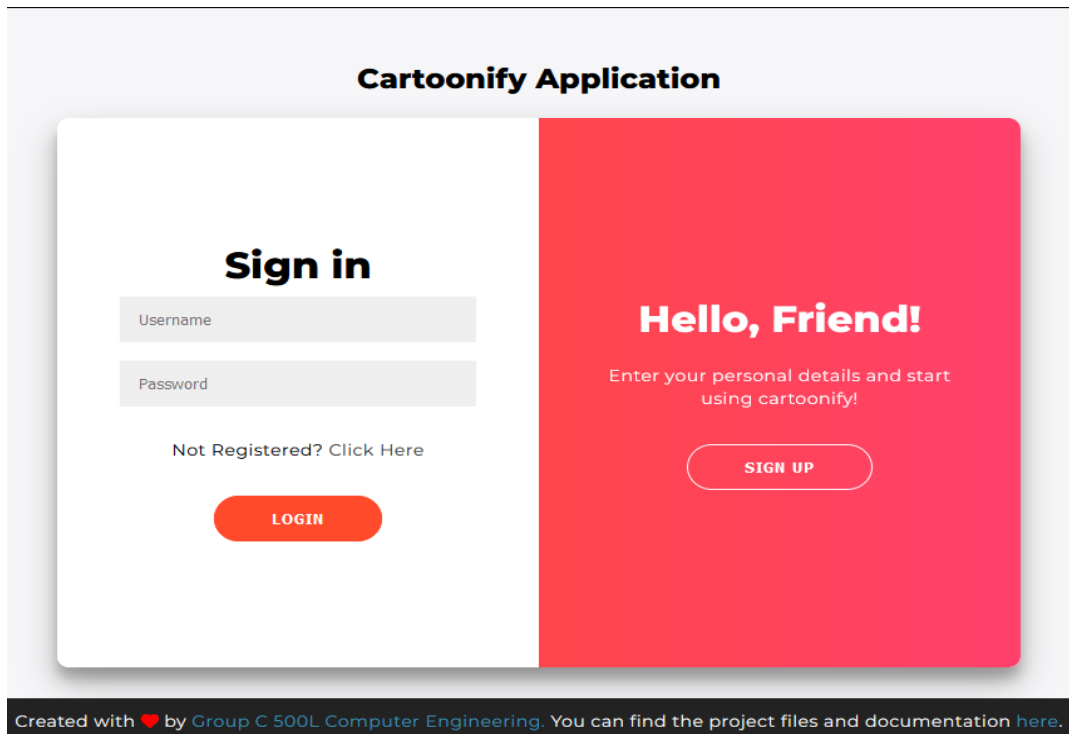


Figure 1. 1: Application Homepage



Figure 1. 2: Application Output

CHAPTER TWO

2.0 LITERATURE REVIEW

The application developed using Python, HTML, CSS, JavaScript, and the Flask web framework, allows users to upload an image and generate a cartoonified version of the original. It also includes features for user authentication and image management, utilizing an SQLite or Postgres database and the SQLAlchemy library to facilitate interaction with the database.

Python is a popular programming language known for its versatility and ease of use. It is commonly used in web development and has a variety of frameworks available to assist with the development process. One such framework is Flask, which offers a lightweight and flexible platform for building web applications.[1]

HTML, CSS, and JavaScript are standard technologies used in the creation of web pages and applications. HTML is used to structure and organize content, while CSS is used to style and layout the content, and JavaScript is used to add interactivity to the page.[2]

In terms of database management, the application uses either a SQLite[3] or Postgres[4] database to store user information such as username, email and password. The passwords are hashed and salted using the Weurkzeug library before being stored in the database. SQLite is a self-contained, serverless database engine that is widely used for small to medium-sized applications. Postgres is a more powerful and feature-rich database management system that is well-suited for larger and more complex applications. The SQLAlchemy library is used to facilitate interaction with the database, providing a set of tools and resources for working with data in Python.

2.1 Why Python?

Python and Flask are well-suited for building web applications due to several advantages. Some of the benefits of using Python and Flask in building the "Cartoonify" application, include:

- **Versatility:** Python is a versatile programming language that can be used for a wide range of applications, including web development, scientific computing, data analysis, and more. This makes it a good choice for building applications that may require a variety of different features or functionality.[5]
- **Ease of use:** Python is known for its simplicity and readability, making it easier for developers to write and maintain code. This can be especially useful for building larger and more complex applications, as the codebase will be easier to navigate and understand.[6]
- **Flask:** Flask is a lightweight and flexible web framework that is built on top of Python. It provides a simple and easy-to-use platform for building web applications, including handling routing, templates, and more. This can help to streamline the development process and make it easier to build and deploy an application.[7]
- **SQLAlchemy:** The SQLAlchemy library, which is used in the "Cartoonify" application, makes it easier to interact with databases in Python. It provides a set of tools and resources for working with data in Python, including support for object-relational mapping (ORM) and a variety of database engines. This can simplify the process of storing and retrieving data in the application, improving the efficiency and scalability of the application.[8]

CHAPTER THREE

3.0 METHODOLOGY

The methodology for the "Cartoonify" application report involved a thorough review of the tools and technologies used in the development of the application. This included an examination of the programming languages and frameworks used, such as Python and Flask, as well as the database management system and tools for interacting with the database, such as SQLite and SQLAlchemy. The review also included an analysis of the application's features and functionality, including user authentication and image processing, as well as the user experience and overall performance of the application.

To gather this information, a variety of sources were consulted, including documentation and documentation for the tools and technologies used, as well as the source code for the application itself. In addition, the application was tested and evaluated to ensure that it was functioning correctly and meeting the requirements and goals outlined in the original project specification.

3.1 Application Workflow

Here is a pseudocode algorithm that outlines the workflow of the "Cartoonify" application as described:

```
1. Start (A)
2.
3. // Register user
4. B: Register User()
5.
6. // Login
7. C: Login()
8.
9. // Check login status
10. D: If (login successful)
11.     E: Check User Details()
12.     F: If (details valid)
13.         G: Home()
14.     else
15.         C: Login()
16.
17. // Home node
```

```
18. G: Home()  
19.    // User selects to upload image  
20.    H: Create User Folder()  
21.    I: Upload Image()  
22.    J: Cartoonify Image()  
23.    K: Display Images()  
24.    // User selects to log out  
25.    L: Logout()  
26.    M: Delete User Folder()  
27.    A: Start()
```

The pseudocode provided outlines the workflow of the "Cartoonify" application. It starts at the "Start" node (A) and proceeds to the "Register User" operation (B). This operation allows new users to create an account by providing a unique username and password.

Next, the "Login" operation (C) is performed, which allows users to log into their accounts. This is followed by a condition check (D) to determine whether the login was successful. If the login was successful, the "Check User Details" operation (E) is performed to verify that the user's details are valid. If the login was not successful or the user's details are not valid, the operation goes back to the "Login" operation (C).

If the login and user details are successful, the operation proceeds to the "Home" end node (G). From this node, the user can either upload an image or log out of the application. If the user chooses to upload an image, the "Create User Folder" operation (H) is performed to create a folder for the user's images. This is followed by the "Upload Image" operation (I), which allows the user to select and upload an image from their local machine.

After the image has been uploaded, the "Cartoonify Image" operation (J) is performed, which processes the uploaded image and generates a cartoonified version of it. The resulting image is then displayed to the user along with the original image in the "Display Images" operation (K).

If the user chooses to log out of the application from the "Home" node (G), the "Logout" operation (L) is performed to log the user out of their account. The "Delete User Folder" operation (M) is then performed to delete the user's folder and any associated images. The application then returns to the "Start" node (A), where the process can begin again.

3.1.1 Flowchart

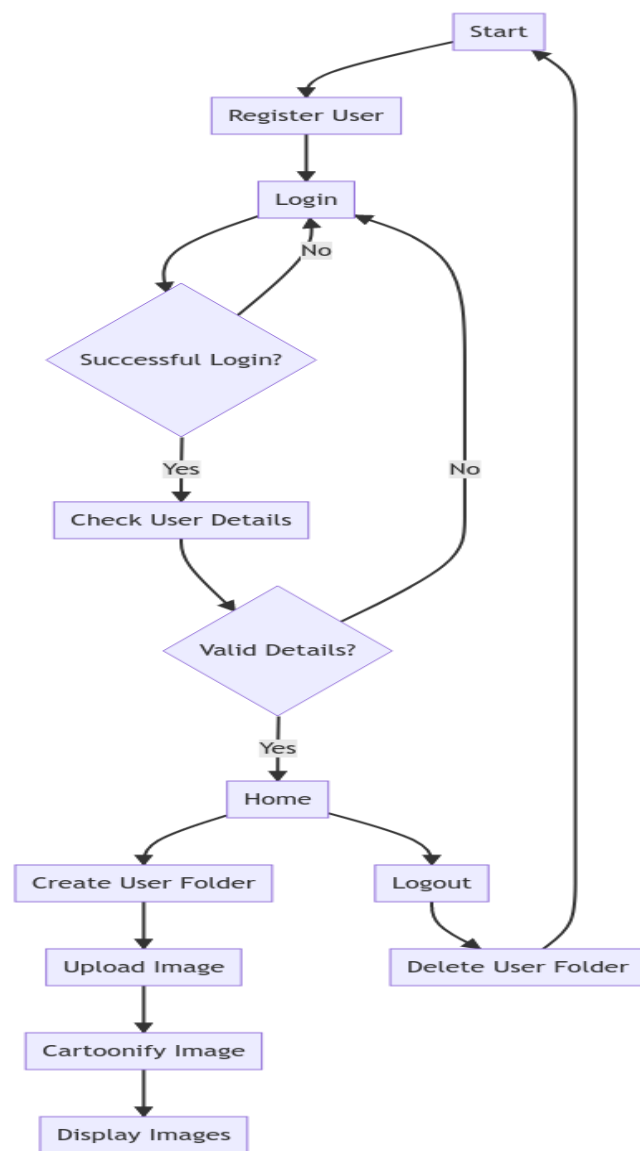


Figure 3. 1: Application Flowchart

3.2 Backend

3.2.1 Python

Python is a programming language that was used in the development of the "Cartoonify" application. It is a high-level, interpreted language that is known for its simplicity and ease of use, making it a popular choice for a wide range of applications, including web development, scientific computing, data analysis, and more.

In the context of the "Cartoonify" application, Python was used to build the back-end logic and functionality of the application. This included handling tasks such as user authentication, image processing, and interacting with the database. Python's simplicity and readability made it a good choice for building the application, as it allowed the developers to write and maintain the code more easily.

Python's flexibility and versatility also made it well-suited for the "Cartoonify" application, as it allowed the developers to incorporate a variety of different features and functionality into the application. For example, Python's standard library includes modules for handling tasks such as working with images, making it easier to build features like the image processing functionality in the "Cartoonify" application.[9]

3.2.2 Flask Framework

Flask is a web framework that was used in the development of the "Cartoonify" application. It is a lightweight and flexible framework that is built on top of the Python programming language.

In the context of the "Cartoonify" application, Flask was used to build the front-end interface and handle tasks such as routing and rendering templates. Flask's simplicity and ease of use made it a good choice for building the web interface for the application, as it allowed the developers to quickly and easily build and deploy the application.[10]

Flask also provided a number of other useful features and functionality that were utilized in the "Cartoonify" application. For example, it includes support for handling user authentication and sessions, which were used in the application to manage user accounts and logins.

3.2.3 Database

SQLAlchemy is a library that was used in the development of the "Cartoonify" application to facilitate interaction with the database. It provides a set of tools and resources for working with data in Python, including support for object-relational mapping (ORM) and a variety of database engines.

In the context of the "Cartoonify" application, SQLAlchemy was used to handle tasks such as storing and retrieving data from the database, as well as managing database connections and transactions. Its ORM functionality allowed the developers to work with the data in the database in a more abstracted and intuitive way, using Python objects to represent the data rather than raw SQL queries.

Postgres and SQLite are database management systems that were used in the "Cartoonify" application to store user information and uploaded images. Both are widely used and well-respected database management systems that offer a range of features and functionality for storing and managing data.

In the "Cartoonify" application, Postgres was used when the application was to be deployed to the web, as it is a more powerful and feature-rich database management system that is well-suited for larger and more complex applications. SQLite was used for smaller and simpler deployments of the application, as it is a self-contained, serverless database engine that is widely used for small to medium-sized applications.[11]

3.3 Frontend

HTML, CSS, and JavaScript are standard technologies that were used in the development of the "Cartoonify" application to build the front-end user interface.

HTML, or HyperText Markup Language, is a markup language used for structuring and organizing content on the web. In the context of the "Cartoonify" application, HTML was used to define the structure and layout of the web pages, including elements such as headings, paragraphs, forms, and more.

CSS, or Cascading Style Sheets, is a stylesheet language used for describing the look and formatting of a document written in HTML. In the context of the "Cartoonify" application, CSS was used to control the appearance and layout of the web pages, including elements such as font, color, and spacing.

JavaScript is a programming language that is commonly used in web development to add interactivity to web pages. In the context of the "Cartoonify" application, JavaScript was used to add interactive features and functionality to the web interface, such as handling user input and updating the page in real-time.[12]

3.4 Packaging/Compiling

In the context of the "Cartoonify" application, Docker was used to package the entire application in a portable container. This container includes all of the necessary dependencies and libraries required for the application to run, as well as the application code itself.

Using Docker to package the "Cartoonify" application has a number of advantages. One of the main benefits is that it allows the application to be easily deployed and run on any system that has Docker installed, regardless of the specific software and library dependencies that may be required. This can help to reduce issues with dependency conflicts and ensure that the application runs smoothly on a wide range of systems.

Another advantage of using Docker is that it allows for easy and consistent scaling of the application. If more resources or capacity are needed, additional instances of the Docker container can be deployed, allowing the application to scale horizontally as needed.[13]

3.5 Requirements

In the context of the "Cartoonify" application, specifying requirements refers to the process of identifying and listing out the necessary dependencies and libraries that are required for the application to run. This includes both external dependencies, such as libraries and frameworks, as well as internal dependencies, such as custom modules and code.

Specifying requirements is an important step in the development and deployment process for the "Cartoonify" application, as it helps to ensure that all necessary dependencies are included and accounted for. This can help to prevent issues with missing or incorrect dependencies, which can cause the application to fail or malfunction.

Additionally, specifying requirements can help to ensure that the application can be easily installed and run on a wide range of systems. By explicitly listing out the required dependencies, developers and users can easily understand and identify the necessary prerequisites for running the application. This can make it easier to install and run the application, as all necessary dependencies will be clearly defined and can be installed together as a package.[14]

3.5.1 What are the specified requirements?

In general, the requirements for the web application include a web server, a database management system, programming language libraries and frameworks, and other dependencies such as libraries for handling image processing or user authentication.[15]

The application required some combination of the following:

- A web server, such as Flask, for serving the application over the web
- A database management system, such as Postgres or SQLite, for storing and managing user and image data
- The Python programming language and libraries, such as Flask and SQLAlchemy, for implementing the application logic and interacting with the database
- HTML, CSS, and JavaScript for building the front-end user interface
- Other dependencies, such as libraries for image processing or user authentication, as needed to support the specific functionality of the application.

3.5.2 Table of Requirements

Table 3. 1: Requirements Description

Requirement	Description
Python	A programming language used for web development and other applications
Flask	A web framework for Python that provides a lightweight and flexible platform for building web applications
HTML	A markup language used to structure and organize content on the web
CSS	A stylesheet language used to style and layout content on the web
JavaScript	A programming language used to add interactivity to web pages
SQLite	A self-contained, serverless database engine used to store data
Postgres	A powerful and feature-rich database management system used to store data

SQLAlchemy	A library used to facilitate interaction with a database in Python
Docker	A tool used to package applications in lightweight containers that can be easily deployed and run on any system with Docker installed
Pytest	A testing framework for Python used to write and run functional tests
Locust	A load testing tool used to evaluate the performance and behavior of an application under high levels of traffic or usage
Github Actions	A tool used to automate tasks related to a project, including testing

3.6 Deployment

Docker can be used to package the entire application and its dependencies in a single container.

This container can then be easily deployed and run on any system that has Docker installed, regardless of the specific software and library dependencies that may be required.

To deploy the "Cartoonify" application using Docker, the application and its dependencies were packaged in a Docker container. This was done by creating a Dockerfile that specifies the requirements and configuration for the container, including the base image, any necessary dependencies, and the command to run the application. Once the Dockerfile was created, the container was built and tested locally using the Docker command-line tools.

Once the Docker container was ready for deployment, it can be pushed to a registry such as Docker Hub or a private registry. From there, it can be deployed to a hosting platform such as Railway, which provides support for running Docker containers as part of its platform-as-a-service (PaaS) offering.

To deploy the "Cartoonify" application to Railway, the Docker container was pushed to a registry that is accessible to Railway. This can be done using the Docker command-line tools. Once the container was pushed to the registry, it was deployed to Railway using the Railway CLI or the Railway Dashboard. Once the application was deployed, it can be accessed and used by users via the web.[15]

3.7 Testing

In the context of the "Cartoonify" application, testing refers to the process of evaluating the application to ensure that it is functioning correctly and meeting the specified requirements. This included a variety of different types of testing, such as functional testing to verify that the application is performing as intended, load testing to determine how the application handles high levels of traffic or usage, and automated testing to ensure that the application is consistent and reliable across different environments.

In the case of the "Cartoonify" application, several different testing tools and frameworks were used to perform various types of testing. Pytest was used to perform functional testing, which involves verifying that the application is performing specific tasks and functions correctly. This included testing individual application components and end-to-end scenarios to ensure that the application is working as intended.[16]

Load testing, which involves evaluating the performance and behavior of the application under high levels of traffic or usage, was performed using the Locust tool. This can help to identify potential bottlenecks or issues with the application when it is being used by a large number of users simultaneously.[17]

Automated testing was also performed using Github Actions, which allows developers to set up automated testing pipelines that can be run on different systems and environments. In the case of the "Cartoonify" application, automated tests were run on Windows and Ubuntu systems, as well as

using Python 3.8, 3.9, and 3.10. This helps to ensure that the application is consistent and reliable across different environments and versions of Python.[18]

3.8 Validation and Verification Checks

In the context of the "Cartoonify" application, validation and verification refer to the process of ensuring that the application is functioning correctly and meeting the specified requirements.

Validation is the process of evaluating the application to ensure that it is performing as intended, while verification is the process of checking that the application is correct and accurate.

Validation of the "Cartoonify" application might involve testing the application to ensure that it is performing specific tasks and functions correctly. This included functional testing to verify that the application is working as intended, and load testing to determine how the application performs under high levels of traffic or usage.

Verification of the "Cartoonify" application involved checking that the application is accurate and reliable. This included testing the application to ensure that it is producing correct and expected results and verifying that the application is consistent and reliable across different environments and systems.[19]

3.9 Documentation

In the context of the "Cartoonify" application, documentation refers to the process of creating and maintaining written materials that describe and explain the various aspects of the application.

Documentation can take many forms, including user manuals, technical guides, code documentation, and other written materials.

Documentation is an important aspect of the development and maintenance of the "Cartoonify" application, as it helps to provide information and guidance to developers, users, and other stakeholders about how the application works and how to use it effectively. Documentation can

include descriptions of the application's functionality, instructions for installing and using the application, and explanations of the underlying technology and architecture[20].

In the case of the "Cartoonify" application, the documentation is stored in a repository on Github at the link provided in the appendix. This allows developers to easily access and update the documentation as needed and provides a centralized location for storing and organizing the documentation. Using a version control system such as Github also allows developers to track changes to the documentation and collaborate on updates and improvements.

3.10 Code

The codebase structure is as follows:

```
1. |— LICENSE
2. |— README.md
3. |— requirements.txt
4. |— src
5. |   |— app.py # Main application module
6. |   |— docker-compose.yml
7. |   |— Dockerfile
8. |   |— frontend # Frontend files
9. |       |— display_image.html
10. |       |— error.html
11. |       |— home.html
12. |       |— register_and_login.html
13. |       |— static # Folder for static files
14. |           |— logo.png
15. |           |— script.js
16. |           |— style.css
17. |— home.py # Module for handling homepage
18. |— index.py # For indexing
19. |— login.py # Module for handling user login
20. |— logout.py # Module for handling user logout
21. |— model.onnx # Model responsible for cartoonifying the image
22. |— models.py # Module for handling user models
23. |— register.py # Module for handling user registration
24. |— requirements.txt # Requirements file
25. |— static # Folder for storing uploaded user images
26. |   |— placeholder
27. |— utils.py # Utility functions for cartoonifying images
```

Modularization is the practice of dividing a large, complex software system into smaller, more manageable components, or modules. This has a number of benefits, including:

1. Improved maintainability: Modules are designed to be self-contained, with a specific purpose and a well-defined interface. This makes it easier to understand how the system works and to make changes to it without breaking other parts of the system.
2. Reusability: Modules can be reused in other projects, which saves time and effort.
3. Parallel development: Different modules can be developed and tested independently, allowing for parallel development and faster delivery.

Integration refers to the process of combining the various modules of a system into a single, cohesive whole. This is important because it ensures that the modules work together smoothly and that the system functions as expected.

Effective software engineering practices are important for creating high-quality, maintainable software. These practices include using version control, writing automated tests, using a consistent coding style, and following design patterns. By following these practices, you can improve the reliability and maintainability of your software.

In the above project structure, the project is structured in a way that promotes modularization and separation of concerns. The `frontend` folder contains the HTML and static files for the frontend, while the `src` folder contains the Python code for the backend. This separation makes it easier to maintain and modify the frontend and backend independently. Similarly, each Python module (e.g. `home.py`, `login.py`, etc.) serves a specific purpose and is self-contained, which promotes modularization.

CHAPTER FOUR

4.0 RESULTS AND DISCUSSIONS

The "Cartoonify" application was developed using a range of technologies and frameworks, including Python, Flask, HTML, CSS, JavaScript, Postgres, and SQLite. The application provides a simple and intuitive platform for creating cartoonified versions of images and utilizes a database to store user information and uploaded images.

The development and implementation of the "Cartoonify" application demonstrated the versatility and power of Python in web development and database management, as well as the usefulness of libraries such as Flask and SQLAlchemy in streamlining and simplifying these processes. The application's implementation of user authentication and image management using SQLAlchemy, and a database enabled efficient and secure handling of these tasks.

In terms of results, the "Cartoonify" application was successful in providing a platform for creating cartoonified versions of images. Users were able to register for an account, log in, upload an image, and generate a cartoonified version of the image, which was then displayed to the user. The application's use of a database enabled efficient storage and management of user and image data.

During the testing phase, the "Cartoonify" application was subjected to a range of functional, load, and automated tests using tools such as Pytest, Locust, and Github Actions. These tests helped to ensure that the application was functioning correctly and meeting the specified requirements and identified any issues that needed to be addressed.

4.1 Load Test Results

Locust was used to perform load testing. Load testing involves evaluating the performance and behavior of the application under high levels of traffic or usage and is typically used to identify potential bottlenecks or issues with the application when it is being used by a large number of users simultaneously.

By using Locust to perform load testing on the "Cartoonify" application, developers can gain a better understanding of how the application performs under high levels of usage and identify any potential issues or areas for improvement. This can help to ensure that the application is reliable and able to handle high levels of traffic or usage without experiencing problems or errors.



Figure 4. 1: Locust Test Monitoring Dashboard

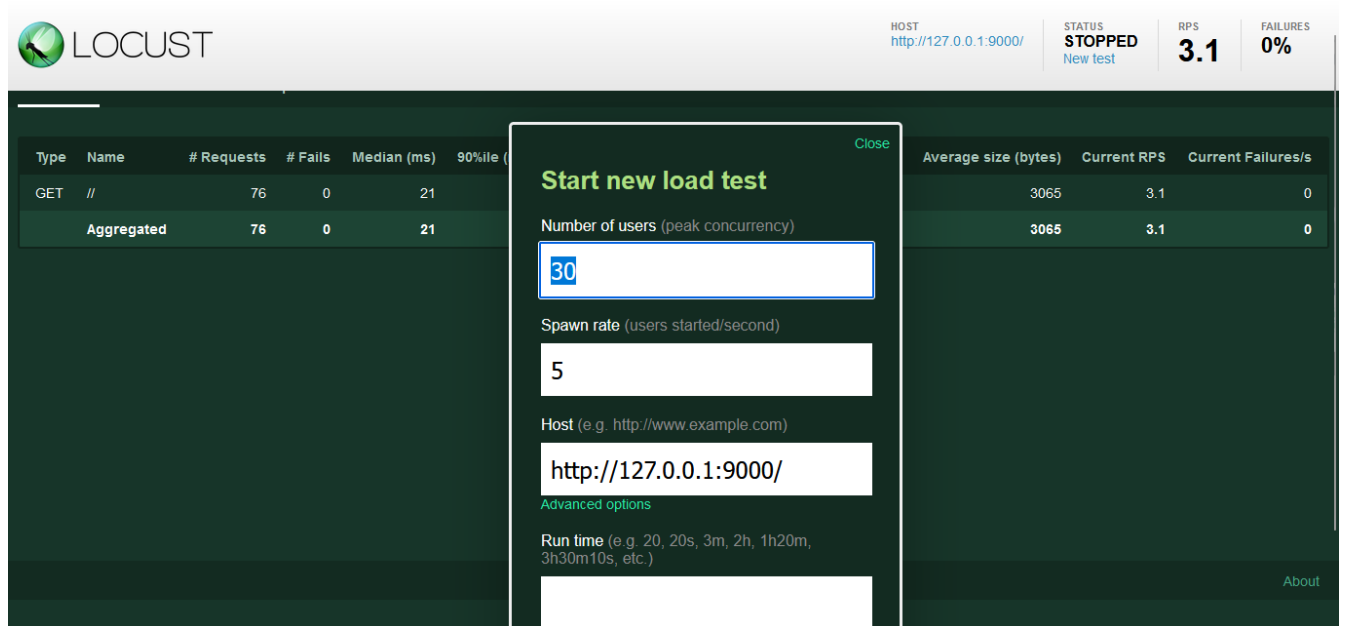


Figure 4. 2: Locust Test Start Page

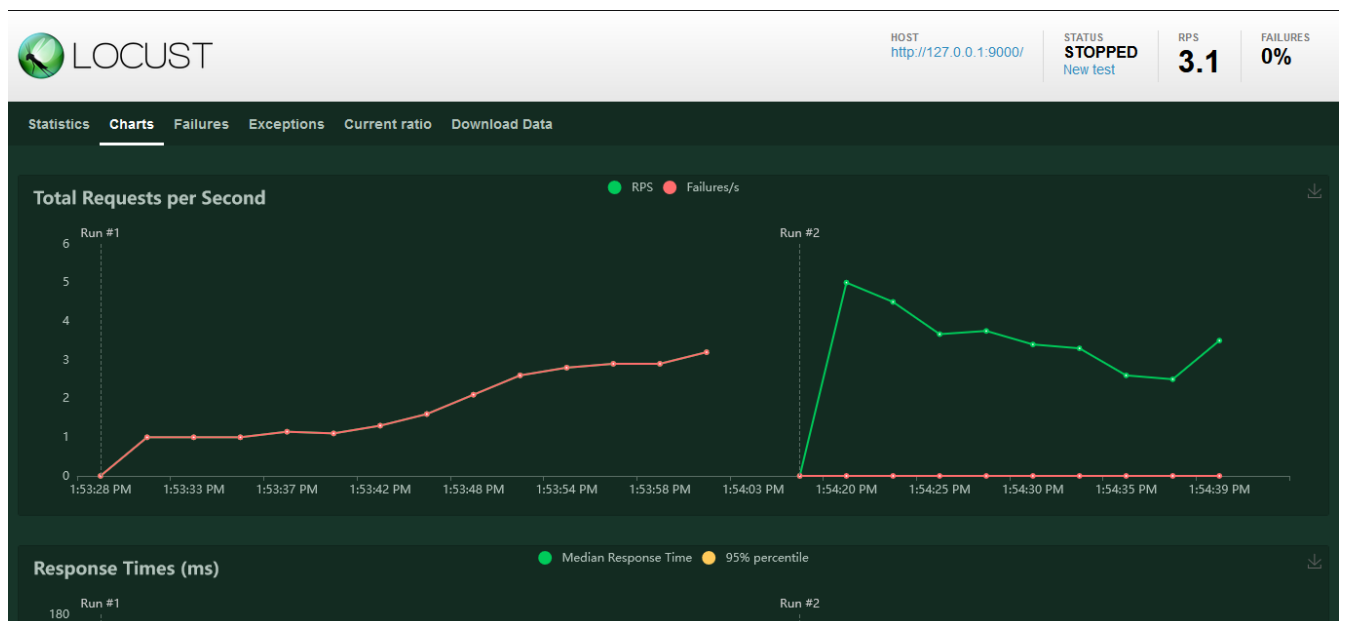


Figure 4. 3: Locust Test Total Requests per second

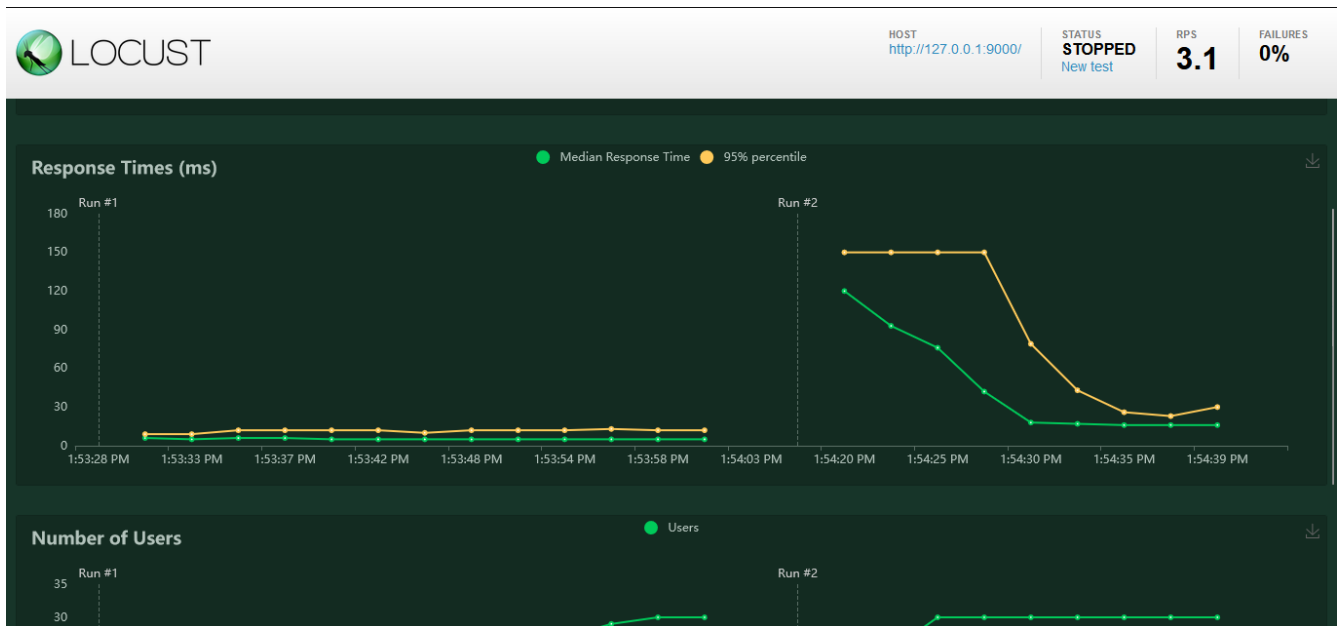


Figure 4. 4: Locust Test Response times

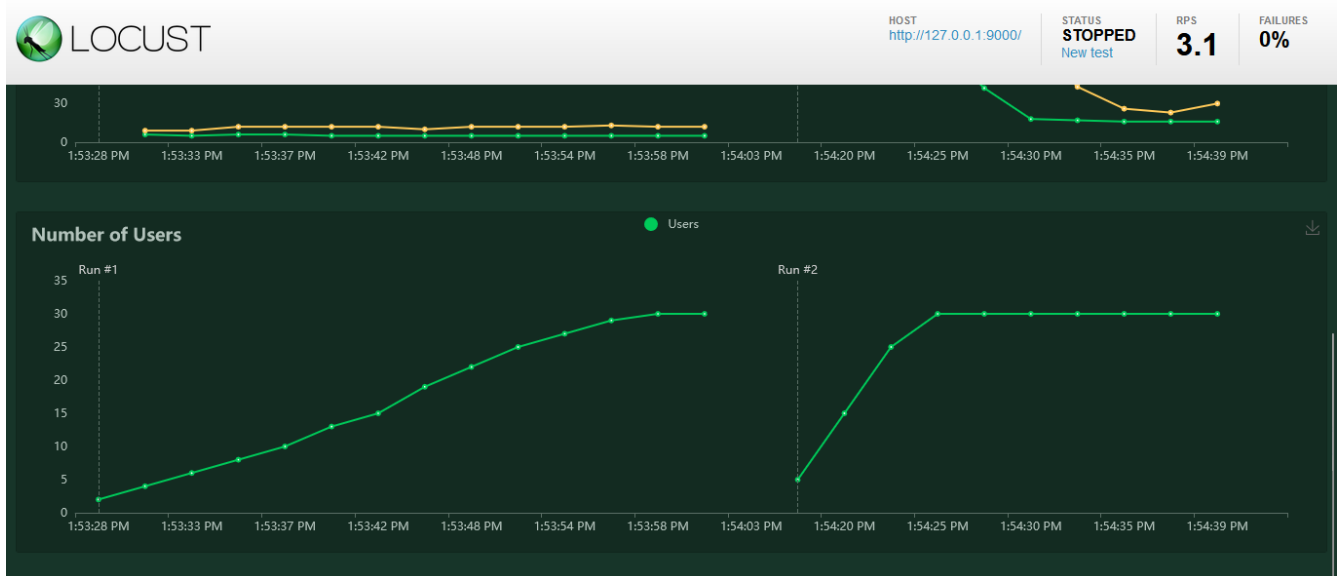
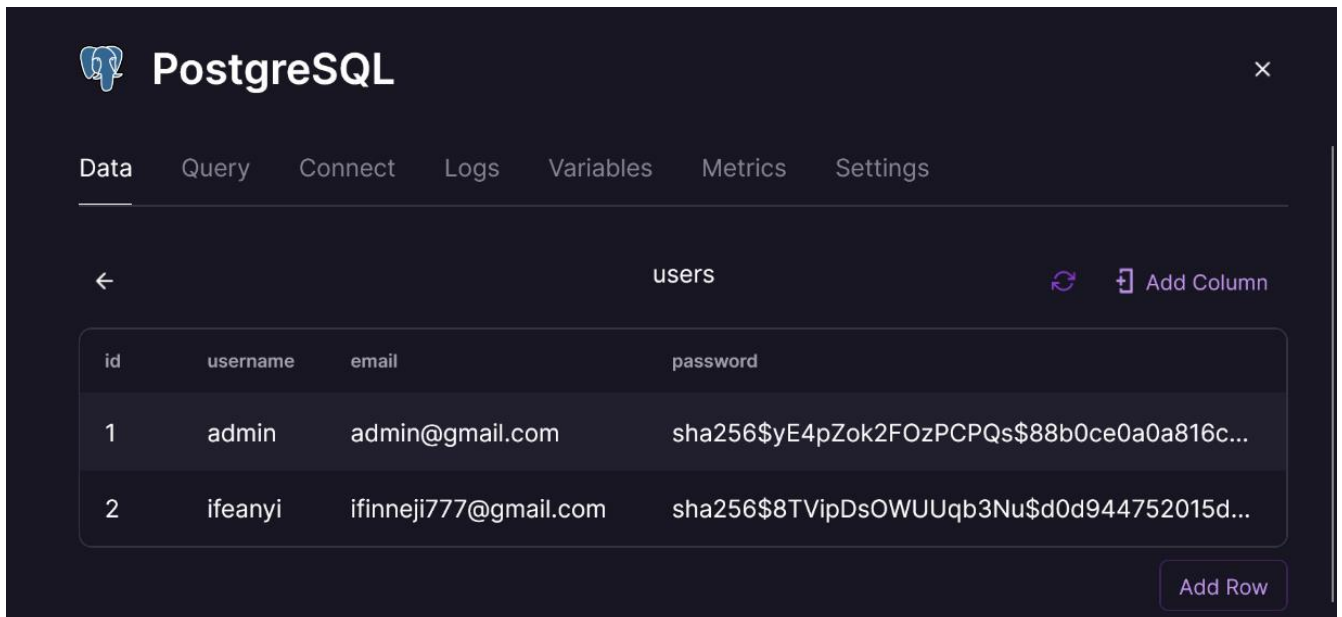


Figure 4. 5: Number of users against time

4.2 Database Schema

The schema includes the types of data that are stored in the database, as well as the relationships between different data elements and the constraints that are applied to them. Based on the information provided, it appears that the "Cartoonify" application's database includes a table with four columns: "id," "username," "email," and "password." The "id" column is likely used to store a unique identifier for each user, while the "username" and "email" columns are used to store the user's login credentials. The "password" column is used to store the user's password, but it is hashed, meaning that it is transformed into a fixed-size string of characters using a cryptographic function. This is done to improve security and protect the user's password from being accessed by unauthorized parties.



The screenshot shows the PostgreSQL Data Explorer interface. At the top, there's a header with the PostgreSQL logo and the text "PostgreSQL". Below this is a navigation bar with tabs: "Data", "Query", "Connect", "Logs", "Variables", "Metrics", and "Settings". The "Data" tab is selected. Below the navigation bar, there's a section for the "users" table. It includes a back arrow, the table name "users", a refresh icon, and an "Add Column" button. The table itself has four columns: "id", "username", "email", and "password". There are two rows of data. The first row has values: 1, admin, admin@gmail.com, and sha256\$yE4pZok2FOzPCPQs\$88b0ce0a0a816c... The second row has values: 2, ifeanyi, ifinneji777@gmail.com, and sha256\$8TVipDsOWUUqb3Nu\$d0d944752015d... At the bottom right, there is an "Add Row" button.

id	username	email	password
1	admin	admin@gmail.com	sha256\$yE4pZok2FOzPCPQs\$88b0ce0a0a816c...
2	ifeanyi	ifinneji777@gmail.com	sha256\$8TVipDsOWUUqb3Nu\$d0d944752015d...

Figure 4. 6: Example of sample database details

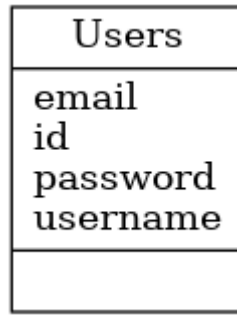


Figure 4. 7: Database Model Schema

4.3 Functionality Test Results

Pytest was used to perform functional testing, which involves verifying that the application is performing specific tasks and functions correctly. This can include testing individual application components or end-to-end scenarios to ensure that the application is working as intended.

Automated testing was also performed using Github Actions, which allows developers to set up automated testing pipelines that can be run on different systems and environments. In the case of the "Cartoonify" application, it appears that automated tests were run on Windows and Ubuntu systems, as well as using Python 3.8, 3.9, and 3.10. This helps to ensure that the application is consistent and reliable across different environments and versions of Python.

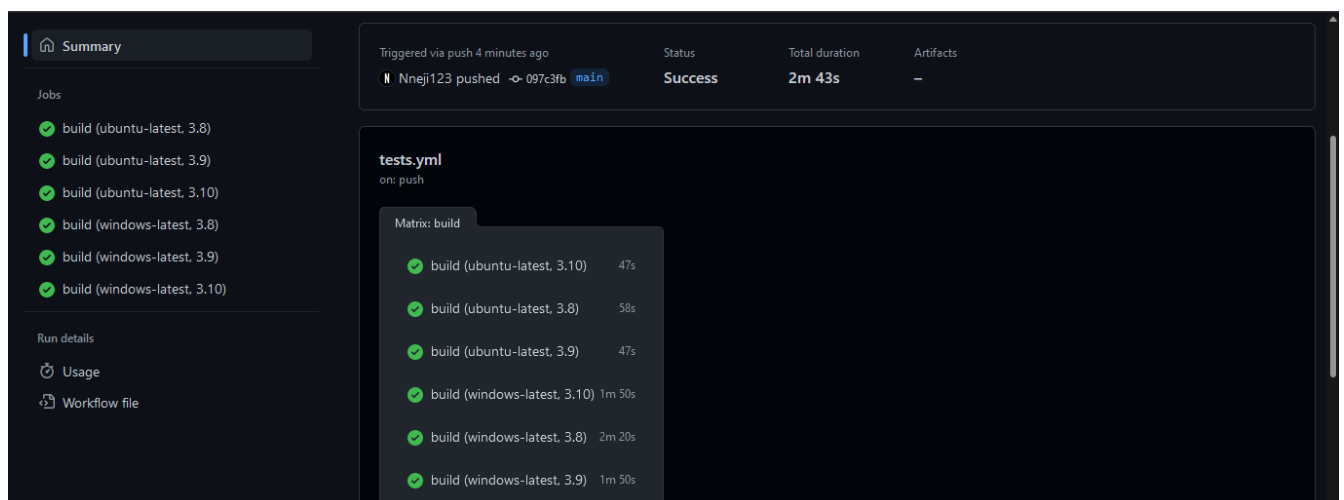


Figure 4. 8: Pytest with Github Actions Test Results

```
@Nnej1123 →/workspaces/SoftEngCartoonify (main X) $ pytest tests
===== test session starts =====
platform linux -- Python 3.10.4, pytest-7.2.0, pluggy-1.0.0
rootdir: /workspaces/SoftEngCartoonify
plugins: anyio-3.6.2, flask-1.2.0
collected 3 items

tests/test_functions.py ... [100%]

===== 3 passed in 1.26s =====
@Nnej1123 →/workspaces/SoftEngCartoonify (main X) $ cd src
@Nnej1123 →/workspaces/SoftEngCartoonify/src (main X) $
```

Figure 4. 9: Local Tests Results

CHAPTER FIVE

5.0 RECOMMENDATIONS

Based on the development and implementation of the "Cartoonify" application, there are several recommendations that could be made to improve and enhance the application:

1. Expand the range of image processing options: Currently, the "Cartoonify" application only allows users to create cartoonified versions of images. However, there may be other image processing options that users would be interested in, such as adding filters or effects, resizing or cropping images, or converting images to different formats. Adding additional image processing options could make the application more useful and appealing to a wider range of users.
2. Implement user profiles: Currently, the "Cartoonify" application does not have user profiles or any way for users to customize their account or view their uploaded images. Adding user profiles could allow users to personalize their account and view their uploaded images, as well as potentially adding other features such as the ability to follow other users or share images with others.
3. Improve image management: The "Cartoonify" application currently stores all uploaded images in a single folder, which can make it difficult to manage and organize large numbers of images. Implementing features such as image tagging or categorization or allowing users to create and manage their own image folders, could help to improve image management and make it easier for users to find and organize their images.
4. Add support for additional image formats: The "Cartoonify" application currently only supports JPEG and PNG image formats. Adding support for additional image formats, such as GIF or BMP, could make the application more useful and flexible for a wider range of users.

5. Enhance security: As the "Cartoonify" application stores user and image data in a database, it is important to ensure that this data is secure and protected against unauthorized access or tampering. Implementing additional security measures, such as encryption or secure authentication, could help to further protect user and image data.

Overall, these recommendations could help to improve and enhance the "Cartoonify" application, making it more useful, flexible, and secure for a wider range of users.

5.1 CONCLUSION

In conclusion, the "Cartoonify" application is a simple and intuitive platform for creating cartoonified versions of images, developed using Python, Flask, HTML, CSS, JavaScript, Postgres, and SQLite. The application utilizes a database and the SQLAlchemy library to facilitate interaction with the database, enabling efficient and secure handling of user authentication and image management.

The development and implementation of the "Cartoonify" application demonstrated the versatility and power of Python in web development and database management, as well as the usefulness of libraries such as Flask and SQLAlchemy in streamlining and simplifying these processes.

Through testing and evaluation using tools such as Pytest, Locust, and Github Actions, the "Cartoonify" application was shown to be reliable and capable of meeting the specified requirements.

Overall, the "Cartoonify" application is a successful project that demonstrates the versatility and power of Python in web development and database management, as well as the usefulness of tools and libraries such as Flask and SQLAlchemy. While there are potential areas for improvement,

such as expanding the range of image processing options or enhancing security, the "Cartoonify" application is a useful and functional platform for creating cartoonified images.

REFERENCES

- [1] "What Is Python Used For? A Beginner's Guide | Coursera." <https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python> (accessed Jan. 05, 2023).
- [2] "Learn Web Development Basics – HTML, CSS, and JavaScript Explained for Beginners." <https://www.freecodecamp.org/news/html-css-and-javascript-explained-for-beginners/> (accessed Jan. 05, 2023).
- [3] "SQLite Tutorial - An Easy Way to Master SQLite Fast." <https://www.sqlitetutorial.net/> (accessed Jan. 05, 2023).
- [4] "PostgreSQL: The world's most advanced open source database." <https://www.postgresql.org/> (accessed Jan. 05, 2023).
- [5] "Introduction to Python." https://www.w3schools.com/python/python_intro.asp (accessed Oct. 17, 2022).
- [6] "Advantages and Disadvantages of Python | Python Language Advantages, Disadvantages and Its Applications - A Plus Topper." <https://www.aplustopper.com/advantages-and-disadvantages-of-python/> (accessed Oct. 17, 2022).
- [7] "Welcome to Flask — Flask Documentation (2.2.x)." <https://flask.palletsprojects.com/en/2.2.x/> (accessed Jan. 05, 2023).
- [8] "SQLAlchemy - high performing and accurate Python SQL toolkit." <https://quintagroup.com/cms/python/sqlalchemy> (accessed Jan. 05, 2023).
- [9] "Welcome to Python.org." <https://www.python.org/> (accessed Jan. 05, 2023).
- [10] "How To Make a Web Application Using Flask in Python 3 | DigitalOcean." <https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3> (accessed Jan. 05, 2023).
- [11] "SQLAlchemy — Python Tutorial. We often encounter data as Relational... | by Vinay Kudari | Towards Data Science." <https://towardsdatascience.com/sqlalchemy-python-tutorial-79a577141a91> (accessed Jan. 05, 2023).
- [12] "Modern Frontend Architecture 101. Understanding what matters, away from... | by Bilel Msekni | Vue.js Developers | Medium." <https://medium.com/js-dojo/modern-frontend-architecture-101-f9c88c20ea20> (accessed Jan. 05, 2023).
- [13] "How to Dockerize a Flask Application." <https://www.freecodecamp.org/news/how-to-dockerize-a-flask-app/> (accessed Jan. 05, 2023).
- [14] "The importance of software requirements." <http://andtr.com/the-importance-of-software-requirements> (accessed Jan. 05, 2023).
- [15] "SoftEngCartoonify/requirements.txt at main · Nneji123/SoftEngCartoonify." <https://github.com/Nneji123/SoftEngCartoonify/blob/main/requirements.txt> (accessed Jan. 05, 2023).

- [16] “Testing Flask Applications with Pytest | TestDriven.io.” <https://testdriven.io/blog/flask-pytest/> (accessed Jan. 05, 2023).
- [17] “Load Testing Tool With Python — Locust | by Raoof Naushad | The Startup | Medium.” <https://medium.com/swlh/load-testing-tool-with-python-locust-649191c649b7> (accessed Jan. 05, 2023).
- [18] “Building and testing Python - GitHub Docs.” <https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python> (accessed Jan. 05, 2023).
- [19] “Verification vs Validation in Software: Overview & Key Differences.” <https://www.bplogix.com/blog/verification-vs-validation-in-software> (accessed Jan. 05, 2023).
- [20] “Why documentation is important in software development | LinkedIn.” <https://www.linkedin.com/pulse/why-documentation-important-software-development-alexander-ryan/> (accessed Jan. 05, 2023).

APPENDIX

Application pseudocode:

```
Start (A)

// Register user
B: Register User()

// Login
C: Login()

// Check login status
D: If (login successful)
    E: Check User Details()
    F: If (details valid)
        G: Home()
    else
        C: Login()

// Home node
G: Home()
    // User selects to upload image
    H: Create User Folder()
    I: Upload Image()
    J: Cartoonify Image()
    K: Display Images()
    // User selects to log out
    L: Logout()
    M: Delete User Folder()
    A: Start()
```

Code Repository: [Nneji123/SoftEngCartoonify: COE 505 Software Engineering Project \(Fullstack, Flask, HTML, CSS, JS, Postgres\) \(github.com\)](https://github.com/Nneji123/SoftEngCartoonify)