Table of Contents:

1. Introduction
2. Description of Your Website
3. Website Design
4. Web Application

4.1. Technologies and Languages Used

4.2. Database Design

4.3. User Interface

4.4. Challenges Encountered

1. Secure Web Application

5.1. User Authentication

5.2. Data Validation

5.3. Other Security Measures

1. References
2. Appendix

# 1. Introduction:

The purpose of this report is to document the development and implementation of a secure web application for a to-do list. The objectives of this project are to design and implement a web application that allows users to create, read, update, and delete tasks, while ensuring the security and integrity of user data.

# 2. Description of Your Website:

The proposed web application is a simple to-do list that allows users to manage their tasks and stay organized. The application provides a user-friendly interface for creating, reading, updating, and deleting tasks, as well as viewing all tasks on a single page. Users can create a new account or log in to an existing account to access their tasks.

# 3. Website Design:

The website design consists of a simple and intuitive layout, with a navigation bar at the top of the page that allows users to access different parts of the application. The user interface is designed to be easy to use, with clear labels and buttons for each action. The website design has some flaws, such as the lack of responsiveness for smaller screens, but overall it provides a clear and straightforward user experience.

# 4. Web Application Implementation:

## 4.1. Technologies and Languages Used:

* Flask web framework
* SQLAlchemy ORM
* SQLite database
* HTML, CSS, and JavaScript for the front-end

## 4.2. Database Design:

The database design consists of a single table, **todo**, with columns for **id**, **title**, **due\_date**, **complete**, and **user\_id**. The **id** column is an auto-incrementing integer that uniquely identifies each task. The **title** column is a string that stores the title of the task. The **due\_date** column is a date that stores the due date of the task. The **complete** column is a boolean that stores whether the task is complete or not. The **user\_id** column is an integer that associates the task with the user who created it.

## 4.3. User Interface:

The user interface consists of a simple layout with a navigation bar at the top of the page. The navigation bar contains links to the home page, login page, registration page, and logout page. The home page displays a table of all tasks, with columns for **title**, **due\_date**, and **complete**. Each row in the table has buttons for editing and deleting the task. The login and registration pages allow users to log in or create a new account. The logout page logs the user out of their account.

## 4.4. Challenges Encountered:

During the implementation of the web application, several challenges were encountered. One challenge was integrating user authentication with the Flask framework. This was overcome by using Flask-Login, which provides secure session management and password hashing. Another challenge was managing database transactions. This was overcome by using SQLAlchemy's session management features.

# 5. Secure Web Application Implementation:

## 5.1. User Authentication:

To ensure the security and integrity of user data, user authentication was implemented using Flask-Login. Flask-Login provides secure session management and password hashing, which ensures that user passwords are stored securely in the database.

## 5.2. Data Validation:

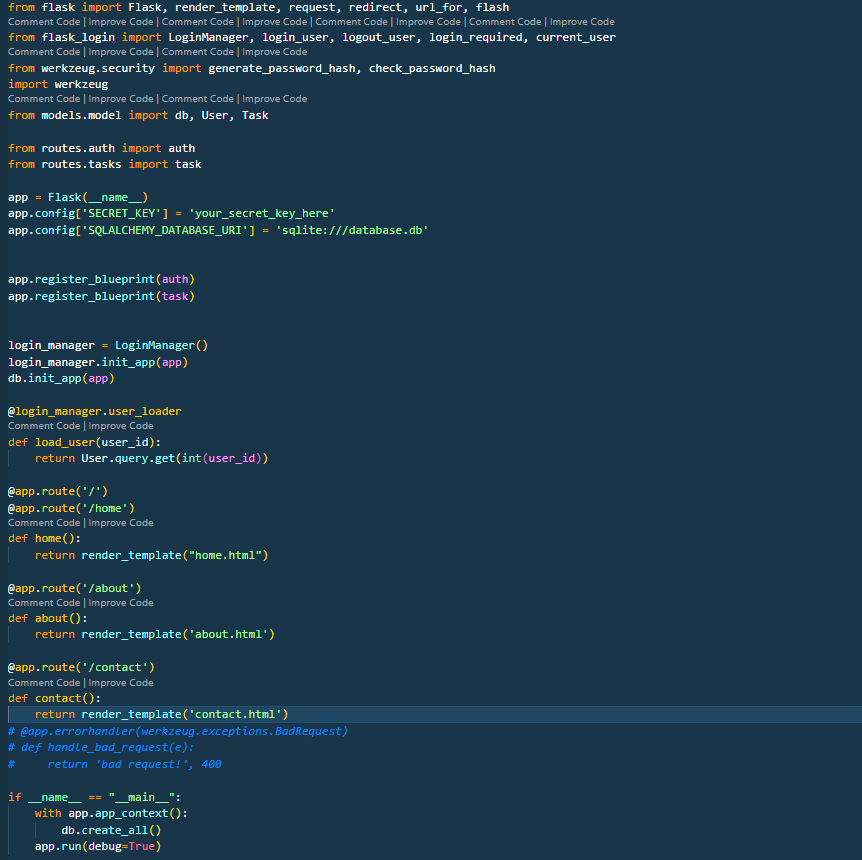
To ensure that user input meets certain criteria before it is saved to the database, data validation was implemented using SQLAlchemy's built-in data validation features. These features ensure that user input meets certain criteria, such as being a valid date or a non-empty string.

# 6. References

* Flask documentation: <https://flask.palletsprojects.com/en/2.1.x/>
* SQLAlchemy documentation: <https://docs.sqlalchemy.org/en/14/>
* Flask-Login documentation: https://flask-login.readthedocs.io/en/latest/

# 7. Appendix

## 7.1 Screenshots

App.py 

Models.py

A screen shot of a computer program

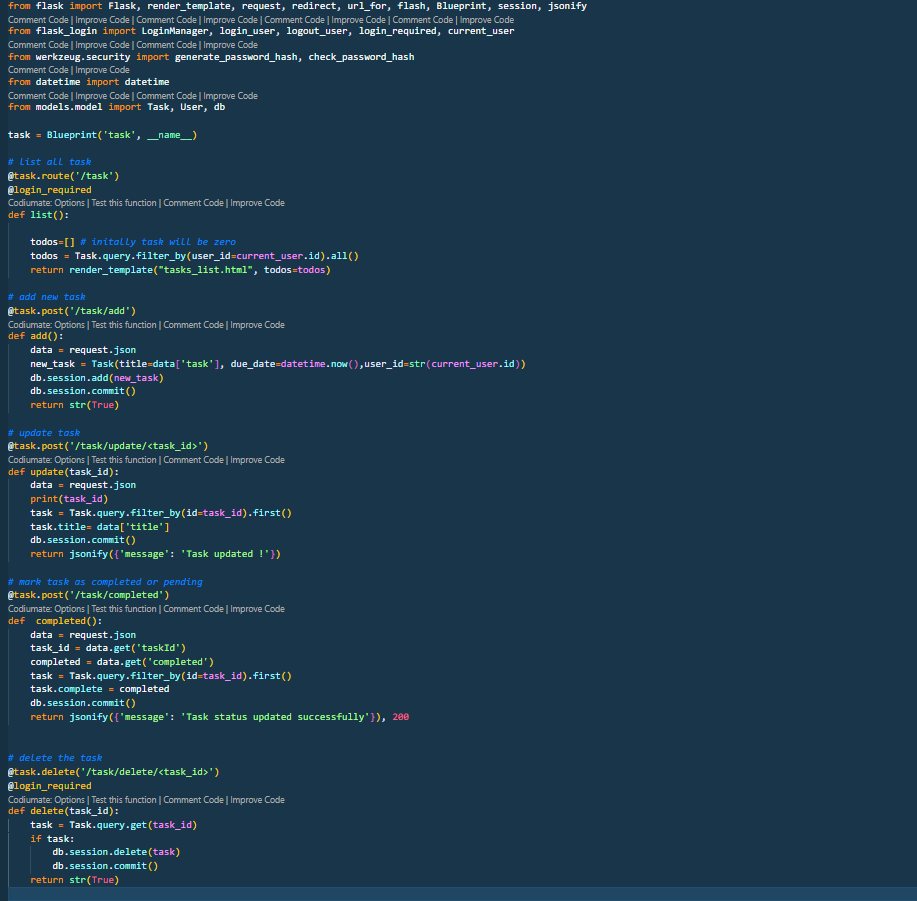
Description automatically generated

Auth.py

A screen shot of a computer code

Description automatically generated

Tasks.py



Base.html

