**ABSTRACT:**

This project focuses on the development of a To-Do application using Flask, a lightweight and flexible Python web framework. The To-Do application allows users to manage their tasks efficiently by providing features such as creating, updating, deleting, and marking tasks as completed.

**INTRODUCTION:**

In the fast-paced world we live in, it's easy for tasks and commitments to slip through the cracks. That's where Flask Todo comes in – your ultimate companion for staying organized and on top of your to-do list.

Flask Todo is a simple yet powerful web application built using Flask, a lightweight and flexible Python web framework. With Flask Todo, you can effortlessly manage your tasks, prioritize your work, and increase your productivity.

**LITERATURE SURVEY:**

* + *Flask Documentation: The official Flask documentation provides comprehensive guides and tutorials for building web applications using Flask. It covers topics such as routing, templates, forms handling, and database integration, which are essential for building a to do application.*
  + *Flask Mega-Tutorial by Miguel Grinberg: This is an extensive tutorial covering various aspects of web development with Flask, including user authentication, database integration with SQL Alchemy, and building RESTful APIs. It can be a valuable resource for understanding how to structure a Flask application for a to do list project.*
  + *SQL Alchemy Documentation: Since most to do applications involve interacting with a database to store tasks, understanding SQL Alchemy for database integration is crucial. The SQL Alchemy documentation provides in-depth explanations and examples for working with databases in Flask applications.*
  + *Bootstrap Documentation: Bootstrap is a popular front-end framework for building responsive and visually appealing web interfaces. Integrating Bootstrap with Flask can enhance the UI of the to do application. The Bootstrap documentation offers guidance on using its components and CSS classes.*
  + *JavaScript and AJAX: Implementing features such as asynchronous task updates or dynamic UI elements in the to do application may require knowledge of JavaScript and AJAX. Resources like MDN Web Docs and JavaScript: The Definitive Guide by David Flanagan can be helpful for learning these technologies.*
  + *RESTful API Design: If you plan to extend your to do application to support mobile or desktop clients, designing a RESTful API can be beneficial. Resources like "RESTful Web APIs" by Leonard Richardson and Mike Amundsen provide insights into best practices for designing RESTful APIs.*
  + *Security Best Practices: Security is crucial when building web applications. Resources like "Web Security Basics" by Mozilla Developer Network and "Flask-Security" documentation can help you understand common security threats and how to mitigate them in your Flask application.*
  + *Deployment: Once your to do application is ready, deploying it to a production server is necessary to make it accessible to users. Guides such as the Flask deployment documentation and tutorials on platforms like Heroku or AWS can assist in deploying Flask applications.*