**Software Design and Engineering**

In this project, the essence of software design and engineering was exemplified through a meticulous process that aimed at conceptualizing, planning, and constructing software applications using established engineering principles and methodologies. The primary objective was to craft a software system of superior quality, characterized by reliability and maintainability.

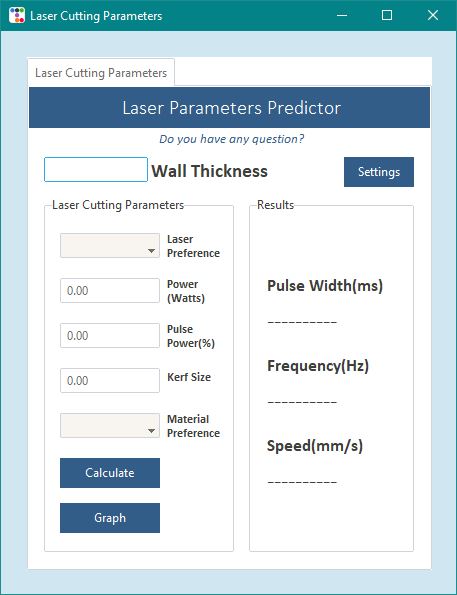
The initial phase involved designing various aspects of the software architecture, including components, modules, interfaces, and data structures. This meticulous design process ensured that the software would fulfill specific requirements while being scalable, efficient, and adaptable to future changes.

Furthermore, the project delved into creating a visually appealing and user-centric interface, emphasizing the importance of user experience design. This involved not only focusing on the visual aesthetics of the interface but also understanding how users interacted with it to enhance usability and overall satisfaction.

An essential aspect of the project was the implementation of a database system to store user information securely. This database not only facilitated user management but also controlled access to the application, ensuring that only authorized users could utilize the software.

Moreover, the project showcased the utilization of algorithms and data structures to manage and process data effectively. Specifically, these algorithms were employed to obtain optimal values for laser parameters, indicating a practical application of these computational tools within the software system.

Throughout the development process, best practices, coding standards, and rigorous testing strategies were integrated to guarantee the software's functionality, performance, and security. These measures were crucial in maintaining the software's robustness and reliability over time, reflecting a comprehensive approach to software design and engineering in the context of this project.



Executed the algorithms and activates the system to display results.

Selects available materials to cut.

Desired kerf size

Desired percentage of the Laser power

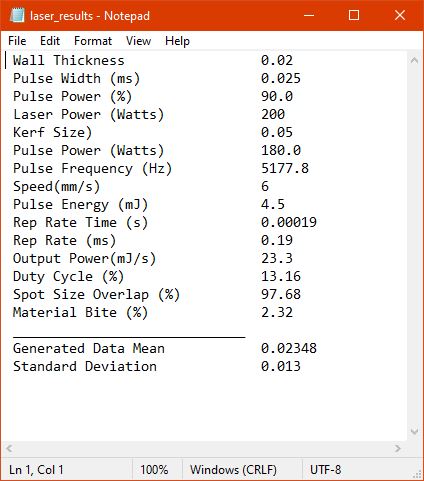
This auto fills when a laser preference is selected. It displays the maximum power of the selected laser.

This preference makes it easy in case of multiple machines.

This cannot be left blank. All algorithms run off this value.

This will display partial results once the “Calculate” button is clicked.

‘Detailed Results



**Databases**

A database is an organized data collection for easy access, management, and updates, acting as a central store ensuring data integrity, security, and consistency through tables, rows, and columns. Users interact via DBMS to query, add, modify, or delete data. They vary from relational (SQL) to NoSQL and NewSQL types. Crucial in organizations, databases handle vast data and support various applications, from simple websites to complex systems. Through prior courses, I grasped database importance, creating, reading, editing, and deleting data. In this project, I am emphasizing user control with an authentication page, granting access if user info matches database records. Utilizing SQLite, Python was employed to manage user storage and develop queries effectively.

# Create account code block  
def create\_account():  
 username = entry\_username.get()  
 password = entry\_password.get()  
  
 # Connect to the database  
 conn = sqlite3.connect("user\_db.sqlite")  
 cursor = conn.cursor()  
  
 # Create the "users" table if it doesn't exist  
 cursor.execute("""  
 CREATE TABLE IF NOT EXISTS users (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 username TEXT NOT NULL,  
 password TEXT NOT NULL  
 )""")  
  
 # conn.close()  
  
 conn = sqlite3.connect("user\_db.sqlite")  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM users WHERE username=?", (username,))  
 existing\_user = cursor.fetchone()  
  
 if existing\_user:  
 messagebox.showerror("Account Creation Failed", "Username already exists")  
 else:  
 cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)", (username, password))  
 conn.commit()  
 messagebox.showinfo("Account Created", "Your account has been created successfully!")  
  
 conn.close()  
  
  
# This code block will delete account information from the database  
def delete\_account():  
 username = entry\_username.get()  
  
 conn = sqlite3.connect("user\_db.sqlite")  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM users WHERE username=?", (username,))  
 existing\_user = cursor.fetchone()  
  
 # This checks to find if the user information exists in the database.  
 if existing\_user:  
 cursor.execute("DELETE FROM users WHERE username=?", (username,))  
 conn.commit()  
 messagebox.showinfo("Account Deleted", "Your account has been deleted successfully!")  
 else:  
 messagebox.showerror("Account Deletion Failed", "Invalid username")  
  
 conn.close()  
  
  
# This block of code checks to find if the user information exists in the database.  
# and if it exists, the system will allow the user to access the main application  
def authenticate\_user():  
 username = entry\_username.get()  
 password = entry\_password.get()  
  
 conn = sqlite3.connect("user\_db.sqlite")  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM users WHERE username=? AND password=?", (username, password))  
 user = cursor.fetchone()  
  
 if user:  
 messagebox.showinfo("Login Successful", "Welcome, {}".format(username))  
 else:  
 messagebox.showerror("Login Failed", "Invalid username or password")  
  
 conn.close()  
  
  
# The block allows the user to update the user password  
def update\_user\_login():  
 username = entry\_username.get()  
 new\_password = entry\_password.get()  
  
 conn = sqlite3.connect("user\_db.sqlite")  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM users WHERE username=?", (username,))  
 existing\_user = cursor.fetchone()  
  
 if existing\_user:  
 cursor.execute("UPDATE users SET password=? WHERE username=?", (new\_password, username))  
 conn.commit()  
 messagebox.showinfo("Update Successful", "Your password has been updated successfully!")  
 else:  
 messagebox.showerror("Update Failed", "Invalid username")  
  
 conn.close()

**Algorithms and Data Structures**

From a previous academic course, I delved into the intricate realm of data structures, the foundational building blocks of computer science enabling the efficient storage, organization, and manipulation of data. These structures serve as the backbone for managing and processing data, allowing for seamless operations and accessibility. Among the array of common data structures are arrays, linked lists, stacks, queues, trees, and graphs, each with its unique characteristics and applicability, tailored to address specific computational challenges.

A profound comprehension of data structures is paramount for devising streamlined algorithms and crafting high-performing code. In a practical project I undertook, my proficiency in data structure management proved instrumental in predicting essential parameters for laser cutting metal sheets of varying thicknesses, ranging from 0.006 inches to 0.06 inches.

The system I worked on showcased a sophisticated workflow. When a user inputs a specific value in the Wall thickness entry, the system springs into action, generating a random dataset composed of 50 points uniformly distributed within the specified thickness range. This dataset's mean value is then computed and compared against the input wall thickness, setting the stage for insightful analysis.

Subsequently, the system iterates, creating a second dataset of 50 points, steered by the correlations unearthed between the datasets in prior steps. This correlation data serves as a critical input for a predictive model embedded within the system, ultimately culminating in the calculation and generation of the desired pulse width for laser cutting operations. Through this intricate process, the fusion of data structure principles and predictive modeling yields a robust system capable of optimizing metal cutting operations with precision and efficiency.

data1 = []  
with open('model\\modelData.txt') as f:  
 for line in f.readlines():  
 data1.append(float(line))  
  
data2 = []  
with open('model\\modelData2.txt') as f:  
 for line in f.readlines():  
 data2.append(float(line))  
  
meanData = statistics.mean(data1)  
  
pulseWidthSpi = round(predict\_model\_mean2, 3)  
  
# Using np.linspace() with num parameter  
x\_value = [x\_value / 1000 for x\_value in range(0, 50)]  
x\_value2 = [meanData]  
y\_value2 = [meanData]  
x\_value3 = [wall\_thickness]  
y\_value3 = [pulseWidthSpi]  
  
# y\_value = [x\_value / 1000 for x\_value in range(0, 50)]  
  
def data\_eng():  
 plt.close()  
 plt.scatter(x\_value, data1, s=10, c='teal')  
 plt.scatter(x\_value2, y\_value2, s=10, c='red')  
 plt.text(meanData, meanData, "mean", fontsize=8)  
 plt.scatter(x\_value3, y\_value3, s=10, c='red')  
 plt.text(wall\_thickness, pulseWidthSpi, "pulse width", fontsize=8)  
 plt.plot(x\_value, data2)  
 # Add correlation line  
  
 plt.title("Pulse Width Prediction", fontsize=10)  
 axes = plt.gca()  
 m, b = np.polyfit(x\_value, data1, 1)  
 X\_plot = np.linspace(axes.get\_xlim()[0], axes.get\_xlim()[1], 100)  
 plt.plot(X\_plot, m \* X\_plot + b, '-')  
 plt.xlabel('Wall Thickness(x)')  
 plt.ylabel('Pulse Width(y)')  
 plt.grid(alpha=.6, color='green', linestyle='--', linewidth=0.5)  
 plt.show()  
  
 # Mean of the predicting data  
  
# Standard deviation of the predicting data  
standard\_dev = round(np.std(data1), 3)  
# Confidence interval of the predicting data  
# confidenceI = interv  
  
design\_para\_btn = ttk.Button(laser\_frame, text='Graph', command=data\_eng)  
design\_para\_btn.place(x=15, y=290, height=30, width=100)

The graph displays the results.

