**Name: Dharmaraja Praveen Krishan**

**Student ID: 20231994**

**The stages of Software Development Lifecycle which I have used at coding.**

The SDLC has been used in the python coding at the different stages so they’re in my coding I have used all the stages and make it have final output.

**1.Planning:** The planning is the first part of the process so we’re reading the scenario and make gatherings of it. So, the first part is to make a class of Request system. And make an init method to initialize the **Date StaffID, Name and the status and the requisition ID** if a counter has been used. For examples:

**date = int(input("Enter the date"))**

**staffID = int(input("Enter the ID"))**

**2. Analysis**: this part is more important than analyzing the scenario and making sure that the functional and non-functional classes are identified. As in my scenario there are six classes, I have defined in my coding they are **staff info, request details, request approval, respond request, display request and request statistics.**

For example:

def user\_info(self):

name = input("Enter your name")

date = int(input("Enter the date"))

        staffID = int(input("Enter the ID"))

        requisitionID = input("Enter your requisitionID")

        self.user\_details = {'name': name, 'date': date, 'staffID': staffID, 'requisitionID': requisitionID}

print("user information collected successfully.\n")

**3.Design:** the designing part is the structure of the coding where it shows which coding comes from one another to another. So, I have arranged all the def functions in order where the scenario is asked and give the perfect outcome of code.

**4.Implementation:** This has been done after the design where the all the structure is analyzed and make sure the process is to run for example I`ll give two results where it goes with order.

def request details(self):

items = []

total = 0

while True:

item\_name = input("Enter item name (or type 'done' to finish): ")

if item\_name.lower() == 'done':

break

cost = float(input(f"Enter cost for{item\_name}: "))

items.append((item\_name, cost))

total += cost

print(f"\nTotal amount for request is: ${total:.2f}")

return total, items

def request\_approval(self, total\_amount):

status = "approved" if total\_amount < 500 else "pending"

requestID = RequestSystem.request\_counter

RequestSystem.request\_counter += 1

request = {

'id' : requestID

'user' : self.staff\_info,

'total' : total\_amount,

'status' : status,

}

self.requst.append(request)

print(f"Request Submitted with ID {request\_id} and status '{status}'.\n")

**5.Testing & integration**: In this part after all the design and structure is complete the code has been run to have the complete outcome. So, if coding is correct the result shown in the scenario should be correct. My scenario was the

**1. Total number of requests submitted**

**2. Total number of approved requests**

**3. Total number of pending requests**

**4.Total number of not approved requests.**

**6.Maintanace:** this is the last part of the SDLC process the scenario and outcomes should be same at every situation and go in flow.