# Chapter 1

**INTRODUCTION**

## Objective:

Software Quality Assurance (SQA) is a set of questions for ensuring quality in software engineering processes. It ensures that developed software meets and complies with the defined or standardized quality specifications. Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards suitable for the project and implemented correctly.

## Aim and Scope of the Project:

The purpose of this Software Quality Assurance (SQA) is to define the techniques,

procedures, and methodologies that will be used upon the software model to assure timely delivery of the software that meets specified requirements within project resources.

The use of this plan will help assure the following:

(1) That software development, evaluation and acceptance standards are developed, documented and followed.

(2) That the results of software quality reviews and audits will be given to appropriate management. This provides feedback as to how well the development effort is conforming to various software development standards.

(3) That test results adhere to acceptance standards

## 

# Chapter 2

**SYSTEM ANALYSIS**

## Introduction:

Analysis can be defined as breaking up of any whole so as to find out their nature, function etc. It defines design as to make preliminary sketches of ; to sketch a pattern or outline for plan. To plan and carry out especially by artistic arrangement or in a skillful way. System analysis and design can be characterized as a set of techniques and processes, a community of interests, a culture and an intellectual orientation.

* + - The various tasks in the system analysis include the following:
      * Understanding application
      * Planning
      * Scheduling
      * Developing candidate solution
      * Performing trade studies
      * Performing cost benefit analysis

This system manages to analyze the report creation and it develops a manual entry of the citizen record.

## Existing System:

In the existing system, a large number of codes are divided into only two modules. So is the existing system, performance analysis takes more time and is also not accurate.

As per Mancoridis et al., the earliest of software metrics deal with the measurement of code complexity and its maintainability. He measured the Modularization Quality (MQ) which is the combination of coupling and cohesion.

Cohesion is measured as the ratio of the number of internal function-call dependencies that actually exist, to the maximum possible internal dependencies. Coupling is measured as the ratio of the number of actual external function-call dependencies between the two subsystems, to the maximum possible number of such external dependencies. The system level MQ is calculated as the difference between the average cohesion and the average coupling.

In the pre-industrial era, markets were monopolized mainly by the Guilds, which were a union of craftsmen formed during the 13th century. The Guilds were responsible for maintaining or verifying the quality of goods and services provided by the members. The guild masters inspected the goods to make sure that the necessary standards were maintained.

Quality was person dependent during this era. Individual skill was the driving factor for quality. People took pride in their products and worked hard to maintain quality. This lasted until the late 19th century when industrialization gained a rapid foothold. Mass production broke the manufacture of goods into simple steps.

The division of work led to specialization and quality started becoming more process-oriented than people oriented. There was a reduction in labor costs as the unskilled workers could be trained to perform specific tasks. But, people lacked motivation and got weary with the monotonous and repetitive work. Moreover, as the jobs were split into different levels such as lower level tasks, the workers found it difficult to relate with and be proud of the final product.

They were general discontent among the laborers with the work they were doing and as a result, the quality of products started going down significantly. Mass production also led to manipulation and exploitation of unskilled laborers. Moreover, the process made workers dispensable and increased the power vested on the foreman and managers, which often led to misuse and exploitation.

## Proposed System:

To overcome the drawbacks of the existing system, the proposed system has been evolved. This project aims to reduce the paperwork and saves time.

The system provides with the best user interface. Here the details are maintained in computers so it becomes easier for the user to document and review the quality assurance plan such as tests and milestones in testing.

Advantages of proposed system:

* SQA produce high quality software.
* High quality application saves time and cost.
* SQA is beneficial for better reliability.
* SQA is beneficial in the condition of no maintenance for long time.
* High quality commercial software increase market share of company.
* Improving the process of creating software.
* Improves the quality of the software.

# Chapter 3

**SYSTEM REQUIREMENT SPECIFICATION**

This document details the requirements of Management System for Distributed Environment. The software requirements shall be specified for all the phases of Management Systems.

## Description of software:

A system requirement specification is a complete description of the behavior of the software to be developed and intended purpose to develop that software. To reach specified requirements the developers need less time and effort and the development cost is minimized.

A good SRS defines how the application will interact with the system hardware and users in real world situations.

## System Requirements:

The following are the minimum Hardware Requirements: Processor : Intel Core i5

RAM : Minimum of 2GB

Hard Disk : Minimum of 50GB

## 3.2.1 Software Requirements:

Software Requirements deal with defining software resource requirements and pre- requisites that need to be installed on a computer to provide optimal functioning of an application.

These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

Operating System : Windows 8/10(64 bit) Development Environment : Visual Studio Code

Developing Language : Python (tkinter framework)

# Chapter 4

**PROJECT DESCRIPTION**

## 4.1 Problem definition:

Quality assurance is a set of activities intended to establish confidence that quality requirements will be met. It is one part of quality management. The quality assurance function should be a cross functional activity in which all functional departments play a specific role in planning for and analyzing quality. Quality control is considered part of the quality functions. It is a regulatory process through which each department measures the actual quality performance, compares it with the standard, and acts on the difference.

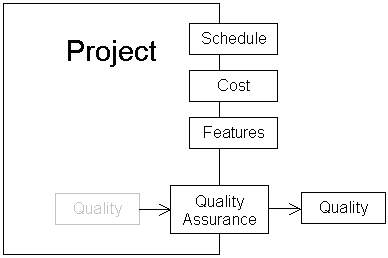


Fig. 4.1

**SYSTEM DESIGN:**

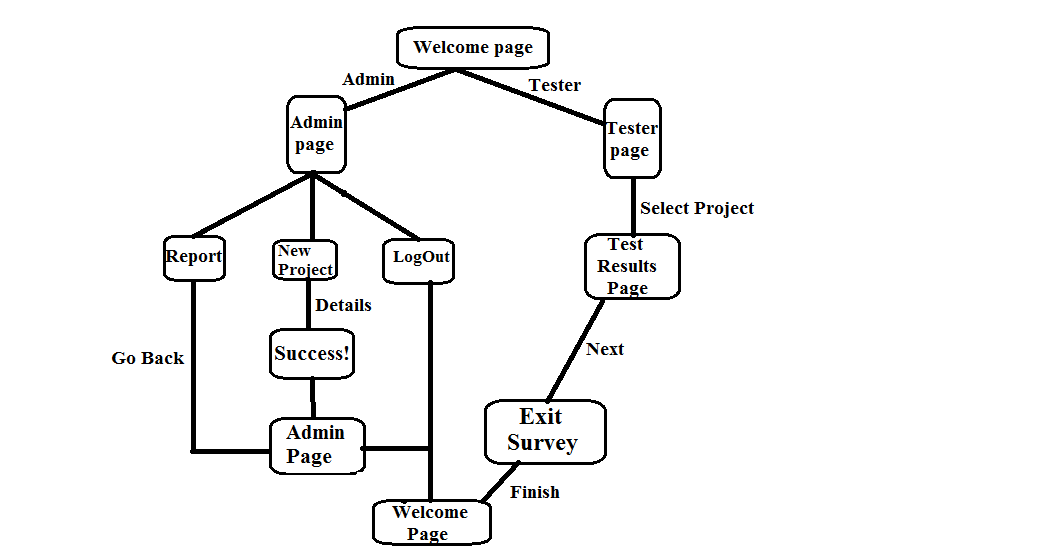
****

Fig. 4.2

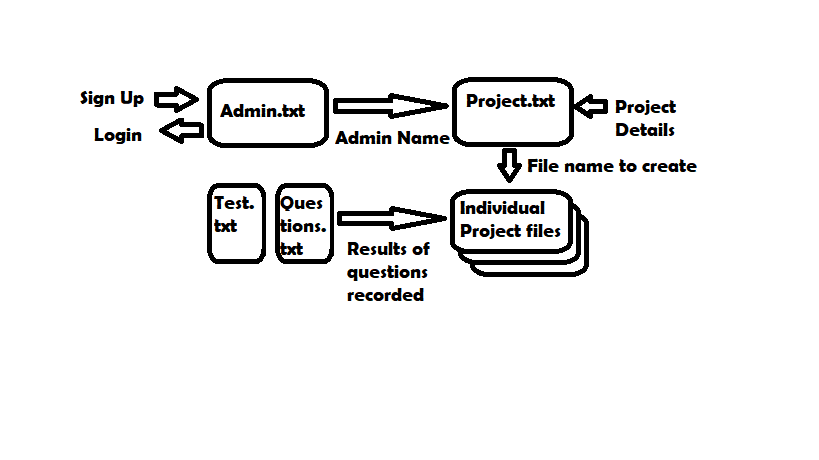


Fig. 4.3

**CHAPTER 5**

**SYSTEM IMPLEMENTATION**

System implementation is the important stage of the project when the theoretical design is tuned into practical system. The main stages in the implementation are as follows:

* + - Planning
    - Training
    - System testing and
    - Change over planning

A planning is the first task in the system implementation. At the time of implementation of any system, people from different departments and system analysts involved. They are confirmed to practical problem of controlling various activities of people outside their own data processing departments.

## Selection of language:

Implementation phase should perfectly map the designed document in a suitable programming language in order to achieve the necessary final and correct product. The Python programming language is used with tkinter framework for implementation purpose.

Although, each programming language has its own strength and weaknesses , the reason for choosing a particular language may ultimately be based on factors having nothing to do with the technical merits of the language itself. Such factors may include the availability of compilers for the host/target, the maturity and efficiency of available compilers.

The availability of programmers who already knew the language, whether the language can easily interface with existing systems such as graphical user interfaces, the existence of legacy software written in a specific language, or how well the language fits in with adopted COTS products.

## Python:

Python is a multiparadigm, general-purpose, interpreted, high-level programming language. Python allows programmers to use different programming styles to create simple or complex programs, get quicker results and write code almost as if speaking in a human language. Some of the popular systems and applications that have employed Python during development include Google Search, YouTube, Bit Torrent, Google App Engine, Eve Online, Maya and iRobot machines.

There are two attributes that make development time in Python faster than in other programming languages:

Python is an interpreted language, which precludes the need to compile code before executing a program because Python does the compilation in the background. Because Python is a high-level programming language, it abstracts many sophisticated details from the programming code. Python focuses so much on this abstraction that its code can be understood by most novice programmers.

Python code tends to be shorter than comparable codes. Although Python offers fast development times, it lags slightly in terms of execution time. Compared to fully compiling languages like C and C++, Python programs execute slower. Of course, with the processing speeds of computers these days, the speed differences are usually only observed in benchmarking tests, not in real-world operations. In most cases, Python is already included in Linux distributions and Mac OS X machines.

**tkinter framework:**

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

**To create a tkinter:**

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

## 

## System maintenance:

The maintenance phase of the software cycle is the time in which software product performs useful work. After the system is successfully implemented it should be maintained in a proper manner.

The purpose of maintenance process is to sustain the capability of a system to provide a service. This process monitors the system's capability to deliver services, records problems for analysis, Takes corrective, adaptive, perfective and preventive actions and confirms restored capability.

As a result of the successful implementation of the maintenance process:

* A maintenance strategy is developed.
* Maintenance constraints are provided as inputs to requirements.
* Replacement system elements are made available.
* Services meeting stake holder requirements are sustained.
* The need for corrective design changes is reported.
* Failure and life time data is recorded.

The organization responsible for maintaining the system should have clear thresholds established to determine whether a change requested by end users, changes to correct latent defects, or changes required to fulfill the evolving mission are within the scope of a maintenance change or require a more formal project to step through the entire systems engineering life cycle. Evaluation criteria to make such a decision could include cost, schedule, risk or critical characteristics.

# Chapter 6

**SYSTEM TESTING**

**System testing** is testing conducted on a complete integrated system to evaluate the system's compliance with its specified [requirements](https://en.wikipedia.org/wiki/Requirements). System testing takes, as its input, all of the integrated components that have passed [integration testing](https://en.wikipedia.org/wiki/Integration_testing). The purpose of integration testing is to detect any inconsistencies between the units that are integrated together.

System testing seeks to detect defects both within the "inter-assemblages" and also within the system as a whole. System testing is performed on the entire system in the context of a [Functional Requirement](https://en.wikipedia.org/wiki/Functional_requirements) Specification (FRS) and/or a [System Requirement](https://en.wikipedia.org/wiki/Requirements_analysis) Specification (SRS).

System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification.

System testing is the stage before system implementation where the system is made error free and all needed modifications are made. The system was tested with test data and necessary corrections to the system were carried out. All the reports were checked by the user and approved. The system was user friendly with online help to assist user where necessary.

## Test Plan:

A document describing the scope, approach, resources and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process.

Making Test Plan has multiple benefits:

* + - * Test Plan helps us determine the **effort** needed to validate the quality of the application under test.
      * Help people outside the test team such as developers, business managers, customers **understand** the details of testing.
      * Test Plan **guides** our thinking. It is like a rule book, which needs to be followed.
      * Important aspects like test estimation, test scope[, Test Strategy](https://www.guru99.com/how-to-create-test-strategy-document.html) are **documented** in Test Plan, so it can be reviewed by Management Team and re-used for other projects.

#### Major testing activities are:

* + - * Test units.
      * Features to be tested.
      * Approach for testing.
      * Test deliverables.
      * Schedule
      * Personal Allocation

#### Test Units:

A test case is major activity in the testing process. In this project, we have performed two levels of testing.

* + - * Unit Testing.
      * Integration Testing

## Unit Testing:

Unit testing is undertaken when a module has been created and successfully reviewed. In order to test a single module we need to provide with complete environment i.e. beside the module we would require.

* + - The procedure belonging to other modules that the module under test calls.
    - Non local data structures that the module accesses.
    - A procedure to call the functions of the module under test with appropriate parameters
    - Unit testing was done on each and every module that is described under module description

## Integration Testing:

In this type of testing, we test the various integration of the project module by providing the input. The primary objective is to test the module interfaces in order to ensure that no errors are occurring when one module invokes the other modules.

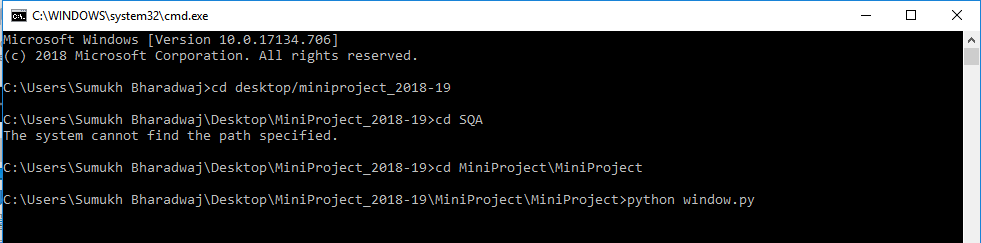
Some different types of integration testing are big-bang, mixed (sandwich), risky- hardest, [top-down, and bottom-up](https://en.wikipedia.org/wiki/Top-down_and_bottom-up_design). Other Integration Patterns are: collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration.

* + - In the **big-bang approach**, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. This method is very effective for saving time in the integration testing process.
    - **Bottom-up testing** is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing.
    - **Top-down testing** is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module.
    - **Sandwich testing** is an approach to combine top down testing with bottom up testing

# Chapter 7

**WORKING OF APPLICATION**

# Open CMD and after moving to the project directory, run window,py



# Fig. 7.1

1. Follow the instructions on the screen to proceed

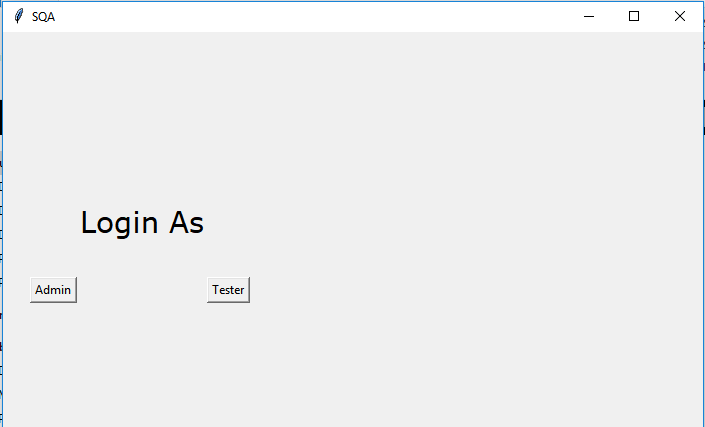


Fig. 7.2

1. Admin Mode:

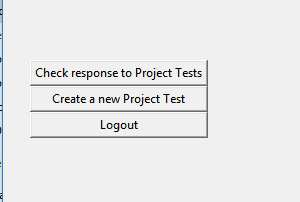


Fig. 7.3

* 1. Project creation: To create a new project, just click on the button and enter name and ID

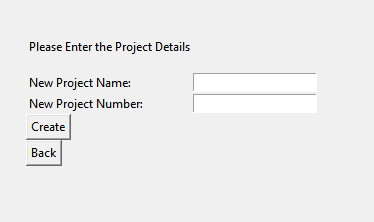


Fig. 7.4

* 1. View Report: To View reports, click on Check Responses, select the project and click view results:

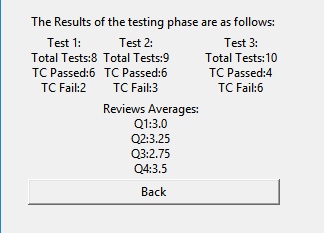


Fig. 7.5

1. Tester Mode:
   1. Click on Tester in the login page:
   2. Then select the desired project and click go

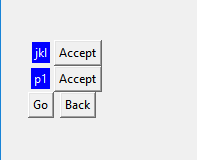


Fig. 7.6

4.3 Enter in values as specified and click next

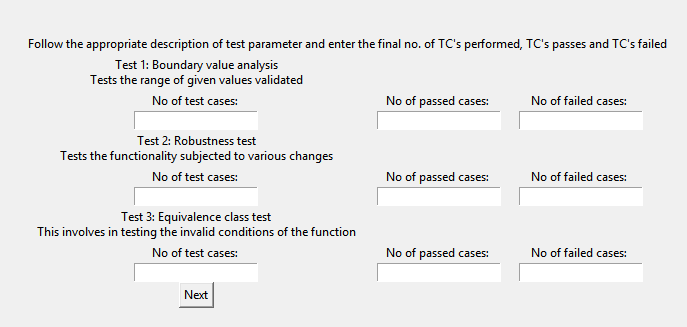


Fig. 7.7

4.4 Lastly enter the Exit survey of the application tested and click finish

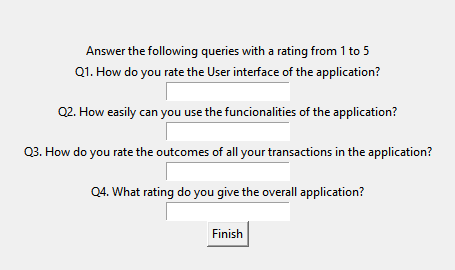


Fig. 7.8

Note: After one transaction, you should close the application to save the changes onto the file.

**OTHER SNAPSHOTS**

1. Files used:

1.1: Admin.txt (Read and Write): The code fetches the credentials of the administrator from here

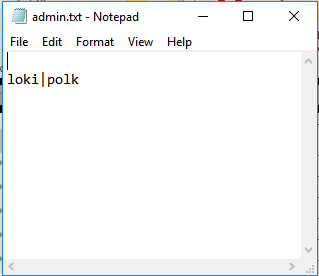


Fig. 7.9

1.2: Project.txt (Read and Write): This contains the details of the created projects by the admin

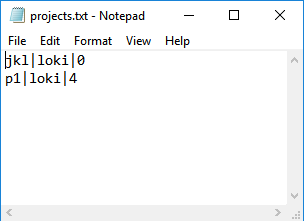


Fig.7.10

1.3: tests.txt (Read only): The type of tests and the description is fetched from here:

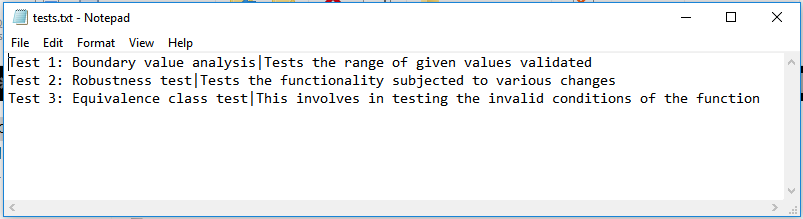


Fig. 7.11

1.4: p1.txt (Write Only): The filename will be the individual projects created by the admin; this contains the test answers given by the testers.

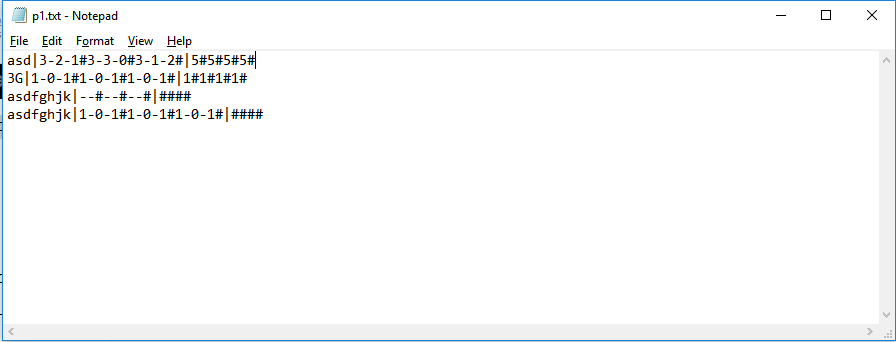


Fig. 7.12

# Chapter 7

**CONCLUSION**

With the continuous development and progress of the times, the software quality standard has made great contributions to the development and improvement of software engineering, and has created favorable conditions for software quality assurance. The high quality assurance of software needs our scientific researchers to refine and improve the standard of software quality standard continuously. Not only do we need to mine software testing methods, create automated software testing tools, and constantly improve the testing process management mechanism, but also the development of software engineering must conform to the software quality standards that are constantly developing and updating in today's society. Only in such a perfect software engineering mechanism can high quality software be developed.

# FUTURE DEVELOPMENT

The project has a very vast scope in future. The project can be implemented on internet in future. Project can be updated in near future as & when requirement for the same arises, as its very flexible in terms of expansion. There is always room for improvements. Hence, the project can be improved further to accommodate the user requirements.

The following are the future scope for this project:

* Building the classes and modules so that the functions code should be eliminated.
* Multiple users can login at a time.
* Auto generation of forms for every user when they create their account.
* Can be used over a network.
* Security given is different at user level and admin to give all rights to use the whole software & data entry operator has given a limit for working.
* Admin should be able to create custom test questions and exit surveys for each project.

**Chapter 8**

**APPENDIX**

**Source Code:**

import tkinter as tk

import os

from tkinter import messagebox

import pdb

creds = 'admin.txt'

proj = 'projects.txt'

global key

global key2

global key3

LARGE\_FONT = ("Verdana",22)

def Details(p):

with open(key3, 'a') as f:

f.write(p)

f.write('|')

f.close()

return

def Assign(pop):

global key3

key3 = pop+'.txt'

return

def CheckLogin():

global key

with open(creds) as f:

for line in f:

if nameEL.get() in line:

matchedLine = line

l = matchedLine.split('|')

uname=l[0]

pword=l[1].rstrip()

if nameEL.get() == uname and pwordEL.get() == pword:

key=uname

logged()

elif pwordEL.get() != pword:

messagebox.showinfo("Oops!", "Invalid Name or Password")

def logged():

done.destroy()

log1 = SQA2()

def FSSignup():

with open(creds, 'a') as f:

f.write(nameE.get())

f.write('|')

f.write(pwordE.get())

f.write('\n')

f.close()

messagebox.showinfo("Success", "Profile Created")

return

def Creation():

with open(proj, 'a') as f:

f.write(nameP.get())

f.write('|')

f.write(key)

f.write('|')

f.write(numbP.get())

f.write('\n')

f.close()

w=nameP.get()+'.txt'

with open(w,'w+') as a:

a.close

messagebox.showinfo("Success", "Project Test Created")

class SQA2(tk.Tk):

def \_\_init\_\_(self, \*args, \*\*kwargs):

tk.Tk.\_\_init\_\_(self, \*args, \*\*kwargs)

tk.Tk.wm\_title(self, "SQA")

tk.Tk.wm\_geometry(self, "700x550")

box = tk.Frame(self)

box.pack(expand=True)

box.grid\_rowconfigure(0, weight=1)

box.grid\_columnconfigure(0, weight=1)

self.frames = {}

for F in (StartPage, Testlog, Adlog, Testchos, Adchos, Quest, Adops, Results, Create, Testing):

frame = F(box, self)

self.frames[F]=frame

frame.grid(row=0,column=0,sticky="nsew")

self.show\_frame(Adops)

def show\_frame(self, boxes):

frame = self.frames[boxes]

frame.tkraise()

class SQA(tk.Tk):

def \_\_init\_\_(self, \*args, \*\*kwargs):

tk.Tk.\_\_init\_\_(self, \*args, \*\*kwargs)

tk.Tk.wm\_title(self, "SQA")

tk.Tk.wm\_geometry(self, "700x550")

box = tk.Frame(self)

box.pack(expand=True)

box.grid\_rowconfigure(0, weight=1)

box.grid\_columnconfigure(0, weight=1)

self.frames = {}

for F in (StartPage, Adlog, Testchos, Adchos, Quest, Adops,Testing, Results, Create, Testlog):

frame = F(box, self)

self.frames[F]=frame

frame.grid(row=0,column=0,sticky="nsew")

self.show\_frame(StartPage)

def show\_frame(self, boxes):

frame = self.frames[boxes]

frame.tkraise()

class StartPage(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self,parent)

label = tk.Label(self, text="Login As\n", font=LARGE\_FONT)

label.grid(row=5, column=1, sticky="w")

button1 = tk.Button(self, text="Admin", command=lambda:controller.show\_frame(Adlog))

button1.grid(row=10, column=0, sticky="w")

button2 = tk.Button(self, text="Tester", command=lambda:controller.show\_frame(Testchos))

button2.grid(row=10, column=2, sticky="w")

class Adlog(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self,parent)

global pwordE

global nameE

intruction = tk.Label(self, text='Please Enter new Credidentials\n')

intruction.grid(row=0, column=0, sticky="E")

nameL = tk.Label(self, text='New Username: ')

pwordL = tk.Label(self, text='New Password: ')

nameL.grid(row=1, column=0, sticky="W")

pwordL.grid(row=2, column=0, sticky="W")

nameE = tk.Entry(self)

pwordE = tk.Entry(self, show='\*')

nameE.grid(row=1, column=1)

pwordE.grid(row=2, column=1)

signupButton = tk.Button(self, text='Signup', command=FSSignup)

signupButton.grid(columnspan=2, sticky="W")

global nameEL

global pwordEL

intruction = tk.Label(self, text='Please Login\n')

intruction.grid(sticky="E")

nameL = tk.Label(self, text='Username: ')

pwordL = tk.Label(self, text='Password: ')

nameL.grid(row=5, column=0, sticky="W")

pwordL.grid(row=6, column=0, sticky="W")

nameEL = tk.Entry(self)

pwordEL = tk.Entry(self, show='\*')

nameEL.grid(row=5, column=1)

pwordEL.grid(row=6, column=1)

loginB = tk.Button(self, text='Login', command=CheckLogin)

loginB.grid(columnspan=2, sticky="W")

class Create(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self,parent)

global nameP

global numbP

intruction = tk.Label(self, text='Please Enter the Project Details\n')

intruction.grid(row=0, column=0, sticky="E")

nameL = tk.Label(self, text='New Project Name: ')

projn = tk.Label(self, text='New Project Number: ')

nameL.grid(row=1, column=0, sticky="W")

projn.grid(row=2, column=0, sticky="W")

nameP = tk.Entry(self)

numbP = tk.Entry(self)

nameP.grid(row=1, column=1)

numbP.grid(row=2, column=1)

signupButton = tk.Button(self, text='Create', command=Creation)

signupButton.grid(columnspan=2, sticky="W")

backButton = tk.Button(self, text='Back', command=lambda:controller.show\_frame(Adops))

backButton.grid(columnspan=2, sticky="W")

class Testchos(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self,parent)

i=0

with open(proj) as f:

for line in f:

txt=line.split('|')

i+=1

tk.Label(self,text=txt[0],bg='blue',fg='white').grid(row=i, column=1)

tk.Button(self,text="Accept",command=Assign(txt[0])).grid(row=i, column =3)

submit=tk.Button(self, text='Go', command=lambda:controller.show\_frame(Testlog))

submit.grid(row=15,column=1)

backButton = tk.Button(self, text='Back', command=lambda:controller.show\_frame(StartPage))

backButton.grid(row=15,column=3)

class Adchos(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self,parent)

i=0

with open(proj) as f:

for line in f:

txt=line.split('|')

i+=1

tk.Label(self,text=txt[0],bg='blue',fg='white').grid(row=i, column=1)

tk.Button(self,text="Accept",command=Assign(txt[0])).grid(row=i, column =3)

submit=tk.Button(self, text='View Results', command=lambda:controller.show\_frame(Results))

submit.grid(row=15,column=1)

backButton = tk.Button(self, text='Back', command=lambda:controller.show\_frame(Adops))

backButton.grid(row=15,column=3)

class Testlog(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self,parent)

global nameT

intruction = tk.Label(self, text='Please Enter your Details\n')

intruction.grid(row=0, column=0, sticky="E")

nameL = tk.Label(self, text='Name: ')

nameL.grid(row=1, column=0, sticky="W")

nameT = tk.Entry(self)

nameT.grid(row=1, column=1)

signupButton1 = tk.Button(self, text='Test', command=lambda:[Details(nameT.get()),controller.show\_frame(Testing)])

signupButton1.grid(columnspan=2, sticky="W")

class Quest(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self,parent)

tk.Label(self, text="Answer the following queries with a rating from 1 to 5").grid(row=0,columnspan=2)

i=1

self.hen=[]

global h

with open('questionnaire.txt','r') as f:

for line in f:

tk.Label(self,text=line.rstrip()).grid(row=(i\*2))

h=tk.Entry(self)

h.grid(row=(i\*2)+1, column=0)

self.hen.append(h)

i+=1

f.close

tk.Button(self,text='Finish',command=lambda:[self.writer(i-1),self.seperator(),controller.show\_frame(StartPage)]).grid()

def writer(self,l):

i=0

with open(key3,'a') as f:

for i in range(l):

f.write(self.hen[i].get())

f.write('#')

f.close()

def seperator(self):

with open(key3,'a') as f:

f.write('\n')

f.close()

class Testing(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self,parent)

tk.Label(self, text="Follow the appropriate description of test parameter and enter the final no. of TC's performed, TC's passes and TC's failed").grid(row=0,columnspan=4)

i=1

self.nen=[]

self.pen=[]

self.den=[]

global n

global p

global d

with open('tests.txt','r') as f:

for line in f:

txt=line.rstrip().split('|')

tk.Label(self,text=txt[0]+'\n'+txt[1]).grid(row=(i\*2))

n=tk.Entry(self)

n.grid(row=(i\*2)+1,column=0)

self.nen.append(n)

p=tk.Entry(self)

p.grid(row=(i\*2)+1,column=1)

self.pen.append(p)

d=tk.Entry(self)

d.grid(row=(i\*2)+1,column=2)

self.den.append(d)

i+=1

f.close()

tk.Button(self,text='Next',command=lambda:[self.writer(i-1),self.seperator(),controller.show\_frame(Quest)]).grid()

def writer(self,l):

with open(key3,'a') as f:

i=0

for i in range(l):

f.write(self.nen[i].get())

f.write('-')

f.write(self.pen[i].get())

f.write('-')

f.write(self.den[i].get())

f.write('#')

f.close()

def seperator(self):

with open(key3,'a') as f:

f.write('|')

f.close()

class Adops(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self,parent)

button1 = tk.Button(self, text='Check response to Project Tests', command=lambda:controller.show\_frame(Adchos))

button2 = tk.Button(self, text='Create a new Project Test', command=lambda:controller.show\_frame(Create))

button3 = tk.Button(self, text='Logout', command=lambda:controller.show\_frame(StartPage))

button1.grid(columnspan=5, sticky="nsew")

button2.grid(columnspan=5, sticky="nsew")

button3.grid(columnspan=5, sticky="nsew")

class Results(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self,parent)

tk.Label(self,text='The Results of the testing phase are as follows:').grid(row=0,columnspan=3,sticky='w')

with open(key3,'r') as f:

a=b=c=a1=b1=c1=d1=d2=d3=d4=i=0

for line in f:

i+=1

txt=line.split('|')

tes=txt[1].rstrip().split('#')

a+=int(tes[0].split('-')[0])

b+=int(tes[1].split('-')[0])

c+=int(tes[2].split('-')[0])

a1+=int(tes[0].split('-')[1])

b1+=int(tes[1].split('-')[1])

c1+=int(tes[2].split('-')[1])

que=txt[2].rstrip().split('#')

d1+=int(que[0])

d2+=int(que[1])

d3+=int(que[2])

d4+=int(que[3])

if(i!=0):

tk.Label(self, text='Test 1:\nTotal Tests:'+str(a)+'\nTC Passed:'+str(a1)+'\nTC Fail:'+str(a-a1)).grid(row=1,column=0,sticky='w')

tk.Label(self, text='Test 2:\nTotal Tests:'+str(b)+'\nTC Passed:'+str(b1)+'\nTC Fail:'+str(b-b1)).grid(row=1,column=1,sticky='w')

tk.Label(self, text='Test 3:\nTotal Tests:'+str(c)+'\nTC Passed:'+str(c1)+'\nTC Fail:'+str(c-c1)).grid(row=1,column=2,sticky='w')

tk.Label(self, text='Reviews Averages:\nQ1:'+str(d1/i)+'\nQ2:'+str(d2/i)+'\nQ3:'+str(d3/i)+'\nQ4:'+str(d4/i)).grid(row=6,column=1,sticky='w')

f.close()

button3 = tk.Button(self, text='Back', command=lambda:controller.show\_frame(Adops))

button3.grid(columnspan=5, sticky="nsew")

done = SQA()

done.mainloop()

from tkinter import \*

root=Tk()

entries = []

for i in range(10):

en = Entry(root)

en.grid(row=i+1, column=0)

entries.append(en)

def hallo():

for entry in entries:

print(entry.get())

button=Button(root,text="krijg",command=hallo).grid(row=12,column=0)

root.mainloop()

# REFERENCES

* **Tkinter GUI tutorials by sentdex**,.

<https://www.youtube.com/watch?v=HjNHATw6XgY&list=PLQVvvaa0QuDclKx-QpC9wntnURXVJqLyk>

* **Mr. Kumar Kaushik S**, BE

Machine Learning Engineer, Ascent Robotics

* **Tutorials Point**,

Python-tkinter GUI