UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
SISTEMAS DE INFORMAÇÃO

ROTEIROS E ELABORAÇÕES DE TESTES NA FUNÇÃO DE LOGIN

BRUNO FRANÇA DO PRADO

MANAUS, 14 DE ABRIL DE 2022

BRUNO FRANÇA DO PRADO

ROTEIROS E ELABORAÇÕES DE TESTES NA FUNÇÃO DE LOGIN

Trabalho apresentado pelo professor Jonathas Silva dos Santos para análise, estudo e pesquisa de casos de testes, a fim de ampliar o conhecimento na matéria de Tópicos Especiais em Engenharia de Software.

1. FUNÇÃO SELECIONADA DE UMA IMPLEMENTAÇÃO PRÓPRIA

Para a elaboração desse trabalho, busquei um programa/função de login de uma aplicação antiga de busca de empregos, que solicitava os dados pessoais de um CPF ou CNPJ para cadastro e a seguinte função para login.

O programa em si precisa aceitar o e-mail e a senha do usuário, fazer o hash da senha, armazenar em um arquivo de texto junto com o e-mail e concluir o processo de registro. No processo de login, o programa novamente aceitará o e-mail e senha do usuário, fará o hash da senha, verificará com o e-mail e o hash da senha armazenados no arquivo de texto anteriormente, imprimirá uma mensagem de sucesso confirmando o e-mail e a senha digitada está correta ou imprima uma mensagem de falha se o e-mail e a senha estiverem incorretos.

```
import hashlib

def signup():
    email = input("Enter email address: ")
    pwd = input("Enter password: ")
    conf_pwd = input("Confirm password: ")
    if conf_pwd == pwd:
        enc = conf_pwd.encode()
        hash1 = hashlib.md5(enc).hexdigest()
        with open("credentials.txt", "w") as f:
            f.write(email + "\n")
            f.write(hash1)
        f.close()
        print("You have registered successfully!")
    else:
        print("Password is not same as above! \n")

def login():
    email = input("Enter email: ")
    pwd = input("Enter password: ")
    auth = pwd.encode()
    auth_hash = hashlib.md5(auth).hexdigest()
    with open("credentials.txt", "r") as f:
        stored_email, stored_pwd = f.read().split("\n")
    f.close()
    if email == stored_email and auth_hash == stored_pwd:
         print("Logged in Successfully!")
     else:
         print("Login failed! \n")
while 1:
    print("********** Login System **********")
    print("1.Signup")
    print("2.Login")
    print("3.Exit")
    ch = int(input("Enter your choice: "))
    if ch == 1:
        signup()
    elif ch == 2:
        login()
    elif ch == 3:
        break
    else:
        print("Wrong Choice!")
```

2. ROTEIRO DE CASOS DE TESTE USANDO TÉCNICA FUNCIONAL

Em primeiro plano, vou utilizar da técnica funcional para elaborar meu roteiro de testes, usando os critérios de partição de equivalência e análise do valor limite.

2.1. ROTEIRO DE TESTES

Roteiro 01	Testar funcionalidade de LOGIN em uma aplicação

Teste	**Dados de Entrada**	**Saída Esperada**
Informar usuário válido e senha inválida	email@subdomain.example.com 123	Inválido!
Informar usuário inválido e senha válida	_____@example.com Zz]P^22dc	Inválido!
Informar usuário válido e senha válida	email@subdomain.example.com Zz]P^22dc	Válido!
Informar usuário inválido e senha inválida	email@123.123.123.123 123	Inválido!
Usuário e senha válidos porém não correspondentes	email@subdomain.example.com ZzZzZ[][22	Inválido!
Deixar usuário e senha em branco	- -	Inválido!
Inserir apenas caracteres numéricos no usuário	12345 Zz]P^22dc	Inválido!
Inserir apenas caracteres numéricos na senha	email@subdomain.example.com 472937592	Inválido!

2.2. PARTIÇÃO DE EQUIVALÊNCIA

O que é válido:

- Um endereço de email válido, com caracteres alfanuméricos;
- Uma senha com caracteres alfanuméricos de no mínimo 8 caracteres.

O que não é válido:

- Um email com apenas números ou apenas letras, sem a presença do domínio com @;
- Uma senha com apenas números;
- Uma senhas com apenas letras;
- Uma senha sem letras maiúsculas;
- Senha senha sem letras minúsculas;
- Senha com ao menos um caracter especial.

2.3. ANÁLISE DO VALOR LIMITE

Na entrada (320 caracteres):

- O limite é um máximo de 64 caracteres na "parte local" (antes do "@") e um máximo de 255 caracteres na parte do domínio (após o "@") para um comprimento total de 320 caracteres.

Na saída (128 caracteres):

- O comprimento máximo é de 128 caracteres. As senhas com menos de 20 caracteres geralmente são consideradas fracas, se consistirem apenas em caracteres minúsculos de origem latina.

3. ROTEIRO DE CASOS DE TESTE (TÉCNICA ESTRUTURAL)

Roteiro 02	Testar funcionalidade de LOGIN em uma aplicação

Teste	Dados de Entrada	Saída Esperada
Informar usuário válido e senha inválida	email@subdomain.example.com 123	Inválido!
Informar usuário válido e senha válida	email@subdomain.example.com Zz]P^22dc	Válido!
Informar usuário inválido e senha inválida	email@123.123.123.123 123	Inválido!
Inserir apenas caracteres numéricos no usuário	12345 Zz]P^22dc	Inválido!

2.1. GRAFO DE FLUXO DE CONTROLE

2.2. NÍVEL DE COBERTURA

O nível de cobertura 7 foi atingido nesse algoritmo; ao realizar os testes, podemos ver que além da função de criptografar e gerar um txt para guardar os dados de usuário, o algoritmo também percorre os diferentes comandos, decisões, condições, loops e caminhos para determinar se sua senha/email foram inseridos incorretamente ou corretamente, assim como vários pormenores de definir tipos de senha e email válido no momento do primeiro input.