PROJETO PRÁTICO – TESTES E QUALIDADE DE SOFTWARE
Sistema sob Teste (SUT): TaskManager UNISAGRADO – Gerenciador de Tarefas

1. SISTEMA A SER TESTADO (SUT)

Nome: TaskManager UNISAGRADO
Tipo: Aplicação web simples de gerenciamento de tarefas para alunos.

1.1. Descrição geral do sistema

O TaskManager UNISAGRADO permite que um usuário autenticado gerencie suas tarefas acadêmicas.
Funcionalidades principais:
- Autenticação (login/logout).
- CRUD de tarefas: criar, listar, editar, excluir.
- Marcar tarefas como concluídas.
- Filtrar tarefas por status (Pendentes/Concluídas).
- Limites:
  - Título da tarefa com tamanho máximo (100 caracteres).
  - Número máximo de tarefas ativas por usuário (100).
- Mensagens claras de erro e sucesso.

1.2. Tipos de teste possíveis no SUT

- Testes funcionais: login, CRUD de tarefas, filtros, limites.
- Testes de integração/sistema/aceitação: fluxo completo (login → criar tarefa → concluir → excluir → log
- Teste não funcional: usabilidade (clareza das mensagens, facilidade de uso) e/ou desempenho simples

2. FASE 1 – DESCOBERTA E REQUISITOS TESTÁVEIS

2.1. Requisitos Funcionais (RF)

RF01 – Login de usuário
O sistema deve permitir login com e-mail e senha cadastrados, validando campos obrigatórios e credenc

RF02 – Logout
O sistema deve permitir que o usuário encerre a sessão e impeça acesso às telas internas sem novo log

RF03 – Cadastro de tarefa
O usuário pode criar tarefas com:
- título (obrigatório, até 100 caracteres),
- descrição (opcional),
- data de vencimento (opcional),
- status (inicialmente “Pendente”).

RF04 – Listagem de tarefas
Exibir lista de tarefas do usuário logado com título, status e data de vencimento (se houver).

RF05 – Edição de tarefa
Permitir editar os dados da tarefa, respeitando as mesmas validações do cadastro.

RF06 – Exclusão de tarefa
Permitir excluir tarefa com confirmação prévia.

RF07 – Marcar tarefa como concluída

Permitir atualizar status de “Pendente” para “Concluída” com destaque visual.

RF08 – Filtro de tarefas por status
Permitir filtrar entre Todas, Pendentes e Concluídas.

RF09 – Limite de título e quantidade de tarefas
- Título com no máximo 100 caracteres.
- Limite de 100 tarefas ativas por usuário.

RF10 – Mensagens de feedback
Exibir mensagens claras de sucesso e erro em todas as operações.

2.2. Requisitos Não Funcionais (RNF)

RNF01 – Usabilidade
As mensagens de erro/sucesso devem ser objetivas, e os botões devem ter rótulos claros (por exemplo,

RNF02 – Desempenho simples
A listagem de tarefas deve ser carregada em até 2 segundos com até 200 tarefas para um usuário.

2.3. Cenários em formato Dado / Quando / Então (exemplos)

Cenário 1 – Login com credenciais válidas
Dado que o usuário está na tela de login
E preenche e-mail e senha válidos
Quando clicar em “Entrar”
Então o sistema deve autenticar o usuário
E redirecionar para a lista de tarefas
E exibir mensagem “Login realizado com sucesso”.

Cenário 2 – Criar tarefa sem título
Dado que o usuário está autenticado
E está na tela “Nova tarefa”
Quando clicar em “Salvar” deixando o título vazio
Então o sistema não deve criar a tarefa
E deve exibir “Título é obrigatório”.

Cenário 3 – Limite máximo de tarefas
Dado que o usuário possui 100 tarefas ativas
Quando tentar criar mais uma tarefa
Então o sistema não deve criar a tarefa
E deve exibir “Limite máximo de tarefas ativas atingido”.

Cenário 4 – Marcar tarefa como concluída
Dado que o usuário está autenticado
E possui uma tarefa pendente
Quando clicar em “Concluir”
Então o status deve ser alterado para “Concluída”
E a tarefa deve aparecer visualmente como concluída
E deve ser exibida a mensagem “Tarefa concluída”.

2.4. Fluxos principais

Fluxo de Login
- Acessar /login → informar e-mail e senha → clicar em “Entrar” → acessar /tarefas.

Fluxo de Cadastro de Tarefa
- Na lista de tarefas → clicar “Nova tarefa” → preencher dados → “Salvar” → ver tarefa na lista.

Fluxo de Edição de Tarefa
- Lista de tarefas → selecionar tarefa → “Editar” → alterar dados → “Salvar” → lista atualizada.

Fluxo de Exclusão de Tarefa
- Lista de tarefas → “Excluir” → confirmar exclusão → tarefa removida.

Fluxo de Marcar como Concluída
- Lista de tarefas → clicar “Concluir” → status atualizado → destaque visual.

Fluxo de Logout
- Qualquer tela interna → clicar “Sair” → retornar à tela de login.

2.5. Riscos identificados

Risco 1 – Falhas de validação: aceitar títulos vazios ou muito longos pode prejudicar a organização do u
Risco 2 – Problemas de autenticação: login mal implementado pode permitir acesso indevido ou bloquea
Risco 3 – Usabilidade baixa: botões confusos ou mensagens genéricas podem causar uso incorreto do s
Risco 4 – Perda de dados: erros ao salvar ou excluir tarefas podem gerar perda de informações importan
Risco 5 – Desempenho ruim na listagem: muitas tarefas podem deixar a tela lenta e prejudicar a experiê

3. FASE 2 – PLANO DE TESTE E GESTÃO

3.1. Objetivos do plano de teste
- Verificar se o TaskManager UNISAGRADO atende a todos os requisitos funcionais e não funcionais de
- Garantir que os fluxos principais (login, CRUD de tarefas, filtros, logout) funcionem sem falhas.
- Detectar defeitos antes da entrega final.
- Medir cobertura de requisitos e qualidade geral do sistema.

3.2. Tipos de teste
- Testes de unidade.
- Testes de integração.
- Testes de sistema.
- Testes de aceitação.
- Testes de regressão.
- Teste não funcional (usabilidade e desempenho simples).

3.3. Ferramentas
- Git + GitHub.
- GitHub Issues ou Jira.
- Ferramenta de testes unitários em Python (por exemplo, pytest/unittest).
- Cypress/Playwright ou ferramenta equivalente para testes de interface (UI).
- Postman + Newman para testes de API.
- GitHub Actions para CI/CD.
- Google Docs/Word/Planilhas para documentação.

3.4. Cronograma (exemplo)
- Semana 1: SUT, requisitos, plano de testes, matriz inicial.
- Semana 2: Implementação inicial + casos de teste + massa de dados.
- Semana 3: Execução do 1º ciclo de testes manuais.
- Semana 4: Correções + automação mínima + 2º ciclo (regressão).
- Semana 5: Métricas, relatório final e apresentação.

3.5. Critérios de início e fim dos testes
- Início: funcionalidades principais implementadas, ambiente pronto, casos prioritários documentados.
- Fim: requisitos críticos cobertos, casos executados, sem defeitos críticos abertos, automação mínima r

4. FASE 3 – MATRIZ DE RASTREABILIDADE (modelo)

Colunas sugeridas:
- ID Requisito
- Descrição do Requisito
- ID Caso de Teste
- Descrição do Caso de Teste
- Técnica (CE, VL, TD, E2E, NF)
- Evidência
- ID Defeito
- Status

Exemplo de linhas:
RF01 – Login com e-mail e senha válidos → CT-LOGIN-01 → Login com credenciais corretas → CE → e
RF01 – Login com e-mail e senha válidos → CT-LOGIN-02 → Login com senha inválida → CE → evidên
RF03 – Cadastro de tarefa → CT-TASK-01 → Criar tarefa com título válido → CE → evidência Z → - →
RF09 – Limite de 100 tarefas → CT-TASK-05 → Criar 101ª tarefa → VL → evidência W → DEF-03 → Fa
RNF02 – Desempenho da listagem → CT-PERF-01 → Medir tempo de resposta → NF → evidência T →

5. FASE 4 – CASOS DE TESTE (resumo dos principais)

CT-LOGIN-01 – Login com credenciais válidas (CE)
CT-LOGIN-02 – Login com senha inválida (CE)
CT-TASK-01 – Criar tarefa com título válido (CE)
CT-TASK-02 – Criar tarefa sem título (VL – limite inferior inválido)
CT-TASK-03 – Título com exatamente 100 caracteres (VL – limite superior válido)
CT-TASK-04 – Título com 101 caracteres (VL – acima do limite)
CT-TASK-05 – Limite de 100 tarefas ativas (VL + CE)
CT-E2E-01 – Fluxo completo login → criar → concluir → excluir → logout (E2E)
CT-NF-01 – Desempenho da listagem de tarefas (NF)

6. FASE 5 – DADOS E AMBIENTE (resumo)

Massa de dados:
Usuários de teste: U01 (padrão), U02 (100 tarefas), U03 (sem tarefas).
Tarefas: 100 tarefas ativas para o usuário U02, com títulos “Tarefa 1” a “Tarefa 100”.

Ambiente de teste (exemplo):
- SO: Windows / Linux / macOS.
- Back-end: Python + Flask, com armazenamento em memória ou banco leve (por exemplo, SQLite).
- Front-end: HTML/CSS/JavaScript simples consumindo a API Flask.
- Navegador: Chrome/Firefox.

README (estrutura):
- Pré-requisitos.
- Instalação.
- Configuração.
- Como executar o sistema.
- Como rodar os testes.

7. FASE 6 – EXECUÇÃO MANUAL E DEFEITOS (resumo)

Planilha de execução:
- ID CT, Data, Ciclo, Executor, Resultado, Observações.

Registro de defeitos (Issue):
- Título, Descrição, Passos para reproduzir, Resultado esperado, Resultado obtido, Severidade, Priorida

8. FASE 7 – AUTOMAÇÃO MÍNIMA (resumo)

- Automação de login válido/inválido (UI).
- Automação do fluxo E2E (UI).
- Testes de API (Postman + Newman) para login e tasks.

9. FASE 8 – TDD E CI/CD (resumo)

Exemplo de TDD:
- Teste da função de validação do título (vazio, 100 caracteres, 101 caracteres).

CI/CD (GitHub Actions):
- Workflow para rodar testes unitários em Python e Newman a cada push/pull request.

10. FASE 9 – MÉTRICAS E RELATÓRIO FINAL (resumo)

Métricas:
- Cobertura de requisitos.
- Taxa de aprovação dos testes.
- Defeitos por severidade.
- Reabertura de defeitos.
- Riscos ainda pendentes.

Relatório final:
- Introdução, resumo dos testes, métricas, principais defeitos, lições aprendidas, recomendações.

11. FASE 10 – APRESENTAÇÃO FINAL (roteiro)

- Contexto e SUT.
- Requisitos e fluxos principais.
- Plano de teste e matriz de rastreabilidade.
- Casos de teste (incluindo E2E e não funcional).
- Execução, defeitos e automação.
- TDD, CI/CD e métricas.
- Conclusão.