Estratégia de Testes

Nome do Projeto	VolunTies
Prioridades do Teste	Tendo em vista que se trata de uma plataforma de voluntários, onde irá existir a existência de pessoas como organizações que tenham esse interesse em comum, não se espera encontrar um volume grande como o de uma rede social convencional, o que justifica a prioridade nos seguintes tipos de testes: - Estabilidade - Velocidade - Qualidade - Feedback Vale ressaltar também os seguintes pontos: Para ser bem utilizado em dispositivos diversos e por públicos de diferentes perfis, existe a necessidade de priorizar os testes de usabilidade e de interface de usuário. Outro ponto que exige atenção nos testes é a integração com recursos externos para a utilização de algumas funcionalidades, como o uso das conversas inteligentes com chatbot através da API do Chat GPT ou a integração com redes sociais. Ainda de acordo com a especificação do projeto, gerenciamento de permissões, privacidade e questões de segurança são importantes para o projeto, sendo interessante ter isso como prioridade também.

Identificação dos Tipos de Teste	Usaremos a metodologia do TDD, para o desenvolvimento de um sistema mais mutável, de fácil alteração e manutenção.
	Além disso, seguiremos a pirâmide de testes para a implementação dos testes no sistema, iniciando por testes de unidade e prosseguindo posteriormente para testes de integração.
	Como existem requisitos funcionais e não funcionais, cada parte terá os testes apropriados para a mesma, sendo:
	- Requisitos Funcionais:
	  - Teste Unitário (Automatizado)
	  - Teste de Integração

	- Requisitos Não - Funcionais:
	  - Teste de Usabilidade
	  - Teste de Acessibilidade
	- Infraestrutura:
	  - Teste de Segurança
	
	Pontos importantes sobre alguns tipos de testes:
	
	- Testes de Performance:
	  - Visando a análise da capacidade e resposta da aplicação, será aplicado testes de estabilidade verificando a consistência da mesma ao delongar o uso da aplicação.
	  - Aplicaremos testes de estresse para verificar até onde a aplicação aguenta sem a falha da mesma.
	  - Será utilizado também testes de Carga para quantificar e avaliar a capacidade da aplicação com o aumento do número de usuários usando simultaneamente o sistema.
	- Testes de Segurança:
	  - Testes de Injeção: Será verificada a possibilidade de invasão do sistema a partir de SQL injection, NoSQL Injection etc.Bem como outras vulnerabilidades contidas em formulários UTL e cabeçalhos.
	  - Testes de autenticação e Autorização: Verificaremos se as configurações de acesso e mecanismos de autenticação estão devidamente implementadas e configuradas, e averiguar se funcionalidades protegidas podem ser acessadas por usuários que não possuem as devidas autorizações.
	- Testes Funcionais:
	  - Testes de Integração: Será executada a testagem do sistema em sua totalidade, verificando a conformidade entre os diferentes módulos e funcionalidades do sistema entre si.

Automação dos Testes	Estratégia de automação: - Identificar cenários críticos: Priorizando os testes para as funcionalidades mais importantes, como cadastro, perfil e interações sociais. - Priorizar os testes: Dar ênfase aos testes das áreas mais críticas e frequentemente utilizadas. - Escolher ferramentas de automação: Utilizar ferramentas como Selenium WebDriver, JUnit, TestNG e Cucumber. - Desenvolver scripts de teste: Criar scripts modulares e reutilizáveis para automatizar os cenários identificados.

	- Implementar integração contínua: Configurar um sistema para executar automaticamente os testes sempre que houver uma nova versão do código. - Realizar testes de carga e desempenho: Garantir que o site possa lidar com grandes volumes de usuários sem queda de desempenho. - Manter testes de regressão: Certificar-se de que as alterações no código não quebrem funcionalidades existentes. - Usar o feedback para iterar: Utilizar os resultados dos testes para fornecer feedback aos desenvolvedores e melhorar continuamente o processo de desenvolvimento e os casos de teste. Ferramentas a serem utilizadas para cada nível de teste: Nível de Unidade (Unit Level): Ferramentas: JUnit ou TestNG. Estas ferramentas são ideais para testes de componentes individuais do código, como classes e métodos. Elas são usadas para garantir que cada unidade de código (geralmente uma função ou método) funcione conforme o esperado. Nível de Integração (Integration Level) Ferramenta: Selenium WebDriver. Estas ferramentas são utilizadas para testar a integração entre diferentes componentes do sistema. Por exemplo, testar a interação entre o frontend e o backend, ou entre diferentes microserviços. Nível de Sistema (System Level): Ferramentas: Selenium WebDriver (para testes de interface do usuário), Cucumber (para testes de aceitação), Apache JMeter (para testes de carga e desempenho). Estas ferramentas são usadas para testar o sistema como um todo, garantindo que todas as partes integradas funcionem corretamente juntas. Isso inclui testes de funcionalidade, usabilidade, desempenho e segurança em um ambiente mais próximo do ambiente de produção.

	Serão automatizados todos os testes possíveis que estejam mais relacionados com a lógica e a implementação, ou seja, testes no quesito de usabilidade e acessibilidade não serão automatizados. Já os seguintes testes, serão completamente ou em partes automatizados: - Teste Unitário - Teste de Integração - Teste de Segurança
Etapas dos Testes	Planejamento, preparação, execução, análise, relatórios.
Papéis de Teste na Equipe	Tendo em vista que a equipe seja composta por desenvolvedores, DevSecOps, QA, designers e PO, as responsabilidades sobre os diferentes tipos de testes podem ser divididas da seguinte maneira: - Testes Unitários: primariamente desenvolvedores, podendo ser também pelo DevSecOps e QA - Testes de Integração primariamente desenvolvedores, podendo ser também pelo DevSecOps e QA - Testes de Segurança: primariamente pelo QA e DevSecOps - Testes de Acessibilidade: primariamente desenvolvedores e designers, mas também QA - Testes de Interface e Usabilidade: primariamente designers, com participação do PO e também do QA
Ambientes de Teste	Ambiente de Desenvolvimento: Ambiente onde os desenvolvedores trabalham em novos recursos e correções de bugs. Testes: Testes unitários executados localmente pelos desenvolvedores. Ambiente para integração contínua configurado para executar testes automatizados sempre que há uma nova confirmação de código. Ambiente de Testes: Ambiente dedicado para testar novas funcionalidades e correções antes de serem implantadas no ambiente de produção. Testes: Testes de unidade e integração automatizados. Testes manuais realizados por testadores para garantir a usabilidade e a qualidade geral do sistema.

	Testes de carga e desempenho para avaliar o comportamento do sistema sob carga simulada. Ambiente de Produção: Ambiente onde o site é acessível ao público. Testes: Testes de regressão automatizados para garantir que as alterações não introduzem regressões no sistema em produção. Monitoramento constante da saúde do sistema para detectar e corrigir problemas rapidamente.
Cronograma de Testes	Cada Sprint dura em torno de 2 semanas. Consistiria de tarefas de história de usuário, tarefas técnicas e tickets de bugs. Uma release é entregue a cada sprint de duas semanas. Assim, uma sprint deve durar duas semanas, com entregas regulares no final de cada sprint, totalizando duas sprints por mês e duas releases por mês. Tendo em mente as seguintes divisões de tempo para cada tipo de complexidade durante o desenvolvimento: - Easy backlog - 5 dias de desenvolvimento e 5 dias de teste. - Medium Backlog - 7 dias de desenvolvimento e 3 dias para teste. - Hard Backlog - 8 dias de desenvolvimento e 2 dias para teste.
Ciclo de Vida das Tarefas	Um item é considerado pronto para desenvolvimento quando ele foi documentado e priorizado dentro do backlog. Ele se torna pronto para teste a partir do momento em que ele atinge o seu mvp, ou seja, é capaz de executar de maneira simplória a tarefa para qual está sendo desenvolvido. Ele se torna concluído para desenvolvimento assim que ele atinge seu potencial máximo, executando com maestria a tarefa para qual é designado e recebe uma cobertura de casos de teste de no mínimo 70%. As atividades de teste são inseridas em diferentes estágios do ciclo de vida do desenvolvimento de software. Elas começam no planejamento de requisitos, continuam durante o desenvolvimento com testes unitários e testes de integração, seguidos pela integração contínua e testes de aceitação. Após cada alteração no código, são realizados testes de regressão para garantir a qualidade do software entregue. O objetivo é identificar e corrigir problemas o mais cedo possível.

Monitoramento de Testes	Serão utilizadas as seguintes métricas para monitoramento da efetividade da estratégia de testes: - Cobertura de Testes - Tempo gasto com os Testes