Título: CCC (Community Care Connect)

1. **Finalidade (Justificativa)**
   O Community Care Connect é um projeto destinado a desenvolver uma ferramenta de desktop para ajudar as ONGS e projetos de caridade a acompanhar e gerenciar seus recursos. A justificativa para este projeto reside na necessidade de facilitar o acesso das pessoas a recursos importantes, promovendo assim o bem-estar e a qualidade de vida.

2. **Descrição** (Problema)
   Atualmente, muitas comunidades enfrentam dificuldades para acessar informações atualizadas e precisas sobre os recursos disponíveis em seus estoques. Isso pode levar a problemas como falta de alimentos e assim à assistência social. Além disso, a falta de uma plataforma centralizada para gerenciar essas informações dificulta a coordenação entre organizações locais e autoridades para otimizar a distribuição de recursos. O Community Care Connect visa resolver esse problema, fornecendo uma ferramenta fácil de usar para rastrear, atualizar e acessar informações sobre os recursos comunitários.

3.Objetivo

O objetivo principal do Community Care Connect é desenvolver uma aplicação de desktop que permita às comunidades acompanhar e gerenciar eficientemente os recursos disponíveis em suas áreas locais. Especificamente, o projeto visa:

- Facilitar o acesso dos membros da comunidade a informações sobre recursos como centros comunitários, pontos de coleta de alimentos e clínicas de saúde.
- Promover a colaboração entre organizações locais e autoridades para otimizar a distribuição de recursos.
- Fortalecer a coesão comunitária ao fornecer uma ferramenta útil para os residentes ajudarem uns aos outros.
- Aumentar a conscientização sobre os recursos disponíveis na comunidade e como acessá-los.

4. Critérios para o Sucesso (Benefícios Esperados)

(Metas)

1. Aumento no Acesso e Utilização dos Recursos Comunitários:

- Meta: Melhorar o acesso e a utilização dos recursos comunitários, facilitando que mais pessoas se beneficiem dos serviços disponíveis.

- Benefício Esperado: Maior conscientização e utilização dos recursos disponíveis, resultando em melhorias diretas na qualidade de vida dos membros da comunidade.

2. Melhoria na Coordenação Entre ONGs, Projetos de Caridade e Autoridades Locais:

- Meta: Estabelecer uma plataforma de comunicação eficaz que facilite a colaboração e coordenação entre todas as partes envolvidas.

- Benefício Esperado: Melhor coordenação resulta em respostas mais rápidas e eficientes às necessidades da comunidade, evitando a sobreposição de esforços e garantindo que os recursos cheguem a quem mais precisa.

3. Eficiência na Atualização de Informações:

- Meta: Garantir que as informações sobre os recursos disponíveis sejam atualizadas regularmente com precisão.

- Indicadores de Sucesso: Frequência e precisão das atualizações de dados, número de erros reportados pelos usuários.

- Benefício Esperado: Informações precisas e atualizadas garantem que os usuários possam confiar na ferramenta para tomar decisões informadas sobre onde e como acessar os recursos de que precisam.

5. Equipe [Financiamento]

1. Gestor de Projetos:

   - Nome: Carlos Souza

   - Responsabilidades: Supervisão geral do projeto, garantia de que os prazos e objetivos sejam cumpridos, coordenação das atividades da equipe e comunicação com as partes interessadas.

2. Desenvolvedor de Software:

   - Nome: Cleorbeth Santos

   - Responsabilidades: Desenvolvimento da aplicação de desktop, implementação das funcionalidades necessárias, testes de software e manutenção.

3. Designer de UI/UX:

   - Nome: Giuliano Lima

   - Responsabilidades: Coleta e organização de informações sobre recursos disponíveis, estabelecimento de parcerias com ONGs e autoridades locais, e fornecimento de dados precisos para a aplicação.

4. Especialista em Recursos Comunitários:

   - Nome: Erick Silva

   - Responsabilidades: Coleta e organização de informações sobre recursos disponíveis, estabelecimento de parcerias com ONGs e autoridades locais, e fornecimento de dados precisos para a aplicação.

Financiamento: \\

6. Principais Entregas

1. Protótipo Funcional da Aplicação de Desktop:

    - Descrição: Desenvolvimento de uma versão inicial da aplicação com funcionalidades básicas, incluindo cadastro de recursos, consulta de disponibilidade e interface de usuário intuitiva.

2. Versão Beta para Testes:

    - Descrição: Lançamento da versão beta da aplicação para um grupo selecionado de ONGs e projetos de caridade, com funcionalidades avançadas, como gestão de estoques e relatórios de uso.

3. Programa de Suporte ao Usuário:

    - Descrição: Estabelecimento de um programa de suporte ao usuário, incluindo um canal de atendimento ao cliente e uma seção de perguntas frequentes (FAQ) na aplicação.

7. EAP

1.Planejamento do Projeto
    1.1 Definição do escopo
    1.2 Planejamento dos recursos
    1.3 Planejamento de cronograma
    1.4 Planejamento de comunicação
2.Desenvolvimento do Protótipo
    2.1 Análise de requisitos
        2.1.1 Requisitos funcionais
        2.1.2 Requisitos não funcionais
    2.2 Design da aplicação
        2.2.1 Arquitetura do sistema
        2.2.2 Design de interface (UI/UX)
    2.3 Desenvolvimento do protótipo funcional
        2.3.1 Desenvolvimento de funcionalidades básicas
        2.3.2 Testes Unitários e de integração
3.Versão Beta
    3.1 Desenvolvimento de funcionalidades avançadas
        3.1.1 Gestão de estoques
    3.2 Testes de usabilidade
    3.3 Implementação de feedback
    3.4 Lançamento da versão beta
4.Programa de Suporte ao usuário
    4.1 Desenvolvimento de material de suporte
        4.1.1 FAQ
        4.1.2 Documentação do usuário
    4.2 Estabelecimento do canal de atendimento ao cliente
        4.2.1 Suporte por email
        4.2.2 Suporte telefônico
    4.3 Treinamento de usuários

5.Implementação e Manutenção
    5.1 Implementação na comunidade
        5.1.1 Treinamento de ONGs e Projetos de caridade
        5.1.2 Distribuição e configuração da aplicação
    5.2 Monitoramento e avaliação
        5.2.1 Coleta de feedback
        5.2.2 Análise de dados de uso
    5.3 Atualizações e manutenção
        5.3.1 Correções de bugs
        5.3.2 Atualizações de funcionalidades
6.Encerramento do Projeto
    6.1 Revisão do projeto
    6.2 Documentação final
    6.3 Relatório de lições aprendidas
    6.4 Entrega final

8. Critérios de avaliação

1. Métricas de Sucesso

Indicadores-Chave de Desempenho (KPIs):

Eficiência Operacional:
Tempo de Resolução de Problemas: Tempo médio para resolver problemas técnicos ou bugs relatados.
Tempo de Carregamento da Aplicação: Tempo médio de carregamento da aplicação durante o uso normal.
Tempo de Atualização dos Dados: Tempo médio necessário para atualizar as informações de recursos na aplicação.

Qualidade do Produto:
Taxa de Erros/Defeitos: Número de erros ou defeitos reportados por usuários por mês.
Taxa de Rejeição de Funcionalidades: Percentual de funcionalidades que são rejeitadas pelos usuários após a implementação.
Conformidade com Requisitos: Percentual de requisitos do projeto atendidos conforme especificado nas documentações de requisitos.

Satisfação do Cliente:
Pontuação de Satisfação do Usuário (CSAT): Pontuação média de satisfação dos usuários, coletada através de pesquisas pós-uso.
Net Promoter Score (NPS): Medida da probabilidade de os usuários recomendarem a aplicação a outros.
Número de Solicitações de Suporte: Número de solicitações de suporte recebidas e resolvidas.

Impacto Comunitário:

Número de Recursos Registrados: Quantidade de recursos cadastrados e atualizados na aplicação.

Taxa de Utilização de Recursos: Percentual de recursos registrados que são efetivamente utilizados pelos usuários.

Feedback da Comunidade: Qualidade e quantidade de feedback recebido dos usuários finais sobre a utilidade da aplicação.

2. Processos de Avaliação

Detalhes sobre Avaliação:

Revisões Periódicas:

Mensais: Reuniões de revisão mensal para avaliar o progresso do projeto em relação aos objetivos e metas estabelecidos. Acompanhamento de KPIs operacionais e resolução de quaisquer problemas identificados.

Trimestrais: Avaliações trimestrais mais profundas para revisar o desempenho geral do projeto, avaliar a qualidade do produto e fazer ajustes conforme necessário.

Auditorias de Qualidade:

Auditoria Inicial: Realização de uma auditoria de qualidade após o desenvolvimento do protótipo para garantir que os padrões iniciais sejam atendidos.

Auditorias de Conformidade: Auditorias regulares durante o ciclo de vida do projeto para garantir conformidade com os padrões de qualidade e requisitos do projeto.

Relatórios de Progresso:

Relatórios Semanais: Atualizações semanais para a equipe do projeto e partes interessadas sobre o progresso, problemas e próximas etapas.

Relatórios de Fase: Relatórios detalhados ao final de cada fase principal do projeto (protótipo, versão beta, lançamento) para avaliar o sucesso da fase e preparar para a próxima.

3. Padrões de Qualidade

Padrões de Qualidade a Serem Atingidos:

Desenvolvimento de Software:

Segurança: A aplicação deve atender aos padrões de segurança, incluindo criptografia de dados e proteção contra vulnerabilidades conhecidas.

Desempenho: A aplicação deve ter um tempo de resposta inferior a 2 segundos para operações comuns e um tempo de inatividade máximo de 0,5% ao mês.

Compatibilidade: A aplicação deve ser compatível com as principais versões de sistemas operacionais e deve funcionar em diferentes tipos de hardware com desempenho aceitável.

Interface de Usuário (UI/UX):

Usabilidade: A interface deve ser intuitiva e fácil de usar, com uma taxa de sucesso de navegação mínima de 90% em testes de usabilidade.

Acessibilidade: A aplicação deve seguir as diretrizes de acessibilidade, garantindo que pessoas com deficiência possam usá-la eficazmente.

Documentação e Suporte:

Documentação Completa: Deve haver documentação completa e atualizada disponível para usuários e equipe de suporte, incluindo guias de usuário e manuais técnicos.

Suporte ao Usuário: Um programa de suporte deve estar disponível 24/7 com um tempo de resposta para solicitações de suporte inferior a 24 horas.

9.

10. Hipótese-chave (Viabilidade Tecnológica)

1. Hipóteses-Chave:

Plataforma de Desenvolvimento: A linguagem de desenvolvimento escolhida é a linguagem C

Ferramentas de Desenvolvimento: As ferramentas e bibliotecas de desenvolvimento utilizadas para o desenvolvimento do projeto é a IDE CodeBlocks, e as bibliotecas de C, <stdio.h>, <stdlib.h>, <string.h>

Infraestrutura de Hardware e Software:

Sistema Operacional: A aplicação será compatível com os principais sistemas operacionais de desktop (Windows, macOS, Linux).

Recursos de Hardware: A aplicação funcionará eficientemente em hardware com especificações mínimas recomendadas (processador de 2 GHz, 4 GB de RAM, 500 MB de espaço em disco).

2. Analise Riscos Tecnológicos

Mudanças em Tecnologias de Desenvolvimento:

Descrição: Mudanças nas tecnologias ou ferramentas de desenvolvimento podem afetar o progresso e a manutenção do projeto.

Mitigação: Manter-se atualizado sobre as tendências tecnológicas e planejar atualizações regulares e manutenção para adaptar-se às mudanças.

Problemas de Escalabilidade:

Descrição: A aplicação pode enfrentar problemas de desempenho à medida que o número de usuários e dados cresce.

Mitigação: Projetar a arquitetura da aplicação para escalabilidade desde o início e realizar testes de carga para identificar e resolver problemas de desempenho.

Vulnerabilidades de Segurança:

Descrição: Potenciais vulnerabilidades podem ser exploradas por ataques cibernéticos.

Mitigação: Implementar práticas de segurança robustas, realizar auditorias de segurança regulares e responder rapidamente a quaisquer descobertas de vulnerabilidades.

Integração com Sistemas Externos:

Descrição: Dificuldades na integração com APIs externas ou sistemas podem limitar a funcionalidade da aplicação.

Mitigação: Testar a integração com sistemas externos antecipadamente e garantir que a documentação das APIs esteja completa e acessível.

3. Defina Critérios de Sucesso Tecnológico

Critérios de Sucesso:

Desempenho da Aplicação:

Tempo de Resposta: A aplicação deve ter um tempo de resposta médio inferior a 2 segundos para operações principais.

Estabilidade: A aplicação deve ter uma taxa de falhas inferior a 1% durante testes e uso normal.

Funcionalidade e Usabilidade:

Conformidade com Requisitos: 100% dos requisitos funcionais e não funcionais devem ser implementados e validados.

Satisfação do Usuário: A interface e a usabilidade devem ser avaliadas positivamente em testes de usabilidade, com uma taxa de sucesso mínima de 90% em navegação e operações.

11.Restrições

1. Identifique Restrições do Projeto

Restrições do Projeto:

Orçamento:

Limitação de Recursos Financeiros: O projeto deve ser concluído dentro do orçamento definido, o que inclui custos de desenvolvimento, infraestrutura, manutenção e marketing.

Despesas Imprevistas: Custos não previstos podem surgir e precisam ser geridos de forma eficaz.

Prazos:

Data de Entrega: O projeto deve ser concluído até a data limite especificada para o lançamento da aplicação.

Cronograma de Desenvolvimento: Etapas de desenvolvimento e marcos devem ser concluídos conforme o cronograma para evitar atrasos.

2. Priorize Restrições Críticas

Restrições Críticas:

Orçamento:
O orçamento é uma restrição crítica, pois limita a capacidade de alocar recursos para diferentes aspectos do projeto, como desenvolvimento, testes e marketing. Qualquer desvio orçamentário pode comprometer a viabilidade do projeto.

Prazos:
A data de entrega é essencial para garantir que o projeto seja concluído a tempo para atender às necessidades dos usuários e stakeholders. Atrasos podem afetar a utilidade e o impacto do projeto.

3. Descreva Como Gerenciar Restrições

Planos e Estratégias para Gerenciar Restrições:

Gerenciamento de Orçamento:
Planejamento Financeiro: Desenvolver um plano financeiro detalhado e manter um controle rigoroso dos gastos. Reservar um orçamento contingente para imprevistos.
Negociação e Ajustes: Se necessário, negociar com stakeholders para ajustar o escopo do projeto ou buscar fontes adicionais de financiamento.

Gerenciamento de Prazos:
Planejamento Detalhado: Criar um cronograma detalhado com marcos claros e prazos realistas. Usar ferramentas de gerenciamento de projetos para monitorar o progresso.
Gestão de Riscos: Identificar riscos que podem causar atrasos e desenvolver planos de mitigação para abordá-los de forma proativa.

Gerenciamento de Infraestrutura Tecnológica:

Avaliação e Testes: Avaliar a infraestrutura existente e realizar testes de carga para garantir que ela possa suportar a aplicação.
Planejamento de Escalabilidade: Projetar a aplicação para ser escalável, garantindo que possa acomodar o aumento de usuários e dados.

12.Riscos
Riscos Potenciais:

Tecnologia e Desenvolvimento:
Compatibilidade de Sistema (Impacto: Alto, Probabilidade: Média): A aplicação pode enfrentar problemas de compatibilidade com diferentes sistemas operacionais ou versões de software.
Falhas Técnicas (Impacto: Alto, Probabilidade: Média): Bugs ou falhas no software podem afetar a funcionalidade da aplicação e a satisfação do usuário.

Recursos e Orçamento:

Estouro de Orçamento (Impacto: Alto, Probabilidade: Média): Custos adicionais imprevistos podem exceder o orçamento alocado para o projeto.

Escassez de Recursos (Impacto: Alto, Probabilidade: Média): Falta de recursos financeiros ou humanos pode atrasar o desenvolvimento e comprometer a qualidade.

Prazos e Cronograma:

Atrasos no Cronograma (Impacto: Alto, Probabilidade: Alta): Atrasos no desenvolvimento ou na entrega podem impactar a data de lançamento do projeto.

Dependências de Terceiros (Impacto: Médio, Probabilidade: Média): Dependência de APIs externas ou de terceiros pode levar a atrasos ou problemas de integração.

Planos de Mitigação:

Compatibilidade de Sistema:

Plano de Mitigação: Realizar testes extensivos em diferentes sistemas operacionais e configurações de hardware durante o desenvolvimento. Utilizar ferramentas de virtualização para simular diferentes ambientes.

Falhas Técnicas:

Plano de Mitigação: Implementar uma abordagem de desenvolvimento ágil com ciclos regulares de testes e correção de bugs. Estabelecer um processo de controle de qualidade rigoroso e realizar testes beta com usuários reais.

Estouro de Orçamento:

Plano de Mitigação: Criar um orçamento detalhado e monitorar os gastos de perto. Estabelecer um fundo de contingência para cobrir despesas inesperadas e negociar com fornecedores para obter melhores termos.

Escassez de Recursos:

Plano de Mitigação: Planejar a alocação de recursos com antecedência e considerar a contratação de freelancers ou consultores se necessário. Priorizar tarefas e focar em recursos críticos para manter o projeto dentro do orçamento.

Atrasos no Cronograma:

Plano de Mitigação: Desenvolver um cronograma detalhado com buffer para imprevistos. Monitorar o progresso regularmente e ajustar o cronograma conforme necessário. Implementar um processo de gestão de mudanças para lidar com alterações no escopo.

Falta de Adoção pelos Usuários:

Plano de Mitigação: Realizar pesquisas e testes de usabilidade com usuários finais durante o desenvolvimento. Incorporar feedback dos usuários para melhorar a aplicação antes do lançamento.

13. Critérios de Aprovação:

Fase de Planejamento:

Documentação Completa: Todos os documentos de planejamento (escopo, cronograma, orçamento) devem estar completos e aprovados.

Alinhamento com Stakeholders: As expectativas e requisitos dos stakeholders devem estar documentados e confirmados.

Plano de Risco e Mitigação: O plano de gerenciamento de riscos deve estar elaborado e revisado.

Fase de Desenvolvimento:

Marcos de Desenvolvimento: As funcionalidades principais devem estar concluídas e testadas em ambientes de desenvolvimento.

Qualidade do Código: O código deve passar em revisões de código e testes de qualidade, incluindo testes unitários e de integração.

Documentação Técnica: Toda a documentação técnica deve estar completa e revisada, incluindo especificações e manuais de usuário.

Fase de Testes:

Testes de Aceitação: A aplicação deve passar em todos os testes de aceitação definidos, incluindo testes funcionais, de desempenho e de segurança.

Feedback dos Usuários: A aplicação deve ser testada por um grupo de usuários finais e o feedback deve ser incorporado.

Correção de Defeitos: Todos os defeitos críticos identificados durante os testes devem ser corrigidos e validados.

Fase de Implementação:

Preparação para Lançamento: A aplicação deve estar pronta para produção, com todos os requisitos de infraestrutura e suporte atendidos.

Treinamento de Usuários: O treinamento dos usuários finais e das equipes de suporte deve estar concluído.

Planos de Contingência: Planos de contingência e suporte devem estar estabelecidos para lidar com possíveis problemas pós-lançamento.

Fase de Encerramento:

Aceitação Final: A aplicação deve ser formalmente aceita pelos stakeholders com base na conformidade com os requisitos e critérios estabelecidos.

Documentação Final: Toda a documentação do projeto, incluindo relatórios finais e lições aprendidas, deve estar completa e aprovada.

Entrega Formal: A entrega formal do projeto deve ser realizada, incluindo a transferência de propriedade e suporte contínuo.

Processo de Aprovação:

Planejamento:

Reuniões de Revisão: Realizar reuniões de revisão com os stakeholders para discutir e validar os documentos de planejamento.

Documentação de Aprovação: Obter assinaturas ou confirmações formais dos documentos de planejamento e escopo.

Desenvolvimento:

Revisões de Marco: Agendar reuniões para revisar a conclusão de marcos de desenvolvimento e receber aprovação para avançar para a próxima fase.

Relatórios de Qualidade: Submeter relatórios de qualidade e documentação técnica para revisão e aprovação.

Testes:

Testes de Aceitação: Coordenar com os stakeholders para a execução dos testes de aceitação e revisar os resultados.

Correção e Validação: Aprovar as correções de defeitos e validar os resultados dos testes.

Implementação:

Preparação para Lançamento: Realizar reuniões de preparação para lançamento com stakeholders e obter aprovação para a implantação.

Treinamento e Suporte: Confirmar que o treinamento e suporte estão concluídos e obter aprovação para a implementação final.