Desculpe a confusão, mas como um modelo de inteligência artificial, não consigo criar arquivos, projetos ou acessar sites. Entretanto, eu posso ajudar fornecendo um exemplo de como você pode estruturar seu plano de teste e o que incluir em cada seção. No exemplo abaixo, vou usar a estrutura que você mencionou em sua pergunta:

1. **Plano de Teste Funcional**:

- **Escopo do Teste**: Testar as funcionalidades principais do website, incluindo login, práticas de recrutamento, controle de presença e agendamento de folga dos funcionários.
  - **Casos de Teste**:
    - Caso de Teste 1: Login - inserir os dados de usuário e senha corretos e verificar se o usuário foi capaz de entrar no sistema.
    - Caso de Teste 2: Adicionar um novo candidato para recrutamento, verificar se o candidato foi devidamente inserido no banco de dados.
  (continue com os demais casos para todas as funcionalidades que você deseja testar)
- **Critérios de Aceitação**: Para os testes serem considerados passados, todas as ações deverão ser executadas corretamente, e todas as funcionalidades testadas devem funcionar como esperado.

2. **Projeto de Automação em Cypress**:

Aqui recomendaria que você fosse diretamente em sua pasta de projeto Cypress e começasse a escrever seus scripts de teste. Para cada funcionalidade que você está testando, crie um script Cypress separado. Certifique-se de cobrir todos os casos de teste que você definiu.

3. **Instruções de Execução**:

Instruções detalhadas sobre como executar os testes devem ser incluídas aqui, incluindo a instalação de quaisquer pacotes necessários, como instalar o Node.js e o Cypress, comandos para executar os testes e como interpretar os resultados dos testes.

4. **Plano de Testes de Performance**:

Identifique quais aspectos do site que você gostaria de testar para performance. Isso pode incluir o tempo que leva para as páginas carregarem, o tempo que leva para o servidor responder a uma solicitação, ou até mesmo a capacidade de carga do servidor.

5. **Plano de Testes de Segurança**:

Aqui você deve procurar por quaisquer possíveis falhas de segurança no site e garantir que ele está de acordo com as melhores práticas de segurança da web. Testes de

segurança podem incluir testes de injeção de SQL, XSS, CSRF, etc.

6. **Plano de Testes de Acessibilidade**:

Descreva como você vai garantir que o site está acessível a todos. Por exemplo, você pode verificar se todos os formulários têm rótulos devidamente, que todas as imagens têm texto alternativo para usuários com deficiências visuais, e que o site pode ser usado apenas com um teclado.

7. **Estratégia de Teste de Regressão**:

Determine quando os testes de regressão devem ser executados (por exemplo, antes de cada novo lançamento, após cada alteração significativa, etc.), e quais partes do site devem ser testadas.

8. **Documentação de Integração de APIs**:

Em APIs, recomendamos testar a resposta dos endpoints para solicitações válidas e inválidas, status de resposta correto, objetivo do response body e tempo de resposta.