Obs: tive uma certa dificuldade em mexer e utilizar o Jira, então oque era pra estar la introduzi aqui de certa forma;

Foram um total de 6 sprints:

06/01

07/01

09/01

10/01

16/01

17/01

Analisar requisitos

* Pensar nas funcionalidades a serem aplicadas
* Analisar quais as bibliotecas mais adequadas
* Analisar a possibilidade de imagem estática

Projetar arquitetura

* Procurar arquitetura baseada em processamento de imagem
* Teste de bibliotecas

Implementar captura de imagem

* Importar biblioteca
* Criar instancia da classe de captura
* Utilizar o método read () para captura de imagem
* Tratar com testes a imagem
* Redimensionar o tamanho
* Utilizar a imagem para processamento

Implementar extração de gabarito

* Utilizar função de detecção de contornos
* Utilizar função de corte para extrair a região do gabarito
* Aplicar processamento de imagem na conversão de escala cinza

Implementar identificação de respostas

* Utilizar posições pré-definidas para delimitar área da questão
* Usar função de processamento de imagem para contar a quantidade de pixels
* Utilizar lógica de comparação para determinação de certo ou correto
* Armazenar respostas em um vetor
* Utilizar a lógica de comparação com as respostas do gabarito

Implementar comparação de respostas

* Armazenar as respostas corretas em uma lista ou variável pré-definida.
* Iterar sobre as respostas marcadas pelo aluno e compará-las com as respostas corretas na lista/variável.
* Utilizar uma lógica de comparação (ex: if/else) para verificar se as respostas são iguais ou diferentes.
* Armazenar o número de acertos e erros em variáveis.
* Calcular a pontuação baseada no número de acertos.
* Exibir o número de acertos, erros e pontuação para o usuário.

Testar e depurar sistema

* Criar casos de teste: criar entradas de teste que representem diferentes cenários e condições, incluindo casos de teste válidos e inválidos.
* Executar testes: executar o código com as entradas de teste criadas e comparar os resultados obtidos com os resultados esperados.
* Identificar e corrigir bugs: identificar quaisquer erros ou bugs no código e corrigi-los.
* Realizar testes adicionais: depois que os bugs foram corrigidos, executar novamente os casos de teste para garantir que o sistema esteja funcionando corretamente.
* Documentar testes: documentar os casos de teste e os resultados obtidos para referência futura.

Documentação

* Identificar os elementos do sistema que precisam ser documentados, como o funcionamento geral, a arquitetura, as funcionalidades, entre outros.
* Criar um plano de documentação, estabelecendo os objetivos e os públicos-alvo da documentação.
* Escrever a documentação, incluindo descrições detalhadas das funcionalidades, diagramas de fluxo, exemplos de uso e instruções de instalação e configuração.
* Revisar e editar a documentação, garantindo que ela seja clara, precisa e completa.
* Formatar e publicar a documentação, de forma a torná-la acessível para os usuários finais.
* Continuar mantendo e atualizando a documentação ao longo do ciclo de vida do sistema.

Conclusão:

Lições aprendidas: deixar o trabalho pra fazer de ultima hora sempre da ruim por conta de bugs e erros inesperados, a organização dos arquivos, documentos, scrum board e backlog acredito fazerem mais sentido para times e grupos ordenados no desenvolvimento, no meu caso que fiz sozinho, mais atrapalhou do que ajudou.

Dificuldades:

Organização do tempo, acabei deixando de ultima hora e poderia ter trocado para um projeto mais simples e que já tivesse algum maior contato antes, visto que não pude dar continuidade de certa forma e aperfeiçoamento do projeto atual. Manejo e aprendizado de uma biblioteca nova que eu não tinha nenhum conhecimento; por ser python sempre temos algum problema com configuração de ambiente e uma certa dificuldade pra corrigir alguns erros ainda mais numa biblioteca desconhecida.