### Creating Text from images with OCR API

The goal of this project is to implement an OCR solution that leverages the [Terrasect SDK](https://tesseract.patagames.com/help/html/baa0aa10-7805-4ae6-b6e9-9df777c4678c.htm). The C# application should load images from an input folder. Your task is to develop an application that preprocesses images by shifting, rotating, or applying other suitable transformations. After preprocessing, the application should extract text using the Terrasect API.

The quality of the extracted text depends on factors such as image quality, lighting conditions, the angle of the image, and potentially other variables, which should be assessed during the project.

The final solution must function as a console application that accepts various parameters. The output should be clear and include the extracted text from different preprocessed images. Additionally, the final result must provide a comparison of the extraction quality between various preprocessing approaches.

Here’s a detailed sprint plan for implementing the OCR solution using the Terrasect SDK:

**Sprint 1: Project Setup and Initial Research**

**Objectives**:

* Set up the project environment.
* Familiarize the team with the Terrasect SDK.
* Define input/output formats and parameters for the console application.

**Tasks**:

1. Set up the C# project structure.
2. Research Terrasect SDK documentation to understand its API for OCR operations.
3. Define CLI parameters (e.g., input folder path, preprocessing options, output file location).
4. Create sample input images for testing (varied in lighting, angles, and quality).
5. Write a brief technical design document outlining preprocessing and API integration.

**Deliverables**:

* Project skeleton with CLI support.
* Research summary of SDK capabilities.
* Input/output parameter definitions.

**Sprint 2: Preprocessing Framework Development**

**Objectives**:

* Implement a preprocessing module to handle transformations (e.g., rotation, shifting, and lighting adjustments).
* Ensure transformations are modular and configurable.

**Tasks**:

1. Develop a preprocessing pipeline with options for:
   * Image rotation (fixed and auto-alignment).
   * Brightness and contrast adjustment.
   * Cropping or perspective correction.
2. Test preprocessing transformations on sample images.
3. Implement error handling for preprocessing (e.g., invalid image format).
4. Integrate logging to track preprocessing steps.

**Deliverables**:

* Working preprocessing module with test cases.
* Documentation of preprocessing parameters and expected outputs.

**Sprint 3: OCR Integration and Basic Text Extraction**

**Objectives**:

* Integrate the Terrasect SDK for OCR.
* Extract text from preprocessed images.

**Tasks**:

1. Configure the Terrasect SDK in the application.
2. Develop a module to call the SDK and extract text from images.
3. Test OCR functionality on unprocessed and preprocessed images.
4. Implement error handling for OCR (e.g., API failures, unsupported image formats).
5. Log OCR results for each image.

**Deliverables**:

* Functional OCR module integrated with the preprocessing pipeline.
* Initial test results of OCR performance.

**Sprint 4: Comparative Analysis and Optimization**

**Objectives**:

* Compare OCR results between different preprocessing approaches.
* Optimize the preprocessing steps for improved OCR accuracy.

**Tasks**:

1. Create a scoring system to evaluate OCR output quality (e.g., accuracy, completeness, and readability).
2. Analyze results for different preprocessing methods and parameters.
3. Iterate preprocessing transformations for optimal results.
4. Generate a summary of findings (e.g., which transformations improve accuracy).

**Deliverables**:

* Comparison report of OCR results for different preprocessing approaches.
* Optimized preprocessing settings for best OCR performance.

**Sprint 5: Finalization and Documentation**

**Objectives**:

* Finalize the application and prepare user documentation.
* Ensure the application is robust and easy to use.

**Tasks**:

1. Refactor and clean up code.
2. Add comprehensive logging and error messages for user feedback.
3. Write user documentation:
   * How to install and run the application.
   * Explanation of CLI parameters and output files.
   * Troubleshooting guide.
4. Prepare a demo to showcase the solution.

**Deliverables**:

* Fully functional console application.
* User documentation and troubleshooting guide.
* Demo-ready application showcasing text extraction and preprocessing comparison.