Design Notes

1. Defining the Problem

The goal of the assessment was to develop an application that enables users to check eligibility for the O-1A Visa status by uploading a CV based on some predefined criteria derived from their official website.

2. Choice of Framework

a. **Reason -**
Using FastAPI was the required/recommended approach, perhaps because it is easy to use for creating APIs.

b. **Advantages -**
   i. Validate requests automatically
   ii. Capable of generating automatic docs like Swagger UI
   iii. Supports asynchronous programming that can help in scalability

3. API Design

a. Every endpoint of the API correspond to a specific action related to the main function of checking eligibility (eg: GET, POST, etc)
   i. **GET** - gives a simple welcome message showing that the app is working or says there's an error if the app comes across any error while running
   ii. **POST/ assess** - place where users can upload their file and the server processes the same

b. FastAPI's built-in request validation ensures that files are uploaded in the correct format and if there are missing or empty uploads, the application will throw an error. File format checks are also implemented to ensure that the text can be extracted with the functions that we have.

4. Virtual Environment

a. To ensure that the application can run well across different environments without being affected by the global python packages/version issues, we set up a virtual environment and isolate the dependencies.
b. All the required packages/libraries required to run the application are mentioned in the requirements.txt file.

5. Performance

The application's server base is Uvicorn as it supports asynchronous programming. It can handle multiple simultaneous requests and is pretty lightweight, thus being an ideal choice for API services.

6. Error Handling

There are certain automatic errors thrown by FastAPI such as 404 not found and we also have introduced a few error cases - say if the file isn't in the correct format, or if the text is not being extracted from the uploaded files, etc.

7. Testing

We have to follow the URL that takes us to the interactive API documentation using SwaggerUI. Through this platform, users can test the endpoints directly, providing better insights into how to interact with the API. We can eventually integrate automated testing to test the endpoints by using tools like Pytest.

8. Future Work

a. We can add initial authentication steps considering that the data being uploaded is personal/sensitive to the user.
b. This might be even more important if we integrate a database to store user data.
c. We can implement more advanced text extraction techniques and reduce false negatives, potentially also including BERT embeddings to help with improving keyword detection.
d. We can perhaps modify the system to process multiple CVs in bulk.
e. Ultimately, we can aim at cloud deployment for better scalability and generating detailed PDF reports that give a proper summary of the matched criteria and the scores along with scope of improvements, if possible.