**Working Info**

This task was achieved using the Asana REST API to automate the assignment of task due dates and dynamically adjust deadlines based on task priority and movement. Here's a detailed explanation of the technical approach:

**Technical Approach**

**1. Asana API Integration**

* The script uses Asana's GET and PUT endpoints to retrieve tasks from the project, update task details, and assign due dates.
* Key endpoints:
  + /projects/{project\_id}/tasks - Fetches all tasks in the specified project.
  + /tasks/{task\_id} - Retrieves detailed task information, including custom fields like "Priority."
  + /tasks/{task\_id} (PUT) - Updates the due\_on field to set a new due date for tasks.

**2. Mapping Priority to Deadlines**

* The calculate\_due\_date function maps task priorities to preset deadlines:
  + Low: 14 days from today.
  + Mid: 7 days from today.
  + High: 2 days from today.
* If a task's priority is undefined or unrecognized, it defaults to 14 days.

**3. Dynamic Adjustments for "In Progress" Tasks**

* When a task is moved to the "In Progress" section and has a high priority, the script:
  + Identifies all other tasks in the "In Progress" section.
  + Extends their due dates by 2 days.
* This ensures high-priority tasks receive immediate attention while accounting for workload adjustments.

**4. Error Handling and Resilience**

* The script includes robust error handling to address:
  + Invalid or expired API tokens.
  + Missing or incorrectly configured project sections or task priorities.
  + API rate limits or unexpected server responses.
* Informative logging is included to help debug issues and track progress.

**Trial and Error Methods**

**1. Fetching Task Priority**

* Initially, retrieving the priority field was challenging as it resides in custom fields.
* Solution: Used the /tasks/{task\_id} endpoint to get a detailed view of each task and matched the priority field by its name.

**2. Identifying the "In Progress" Section**

* The script needed to dynamically locate the "In Progress" section in the project.
* Solution: Implemented a loop to fetch all sections and match them by name to ensure flexibility.

**3. Preventing Redundant Adjustments**

* A potential issue was tasks in "In Progress" having their due dates extended multiple times unnecessarily.
* Solution: Added logic to track and skip tasks that had already been adjusted during the current status change.

**4. Handling API Rate Limits**

* Frequent API calls initially triggered 429 Too Many Requests errors.
* Solution: Implemented a retry mechanism with exponential backoff to comply with Asana's API rate limits.

**5. Managing Missing or Invalid Data**

* Some tasks lacked priority or had unexpected values, leading to errors.
* Solution: Defaulted to low priority for such tasks and included logs to flag these cases for further review.

This solution ensures that task deadlines are dynamically managed based on their priority and progress, providing a seamless and efficient way to handle project management workflows in Asana.