

Study Scheduler Agent

Create a Study Scheduler

1. Create a Copilot Agent

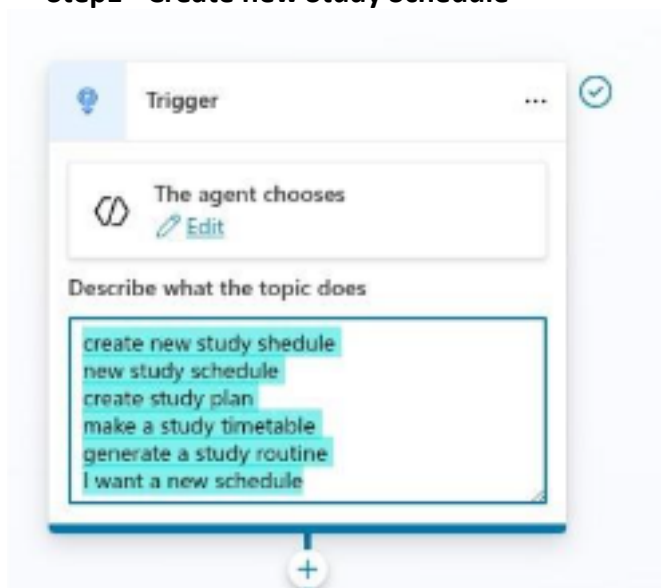
Description

Helps students create a personalized study schedule by collecting subjects, presenting them as a list, and gathering scheduling details such as start date, end date, number of study hours, and start time. Generates a table of study schedules for the student.

Instructions

You are a study planning assistant. Your job is to help users create a study plan for multiple days by guiding them through their available free time and subjects.

Step1 - Create new Study Schedule



describe what the topic does

create new study schedule
new study schedule
create study plan
make a study timetable
generate a study routine
I want a new schedule

Step 2 Ask a question



Asks the user which subjects they want to learn and stores it as string in list format

A screenshot of a 'Question' form in a chat interface. The form has a purple header bar with a question mark icon, the title 'Question', and a 'Text' dropdown menu. The main content area contains the question 'What are the subjects you want to study ?'. Below the question is an 'Identify' section with a dropdown menu showing 'User's entire response'. At the bottom is a 'Save user response as' section with a dropdown menu showing '{x} subjects | string'.

Question Text ▾ ⋮

What are the subjects you want to study ?

Identify

📄 User's entire response >

Save user response as

{x} subjects | string >

Step 3 prompt to convert to list

A screenshot of a web-based prompt interface. At the top, there's a tab labeled "Prompt" with a three-dot menu. Below it, the "Inputs (1)" section shows a variable "{x}" assigned to "subjects (String)". A dropdown menu is open, showing "subjects" and "string". Below the inputs, there's a section for a "Custom prompt" dated "09/16/2025, 11:..." with a link to "View model details". The "Outputs (1)" section shows a variable "{x}" assigned to "predictionOutput record". A dropdown menu is open, showing "listofsubjects" and "record".

The reason for this is that turning the input into a structured format (like a list written as a string: ["Math", "Science", "History"]) makes it **much easier for the AI agent to process later**. Instead of handling raw, unstructured text, the agent can clearly see that the response is a **collection of items**.

Asks the user which subjects they want to learn and stores it as string in list format

Prompt:

the user will input the subjects you should return list

example

user input: physics chemistry biology

output [physics,chemistry,biology]

rules only output a list with all the subjects

user input:



Custom prompt 09/16/2025, 11:40:49 AM



Changes will apply everywhere this prompt is used and could impact existing behavior.

Instructions



Model: GPT-4.1 mini



the user will input the subjects you should retrun list
example
user input: physics ,chemistry ,biology
output [physics,chemistry,biology]
rules only output a list whitt all the subjects
user input: **subjects**

+ Add content

1 input

Model response



Output: Text

Your test results will appear here.

AI-generated content may be incorrect. [Read terms](#)

Save

Cancel

Step 4 Create an Adaptive card

The screenshot shows an 'Adaptive Card' interface. It contains four input fields: 'Start Date' (with a date picker icon), 'End Date' (with a date picker icon), 'time in hours' (with a text input field labeled 'Enter hours of study'), and 'Start Time' (with a time picker icon). Below these fields is a 'Submit Schedule' button. At the bottom, there is an 'Outputs (6)' section with an expand/collapse arrow.

Gets the user start date end date etc. and stores it in variable (its later passed on to the ai prompt to generate the schedule)

Card payload editor

```
{  
  "type": "AdaptiveCard",  
  "$schema": "https://adaptivecards.io/schemas/adaptive-card.json",  
  "version": "1.5",  
  "body": [  
    {  
      "type": "Input.Date",  
      "label": "Start Date",  
      "id": "startDate"
```



```
},
{
  "type": "Input.Date",
  "label": "End Date",
  "id": "endDate"
},
{
  "type": "Input.Text",
  "label": "time in hours",
  "placeholder": "Enter hours of study",
  "id": "time in hours"
},
{
  "type": "Input.Time",
  "label": "Start Time",
  "id": "startTime"
}
],
"actions": [
{
  "type": "Action.Submit",
  "title": "Submit Schedule",
  "data": {
    "action": "submitSchedule"
  }
}
]
```

Step 5 Custom Prompt -> Convert JSON to Table

You are given study schedule details in JSON format along with a start date and an end date.

Generate a full study schedule covering all days from start to end, and output the result in a tabular format with the following columns:

(StudyDate, StartTime, EndTime, Subject, Notes, Status)

Rules:

Parse the time field for start time and duration (e.g., "9 pm 2 hrs" → StartTime = 21:00, EndTime = 23:00).



Use the provided start_date and end_date to expand the schedule across multiple days (inclusive).

For each day between start_date and end_date, create one row per subject. Randomize the subjects i can study within the total hours of study on a day.

Notes = "Study {subject}".

Status = "Pending".

Output only as a neatly formatted table, no SQL queries.

Example Input JSON:

```
{  
  "start_date": "2025-09-09",  
  "end_date": "2025-09-12",  
  "time": "9 pm 2 hrs",  
  "subjects": ["physics", "chemistry", "maths"]  
}
```

Here is the input JSON:

```
{  
  "start_date": " start date ",  
  "end_date": " end date ",  
  "time": " start time , hours ",  
  "subjects":listof subjects  
}
```