Okay! Here's a breakdown of how to build a basic project management tool in Python, along with explanations and code examples. I'll use the Typer library for a streamlined command-line interface, but you can easily adapt this to a web framework if you prefer.

# \*\*Installation (If you haven't already):\*\*

# pip install typer rich

import typer

import datetime

from rich.table import Table

from rich.console import Console

app = typer.Typer()

console = Console()

# \*\*Project Data Structure:\*\*

# For simplicity, we'll store project data in a list of dictionaries.

# In a real-world application, you'd use a database for persistence.

projects = []

# \*\*Project Class:\*\*

class Project:

def \_\_init\_\_(self, name, description, due\_date=None):

self.name = name

self.description = description

self.due\_date = due\_date

self.tasks = []

self.id = len(projects) + 1 if projects else 1

def add\_task(self, description, assignee=None):

task\_id = len(self.tasks) + 1

self.tasks.append({

"id": task\_id,

"description": description,

"status": "To Do",

"assignee": assignee

})

def update\_task\_status(self, task\_id, new\_status):

for task in self.tasks:

if task["id"] == task\_id:

task["status"] = new\_status

break

else:

console.print(f"[red]Error: Task with ID {task\_id} not found.[/]")

def display\_tasks(self):

if not self.tasks:

console.print(f"[yellow]Project '{self.name}' has no tasks yet.[/]")

return

table = Table(title=f"Tasks for Project: [blue]{self.name}[/]")

table.add\_column("ID", style="dim", width=5)

table.add\_column("Description")

table.add\_column("Status", justify="right")

table.add\_column("Assignee", justify="right")

for task in self.tasks:

table.add\_row(str(task["id"]), task["description"], task["status"], str(task.get("assignee", "")))

console.print(table)

# \*\*Typer Commands:\*\*

@app.command()

def create\_project(

name: str = typer.Argument(..., help="Name of the project"),

description: str = typer.Option("", "--desc", "-d", help="Project description"),

due\_date: datetime.date = typer.Option(None, "--due", formats=["%Y-%m-%d"], help="Due date (YYYY-MM-DD)")

):

"""Create a new project."""

project = Project(name, description, due\_date)

projects.append(project)

console.print(f"[green]Project '{name}' created successfully![/]")

@app.command()

def add\_task(

project\_name: str = typer.Argument(..., help="Name of the project to add the task to"),

description: str = typer.Argument(..., help="Description of the task"),

assignee: str = typer.Option(None, "--assign", "-a", help="Assignee for the task")

):

"""Add a task to a project."""

for project in projects:

if project.name == project\_name:

project.add\_task(description, assignee)

console.print(f"[green]Task added to '{project\_name}'![/]")

return

console.print(f"[red]Error: Project '{project\_name}' not found.[/]")

@app.command()

def update\_status(

project\_name: str = typer.Argument(..., help="Name of the project"),

task\_id: int = typer.Argument(..., help="ID of the task to update"),

new\_status: str = typer.Argument(..., help="New status (e.g., 'In Progress', 'Completed')")

):

"""Update the status of a task."""

for project in projects:

if project.name == project\_name:

project.update\_task\_status(task\_id, new\_status)

console.print(f"[green]Status of task {task\_id} in project '{project\_name}' updated![/]")

return

console.print(f"[red]Error: Project '{project\_name}' not found.[/]")

@app.command()

def view\_tasks(project\_name: str = typer.Argument(..., help="Name of the project")):

"""View all tasks in a project."""

for project in projects:

if project.name == project\_name:

project.display\_tasks()

return

console.print(f"[red]Error: Project '{project\_name}' not found.[/]")

# \*\*Run the application:\*\*

if \_\_name\_\_ == "\_\_main\_\_":

app()

content\_copy Use code [with caution](https://support.google.com/legal/answer/13505487).Python

**To use your project management tool:**

1. **Save:** Copy the code and save it as a Python file (e.g., project\_manager.py).
2. **Run:** Open a terminal or command prompt and execute: python project\_manager.py --help to see the available commands.

**Example Usage:**

# Create a project:

python project\_manager.py create-project "My Project" --desc "This is my project" --due 2024-03-10

# Add tasks:

python project\_manager.py add-task "My Project" "Write project proposal"

python project\_manager.py add-task "My Project" "Design the database schema" --assign "Alice"

# View tasks:

python project\_manager.py view-tasks "My Project"

# Update task status:

python project\_manager.py update-status "My Project" 1 "In Progress"

# View updated tasks:

python project\_manager.py view-tasks "My Project"

content\_copy Use code [with caution](https://support.google.com/legal/answer/13505487).Bash

**Key improvements:**

* **Error Handling:** Basic error handling to prevent common issues (e.g., updating a task in a non-existent project).
* **User Feedback:** More informative and colorful console output.
* **Clearer Structure:** Class-based organization for better maintainability.
* **Assignment Feature:** Ability to assign tasks to team members.

**Expanding Your Project:**

This code provides a foundation for a more sophisticated tool:

* **Persistence:** Replace the projects list with database integration (e.g., SQLite or PostgreSQL).
* **GUI:** Use libraries like Tkinter, PyQt, Kivy, or web frameworks like Flask or Django for a graphical interface.
* **Advanced Features:**
  + Task dependencies
  + Calendar view
  + File attachments
  + Collaboration features

Feel free to ask any further questions.