Here’s a general outline of the steps to complete your task manager API project, from setting up the environment to deploying your API:

**Step 1: Project Setup**

1. **Create a New Project Directory**  
   Set up a directory structure for your project.
   * app/: This will hold the core application files (e.g., routes, models).
   * tests/: Contains unit tests for the API.
   * config/: Configuration files for environment variables and settings.

* In gitbash, create folder and set as working directory:

mkdir ~/Desktop/task\_manager\_api

cd ~/Desktop/task\_manager\_api

* Create sub directories

mkdir app tests migrations

touch app/\_\_init\_\_.py app/models.py app/routes.py app/config.py app/services.py

touch run.py requirements.txt .env README.md

* + The generated files are:
    - Run.py - This is where you’ll start your Flask app.
    - Requirements.txt - To list all the dependencies your project needs.
    - .env - To store environment variables securely.
    - README.md - To document your project.

1. **Set Up a Virtual Environment**  
   Create a virtual environment and install dependencies like Flask/FastAPI, SQLAlchemy (for database interaction), JWT (for authentication), and other required libraries.

bash

CopyEdit

python -m venv venv

source venv/bin/activate # On Windows, use `venv\Scripts\activate`

pip install flask flask\_sqlalchemy flask\_jwt\_extended

1. **Create a Git Repository**  
   Initialize a Git repository for version control.

bash

CopyEdit

git init

git add .

git commit -m "Initial commit"

**Step 2: Set Up Database**

1. **Choose a Database**  
   For simplicity, start with SQLite for local development. You can migrate to PostgreSQL or MySQL later for production.
2. **Create Models for Database**
   * Create a User model for authentication (username, email, hashed password).
   * Create a Task model (name, description, due date, priority, category, user\_id).
   * Create a Category model (name).
3. **Set Up SQLAlchemy**  
   Set up a connection to the database using SQLAlchemy.

Example:

python

CopyEdit

from flask\_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

1. **Create Database Tables**  
   Use SQLAlchemy to create tables for users, tasks, and categories.

python

CopyEdit

db.create\_all() # After defining your models

**Step 3: Authentication**

1. **Set Up JWT**  
   Install flask\_jwt\_extended for JWT handling and configure it in your app.

python

CopyEdit

from flask\_jwt\_extended import JWTManager

app.config['JWT\_SECRET\_KEY'] = 'your\_secret\_key'

jwt = JWTManager(app)

1. **Create User Registration Endpoint**  
   Create a POST /register endpoint to allow users to sign up.
2. **Create Login Endpoint**  
   Create a POST /login endpoint to issue JWT tokens when users log in with valid credentials.

**Step 4: Task Management**

1. **Create Task Endpoints**
   * POST /tasks: Allow users to create tasks.
   * GET /tasks: Retrieve all tasks, optionally filtered by category, due date, or priority.
   * PUT /tasks/{id}: Update a task.
   * DELETE /tasks/{id}: Delete a task.
2. **Implement Task Filtering**  
   Allow users to filter tasks by category, due date, and priority by adding query parameters in the GET /tasks endpoint.

**Step 5: Category Management**

1. **Create Category Endpoints**
   * POST /categories: Create a new category for tasks.
   * GET /categories: Get all categories.
   * PUT /categories/{id}: Update a category.
   * DELETE /categories/{id}: Delete a category.

**Step 6: Implement Task Priority**

1. **Add Priority Field to Task Model**  
   Add a priority field to the Task model (e.g., Low, Medium, High).
2. **Allow Sorting/Filtering by Priority**  
   Implement filtering or sorting for tasks based on priority.

**Step 7: Testing**

1. **Write Unit Tests**  
   Write tests to ensure that your API works as expected. Test user registration, login, task creation, updating tasks, and filtering tasks.

Use a testing framework like pytest to write your test cases.

1. **Mock the Database**  
   Use tools like pytest-flask and Factory Boy to mock database operations for testing.

**Step 8: Deployment**

1. **Choose a Deployment Platform**  
   Deploy your API to a platform like Heroku, AWS, or DigitalOcean.
2. **Prepare for Production**
   * Set up environment variables for secrets (like the JWT secret and database URI).
   * Configure logging and monitoring for the production environment.
3. **Deploy**  
   Follow the instructions of the chosen platform to deploy the Flask/FastAPI app.