Beginner-Friendly Roadmap for AutoPilot PM – AI Product Manager Agent

A step-by-step guide to building your AI Product Manager. We'll break it into **8 phases**, focusing on core functionality first.

Phase 0: Foundation Setup

Goal: Set up tools & basic environment.
Tasks:

1. **Install Tools**:
    - Python 3.10+
    - VS Code (or PyCharm)
    - Git & GitHub account
2. **Create Project**:

    ```
    mkdir autopilot-pm
    cd autopilot-pm
    python -m venv venv
    source venv/bin/activate  # Linux/Mac | venv\Scripts\activate (Windows)
    ```

3. **Initialize Files**:

    ```
    touch requirements.txt app.py .env
    ```

Phase 1: Backend + Basic AI Agent

Goal: Build a FastAPI server + simple AI agent.
Tasks:

1. **Install Dependencies**:

    ```
    pip install fastapi uvicorn langchain langgraph openai python-dotenv
    ```

2. **Create `app.py`**:

    ```
    from fastapi import FastAPI
    from langchain.agents import AgentExecutor, create_react_agent
    from langchain_community.llms import OpenAI

    app = FastAPI()
    llm = OpenAI(model="gpt-3.5-turbo")

    @app.post("/plan")
    async def plan_goal(goal: str):
        agent_prompt = f"Break this product goal into tasks: {goal}"
        return llm.invoke(agent_prompt)
    ```

3. **Test Locally**:

    ```
    uvicorn app:app --reload
    ```

    Visit `http://localhost:8000/docs` to test the `/plan` endpoint.

Phase 2: Task Breakdown Logic

Goal: Make the AI split goals into actionable tasks.
Tasks:

1. **Improve the Agent**:
    - Use **ReAct prompting** for step-by-step reasoning.
    - Example prompt:

        ```
        template = """
        You are an AI Product Manager. Break the goal into subtasks:
        Goal: {goal}
        Steps:
        1. Research → 2. Design → 3. Development → 4. Testing → 5. Launch
        Output: JSON list of tasks.
        """
        ```

2. **Add Output Parsing**:
   - Use `LangChain’s Output Parsers` to get structured JSON tasks.

Phase 3: Jira Integration

Goal: Connect to Jira to create/manage tasks.
Tasks:

1. **Get Jira API Access**:
   - Create a Jira account & generate API token.
2. **Install Jira Library**:

   ```
   pip install jira
   ```

3. **Create Jira Tool**:

   ```
   from jira import JIRA

   def create_jira_task(summary: str, description: str):
       jira = JIRA(server="https://your-domain.atlassian.net", basic_auth=("email", "api_token"))
       issue = jira.create_issue(project="PROJ", summary=summary, description=description, issuetype={"name": "Task"})
       return issue.key
   ```

4. **Add to Agent**:
   - Teach the AI: *“If a task is technical, create a Jira ticket.”*

Phase 4: GitHub + Slack Integration

Goal: Monitor code & communicate with teams.
Tasks:

1. **GitHub Integration**:
   - Use PyGithub to track PRs/issues.

   ```
   from github import Github
   g = Github("github_token")
   repo = g.get_repo("owner/repo")
   pulls = repo.get_pulls(state='open')
   ```

2. **Slack Integration**:
   - Use Slack SDK to send messages:

   ```
   from slack_sdk import WebClient
   client = WebClient(token="slack_token")
   client.chat_postMessage(channel="#dev-team", text="Task created: {task}")
   ```

Phase 5: Basic Dashboard (Frontend)

Goal: Build a simple UI to input goals/view progress.
Tasks:

1. **Set Up React App**:

   ```
   npx create-react-app frontend
   cd frontend
   npm install axios tailwindcss
   ```

2. **Create Input Form** (`src/App.js`):

```
function App() {
  const [goal, setGoal] = useState("");
  const handleSubmit = async () => {
    await axios.post("http://localhost:8000/plan", { goal });
  };
  return (
    <div>
      <input value={goal} onChange={(e) => setGoal(e.target.value)} />
      <button onClick={handleSubmit}>Plan</button>
    </div>
  );
}
```

Phase 6: Memory + Feedback Loop

Goal: Make the AI learn from past mistakes.
Tasks:

1. **Add Memory**:

```
from langchain.memory import ConversationBufferMemory
memory = ConversationBufferMemory()
agent = create_react_agent(llm, tools, prompt, memory=memory)
```

2. **Implement Reflection**:
   - After each task, ask the AI:
     "What failed? How to improve next time?"
   - Save insights to a `reflections.json` file.

Phase 7: Reports + Automations

Goal: Generate reports & automate standups.
Tasks:

1. **Auto-Generate Reports**:
   - Use LLM to write summaries:

   ```
   report_prompt = "Generate a weekly status update using {Jira data} and {GitHub activity}."
   ```

2. **Schedule Daily Standups**:
   - Use `APScheduler` to trigger daily Slack summaries:

```
from apscheduler.schedulers.background import BackgroundScheduler

def daily_report():
    report = llm.invoke(report_prompt)
    send_slack_message(report)

scheduler = BackgroundScheduler()
scheduler.add_job(daily_report, 'cron', hour=9)  # 9 AM daily
```

Phase 8: Testing + Deployment

Goal: Launch your MVP.
Tasks:

1. **Test Core Flows**:
   - Goal → Tasks → Jira → Slack updates → Report.
2. **Deploy**:
   - Backend: Deploy FastAPI to Railway or Heroku.
   - Frontend: Host React app on Vercel.
3. **Add Monitoring**:
   - Log AI decisions in a dashboard (e.g., `Loguru` for Python).

Tools & Resources

Purpose	Tools

Purpose	Tools
Backend	FastAPI, Uvicorn, Celery
AI Agent	LangChain, LangGraph, GPT-4/Claude
APIs	Jira REST API, GitHub API, Slack API
Frontend	React, TailwindCSS, Axios
Deployment	Railway (backend), Vercel (frontend)
Learning	LangChain Docs, FastAPI Tutorial

Timeline

- Phases 1-3: Week 1-2
- Phases 4-6: Week 3-4
- Phases 7-8: Week 5

Start small → test each step → iterate! □
Tip: Use mock APIs first (e.g., `pytest-mock`) before integrating real tools.