1) MVP Scope = “Track tasks”

- View a list of tasks
- Add a new task (title + optional course/category + due date optional)
- Mark a task as complete / incomplete
- Delete a task
- Basic input validation (can’t add an empty title)
- Data stored simply (either **in-memory** for the session OR a single **JSON file** in the project folder)

Out of scope for MVP (NOT in Sprint 2)

- Authentication / accounts
- Database (SQLite/Postgres)
- Advanced styling / UI frameworks
- Reminders/notifications
- Sharing/collaboration between users
- Analytics/charts

Objectives

By the end of Sprint 3, the team should have:

- A working Flask app that runs locally
- A task list page that shows tasks
- Ability to add, complete, and delete tasks
- Tasks persist during use (and ideally across restarts via JSON file)
- Basic README instructions to run the app

---

2) Define the MVP

MVP Description:
Our app is a lightweight task tracker that helps users quickly organize what they need to work on. It solves the problem of losing track of to-dos across multiple categories. The intended users are students or persons who want a simple way to list tasks and see what’s next. The primary action is adding a task and viewing it in a list. Users can mark tasks complete when finished and delete tasks they no longer need. The app keeps the experience minimal by avoiding accounts, databases, and complex design. The MVP is still useful because it provides a single place to record tasks and track completion status. If time allows, tasks will be saved to a simple JSON file so they persist after restarting the program.

---

3) Requirements for the MVP

Functional Requirements (what the app must do)

FR1. The system shall display all tasks on a main page (Task List).
FR2. The system shall allow the user to add a new task via a form.
FR3. The system shall validate task input and reject empty titles.
FR4. The system shall allow the user to mark a task as complete/incomplete.
FR5. The system shall allow the user to delete a task.
FR6. The system shall store tasks in a simple structure (in-memory list and/or JSON file).
FR7. The system shall show a confirmation message or visual indication when actions succeed or fail (simple text is fine).

Non-Functional Requirements (what the app won't do)

NFR1. The app shall run locally using Flask on Windows/macOS/Linux.
NFR2. The app shall not require a database or authentication for MVP.
NFR3. The UI shall be minimal HTML (basic readability; no advanced styling required).
NFR4. The app shall handle invalid input gracefully (no server crash on blank form or bad request).
NFR5. The app should load and display the task list quickly for a small number of tasks (e.g., under 200).
NFR6. Code shall be committed to GitHub incrementally with clear commit messages and PR reviews (team collaboration expectation).

---

4) User Stories

US1 — View tasks

As a user, I want to see a list of my tasks so that I know what I need to work on.
Acceptance Criteria:

- When I open the home page, I see a list of tasks
- Each task shows at least: title + status (complete/incomplete)

US2 — Add a task

As a user, I want to add a new task so that I can track new ones.
Acceptance Criteria:

- There is an "Add Task" form
- Submitting a valid title adds it to the list
- Submitting an empty title shows an error and does not add a task

US3 — Mark complete/incomplete

As a user, I want to mark a task complete so that I can track progress.
Acceptance Criteria:

- Each task has a way to toggle status
- Completed tasks clearly look "done" (even just text like "Completed")

US4 — Delete a task

As a user, I want to delete a task so that I can remove tasks I don't need anymore.
Acceptance Criteria:

- Each task has a delete action
- After deletion, it no longer appears in the list

US5 — Persist tasks (optional but still simple)

As a user, I want my tasks saved so that I don't lose them when I restart the app.
Acceptance Criteria:

- If JSON persistence is used: tasks re-appear after restarting the server
- If time doesn't allow, document that MVP persistence is "session-only"

---

5) Product Backlog

Move only Sprint-3-feasible items into **Sprint Backlog**.

Epic A — Project Setup

1. **Create Flask project scaffold**
    - Acceptance: app runs, basic route returns a page
2. **Add README run instructions**
    - Acceptance: README includes how to install/run locally

Epic B — Tasks Core Features

3. **Task data model (simple structure)**
    - Acceptance: tasks stored as dicts (id, title, done, optional fields)
4. **Task list page (GET /)**
    - Acceptance: shows tasks with status
5. **Add task endpoint + form**
    - Acceptance: adds task, validates input

6. **Toggle complete endpoint**
    - Acceptance: user can mark done/undone
7. **Delete task endpoint**
    - Acceptance: removes task

Epic C — Persistence (Simple)

8. **Optional: JSON save/load**
    - Acceptance: tasks persist after restart

Epic D — Quality / Collaboration

9. **Basic error handling**
    - Acceptance: bad inputs don't crash app
10. **Team workflow: PR checklist**
- Acceptance: team uses PR + review process consistently

---

6) Sprint 3 Kanban Board Setup

Required Columns:

Sprint Backlog → To Do → In Progress → In Review → Done

What to put into Sprint Backlog

Move these into **Sprint Backlog**:

- Create Flask project scaffold
- Task data model (simple structure)
- Task list page
- Add task form + endpoint (with validation)
- Toggle complete
- Delete task
- README run instructions

Keep these in **Product Backlog** (not Sprint Backlog yet):

- JSON save/load (if you think it might be tight)
- Extra features like filter/search/sort, due dates, categories
- Styling improvements

How to use the board correctly

- Only issues you truly expect to finish go into **Sprint Backlog**
- From Sprint Backlog, pull into **To Do** when someone is ready to start
- Use **In Review** for PR stage

---

7) README content

Project Name

Task Trackr (Productivity Web App)

Short Overview

Task Trackr is a lightweight Flask web app that helps students/users track tasks in one place. It focuses on core productivity features (view, add, complete, delete tasks) without accounts, databases, or heavy styling.

MVP Description

Our app is a lightweight task tracker that helps users quickly organize what they need to work on. It solves the problem of losing track of to-dos across multiple categories. The intended users are students or persons who want a simple way to list tasks and see what's next. The primary action is adding a task and viewing it in a list. Users can mark tasks complete when finished and delete tasks they no longer need. The app keeps the experience minimal by avoiding accounts, databases, and complex design. The MVP is still useful because it provides a single place to record tasks and track completion status. If time allows, tasks will be saved to a simple JSON file so they persist after restarting the program.