Sprint 0: Planning Document

Product Vision

A simple task management REST API that allows users to create, read, update, and delete tasks with priority levels and completion status. Designed for individual users who need a lightweight backend service for organizing personal tasks.

Product Backlog

User Story 1: Create a Task

As a user
I want to create a new task with a title and description
So that I can track work items

Acceptance Criteria:

- API endpoint POST `/tasks` accepts title (required) and description (optional)
- System returns created task with unique ID and timestamp
- Title must be a non-empty string
- Invalid requests (missing title) return 400 error with message

Story Points: 3
Priority: High (P1)

User Story 2: List All Tasks

As a user
I want to list all my tasks
So that I can see what needs to be done

Acceptance Criteria:

- API endpoint GET `/tasks` returns array of all tasks
- Returns empty array `[]` if no tasks exist
- Each task includes id, title, description, completed status, priority, and timestamps

Story Points: 2
Priority: High (P1)

User Story 3: View Single Task

As a user
I want to view a single task by its ID
So that I can see its full details

Acceptance Criteria:

- API endpoint GET `/tasks/:id` returns single task object
- Returns 404 error if task with given ID does not exist
- Response includes all task fields

Story Points: 2
Priority: High (P1)

User Story 4: Update a Task

As a user
I want to update a task's details or mark it as complete
So that I can track my progress and fix mistakes

Acceptance Criteria:

- API endpoint PATCH `/tasks/:id` accepts partial updates
- Can update: title, description, completed (boolean), priority
- Returns updated task object
- Returns 404 if task not found
- Returns 400 for invalid data

Story Points: 3

Priority: Medium (P2)

User Story 5: Delete a Task

As a user

I want to delete a task

So that I can remove items I no longer need

Acceptance Criteria:

- API endpoint DELETE `/tasks/:id` removes the task
- Returns 204 No Content on successful deletion
- Returns 404 if task not found

Story Points: 2

Priority: Medium (P2)

User Story 6: Assign Priority Levels

As a user

I want to assign priority levels to tasks

So that I can focus on important work first

Acceptance Criteria:

- Tasks can have priority: "low", "medium", or "high"
- Default priority is "medium" if not specified
- Priority can be set on creation and updated later
- Invalid priority values return 400 error

Story Points: 2

Priority: Medium (P2)

User Story 7: Filter Tasks

As a user
I want to filter tasks by completion status
So that I can see only completed or pending tasks

Acceptance Criteria:

- GET `/tasks?completed=true` returns only completed tasks
- GET `/tasks?completed=false` returns only incomplete tasks
- GET `/tasks` without filter returns all tasks

Story Points: 3
Priority: Low (P3)

Backlog Summary

Story	Title	Points	Priority
US-1	Create a Task	3	High (P1)
US-2	List All Tasks	2	High (P1)
US-3	View Single Task	2	High (P1)
US-4	Update a Task	3	Medium (P2)
US-5	Delete a Task	2	Medium (P2)
US-6	Assign Priority Levels	2	Medium (P2)
US-7	Filter Tasks	3	Low (P3)

Total Backlog: 17 Story Points

Definition of Done (DoD)

A user story is considered "Done" when ALL of the following are true:

1. **Code Complete:** All code is written and committed to the main branch
2. **Tests Passing:** Unit tests are written and pass successfully
3. **CI Pipeline Green:** GitHub Actions pipeline runs without failures
4. **Acceptance Criteria Met:** All acceptance criteria have been verified
5. **Code Quality:** Code follows project conventions (consistent naming, proper error handling)
6. **No Critical Bugs:** No known critical or blocking bugs remain
7. **Documentation:** README updated if new setup steps are needed

Sprint 1 Plan

Sprint Goal: Deliver core CRUD functionality and establish CI/CD pipeline

Selected User Stories:

1. US-1: Create a Task (3 points)
2. US-2: List All Tasks (2 points)
3. US-3: View Single Task (2 points)

Additional Sprint 1 Tasks:

- Set up project structure (Express.js, folder organization)
- Initialize Git repository with proper .gitignore
- Configure GitHub Actions CI pipeline
- Write initial unit tests
- Create README with setup instructions

Total Committed Points: 7

Sprint 1 Definition of Done Checklist:

- [ ] Project scaffolded with Express.js
- [ ] Git repository created with incremental commits
- [ ] CI pipeline configured and running
- [ ] US-1, US-2, US-3 implemented and tested
- [ ] All tests passing in pipeline
- [ ] Sprint Review document created
- [ ] Retrospective completed

Sprint 2 Plan (Tentative)

Sprint Goal: Complete remaining CRUD operations and add monitoring

Selected User Stories:

1. US-4: Update a Task (3 points)
2. US-5: Delete a Task (2 points)
3. US-6: Assign Priority Levels (2 points) - if capacity allows

Additional Sprint 2 Tasks:

- Implement process improvements from Sprint 1 retrospective
- Add logging for all API operations
- Add health check endpoint (`/health`)
- Add error tracking/logging

Total Tentative Points: 5-7

Tech Stack

- **Runtime:** Node.js
- **Framework:** Express.js
- **Testing:** Jest + Supertest
- **CI/CD:** GitHub Actions
- **Version Control:** Git + GitHub

Project Structure (Planned)

```
task-manager-api/
├── src/
│   ├── index.js          # App entry point
│   ├── app.js            # Express app setup
│   ├── routes/
│   │   └── tasks.js      # Task routes
│   ├── controllers/
│   │   └── taskController.js
│   ├── models/
│   │   └── task.js       # Task data model
│   └── middleware/
│       └── errorHandler.js
├── tests/
│   └── tasks.test.js     # API tests
├── docs/
│   ├── SPRINT_0_PLANNING.md
│   ├── SPRINT_1_REVIEW.md
│   └── SPRINT_2_REVIEW.md
├── .github/
│   └── workflows/
│       └── ci.yml        # GitHub Actions config
├── package.json
├── .gitignore
└── README.md
```