Tracking Requirements

CS 346 Application
Development

We’re collecting a lot of information

From project planning (given), you can identify:

- **Project resources**: who is on the project?
- **Milestones**: deadlines we need to meet.

Project details need to be tracked. These are mostly provided for you in the project requirements.

From requirements, you should have:

- **Personas**: an abstract representation of a user of a system.
- **Scenarios**: a high-level description of how a system will be used.
- **User Stories**: brief, informal descriptions of software features told from the perspective of the end-user.

These are determined through discussion and information gathering. These also represent *work* that you need to track all-term.

We need to track what we’re doing

Why track this?

- We need a log of what has been done (good/bad).
- Helps us plan and coordinate work around the team.

User stories represent **features that need to work together** to be useful.

We want to track

- What we have done before
- What we are working on now
- What we need to do in the future

We will track progress against user stories.

- The best reflection of what “needs to be done” for our product to be useful.
- Focus on “how features work together to address problems”.
- Focus on “meaningful progress” vs “getting things done”.

Project Details

How to track high-level project goals and deadlines.

What is a milestone?

- A milestone is a goal that you want to achieve in your project.
- Your project will likely have many milestones!
  - **Internal**: getting a library complete so another team can use it.
  - **External**: getting a release ready to demo features to a user.
- For each milestone, you should track:
  - The **date** when you need to achieve it.
  - A **description** of the high-level goal that you need to achieve.
  - A list of **tasks**. These can be features from your requirements, or other tasks that need to be completed.
  - Some **acceptance criteria** i.e., a way to know that you have met the goal.

Example: customer demo

A typical external milestone would be a trade show demo for a customer. e.g.,

Milestone:

- Trade Show Demo

Deadline:

- The day before you travel to the conference.

Goals:

- The features that you intend to deliver to the customer are complete and tested.
- Your computer has the release installed and is setup with sample data.
- You have a glossy marketing brochure and price sheet that you can give them.

Acceptance:

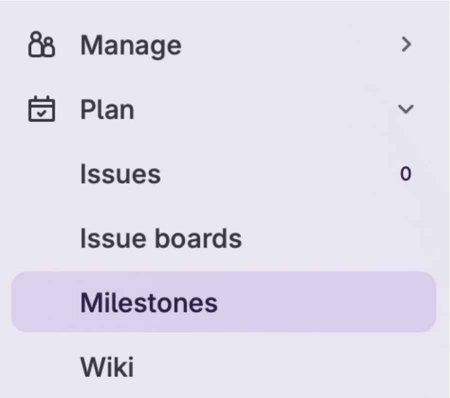
- Demonstrate the demo running on your notebook, with sample data, by the deadline.

Milestones in this course

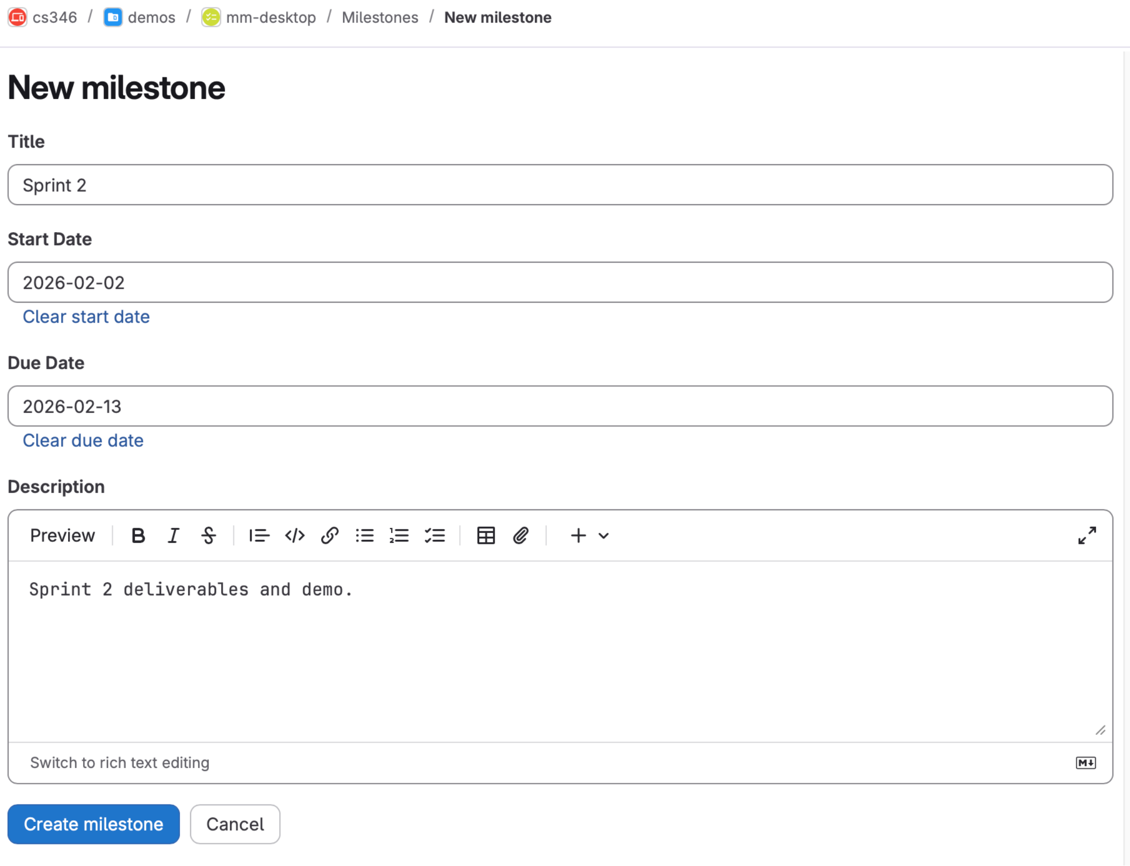
You have been given project deadlines aka milestones.

- **Project proposal**: describe your vision, requirements.
- **Sprint 1**: user interface (incomplete)
- **Sprint 2**: domain layer, user interface wired up.
- **Sprint 3**: data layer, can use “real data” from a database.
- **Sprint 4**: final release, all features complete.
- **Final submission**: documentation, video.

Each of these has a deadline, and a high-level goal.
We can track details in our GitLab project.



`Plan > Milestones` in your GitLab project.


cs346 / demos / mm-desktop / Milestones / New milestone

New milestone

Title

Sprint 2

Start Date

2026-02-02

Clear start date

Due Date

2026-02-13

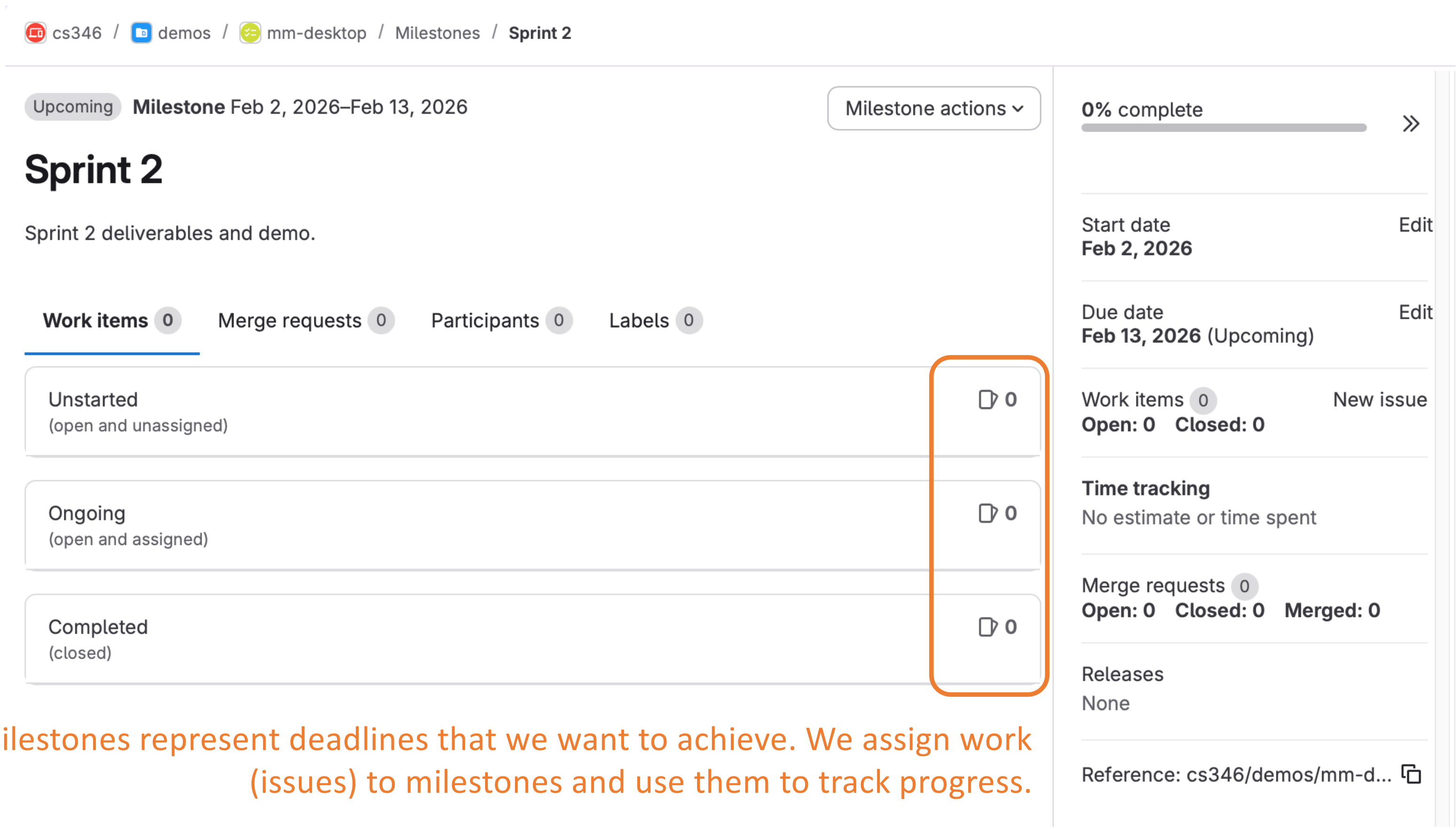
Clear due date

Description

Sprint 2 deliverables and demo.

Preview
Switch to rich text editing
Create milestone
Cancel

Repeat for every deadline in the project.



cs346 / demos / mm-desktop / Milestones / **Sprint 2**

Upcoming **Milestone** Feb 2, 2026–Feb 13, 2026

Sprint 2

Sprint 2 deliverables and demo.

Work items 0 | Merge requests 0 | Participants 0 | Labels 0

Unstarted (open and unassigned)	0
Ongoing (open and assigned)	0
Completed (closed)	0

0% complete

Start date — Edit
Feb 2, 2026

Due date — Edit
Feb 13, 2026 (Upcoming)

Work items 0 — New issue
Open: 0 **Closed: 0**

Time tracking
No estimate or time spent

Merge requests 0
Open: 0 **Closed: 0** **Merged: 0**

Releases
None

Reference: cs346/demos/mm-d...

Milestones represent deadlines that we want to achieve. We assign work (issues) to milestones and use them to track progress.

We ultimately want milestones for each of our deliverables, with work assigned to each one.

What to add first?

- Add **milestones** for each of your project deliverables.
- Date is the demo data from the course schedule.
- Goal is the goal described in the project milestone document.
  - e.g., `Team Project > Milestones > Proposal` on the course website.

- You can add any work that you anticipate to each milestone.
  - However, most of the tasks that you will assign will be taken from your requirements (proposal).
- Let’s talk more about adding `tasks` to our milestones.

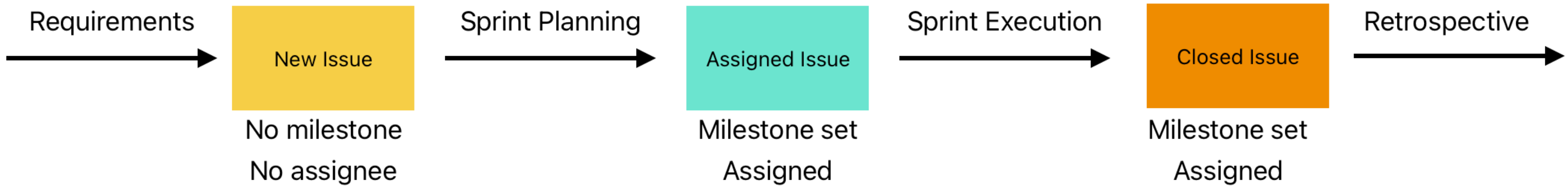
Tasks (aka Issues)

Work items that we assign to our milestones.

What is an issue?

- An `issue` is a unit of work.
  - Used to be called “bugs”; now we talk about “issues” as a more generic term.
  - An issue should represent about ½ a day work for one person.
  - Can be *any work* that needs to be done for a milestone.
- When do you create them?
  - As part of your requirements, prior to starting work. ★
  - When you identify something else that needs to be done, that you aren’t already tracking.
- Examples
  - Issue: “create new Android project.”
  - Issue: “setup Supabase tables to match our local schema.”
  - Issue: “implement new auth mechanism for the login user story.”

The Life of an Issue in GitLab



Requirements → New Issue → Sprint Planning → Assigned Issue → Sprint Execution → Closed Issue → Retrospective

New Issue: No milestone, No assignee

Assigned Issue: Milestone set, Assigned

Closed Issue: Milestone set, Assigned

A new issue should be created as part of the project proposal. It may not have much detail and should remain unassigned.

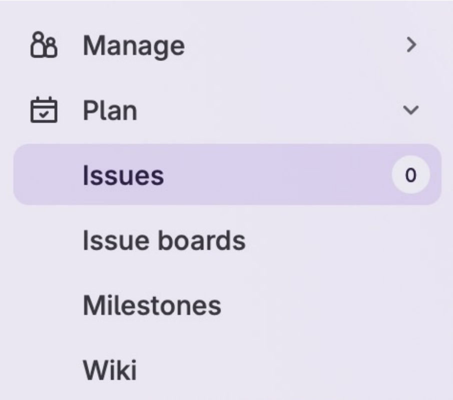
Unassigned issues represent the "product backlog".

During sprint planning, the team determines which issues they want to work on.

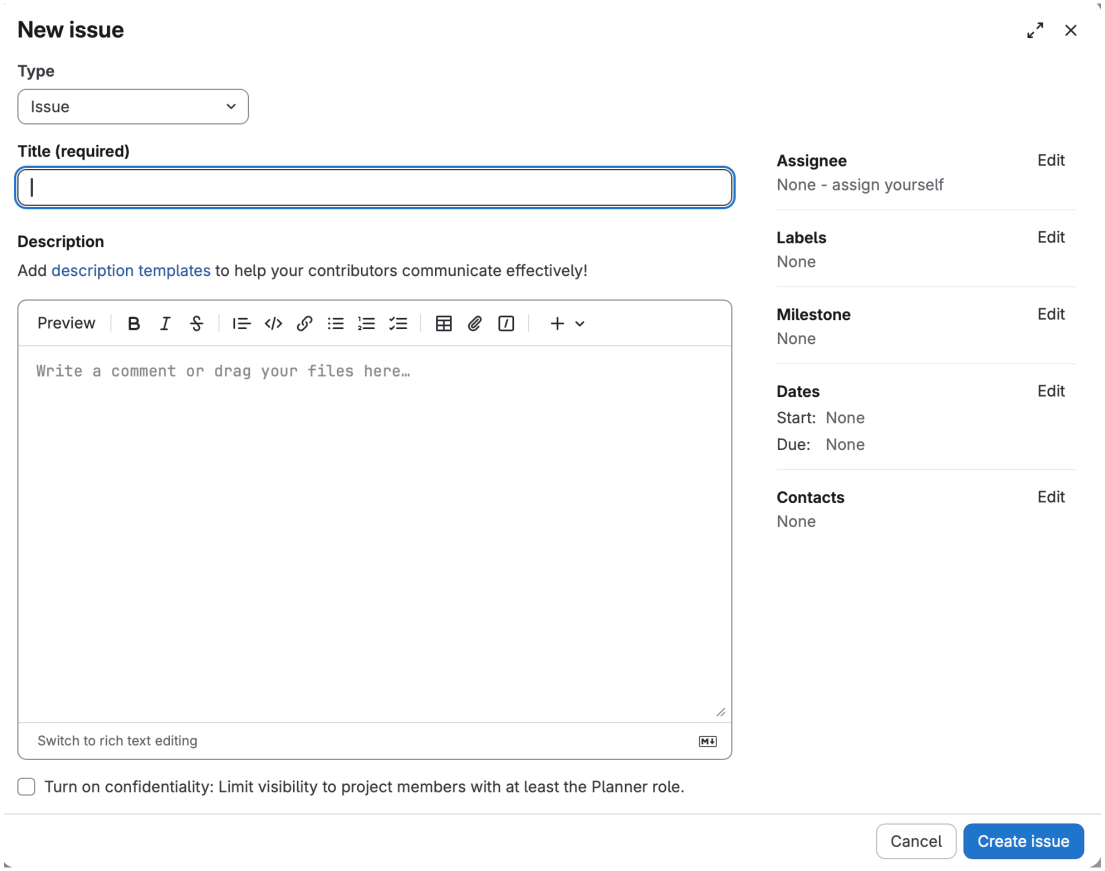
Some issues are moved to the milestone, and assigned to team members. Assigned issues represent the "sprint backlog" for that sprint.

During execution, team members add comments to the issue as they work on it. When complete, the issue can be marked as "closed" in GitLab.

Issues that aren't closed can be added to the next milestone, or returned to the backlog (unassigned).



Plan > Issues in GitLab.



New issue

Type

Issue

Title (required)

Description

Add description templates to help your contributors communicate effectively!

Write a comment or drag your files here…

Switch to rich text editing

□ Turn on confidentiality: Limit visibility to project members with at least the Planner role.

Assignee Edit
None - assign yourself

Labels Edit
None

Milestone Edit
None

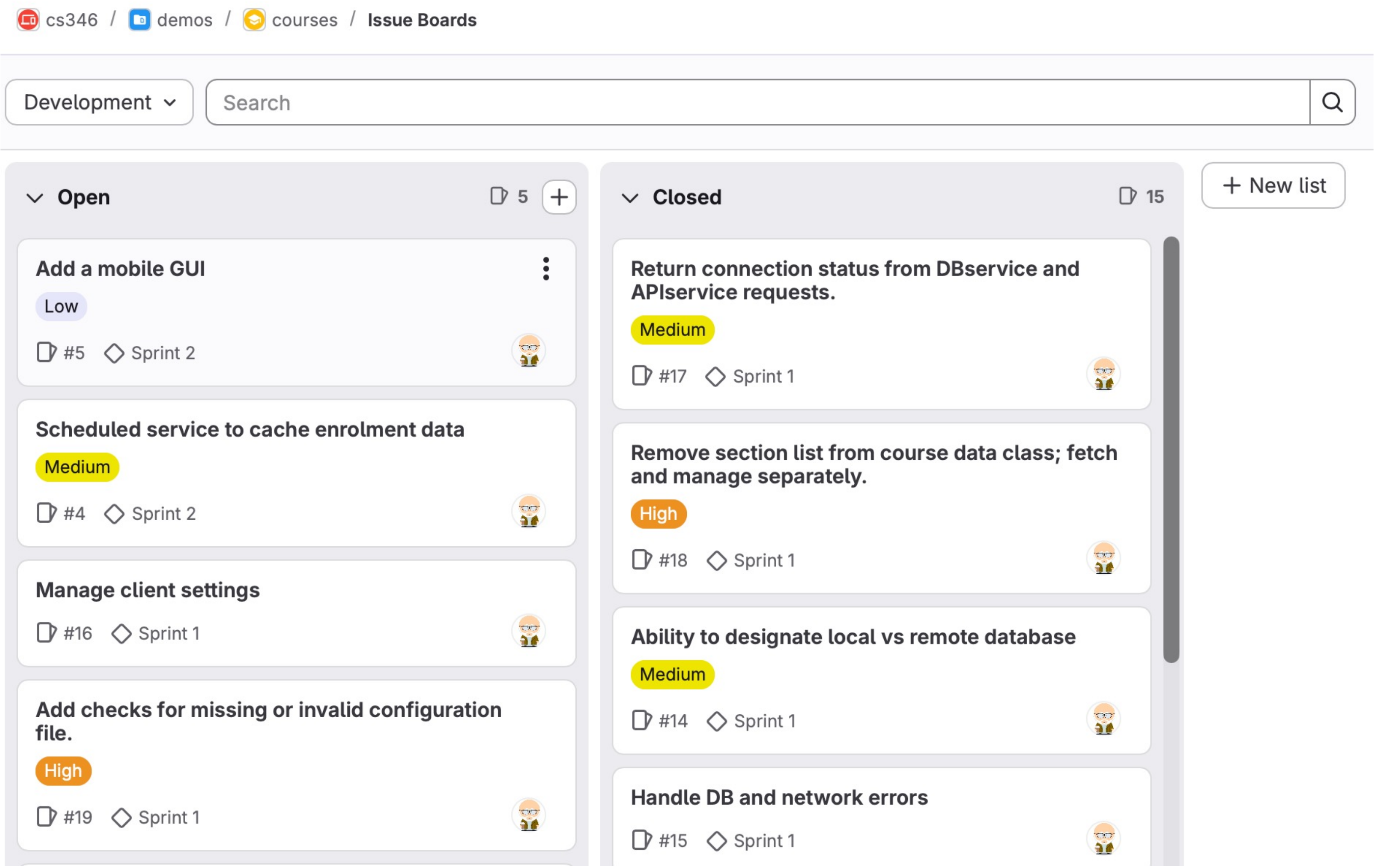
Dates Edit
Start: None
Due: None

Contacts Edit
None

tags
high/low

assign during
planning

optional



cs346 / demos / courses / **Issue Boards**

Open 5

Add a mobile GUI
Low
#5 Sprint 2

Scheduled service to cache enrolment data
Medium
#4 Sprint 2

Manage client settings
#16 Sprint 1

Add checks for missing or invalid configuration file.
High
#19 Sprint 1

Closed 15

Return connection status from DBservice and APIservice requests.
Medium
#17 Sprint 1

Remove section list from course data class; fetch and manage separately.
High
#18 Sprint 1

Ability to designate local vs remote database
Medium
#14 Sprint 1

Handle DB and network errors
#15 Sprint 1

Issue boards are useful for reviewing and assigning work.
You will use these in demos as well.

What do you put in issues?

How to organize your work

What you need to do:

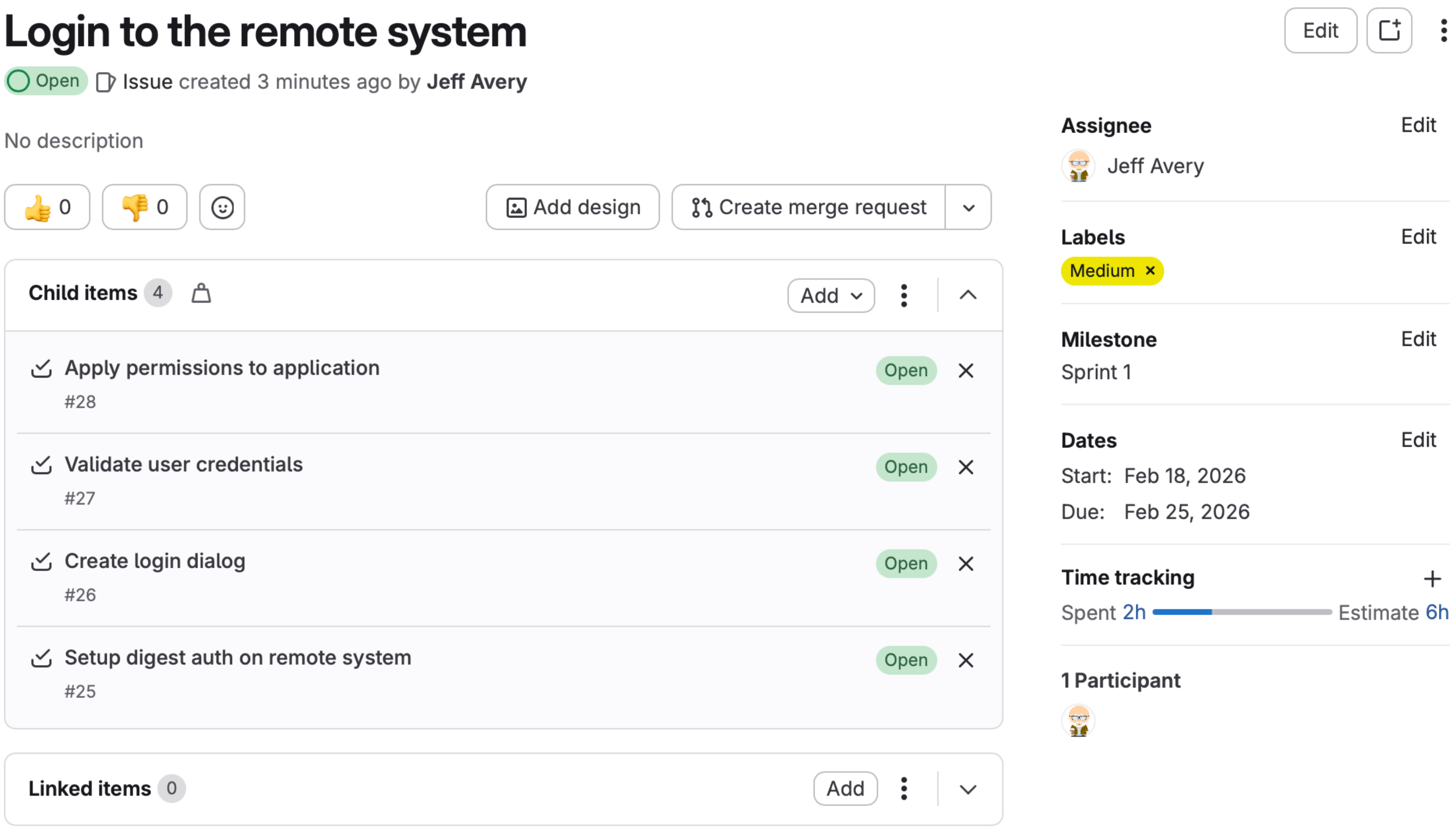
We typically track progress against **user stories**.

1. For each user story, create an issue in GitLab.
2. Within the issue that you've created, add sub-issues for each individual feature that you might deliver.
3. Add details to the feature.
4. Decide when you will start the user story and assign it to a sprint.

You do not need to complete an entire user story in one sprint!

- Assign work by sub-tasks.
- Multiple people can work on the same user story this way.

optional



Login to the remote system

Open · Issue created 3 minutes ago by **Jeff Avery**

No description

Child items 4

- Apply permissions to application #28 — Open
- Validate user credentials #27 — Open
- Create login dialog #26 — Open
- Setup digest auth on remote system #25 — Open

Linked items 0

Assignee
Jeff Avery

Labels
Medium ×

Milestone
Sprint 1

Dates
Start: Feb 18, 2026
Due: Feb 25, 2026

Time tracking
Spent 2h Estimate 6h

1 Participant

optional

A User Story with four sub-tasks that could be spread around the team.
The entire user story should be delivered as a working feature in Sprint 1.

When should you have this done?

- For your GitLab setup (Week 02):
  - Milestones created (proposal/sprints 1-4).
  - Priorities created (high/med/low).

- For your project proposal (Week 04)
  - A list of personas, user stories and requirements (in your wiki/proposal)
  - Issues created for all user stories (completed, in GitLab).
  - Sub-tasks for your first sprint (completed but can be unassigned in GitLab).

- For your first sprint (Week 06).
  - Sub-tasks assigned and mostly closed.