# volunteer\_scheduler Component Module (Decidim)

Create a new Decidim component by running decidim --component volunteer\_scheduler which scaffolds a gem (decidim-module-volunteer\_scheduler) with the correct file structure[[1]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=If%20you%20want%20to%20create,of%20an%20existing%20maintained%20plugin). The root namespace is Decidim::VolunteerScheduler. In **lib/decidim/volunteer\_scheduler/component.rb** register the component with:

Decidim.register\_component(:volunteer\_scheduler) do |component|  
 component.engine = Decidim::VolunteerScheduler::Engine  
 component.admin\_engine = Decidim::VolunteerScheduler::AdminEngine  
 component.permissions\_class\_name = "Decidim::VolunteerScheduler::Permissions"  
 component.settings(:global) do |settings|  
 settings.attribute :enable\_referrals, type: :boolean, default: true  
 settings.attribute :tokens\_name, type: :string, default: "SciCent tokens"  
 # Add other global settings as needed (XP rates, referral levels, etc.)  
 end  
 # hooks, exports/imports can be added here if needed  
end

This matches Decidim’s DSL: each component has a public engine and an isolated admin engine[[2]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=,own%20dependencies%20without%20having%20to)[[3]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=,component.admin_engine%20%3D%20MyComponent%3A%3AAdminEngine). The gemspec should name the module (e.g. decidim-module-volunteer\_scheduler) and declare dependencies on decidim-core. The engines go in lib/decidim/volunteer\_scheduler/engine.rb and .../admin\_engine.rb, with routing defined inside each. For example, the public engine (Decidim::VolunteerScheduler::Engine) might route /tasks and /assignments, while the admin engine (Decidim::VolunteerScheduler::AdminEngine) handles resource management (task\_templates, referrals, etc). Routes mimic other components (see meetings/budgets examples) and use RESTful Rails conventions[[4]](https://docs.decidim.org/en/develop/develop/classes/controllers.html#:~:text=Decidim%20controllers%20are%20plain%20Rails,have%20described%20in%20this%20section)[[5]](https://docs.decidim.org/en/develop/develop/classes/controllers.html#:~:text=,resources).

## Data Models and Scoping

Define your core models under app/models/decidim/volunteer\_scheduler/, each subclassing Decidim::VolunteerScheduler::ApplicationRecord[[6]](https://docs.decidim.org/en/develop/develop/classes/models.html#:~:text=). Every model should include Decidim::HasComponent (scoping it to the component) and any relevant concerns: for example, include Decidim::Followable so volunteers can follow tasks, Decidim::Authorable for ownership, Decidim::Traceable for change history, etc.[[6]](https://docs.decidim.org/en/develop/develop/classes/models.html#:~:text=). Key models include:

* **TaskTemplate**: defines volunteer tasks (e.g. name, description, XP reward, organization\_id). Belongs to the component and an organization (for multi-tenancy).
* **Task**: an instance of a task (could be generated from a template) with fields like title, description, due\_date, xp\_value, status, etc. It belongs\_to :task\_template and belongs\_to :organization. Include Decidim::Followable and Decidim::HasComponent.
* **Assignment**: joins a volunteer (Decidim user) to a Task, with fields user\_id, task\_id, status (applied, approved, completed), xp\_awarded, etc. Belongs to :task and :user. Include Decidim::Authorable if you treat the assigner as author, or simply use strong permissions.
* **Referral**: tracks referral chains. Typical fields: referrer\_id (user), referred\_id (user or email), level (integer 1–5), commission\_awarded (boolean/amount), created\_at. It may belong to the component or directly to an organization if referrals are org-scoped. Track which volunteer referred a new volunteer.
* **XpLedgerEntry**: records XP points earned or spent. Fields like user\_id, source\_type (e.g. "Task", "Referral"), source\_id, points, created\_at. Use this to propagate XP and ledger history.
* **TokenLedgerEntry**: tracks SciCent tokens (internal currency). Fields: user\_id, change (integer), reason (string), and ledger balances. (Since the question says “internal ledger only”, we keep it private; you might use Decidim::RecordEncryptor for sensitive fields if needed.)

Each model must belongs\_to :organization (or acts\_as\_tenant) to enforce multi-organization scoping[[7]](https://docs.decidim.org/en/v0.29/admin/system.html#:~:text=Every%20Decidim%20is%20multi,NGOs%2C%20cooperatives%2C%20etc). Decidim’s multi-tenancy means every resource is inherently linked to an Organization by default[[8]](https://docs.decidim.org/en/v0.29/admin/system.html#:~:text=Every%20Decidim%20is%20multi,NGOs%2C%20cooperatives%2C%20etc). By including Decidim::HasReference in models, you get unique identifiers prefixed per organization. No default scopes are imposed by Decidim[[9]](https://docs.decidim.org/en/develop/develop/classes/models.html#:~:text=Scopes), so enforce scoping in queries (e.g. current\_organization.tasks).

## Routing and Controllers

In **lib/decidim/volunteer\_scheduler/engine.rb**, define routes for the public-facing UI. For example:

Decidim::VolunteerScheduler::Engine.routes.draw do  
 resources :tasks, only: [:index, :show] do  
 resources :assignments, only: [:new, :create]  
 end  
 # additional public routes (e.g. volunteer dashboard)  
end

In **lib/decidim/volunteer\_scheduler/admin\_engine.rb** (the admin UI), route the management interfaces:

Decidim::VolunteerScheduler::AdminEngine.routes.draw do  
 resources :task\_templates  
 resources :referrals  
 resources :xp\_ledger\_entries  
 resources :assignments, only: [:index, :update, :destroy]  
 # etc.  
end

Controllers go under app/controllers/decidim/volunteer\_scheduler/ (public) and app/controllers/decidim/volunteer\_scheduler/admin/ (admin) following Decidim’s conventions[[4]](https://docs.decidim.org/en/develop/develop/classes/controllers.html#:~:text=Decidim%20controllers%20are%20plain%20Rails,have%20described%20in%20this%20section)[[5]](https://docs.decidim.org/en/develop/develop/classes/controllers.html#:~:text=,resources). For example:

module Decidim  
 module VolunteerScheduler  
 class TasksController < ApplicationController  
 include NeedsPermission  
 register\_permissions(Decidim::VolunteerScheduler::TasksController,  
 ::Decidim::VolunteerScheduler::Permissions,  
 ::Decidim::Permissions)  
 def permission\_scope  
 :public  
 end  
  
 def index  
 enforce\_permission\_to :read, :task  
 @tasks = paginate(current\_organization.tasks)  
 end  
 # show, new, create actions...  
 end  
 end  
end

Admin controllers (e.g. Admin::TaskTemplatesController) inherit from Decidim::VolunteerScheduler::Admin::ApplicationController and use enforce\_permission\_to for CRUD[[5]](https://docs.decidim.org/en/develop/develop/classes/controllers.html#:~:text=,resources). Use Decidim form objects (form(...)) and command pattern for create/update actions as in examples.

### UI and Assets

Use Decidim’s UI patterns (Cells for complex views, view hooks, etc.) to build dashboard and admin pages. For dynamic behavior (e.g. live task filtering), use Stimulus controllers placed under app/javascript/controllers and bundled via Webpacker. Register these in Webpacker configuration: add the component’s app/packs path in config/webpacker.yml and declare each JS/CSS entrypoint in config/webpack/custom.js[[10]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=Assets)[[11]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=1,folder%20to%20webpacker.yml). For example, add something like:

# config/webpacker.yml  
default: &default  
 # ...  
 resolved\_paths:  
 - /path/to/decidim-module-volunteer\_scheduler/app/packs

And in config/webpack/custom.js:

environment.config.merge({  
 entry: {  
 'decidim-volunteer-scheduler': path.join(\_\_dirname, '../../decidim-module-volunteer\_scheduler/app/packs/src/decidim/volunteer\_scheduler')  
 }  
});

This ensures Rails’ webpacker picks up your component’s assets[[12]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=1,folder%20to%20webpacker.yml). Place translations under config/locales/decidim\_volunteer\_scheduler.\*.yml following Decidim’s i18n patterns.

## Permissions and Authorization

Define app/permissions/decidim/volunteer\_scheduler/permissions.rb with a class inheriting Decidim::DefaultPermissions[[13]](https://docs.decidim.org/en/develop/develop/classes/permissions.html#:~:text=,scope%20%21%3D%20%3Apublic). In it, implement permissions to grant actions based on role, author, etc. For example:

module Decidim  
 module VolunteerScheduler  
 class Permissions < Decidim::DefaultPermissions  
 def permissions  
 return permission\_action unless user  
 if permission\_action.scope == :admin  
 # only organization admins can manage admin panel  
 permission\_action.allow! if user.admin\_of?(context[:component].organization)  
 elsif permission\_action.action == :read  
 permission\_action.allow!  
 elsif permission\_action.action.in?([:create, :update, :destroy]) && my\_object&.authored\_by?(user)  
 permission\_action.allow!  
 end  
 permission\_action  
 end  
  
 private  
 def my\_object  
 @my\_object ||= context.fetch(:task, nil) || context.fetch(:assignment, nil)  
 end  
 end  
 end  
end

Then register this class in the component manifest (component.permissions\_class\_name = ...)[[14]](https://docs.decidim.org/en/develop/develop/classes/permissions.html#:~:text=Now%20that%20you%20have%20defined,the%20permissions_class_name%20parameter%20as%20follows). In controllers, use enforce\_permission\_to :action, :resource (e.g. :read, :task) which will invoke your permission chain[[15]](https://docs.decidim.org/en/develop/develop/classes/permissions.html#:~:text=def%20index%20enforce_permission_to%20%3Aread%2C%20%3Aresource,for%20your%20business%20model%20end). In views, use allowed\_to? :update, :task for conditional UI.

## Integration with Invitations, Follow-Ups, and Notifications

**Invitations:** To invite volunteers by email (especially if they aren’t on the platform yet), leverage Decidim’s *private participant* invitations. When adding a new volunteer (email) to a private organizer or task, send an invitation email. Decidim’s private participant feature automatically emails a sign-up link to users who don’t exist yet[[16]](https://docs.decidim.org/en/develop/admin/spaces/processes/private_participants.html#:~:text=Invitation). For example, you might call the same mailer that Decidim uses for private-space invitations. This means volunteers follow the normal Decidim registration flow after being invited.

**Follow-Ups:** If volunteers or managers need to “follow” tasks or other entities, simply include the Decidim::Followable concern on those models[[6]](https://docs.decidim.org/en/develop/develop/classes/models.html#:~:text=). Then users clicking “Follow” in the UI will receive notifications of updates, using Decidim’s standard follow/notification system[[17]](https://docs.decidim.org/en/v0.29/admin/features/follows.html#:~:text=Most%20of%20the%20contents%20in,by%20the%20thing%20you%E2%80%99re%20following)[[6]](https://docs.decidim.org/en/develop/develop/classes/models.html#:~:text=).

**Notifications:** Use Decidim::Notifications to alert users of key events. For example, after a volunteer is approved or completes a task, create a notification for that user or for space admins. Many actions (assignment accepted, XP achieved) can reuse Decidim’s background notification jobs. Ensure each model includes Decidim::Traceable or Decidim::Authorable as needed so notifications link to an author or target. Decidim will then handle delivering emails/in-app alerts by default[[18]](https://docs.decidim.org/en/develop/admin/features/participant_actions/notifications.html#:~:text=Understand%20the%20basics). You can also add custom notification types in your jobs or commands when commissions or token changes occur.

## XP Leveling, Referral Rewards, and Token Ledger

Implement **XP leveling** by awarding XP to users in code (e.g. on task completion). Store each XP grant as a ledger entry (XpLedgerEntry), summing up per user for current level. You can mirror Decidim’s badge logic (levels at certain XP thresholds)[[19]](https://docs.decidim.org/en/develop/admin/features/social_features/badges.html#:~:text=Badges%20have%20multiple%20levels%2C%20depending,the%20fifth%20level%20at%20100). In fact, consider defining a custom badge or use Decidim’s achievement framework to represent XP levels.

For **referral rewards**, allow each volunteer to refer new users up to 5 levels deep. On a new registration via referral, trace up the chain of Referral records and award each upline a commission or XP bonus. This propagation can be done in a background job (use ActiveJob as Decidim supports: delayed\_job, Sidekiq, etc.[[20]](https://docs.decidim.org/en/develop/services/activejob.html#:~:text=ActiveJob)). For example, on creation of a Referral, enqueue a job that iterates n = 1..5 levels of referrer = current\_referral.referrer and creates ledger entries or badge awards for each, then stops at level 5 or if no referrer. Decidim’s jobs are agnostic, so configure your queue (e.g. Sidekiq) per standard guides[[20]](https://docs.decidim.org/en/develop/services/activejob.html#:~:text=ActiveJob).

Track **SciCent tokens** with an internal ledger model (TokenLedgerEntry) that records all token movements (earned/spent) per user. Because this is internal only, there’s no public blockchain or API needed – just treat it like virtual currency in your DB. Hook token awards to volunteer actions (e.g. completing high-value tasks might credit tokens) and log them. If tokens need to be configurable or visible, you can reuse Decidim’s share-token or badge systems to some extent, but since the requirement is “internal ledger only”, a custom model is simplest. Use a background job to aggregate or expire tokens if needed.

## Organization-Scoped Data (Multi-Tenancy)

Decidim is multi-tenant by default: each Organization (from the System panel[[7]](https://docs.decidim.org/en/v0.29/admin/system.html#:~:text=Every%20Decidim%20is%20multi,NGOs%2C%20cooperatives%2C%20etc)) has its own domain and data. Ensure your models include belongs\_to :organization (or use the acts\_as\_tenant-style concern) so records are tied to the current organization. Queries should always filter by current\_organization (provided by Decidim middleware). For example, in controllers use current\_organization.tasks so volunteers only see tasks in their organization. Scopes can also be used if your organization subdivides into thematic or territorial scopes[[21]](https://docs.decidim.org/en/develop/admin/settings/scopes.html#:~:text=Most%20of%20the%20spaces%20and,by%20scopes%20when%20exploring%20them). By sticking to Decidim’s multi-org model, no custom tenant logic is needed beyond linking organization\_id and using Decidim’s context.

## Testing

Follow Decidim’s engine testing conventions[[22]](https://docs.decidim.org/en/develop/develop/testing.html#:~:text=Edit%20this%20Page). Use RSpec for Ruby code: write specs under spec/models, spec/forms, spec/commands, spec/controllers, spec/queries, etc., with FactoryBot factories under spec/factories. For example, a spec/models/volunteer\_scheduler/task\_spec.rb, spec/system/volunteer\_scheduler\_tasks\_spec.rb, etc. Use Decidim’s dummy app (via bundle exec rake test\_app) to run your engine’s specs in isolation[[22]](https://docs.decidim.org/en/develop/develop/testing.html#:~:text=Edit%20this%20Page). Include system (Capybara) tests in spec/system to cover user flows (volunteer signup, task signup, referral). Factories should create valid models with minimal attributes (e.g. create a :task\_template, then build an :task with that template). For assets and JS, write Jest tests if needed, as Decidim uses Jest for its front-end.

Finally, leverage Decidim’s CI setup: ensure your .github/workflows includes your new module’s tests as per Decidim’s pattern[[23]](https://docs.decidim.org/en/develop/develop/testing.html#:~:text=We%20have%20configured%20the%20parallel_tests,need%20to%20follow%20these%20steps). This comprehensive structure (models, controllers, views, jobs, and tests) follows Decidim’s conventions[[4]](https://docs.decidim.org/en/develop/develop/classes/controllers.html#:~:text=Decidim%20controllers%20are%20plain%20Rails,have%20described%20in%20this%20section)[[24]](https://docs.decidim.org/en/develop/develop/classes/permissions.html#:~:text=Once%20you%20have%20defined%20those,your%20controllers%20the%20following%20statement) and will be ready for auto-generation or hand-coding.

**Sources:** Decidim developer documentation on components, permissions, models, assets, and testing[[2]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=,own%20dependencies%20without%20having%20to)[[14]](https://docs.decidim.org/en/develop/develop/classes/permissions.html#:~:text=Now%20that%20you%20have%20defined,the%20permissions_class_name%20parameter%20as%20follows)[[6]](https://docs.decidim.org/en/develop/develop/classes/models.html#:~:text=)[[10]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=Assets)[[20]](https://docs.decidim.org/en/develop/services/activejob.html#:~:text=ActiveJob).

[[1]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=If%20you%20want%20to%20create,of%20an%20existing%20maintained%20plugin) [[2]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=,own%20dependencies%20without%20having%20to) [[3]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=,component.admin_engine%20%3D%20MyComponent%3A%3AAdminEngine) [[10]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=Assets) [[11]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=1,folder%20to%20webpacker.yml) [[12]](https://docs.decidim.org/en/develop/develop/components.html#:~:text=1,folder%20to%20webpacker.yml) Components :: Decidim Docs

<https://docs.decidim.org/en/develop/develop/components.html>

[[4]](https://docs.decidim.org/en/develop/develop/classes/controllers.html#:~:text=Decidim%20controllers%20are%20plain%20Rails,have%20described%20in%20this%20section) [[5]](https://docs.decidim.org/en/develop/develop/classes/controllers.html#:~:text=,resources) Controllers :: Decidim Docs

<https://docs.decidim.org/en/develop/develop/classes/controllers.html>

[[6]](https://docs.decidim.org/en/develop/develop/classes/models.html#:~:text=) [[9]](https://docs.decidim.org/en/develop/develop/classes/models.html#:~:text=Scopes) Models :: Decidim Docs

<https://docs.decidim.org/en/develop/develop/classes/models.html>

[[7]](https://docs.decidim.org/en/v0.29/admin/system.html#:~:text=Every%20Decidim%20is%20multi,NGOs%2C%20cooperatives%2C%20etc) [[8]](https://docs.decidim.org/en/v0.29/admin/system.html#:~:text=Every%20Decidim%20is%20multi,NGOs%2C%20cooperatives%2C%20etc) System :: Decidim Docs

<https://docs.decidim.org/en/v0.29/admin/system.html>

[[13]](https://docs.decidim.org/en/develop/develop/classes/permissions.html#:~:text=,scope%20%21%3D%20%3Apublic) [[14]](https://docs.decidim.org/en/develop/develop/classes/permissions.html#:~:text=Now%20that%20you%20have%20defined,the%20permissions_class_name%20parameter%20as%20follows) [[15]](https://docs.decidim.org/en/develop/develop/classes/permissions.html#:~:text=def%20index%20enforce_permission_to%20%3Aread%2C%20%3Aresource,for%20your%20business%20model%20end) [[24]](https://docs.decidim.org/en/develop/develop/classes/permissions.html#:~:text=Once%20you%20have%20defined%20those,your%20controllers%20the%20following%20statement) Permissions :: Decidim Docs

<https://docs.decidim.org/en/develop/develop/classes/permissions.html>

[[16]](https://docs.decidim.org/en/develop/admin/spaces/processes/private_participants.html#:~:text=Invitation) Private participants :: Decidim Docs

<https://docs.decidim.org/en/develop/admin/spaces/processes/private_participants.html>

[[17]](https://docs.decidim.org/en/v0.29/admin/features/follows.html#:~:text=Most%20of%20the%20contents%20in,by%20the%20thing%20you%E2%80%99re%20following) Follows :: Decidim Docs

<https://docs.decidim.org/en/v0.29/admin/features/follows.html>

[[18]](https://docs.decidim.org/en/develop/admin/features/participant_actions/notifications.html#:~:text=Understand%20the%20basics) Notifications :: Decidim Docs

<https://docs.decidim.org/en/develop/admin/features/participant_actions/notifications.html>

[[19]](https://docs.decidim.org/en/develop/admin/features/social_features/badges.html#:~:text=Badges%20have%20multiple%20levels%2C%20depending,the%20fifth%20level%20at%20100) Badges :: Decidim Docs

<https://docs.decidim.org/en/develop/admin/features/social_features/badges.html>

[[20]](https://docs.decidim.org/en/develop/services/activejob.html#:~:text=ActiveJob) ActiveJob :: Decidim Docs

<https://docs.decidim.org/en/develop/services/activejob.html>

[[21]](https://docs.decidim.org/en/develop/admin/settings/scopes.html#:~:text=Most%20of%20the%20spaces%20and,by%20scopes%20when%20exploring%20them) Scopes :: Decidim Docs

<https://docs.decidim.org/en/develop/admin/settings/scopes.html>

[[22]](https://docs.decidim.org/en/develop/develop/testing.html#:~:text=Edit%20this%20Page) [[23]](https://docs.decidim.org/en/develop/develop/testing.html#:~:text=We%20have%20configured%20the%20parallel_tests,need%20to%20follow%20these%20steps) How to test Decidim engines :: Decidim Docs

<https://docs.decidim.org/en/develop/develop/testing.html>