Tamiti Studio Sprint 1 Documentation: Project & Task Management

🚀Sprint Focus

Implement a scalable, modular Project & Task Management system supporting:

- Projects with milestones, members, client context
- Tasks with tagging, dependencies, Kanban grouping
- Comments, progress tracking, role-based permissions
- Smart dashboards, upcoming deadlines, and streak-ready logic

---

Stack Highlights

- **Apps**: `projects`, `tasks`, `common`, `core`
- **Permissions**: Object-level via `core.permissions`
- **Filters**: `django-filter`
- **Pagination**: Reusable `common.pagination.DefaultPagination`
- **Enums**: Centralized enums in `common.enums`
- **Test Suite**: `pytest`, `factory_boy`
- **Admin**: Fully customized per model with filters, search, autocomplete

---

Models Implemented

`projects.models`

- `Project`: Name, status, priority, estimated hours, budget, tags
- `ProjectMember`: User + role in a project (unique constraint)
- `Milestone`: Tied to project, goal-based, with reward tracking
- `ProjectComment`: Internal/client-facing comments, timestamped

`tasks.models`

- `Task`: Linked to project, milestone, assigned user/staff, due date
- `TaskGroup`: Grouping for Kanban layout with order field
- `Task.dependencies`: Self-referencing M2M for chaining

`common.enums`

- Enum classes: `ProjectStatus`, `TaskStatus`, `PriorityLevel`, `ProjectRole`, `OriginApp`

`common.pagination`

- `DefaultPagination`: Reusable, consistent pagination control across views

🏫Views & APIs

`projects/views.py`

- `ProjectListCreateView`: Supports filtering, pagination, creation
- `ProjectDetailView`: RUD with project access restrictions
- `MilestoneListCreateView`, `MilestoneDetailView`
- `ProjectCommentListCreateView`: POST & GET by project
- `project_stats`: Returns total/active/completed project stats
- `upcoming_deadlines`: Tasks & milestones due in next 7 days

`tasks/views.py`

- `TaskListCreateView`, `TaskDetailView`
- `toggle_task_completion`: Toggles and optionally updates streak stats

Filtering & Pagination

`common/pagination.py`

```
class DefaultPagination(PageNumberPagination):
    page_size = 10
    page_size_query_param = 'page_size'
```

`projects/filters.py`, `tasks/filters.py`

- Filter by `status`, `priority`, `name`, `due_date` via `django_filters`

🚚Permissions (core/permissions.py)

```
class IsProjectMember(BasePermission):
    def has_object_permission(self, request, view, obj):
        project = getattr(obj, 'project', None)
        return ProjectMember.objects.filter(project=project,
user=request.user).exists()
```

Test Suite

`tests/`

- `factories.py`: Reusable factories for all models
- `test_models.py`: Unit tests for core logic:
- Project creation
- Task toggling
- Milestone behavior
- Completion % logic

Example:

```
@pytest.mark.django_db
def test_project_completion_percentage():
    project = ProjectFactory()
    TaskFactory.create_batch(3, project=project, is_completed=False)
    TaskFactory.create_batch(2, project=project, is_completed=True)
    project.update_completion_percentage()
    assert project.completion_percentage == 40
```

Admin Customizations

- List filters, search, autocomplete, readonly fields
- Autocomplete for FKs: `project`, `user`, `milestone`, `assigned_to`
- Custom `short_content()` preview for comments

Outcome of Sprint 1

- Scalable task/project model structure
- Ready-to-integrate comment system, dashboard, and notifications
- Fully DRY factories and tests
- Reusable permissions, enums, and pagination in place
- Realistic API endpoints aligned with benchmark design
- `common` app established to hold enums, pagination, shared utilities

Next: Sprint 2 → Finance Manager (requisitions, approvals, income/expense)