# pussla

Version 0.1

# Table of Contents

Contents:

# pussla documentation

Add your content using `reStructuredText` syntax. See the reStructuredText documentation for details.

Welcome to **Pussla** – a “Planning-as-Code” system. We built this because humans aren’t rows in an Excel sheet, and resource planning shouldn’t be a nightmare. With this you can manage 100+ colleagues using the same tools you use to build software: **Git, Markdown, and YAML.**

🇸🇪 The “Pussla” Philosophy

1. **Planning-as-Code:** If it’s not in Git, it didn’t happen.

2. **Context Matters:** Numbers tell us *what*, but Markdown tells us *why*.

3. **Privacy by Design:** We protect our people’s data while staying AI-ready.

4. **Lagom is Best:** Not too little data, not too much—just enough to keep the projects moving.

# Requirements

This page is the source of truth for formal requirements (Sphinx-needs).

Requirement: **Git-based planning** REQ_PUSSLA_001

status: open
tags: core, workflow

The system shall use Git for all resource planning to ensure traceability, version control, and pull-request-based review of changes.

Requirement: **Two-layer architecture** REQ_PUSSLA_002

status: open
tags: security, architecture

The system must separate PII into an Identity Layer and anonymized data into an Allocation Layer to follow Privacy by Design.

Requirement: **AI-safe allocation** REQ_PUSSLA_003

status: open
tags: ai, privacy

The Allocation Layer may only use anonymous aliases. This enables third-party AI analysis without exposing personal names.

Requirement: **Workload validation** REQ_PUSSLA_004

status: open
tags: validation
links incoming: TEST_PUSSLA_004

The system shall use a linter to warn when an individual's total allocation exceeds 100% for any time period. Over-allocation warnings shall not block saving allocations.

Requirement: **PII leak protection** REQ_PUSSLA_005

status: open
tags: security, ci-cd

The CI/CD pipeline must automatically verify that no names or personal data are written in public allocation YAML files.

Requirement: **Context via Markdown** REQ_PUSSLA_006

status: open
tags: ux

Each resource shall have an associated Markdown file where qualitative information about goals and development can be documented, not only numeric data.

Requirement: **Project metadata in frontmatter** REQ_PUSSLA_008

status: open
tags: architecture, data-model

Each project shall have its own Markdown file with YAML frontmatter (for example: owner_alias, start_week, end_week, status) and a free-form Markdown body for scope and risks.

Requirement: **Shareable AI folder without identity data** REQ_PUSSLA_009

status: open
tags: ai, privacy, structure

Test and demo data shall be shareable via a single folder, *tst-data/planning/*, containing *allocations/* and *projects/*, while *tst-data/identity/* remains separate. Legacy path *tst-data/planing/* may be supported for backward compatibility.

Requirement: **Dashboard visualization** REQ_PUSSLA_007

status: open
tags: frontend
links incoming: TEST_PUSSLA_005

The system shall be able to generate a visual heatmap and quarterly forecasts based on the underlying text files.

Requirement: **Possible assignment and leads** REQ_PUSSLA_010

status: open
tags: structure

It shall be possible to track future leads and possible assignments for a person in order to track what projects he/she might be going into. These shall be clearly separately identifiable from decided assignments and be possible to filter out.

Requirement: **Role and skill based staffing lookup** REQ_PUSSLA_011

status: open
tags: staffing, data-model, matching

The system shall track each person's role and skills, and it shall be possible to identify available people for a project based on required role and skill.

Requirement: **Dashboard filter for past periods** REQ_PUSSLA_012

status: open
tags: dashboard, ux, filtering

The dashboard shall allow users to hide or show past weeks and past months, so users can focus on the current week and future periods when planning.

Requirement: **Dashboard grouped calendar headers** REQ_PUSSLA_013

status: implemented
tags: dashboard, ux, frontend
links incoming: TEST_PUSSLA_002

The dashboard shall show grouped headers with Year at the top, Month (without year) in the middle, and Week at the bottom. Month headers shall also show allocation percentage for each month group.

Requirement: **Dashboard weekly footer summary** REQ_PUSSLA_014

status: implemented
tags: dashboard, frontend, reporting
links incoming: TEST_PUSSLA_002

The dashboard shall show a footer row with allocation percentage per week column. Footer values shall update when visible users/weeks change due to filters.

Requirement: **Dashboard week label format** REQ_PUSSLA_015

status: implemented
tags: dashboard, frontend, formatting
links incoming: TEST_PUSSLA_001

The dashboard shall display week labels in *Www* format (for example *W01*) while keeping canonical storage/API keys in *YYYY-Www* format.

Requirement: **Dashboard editable weekly allocations** REQ_PUSSLA_016

status: implemented
tags: dashboard, editing, api, data-write

links incoming: TEST_PUSSLA_003

The dashboard shall allow editing allocations for a selected person-week with support for multiple projects in the same week. Saving shall require explicit user action in the GUI (Save), support cancel/discard, and persist changes to allocation YAML files. Overbooking (>100%) is allowed and shall not block save.

# Requirement Status Matrix

Requirement status overview

| ID | Title | Status | Tags |
|---|---|---|---|
| REQ_PUSSLA_001 | Git-based planning | open | core; workflow |
| REQ_PUSSLA_002 | Two-layer architecture | open | security; architecture |
| REQ_PUSSLA_003 | AI-safe allocation | open | ai; privacy |
| REQ_PUSSLA_004 | Workload validation | open | validation |
| REQ_PUSSLA_005 | PII leak protection | open | security; ci-cd |
| REQ_PUSSLA_006 | Context via Markdown | open | ux |
| REQ_PUSSLA_007 | Dashboard visualization | open | frontend |
| REQ_PUSSLA_008 | Project metadata in frontmatter | open | architecture; data-model |
| REQ_PUSSLA_009 | Shareable AI folder without identity data | open | ai; privacy; structure |

| ID | Title | Status | Tags |
|---|---|---|---|
| REQ_PUSSLA_010 | Possible assignment and leads | open | structure |
| REQ_PUSSLA_011 | Role and skill based staffing lookup | open | staffing; data-model; matching |
| REQ_PUSSLA_012 | Dashboard filter for past periods | open | dashboard; ux; filtering |
| REQ_PUSSLA_013 | Dashboard grouped calendar headers | implemented | dashboard; ux; frontend |
| REQ_PUSSLA_014 | Dashboard weekly footer summary | implemented | dashboard; frontend; reporting |
| REQ_PUSSLA_015 | Dashboard week label format | implemented | dashboard; frontend; formatting |
| REQ_PUSSLA_016 | Dashboard editable weekly allocations | implemented | dashboard; editing; api; data-write |

# Requirement-Test Gap Analysis

The tables below assume tests link to requirements via `:links: REQ_...`.

No needs passed the filters

No needs passed the filters

# Test Coverage

This page maps unit test cases to requirements using Sphinx-needs `test` items.

Test Case: **Dashboard week label formatter (Www)** TEST_PUSSLA_001

status: passed
tags: dashboard, frontend, formatting, unit
links outgoing: REQ_PUSSLA_015

Covered by `tests/test_week_format.js`. Verifies canonical ISO week keys are rendered as `Www` labels.

Test Case: **Dashboard calendar header grouping and month percentages** TEST_PUSSLA_002

status: passed
tags: dashboard, frontend, unit
links outgoing: REQ_PUSSLA_013, REQ_PUSSLA_014

Covered by `tests/test_header_groups.js`. Verifies year/month grouping and percentage aggregation logic used by headers/footer.

Test Case: **Dashboard editable weekly allocations writeback** TEST_PUSSLA_003

status: passed
tags: dashboard, editing, api, unit
links outgoing: REQ_PUSSLA_016

Covered by `tests/test_dashboard_editing.py`. Verifies multi-project week updates, YAML persistence behavior, and payload validation.

Test Case: **Validation helpers for planning data format** TEST_PUSSLA_004

status: passed
tags: validation, unit
links outgoing: REQ_PUSSLA_004

Covered by `tests/test_validation.py`. Verifies validation regex behavior for ISO week format and selected PII detection patterns.

Test Case: **Planning aggregation helpers** TEST_PUSSLA_005

status: passed
tags: aggregation, unit
links outgoing: REQ_PUSSLA_007

Covered by `tests/test_aggregation.py`. Verifies helper behavior used for planning aggregation outputs.

# Test-to-Requirement Matrix

Tests and linked requirements

| ID | Title | Status | Links |
|---|---|---|---|
| TEST_PUSSLA_001 | Dashboard week label formatter (Www) | passed | REQ_PUSSLA_015 |
| TEST_PUSSLA_002 | Dashboard calendar header grouping and month percentages | passed | REQ_PUSSLA_013; REQ_PUSSLA_014 |
| TEST_PUSSLA_003 | Dashboard editable weekly allocations writeback | passed | REQ_PUSSLA_016 |
| TEST_PUSSLA_004 | Validation helpers for planning data format | passed | REQ_PUSSLA_004 |
| TEST_PUSSLA_005 | Planning aggregation helpers | passed | REQ_PUSSLA_007 |

Created using Sphinx 8.2.3.