Web app

Getting Started

Tech Stack

- **Language:** Python 3.14 (Managed via Nix)
- **Framework:** Django
- **Package Manager:** uv (Fastest Python resolver/installer)
- **Environment:** Nix (with the `std` library)
- **Command Runner:** `just` (alternative to `make`)

Setting Up the Environment

> FIXME: this is wrong

You do not need to install Python or Django manually. You only need **Nix**.

Step 1: Install Nix

If you don't have it, install Nix and enable “experimental features” (Flakes and Nix Command).

Consider doing the latter like that:
1) `sudo mkdir /root/.config/nix`

2) `sudoedit /root/.config/nix/nix.conf`

3) In there, add this line `experimental-features = nix-command flakes`

Step 2: Enter the Development Shell

Navigate to the project root and run:

```
nix develop
```

What happens when you run this? Nix reads `shells.nix` and `packages.nix` to:

1. Download and provide **Python 3.14**.
2. Install system tools like `curl`, `git`, `uv`, and `just`.

3. Set up your `PATH` to include the project's virtual environment (`.venv/bin`).
4. Activate a custom shell prompt (the "embed console").

Step 3: The "Just" Command Runner

Once inside the Nix shell, we use a tool called `just` to run common tasks.

Command	Action
`just`	Lists all available commands.
`just init-db`	**Run this first.** It migrates the DB and asks you to create a Superuser.
`just dev`	Starts the Django development server.
`just make-migrations`	Generates new DB migration files after model changes.
`just fmt`	Automatically formats all code using `treefmt`.
`just test`	Runs the Django test suite.

Step 4: Running the dev server

Consider using these commands after all previous steps:

`nix init-db` creates and initializes the DB, prompting for superuser credentials.
`nix dev` starts a local dev server.
Congratulations! You are ready to develop.

Structure of the project

Architecture & Frontend Patterns (Midas)

The RFID Tracker (codename **Midas**) follows a traditional Django structure but uses a “Single File Component” philosophy for its views.

Project Organization

The project is split into the core configuration and the functional app logic.

Directory / File	Role
`midas/`	The active module containing the latest logic, models, and views.
`webui/`	Legacy code kept for bookkeeping; **do not use** for new features.
`templates/midas/`	The “Face” of the app. Contains HTML layouts, CSS, and JS.
`models.py`	Database schema (RFID tags, teams, hours).
`views.py`	The “Controller.” Fetches data from DB and performs calculations.
`urls.py`	Maps URLs to specific Python functions in `views.py`.

The Frontend Pattern

Layout Strategy

Every page follows this block-based hierarchy:

1. `{% extends 'midas/base.html' %}`: Inherits the sidebar, navbar, and global styles.
2. `{% block head %}`: Contains page-specific CSS and external libraries (like Chart.js).
3. `{% block main %}`: The actual HTML content (cards, forms, tables).

4. `<script>`: Local logic for charts or UI interactions.

The “Data Bridge” (Python to JavaScript)

To keep the code clean and secure, we use a specific pattern to pass data from the Python backend to the JavaScript frontend.

The `json_script` Filter

Instead of messy string concatenation, we use Django's `json_script` tag. This safely injects Python dictionaries into the HTML as a JSON object that JS can read.

In the HTML Template:
`{{ page_data|json_script:"frontend-data" }}`

In the JavaScript block:
`const appData = JSON.parse(document.getElementById('frontend-data').textContent);`
`// Now appData.chart.labels is ready for Chart.js!`

Backend Logic & Calculation

The heavy lifting is done in the Python files before the page even loads:

- **Calculations:** Logic like “Total Hours” or “Avg per Member” is calculated in `views.py` or helper files like `statistics.py`.
- **Database Interaction:** Uses Django’s ORM (Object-Relational Mapper) to query the `db.sqlite3` (dev) or the production DB.
- **Forms:** Django’s `forms.py` handles input validation when you change dates or teams in the dashboard.

Admin dashboard

Django provides its own built-in admin dashboard which can be reached through “Admin Dashboard” button on the sidebar. (if you are a superuser).