# Picture-by-Picture Functional Specification Document (FSD)

**Project:** PythonDemo (Django MVP)

**Repository:** https://github.com/mominijaz2030/PythonDemo

**Purpose:** This FSD breaks the application into a sequence of “pictures” (screens / interactions). Each picture contains: UI sketch placeholder, user goal, inputs, outputs, functional requirements, data interactions, and acceptance criteria. Use this document to implement UI, write test cases, or create design assets.

## Table of Contents

1. Overview & Scope
2. Actors & Roles
3. Data Model (summary)
4. Picture 1 — Landing / Home (App Shell)
5. Picture 2 — User Registration / Login
6. Picture 3 — To‑Do List Dashboard
7. Picture 4 — Create / Edit Task Modal
8. Picture 5 — Task Filters & Bulk Actions
9. Picture 6 — User Profile / Settings
10. Picture 7 — Static Pages / About / Objectives
11. Non‑functional Requirements
12. API / URL Endpoints
13. Acceptance Criteria & Test Cases

## 1. Overview & Scope

This repo is an MVP containing Django apps: todolistapp, usersapp, templates, and static assets. The scope of this FSD is the core flows for a simple authenticated to‑do application with user management: registration/login, CRUD for tasks, task listing, filtering, and basic settings.

## 2. Actors & Roles

* **Anonymous Visitor** — can view landing/about and register/login.
* **Authenticated User** — can create, update, delete, mark done/undone, filter tasks, manage profile.
* **Admin** — Django admin privileges for user/task management (out of scope for UI sketches but supported by Django admin).

## 3. Data Model (summary)

Based on repo structure (todolistapp): - TaskList model - task: CharField(max\_length=500) - done: BooleanField(default=False) - likely created\_at, updated\_at, user (assumed/typical) — if not present, add ForeignKey(User, on\_delete=...) to scope tasks to users. - User — Django built-in auth.User or custom user in usersapp.

## Picture 1 — Landing / Home (App Shell)

**Visual (placeholder):** ![Landing page](static/images/landing.png)

**File(s) in repo:** templates/ (look for index.html, base.html)

**User goal:** See app intro, navigation links (Login, Register, To‑Do), quick CTA to sign up.

**Inputs:** Clicks on navigation, CTA.

**Outputs:** Redirect to login/register or dashboard (if logged in).

**Functional Requirements:** - Display app name and short description. - Show navigation bar linking to: Home, Objectives, To‑Do (dashboard), Login, Register. - If user is authenticated, show username + Logout instead of Login/Register.

**Data interactions:** None (read only).

**Acceptance Criteria:** - Page loads within 2s. - Nav links route to correct URLs. - Authenticated state toggles nav as described.

## Picture 2 — User Registration / Login

**Visual (placeholder):** ![Auth pages](static/images/auth.png)

**File(s):** usersapp/ (look for views and templates for registration/login)

**User goal:** Create account or sign in to manage tasks.

**Inputs:** email/username, password, confirm password (register); username/email and password (login).

**Outputs:** On success, redirect to To‑Do Dashboard. On failure, show inline error messages.

**Functional Requirements:** - Registration validates required fields and password confirmation. - Login authenticates and sets session. - CSRF protection enabled. - Passwords stored via Django’s secure hashing.

**Acceptance Criteria:** - Registering with existing username/email fails with clear message. - Correct credentials log the user in and redirect to dashboard.

## Picture 3 — To‑Do List Dashboard (Main Screen)

**Visual (placeholder):** ![Dashboard](static/images/dashboard.png)

**File(s):** todolistapp/templates/ (dashboard template)

**User goal:** View list of tasks, mark done/undone, navigate to create/edit.

**Inputs:** Click on task to toggle done, click Add Task, click Edit, click Delete.

**Outputs:** Task list refreshes to reflect changes; toast/alert messages for actions.

**Functional Requirements:** - Show list of tasks for the logged in user, ordered by most recent or incomplete first. - Each task shows: text, done indicator, edit button, delete button, created date. - Provide Add Task input at the top (inline form) or Add Task button that opens modal. - Success and error messages displayed via Django messages framework.

**Data interactions:** - Read: GET tasks filtered by request.user. - Write: POST create task; POST/PUT update; POST delete.

**Acceptance Criteria / Example Flows:** - Adding a task with empty text returns validation error. - Clicking complete toggles done state and updates UI. - Deleting a task removes it after user confirmation.

## Picture 4 — Create / Edit Task Modal

**Visual (placeholder):** ![Create Edit Task](static/images/task\_modal.png)

**User goal:** Add or modify task details quickly.

**Inputs:** task text, optional due date/priority (if added).

**Outputs:** On success, modal closes and dashboard updates; on validation error, show inline.

**Functional Requirements:** - Modal contains textarea/input for task and Save/Cancel buttons. - Editing pre-fills the current task text. - Support AJAX submission (optional) for smoother UX.

**Acceptance Criteria:** - Modal shows proper validation messages. - After save, task is visible in list with new values.

## Picture 5 — Task Filters & Bulk Actions

**Visual (placeholder):** ![Filters](static/images/filters.png)

**User goal:** Filter tasks (All / Active / Completed), search, bulk-delete or bulk-mark-done.

**Inputs:** filter buttons, search input, checkbox multi-select + bulk action dropdown.

**Outputs:** Filtered list, success messages on bulk actions.

**Functional Requirements:** - Provide three filter states and a search input that filters by substring. - Allow selecting multiple tasks with checkboxes and performing bulk actions (Mark Done, Delete). - Confirm destructive bulk actions.

**Acceptance Criteria:** - Filters apply on client or server and show accurate counts. - Bulk actions affect only selected tasks and show summary of how many affected.

## Picture 6 — User Profile / Settings

**Visual (placeholder):** ![Profile](static/images/profile.png)

**User goal:** Update profile (display name, email) and change password.

**Inputs:** form fields for profile updates; current password + new password for change.

**Outputs:** Confirmation messages, reauthentication if required.

**Functional Requirements:** - Profile update endpoints validate email uniqueness and proper formats. - Password change requires current password.

**Acceptance Criteria:** - Successful profile changes reflect in navbar and user account. - Password change requires correct current password.

## Picture 7 — Static Pages / Objectives

**Visual (placeholder):** ![Objectives page](static/images/objectives.png)

**File(s):** Objectives/ folder likely contains static content or docs.

**User goal:** Read about project purpose, goals, and resources.

**Functional Requirements:** - Render Objectives content as HTML pages accessible from navbar. - Serve static images from static/images/.

**Acceptance Criteria:** - Pages load and present content from the Objectives folder.

## Non‑functional Requirements

* Security: Use Django CSRF, authenticate views, protect user data.
* Performance: Page responses under 300ms for normal loads (local dev not required, but keep templates efficient).
* Accessibility: Forms labeled, focus management for modals, sufficient color contrast.
* Internationalization: Use Django translation hooks if needed.

## API / URL Endpoints (Suggested mapping)

* GET / — Landing page
* GET /objectives/ — Objectives documentation
* GET /login/, POST /login/
* GET /register/, POST /register/
* GET /tasks/ — Dashboard (requires auth)
* POST /tasks/add/ — Create task
* POST /tasks/<id>/toggle/ — Toggle done
* POST /tasks/<id>/edit/ — Update task
* POST /tasks/<id>/delete/ — Delete task
* POST /tasks/bulk/ — Bulk actions
* GET/POST /profile/ — Profile settings

## Acceptance Criteria & Sample Test Cases

**Add Task:** - Given an authenticated user, when they submit a non-empty task, then task appears in list and DB.

**Toggle Done:** - Given a task, when user toggles done, then DB done flips and UI updates.

**Auth:** - Login with invalid credentials shows an error and prevents access to /tasks/.

**Bulk Delete:** - Selecting 3 tasks and pressing delete shows confirmation and removes 3 tasks.

## Implementation Notes & Next Steps

1. Verify TaskList model for user relation; add if missing to scope tasks per-user.
2. Add unit tests for task CRUD and auth flows.
3. Create simple wireframe images (placeholders in static/images/) or replace placeholders with real screenshots.
4. Consider adding AJAX endpoints for better UX (optional).

*End of FSD (picture-by-picture).*

*If you want, I can now:* - generate wireframe PNGs for each picture and embed them into this doc, or - convert this FSD into a PDF or printable Word doc, or - expand any picture into a low/hi‑fi mockup with exact field names and HTML snippets.