StickyNotes Application — Software Documentation (Part 2)

Project: StickyNotes Application (Part 2)
Theme: Modern minimal look
Migration target: MariaDB

1. Introduction & Purpose

The StickyNotes app allows authenticated users to create, edit, and delete short notes.
Notes are stored in a MariaDB database. This document summarises requirements, design, UML, schema, and testing notes for the application.

2. System Requirements

- Python 3.10+
- Django 4.2+
- MariaDB server (10.4+ recommended)
- mysqlclient Python package
- Basic web browser for UI

3. Functional Requirements

- FR1: Users can create a sticky note with title, content, and color.
- FR2: Users can edit their own notes.
- FR3: Users can delete their own notes.
- FR4: Notes display in a responsive grid sorted by last update time.
- FR5: Admin can moderate notes via Django admin panel.

4. Non-Functional Requirements

- NFR1: App should be responsive and load quickly (<2s on simple infra).
- NFR2: Data persisted in MariaDB with ACID guarantees.
- NFR3: Authentication enforced for create/edit/delete actions.

5. Design Overview (MVC)

- Model: StickyNote (author, title, content, color, timestamps)
- View: Django views for listing, creating, editing, deleting
- Controller: Django routing and forms
- Templates: presentational layer using a modern minimal CSS theme.

6. CRUD Flows

- Create: POST to /note/create/ with form data; server saves note with request.user
- Read: GET / lists notes; details shown inline
- Update: POST to /note/<pk>/edit/ to update fields
- Delete: POST to /note/<pk>/delete/ to remove

7. Database Schema (MariaDB)

- Table: stickynotes_app_stickynote
  - id (PK), author_id (FK -> auth_user), title (varchar 120), content (text), color (varchar 20), created_at (datetime), updated_at (datetime)

8. UML (high-level)

- Use Case: Authenticate -> Create/Edit/Delete/View Notes
- Class: StickyNote (fields as above)
- Sequence: User -> Form -> Server -> DB -> Response

9. Testing & Results

- Unit tests: model creation, form validation, view permission checks (not included in skeleton)
- Manual tests performed: note creation, edit, delete as sample user
- Migration: configure MariaDB and run makemigrations/migrate

10. Deployment Notes

- Update DATABASES in settings.py with production credentials
- Use environment variables for SECRET_KEY and DB credentials
- Serve static files via nginx in production; use whitenoise for simple setups

11. Conclusion

This skeleton implements the core data model and UI for a modern-styled sticky notes app.
To fully run it, install dependencies and connect to a MariaDB instance as described.